**42028: Deep Learning and Convolutional Neural Network**

**Assignment -3 Project Report**

Keyu Chen, Wanrong Hong, Nianji Yang

of Technology Sydney

**Project Title: Chest X-Ray Images for Pneumonia Classification**

**Team Name:** Go Away COVID19!

**Project Number:** 12

**Supervisor: Nabin Sharma**

**Team Members:**

|  |  |  |
| --- | --- | --- |
| **Student Name** | **Student ID** | **Email ID** |
| **Wanrong Hong** | **13135684** | **wanrong.hong-1@student.uts.edu.au** |
| **Keyu Chen** | **13261021** | **keyu.d.chen@student.uts.edu.au** |
| **Nianji Yang** | **13012531** | **nianji.yang@student.uts.edu.au** |

**Date:** 18/06/2020

**Table of Contents**

[Abstract 3](#_Toc43336408)

[1. Introduction and Background 4](#_Toc43336409)

[1.1 The problem you tried to solve 5](#_Toc43336410)

[1.2 Motivation 5](#_Toc43336411)

[1.3 Application 6](#_Toc43336412)

[1.4 Dataset 6](#_Toc43336413)

[2 Overview of the architecture/system 7](#_Toc43336414)

[2.1 Flow FlowChart 7](#_Toc43336415)

[2.2 CNN Architecture Design 7](#_Toc43336416)

[2.3 GUI Design (Eric) 8](#_Toc43336417)

[3 Results and Evaluation 11](#_Toc43336418)

[3.1 Data Augmentation 12](#_Toc43336419)

[3.2 Initial Experiment 12](#_Toc43336420)

[3.2.1 Initial Experiment architectures and settings 12](#_Toc43336421)

[Baseline Models 12](#_Toc43336422)

[Inception v3: 12](#_Toc43336423)

[AlexNet: 13](#_Toc43336424)

[GoogLeNet: 14](#_Toc43336425)

[3.2.2 Initial Experiment results 16](#_Toc43336426)

[3.2.3 Discussion 16](#_Toc43336427)

[3.3 Fine-tuning the model 17](#_Toc43336428)

[3.3.1 Experiment settings 17](#_Toc43336429)

[3.3.2 Experiment Results 20](#_Toc43336430)

[3.4 Evaluation and Discussion 20](#_Toc43336431)

[3.5 Limitations 21](#_Toc43336432)

[4 Discussion and Conclusions 23](#_Toc43336433)

[5 References 23](#_Toc43336434)

[6 Appendices (Mandatory Section) 25](#_Toc43336435)

# 

# Abstract

Deep learning has played an important part in image recognitions and it has become very popular in revolutionizing disease diagnosis. With the current worldwide outbreak of COVID-19, medical support algorithms become demanding and urgent. A well-developed artificial intelligence tool could effectively improve productivity as it could potentially review enormous amounts of images in less time. As a group of data analytical students, we aim to develop a medical image recognition tool which diagnoses patients with bacterial and viral pneumonia based on the provided X-ray images of lungs utilizing supervised deep learning CNN algorithms. While this approach is still in the stage of developing, it can potentially be utilized in the real world and expedite early diagnosis. It can also become a faster and earlier diagnosis for other diseases by analysing X-ray or other clinical images.

# 1. Introduction and Background

## 1.1 The problem you tried to solve

Radiography (i.e. the use of X-ray) revolutionizes the field of medical diagnosis as it captures the features of internal organs hidden from the naked eye. Over the last century, endless efforts have been devoted to summarising the X-ray features of various diseases while optimizing the X-ray technique to achieve a quick and accurate diagnosis. In current medical practice, the recognition and analysis of an X-ray image still rely on experienced medical professionals and it may take hours to days depending on a wide range of factors. Such heavy reliance on human expertise limits the speed and scale of X-ray image analysis, which has important implications for public health. For instance, in a scenario where large medical imaging data need to be processed promptly, e.g. in the current pandemic of COVID-19 where tens of thousands of chest X-ray images of suspected cases are generated daily, the relative shortage of medical experts will result in a delay in analyzing X-ray images and consequently slow down the diagnosis process, which will result in a large number of infected cases unrecognized and accelerated the spread of the disease. Such delay in diagnosis also affects the primary healthcare system, where the health workers are usually noted trained in reading X-rays and need to resort to specialists. In emergent situations, such waiting time can lead to missed or wrong diagnosis or missed opportunities for swift medical interventions.

The above scenarios signify the demand for a solution to accelerating the recognition and analysis of X-ray images, particularly when large-scale medical imaging data is available. The discrepancy between the usual time taken for analyzing X-ray images and the need for a much quicker analysis is the problem we aim to resolve in this project.

Deep learning has been employed in image recognition and has seen some applications in recognizing medical images in a wide range of conditions (Kondo et al. 1999, Blahuta et al. 2012, Kondo 2003). In this project, we aim to develop an assistive medical image diagnosing tool which can recognize and analyze large-scale X-ray images swiftly through the application of Convolutional Neural Network and other comparative algorithms, such as Recurrent Neural Network. For practical reasons, we focused on the X-rays of only a selective patient cohort – those suspected of bacterial and viral pneumonia and the tool we designed will be able to classify the patients with or without pneumonia. We are not including X-ray images of every condition in this project as it is unrealistic and unnecessary, and this selection is sufficient for reflecting the capacity of our tool.

## 1.2 Motivation

As mentioned in the previous paragraph, the heavy dependence on human expertise in the analysis of X-ray images significantly slows down the diagnosis process. The motivation behind this project is to design a powerful tool in medical image analysis, which will contribute to faster diagnosis and potentially benefit the public health and primary healthcare sector.

In the public health sector, this project aims to provide a tool to assist in quick medical screening and diagnosis in scenarios where large-scale medical image data need to be processed promptly to inform decision making. Such capability will be highly advantageous in a pandemic setting such as the current COVID-19 outbreak where the exponential increase of suspected cases exceeds the diagnosing capacity of the healthcare system. Where a viral test is unavailable, a quick X-ray scan can serve as the initial screening tool to identify suspected pneumonia cases so that isolation procedures can be implemented; meanwhile, an estimate of prevalence can be generated at the initial stage based on a large-scale population X-ray screening, empowering the health professionals and policymakers to come up with swift solutions to curb the spread of the disease.

In the primary health sector, this quick assistive diagnosis tool will also aid health workers with fast triage. While awaiting the confirmative report from specialized radiologists, primary health carers can make a quicker and more accurate diagnosis based on the recommendations of this tool and implement necessary medical interventions as early as possible. The chance of missed diagnosis or treatment delay will be greatly reduced.

## 1.3 Application

The project offers an assistive diagnosing tool based on a fast analysis of X-ray images. It can be adopted in a wide range of medical settings. As stated in 1.2, a primary application will be a public health system where it can be utilized as an efficient screening tool for large populations. Estimated prevalence can be calculated at very early time points. Combining other data, such as demographics, age distribution, and geographical information, scientists can study the disease from multiple aspects. In a global pandemic such as the current COVID-19 outbreak, this tool can be used for generating first-hand data from a large population, thus enabling timely decision making from policymakers.

Further, it can be adopted as a quick assistive diagnosing tool not only for primary health care workers but also for specialists. Other than aiding primary health workers in reading X-rays, this deep learning-empowered image-analysing tool will also be able to classify more complicated and ambiguous X-ray images, based on the training data chosen. The ability to discriminate against pneumonia X-rays is simply an example of such an application.

