Final Report

CMPE/CISC 452

Group 43

December 2020

Group Members:

Rachel Coreau | 20066181

Owen Kent | 20055228

Adam Grace | 20055169

Anshul Pattoo | 20124223

Problem Description and Motivation

The new coronavirus (COVID-19), declared by the World Health Organization (WHO) as a pandemic, has resulted in more than 71.2 million infections and 1.6 million deaths worldwide [1]. The widespread and severe impact of this event has highlighted the need to gather all available resources towards its mitigation. While a diagnosis of COVID is confirmed using a polymerase chain reaction (PCR), infected patients with pneumonia may present on chest x-ray images with a pattern that is only moderately characteristic for the human eye [2]. Towards this end, we examined the possibility of employing neural networks that can assist in the diagnosis of patients through chest x-rays.

Our work builds on numerous state-of-the-art convolutional neural networks (CNNs) and aims to perform a classification of chest x-ray data into important categories. Due to the use of a smaller dataset, our implementations thoroughly fine-tune the following CNN base models: ResNet50, EfficientNetB7, VGG-16, and NASNetMobile. These base models use selected pre-trained weights from the ImageNet dataset. We also implement other extensions in terms of data pre-processing to improve network learning.

All of our models develop existing implementations published on Kaggle.

Data Description

CoronaHack-Chest X-Ray-Dataset

We used the open access CoronaHack-Chest X-Ray-Dataset for our projects [3]. It provides 5910 chest X-rays of healthy and pneumonia-infected patients. All pneumonia chest X-rays are labelled as stress-smoking, viral or bacterial, and some are further classified by various causes (one being COVID-19). Table 1 shows all categories, and their image counts in the complete dataset.

Table 1: Summary of the CoronaHack -Chest X-Ray-Dataset [3].

Label	Virus Category: Label 1	Virus Category: Label 2	Image Count
Normal	N/A	N/A	1576
Pneumonia	Stress-Smoking	ARDS	2
Pneumonia	Virus	Unknown	1493
Pneumonia	Virus	COVID-19	58
Pneumonia	Virus	SARS	4
Pneumonia	Bacteria	Unknown	2772
Pneumonia	Bacteria	Streptococcus	5

The images are provided in two folders, one contains 5286 images intended for training and the other contains 624 images intended for testing (about an 89% / 11% split). Interestingly, the testing images are only labelled as normal, viral pneumonia, or bacterial pneumonia. Because of this, training and testing sets must be created manually for more specific classification tasks, such as identifying COVID-19 chest x-rays. Labels with references to the images are provided in a spreadsheet, as is a data summary similar to the one found in Table 1. Data preprocessing was performed individually since we all had different classification tasks.

Dr. Joseph Cohen's COVID Image Data Collection

One of the models gathered additional COVID-positive x-rays from another dataset: Dr. Cohen's GitHub COVID image data collection [4]. This dataset was commonly used in the related studies. The metadata contains many attributes, including the study from which the corresponding image was obtained. The attributes of interest to us are finding and filepath. The finding attribute indicates whether the x-ray is of a healthy patient or pneumonia patient. If the x-ray shows pneumonia, the particular infection causing the pneumonia is indicated as well. The filepath attribute simply indicates the image file name associated with the record.

The finding values are arranged in a hierarchy. An example finding value would be "Pneumonia/Bacterial/Mycoplasma" or "Pneumonia/Viral/COVID-19".

Type	Genus or Species
Viral	COVID-19 (SARSr-CoV-2) SARS (SARSr-CoV-1) Varicella Influenza
Bacterial	*Streptococcus* spp. *Klebsiella* spp. *Escherichia coli* *Mycoplasma* spp. *Legionella* spp. Unknown *Chlamydophila* spp.
Fungal	*Pneumocystis* spp.
Lipoid	Non applicable

Figure 1: The figure shows the various labels for the finding attribute and captures its hierarchical composition [4].

ResNet50 | Rachel Coreau | 20066181

Data Preparation

Original: Firstly, the train data is filtered to just represent Label: Normal and Pneumonia, and Label_2_Virus_category: COVID-19. Next, the target and class features are created where the target is either 0 (if Normal) or 1 (if COVID-19). Thus, each train image consists of the X_ray_image_name (.jpeg name), the class (0 or 1), the target (0,1), and Label_2_Virus_category (COVID-19 or ___). The COVID-19 images are then augmented to increase the diversity of the samples and the amount of data and added with the rest of the train data. Finally, the train and test images are converted to tensors.

Extensions: The first extension I made to the data preparation was to include more dimensions in augmenting the COVID-19 images. In addition to sear_range (applying shearing transformations) and zoom_range (zooming inside pictures), I used rotation_range (rotate pictures), rescale (scale with a 1/255 factor to target values between 0 and 1), width_shift (translate picture vertically), height_shift (translate picture horizontally), horizontal_flip (flipping images horizontally), vertical_flip (flipping images vertically), and brightness_range (adjust the brightness). This was done to further increase the diversity between the images.

