# Classifying COVID-19 in Chest X-Ray Image using Lightweight Pre-trained Convolutional Neural Networks.

Jaturong Kongmanee

CSE802 Term Project, Spring 2021

## 1 Introduction

Coronaviruses are a group of RNA viruses that cause respiratory tract infections and range from common cold varieties to severe pandemics. As of March 11, 2020, the World Health Organization officially declared the COVID-19 as a pandemic [4]. Since then, the COVID-19 virus has rapidly spread into multiple countries and has become a serious global health problem [18]. This virus directly infects the lungs, which potentially causes deadly respiratory syndromes [17]. An early diagnosis is a crucial element to prevent the spread of the disease and provide the proper treatment for infected patients.

One of the popular testing methods involves human analysis of computed tomography (CT) scans since certain abnormalities in the chest region of the medical imaging have been shown to be unique to COVID-19 [1]. However, expert practitioners and high-cost technology are required, which results in in short supply in less developed countries. Moreover, overloaded hospitals cannot satisfy this form of COVID-19 testing but demand easier, cheaper, and accurate screening methods.

Recent studies have shown that deep learning could be used to automate COVID-19 diagnosis and improve diagnosis efficiency and accuracy using x-ray images [15] [13]. Impressively, most of the studies working on COVID-19 x-ray image classification report high performances in this task. However, the experiments were tested on a particular dataset of x-ray images, which could potentially limit the generalization of image classifiers. This is due to the fact that the classifiers might possibly learn features that are specific to a particular dataset instead of the disease itself.

In this project, we develop and compare the pre-trained deep neural networks ImageNet based classifiers for the binary image classification task. The selected image classifier is fine-tuned with different hyperparameter configurations to improve the performance [9]. To improve the generalization of the proposed image classifiers that possibly learn specific features of a particular dataset instead of the disease itself, we train classifiers on various chest x-ray image datasets, which are collected and combined into a single dataset. However, the size of a collected dataset is small and the samples that belong to the COVID class are limited, so we incorporate data augmentation and leverage the pre-trained Convolutional Neural Networks (CNNs). Moreover, we seek the lightweight neural network architectures for a good balance between accuracy, speed, and size, which will benefit Artificial Intelligence (AI) applications used in real-world production environments.

## 2 Proposed Work

In this section, the proposed work is described step-by-step respectively. The details and graphical figures for each step are provided. Overall, we develop four binary image classifiers for the COVID-19 chest x-ray image classification task. These four deep neural network-based binary classifiers are built based on the pre-trained weights from the ImageNet dataset. We then compare these models using the Receiver Operating Characteristics (ROC) curve and the Area under the ROC Curve(AUC) metrics to select only the best model for fine-tuning the model's accuracy by varying the percentages of unfreezing layers of the pre-trained model. Moreover, we develop a highly interactive web application to allow AI researchers and practitioners to upload their classifier models for public use. In this way, people involved in COVID-19 treatment (e.g., medical doctors, nurses, expert practitioners, etc.) can leverage the deployed models for COVID-19 diagnosis, which will help improve the screening step.

## 2.1 Image Preprocessing

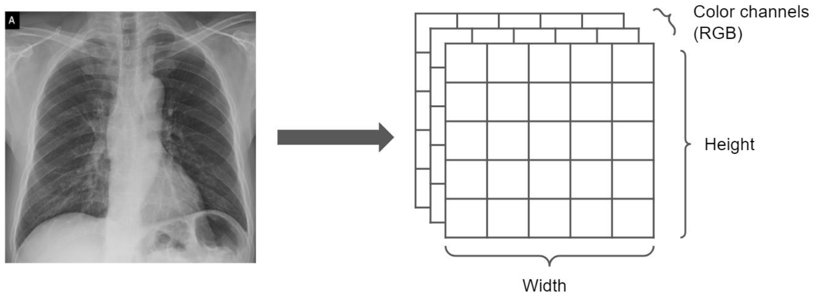


Figure 1: A 3D image data tensor

Before inputting image data into the image classifier, we read and decode these JPEG images to RGB grids of pixels and convert them into floating-point tensors. The tensors of pixel values are resized into 224x224 pixels. Afterward, these pixel values (between 0 and 255) stored in tensors are rescaled to the [0, 1] interval. Specifically, the input image was essentially represented as three-dimensional tensors as shown in Figure 1. These dimensions represent the image width, height, and number of channels (R-red, G-green, and B-blue). Each value in the tensors represents the intensity of each individual pixel. After the image preprocessing step, the three-dimensional tensors are flattened as a $d$-dimensional vector ($d$ is 224x224 pixels) and fed into the neural network classifier. Here, other techniques such as segmentation to separate lung regions of interest (ROI) from background and thresholds to remove overlaying content (e.g., labels) can be applied to improve the classifier's performance.

