Classifying Pneumonia in Chest X-Rays of COVID-19 Patients

By Andrew Avola, Kristen Cocciante, Amit Deb, and Joseph Henderson

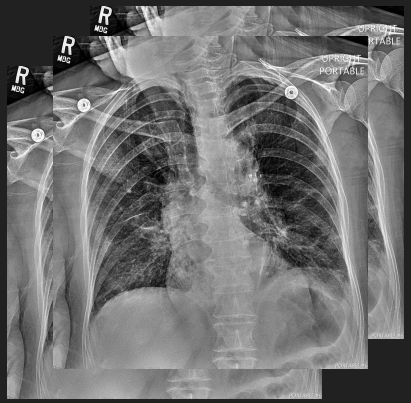
Our Project

- We have developed a model with a 92% accuracy with the purpose of assisting doctors in identifying pneumonia is the lungs of patients with COVID-19
- Through preprocessing, the inputted images are resized, eliminating unnecessary detail. This allows the model to more accurately classify abnormal levels of opacity which may indicate pneumonia

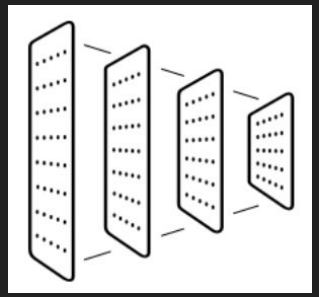
Our goal

- Balance out the data in order to develop a suitable training set and avoid patient bias and data imbalance
- Develop a model which accurately classifies the chest X-Rays of COVID-19 patients into one of four categories of pneumonia

Input: Images of Chest X-rays



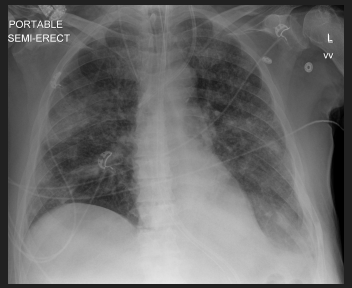
CNN: InceptionV3, plus our own set of layers

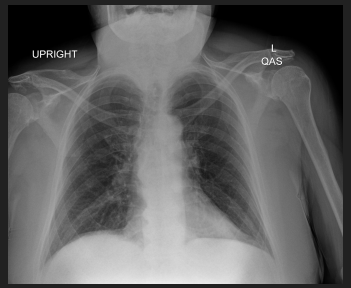


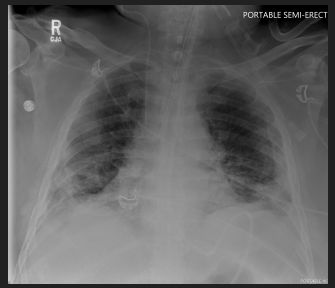
Categories
Typical
Atypical
Indeterminate
Negative

Data Categories

- Typical: lungs containing the typical signs for pneumonia, which are certain levels of opacity in certain lobes of the lungs
- Atypical: lungs containing other abnormalities other than the typical signs for pneumonia, e.g. pulmonary edema, lobar consolidation, etc
- Indeterminate: lungs which do not contain the typical signs for pneumonia but do contain certain opacities indicative of airspace disease
- Negative: lungs which do not contain pneumonia

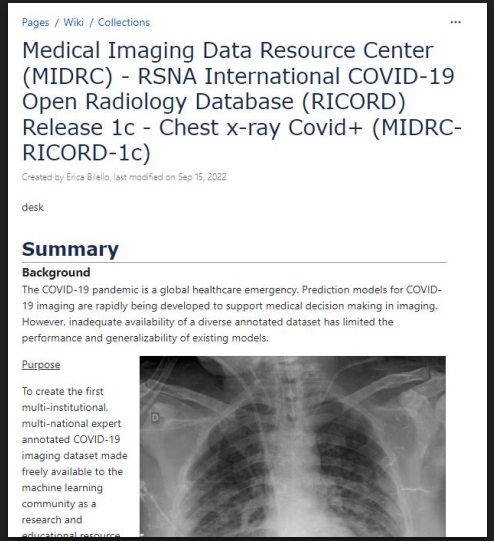






Data Source: MIDRC-RICORD Radiology Database

- Data sourced from Cancer Imaging Archive
- Released in an anonymized form
  - Maintains patient confidentiality
  - Only information we have access to is the patient's gender and age at the time of the scan