Model Implementation and Validation

Original: The original model is an implementation of transfer learning using the ResNet50 model [5]. The train dataset is shuffled in order to reduce over-fitting by avoiding bias/patterns before training. The batch size is set to 16, the buffer to 1000. Next, the base model is defined as RESNET-50 with include_top = False to allow another output layer to be added and trained. This model is set to trainable = False because it prevents the weights in a layer to be updated during training. If training = True, it would override the pretrained weights in the model. A new Sequential model is created, which includes the RESNET-50 model, GlobalAveragePooling2D(), Dense(128), Dropout(0.2), and Dense(1, activation = "sigmoid"). Global Average Pooling calculates the average output of each feature in the previous layers, which removes a large number of trainable parameters. The dense layer specifies the output shape and performs the dot product between the input and weight matrix. The dropout layer sets a fraction of the input units to 0, which helps prevent over-fitting. The callback is defined as stopping training when there is no substantial value loss after 3 epochs.

Extensions: The first extension to the model was fine-tuning the pre-existing model. This involved unfreezing the highest layer of the RESNET model, allowing it to be trainable. Therefore, very obvious features in the starting layers are not changeable, but the most specific features are allowed to be fine-tuned. Dense and Dropout layers were added to the model until the test error did not improve anymore. With these changes, after the third epoch the accuracy started decreasing, so the number of epochs was reset to 2.

Results

Original: The original results were on average 70% taking 4 epochs. The train data trained 0 for Normal images and 1 for COVID-19 images, however the test data does not contain any COVID-19 images, so it is being tested on 0 for Normal images and 1 for Pneumonia. This contributes to the accuracy largely fluctuating each time the code is run. Therefore, after testing 10 epochs, the accuracy is very high but the val_accuracy is significantly lower and overfitting.

	precision	recall	f1-score	support
0	0.58	0.85	0.69	234
1	0.87	0.64	0.74	390
accuracy			0.71	624
macro avg	0.73	0.74	0.71	624
weighted avg	0.76	0.71	0.72	624

Figure 2: Evaluation metrics of the original model [5].

Extensions: After the extensions, the results were on average 75% taking 2 epochs. The graph created below shows the relationship between the train and test accuracy. In conclusion, obtaining more COVID-19 images and including them in the test data would increase the test accuracy.

```
In [79]: model.fit(train_batches, epochs=2, validation_data= test_batches, callbacks=[callbacks])

         # Predict the test data
         pred = model.predict_classes(np.array(test_arrays))

         Epoch 1/2
         164/164 [==============================] - 371s 2s/step - loss: 0.7337 - accuracy: 0.7116 - val_loss: 0.5709 - val_ac
         curacy: 0.7051
         Epoch 2/2
         164/164 [==============================] - 366s 2s/step - loss: 0.4722 - accuracy: 0.8182 - val_loss: 0.5639 - val_ac
         curacy: 0.7452

In [80]: from sklearn.metrics import classification_report, confusion_matrix
         print(classification_report(test_data['target'], pred.flatten()))

                       precision    recall  f1-score   support

                    0       0.64      0.72      0.68       234
                    1       0.82      0.76      0.79       390

             accuracy                           0.75       624
            macro avg       0.73      0.74      0.73       624
         weighted avg       0.75      0.75      0.75       624
```

Figure 3: Training results and evaluation metrics of the extended network.

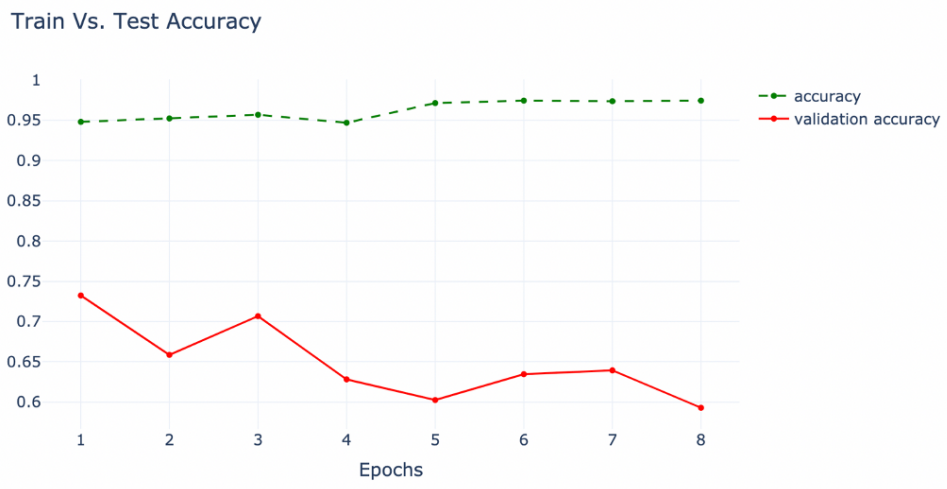

Figure 4: Accuracies of the model as a function of epoch.