Besides facilitating diagnosis, the fast-medical X-ray image analysing tool can also be applied in the radiotherapy-orientated treatment, such as planning for surgical navigation (Duncan & Ayache 2000). The fast image recognition and analysis will provide surgeons with real-time surgical route solutions based on the anatomy reflected in the X-ray images.

## 1.4 Dataset

The dataset is extracted from <https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia> Overall, it contains greyscale X-ray images of two categories: normal chest; and chest with bacterial or viral pneumonia. A total of 5,863 images are included. The chest X-rays are of a standard anterior-posterior view and were originally sourced from paediatric patients of one to five years old. All chest X-rays were taken as part of patients’ routine clinical care.

For quality control purposes, all the images were screened initially to remove any low-quality or unreadable scans. The classification of the images was performed by two experienced physicians and double-checked by a third expert to account for possible grading errors. The full dataset was published in Jan 2018 along with the task. The three split sets of datasets are as follows: The training set contains 5,216 sample images, of which 1341 images of normal chest and 3875 images of bacterial or viral pneumonia are included. The validation set includes 16 sample images divided into 2 categories of 8 images each. The test set contains 624 images, with 390 images showing pneumonia and 234 normal chest X-rays.

# 2 Overview of the architecture/system

The system is mainly constructed in 3 parts: the graphical user interface, Deep Learning CNN model for training and the presentation of the classification result. The following section will describe the flow of the system, architecture of the chosen CNN model and the demo of the GUI design.

## 2.1 System Flow Chart

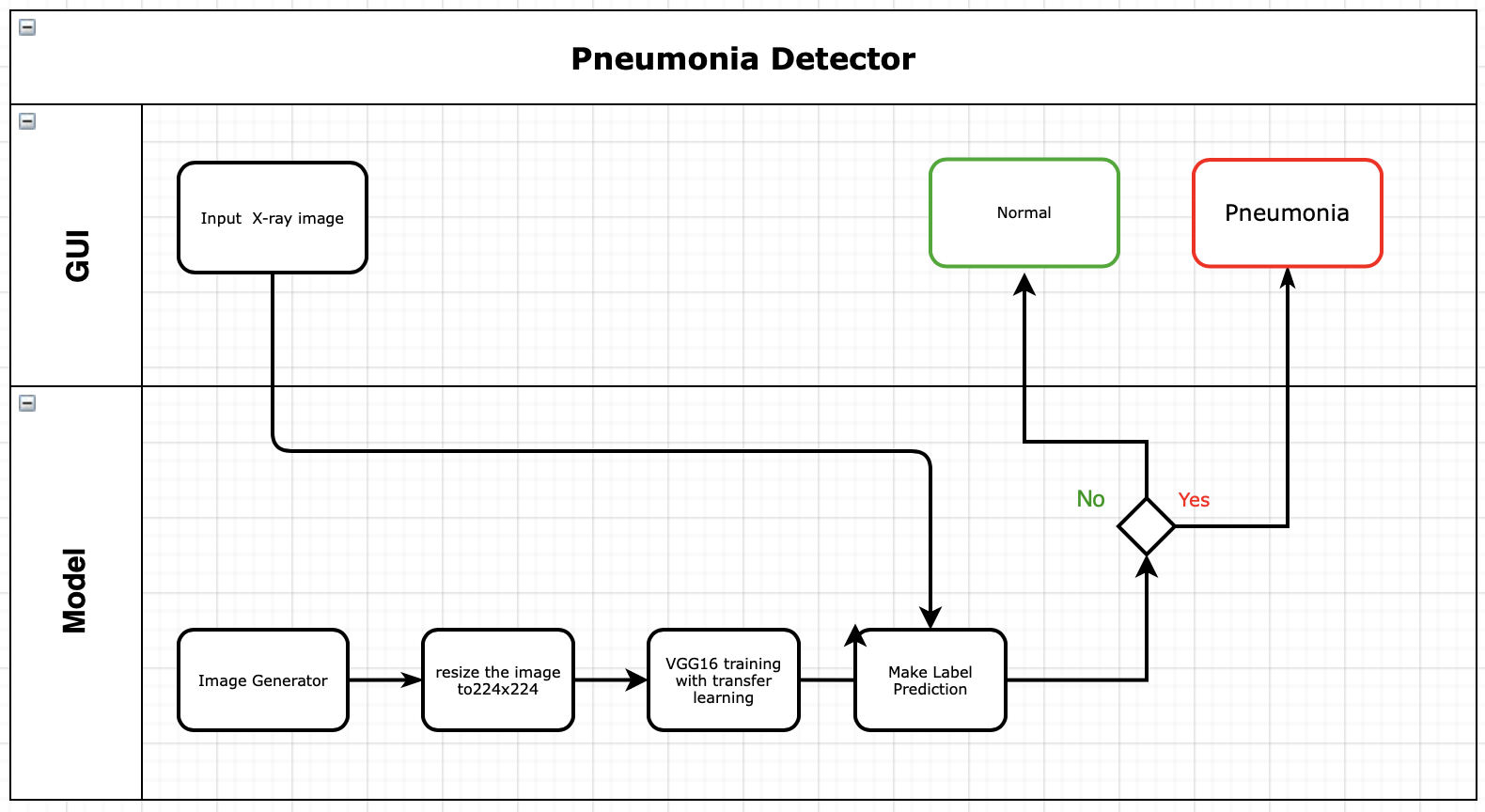


Fig 1. Workflow of Pneumonia Detector

As Figure shows, an input image, which can be any size, will be resized into 224x224. Then, the resized image will further be classified to health or pneumonia, and the GUI will present the result.

## 2.2 CNN Architecture Design

The design of the CNN architecture is the core element of the system as it highly affects the accuracy of the output, hence the model is carefully decided based on features of the architecture and the dataset.

The final model VGG16 is chosen from a variety of Deep Learning Convolutional Neural Networks via comparing model performance and complexity. VGG 16 is one of the winning architectures in ILSVR competition in 2014, it contains 16 layers including convolutional layers with filter size of 3x3 and stride of 1, and MaxPooling layers with filter size of 2x2 and stride of 2 (Thakur 2019). Dropouts are also added in the fully connected layer. VGG16 is one of the most excellent CNNs that produces satisfactory results although it is computational costly.