Pages / Wiki / Collections

Medical Imaging Data Resource Center (MIDRC) - RSNA International COVID-19 Open Radiology Database (RICORD) Release 1c - Chest x-ray Covid+ (MIDRC-RICORD-1c)

Created by Erica Bilello, last modified on Sep 15, 2022

desk

Summary

Background

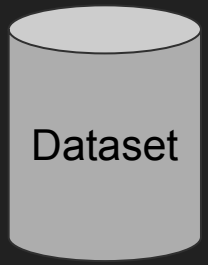
The COVID-19 pandemic is a global healthcare emergency. Prediction models for COVID-19 imaging are rapidly being developed to support medical decision making in imaging. However, inadequate availability of a diverse annotated dataset has limited the performance and generalizability of existing models.

Purpose

To create the first multi-institutional, multi-national expert annotated COVID-19 imaging dataset made freely available to the machine learning community as a research and

Details:
Name - [redacted]
Residence - [redacted]
Age - 59
Gender - Male

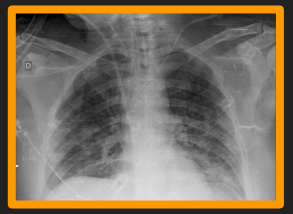
Parsing through Anonymized Data



Clinical Data

Annotations

Scan is sourced from dataset.



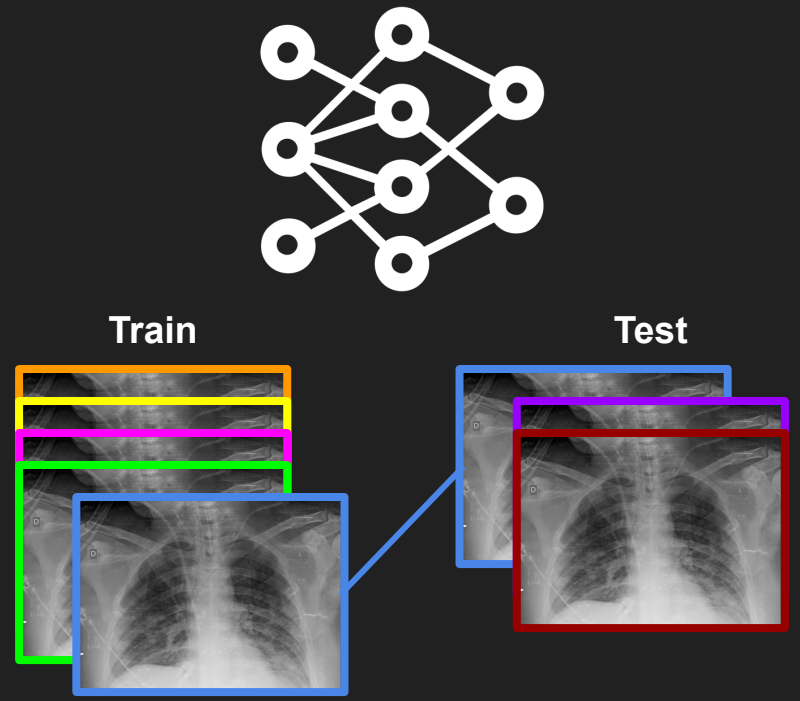
Scan is looked up in the CD file and assigned a category.

Scan is cross compared with the annotations and associated with a patient.

- Two accompanying data files
  - Clinical data file which associates scans with a category
  - Annotations file which associates scans with a given patient
- The file names of scans create correspondences between the scan and the files, allowing us to cross reference the scan between each file

The Memorization Problem: Initial Approach

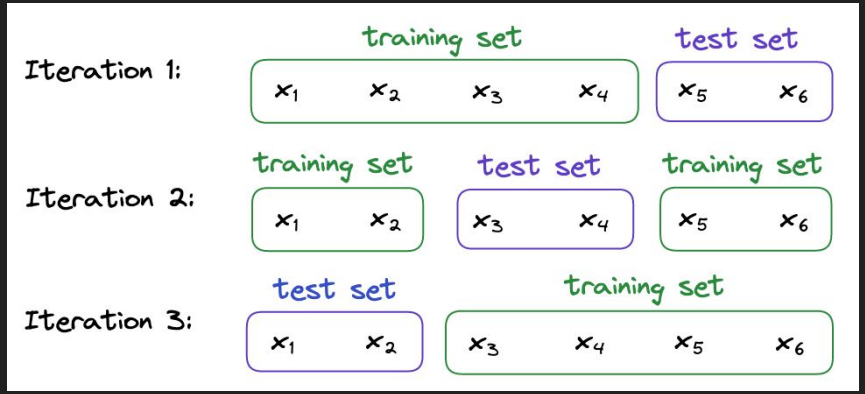
- Initial approach was to pool scans and associate them with their labels before the 75/25 split, disregarding their associations with the patient
- This caused the same patient's scan to appear in both the train and test set
    - Causes the model to MEMORIZE the appearance of the patient instead of meaningfully recognizing features