EfficientNetB7 | Owen Kent | 20055228

Data Preparation

The Corona-Hack metadata file is read to guide the data preparation phase. The labels of the data are altered from 'Normal' or 'Pneumonia' to 0 or 1, respectively. The dataset is then split into the two, preset subsets: training and testing. These two subsets are then further divided into two inner-subsets: normal or pneumonia. The images in each of the four subsets are then saved to their own respective directories; the reason behind this is that by saving the images in class-dependent directories, Pandas allows for their class to inferred for ease of use. Prior to entering the network, the images are augmented through shearing, rotation, shifting, zooming, and flipping; this increases the size and diversity of the entire dataset and can lead to improved results.

Model Implementation and Validation

The original model [6] implements transfer learning using the EfficientNetB7 for the binary classification of pneumonia. The EfficientNetB7 is entirely non-trainable except for its block7c_project_conv layer [6], which is an interior convolutional layer with 2,457,600 trainable parameters. Following the output of the EfficientNetB7, the model implements a max pooling layer, followed by dropout layer, followed by a flatten layer, and finally a 1-node dense layer with a sigmoid activation for the output. The model utilizes the binary cross entropy loss function due to its binary output and the Adam optimizer.

The model was altered first by changing the layers of which are fine-tuned in the EfficientNetB7. Rather than one of the middle layers being tuned, the final three layers of the network were set to be trainable which includes an activation layer with no trainable parameters, a batch normalization layer with 10,240 trainable parameters, and a convolutional layer with 1,638,400 trainable parameters. Although the number of trainable parameters is less than the original model, the fact that they are the final layers of the pre-trained CNN means that they detect key details, or features, of the images they are trained on, and thus, are vital for the classification of images.

In addition to the alteration of the fine-tuned layers, a 100-node dense layer with a sigmoid activation was added prior to the output layer. The reason behind the additional dense layer is that the output from the pre-trained CNN contains the key details, or features, from the images. The added dense layer can then be utilized to combine and access these details to aid in the classification process.

The architecture of the final model can be seen in the figure below.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
efficientnet-b7 (Functional) (None, 7, 7, 2560)        64097680
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 3, 3, 2560)        0
_________________________________________________________________
flatten (Flatten)            (None, 23040)             0
_________________________________________________________________
dense (Dense)                (None, 100)               2304100
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 101
=================================================================
Total params: 66,401,881
Trainable params: 3,947,721
Non-trainable params: 62,454,160
```

Figure 5: Architecture of the final network displaying the layers and parameter counts.

Finally, in light of the changes made and highlighted above, it was found through iterative training that the model performed optimally with a lesser batch size of 32 from 64 and for it to be trained only for 5 epochs rather than 15 [6].

Results

The network was able to accurately classify and differentiate between pneumonia and normal chest x-ray images. The new model marginally outperformed the base model with its accuracy and AUC; the new model had a training AUC and accuracy of 0.97-0.99 and 0.93-0.95 respectively and had a testing AUC and accuracy of 0.96-0.98 and 0.87-0.91 respectively. These results were slightly superior to the base model which produced a training AUC and accuracy of 0.9778 and 0.9372 respectively and a testing AUC and accuracy of 0.9632 and 0.9045 respectively [6]. The results recorded throughout the training of the network can be seen in the figure below.

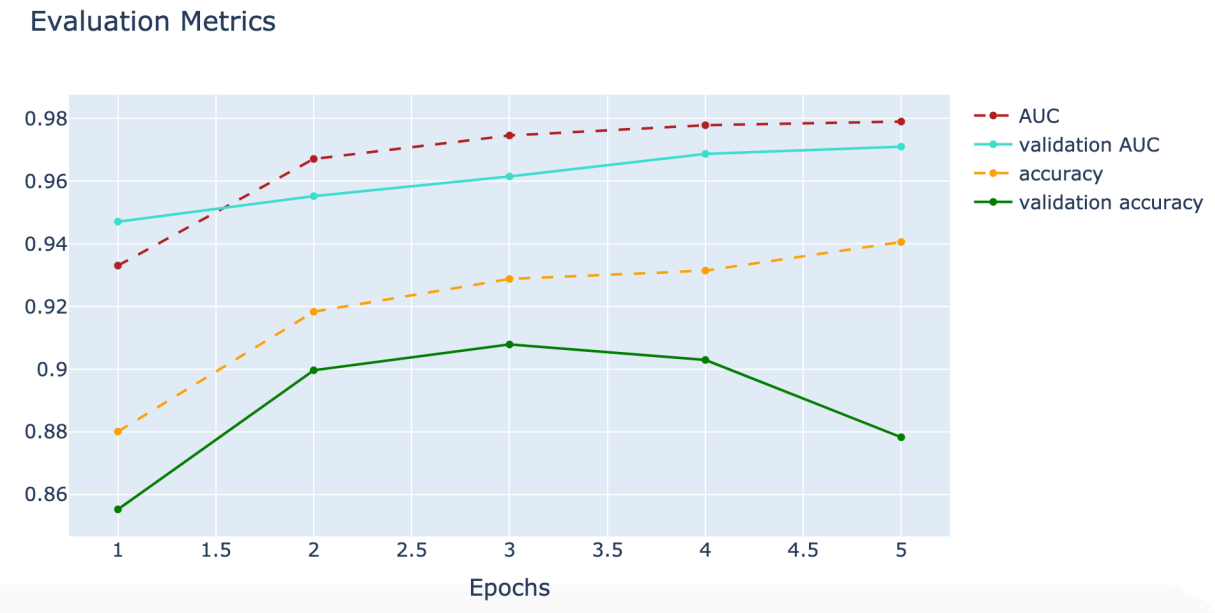

Figure 6: Display of the evaluation metrics as a function of epoch.

Although improvements in the model's accuracy were marginal, the noteworthy improvement of the model lies with its convergence during training. The change in layers being fine-tuned and the addition of a dense layer prior to the output led to a significant improvement in training speed. The model consistently achieved equal, if not better, results to the original model after training for 15 epochs in just 5 epochs. The altered model could be trained in one third of the time of the original.

VGG-19 | Adam Grace | 20055169

My goal was to create a convolutional neural network that employs transfer learning to classify between healthy, viral pneumonia and bacterial pneumonia chest x-rays. I used Jupyter Notebook (Python programming language) and Keras to develop my model.

Data Preparation

Using the Pandas library, I read the meta data spreadsheet that was included in the dataset into a DataFrame object. I remove the 2 stress-smoking pneumonia rows since my classification task does not include this type of pneumonia. I added a class column (Normal, Virus or Bacteria) and target column (0, 1 or 2) to clearly identify the desired classification of each of the images. I then partitioned this object into a training DataFrame object and a testing DataFrame object based on the labels provided by the meta data's first column. I removed all columns from the two objects except for the X-ray image filename, the class and the target. This produced 5284 training images and 624 testing images (about 89% training, 11% testing). The image count in each class can be found in Figure 7. I decided that image augmentation was not necessary since there was a sufficient number of images in each class. If I had attempted a more specific classification task, such as identifying COVID-19 chest X-rays, I would have augmented the classes that had few images provided by the dataset.