## 2.2 Data Augmentation

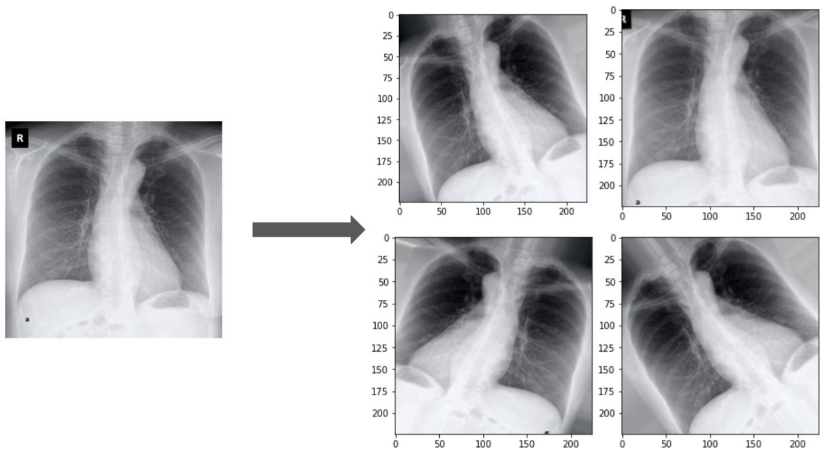


Figure 2: Generation of an chest x-ray image via random data augmentation

Since the chest x-ray image dataset is collected from various sources and each class has a limited amount (will be described in detail in the dataset section), this may cause the overfitting problem and results in the classifier that cannot generalize well to new unseen data. To mitigate the overfitting problem, we incorporate the data augmentation technique on the training set to make the proposed models exposed to more aspects of data distribution. Specifically, data augmentation generates more training samples by augmenting the samples via a number of random transformations. In this way, during the training time, the models will never see the same picture due to the randomness of image transformations. The random transformations applied in this project are (i) randomly rotate an image in the range between 0-40 degrees; (ii) randomly zoom inside an image; and (iii) randomly flip an image horizontally. The resulting augmented images are shown in Figure 2.

## 2.3 Using Pre-trained Convolutional Neural Networks

Table 1: Architectural details of selected pre-trained ImageNet models.

| Model | Top-1 Accuracy | Top-5 Accuracy | Size (MB) | Trainable Parameters |
|---|---|---|---|---|
| MobileNetV2 | ***0.713*** | ***0.901*** | ***14*** | ***3,538,984*** |
| InceptionV3 | 0.779 | 0.937 | 92 | 23,851,784 |
| ResNet50V2 | 0.760 | 0.930 | ***98*** | ***25,613,800*** |
| Xception | ***0.790*** | ***0.945*** | 88 | 22,910,480 |

One of the standards and highly effective approaches to improve the deep neural network-based classifier on a small image dataset is to apply a pre-trained neural network [14]. A pre-trained neural network is a trained neural network model that was previously trained on a larger and more generic dataset such as the ImageNet dataset [5]. This makes the spatial hierarchy of features learned by the pre-trained network useful for many different computer vision problems. We develop four classifiers using TensorFlow and Keras. Each classifier consists of a based model with pre-trained weights trained on the ImageNet dataset, followed by three 64-node dense layers and two 32-node dense layers with ReLu activation function [11], and a 2-node prediction layer (the last layer) with Softmax activation function. We applied a binary cross-entropy as the loss function for this binary image classification. Accuracy is used as the primary performance metric. To compare the effectiveness of models, the Receiver Operating Characteristics (ROC) curve and the Area under the ROC Curve (AUC) metrics was used.

The pre-trained networks used in this experiment are MobileNetV2 [7], ResNet50V2 [6], InceptionV3 [16], and Xception [3]. We select four pre-trained ImageNet models based on their performances and popularity. Table 1 shows accuracy, size, and trainable parameters of these selected models. We observe that the accuracy of the model is proportional to the number of trainable parameters as highlighted by the red and black colors. (See full list comparison on `https://keras.io/api/applications/`).