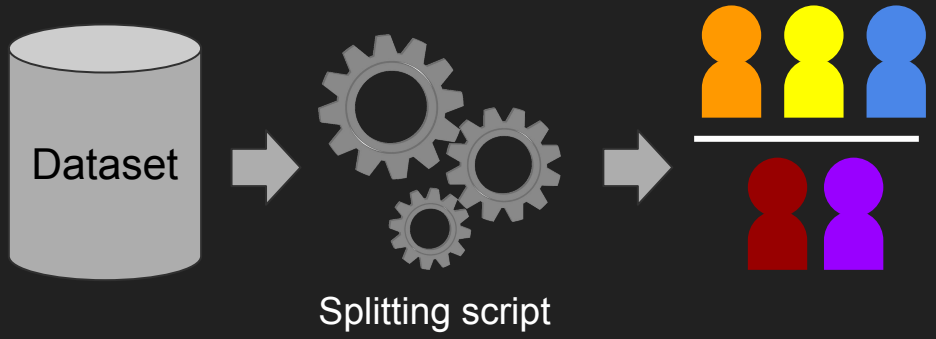
Here the same patient's scans are split across both the train and test sets. This will cause the model to simply memorize the patient's appearance.

The Memorization Problem: Leave-Several-Out

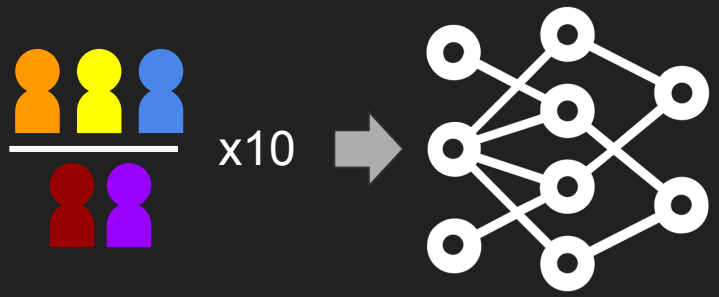
- To ensure that our model does not memorize patients, a more delicate split is required
- Grouping the scans by patient and then perform our train/test split
  - This ensures that no patient will have scans in both the train and test set
- We can perform this splitting process multiple times and train our model on a different split each iteration



The Memorization Problem: Splitting Across Patients

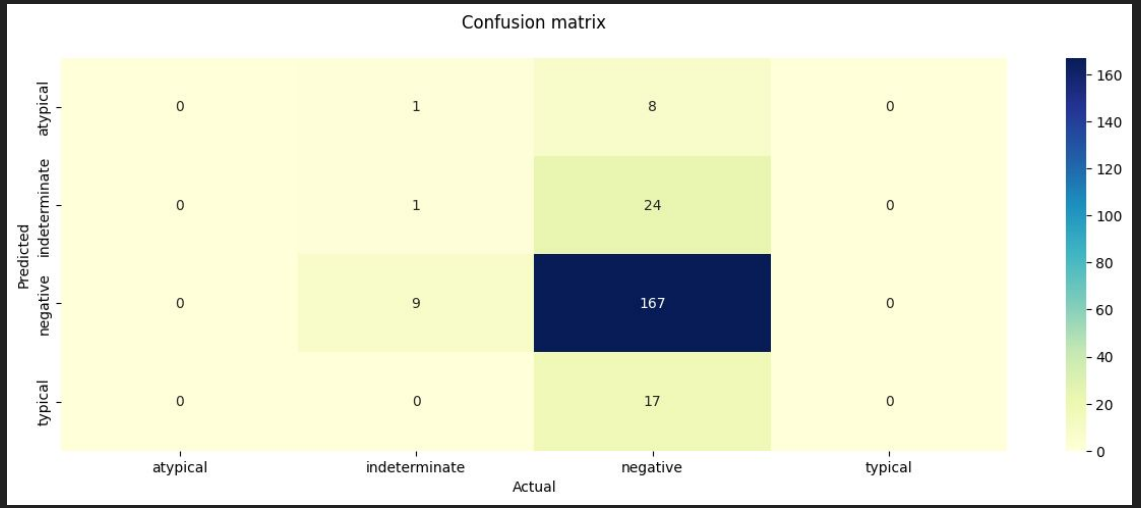


- Splitting script generates a 75/25 split of patients
  - Ensures variability - each split is different
- 10 splits are created and then utilized in the training process via the Leave-k-out technique