```
Training Data Image Count
Bacteria    2535
Virus       1407
Normal      1342
Name: Class, dtype: int64
------------------------
Testing Data Image Count
Bacteria    242
Normal      234
Virus       148
Name: Class, dtype: int64
```

Figure 7: Image classes for both the training and testing subsets.

I used Keras's image processing functionality to load the images in a 224x224x3 shape (required by the VGG-19 model) and convert them into arrays. I then rescaled the arrays to values between 0 and 1, then appending them to a training list or testing list. I converted these two lists into Tensor objects and created separate Tensor objects for their desired classifications. I grouped the training images and desired classifications into one tensor and the testing images and desired classifications into another. Lastly, I shuffled the order of the images (the meta data was partially ordered) and created batches of 16 images.

Model Implementation and Validation

I chose to use the VGG-19 pre-trained model with the ImageNet weights as my base model for transfer learning because it has been shown to be successful on other chest X-ray classification tasks relating to pneumonia [7] [8]. One journal article investigated 17 pre-trained models for classifying chest X-rays between COVID-19 and healthy or other pneumonia infections (binary classification) [7]. Although they found DarkNet-19 to be the most successful, VGG-19 had a 90% validation accuracy. They used the same

CoronaHack-Chest X-ray-Dataset as well as accredited image data from the Vancouver General Hospital. Another journal article investigated the VGG-16 and VGG-19 pre-trained models for classifying chest X-rays between healthy or pneumonia infected (binary classification) [8]. They found that VGG-19 was the more successful of the two, with an accuracy of about 96%. They used the Chest X-Ray Images dataset that is publicly available on Kaggle. Unfortunately, neither article provides access to the source code used to train their VGG-19 based neural networks.

The source code that I started with is a Jupyter Notebook submission from the CoronaHack -Chest X-Ray-Dataset page on Kaggle [5]. It attempts to classify the chest X-rays between COVID-19 and healthy (binary classification) using ResNet-50 as the pre-trained base model. I repurposed this code for my own contrasting classification task. After configuring the data preparation as outlined above, I imported the VGG-19 model with the ImageNet weights as my base model. I set all base model layers to not be trainable and didn't include the top 3 fully connected layers. I then had a global average pooling layer to minimize overfitting and reduce the number of parameters, followed by a 128-output dense layer with no activation function. This dense layer was what required the most training, leading to about 25 minutes per epoch on my laptop (Intel® Core™ i7-8550U CPU @ 1.80GHz). Next was a dropout layer with a dropout fraction of 0.2 to prevent overfitting. Lastly, a 3-output dense layer with a softmax activation function was used to classify the chest X-rays as either normal, viral pneumonia or bacterial pneumonia. In initial attempts I used a sigmoid activation function in this final layer, but I realized that softmax was more appropriate for a non-binary classification task. I also changed the loss function from binary cross entropy to categorical cross entropy for the same reason.