### 

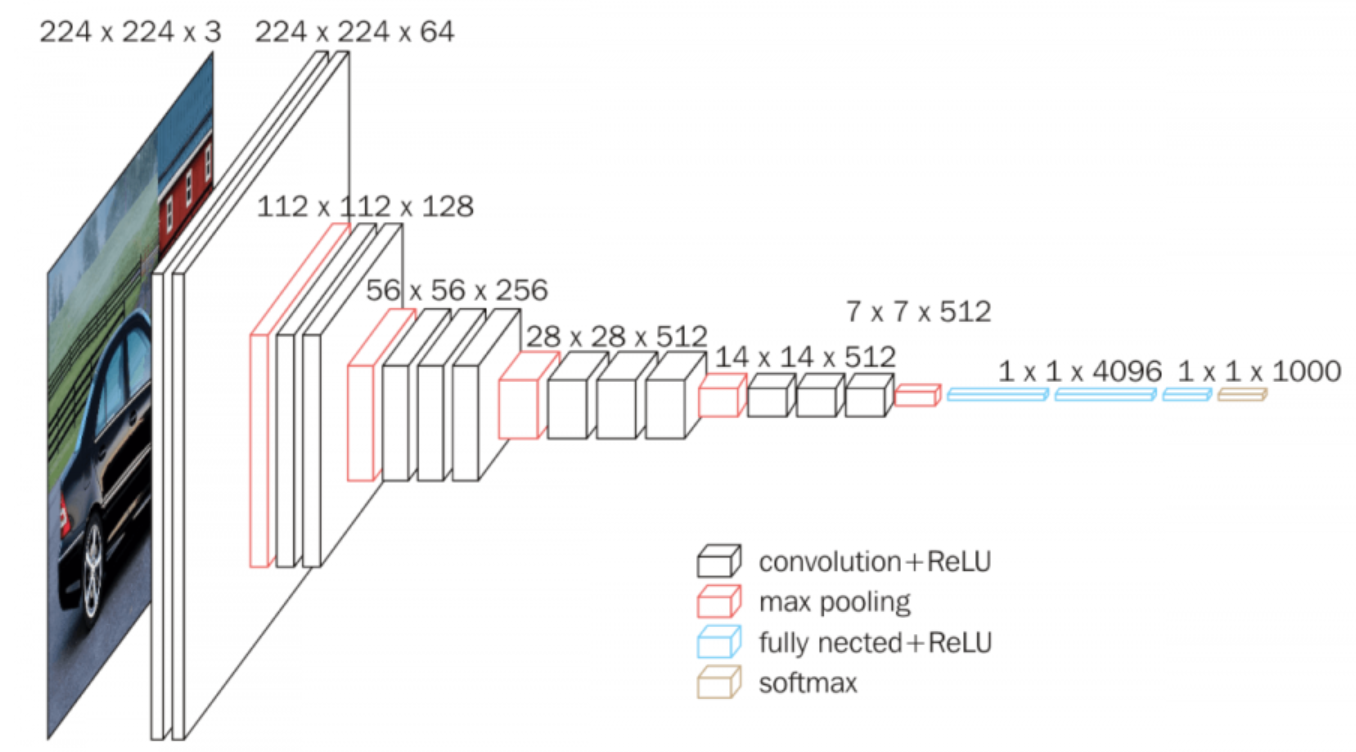
****

Fig 2. Architecture of VGG16

We applied VGG16 from Keras packages using transfer learning based on ImageNet, the base model has been pre-trained to extract features. We have experimented with different numbers of frozen convolutional layers to fine-tune the model. it is discovered that the model complexity becomes higher as more convolutional layers are trainable. Details on layer adjustment and parameter settings will be further explained in the following section

## 2.3 GUI Design

Whenever users open the application, the window will be shown in the middle of the monitor screen. The size of the window is fixed and cannot be changed. There are three menus in the application window, shown as below. For example, when users click on the “File” menu, it will show few options. If users click on the Exit, the application will be closed automatically.

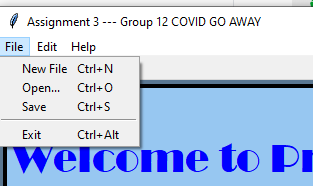


Fig 3. GUI menu

Our GUI is composed of two pages/tabs. The first one is an “Information” page, which is used to extract information from users. The second one is “Detection” page, which is used for X-ray image uploading and diagnosis. Both of the two pages are shown below.

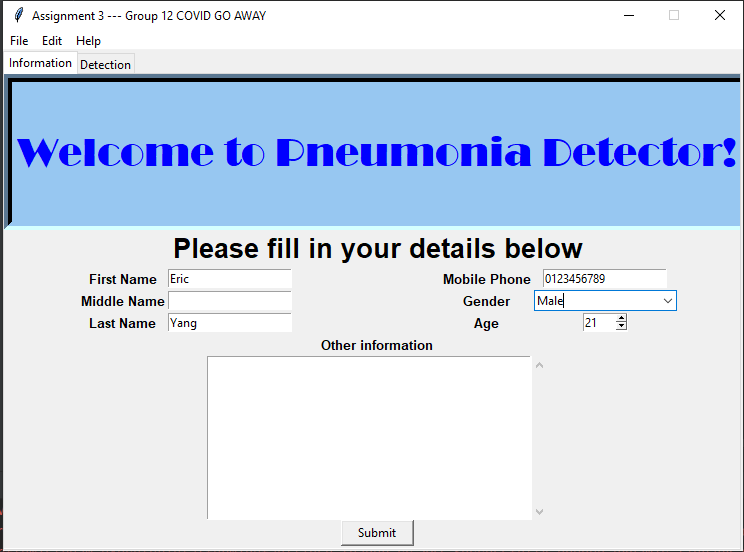


Fig 4. GUI first page

As mentioned above, the first page is a Welcome page. Users can fill in their information in a textbox next to each label. Among them, there are a combobox for gender selection and a spinbox for age. Also, there is a scrolledtext window at bottom if other information is needed to add for users. The “submit” button is placed at the end after filling all information. When users click the button, it will turn to blue and suggest users to go to the next page, shown as below.



Fig 4. Button changed after click

The second page is for users to upload X-Ray images and get prediction/diagnosis. Before they do anything, the page shows like below. There are two buttons, “Open Image” and “Process Image”.

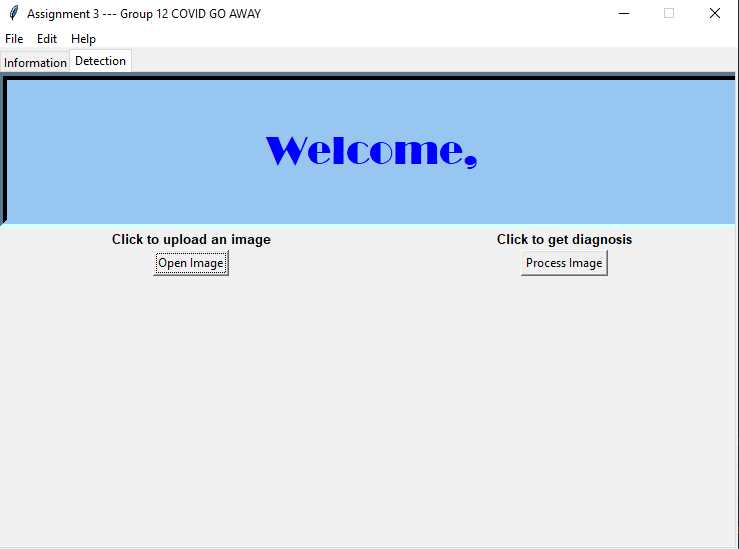


Fig 5. GUI second page before image classification

If users fill in their First Name on the first page, the name will be automatically shown on the top. After users click the “Open Image” button and upload proper images, the button will be changed, and the uploaded image will be shown below. When users select an image with jpeg format, a message window will pop up, showing the file’s information, shown as below.

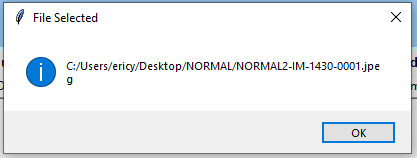


Fig 6. Messagebox with information of selected image

If users select an image which is not jpeg, it will show an error message, shown as below.

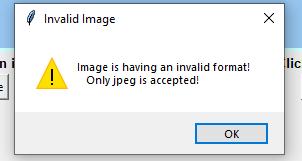


Fig 7. Error message

Then, if users click the “Process Image” button, they will get the results. If the X-ray image shows no pneumonia, it will show “You are healthy! Please keep it!”, shown as below. If the prediction is positive, it will show “The diagnosis result is Pneumonia. Please contact your GP asap!”. A convenient function for this application is that users can upload multiple images repeatedly and get multiple results.



Fig 8. GUI second page after image classification

# 3 Results and Evaluation

## 3.1 Data Augmentation

Data augmentation is utilized to artificially increase the size of the train set by generating other versions of images in the original dataset. As shown below, the original images are rotated, shifted, zoomed and flipped to expand the dataset.



