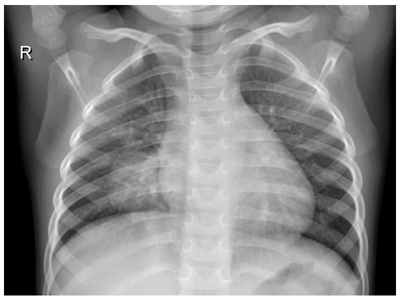
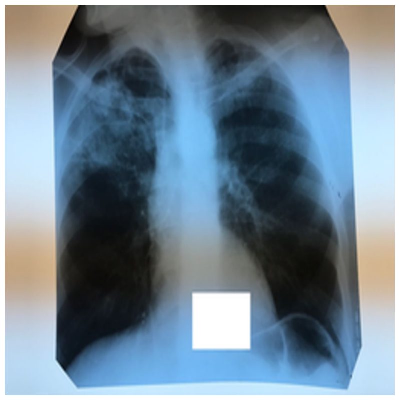
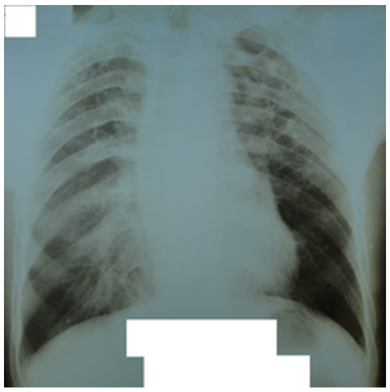
Classification of Chest X-Rays Using Transfer Learning

Introduction

The objective of this project is to classify images of chest x-rays into three classes: Normal, COVID-19, and Pneumonia. Classification was performed using transfer learning by implementing the Inception V3 and ResNet 101 V2 architectures.

Data Cleaning and Data Augmentation

The image data was pulled from multiple sources and compiled by Kaggle user, JPTIPTJ[1]. In addition to the classes mentioned above, the data also had a TUBERCULOSIS class of images. The TUBERCULOSIS images had far more distortions and inconsistencies than the images for the other classes. The below images show examples of the PNEUMONIA data and the TUBERCULOSIS data. The PNEUMONIA image is close to black and white and shows a great amount of detail. The TUBERCULOSIS data, on the other hand, has inconsistencies with cropping, with contrast, and with color balance. Most concerning, however, is that most images in this class have white boxes in diverse locations. This strongly biases the data in that a model may simply learn to classify any image with a white box as TUBERCULOSIS without learning the nuances of the x-ray itself.

"Pneumonia" class example:	"Tuberculosis" class examples:
	 

A great deal of preprocessing would need to be done specifically to images in the TUBERCULOSIS class in order to make them useful in a model. Therefore, I decided to focus the models instead on the other three classes.

The image data was already split into Test and Train datasets, with 730 images in Test and 5,676 images in Train. I imported the data and performed the following data augmentation:

[1] https://www.kaggle.com/datasets/jtiptj/chest-xray-pneumoniacovid19tuberculosis

- Rescale the images
- Random rotation of ≤ 5°
- Random shear intensity of ≤ 5°
- Random hight shift of ≤ 15%
- Random width shift of ≤ 15%
- Random zoom between 90% and 110%
- Random brightness adjustment between 80% and 120%
- Random horizontal flip
- Fill mode = ‘constant’

The classes in the training data are imbalanced with 460 in the COVID19 class, 1,341 in the NORMAL class, and 3,875 in the PNEUMONIA class. Therefore, I assigned weights to the three classes respectively 4.11, 1.41, and 0.49.

<u>MODEL #1: Inception V3</u>

The first framework I used for transfer learning was Inception V3. I imported the “imagenet” weights for the model and set *include_top* to “False” as I would be adding subsequent layers to the top. For the first round of training, I froze the layers from the base model so I could train only the final layers I would be adding myself.

Starting with the base model, I added the following layers to the end:

- **Global Average Pooling 2D** - A faster, more robust method than a standard fully-connected layer
- **Dropout (set to 50%)** - Used to reduce the potential for overfitting. Perhaps an unnecessary layer in this case given that Global Average Pooling already reduces overfitting.
- **Dense** - Final layer using a softmax classifier

To compile the model, I selected the “Adam” optimizer using the default 0.001 learning rate. The loss function used was categorical crossentropy. I also created a function to be able to track the f1 metric of each step of the model.