```
model = Sequential()
model.add(base_model)
model.add(GlobalAveragePooling2D())
model.add(Dense(128))
model.add(Dropout(0.2))
model.add(Dense(3, activation = 'softmax'))

model.compile(optimizer='adam', loss = 'categorical_crossentropy', metrics=['accuracy'])
```

Figure 8: Code snippet showing the architecture and configuration of the model.

After training the model over 5 epochs with a batch size of 16, I generated a classification report and confusion matrix using the testing dataset, which can be found below. The weights achieved after only the second epoch were used because the validation loss was greater for the third and fourth epoch.

Results

The model achieved a very poor overall accuracy of 56% on the testing images. The normal class achieved perfect precision but very low recall, indicating that there were no false positives but many false negatives. After looking at the confusion matrix, it is evident that this is because many normal chest X-rays were being classified as bacterial pneumonia chest X-rays. This is further reflected in the perfect recall and low precision of the bacterial pneumonia class. I believe that this may be due to the ratio of bacterial pneumonia training images and normal or viral pneumonia training images (there are approximately twice as many relative to the other two classes).

Table 2: Classification report from the trained model predicting the testing data.

Class	Precision	Recall	F1-score
Normal	1.00	0.12	0.22
Viral Pneumonia	0.56	0.53	0.55
Bacterial Pneumonia	0.53	1.00	0.69

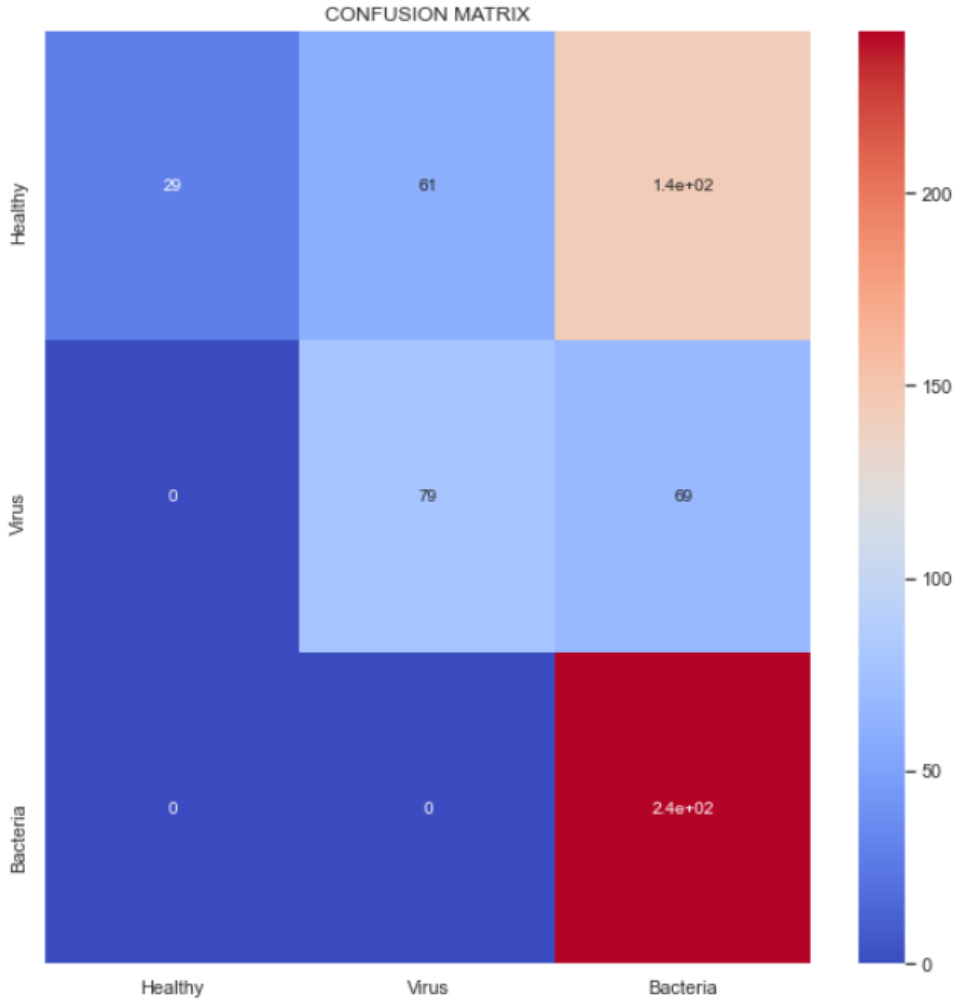

Figure 9: Confusion matrix from the trained model predicting the testing data.

I believe that my results can be greatly improved by making two changes. First, the training dataset should be selected so that the number of images of each class is approximately equal. This can be done by removing some bacterial pneumonia images, or by augmenting the normal and viral pneumonia images. Second and most important, the addition of another or several more dense layers immediately after the base model is likely to improve classification. However, with this change comes a longer training time. I attempted to try this using the GPUs provided by Google Colab, however I was unable to load the chest X-ray images into the provided RAM without crashing the program. This could likely have been fixed by loading one batch at a time, however due to time constraints I was unable to figure out how to do this.

NASNetMobile | Anshul Pattoo | 20124223

My model classifies chest X-rays into COVID-positive images and healthy ones.

NASNetMobile was selected as the CNN base model because it is not largely studied in the related literature. Only *one* study was found to examine NASNetMobile, and the study's authors do not make their source code available to the public. Thus, an implementation from Kaggle that fine-tunes ResNet50 was re-purposed for my work.

The model was developed in Python, and the notable libraries used were Scikit-learn and Keras. The development was done in Google Colab. As an aside, please note that any mention of "base model" or "base CNN" in my refers to the NASNetMobile model itself.

Data Preparation

The quality of any model is generally constrained by the quality of the training data. Deep learning models in particular perform extraordinarily well when there is a large and diverse dataset.

The CoronaHack-Chest X-Ray-Dataset alone is not sufficient to train our model, even considering that the base model is pre-trained on ImageNet. Thus, further reputable data was added. This dataset is from Dr. Joseph Cohen's GitHub repository and is commonly used amongst researchers for image classification of COVID X-rays. Below, I outline the series of major steps taken for data preparation and indicate differences from the original source code. The metadata file of each dataset guided the pre-processing of the data.