Fig 9. Sample augmented images

To comply with the model evaluation, the batch size of the train generator is set to 20 while the batch sizes of the test and validation generator are set to 10 and 1 accordingly. Class mode of ‘binary’ is implemented since there are only 2 classes in the dataset.

## 3.2 Initial Experiment

4 baseline models have been selected to obtain intuitive understandings of the overall performance of the targeted image classification using Deep Learning Convolutional Neural Networks. The initial settings architecture designs of the baseline models are demonstrated below.

Since our dataset only contains 2 classes (ie. normal and pneumonia), all of the experimented models have compiled with binary cross entropy which returns the probability of the detection being 1 (ie. pneumonia).

### 3.2.1 Initial Experiment architectures and settings

## Baseline Models

### Inception v3:

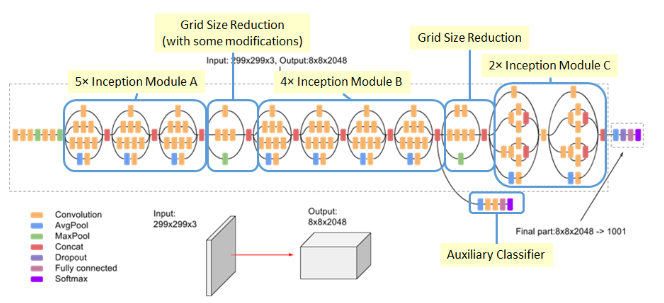


Fig 10. Architecture of the Inception v3



Fig 11. Settings of the Inception v3

Inception v3 is another convolutional architecture that contains 1000 classes that were pre-trained on ImageNet originally (Krizhesky et al, 2012). Inception v3 network stacks 11 inception modules with convolutional filters, pooling layers and linear activation functions in each module (Szegedy et al, 2015). The architecture is implemented with the input size of 224x224x3, the Inception v3 contains 21,804,833 parameters in total, 2,049 are trainable and 21,802,784 are non-trainable. We trained the model with frozen convolution base layers as our baseline. We have added a global average pooling layer and a dropout layer for regularization. At the end, a sigmoid layer is applied for predicting binary class.

### AlexNet:

The initial AlexNet consists of 5 convolutional layers, 3 max pooling layers and 3 fully connected layers (Krizhevsky et al, 2012). Relu activation function is used after every convolutional and fully connected layer. 8 Batch Norm layers are added to normalize the output of the previous activation layer through subtracting the batch mean and dividing them by the standard deviation. Dropout is used in fully connected layers to reduce the size of the network and avoid overfitting.

****

Fig 12. Architecture of standard AlexNet

The architecture is implemented with the input size of 224x224x3, the standard AlexNet contains 28,083,756 parameters in total, 28,062,620 are trainable and 21,136 are non-trainable.

Optimizer: RMSProp

Learning rate: 1e-4

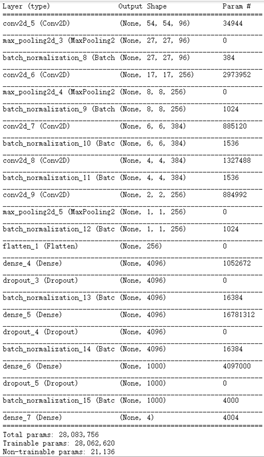
****

Fig 13. model summary of AlexNet

### GoogLeNet:

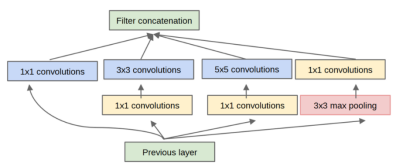
****

Fig 14. Architecture of the inception V1 module

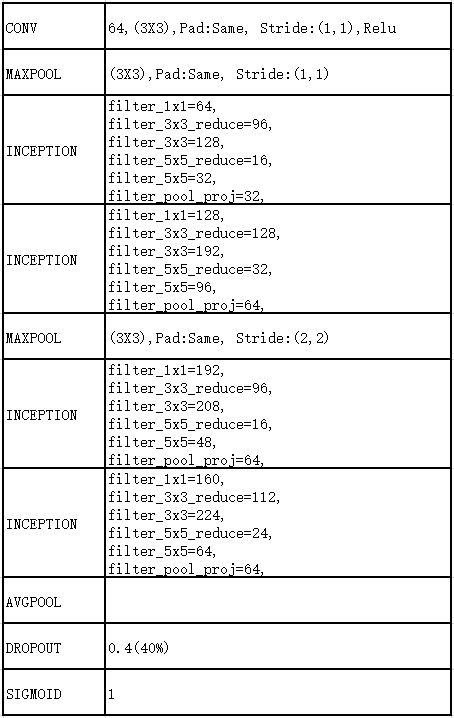
****

Fig 15. Settings of the Micro-GoogLeNet

Micro-GoogLeNet architecture is presented above. Figure 5 shows the inception V1 module. As shown in Figure 6 The whole architecture firstly applies a convolution layer for feature extraction, using Relu as its activation, a maxpool layer, 2 inception V1 module and another maxpool layer, 2 inception V1 module followed by an average pool and a dropout layer. Last but not least, a sigmoid layer is applied for predicting class.

Optimizer: SGD

Learning rate: decay learning rate.

**VGG16:**

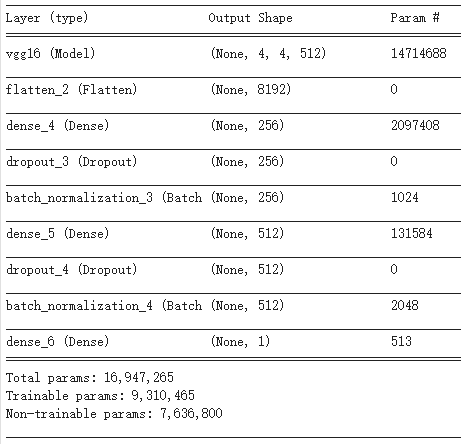
****

Fig 16. Summary of VGG16

We applied VGG16 from keras packages using transfer learning based on ImageNet. We partially trained the last 3 CONV layers and full connected layers since our dataset distribution is not similar to ImageNet, the number of classes are different and our dataset is not very large. The parameter settings are presented by Figure 4. Particularly, dropout\_3 (Dropout) is 0.5(50%) and dropout\_4 (Dropout) is 0.5(50%).

#### 

### 3.2.2 Initial Experiment results

We have fit the dataset into 4 baseline models in the initial experiment and obtained results of training, validation and test.

|  |  |  |  |
| --- | --- | --- | --- |
| **CNN Architecture** | **Train Accuracy** | **Validation Accuracy** | **Test Accuracy** |
| Inception v3 | **74.3%** | **50%** | **61.5%** |
| AlexNet | **94.8%** | **56.3%** | **83.4%** |
| GoogLeNet | **93.2%** | **64.0%** | **87.6%** |
| VGG16 | **96.3%** | **68.7%** | **89.7%** |

### 

### 3.2.3 Discussion

As demonstrated in the table above, Micro GoogLeNet and VGG16 have outperformed the other two algorithms in terms of accuracy. While VGG16 has the highest accuracy in both training and testing, it is much more complex than Micro GoogLeNet as it contains more parameters and requires longer training time. Further improvements and comparisons are required to select the most suitable model as our final CNN to be embedded in the system.