## 2.4 Fine-tuning

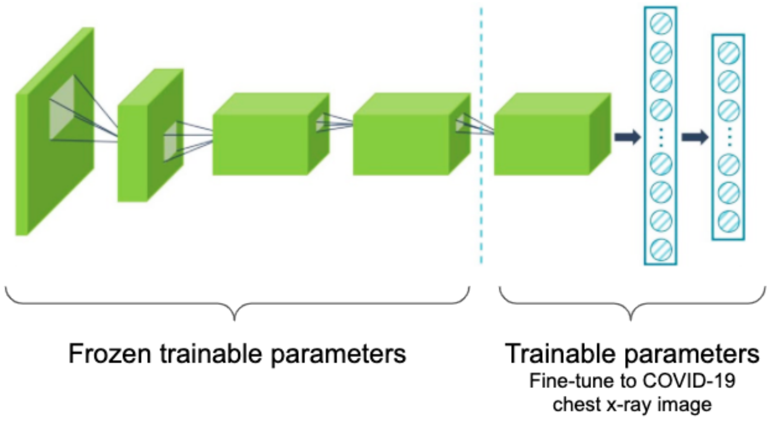
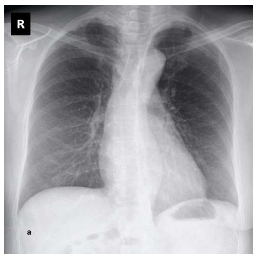

Frozen trainable parameters
Trainable parameters
Fine-tune to COVID-19 chest x-ray image

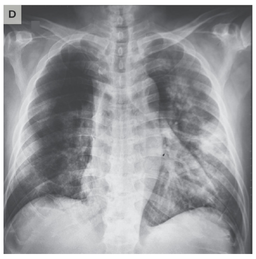
Figure 3: Fine-tuning pre-trained Convolutional Neural Networks

In this step, we aim to improve the effectiveness of the selected model by $fine - tuning$. Generally, fine-tuning consists of unfreezing a few of the last layers of a pre-trained model, and jointly training both the newly added layers and top layers of the frozen pre-trained model. Note that the details of the added part of the model are described in the previous section. The weights of the frozen pre-trained model are not updated during the training because we want to preserve the representations previously learned on the larger and more generic dataset. Here, we slightly adjust the weights of the newly added layers in order to make the feature representation of the model being retrained more relevant to a particular problem (i.e., chest x-ray image classification in this work). Thus, after the best classifier out of four is selected, we varied the percentage of trainable layers from 33% to 55% to compare the effect of the percentage of unfreezing layers on the selected model's accuracy. A summary of the selected model consisting of the layer types along with their order on the model, the input and output shape of each layer, the number of parameters in each layer, and the total number of parameters in the model can also be found in Figure 10 in the appendix section.

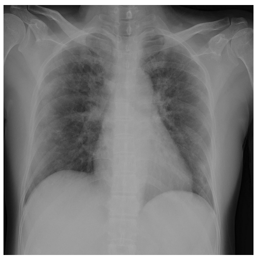
## 3 Dataset



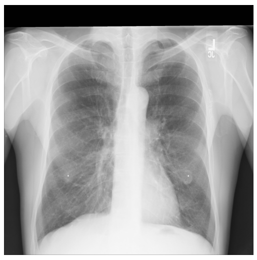
(a) Covid, Train



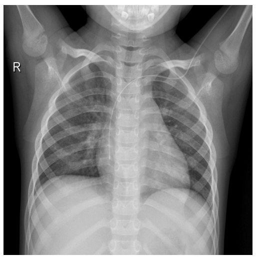
(b) Covid, Validate



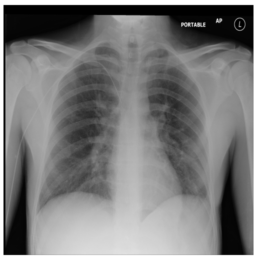
(c) Covid, Test



(d) Non-Covid, Train



(e) Non-Covid, Validate



(f) Non-Covid, Test

Figure 4: Examples of images from Training, Test, and Validation Set

| **Classes** | **Number of Images** | | | | **Class Prior** |
|---|---|---|---|---|---|
| | **Train** | **Validate** | **Totals** | **Test** | |
| COVID-19 | 214 | 26 | 240 | 26 | 24.3% |
| Non-COVID | 692 | 86 | 748 | 86 | 75.7% |
| Total | 906 | 112 | 988 | 112 | 100% |

