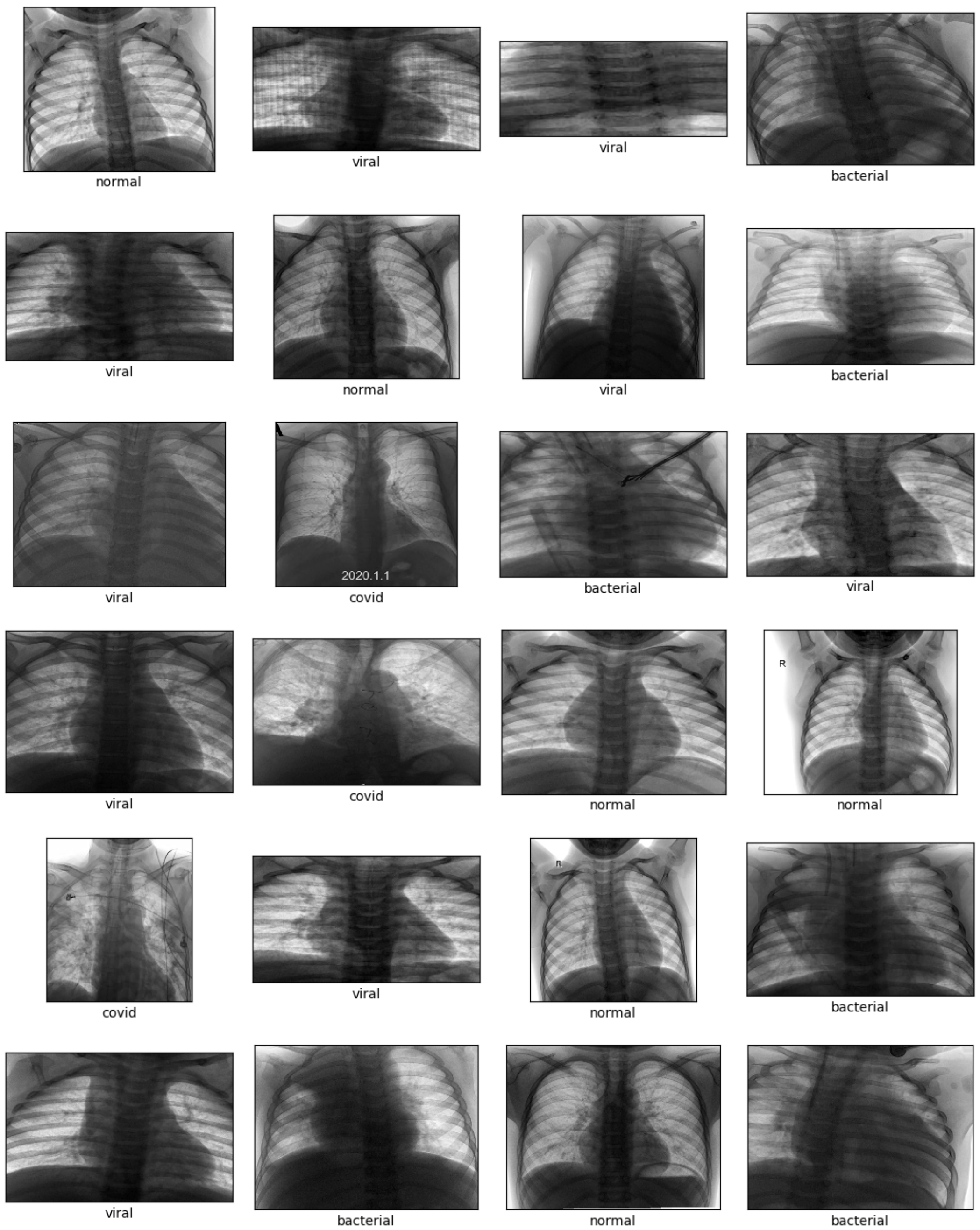
# COMS 4771 COVID-19 Kaggle Competition Writeup

Lawrence Chillrud (lgc2139)

lgc2139@columbia.edu

May 9, 2020



# Contents

# 1 Exploring the dataset

## 1.1 Initial Exploration

As the Kaggle competition notes [1],

> “The training dataset includes 1127 chest x-rays drawn from several different sources (of varying size and quality) and a set of multiclass labels indicating whether each patient was healthy or diagnosed with bacterial pneumonia, viral pneumonia, or COVID-19. The test data includes 484 images without labels, for which you will predict a diagnosis.”

I began by examining a few of the images from both the training and test datasets:

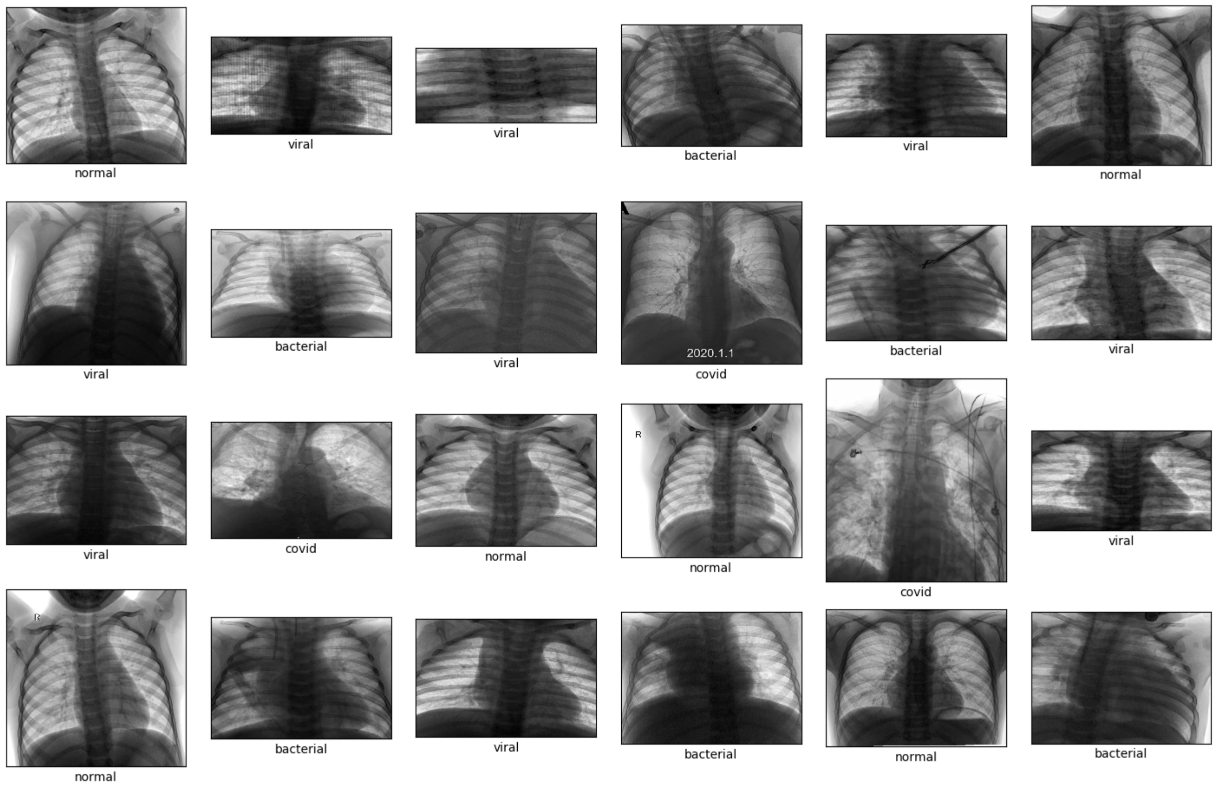


**Figure 1:** Sample training images (with labels).

The first 24 images from the training dataset, with their accompanying labels.

The x-rays from the test dataset look much the same. The biggest takeaway I got was that I would need to standardize the image size for each x-ray at some point during pre-processing. From here, I looked at the class breakdown of the training data, finding that x-rays of patients with COVID-19 made up a little less than 7% of the training dataset (77 of the 1127 training x-rays):

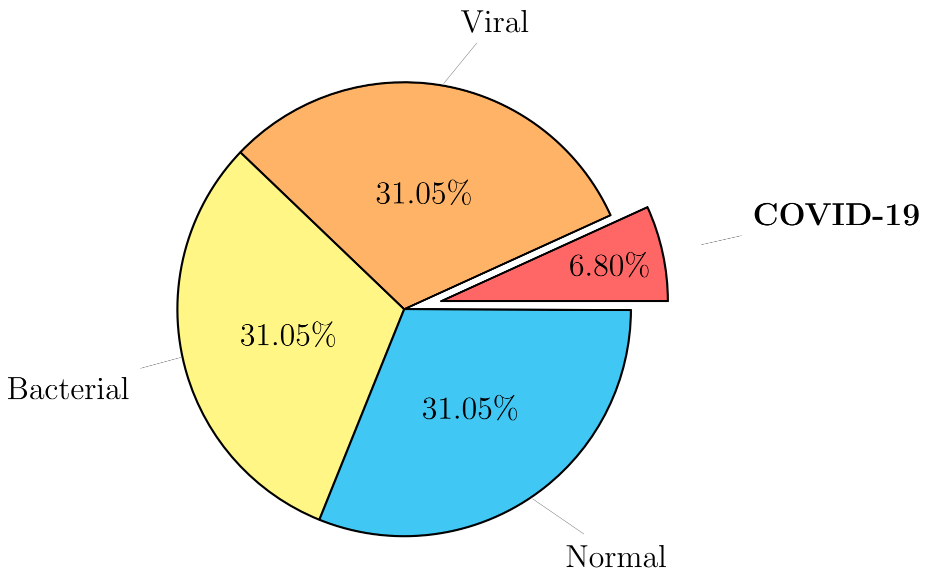
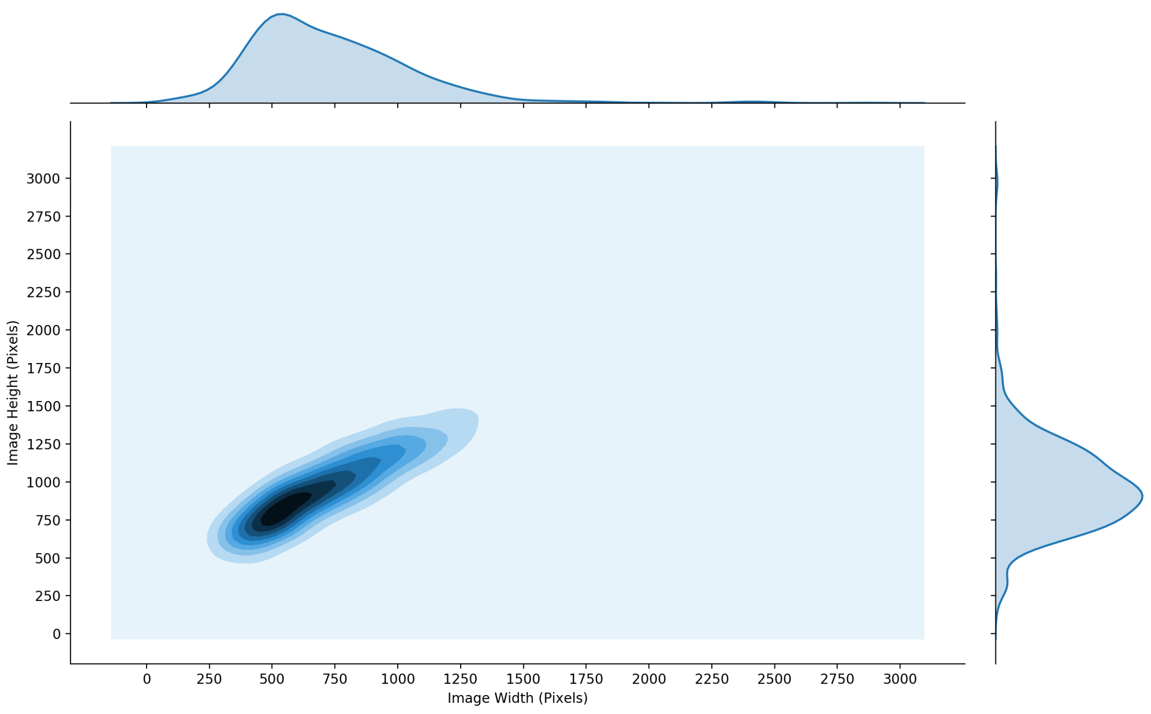