## 3.3 Fine-tuning the model

The following section will explain the fine-tuning process for both VGG16 and Micro GoogLeNet (Inception v1) and the final result derived from the experiment. The fine-tuning process mainly involves increasing/decreasing the number of trainable convolutional layers in transfer learning and adjusting hyperparameters on the model with better performance.

### 3.3.1 Experiment settings

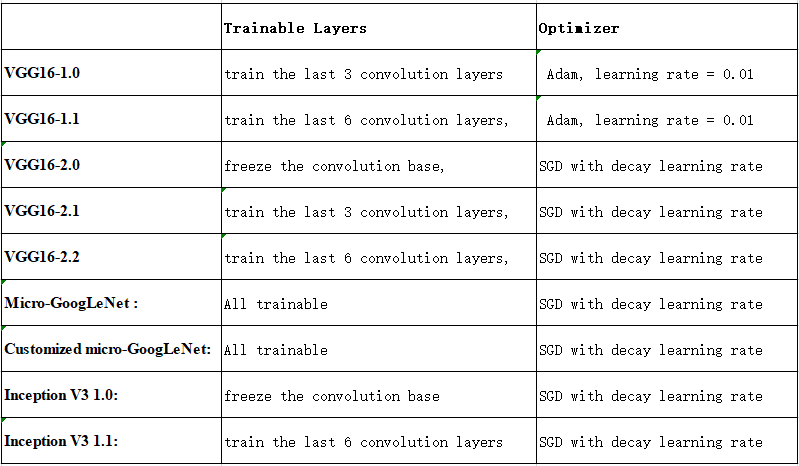


Fig 17. table of experiment settings on conducted models of VGG16

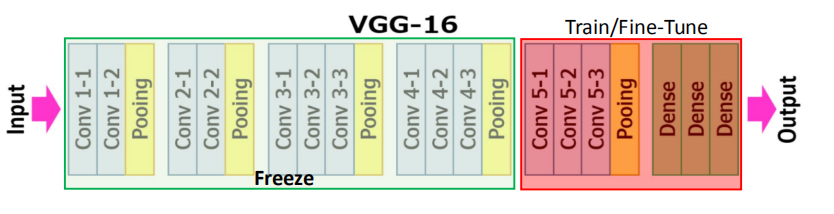
VGG16-2.0: all the convolutional layers are frozen, features are absolutely extracted from the pre-trained model. Optimizer of SGD is applied with learning rate decay.

****

Fig 18. VGG16-2.0

VGG16-1.0: the first 10 convolutional layers along with 4 MaxPooling layers are frozen and the last 3 convolutional layers, the MaxPooling layer and the dense layers are trainable. Optimizer of Adam with default learning rate 0.01 is implemented.

VGG16-2.1: similar to VGG16-1.0, only the last 3 convolutional layers, MaxPooling layer and 3 dense layers are trainable. The optimizer of SGD with learning decay that starts from an initial learning rate of 0.01 is applied.

****Fig 19. VGG16-1.0 and VGG16-2.1

VGG16-1.1: the first 7 convolutional layers along with 3 MaxPooling layers are frozen, the last 6 convolutional layers are trained upon the pre-trained weights, together with 2 MaxPooling layers and 3 dense layers. The model is compiled with Adam with a learning rate of 0.01.

VGG16-2.2: similar to VGG16-1.1, the first 10 layers are frozen while the last 11 layers are trained upon the pre-trained weights obtained from ImageNet. The optimizer of SGD with learning rate decay starting from initial learning rate of 0.01 is implemented.

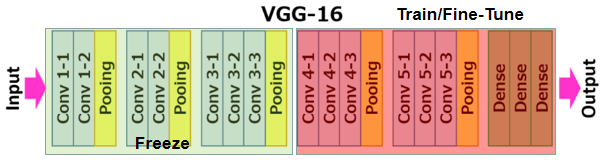
****

Fig 20. VGG16-1.1 and VGG16-2.2

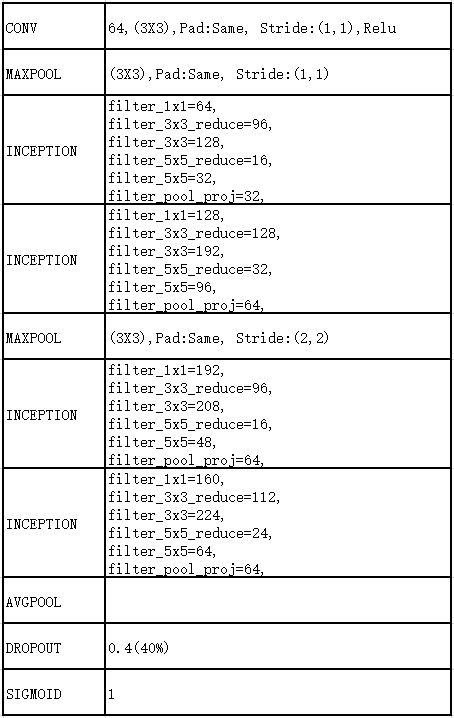
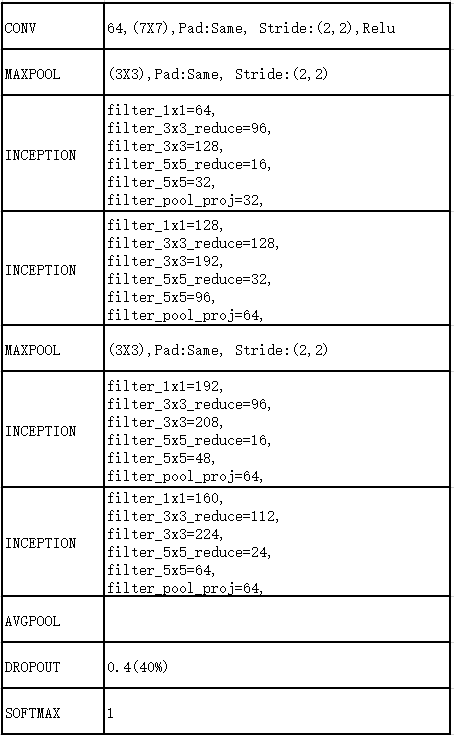
****

Fig 21. baseline Micro-GoogLeNet vs customized Micro-GoogLeNet

Baseline Micro-GoogLeNet and Customized micro-GoogLeNet: All the layers are trainable. The settings are presented in the above figure.

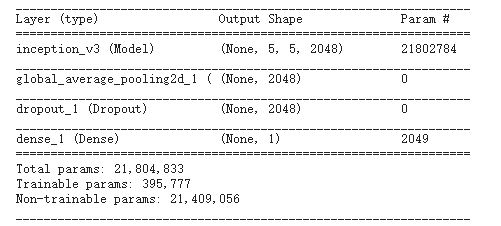


Fig 22. Settings of Inception v3 1.0(left) and Inception v3 1.1(right)

Inception V3 1.0: all the convolutional layers are frozen, features are absolutely extracted from the pre-trained model. Optimizer of SGD is applied with learning rate decay.

Inception V3 1.1:

Only the last convolutional layer, the MaxPooling layer and the dense layer are trainable. Optimizer of SGD is applied with learning rate decay. parameters settings are presented above.