Table 2: Dataset Distribution

The datasets used in this work come from the hackathon session in Deep Learning and Artificial Intelligence Summer School 3 [2] (DLAI3, `https://deeplearningandaiwinterschool.github.io/`). The dataset was collected from various sources, but images from each source were limited. The dataset consists of two sets: (i) the mini-dataset; and (ii) the dataset of annotated CXR images. The mini-dataset consists of 50 images that belong to two classes: COVID and non-COVID. Each class consists of 25 images. For the annotated CXR images dataset, it is split into training and testing sets. The training set consists of 125 COVID images and 551 non-COVID images. The testing set consists of 118 images of the COVID class and 290 images of the non-COVID class. Thus, there are (125+25+118 = 268) images that belong to the COVID class and (551+25+290 = 866) images of the non-COVID class. The images from these two sets were combined and randomly split into an 80% training set, 10% validation set, and 10% test set. Examples of images from the training, test, and validation set are shown in Figure 4. Table 2 shows the final number of images of two classes in each set. We obviously see that the DALAI3 dataset is an imbalanced dataset. Only about 24.3% of the whole samples are COVID-19 samples. From the literature, we have found that there are two main approaches that can mitigate the problem of the imbalanced dataset: (i) data sampling; (ii) algorithmic adjusting. Specifically, data sampling manipulates each class samples from the whole dataset to have an equal number of samples. For example, down-sampling reduces the number of samples of the majority class to match the number of the minority classes. Algorithmic adjusting modifies some parts of the algorithm to mainly focus on the minority class samples. One of the examples is to modify the loss function. A common loss function for an imbalanced dataset is Focal loss [10]. Inspired by the data sampling technique, we apply data augmentation only for the COVID class to increase the number of COVID class samples to match the non-COVID class samples.

## 4 Experiment and Results

### 4.1 Training

We used Adam [8] as an optimizer with a learning rate of 0.001 for all models. We applied a mini-batched of 32 examples and trained the models for 40 epochs maximum. Two dropout layers were applied after the fully connected layers. The early stopping technique was applied to stop the training process if the model's accuracy does not increase for 10 consecutive epochs.

### 4.2 Computational Setup

All experiments were performed on Google Colab (`https://colab.research.google.com/`). Python3 was the main programming language used with Graphic User Interface (GPU) as a hardware accelerator. NumPy and Pandas libraries were used for data preprocessing. TensorFlow V2 and Keras were mainly used for modeling and data manipulation including splitting the dataset into training, validation, and testing set. We use Matplotlib library for visualization.

## 4.3 ROC curve analysis

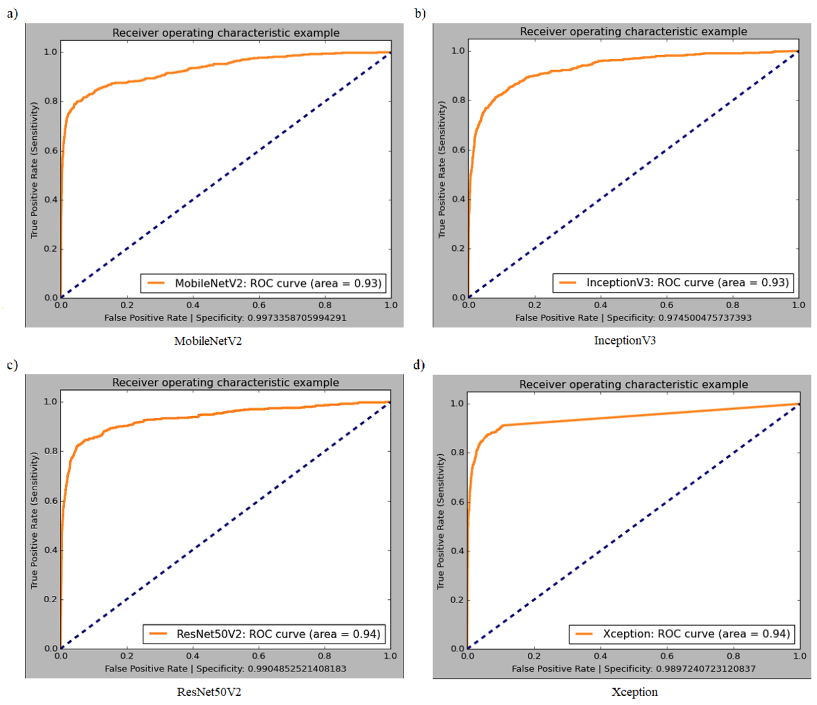


Figure 5: ROC curve comparison of initial four classifiers evaluated on the test set

To evaluate and visualize the performances of the classifiers, the Receiver Operating Characteristics (ROC) curve and the Area under the ROC Curve (AUC) are used. As illustrated in Figure 5, the ROC curves of all four models have highly AUC values. Interestingly, even though MobileNetV2 has a much smaller model size, and a lot less trainable parameters, its ROC curve has AUC 0.93 which is about the same as other bigger models. As one of our goals is to come up with a lightweight model that is friendly for resource-constrained devices. Thus, we select the pre-trained MobileNetV2 based model as a candidate model to further training to improve the model's performance.