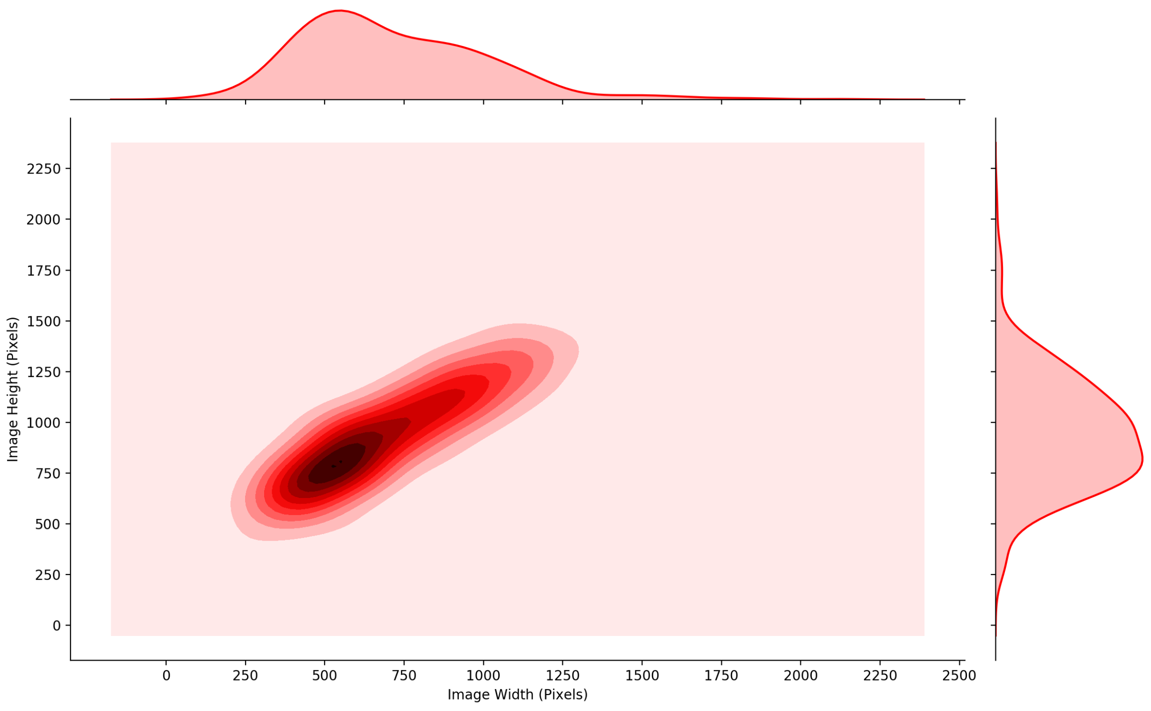
**Figure 2:** Class breakdown of the training data.

The viral, bacterial, and normal (healthy) classes of x-rays each had 350 images in the training data, while the covid class only offered 77 images for training.

As we saw in Fig. 1, the image size varied quite a bit within training dataset. The same is true of the test dataset. In anticipation of standardizing the image size during pre-processing, I decided I would also look at the distribution of image sizes in each dataset, given in Fig. 3:



**(a)** Training dataset



**(b)** Testing dataset

**Figure 3:** Distribution of image sizes.

## 2 Pre-processing

### 2.1 Initial pre-processing

Looking at Fig. 3, I determined that resizing all of the images in the training and test data to 512×512 made sense. The images shown in Fig. 1 have been displayed again in Fig. 4 after resizing to 512×512. I also normalized all of the pixel values to be floating points between 0 and 1, rather than integers between 0 and 255. This was done to make the maths for the neural network a little more straight forward.

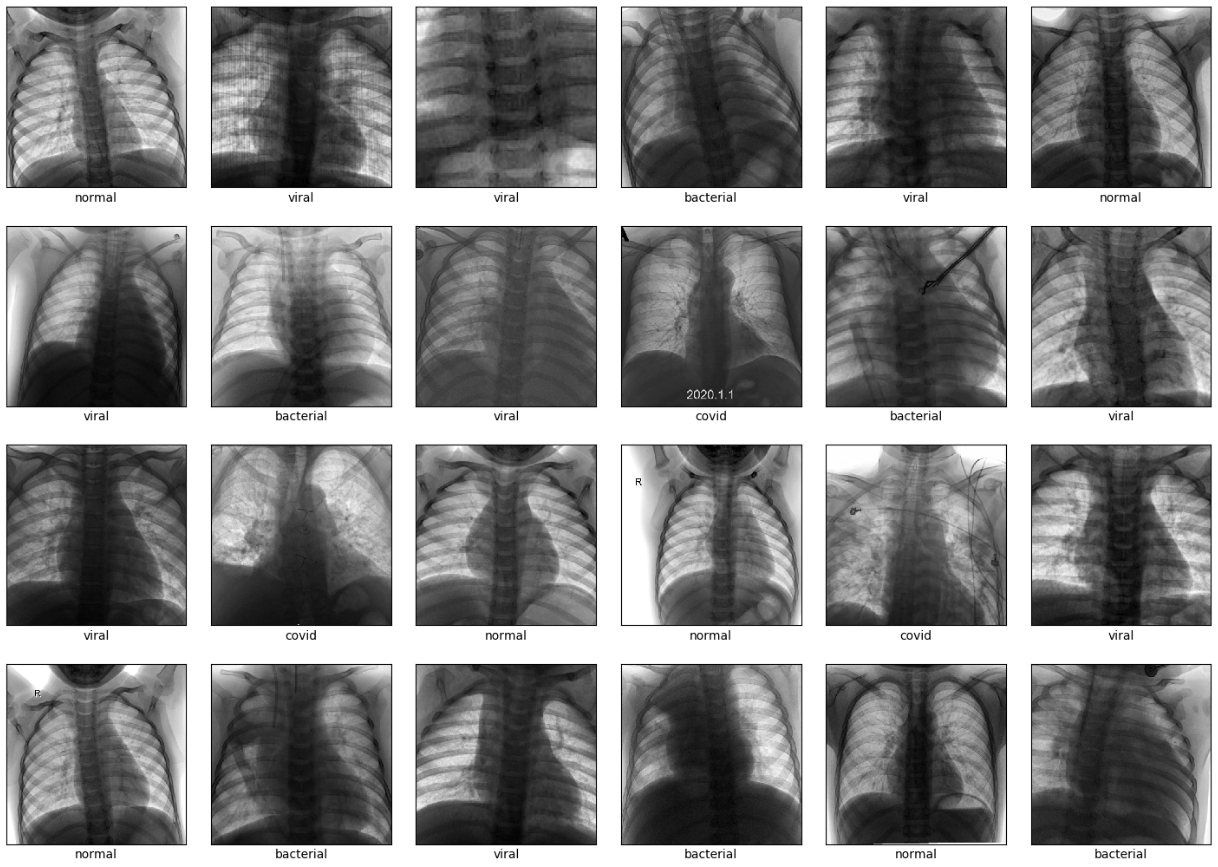


**Figure 4:** Resized training images (with labels).

The first 24 images from the training dataset, resized to 512×512 pixels, with the pixel values normalized to be floats between 0 and 1.

## 3 Algorithms

To start the project, I first looked through Google Scholar for some papers on related problems, and stumbled upon Rajpurkar et. al.'s paper, "CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning" [3]. They used a 121-layer Dense Convolutional Neural Network, trained on the ChestX-ray 14 dataset. The ChestX-ray 14 dataset contains 112,120 frontal-view x-ray images of 30,805 unique patients. They also utilized transfer learning, initializing the weights of their network with the weights from a model pretrained on the ImageNet database. Another paper implemented transfer learning with the ImageNet database as well: Kermany et. al.'s "Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning" [2]. Below is the graphical abstract from their paper:

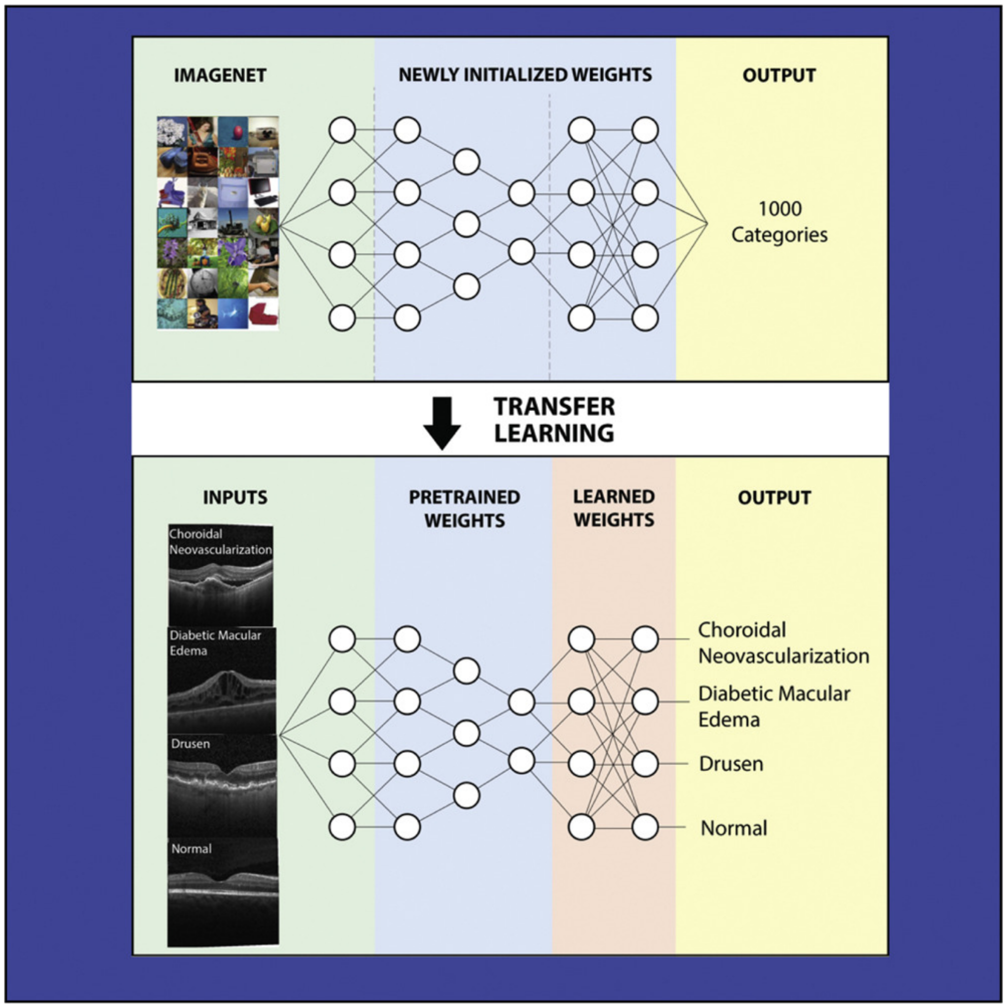


**Figure 5:** The graphical abstract from Kermany et. al.'s "Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning" [2].