### 3.3.2 Experiment Results

As demonstrated in the table below, VGG16-1.0 and VGG16-2.1 have achieved higher test accuracies than the other versions of VGG16 have. While VGG16-2.1 has obtained a bit lower test accuracy than VGG16-1.0, it has lower loss in train, validation and test and higher validation accuracy. Hence, it is determined that VGG16-2.1 has outperformed other versions of VGG16 and concluded that the dataset works the best with 7 trainable layers, the use of Learning Rate Decay also optimizes the model performance.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **CNN Architecture** | **Train Accuracy** | **Train Loss** | **Validation Accuracy** | **Validation Loss** | **Test Accuracy** | **Test Loss** |
| VGG16-1.0 | **96.3%** | **0.962** | **68.7%** | **0.121** | **89.7%** | **0.397** |
| VGG16-1.1 | **96.7%** | **0.094** | **56.3%** | **0.000629** | **87.7%** | **0.164** |
| VGG16-2.0 | **85.3%** | **0.333** | **50%** | **2.373** | **37.5%** | **1.079** |
| VGG16-2.1 | **92.7%** | **0.214** | **75%** | **0.042** | **88.5%** | **0.204** |
| VGG16-2.2 | **92.8%** | **0.188** | **62.5%** | **3.49** | **80.2%** | **0.006** |

Fig 23. table of experiment results on conducted VGG16

As demonstrated in the table below, Micro-GoogLeNet has reached the highest training and testing accuracy among all inception models, followed by customized-GoogLeNet. the performance of Inception v3 1.0 and Inception v3 1.1 are similar, which are much lower than Micro-GoogLeNet and customized Micro-GoogLeNet.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CNN Architecture** | **Train Accuracy** | **Train Loss** | **Validation Accuracy** | **Validation Loss** | **Test Accuracy** |
| Micro-  GoogLeNet | **93.2%** | **0.169** | **64%** | **0.149** | **87.5%** |
| Customized-  GoogLeNet | **92.5%** | **0.178** | **54.7%** | **1.619** | **77.6%** |
| inception V3 1.0 | **74.3%** | **3.94** | **50%** | **15.2** | **61.5%** |
| inception V3 1.1 | **74.2%** | **3.95** | **50%** | **15.2** | **62.2%** |

Fig 24. table of experiment results on conducted Inceptions

We have come to the conclusion that VGG16 has a better overall performance than Inception by comparing a set of fine-tuned VGG16 and Inception. VGG16-2.1 is selected as the final CNN architecture to be embedded in our designed system.

## 3.4 Evaluation and Discussion

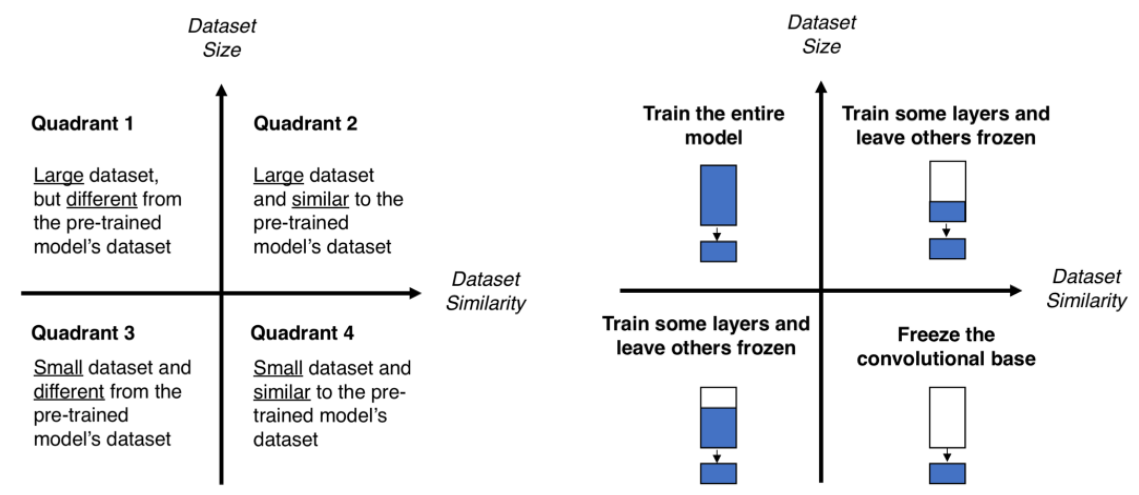


Fig 25. Evaluation of transfer learning process

The experiment results have clearly demonstrated the difference in the performance in respect to different transfer learning processes. Since the trained images in ImageNet are inclusive to the provided dataset but different, it is not a good idea to freeze all the convolutional layers as the features of images in the training set are not fully extracted by relying on a fixed feature extraction mechanism (Marcelino, 2018). Ideally, the size and similarity of the dataset to the dataset trained in ImageNet fit the conditions in Quadrant 3 as presented below, although both models with 7 frozen layers have slightly worse performance than the models with 3 frozen convolutional layers. This can be explained by the settings of hyperparameters as lower layers extract general features while higher layers extract specific features (Marcelino, 2018). Since our dataset is relatively small, more layers are required to be frozen to avoid overfitting.

Inception v3 performances are not as well as Micro-GoogLeNet and customized Micro-GoogLeNet, since we have almost frozen the whole convolution base layers. Specifically, if we train more convolution layers on the Inception v3 model, the performance should have been better. However, compared to Inception v3 1.0, Inception v3 1.1 has not improved, thus we did not apply more experiments in Inception v3.

On the other hand, since the validation set is too small in our experiments, we have implemented the validation set argumentation. However, the performance did not improve.

# 3.5 Limitations

As per wrong prediction samples demonstrate, the model is capable of detecting bacteria lungs based on the clarity of X-ray images. The X-ray images of normal lungs that illustrate darker and clearer background in the lungs region can be accurately identified by the model while the lung images that present blurry lines in the lungs region are more likely to be classified as pneumonia.

While the Deep Learning CNN has obtained 88.5% test accuracy, there’s still a risk of misclassifying based on the nature of the dataset if we only rely on computer vision. Since the initial hyperparameters can largely impact the final results, it is recommended to have someone who is from a medical background to analyze the features of the original images so that the initial parameters can be adjusted.

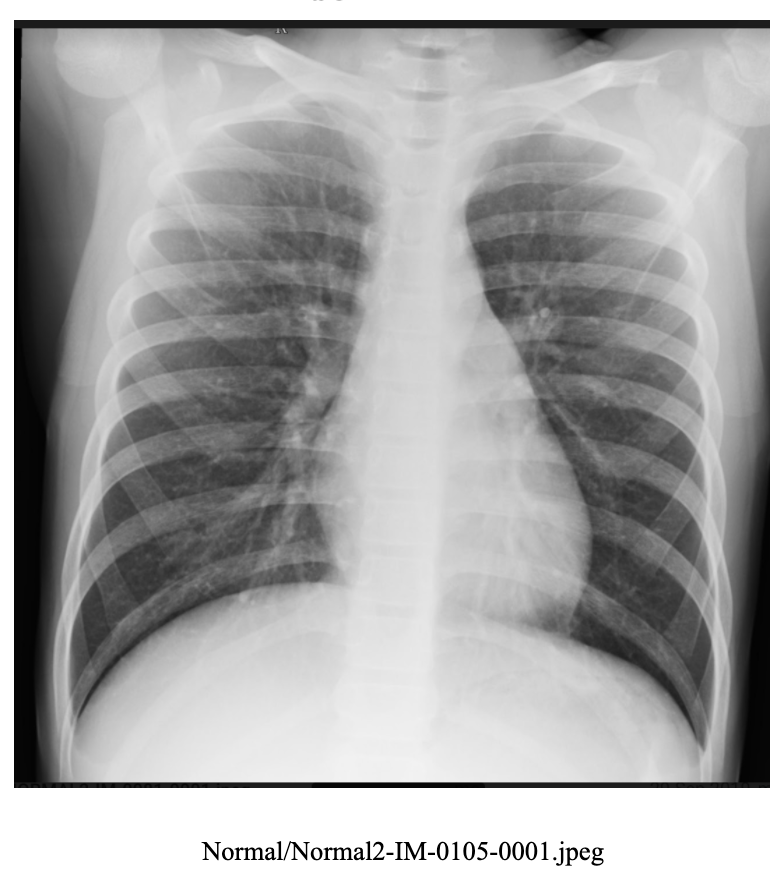


Fig 26. Wrong predictions (predicted: pneumonia, actual: normal)