## 4.4 Effect of percentage of unfreezing layers of the pre-trained MobileNetV2 based model

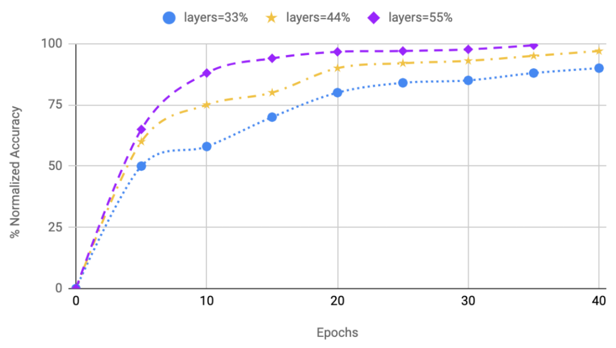


Figure 6: The effect of percentages of unfreezing layers of the pre-trained MobileNetV2 based models

To improve the effectiveness of the pre-trained MobileNetV2 model, we varied the percentage of trainable layers from 33% to 55% and results in three models with different percentages of unfreezing layers. Here, we still use the Adam optimizer with a learning rate of 0.001, and weight decay of 0.0001. As shown in Figure 6, the results from our experiments indicate that a higher percentage of unfreezing layers provides a higher accuracy and takes fewer epochs to reach convergence. A plot of the network neural network graph showing layers and their order in the model, the input and output shape of each layer is illustrated in Figure 9 in the appendix section.

## 4.5 Grad-CAM visualization

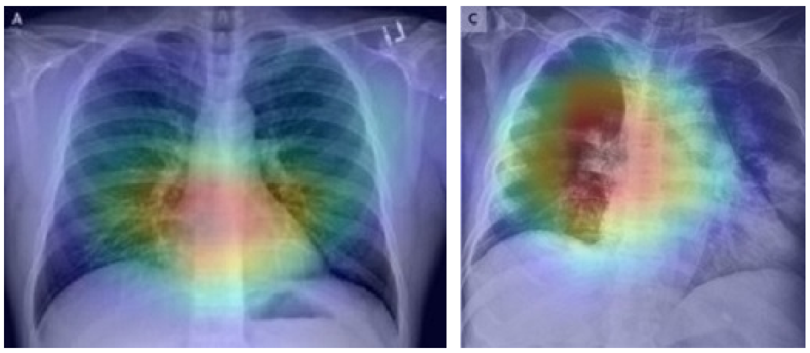
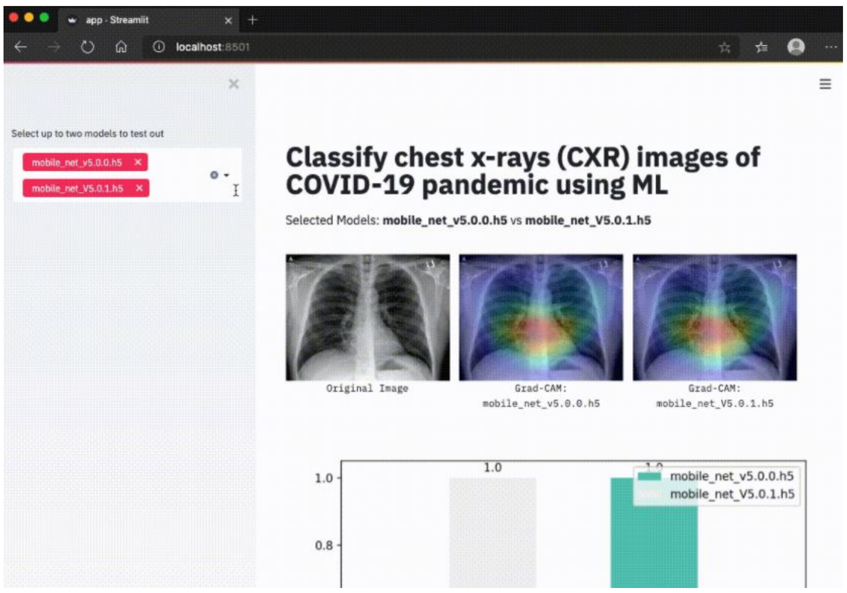


Figure 7: Samples of Grad-CAM visualization on chest x-ray images that were correctly classified by the pre-trained MobileNetV2 based model

To help further understand the results and what parts of the input images the neural networks internally pay attention to, we incorporate a visualization technique called Grad-CAM [12]. The Gradient-weighted Class Activation Mapping (Grad-CAM) looks into the activation maps of a specified convolutional layer to visualize how input images affect the output of neural networks. These heatmaps are generated from the gradient of a last convolutional layer of a pre-trained MobileNetV2. As shown in Figure 7, bright red indicates the region of pixels that carry the highest weight parameters and importance in the model's prediction of the image's ground truth. However, the domain knowledge is needed to finalize whether the heatmaps focus on the correct part.