Data Imbalance: The Problem

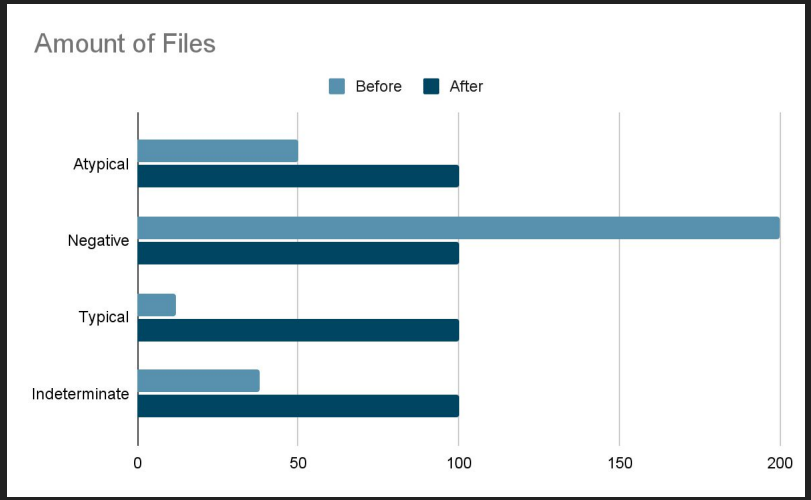
- Another issue we encountered with the data
  - Very high imbalance of negative scans compared to other 3 categories
- Without addressing this imbalance, the model was more likely to falsely predict a scan as negative



An overabundance of negative scans in the initial dataset led to our model predicting almost all scans as negative. The model associated with the above matrix had an accuracy of about 75%, but only because nearly ¾ of the scans in the data set are negative and so the accuracy value was not high, but just a byproduct of the guessing patterns of the model.

Data Imbalance: The Goal

- We have to decide how to normalize
  the amount of data given to us.
  - Double files in the smallest category
  - Make the other categories file amount to the doubled smallest category amount



Example of what might happen to each batch of images after the data imbalance is corrected

Data Imbalance: Filling

In order to fill the categories to match the amount of files to the doubled category we need to fill.

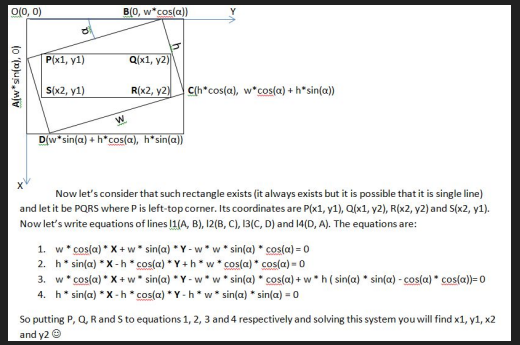
There are two ideas to fill data:

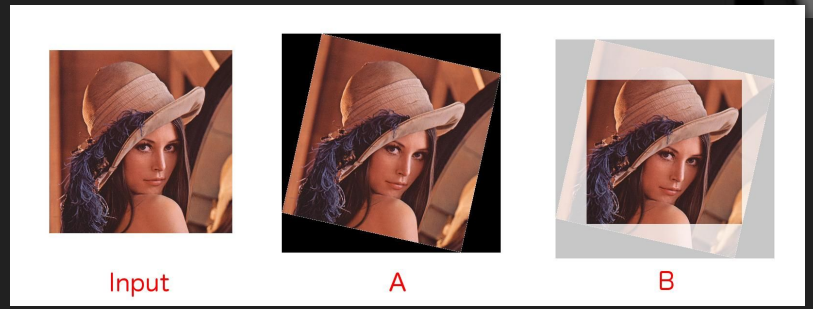
- Clone a file in a category until we reach the normalized amount
  - We run into the issue of having bias because we would train on double the amount of the same images
- Clone and augment the file in until we reach the normalized amount
  - We do augmentation by rotating the image slightly by -4 to 4 degrees to simulate a doctor perhaps scanning in a xray slight off or a patient not being centered

