COMS4771 Final Project

Maggie Yin (mzy2001)

4/19/20

Developing Models

I initially tried to use the perceptron algorithms from homework 2 that were quite successful on the MNIST multi-class classification task. I focused on V2 because that was the one that was most successful in the homework and most successful in preliminary training. While the testing accuracy got to around 65%, submitting my predictions to Kaggle resulted in a max accuracy of 28%. This is when I switched gears to deep learning and developing a tensorflow/keras neural network for the task at hand.

Having never developed a NN before, I first tried a general CNN that seemed to work well for many image classification tasks that I read about online[1]. My model had six convolutional layers and three fully connected layers. In pairs, the filters for the convolutional layers were 64, 128, and 256. I added dropout (10% between convolutional layers and 50% between fully connected layers), with the aim of reducing overfitting. I also made use of batch normalization and max pooling. I experimented with different activations, optimizers, epochs, batch sizes, and callbacks. Something that I think really made a difference was learning rate reduction; if, during training, validation accuracy stopped increasing, then the learning rate was reduced. Figure 1 shows results of a sample run of k-fold cross validation using my model. It achieved: Accuracy: mean=74.695 std=0.546, n=3. The best this model achieved after training was 79% on the Kaggle leaderboard, and 80% on my holdout test dataset.

The next model I tried was a transfer learning model using ResNet50 as the base model and a custom top including 2 dense layers, a leaky relu layer, dropout, batch normalization, and average pooling. I trained this model with the same callbacks/epochs/batch size as the previous. However I used an SGD optimizer with initial learning rate as 0.01 (instead of adam). This model performed better than the previous, with an accuracy of 87% on my test dataset and 86% on the Kaggle leaderboard.

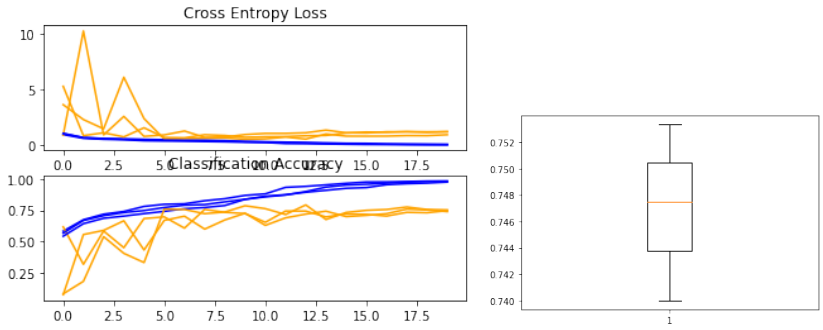


Figure 1: k-fold cross validation training accuracy and loss

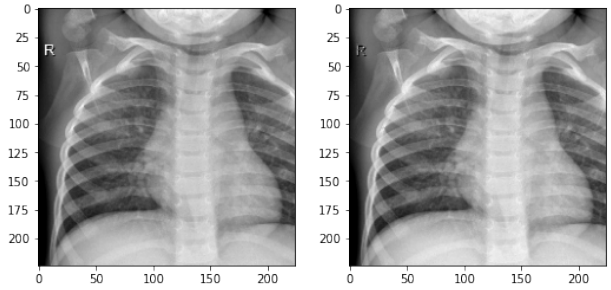


Figure 2: Before and after text removal (letter R on the left side)

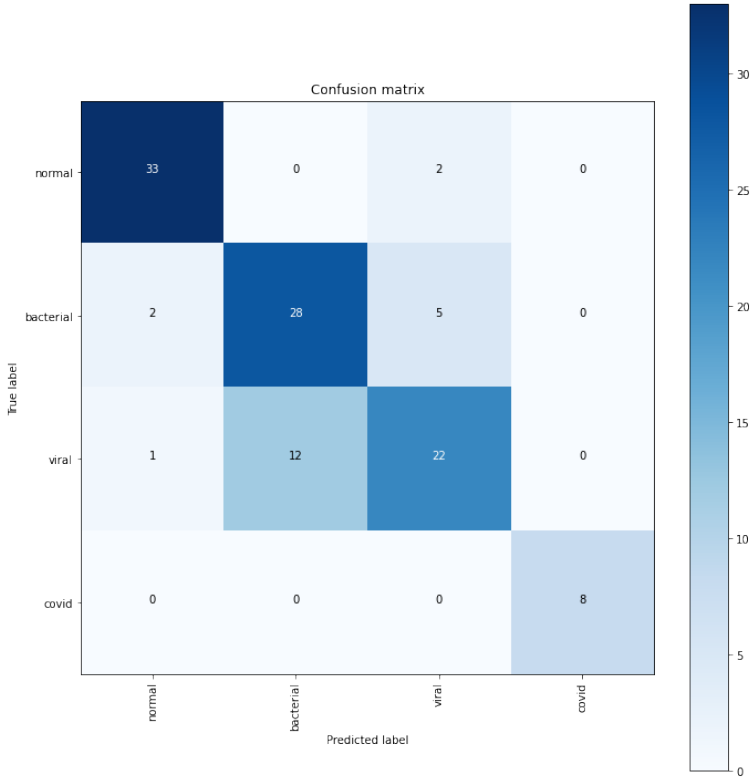
Preprocessing

Steps I took for image preprocessing included resizing all images to 128x128, and normalizing all pixel values to [0,1]. I trained using a data generator for data augmentation; I used rotation (30), width shift (0.2), horizontal flip, and zoom (0.2). This helps in avoiding overfitting as the augmented images will generalize better to images outside of the training dataset. In terms of uneven distribution of classes in the training data (low representation of covid class), I calculated class weights based on the histogram of the training data to be used during training, in order to make sure we are learning this minority class.