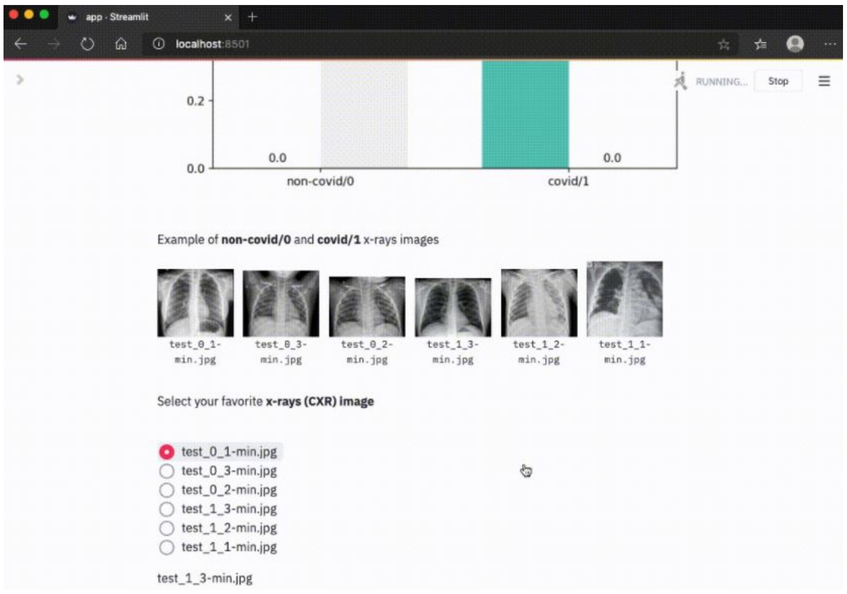
## 4.6 A highly interactive web application



**Classify chest x-rays (CXR) images of COVID-19 pandemic using ML**

Selected Models: **mobile_net_v5.0.0.h5** vs **mobile_net_V5.0.1.h5**

(a) Select models to classify the chest x-ray images and compare their performance.



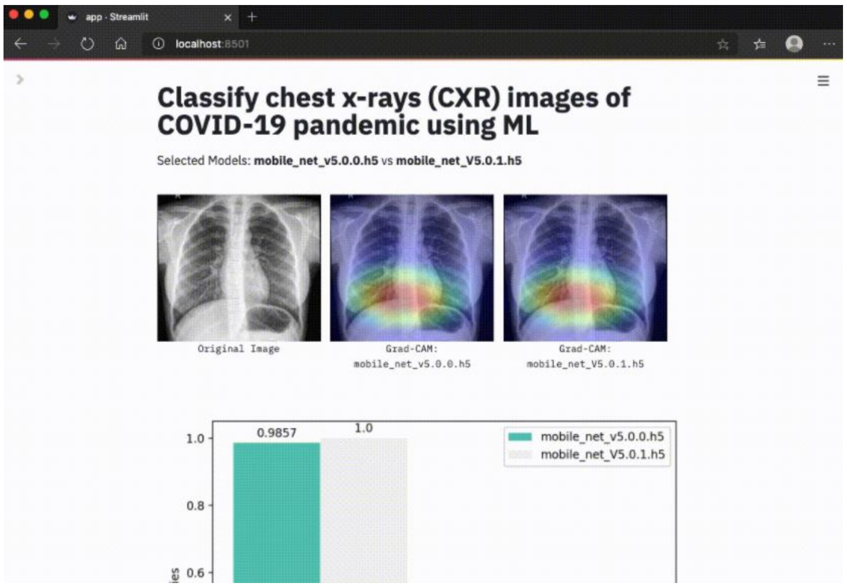
Example of **non-covid/0** and **covid/1** x-rays images

Select your favorite **x-rays (CXR) image**

[✓ test_0_1-min.jpg]
test_0_3-min.jpg
test_0_2-min.jpg
test_1_3-min.jpg
test_1_2-min.jpg
test_1_1-min.jpg

test_1_3-min.jpg

(b) Select samples to test the performance of the selected models.