Data Imbalance: Remove Margins From Rotated Images

In order to remove these margins for the rotated image

We need perform some several operations to achieve this result, including calculating the bounds and scaling the result to achieve the max area of the image[1][2]





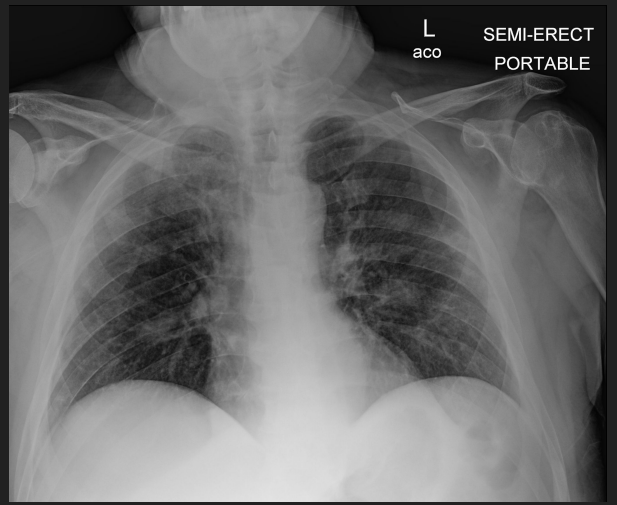
Data Imbalance: Culling

The categories we cull are categories whose file amounts are greater than the normalized amount. We randomly cull files in those categories until we reach the normalized amount. This is so that we don't have so much bias in our training set.

Pre-Preprocessing the Data for the Model

We need to make sure these images are ready for the model to accept

- Images need to be unsigned integer 8 bit values
- Images need to have pixel values between 0 - 255
- Images need have a color channel
- Images need to be 299 pixels by 299 pixels



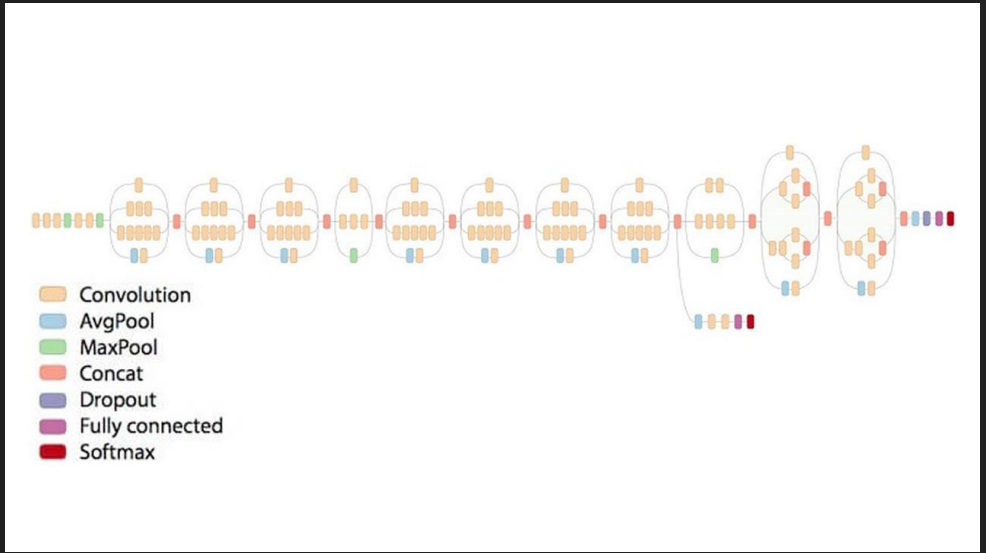
Efficiency of Loading Images

Loading through this script takes about a hour to complete. There are two ways to run it:

- With the model script
  - The problem with this script is that if the model needs to be adjusted, we would need to run the whole script again.
- Loading the images into a np files
  - The images will be converted to a np array then appended to another np array then saved to a np file. This process will take an hour to complete
  - The model can load the file in less than a second. Thus having a very elegant way to have the images loaded in very fast.