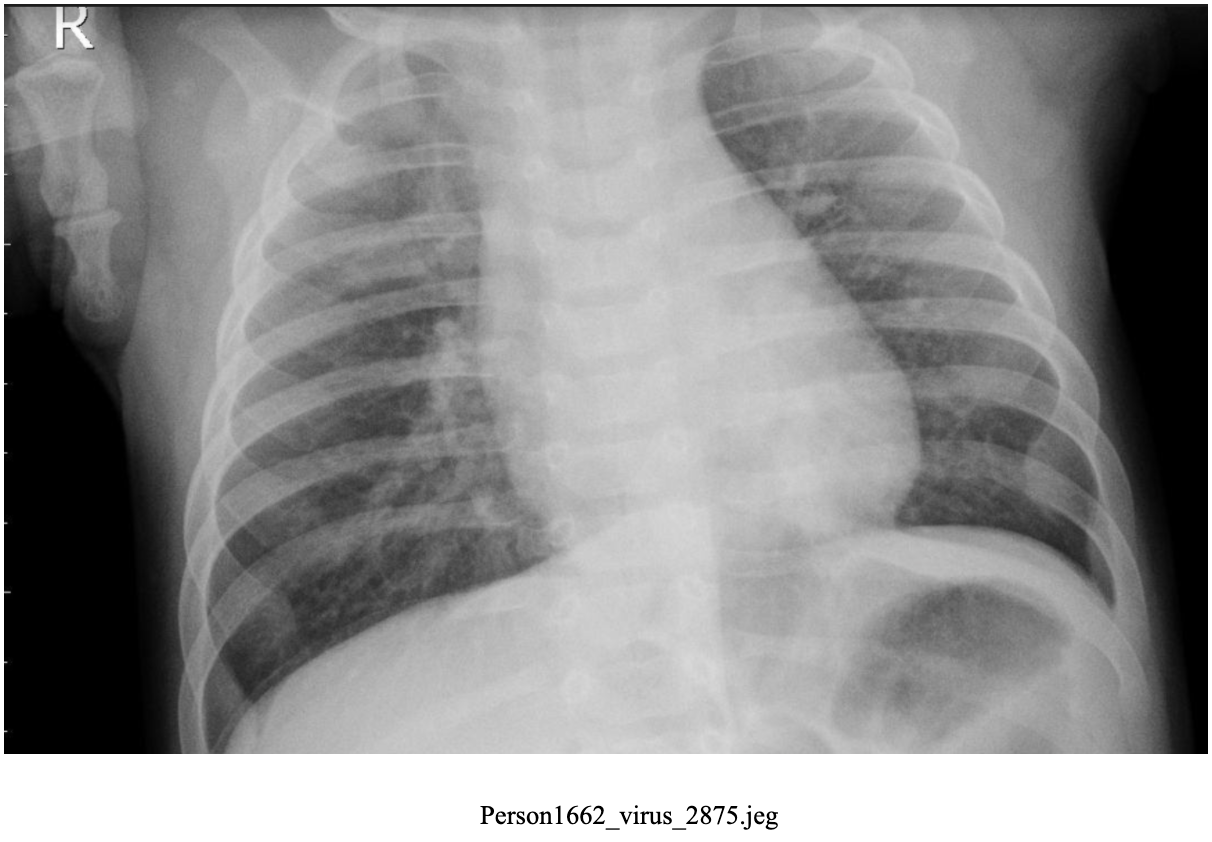
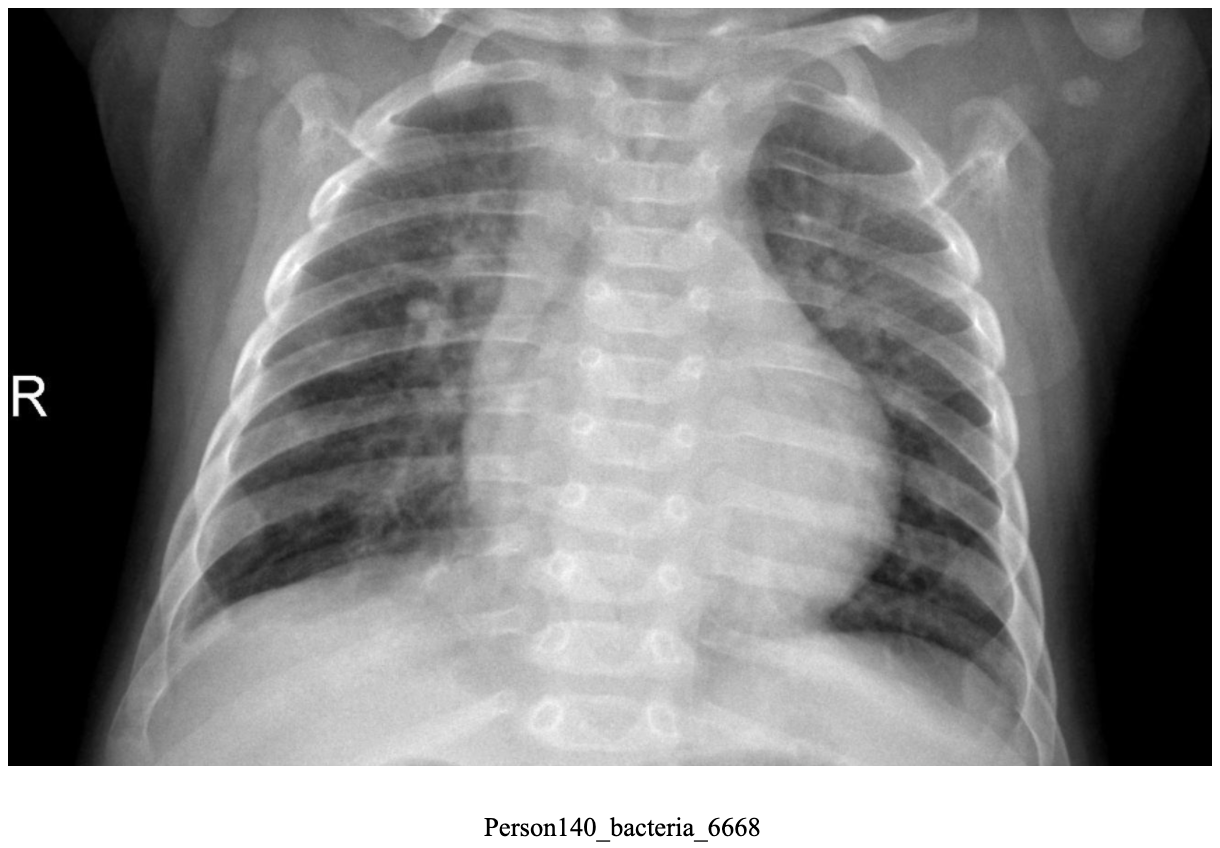


Fig 27. Wrong predictions (predicted: normal, actual: pneumonia)