**Classify chest x-rays (CXR) images of COVID-19 pandemic using ML**

Selected Models: **mobile_net_v5.0.0.h5** vs **mobile_net_V5.0.1.h5**

(c) The performance of the selected models is shown along with Grad-CAM visualizations showing what parts of the input images the models internally pay attention to

Figure 8: A highly interactive web application to test and compare the trained deep learning models for COVID-19 x-rays (CXR) images classification

To aid the screening methods for the early COVID-19 diagnosis for preventing the spread of the disease, we develop a highly interactive web application for testing and comparing the trained deep learning models for COVID-19 x-rays (CXR) images classification. In this way, the more accurate developed classification model can be uploaded and publicly used to classify COVID-19 x-rays (CXR) images. Figure 8 shows the screenshot of a workflow of the developed web application step-by-step. As shown in Figure 8a, users can select models to classify the chest x-ray images and compare their performance. Figure 8b illustrates the case in which our web application also provides some samples to users who do not have the input data for the models but want to test out the models. Lastly, the performance of the selected models is shown along with Grad-CAM visualizations showing what parts of the input images the models internally pay attention to, as illustrated in Figure 8c.

## 5 Conclusion

The need for a quick, cheap, and accurate screening method for COVID-19 in patients has inspired and motivated us to work on this project. This work develops and evaluates the performance of four customized pre-trained ImageNet based deep neural networks at binary-class COVID-19 classification task. The techniques such as data augmentation, increasing dropout layers, and applying weight decay can mitigate the overfitting partially caused by the dataset provided. Also, these techniques have shown to be useful for image classifiers to learn quickly and generalize better. The lightweight pre-trained MobileNetV2 has found to provide the best accuracy considering the model's size and trainable parameters, resulting in the resource-efficient model. This enables deploying the lightweight deep learning-based model on devices with limited computing power such as mobiles and embedded systems to perform real-time image classification. The Gradient-weighted Class Activation Mapping (Grad-CAM) technique was incorporated to better understand the model's prediction results. Lastly, the highly interactive web application was developed to showcase how to deploy and leverage the lightweight models to the real-world production

The project can be extended by incorporating broader categories in larger datasets to obtain more robust and generic features. The attention-based mechanism, as a more powerful technique for selectively focusing on particular parts of an input image, can be applied to hopefully improve our lightweight models developed.

## References

[1] Tao Ai et al. “Correlation of chest CT and RT-PCR testing for coronavirus disease 2019 (COVID-19) in China: a report of 1014 cases”. In: *Radiology* 296.2 (2020), E32–E40.

[2] Jonathan H Chan. *DLAI3 Hackathon Phase3 COVID-19 CXR Challenge*. 2020. DOI: `10.34740/KAGGLE/DSV/1347344`.

[3] François Chollet. “Xception: Deep learning with depthwise separable convolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1251–1258.

[4] *Coronavirus Disease (COVID-19) Situation Reports*. URL: `https://www.who.int/emergencies/diseases/novel-coronavirus-2019/situation-reports`.

[5] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[6] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[7] Andrew G Howard et al. “Mobilenets: Efficient convolutional neural networks for mobile vision applications”. In: *arXiv preprint arXiv:1704.04861* (2017).

[8] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[9] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *nature* 521.7553 (2015), pp. 436–444.

[10] Tsung-Yi Lin et al. “Focal loss for dense object detection”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988.

[11] Vinod Nair and Geoffrey E Hinton. “Rectified linear units improve restricted boltzmann machines”. In: *ICML*. 2010.

[12] Ramprasaath R Selvaraju et al. “Grad-cam: Visual explanations from deep networks via gradient-based localization”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 618–626.

[13] Heshui Shi et al. “Radiological findings from 81 patients with COVID-19 pneumonia in Wuhan, China: a descriptive study”. In: *The Lancet Infectious Diseases* (2020).

[14] Hoo-Chang Shin et al. “Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning”. In: *IEEE transactions on medical imaging* 35.5 (2016), pp. 1285–1298.

[15] Krishna Kant Singh, Manu Siddhartha, and Akansha Singh. “Diagnosis of Coronavirus Disease (COVID-19) from Chest X-ray images using modified XceptionNet”. In: *Romanian Journal of Information Science and Technology* 23.657 (2020), pp. 91–115.

[16] Christian Szegedy et al. “Rethinking the inception architecture for computer vision”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.

[17] Yang Yang et al. “Laboratory diagnosis and monitoring the viral shedding of 2019-nCoV infections”. In: *MedRxiv* (2020).