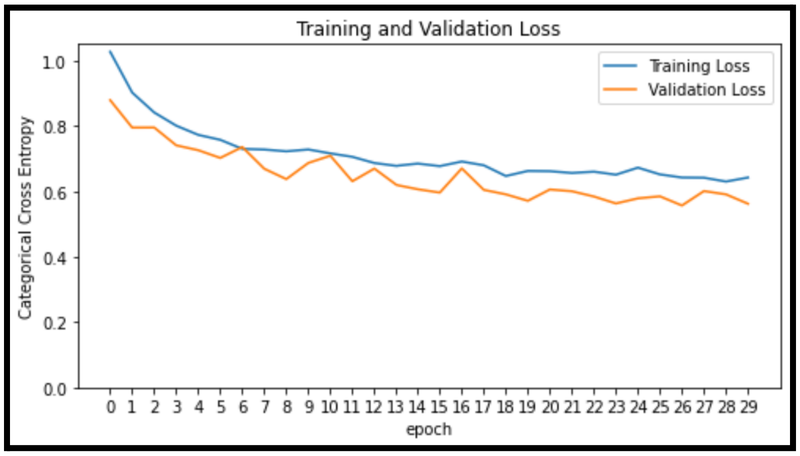
The model summary below indicates there are 21,802,784 parameters in the base Inception V3 model, and 6,147 trainable parameters in the final layer.

```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_2 (InputLayer)         [(None, 256, 256, 3)]     0         
_________________________________________________________________
tf_op_layer_RealDiv (TensorF [(None, 256, 256, 3)]     0         
_________________________________________________________________
```

```
tf_op_layer_Sub (TensorFlowO [(None, 256, 256, 3)]     0
_________________________________________________________________
inception_v3 (Functional)    (None, 6, 6, 2048)        21802784
_________________________________________________________________
global_average_pooling2d (Gl (None, 2048)              0
_________________________________________________________________
dropout (Dropout)            (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 3)                 6147
=================================================================
Total params: 21,808,931
Trainable params: 6,147
Non-trainable params: 21,802,784
_________________________________________________________________
```

I fit the model for 30 epochs and a batch size of 128. I also established an early stopping trigger when the validation loss function does not improve after five epochs. If early stopping is triggered, the model will use the weights from the epoch with the lowest validation loss.

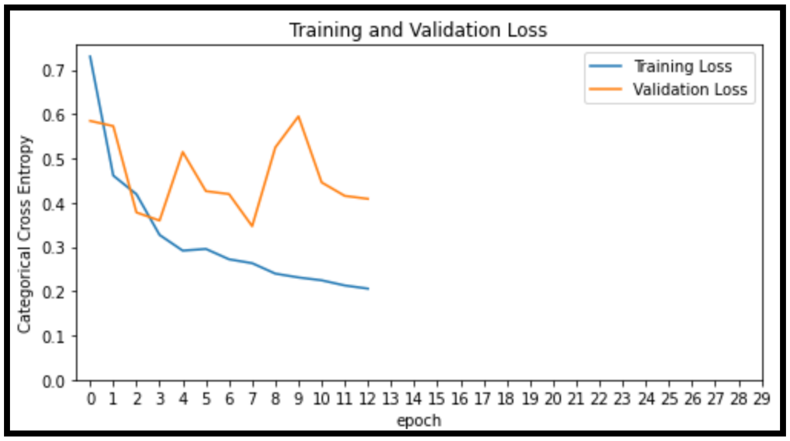
Once fitting was complete, this model achieved a validation loss of 0.5564 and a validation f1 score of 0.8045. The plot below shows the training and validation loss over the 30 epochs.



The plot indicates that training continued to improve throughout, though it flattened out in the end. It is worth noting that the training loss was consistently higher than the validation. This could potentially be due to the fact that the model was more challenged by the training data (where the images were augmented as detailed above) than it was by the test data where no augmentation was applied.