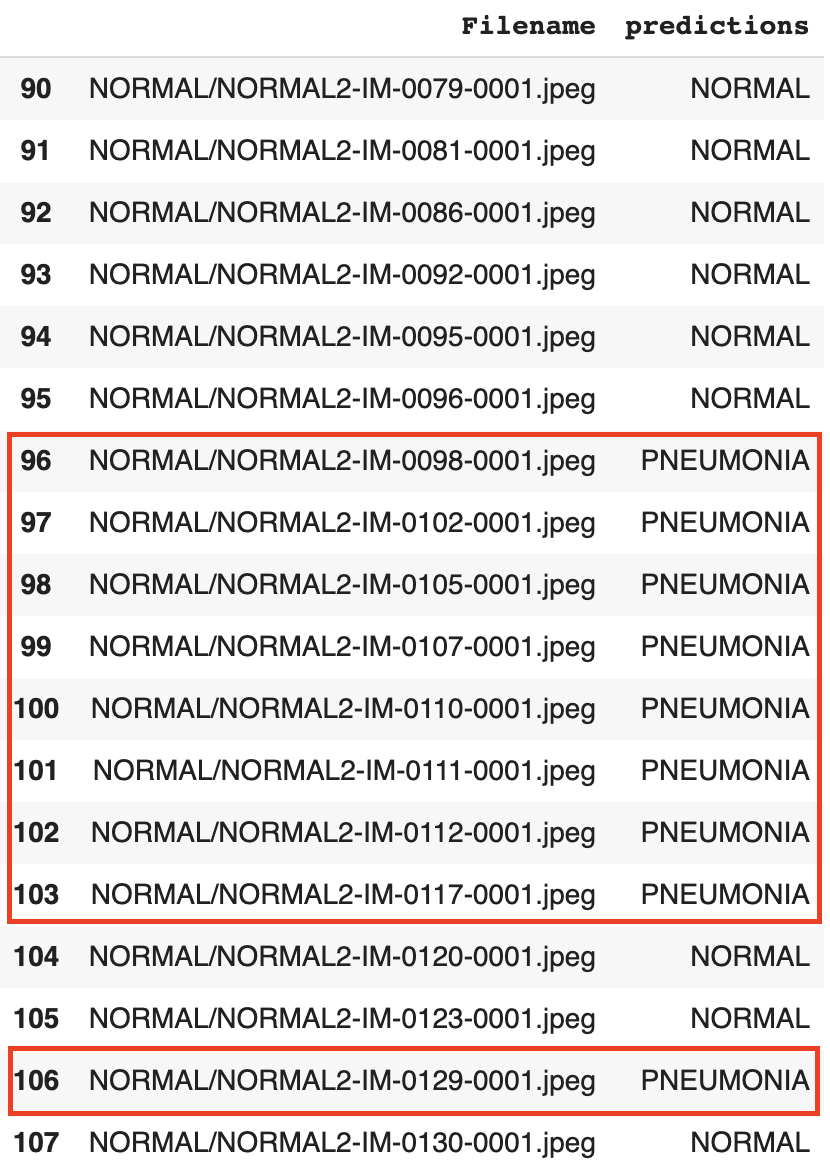


Fig 28. file list of predictions vs actual

Another limitation that draws our attention is that the size of the dataset is too small to justify the accountability of the system. While the training dataset accounts for 89% of the total data set and testing dataset accounts for around 10%, only 0.27% of the total dataset is split into the validation set. For a classification task, the validation set, which only takes up 0.27% of the whole dataset, is so small that the accuracy could have a large variance. In our experiments, argumentation, which causes higher computation cost, is applied to enrich our dataset. However, the argumentation of the validation set did not bring a better performance of classification.

The class of input images can be detected by our model with high accuracy, nonetheless, the position and the orientation of the virus/bacteria are not encoded to the predictions.

# 4 Discussion and Conclusions

In conclusion, our approach is to recognize and classify chest x-ray images utilizing the Deep Learning Convolutional Neural Network of VGG16 with transfer learning. The model is chosen from a range of CNN models, the most suitable is selected via measuring the train, validation, and test accuracy and loss. A simple GUI is designed for users to easily access the system and provide test input images. Although VGG16 has shown outstanding classification results in the experiment, the complexity and computational cost is relatively high. We need to understand the foundation of the deep learning models as well as the features of the dataset since it requires related medical knowledge.

At this stage, the system is only applicable in detecting pneumonia based on x-ray images with high resolutions, further improvements can be performed to identify pneumonia with detecting the localization of the infected area with precision recalls. The scope of the project can also be extended to incorporate different medical images such as MRI scans, CT scans, etc.

# 5 References

[1] Blahuta, J., Soukup, T., Čermák, P., Rozsypal, J., & Večerek, M. (2012, May). Ultrasound medical image recognition with artificial intelligence for Parkinson's disease classification. In *2012 Proceedings of the 35th International Convention MIPRO* (pp. 958-962). IEEE.

[2] *C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, ‘Rethinking the Inception Architecture for Computer Vision’, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 2818-2826, doi: 10.1109/CVPR.2016.308.*

[3] Duncan, J. S., & Ayache, N. (2000). Medical image analysis: Progress over two decades and the challenges ahead. *IEEE transactions on pattern analysis and machine intelligence*, *22*(1), 85-106.

[4] Kondo, T., Pandya, A. S., & Zurada, J. M. (1999, July). GMDH-type neural networks and their application to the medical image recognition of the lungs. In *SICE'99. Proceedings of the 38th SICE Annual Conference. International Session Papers (IEEE Cat. No. 99TH8456)* (pp. 1181-1186). IEEE.

[5] Kondo, T. (2003). Revised GMHD-type neural networks with radial basis functions and their application to medical image recognition of stomach. *Systems Analysis Modelling Simulation*, *43*(10), 1363-1376.

[6] Krizhevsky. A, Sutskever. I & Hinton, G, ‘ImageNet Classification with Deep Convolutional Neural Networks’, *Neural Information Processing Systems*, vol. 25, 2012, 10.1145/3065386

[7] Marcelino, P. 2018, *Transfer learning from pre-trained models*, TowardsDataScience, viewed 15 June 2020,

<<https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751>**>**

[8] Thakur, R. 2019, *Step by step VGG16 implementation in Keras for beginners*, TowardsDataScience, viewed 15 June 2020,

<<https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c>>

# 6 Appendices (Mandatory Section)

# A. Individual Contribution

Each group member should write half-page (max) about his or her contribution in the project development activities.

Keyu – uploading the dataset in the shared environment, pre-processing the data, experimenting different baseline CNN architectures, fine-tuning VGG16, monitoring the performance of the model and saving checkpoints for Nianji to connect the CNN training results to GUI, researching, presenting test results, finalizing the report by organizing formats and refereeing.

Nianji – collecting the dataset, demonstrating weekly update of the project progress with supervisors, introducing the background, motivation and defining the project objectives, developing the GUI design and presenting the demo and produce the final report and presentation.

Wanrong – defining the workflow of the project, training and testing different baseline models and fine-tuning VGG16 with Keyu, producing the experiment settings and results tables, brainstorming new ideas for improvement, debugging the existing error and producing the final report and presentation.

All team members are highly devoted and active throughout the entire project development, findings and observations are frequently discussed in weekly meetings with the project update. Each team member has contributed equally to the completion of the project.

# B. Individual Contribution Split

|  |  |
| --- | --- |
| **Team Member Name** | **Percentage** |
| **[Keyu Chen]** | **33.33%** |
| **[Wanrong Hong]** | **33.33%** |
| **[Nianji Yang]** | **33.33%** |

All the team members have discussed and agreed on the above individual contribution to the project.

**[Keyu Chen]:**

**[Wanrong Hong]:**

**[Nianji Yang]:**