## 3.1 CNN v.1.0

Inspired by the CheXNet paper and Kermany et. al.'s paper, I began by writing a fairly vanilla convolutional neural network (CNN) in Python using the TensorFlow package, with the idea of eventually upgrading the base model I wrote via transfer learning, as done in both papers. The first model I wrote had the following architecture and parameters:

| Layer | Neurons | Activation function |
|---|---|---|
| Convolutional Layer 1: | 32 convolutions, (3, 3) | ReLU |
| Max Pooling Layer 1: | (2, 2) | |
| Convolutional Layer 2: | 64 convolutions, (3, 3) | ReLU |
| Max Pooling Layer 2: | (2, 2) | |
| Flatten | | |
| Dense Layer 1: | 128 | ReLU |
| Dense Layer 2: | 4 | Softmax |

**Table 1:** CNN v.1.0 architecture

| General Parameters | |
|---|---|
| Image size: | (512, 512) |
| Batch size: | 32 |
| Optimizer: | Adam |
| Loss function: | Categorical cross entropy (sparse) |

**Table 2:** CNN v.1.0 parameters

These will serve as the parameters for all of the models going forward, unless otherwise noted.

The model as a whole has 130,075,652 total parameters (weights), all of which are trainable (i.e. must be learned). Training time took about 22 minutes and 30 seconds (for 5 epochs). The training loss and accuracy for each epoch are reported below in Table 3, along with the score on the Kaggle competition's public leaderboard:

| Epoch | Loss | Accuracy |
|---|---|---|
| 1 | 5.6336 | 0.3487 |
| 2 | 1.0010 | 0.5563 |
| 3 | 0.7642 | 0.6575 |
| 4 | 0.6404 | 0.7116 |
| 5 | 0.5475 | 0.7587 |
| Test (submission) | n/a | score = 0.56209 |

**Table 3:** CNN v.1.0 training loss, accuracy, and Kaggle public leaderboard score.

Training and testing this model yielded the following notes: Firstly, the time to train took too long considering this is simply a bare-bones model. This is largely due to the fact that the x-ray images were of a fairly large size (512×512), driving up the number of parameters that needed to be trained. Secondly, while I had a hunch that the model could be overfitting, there was no real way to be sure without plotting the loss of both the training and test sets. Without any given labels for the test set, I decided that a validation set should be made out of a fraction of the training set for the next model. Next, I examined the class breakdown of the model's predictions on the test data, given in Fig. 6 below:

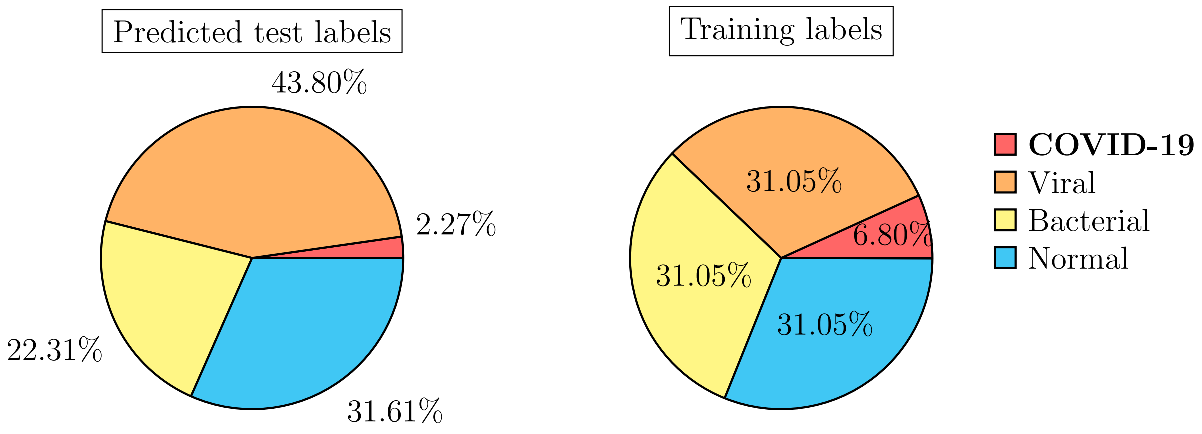


**Figure 6:** Left: CNN v.1.0 breakdown of predicted labels on the test dataset. Right: Fig. 2's breakdown of labels in the training dataset.

Of the 484 test x-ray images, the CNN v.1.0 model only predicted COVID-19 eleven times, while it predicted viral pneumonia 212 times, bacterial pneumonia 108 times, and healthy 153 times. This deviates significantly from the distribution we see present in the training dataset.

Finally, this model varied significantly in its predictions each time it was trained anew. On a completely different training run, separate from the results described above in Table 3 and Fig. 6, the model managed to end epoch 5 with about 83% training accuracy, and scored 0.6666 on the Kaggle public leaderboard. Interestingly, the predictions that resulted in that Kaggle score (the high for this model) saw a huge increase in the COVID-19 class share (where 11% of images were predicted as COVID-19). This increase helps account for the higher Kaggle score, since the metric used on Kaggle gives heavy preference for classifying COVID-19 x-rays correctly over the other classes. But, similarly to the run described in above in Table 3 and Fig. 6, these predictions struggled to reflect the training data's class breakdown; for example, bacterial pneumonia was only predicted 15% of the time, when it comprises nearly a third of the training data. Therefore, work needs to be done to improve the model's consistency, as well as its reflection of the training data (assuming the distribution of training data reflects that of the test data).

## 3.2 CNN v.1.1

Taking into account all of the findings from CNN v.1.0 gave rise to the following changes:

1. To help with training time, I resized all images to 64×64 pixels, rather than the original 512×512. Once things are working better, the image size can be brought back up to 512×512 to try and get more information out of each x-ray.
2. The training data was divided into two datasets: (1) A slightly smaller training set, composed of the first 80% of the x-ray images in the original training data, giving 901 images total. (2) A validation set, consisting of the last 20% of the x-ray images in the original training data, giving 226 images. These two datasets were then used to monitor potential overfitting.
3. I imposed a kind of class weighting, in which the loss function is weighted differently during training depending on what class of image the neural network is training on at a given time. Essentially, this is to tell my model to "pay more attention" to samples from the under-represented COVID-19 class. The class weighting is as follows: covid: 5, viral: 1, bacterial: 1, normal: 1 (i.e. covid training samples are weighted 5 times more than samples from other classes in the loss function calculations during training).
4. I also shuffled the training data before every epoch to try and help the consistency / reproducibility of the model.

Having made all of these changes, the model as a whole has 1,625,092 total parameters (weights), all of which are trainable (i.e. must be learned). Training time took 2 minutes and 43 seconds for 50 epochs, a drastic improvement from the prior model; CNN v.1.1 was a little less than ten times faster to run ten times as many epochs, i.e. CNN v.1.1 was slightly less than 100 times faster than CNN v.1.0 per epoch. The training loss and accuracy along with the validation loss and accuracy for each epoch are reported below, along with the score on the Kaggle competition's public leaderboard at different checkpoints along the model's 50 epochs:

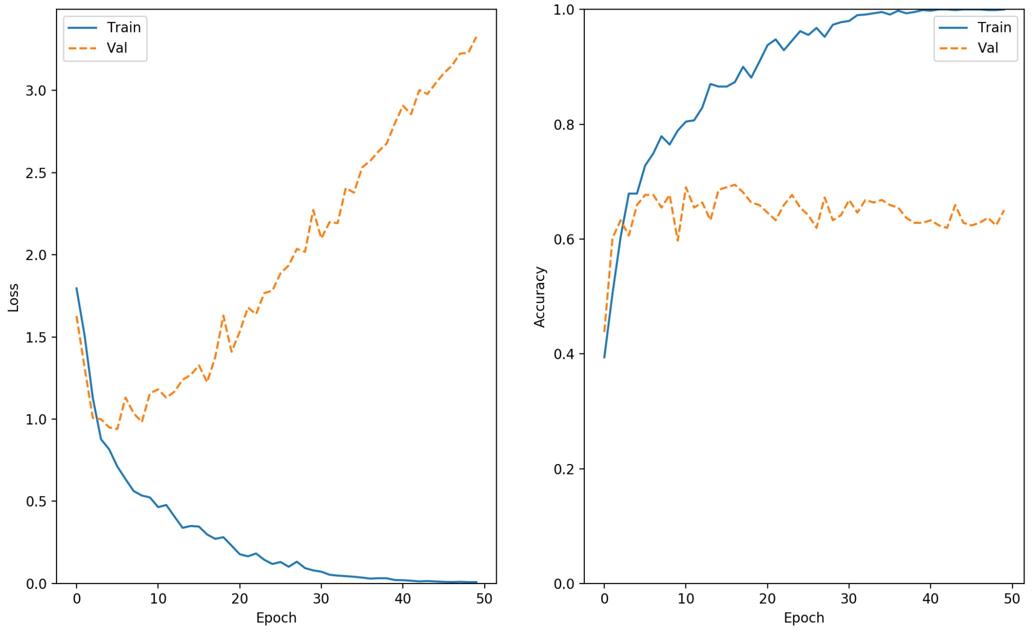


**Figure 7:** CNN v.1.1 loss and accuracy on both the training and validation sets.

(Note: x-axis is off by 1, i.e. it should read from epoch 1 to 51.)

| Epoch | Train Acc | Val Acc | Train Loss | Val Loss | Kaggle Score |
|---|---|---|---|---|---|
| 5 | 0.6792 | 0.6593 | 0.8164 | 0.9495 | 0.67973 |
| 25 | 0.9623 | 0.6549 | 0.1188 | 1.7816 | 0.69607 |
| 50 | 1.0000 | 0.6504 | 0.0083 | 3.3268 | 0.65359 |

**Table 4:** CNN v.1.1 scores on the Kaggle public leaderboard after various epochs.

Interestingly, overfitting is occurring, but not as soon as I'd expect. Looking at Fig. 7, and

I would expect the model to start overfitting to the training data at around epoch 5 or 6. However, the validation accuracy stays fairly constant instead of dropping, while the Kaggle score actually increases for quite a while past epoch 5 before decreasing again. I assume this is due to the small sizes of our training, validation, and testing datasets. I next decided to plot a confusion matrix just to see where our model so far does well and where it struggles:

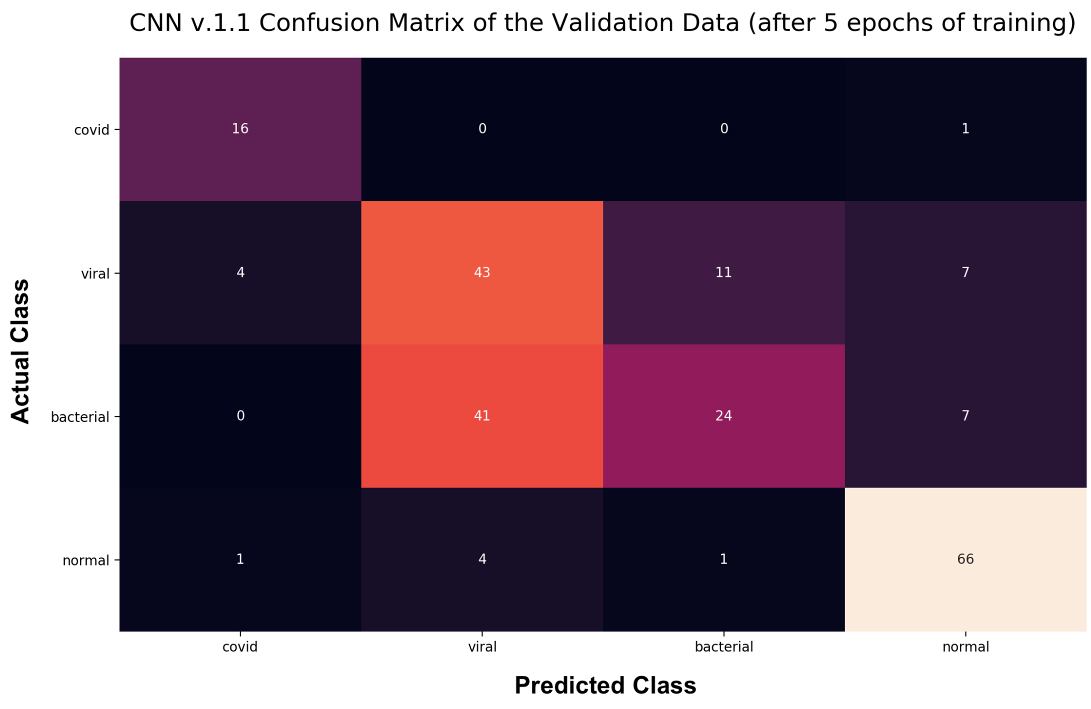


**Figure 8:** Confusion matrix for CNN v.1.1 after 5 epochs of training.

We can see that the model has little difficulty in confidently identifying covid x-rays: it correctly classified 16 of the 17 covid x-rays in the validation set, mistaking one as a normal (healthy) x-ray. The same can be said of the model's ability to predict healthy x-rays, with 66 correctly identified out of the 72 present. The confusion lies in the model's approach to bacterial x-rays: only one third of the bacterial x-ray images were correctly identified, with virtually the entire the other two thirds being mistaken for viral x-rays. However, viral x-rays seemed to be relatively distinct, with 43 correctly identified out of 65, and only 11 mistaken for bacterial x-rays. To get a better understanding for how the neural network is making its predictions, i.e. what features (pixels) it is using to base its decisions, I decided to plot sample x-ray images from the validation set as they go through the convolutions of the network (again after the model has trained for 5 epochs):

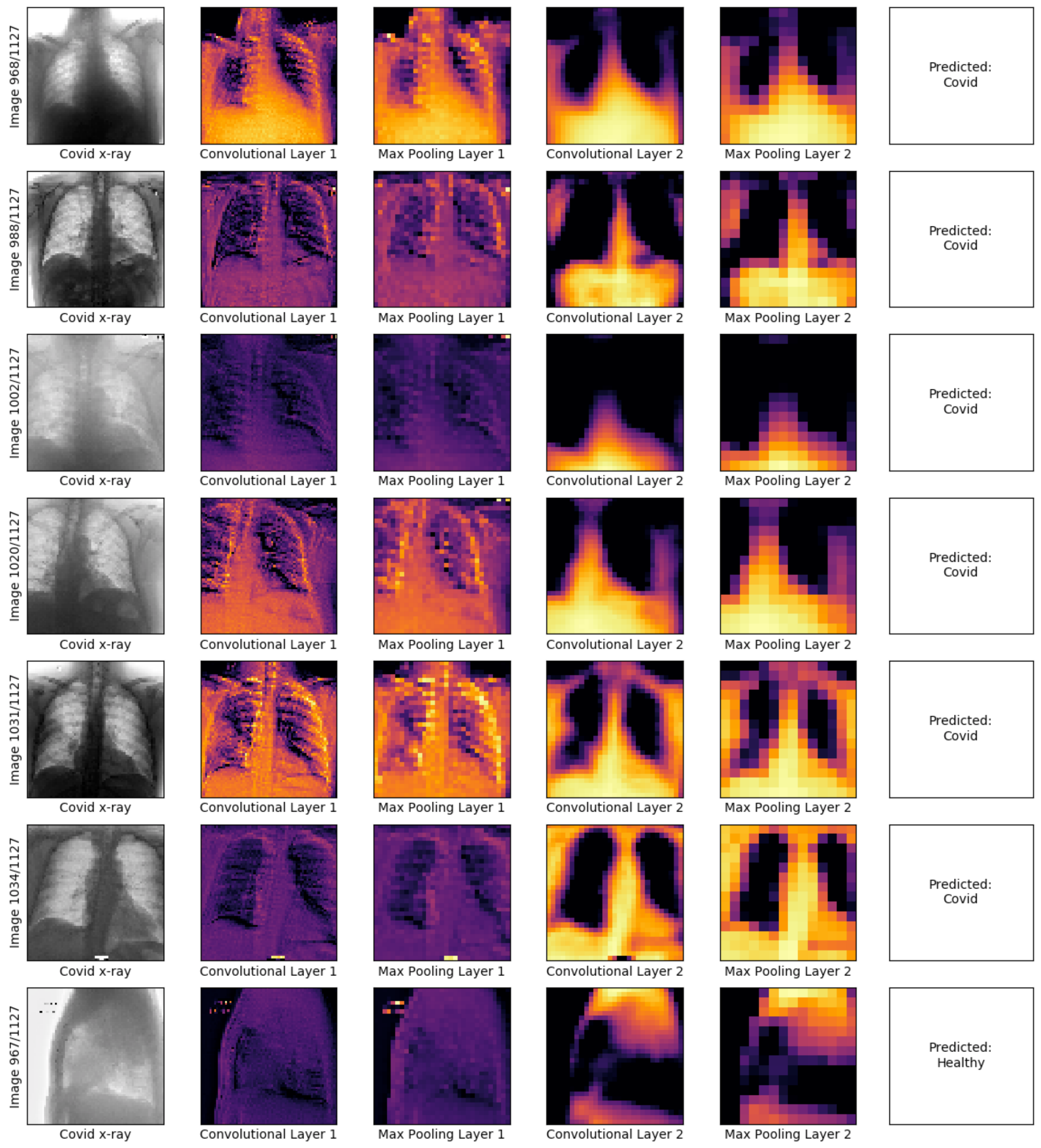


**Figure 9:** Seven COVID-19 x-rays from the validation set shown going through each convolutional layer of the neural network (after training for 5 epochs).

As we can see, the first six x-rays were all correctly identified as covid x-rays; in each one, the predominant shape the neural network has extracted from the final max pooling layer is a triangular mass located towards the bottom of the lungs. Sometimes this triangle has arms flanking it, which appear to correspond to the outline of the ribcage. In the seventh x-ray, the model misclassified the image as being healthy, probably because this x-ray is a side-view, rather than the more common frontal view that dominates the training dataset.

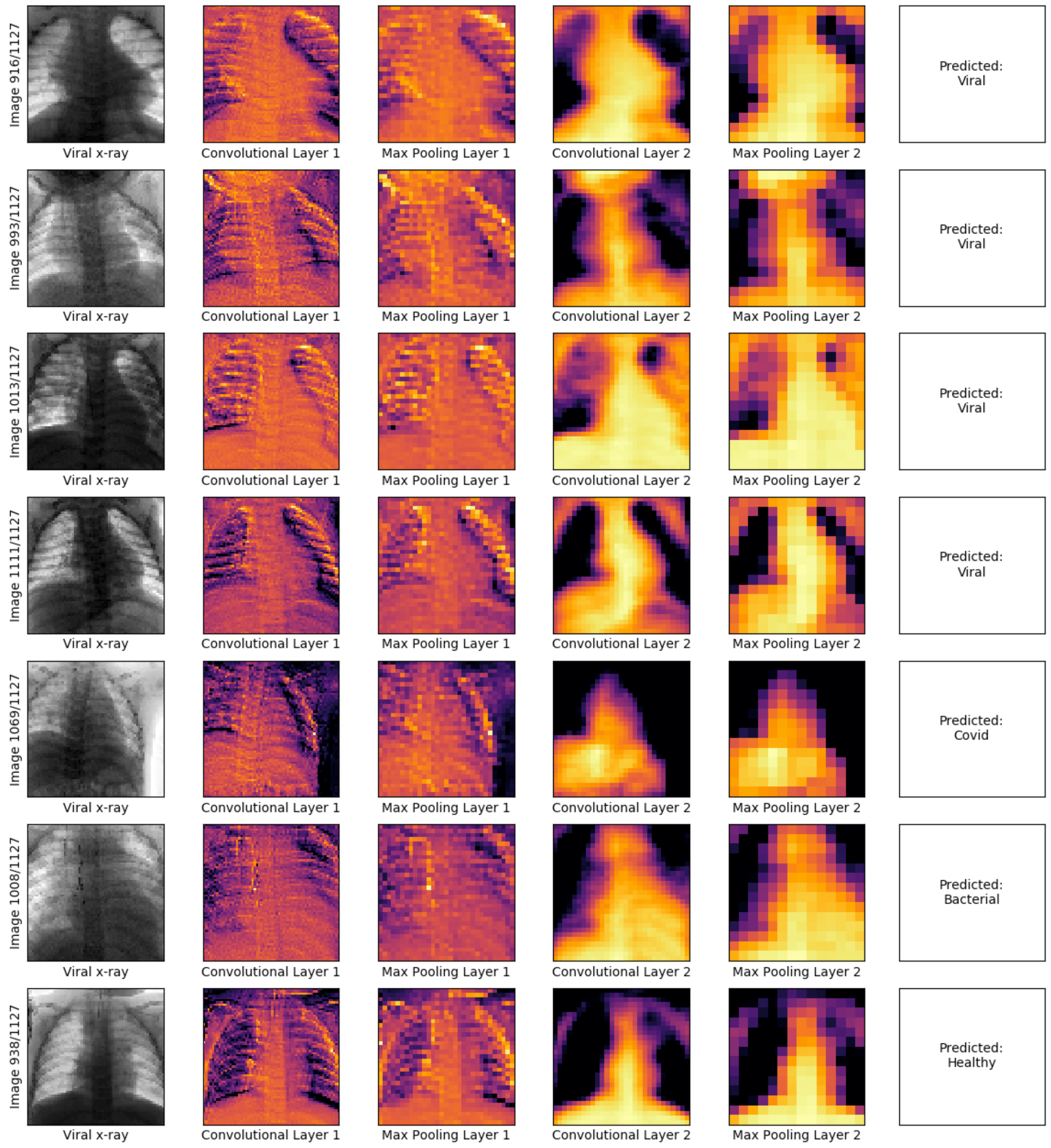


**Figure 10:** Seven VIRAL x-rays from the validation set shown going through each convolutional layer of the neural network (after training for 5 epochs).

Here, the first four x-rays are correctly identified as viral pneumonia; in each one, the predominant shape the neural network has extracted from the final max pooling layer is a large, "squiggly" mass that extends from the top of the frame to the bottom, leaving very little negative space. The last three x-rays have all been misidentified by the model, with each one's final max pooling layer displaying structures corresponding quite closely to their predicted categories (Figs. 9-12).