1. Any records that indicate a pneumonia *not* caused by COVID-19 were removed. Such records are not considered for the classification task. *The original source code does not complete a Normal vs. COVID separation in the test set. This separation is only done in the training set.* The original test set instead does a separation between an X-ray of *any* pneumonia and a healthy X-ray. This produces a notable discrepancy between how the original model learns and predicts. The original model learns to separate COVID X-rays from healthy ones but is tasked to classify *general* pneumonia X-rays from healthy ones. *It is unclear whether this choice was deliberate or accidental*.

Given our classification task, the modified model aims to predict on an image of COVID or healthy, and trains on such images.

2. The Corona-Hack dataset separates the images into two folders: train and test. Given this, two arrays were generated: train_arrays and test_arrays. Each image is converted to an array of shape (224, 224, 3), which is different from the shape implemented in the original code. The original code uses ResNet50 as the CNN base, which accepts an image of a different shape.

3. train_arrays and test_arrays are concatenated into a single dataset: x_dataset. The original code does not do this. This was a deliberate design choice. The goal is to eliminate any bias that might occur in the model from the pre-determined separation between the training and test set. A separate dataset containing the corresponding output labels, y_dataset, was developed as well.

4. Since the proportion of COVID-positive samples in the dataset was extremely low, COVID-positive samples from another dataset were added to our dataset. These records were taken from Dr. Joseph Cohen's GitHub repository. This addition of COVID-positive records from a separate dataset was not

done in the original source code. The necessary additions were made to the x_dataset and y_dataset arrays.

5. The proportion of COVID-positive samples in the overall dataset was still low, so more COVID-positive samples were generated through data augmentation. The original source code does implement data augmentation, but the augmentation conducted by the modified code is more extensive. The modified code completes four additional augmentations: random rotations in the range of [-40°, 40°], mirror reflections, and shifts along both the X-axis and Y-axis by 20%. The batch_size was also deliberately kept at 4 for each augmented image. It is important to not introduce *too* much noise into the data, as this may adversely affect the model's learning. This section is also different from the original code, as the original code implements a batch_size of 32. These augmented records were added to the existing dataset (i.e., the proper additions to x_dataset and y_dataset were made).

6. The proportion of COVID-positive samples in the overall dataset was finally deemed sufficient. The The train_test_split function of the sklearn.model_selection module was used to generate a 70-30 train-test split. This function eliminates any bias that might emerge from the ordering of the data. It randomly selects samples for the training set and test set.

7. The training and tests sets were converted to tensors, and batches were subsequently developed. None of this implementation was changed.

Model Implementation and Validation

The changes to the network itself are as follows:

1. All layers of the base CNN were frozen except for the last layer: separable_conv_2_bn_normal_left.

2. Several dense layers, each with a carefully selected number of neurons, were added. A dropout layer was placed after each added dense layer.

There was a great degree of trial-and-error involved in designing this network structure. Addressing the question of the appropriate number of layers and neurons is generally difficult. Thus, different dense-dropout layer combinations, each with different parameter values, were extensively attempted. The kerastuner module assisted greatly in the selection of the parameters for both the dense and dropout layers.

Dropout layers in particular were added to prevent the model from overfitting to the training data. This is a common technique applied for regularization.