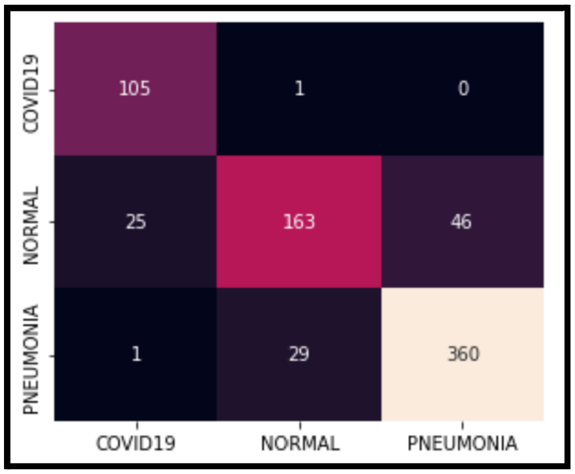
The next step was to fine-tune the model. I did so by unfreezing the layers of the base model. In addition I reduced the learning rate for the optimizer to 1x10^{-5}.

The plot below shows the training and validation loss of each epoch when fine-tuning the model.



We see that both training and validation loss decreases rapidly from the initial training, however by the fourth epoch the plots begin to diverge. The training loss continues to improve while the performance of the validation data varies greatly between each epoch. This indicates the model was most likely beginning to overfit to the training data. As the validation data stopped improving after epoch 7, the early stopping was triggered after epoch 12. Because early stopping was triggered, the model reverted the weights back to those from the step in the training which had the lowest validation loss. This best validation loss is 0.3468, and its corresponding f1 score is 0.8569.

Below shows the confusion matrix from the final results of the fine-tuning step.



	COVID19	NORMAL	PNEUMONIA
COVID19	105	1	0
NORMAL	25	163	46
PNEUMONIA	1	29	360

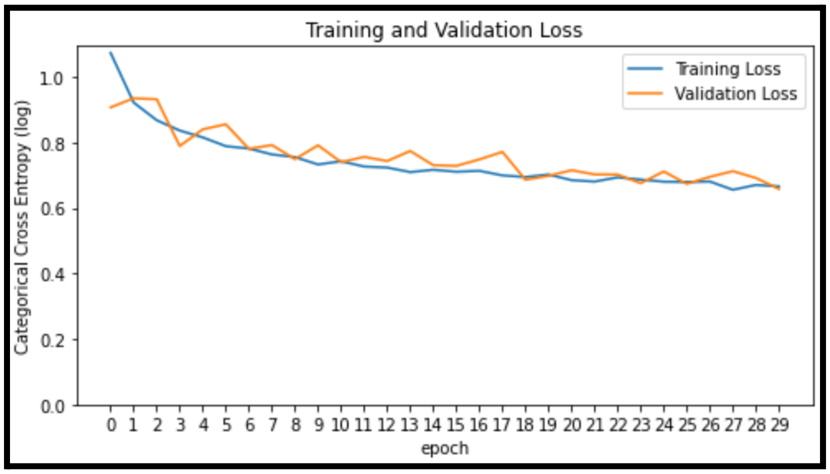
We see that all but one (or 99.1%) of the COVID19 records were classified correctly. For the PNEUMONIA class 92.3% of the records were correctly classified. Most of the incorrectly classified records were wrongly predicted to be NORMAL instead of PNEUMONIA. The NORMAL class was only properly predicted 69.7% of the time. Taken together, these results indicate that the model as a diagnostic tool is more likely to return false-positives for illnesses than false-negatives.

MODEL #2: ResNet101 V2

The next framework used for transfer learning was ResNet101 V2. I set up the model in the same manner as the Inception V3 model with a Global Average Pooling layer and a Dropout layer. The summary below indicates 42,626,560 parameters in the base model and 6,147 trainable parameters in the final layer.