[18] Zi Yue Zu et al. “Coronavirus disease 2019 (COVID-19): a perspective from China”. In: *Radiology* (2020), p. 200490.

## A Appendix

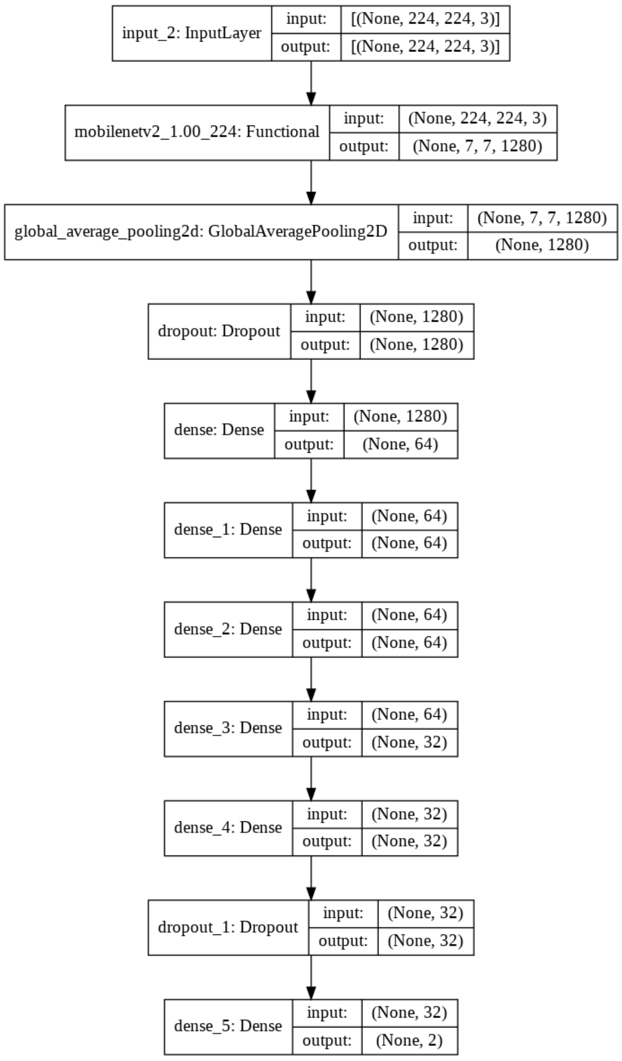
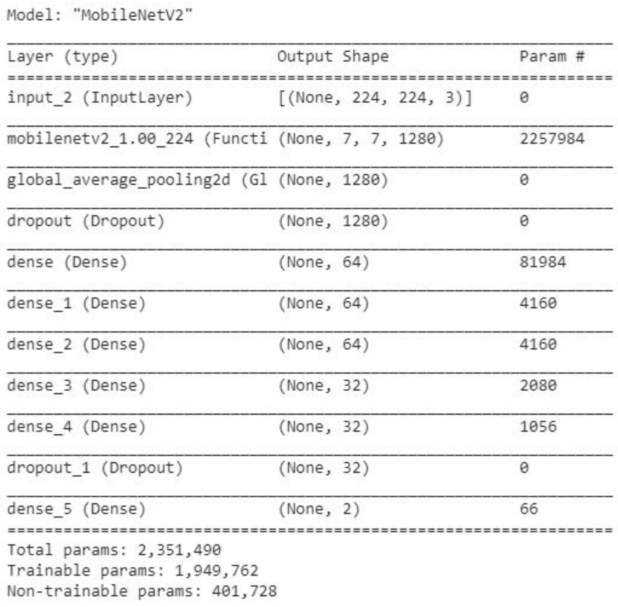


Figure 9: A plot of the network neural network graph showing layers and their order in the model, the input and output shape of each layer



```
Model: "MobileNetV2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
mobilenetv2_1.00_224 (Functi (None, 7, 7, 1280)        2257984
_________________________________________________________________
global_average_pooling2d (Gl (None, 1280)              0
_________________________________________________________________
dropout (Dropout)            (None, 1280)              0
_________________________________________________________________
dense (Dense)                (None, 64)                81984
_________________________________________________________________
dense_1 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_2 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_3 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_4 (Dense)              (None, 32)                1056
_________________________________________________________________
dropout_1 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_5 (Dense)              (None, 2)                 66
=================================================================
Total params: 2,351,490
Trainable params: 1,949,762
Non-trainable params: 401,728
```

Figure 10: A summary of the selected model consisting of the layer types along with their order on the model, the input and output shape of each layer, the number of parameters in each layer, and the total number of parameters in the model