This is an image-classification problem, so I used a convolutional neural network (CNN), which achieved a reasonable accuracy. Images have many pixels, each corresponding to an input feature/dimension. The high input dimensionality causes difficulties to most learning methods. However, CNNs can use the spatial organization information from local patches of the image, leading to a model with fewer parameters. It is thus one of the most used methods in image classification.

The first step was to preprocess the images. First, I resized the images to 256 by 256 and normalized them to the range $[0, 1]$. I then used contrast enhancement which improved the result by 3.5%. The categories include bacterial, viral, COVID, and normal, but there were roughly $4.5\times$ less examples in the COVID category. Therefore, I also performed data augmentation of the COVID class. The data augumentation included a small random scaling up ($\sim 10\%$) and rotations of -2 to +2 degrees. The data augmentation improved the accuracy by approximately 5%. These processes are illustrated in the figures below.

I implemented the CNN classifier in Python using Keras. The CNN comprised 5 sets of Conv2D and MaxPooling layers, followed by two Dense layers. I set up 5 Conv2D layers with 32, 64, 64, 64, and 64 filters respectively. The Dense layers had 128 and 4 nodes respectively. Before each Dense layer, I also added Dropout with 0.2 drop rate, which seemed to help reduce the difference in the loss between training and validation sets.

For training, I splitted the original training data into 80% for training and 20% for validation. I initially used 85% and 15%. I changed it because the results from the initial cross-validation deviated from the true results too much and were not consistent. Since the total number of data in the original training set is not particularly large, my improvement reduced the stochastic variability in the validation set.

I then used a learning rate of 10^{-4} with the Nadam optimizer, 32 batch size, and a maximum of 200 epochs with early-stopping. Early-stopping interrupted training if the validation loss did not improve for 10 epochs. I monitored the "loss" (categorical cross-entropy in this case) and "accuracy" metrics for both training and validation sets. The

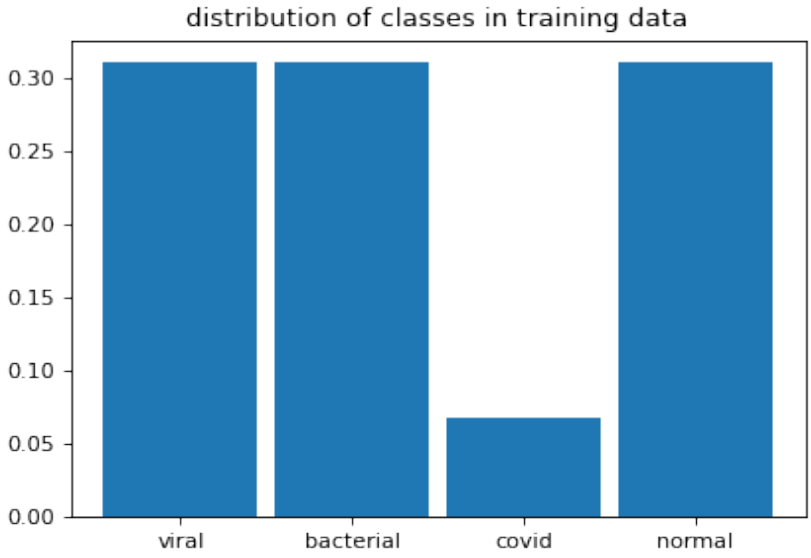

Figure 1: Data distribution per class before data augmentation.