Transfer Learning

- Initial thoughts were to find a pre-trained transfer learning model for image classification
- We chose InceptionV3 which has attained greater than 78.1% accuracy in 170 epochs

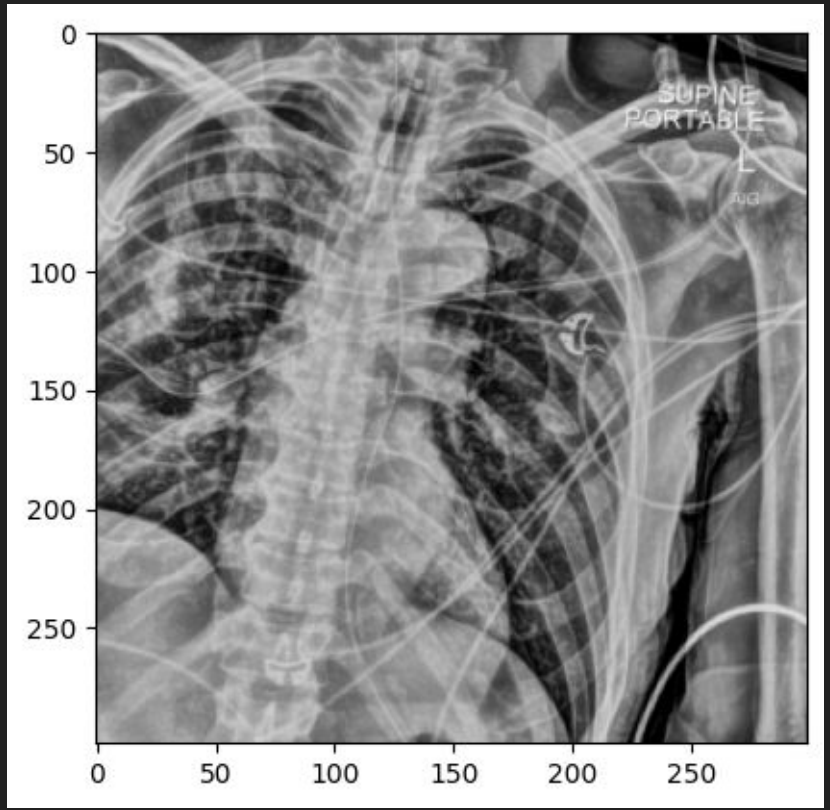


GPU Compatibility

- Tensorflow has the ability to allow GPU compatibility for high intensity computations
- We installed Tensorflow GPU and downloaded proper drivers for our Tensorflow version
- Computation speed increased from 3 minute epochs to 4 second epochs with a RTX 2080 Super

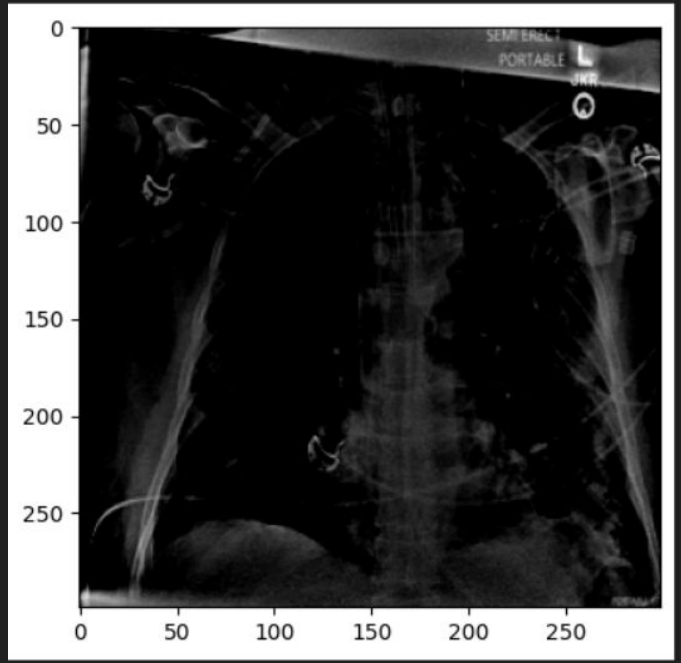
Obtain Proper Splits of Data

- Needed 10 splits of training and testing data
- Data needed to be 299x299 pixels for the InceptionV3 pre-processing pipeline
- 1 Hour of computation time per split of data being obtained
- Stored as numpy arrays to drastically speed up image loading time into Jupyter Notebook



InceptionV3 Pre-processing

- Using InceptionV3's pre-processing functions will allow us to make sure that the images in our arrays are all of the same size
- Model pre-processing may shorten model training time and also speed up model inference