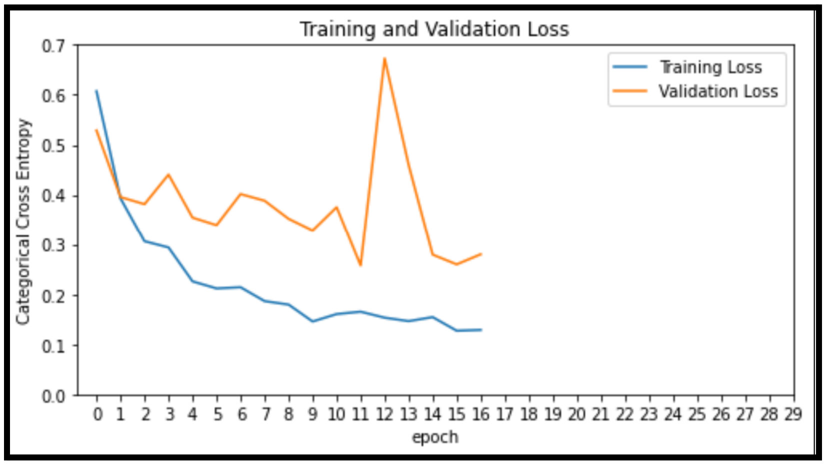
```
Model: "functional_19"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_20 (InputLayer)        [(None, 256, 256, 3)]     0
_________________________________________________________________
tf_op_layer_RealDiv_6 (Tenso [(None, 256, 256, 3)]     0
_________________________________________________________________
tf_op_layer_Sub_6 (TensorFlo [(None, 256, 256, 3)]     0
_________________________________________________________________
resnet101v2 (Functional)     (None, 8, 8, 2048)        42626560
_________________________________________________________________
global_average_pooling2d_7 ( (None, 2048)              0
_________________________________________________________________
dropout_10 (Dropout)         (None, 2048)              0
_________________________________________________________________
dense_10 (Dense)             (None, 3)                 6147
=================================================================
Total params: 42,632,707
Trainable params: 6,147
Non-trainable params: 42,626,560
_________________________________________________________________
```

As with the Inception V3 model, I used the Adam optimizer. The learning resulted in a validation loss of 0.6577 and an f1 score of 0.7593. The plot below shows the progress of the loss functions throughout the training.



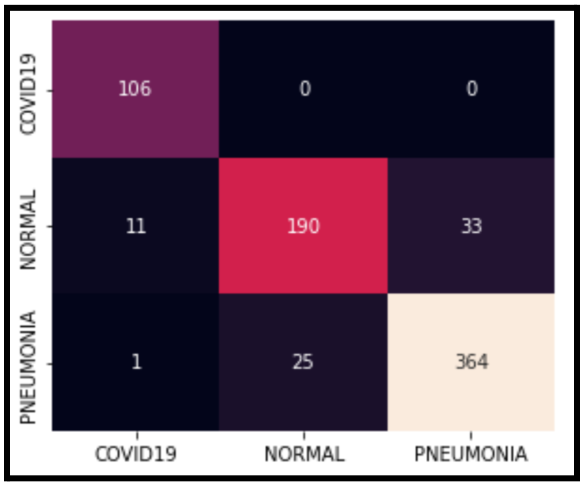
We see that the both loss functions had a larger decrease in the first five epochs than in subsequent ones, however neither plot appears to flatten out indicating that additional epochs may provide an improvement over what was achieved in 30 epochs.

The model was then fine-tuned using the same methodology as with Inception V3.



The history chart shows early stopping was triggered. Both training and validation loss continued to decrease until epoch 12 when validation loss had a sharp increase. In subsequent epochs validation loss never fell back to the score it achieved in epoch 11, which indicates the training got to a point where it was overfitting to the training data. The best score achieved was a validation loss of 0.2576 and an f1 score of 0.9052.

The confusion matrix of the final model is shown below.



	COVID19	NORMAL	PNEUMONIA
COVID19	106	0	0
NORMAL	11	190	33
PNEUMONIA	1	25	364

It shows the model predicted every true positive for the COVID19 class correctly. It predicted 93.3% of the true positives for PNEUMONIA correctly. For the NORMAL class, 81.2% were predicted correctly. These results indicate that the model is more likely to predict false-positives for COVID19 and PNEUMONIA rather than to miss true positives for either. Again, in a clinical setting, one would prefer to have the errors skew more towards false positives than false negatives.

Conclusion

Comparing the performance of the two models, the ResNet101 V2 model had the lowest validation loss and the highest f1 score, though the performance Inception V3 was only slightly worse than that of ResNet101 V2. The ResNet model had recall scores of 100% and 93.3% for the COVID19 and PNEUMONIA classes respectively, indicating a low likelihood of the model predicting a false-negative of either class. The precision of the predictions was 89.8% and 91.7% respectively. This indicates that there is roughly a 10% likelihood of returning false-positive predictions for both classes.

In a clinical setting, one would want to err on the side of providing a diagnosis in instances where the patient is normal than the side of not providing a diagnosis when the patient has a malady such as the classes in these models. Therefore we would give more weight to false-negatives than false-positives in assessing how much error we are comfortable with. To that end, this model may be quite useful for diagnosing COVID19, but the false-negatives associated with the PNEUMONIA class may need to be reduced in order to be used for that diagnosis.