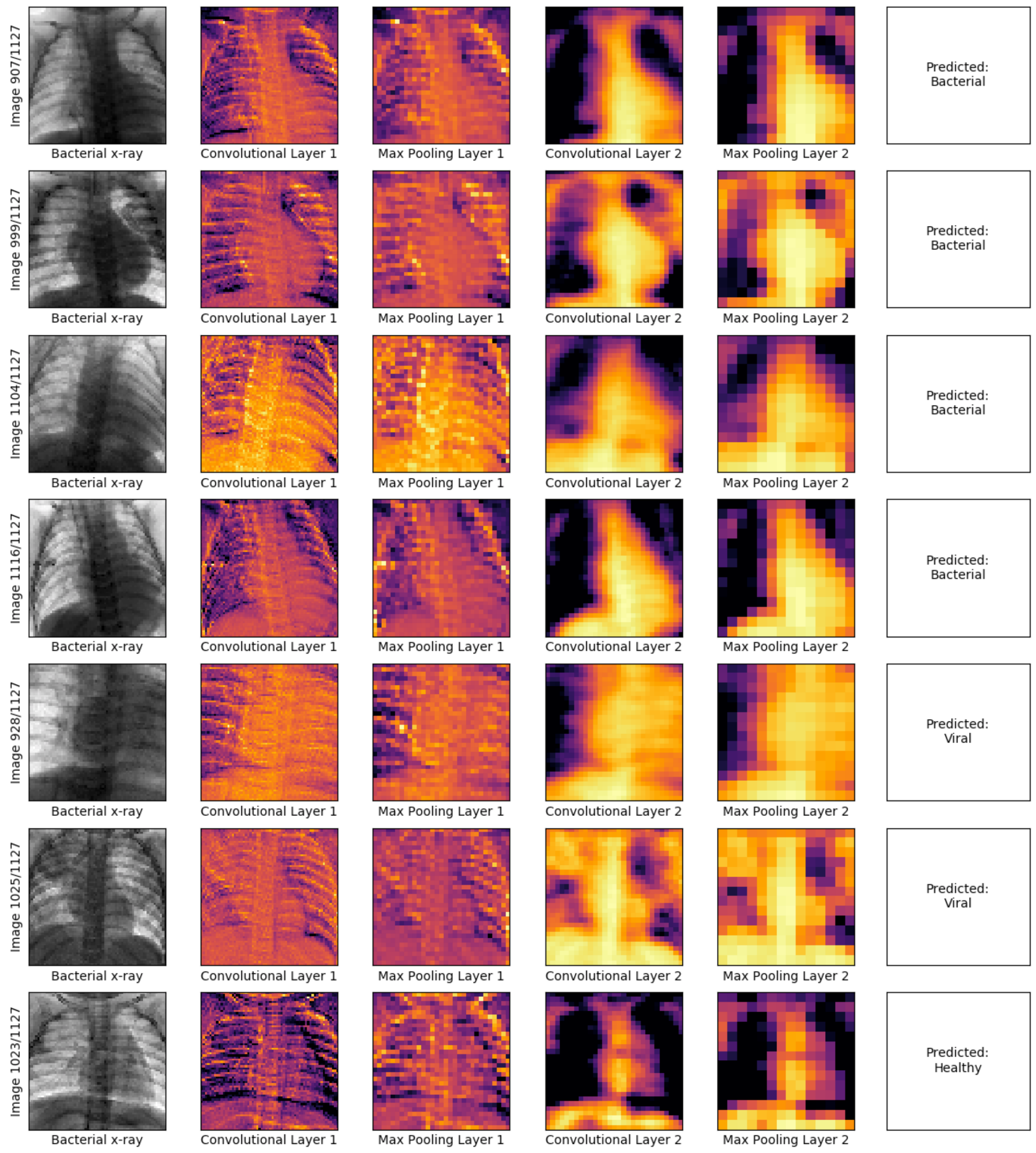


**Figure 11:** Seven BACTERIAL x-rays from the validation set shown going through each convolutional layer of the neural network (after training for 5 epochs).

The first four x-rays shown above have been correctly identified as bacterial pneumonia; similar to the viral x-rays, the predominant structure extracted by the model in these cases is very large, with little negative space. The only distinction my naked eye can find between the viral x-rays presented in Fig 10 and the ones here, is that the viral x-rays are more "squiggly" than these "lumpier" bacterial x-rays. The last three x-rays were incorrectly classified: two as viral and one as healthy (normal).

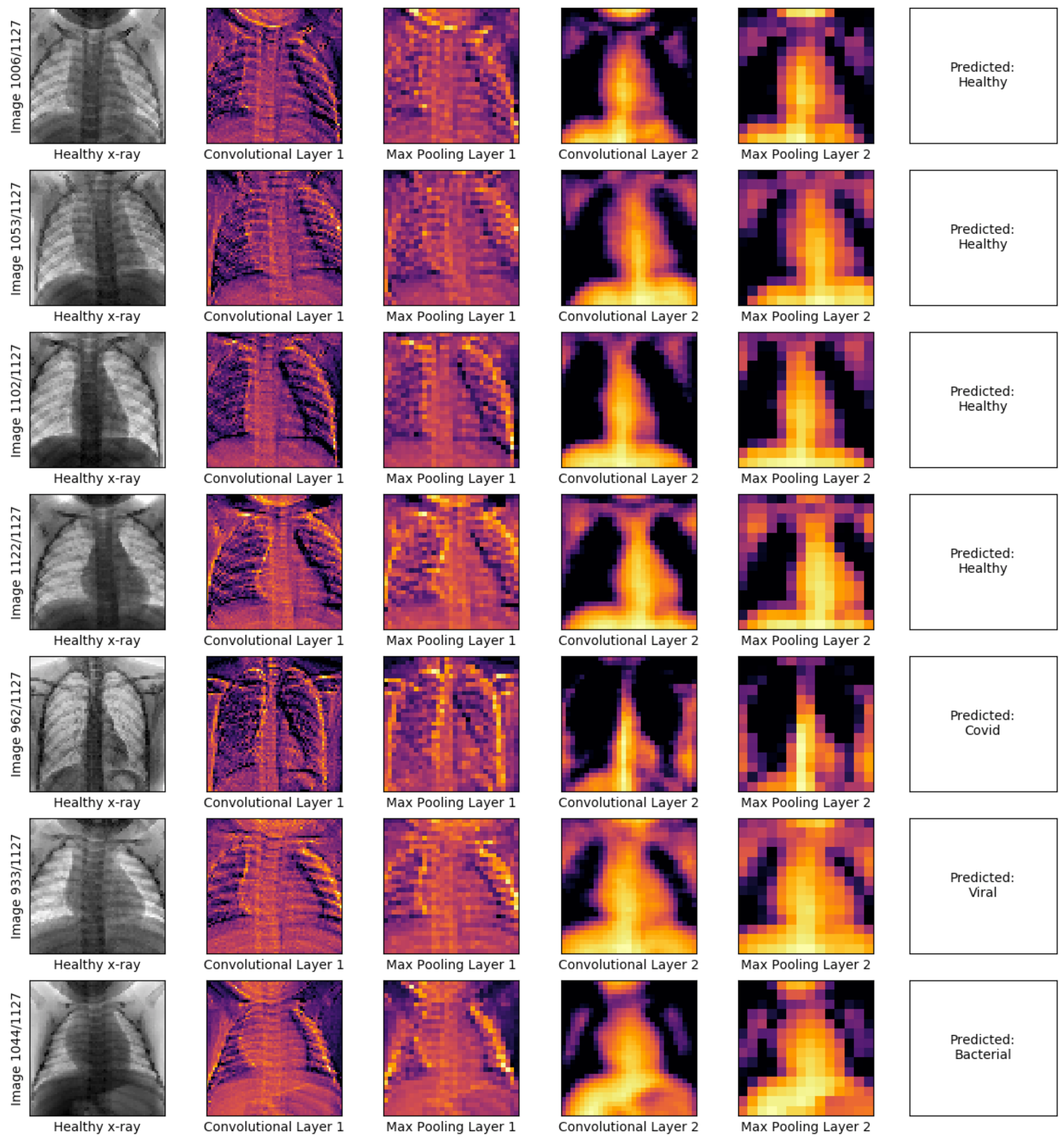
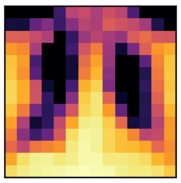


**Figure 12:** Seven HEALTHY (normal) x-rays from the validation set shown going through each convolutional layer of the neural network (after training for 5 epochs).

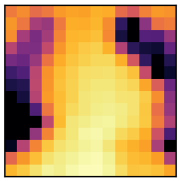
Top four x-rays have been correctly identified as healthy (normal); each one's final max pooling layer displays a structure with lots of negative space relative to the other three classes (Figs. 9-11). The last three x-rays have been misclassified: the one predicted to be covid shows the familiar triangular shape with arms flanking it, the one misidentified as viral shows the "squiggly" mass found in the viral images, while the one predicted as bacterial does not have much negative space, just like the other bacterial images.

Figs. 9-12 show examples of correctly and incorrectly identified covid, viral, bacterial, and normal x-rays going through the neural network (respectively). Each figure displays seven x-rays, with each x-ray getting its own row. In every figure, the first four x-rays are correctly identified, while the last three are misclassified (except for Fig. 9, which has the first six images correctly identified, and the final image misclassified). While each row displays a single x-ray, each column displays that x-ray after having gone through a different convolutional layer of the neural network. The final convolutional layer, i.e. the second max pooling layer (fifth column), displays essentially what features the neural network is utilizing when making a prediction and distinguishing between different classes. This has given rise to four different structures that could potentially characterize each class of x-ray, as well as provide clinicians with useful indicators when attempting to diagnose and combat COVID-19 using only x-ray images:

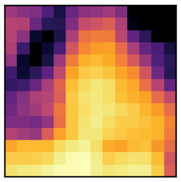
1. **COVID-19** can be characterized by a triangular structure that sits low, towards the bottom of the lungs, while arms are sometimes seen flanking this structure that correspond to the outline of the ribcage.



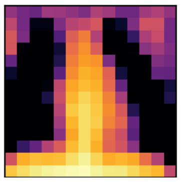
2. **VIRAL** pneumonia can be characterized by a large, "squiggly" mass that extends from the top of the ribcage to the bottom, leaving very little negative space.



3. **BACTERIAL** pneumonia can be characterized by a large mass, "lumpy" in shape, that leaves little negative space. Somewhat triangular in shape, but distinct from COVID-19 in that it extends from the top of the chest to the bottom, rather than sitting low.



4. **NORMAL** (healthy) x-rays can be characterized by loads of negative space, with the only definitive features being the spine and ribcage outline.