First Run-through

- Passing our first set of data through the model before any obtained weights gave us around 50% validation accuracy
- After all 10 data sets we obtained around 88% validation accuracy

```
lr: 1.0000e-04
Epoch 7/50
...
23/24 [==========================>..] - ETA: 0s - loss: 0.2776 - accuracy: 0.9130
Epoch 12: val_accuracy did not improve from 0.90449
24/24 [==============================] - 2s 85ms/step - loss: 0.2749 - accuracy: 0.9149 - val_loss: 0.3764 - val_accuracy: 0.8708 -
lr: 1.0000e-04
Epoch 12: early stopping
```

Keras Tuner

- We used Keras Tuner to test our model for optimization, looking for our highest validation accuracy
- Keras Tuner allowed us to find different hyper parameters in a specified range for the units in our dense layers
- On our first of 10 sets of data, we obtained a validation accuracy of ~57%

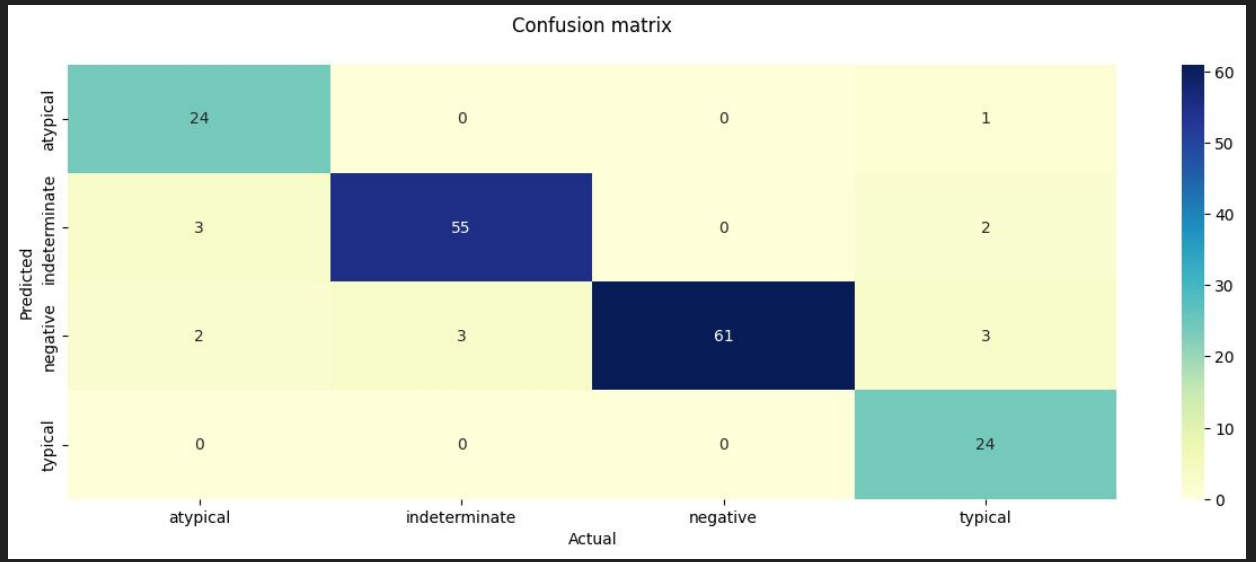
```
Best val_accuracy So Far: 0.5730336904525757
Total elapsed time: 00h 19m 32s
INFO:tensorflow:Oracle triggered exit
```

```
Dense(hp.Int("input_units1", min_value = 32, max_value = 1024, step = 32), activation = 'relu', name = 'Dense_1'),
Dropout(0.2),
Dense(hp.Int("input_units2", min_value = 32, max_value = 512, step = 32), activation = 'relu', name = 'Dense_2'),
Dropout(0.2),
Dense(hp.Int("input_units3", min_value = 16, max_value = 64, step = 16), activation = 'relu', name = 'Dense_3'),
Dropout(0.2),
Dense(4, activation = 'softmax')
```