The final network topology that led to high performance was the base model followed by a wide network (i.e. dense layers each containing a high number of neurons, with dropouts) and deep network (i.e. dense layers each containing a low number of neurons, with dropouts).

3. The learning rate was also optimized with kerastuner. The learning rate selected was 0.005, with a decay rate of 0.005/10. Learning rate decay provides an easy way to control overfitting for large neural network models. In the original model, a learning rate of 0.001 was implemented, with no decay rate. The number of epochs that provided for successful convergence was 3.

It is important to mention the dead-ends involved in building the modified model. This included combining the CNN with another classifier, such as a multi-layer perceptron or random forest algorithm.

Another dead-end was freezing blocks of layers in the base CNN. Neither of these techniques brought strong results.

Results

The original code from Kaggle fine-tunes ResNet50. Even though I have replaced ResNet50 with NASNetMobile in cell 28 of the provided original code file, it would still not be sensible to compare the performance of my model to the model implemented in the original code. As mentioned previously, the classification task associated with the original source code was to distinguish pneumonia X-rays from normal ones. *Our* task is to distinguish COVID X-rays from normal ones.

As a side note, the *provided* version of the original file replaces ResNet50 with NASNetMobile and changes the input image shape to (224, 224, 3). This is to allow comparison between the two codes only if it is of interest.

Only one study was found to implement NASNetMobile on chest X-rays [9]. Thus, more aptly, the results of our model are compared to the results of the model from the study. Below is a table summarizing the results of *various* models from the study, with NASNetMobile included. It is sensible to be skeptical of the result for NASNetMobile as this is the *only* study found to implement such model. Note that it is unclear whether this paper distinguishes between COVID and healthy X-rays: the labels used in the study are "COVID-19" and "Non-COVID-19". It is assumed that such labels indicate a COVID vs. healthy distinction.

Our performance results are robust and match that of the study. The test accuracy is within the range of 97-99%. If one or two more epochs were run, both the training and testing accuracy would be brought up to 100%. It would not be sensible to run such additional epochs, because our model would be too specific — too tailored — to the dataset used. Our dataset is too small to accept a 100% accuracy.

Table 4. Overall model's performance on X-ray train set.

Model	Performance			
	Accuracy	Precision	Recall	F1-Score
VGG16	1.0	1.0	1.0	1.0
InceptionResNetV2	0.99	0.99	0.99	0.99
ResNet50	0.64	0.79	0.64	0.58
DenseNet201	1.0	1.0	1.0	1.0
VGG19	0.98	0.98	0.98	0.98
MobileNetV2	1.0	1.0	1.0	1.0
NasNetMobile	1.0	1.0	1.0	1.0
ResNet15V2	1.0	1.0	1.0	1.0

Table 5. Overall model's performance on X-ray test set.

Model	Performance			
	Accuracy	Precision	Recall	F1-Score
VGG16	0.97	0.98	0.97	0.97
InceptionResNetV2	0.97	0.98	0.97	0.97
ResNet50	0.64	0.79	0.64	0.58
DenseNet201	0.97	0.98	0.97	0.97
VGG19	0.91	0.93	0.91	0.91
MobileNetV2	0.97	0.97	0.97	0.97
NasNetMobile	1.0	1.0	1.0	1.0
ResNet15V2	0.99	0.99	0.99	0.99

Figure 10: Results from models implemented by study. NASNetMobile is indicated in the second-to-last row of each table.

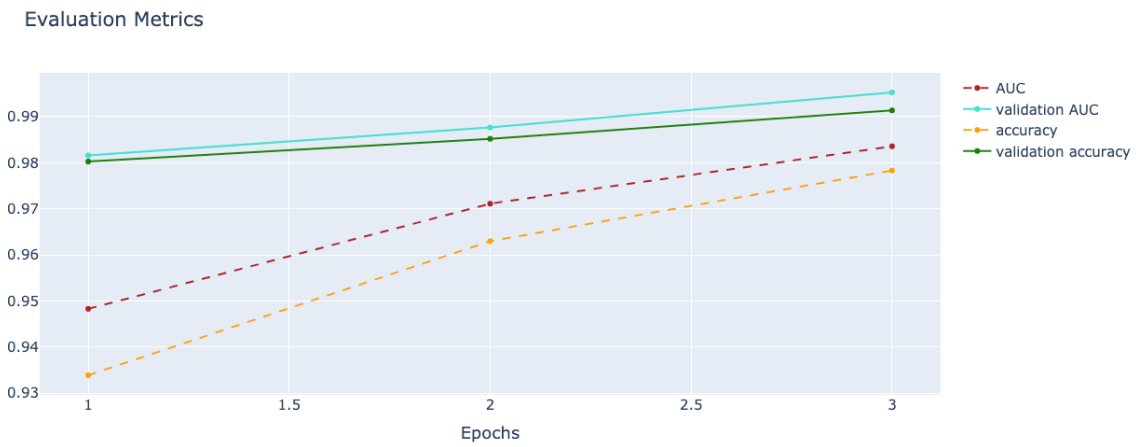

Figure 11: These are the evaluation metrics of the model while the model was training. "accuracy" and "AUC" represent training accuracy and AUC, respectively.

	precision	recall	f1-score	support
0.0	1.00	0.99	0.99	458
1.0	0.98	1.00	0.99	352
accuracy			0.99	810
macro avg	0.99	0.99	0.99	810
weighted avg	0.99	0.99	0.99	810

Figure 12: Classification report from model.

Predicted	0	1
Actual		
0.0	452	6
1.0	1	351

Figure 13: Confusion matrix from model.