Something I noticed was that many images had text (for example, a white R on the side of the image) that I thought adding noise. I attempted to remove the text (essentially the brightest pixels) using thresholding and inpainting. See figure 2 for the difference before and after text removal. Perhaps because the inpainting was not perfect for a lot of the images, I don't think this actually led to any improvement in the accuracy.

Error Analysis

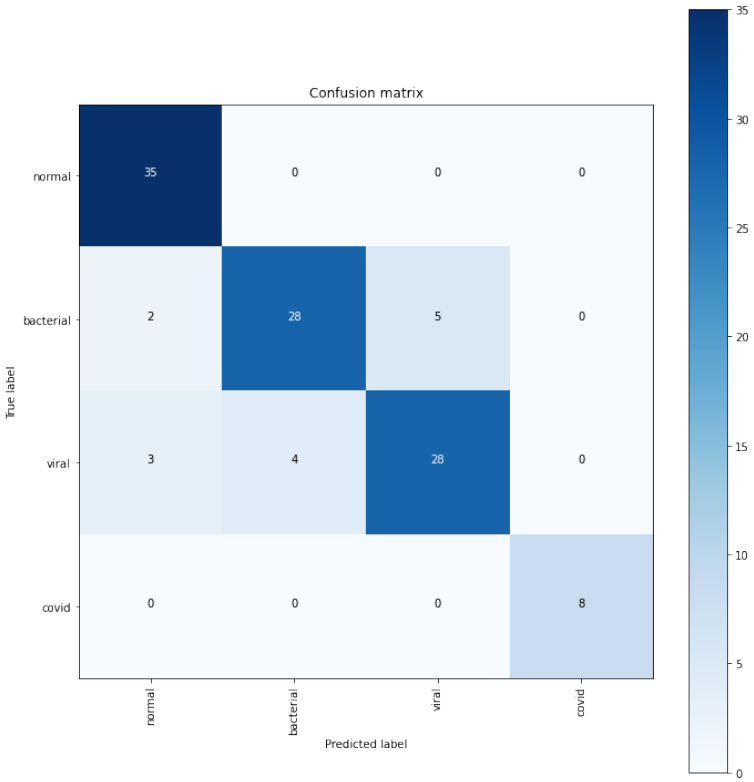
Below is the confusion matrix and scores for my CNN model that achieved 80% accuracy on my holdout test dataset and 79% on the Kaggle leaderboard.



Precision: 0.8065049333740208
Recall: 0.8053097345132744
F1: 0.8029295982383978

The confusion matrix shows that the two classes that get "confused" and misdiagnosed for the other the most are bacterial and viral. This is not surprising. What is surprising is that covid was not once misclassified or confused with another class. Given that it is a minority class and it does share lots of similarities with the bacterial and viral classes, the fact that it was not misdiagnosed perhaps suggests that there really are some key differences and features in the covid xrays that can be picked up by the classifier.

Here is the confusion matrix and scores for the transfer learning model using ResNet50 that achieved 87% accuracy on my test dataset and 86% on the Kaggle leaderboard.



Precision: 0.8756368999731832
Recall: 0.8761061946902655
F1: 0.8738400656788934

As shown, this model had better accuracy than the previous and was able to have less misclassifications of viral and normal. It again predicted correctly all the covid cases. However there were still a good amount of cases where viral and bacterial were misclassified as normal. This is the type of error that would be a real problem if this method were to be applied in a clinical setting.

Challenges

One of the biggest challenges for me during this project was knowing where to start; constructing a model wasn't the hard part, but rather it was knowing

how to go from jpg images in directories to organized, labelled, and correctly preprocessed data that proved to be the most important. There were so many different ways to do this, and I spent an unimaginable amount of time stuck training a bunch of different models and submitting predictions to Kaggle that wouldn't get past 25% accuracy (even though they were $> 70\%$ on my training and testing data). It ended up being because the test data had been shuffled before making predictions so I was submitting essentially random predictions. If I were to do this project again, I would definitely invest much more time in the beginning making sure I handled and processed the data correctly so that it actually looked like what I intended it to look like.

References

1. https://towardsdatascience.com/a-simple-cnn-multi-image-classifier-31c463324fa

   https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-to-cl

   https://blog.keras.io/building-powerful-image-classification-models-using-very-little-
   html

   https://www.kaggle.com/nikkonst/plant-seedlings-with-cnn-and-image-processing/
   comments

   https://www.kaggle.com/suniliitb96/tutorial-keras-transfer-learning-with-resnet50

   https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33

   https://www.pyimagesearch.com/2016/10/31/detecting-multiple-bright-spots-in-an-image-w

   https://medium.com/neuralspace/kaggle-1-winning-approach-for-image-classification-chal