## 3.3 Transfer Learning v.2.0

At this point I decided to introduce transfer learning. For the first model to implement transfer learning, I chose my base model as a pretrained neural network from TensorFlow, an ImageNet classifier. The one I chose can be found at:

`https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4`

The base model takes images of size 224×224×3 pixels (all 3 RBG channels are present), and outputs 1280 neurons by default. So, I resized all of the images to be the appropriate 224×224×3. I then connected this output to a dense layer of my own, with only 4 neurons (the classification head of the network). This results in 2,263,108 total parameters, with only 5,124 of them being trainable (the base model is frozen, so the only trainable weights are the connections between the 1280 output neurons from the base connected to the 4 neurons of my own dense layer that acts as a classification head, along with the 4 neurons from the dense layer, i.e. 1280*4 + 4 = 5124). I then trained the model for 40 epochs (with the same 80/20 train/val split from CNN v.1.1), which took about 18 minutes and 30 seconds. I imposed the same class weighting introduced in CNN v1.1, as well as the same shuffling of the data. The loss and accuracy over the 40 epochs can be seen below, along with the score on the Kaggle competition's public leaderboard at two different checkpoints:

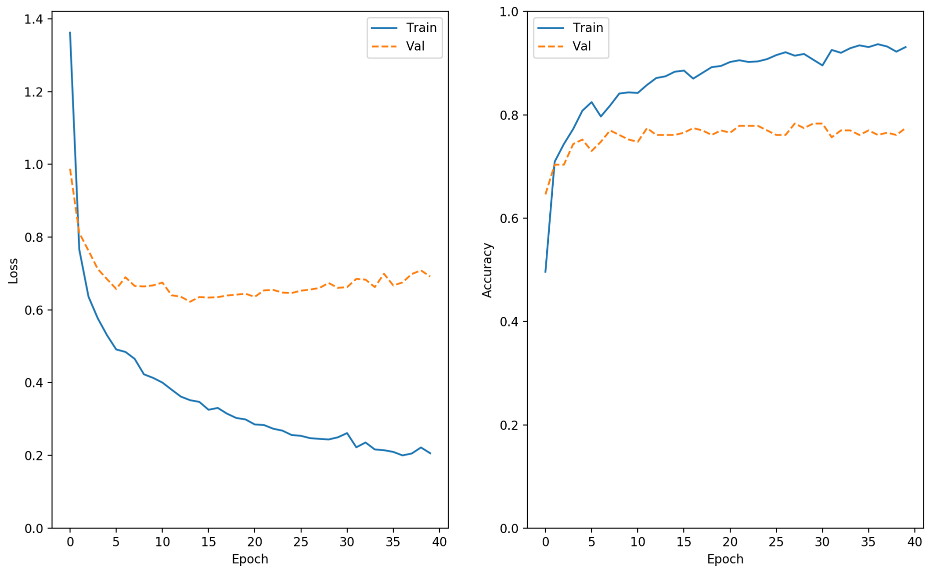


**Figure 13:** Transfer Learning v.2.0 loss and accuracy on both the training and val. sets.

(Note: x-axis is off by 1, i.e. it should read from epoch 1 to 41.)

| Epoch | Train Acc | Val Acc | Train Loss | Val Loss | Kaggle Score |
|---|---|---|---|---|---|
| 4 | 0.7725 | 0.7434 | 0.5769 | 0.7124 | 0.81372 |
| 20 | 0.8946 | 0.7699 | 0.2989 | 0.6443 | 0.81699 |
| 40 | 0.9312 | 0.7743 | 0.2060 | 0.6915 | n/a |
| 12[†] | n/a | n/a | n/a | n/a | 0.83006 |

**Table 5:** Transfer Learning v.2.0 scores on the Kaggle public leaderboard after various epochs.

[†] Note: This Kaggle score is the high for this model, and came via two small changes: firstly, the learning rate was reduced by a factor of 0.1 every time the validation loss was seen to plateau / not improve for 2 epochs. Secondly, the best model was saved over the course of 20 epochs (that being the model after the 12th epoch).

Finally, a confusion matrix of the validation dataset is plotted below (after 20 epochs):

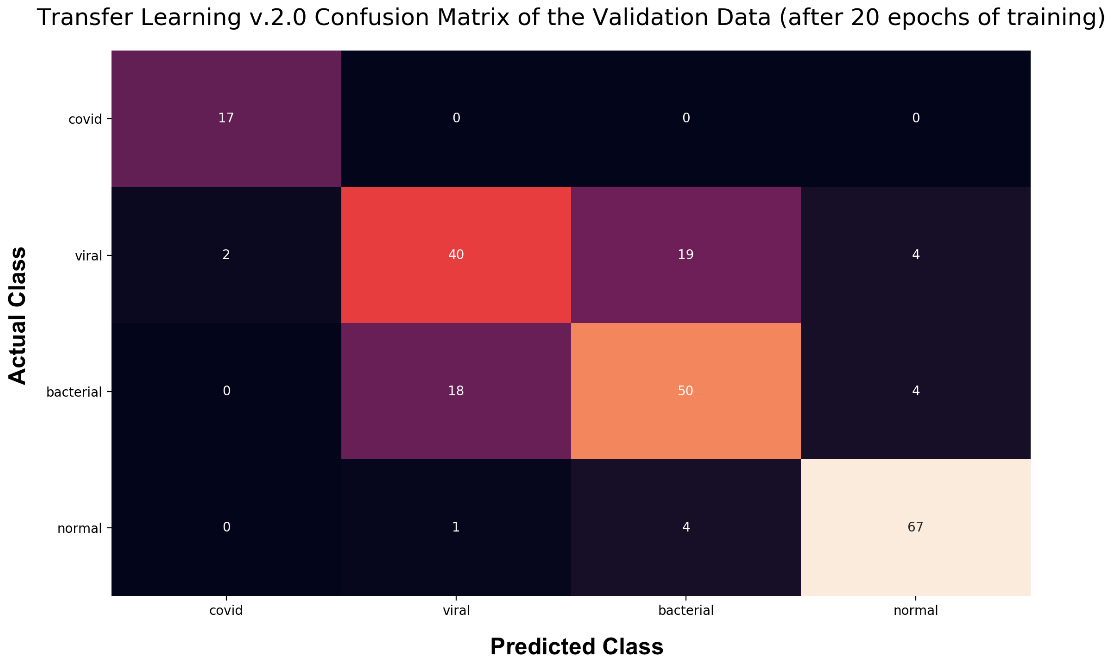


**Figure 14:** Confusion matrix for Transfer Learning v.2.0 after 20 epochs of training.

The main problem this model has is its lack of interpretability: since the architecture is not that of a convolutional neural network, we cannot perform the same analysis we did for the CNN v1.1 model. Therefore, one final model will be examined.

## 3.4 Transfer Learning CNN v.3.0

This model again utilizes transfer learning, with the base model this time being a pretrained *convolutional* neural network from TensorFlow, trained on the ImageNet dataset again. The chosen model is the VGG16 model: a CNN with 19 layers total that takes images of the shape 224×224×3 (all three RBG channels present). I flattened the final convolutional layer's output from the VGG16 model, and fed it to a dense layer of 4 neurons, acting as the classification head of the model. All but the very last three layers of the VGG16 model were frozen, leaving the model as a whole with 2,460,164 trainable parameters out of the 14,815,044 total. It took approximately 46 minutes to train the model for 9 epochs. Other than these changes, the parameters and specifics of the model follow those of the others I've discussed throughout section 3. Below I have shown the loss and accuracy over 9 epochs of training, along with the score on the Kaggle competition's public leaderboard at a few different checkpoints (note: the Fig. 15 only shows the loss/accuracy for the first 9 epochs, while the Table 6 displays scores the model received after having gone through some extra training):

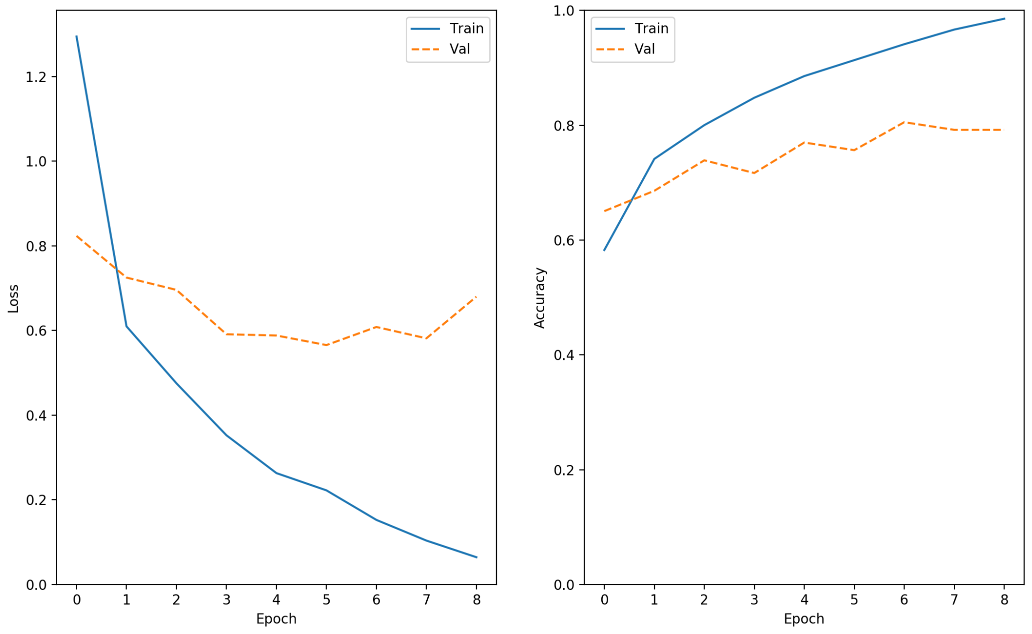


**Figure 15:** Transfer Learning CNN v.3.0 loss and accuracy on both the training and val. sets.

(Note: x-axis is off by 1, i.e. it should read from epoch 1 to 10.)

| Epoch | Train Acc | Val Acc | Train Loss | Val Loss | Kaggle Score |
|---|---|---|---|---|---|
| 9 | 0.9856 | 0.7920 | 0.0645 | 0.6799 | 0.83986 |
| 11 | 0.9900 | 0.8097 | 0.0445 | 0.7432 | 0.83006 |
| 13 | 0.9900 | 0.8186 | 0.0422 | 0.7579 | 0.83986 |

**Table 6:** Transfer Learning CNN v.3.0 scores on the Kaggle public leaderboard after various epochs.

Below is a confusion matrix of the validation dataset (after 9 epochs):

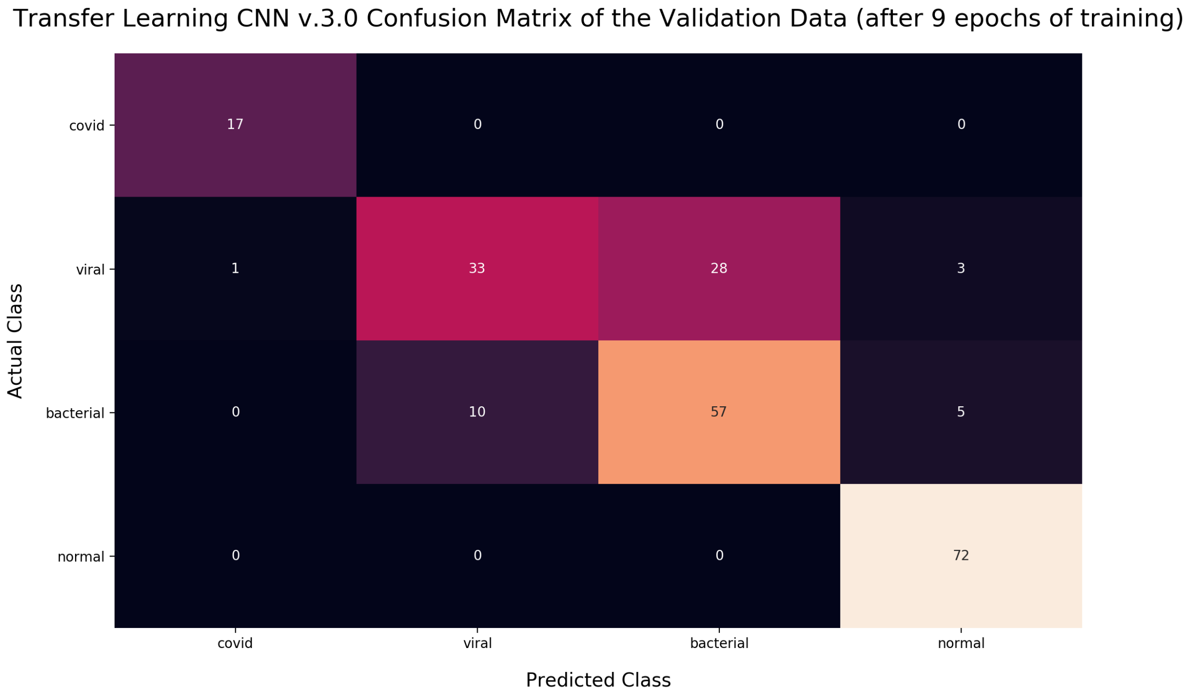


**Figure 16:** Confusion matrix for Transfer Learning CNN v.3.0 after 9 epochs of training.

This, as we can see from Fig. 15 and Table 6, is our best model yet. The confusion matrix supports this. The model confidently identifies covid x-rays and normal x-rays, and has gotten much better at classifying bacterial pneumonia (slightly at the expense of viral pneumonia, where the model struggles to tell it apart from bacterial). Finally, we can examine some specific sample x-ray images going through the model layer by layer, to try and interpret / understand what governs the model's predictions, i.e. find out which features the model is relying on (again, I am examining the model after 9 epochs):