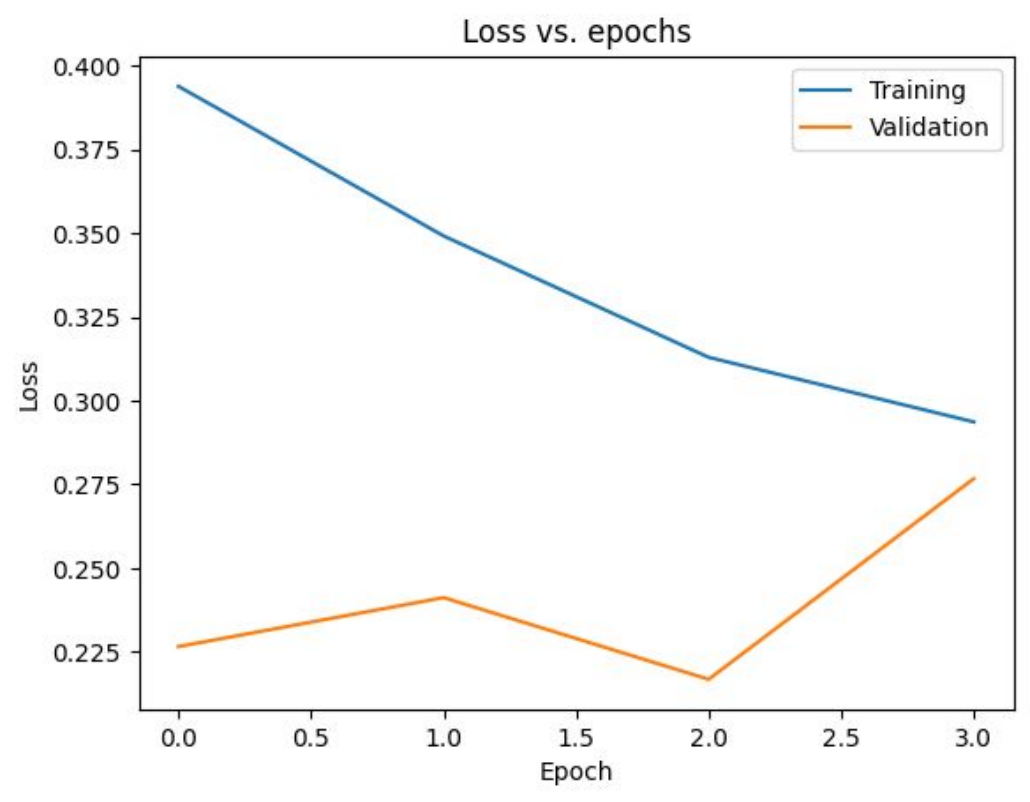
Final Run-through After Optimization

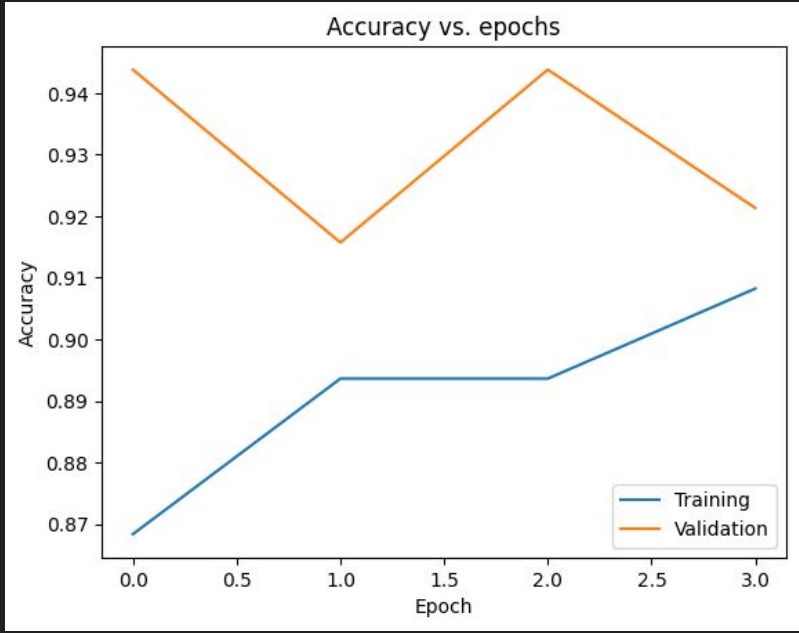
- After Keras Tuner optimizations, we ran all 10 sets of data through the model again for re-training
- Obtained 92% test accuracy and 96.5% training accuracy

```
Test:
accuracy: 0.921
Train:
accuracy: 0.965
```



Graphs After Optimization





Sources

Cancer Imaging Archive COVID Pneumonia Dataset

K-Fold Cross Validation

Math Behind Rotating Images and Getting Max Area

Rotating Images and Getting Max Area