It is clear that the results are strong. More training will lead to both a 100% training and test accuracy, but the model is deliberately kept at a lower accuracy because such model would *not* be a suitable final model of choice. The 100% accuracy can be reached by increasing the number of epochs.

Qualitatively, this model is on par with that implemented in the study. It would be useful for future research work to implement NASNetMobile so our results can be corroborated.

Conclusion

Our team leveraged transfer learning for our differing classification tasks to varying levels of success. The ResNet-50 model was built upon to classify between healthy and COVID-19 chest X-rays, achieving an accuracy of 75%. The EfficientNetB7 model achieved a validation accuracy of 87%-91% upon being trained to classify between healthy and pneumonia-infected chest X-rays. The VGG-19 model was implemented to classify between healthy, viral pneumonia-infected and bacterial pneumonia-infected chest X-rays, achieving an accuracy of 56%. The extended NASNetMobile model was tasked to classify between healthy and COVID chest X-rays, achieving a validation accuracy of 97%-99%.

Overfitting was a common issue with these models. This was evident when the validation accuracy of our models began to decrease within the first few epochs of training. In some cases, dropout layers were employed to prevent this with limited success. Our models would benefit from more training images, or improved data augmentation. This is a particularly difficult issue for COVID-19 classification tasks since the number of COVID-19 chest X-rays images that are available in the CoronaHack -Chest X-ray-Dataset (and others) is few.

Another related issue is the classifications provided by the CoronaHack-Chest X-ray-Dataset. Every image was labeled as healthy, viral pneumonia, bacterial pneumonia or stress-smoking pneumonia, however relatively few had secondary labels such as Streptococcus, SARS or COVID-19. Only using this dataset therefore limits the diversity of classification tasks that can attempted. To further extend the majority of the networks, the addition of a better balanced, larger dataset should be explored.

Bibliography

[1] E. Dong, H. Du and L. Gardner, "An interactive web-based dashboard to track COVID-19 in real time," *The Lancet,* vol. 20, no. 5, pp. 533-534, 2020.

[2] M.-Y. Ng, E. Y. P. Lee, J. Yang, X. Li, C. K.-M. Lui, F. Yang, X. Li, H. Wang, M. M.-s. Lui, C. S.-Y. Lo and Leung, "Imaging Profile of the COVID-19 Infection: Radiologic Findings and Literature Review," *Radiology: Thoracic Imaging,* vol. 2, no. 1, 2020.

[3] Praveen, "CoronaHack -Chest X-Ray-Dataset," Kaggle, [Online]. Available: https://www.kaggle.com/praveengovi/coronahack-chest-xraydataset.

[4] J. P. Cohen, P. Morrison, L. Dao, K. Roth, T. Q. Duong and M. Ghassemi, "COVID-19 Image Data Collection," [Online]. Available: https://github.com/ieee8023/covid-chestxray-dataset.

[5] J. W. Balagot, "Deep Learning on COVID-19," [Online]. Available: https://www.kaggle.com/dellectron/deep-learning-on-covid-19.

[6] R. Nambiar, " Image classification using EfficientnetB7," [Online]. Available: https://www.kaggle.com/rahulvv/image-classification-using-efficientnetb7.

[7] M. Elgendi, M. U. Nasir, Q. Tang, R. R. Fletcher, N. Howard, C. Menon, R. Ward, W. Parker and S. Nicalaou, "The Performance of Deep Neural Networks in Differentiating Chest X-Rays of COVID-19 Patients From Other Bacterial and Viral Pneumonias," *Fronitiers in Medicine,* vol. 7, 2020.

[8] W. Setiawan and F. Damayanti, "Layers Modification of Convolutional Neural Network for Pneumonia," *Journal of Physics: Conference Series,* vol. 1477, 2020.

[9] M. M. Ahsan, K. D. Gupta, M. M. Islam, S. Sen, M. L. Rahman and M. S. Hossain, "COVID-19 Symptoms Detection Based on NasNetMobile with Explainable AI Using Various Imaging Modalities," *Machine Learning and Knowledge Extraction,* vol. 2, no. 4, pp. 490-504, 2020.

[10] M.-Y. Ng, E. Y. P. Lee, J. Yang, F. Yang, X. Li, H. Wang, M. M.-s. Lui and C. S.-Y. Lo, "Imaging Profile of the COVID-19 Infection: Radiologic Findings and Literature Review," *Radiology: Cardiothoracic Imaging,* 2020.