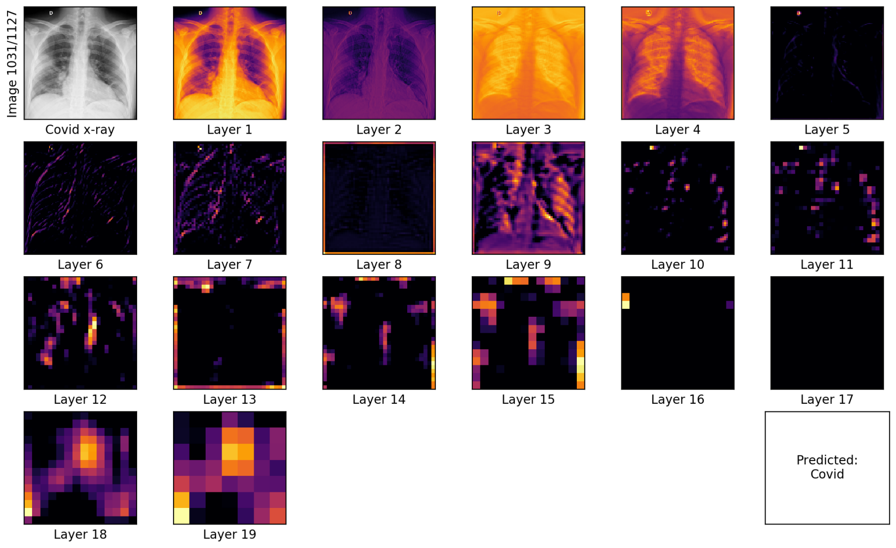


**Figure 17:** A single covid x-ray correctly predicted by the model shown going through each convolution of the neural network.

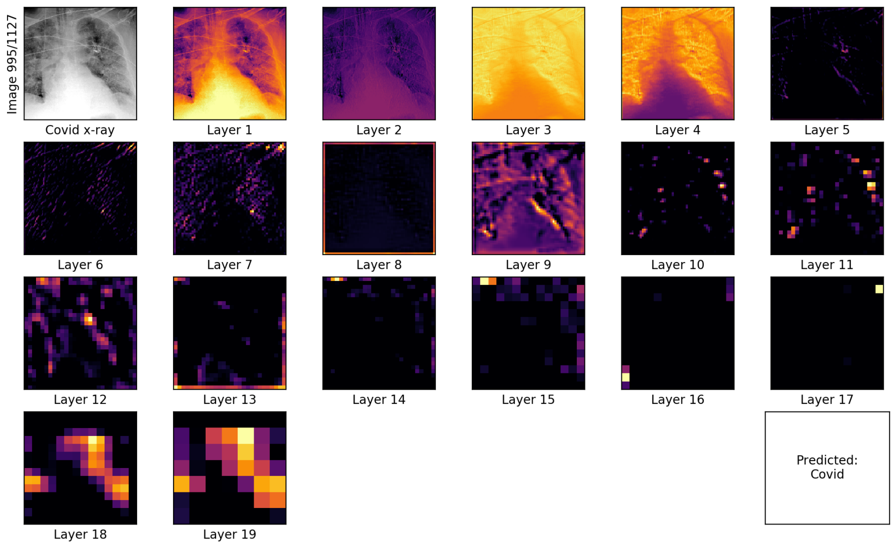


**Figure 18:** Another covid x-ray correctly predicted by the model shown going through each convolution of the neural network.

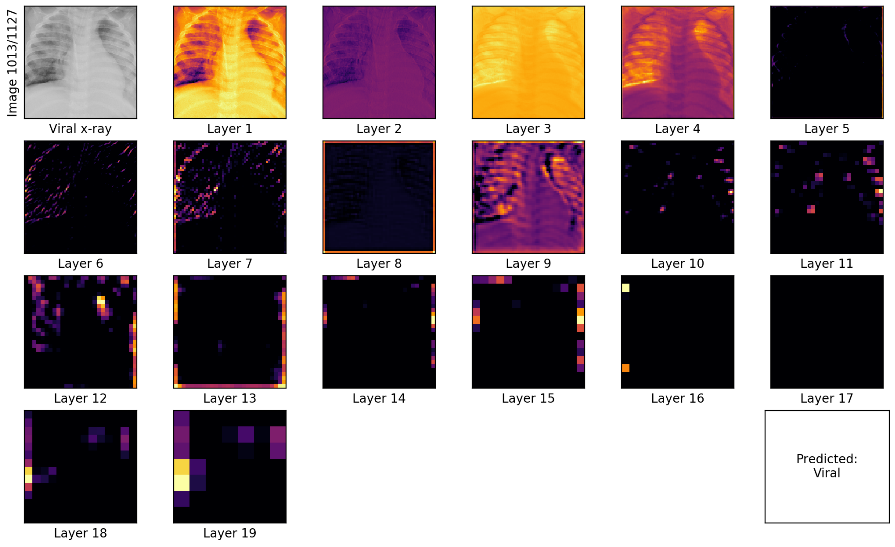


**Figure 19:** A single viral x-ray correctly predicted by the model shown going through each convolution of the neural network.

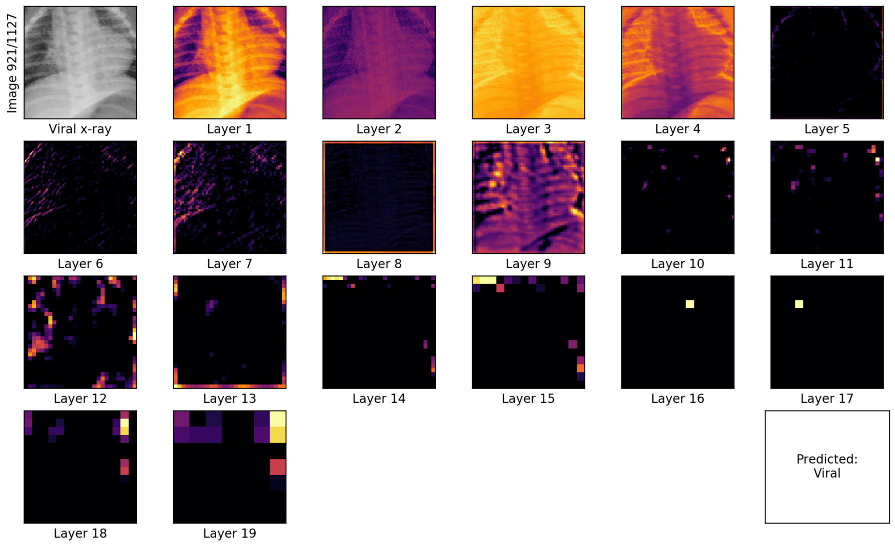


**Figure 20:** Another viral x-ray correctly predicted by the model shown going through each convolution of the neural network.

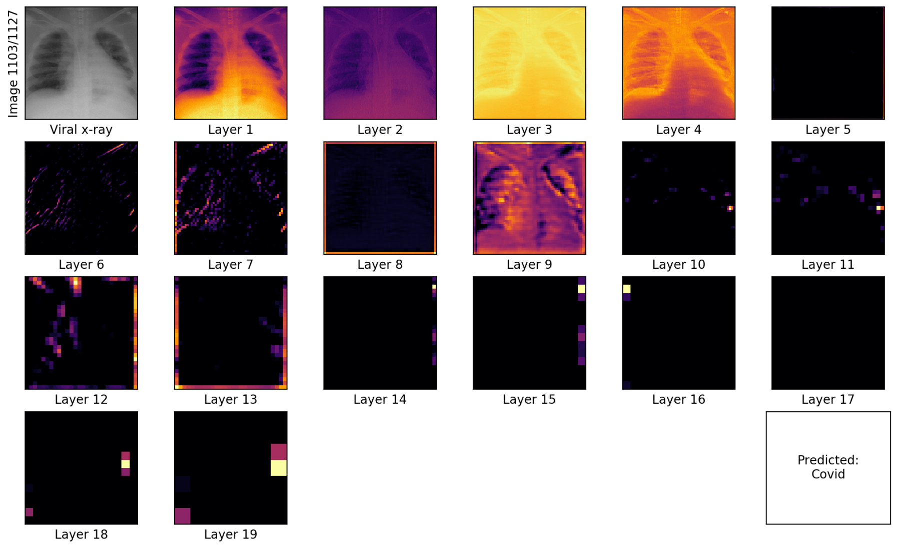


**Figure 21:** A viral x-ray misidentified by the model as a covid x-ray shown going through each convolution of the neural network.

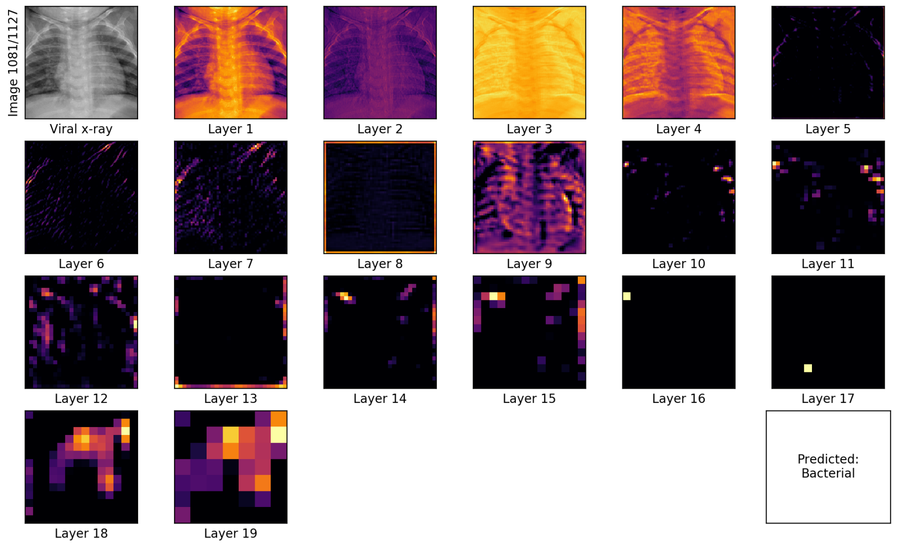


**Figure 22:** A viral x-ray misidentified by the model as bacterial pneumonia shown going through each convolution of the neural network.

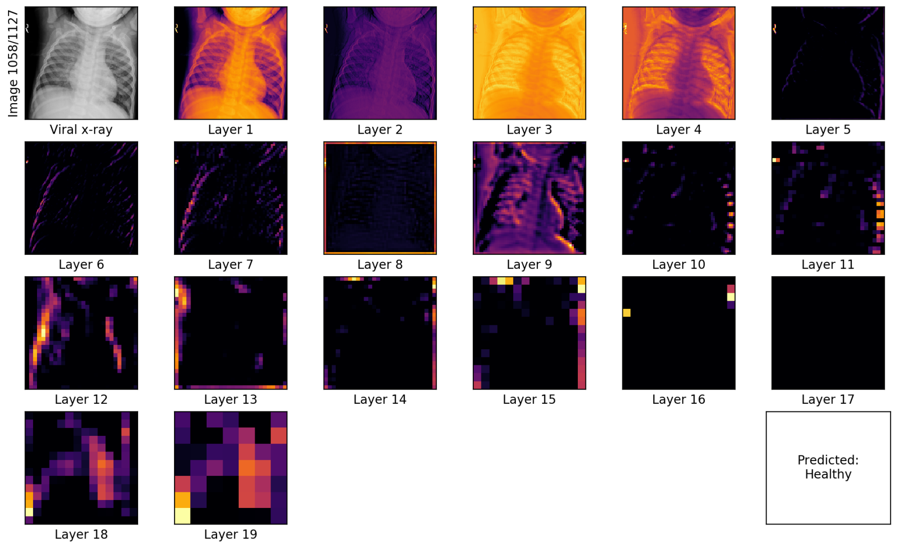


**Figure 23:** A viral x-ray misidentified by the model as healthy (normal) shown going through each convolution of the neural network.