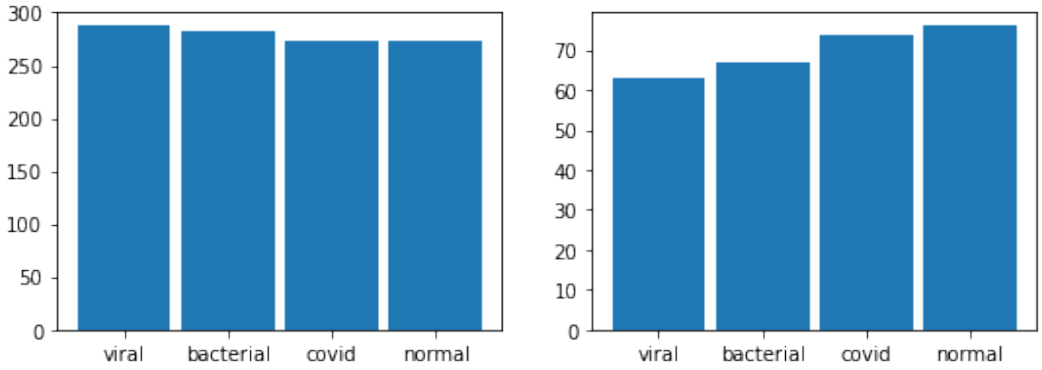

Figure 2: Data distribution per class after data augmentation and random splitting into training (left) and validation (right) sets.

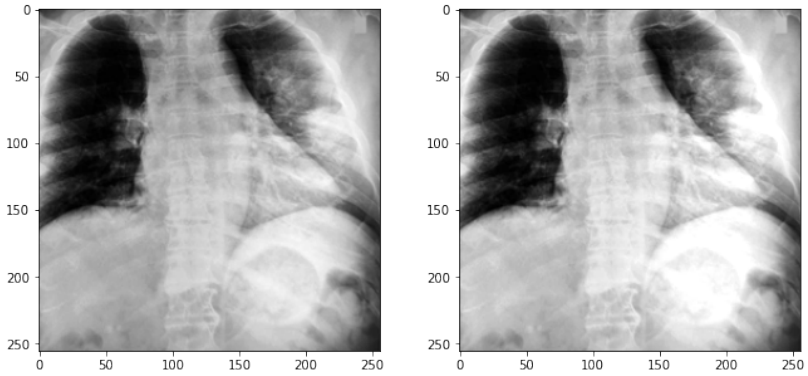

Figure 3: Original (left) and contrast enhanced (right) images.

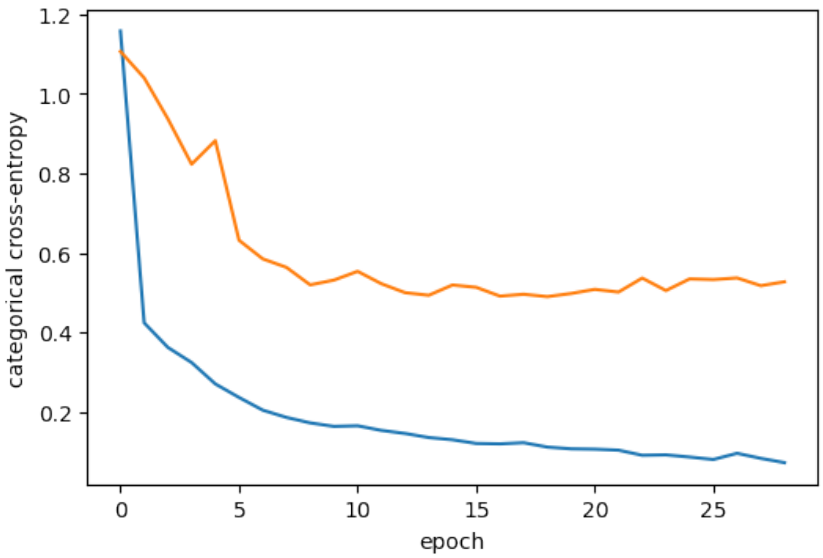

Figure 4: Loss metric learning curve for the training (blue) and validation (orange) sets.

corresponding learning curves are shown in Figures 4 and 5. The figures illustrate that there is still a large gap in performance between the training and validation sets, even after adding dropout. This means that the CNN is still overfitting the training data, which would explain the additional drop in performance with regard to the Kaggle results leaderboard.

I also looked at the confusion matrix of the validation set results, shown in Figure 6, to analyze the distribution of the classification performance for different classes. I found that the CNN seemed capable to easily distinguish different classes. Most of the misclassification seems to be between the bacterial and viral classes. This is somewhat surprisingly considering that COVID is also viral. It could mean that COVID attacks the respiratory system very differently than common viruses, which is reasonable considering the severity of symptoms of COVID infections. From a clinical application perspective, this classifier would likely work quite reliably for normal and COVID classes, but less so with regard to the bacterial and viral classes. It is not only giving misclassification between the bacterial and viral classes, but it also generates a significant fraction of cases classified as normal.