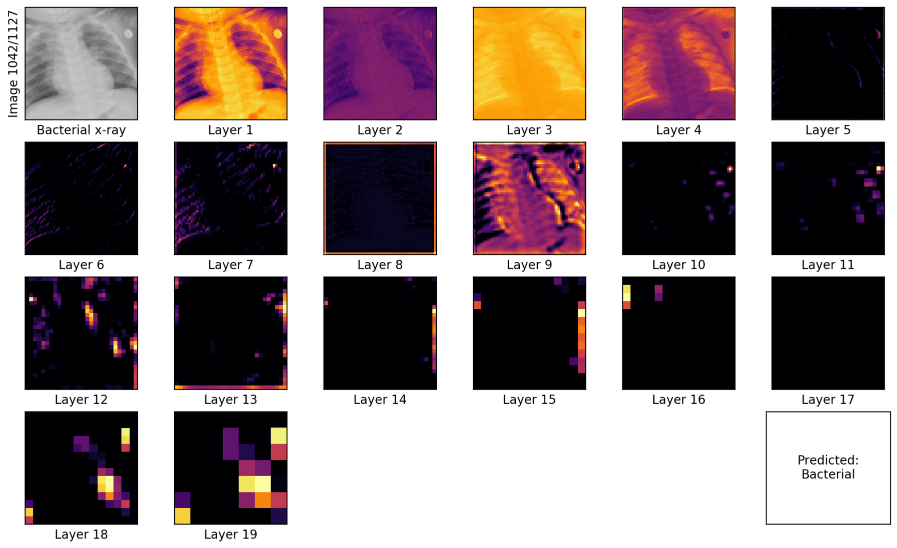


**Figure 24:** A single bacterial x-ray correctly classified by the model shown going through each convolution of the neural network.

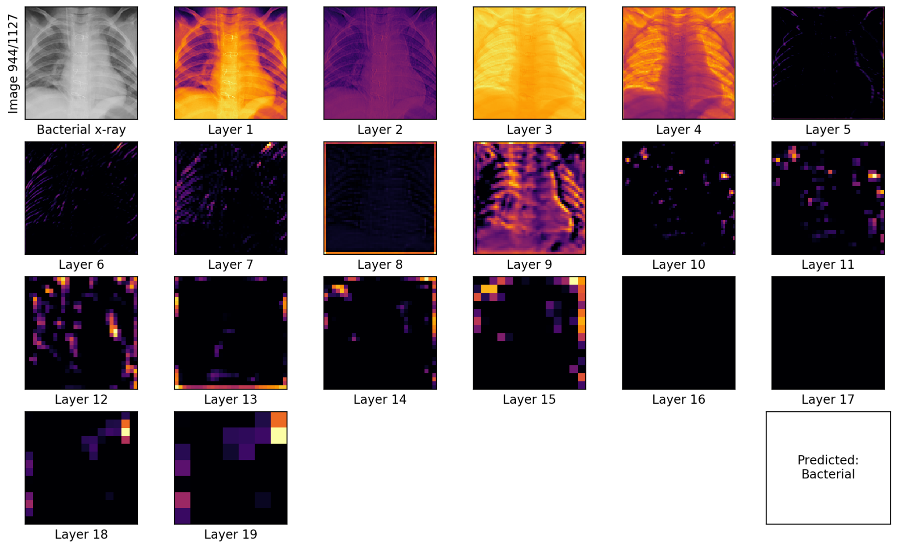


**Figure 25:** Another bacterial x-ray correctly classified by the model shown going through each convolution of the neural network.

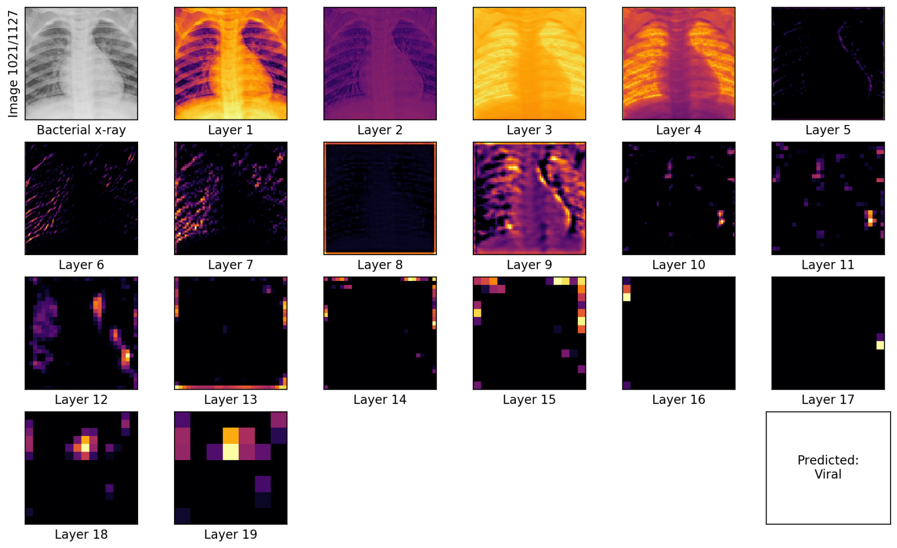


**Figure 26:** A bacterial x-ray misidentified by the model as viral shown going through each convolution of the neural network.

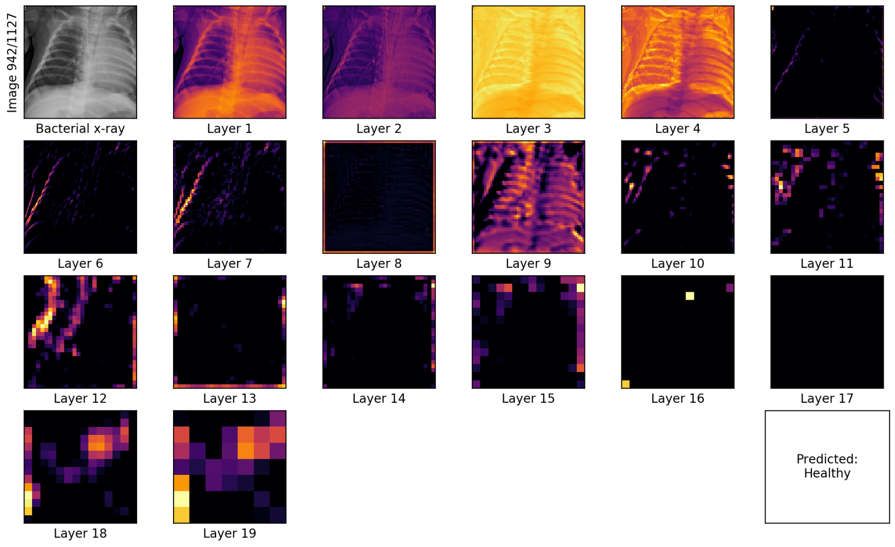


**Figure 27:** A bacterial x-ray misidentified by the model as healthy (normal) shown going through each convolution of the neural network.

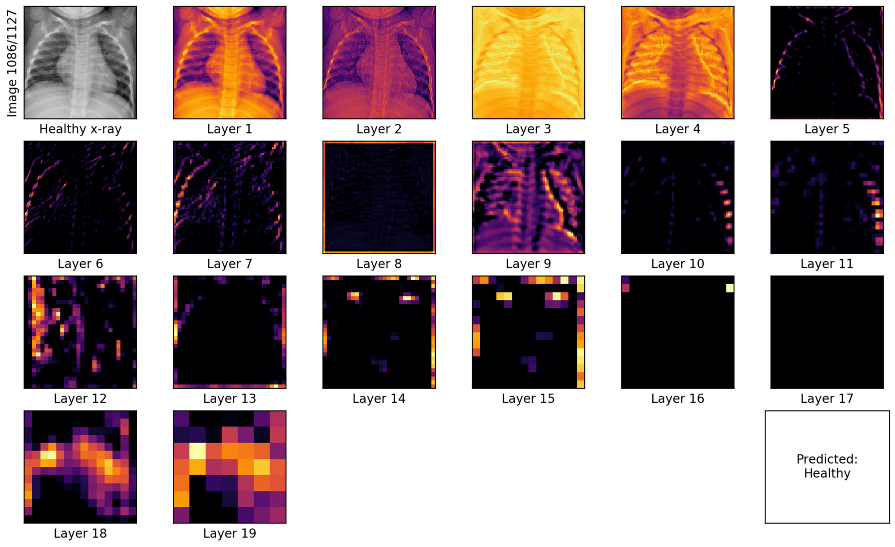


**Figure 28:** A single healthy (normal) x-ray correctly identified by the model shown going through each convolution of the neural network.

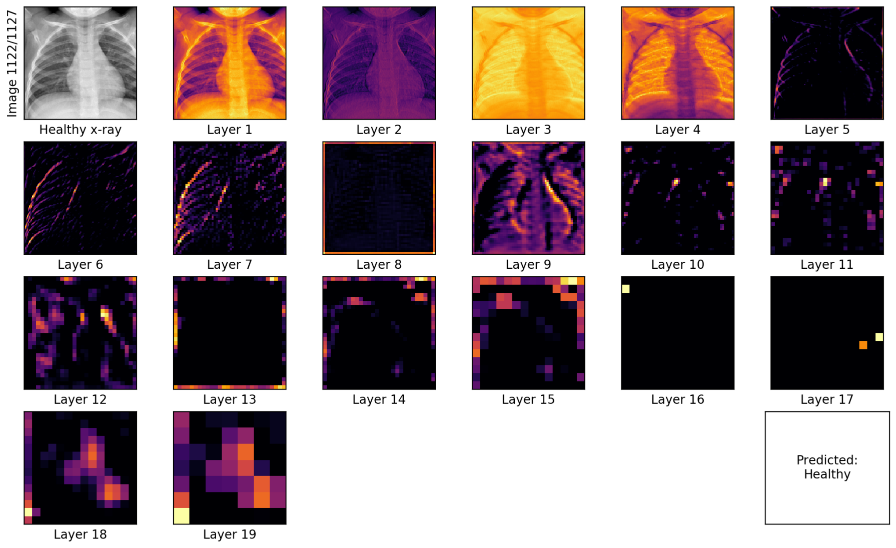


**Figure 29:** Another healthy (normal) x-ray correctly identified by the model shown going through each convolution of the neural network.

## 4 References

[1] Coms 4771 covid challenge.

[2] Kermany, D. S., Goldbaum, M., Cai, W., Valentim, C. C., Liang, H., Baxter, S. L., McKeown, A., Yang, G., Wu, X., Yan, F., et al. Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell 172*, 5 (2018), 1122–1131.

[3] Rajpurkar, P., Irvin, J., Zhu, K., Yang, B., Mehta, H., Duan, T., Ding, D., Bagul, A., Langlotz, C., Shpanskaya, K., et al. Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. *arXiv preprint arXiv:1711.05225* (2017).

# List of Figures

## List of Tables