Interpreting the predictions of a CNN is intricate. I would perform the analysis presented in the article by Zeiler and Fergus [1], which highlighted which patterns of the X-Ray images are considered the most significant for the classification. The intuition of the approach is

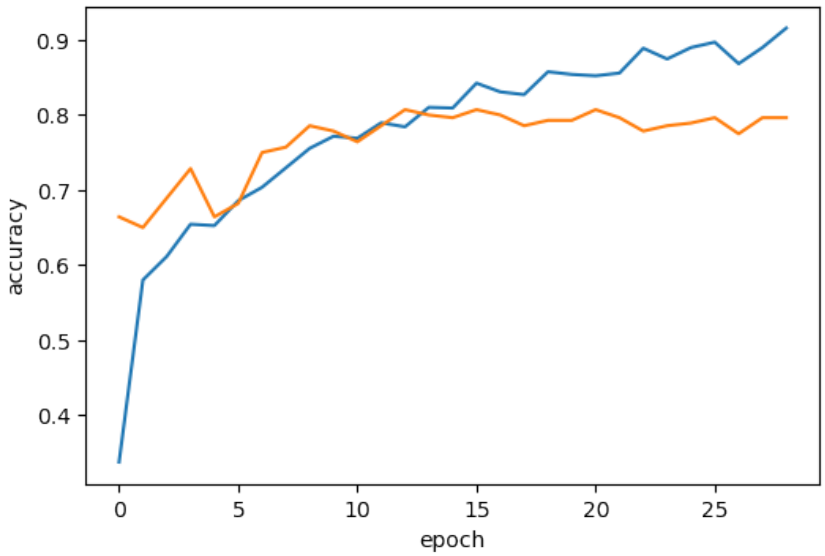

Figure 5: Accuracy metric learning curve for the training (blue) and validation (orange) sets.

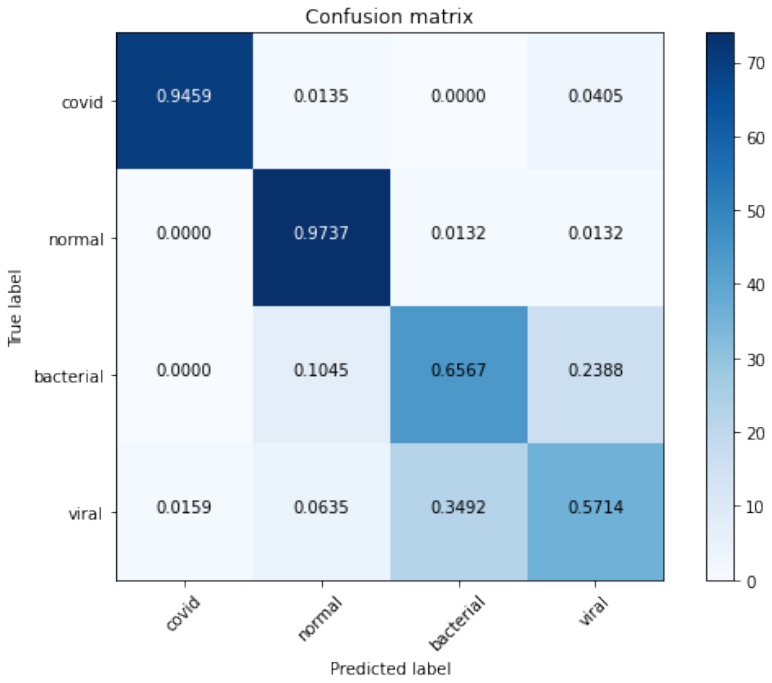

Figure 6: Confusion matrix of the classification results in validation set.

to pull the most important image patterns, which is similar to how clinicians would inspect these images.

I found that the training of the CNN always exhibited a significant gap between performance in training and validation sets. This was an issue, and adding Dropout layers helped. The gap is still significant, so I would need to spend more time exploring alternate regularization techniques or trying frameworks which regularize the CNN by learning representations [2].

References

[1] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[2] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pages 3581–3589, 2014.