COMS 4771 - Covid Kaggle Competition

John Daciuk
UNI: jpd2184

May 9, 2020

1 Exploratory Data Analysis

1.1 Visual Inspection

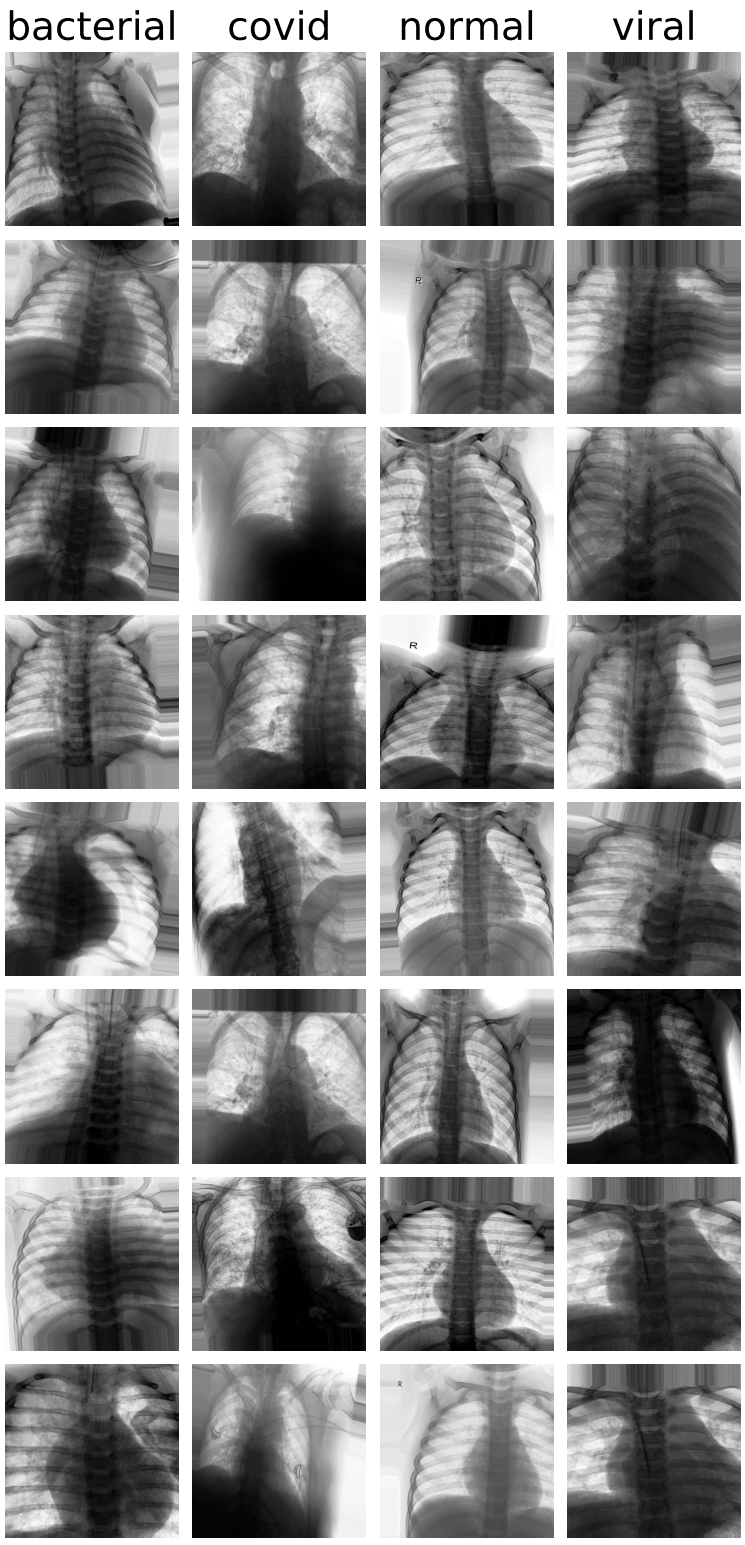

Figure 1: First look at covid-19 x-ray dataset

We begin by using the Keras data generator to reshape the images into 512×512 squares. In addition, we apply preprocessing to gives the images small random perturbations in rotation, brightness, shear, width shift, height shift and zoom level. Let's have a look at some random samples of each class, as shown in figure 1. The `covid` images tend to have a larger black area than the others. In addition, the `covid` images also tend to have rib cages that are over exposed. The `normal` images are probably the easiest to distinguish by eye from the others; they don't have the large dark areas seen in the others, which aligns with what we might expect. Perhaps these dark areas we see are caused by fluid build up in the lungs, which may be a symptom of many diseases. By contrast, `covid`, `bacterial` and `viral` seem fairly difficult to distinguish from one another.

1.2 Data Quantity

1.2.1 Challenges

In total we have 1139 training images. About 7% of them are `covid` and the rest are evenly distributed across the other classes. Considering the images each contain hundreds of thousands of pixels, we are saddled with far more features than observations. The curse of dimensionality is one of the central challenges of this dataset that we will try to overcome.

1.2.2 Solutions

One simple step in overcoming the curse of dimensionality is to resize the files, blurring neighboring pixels together. We can see the affect of different sizes on image clarity in figure 2 below.

Figure 2: The effect of resolution on image quality

Ultimately, we'd like to choose a resolution that is as low as possible which still preserves the distinguishing features of the 4 classes. 32×32 seems plausible, but is likely too low. By contrast, 256×256 likely adds too many details that a classifier is liable to overfit on. 64×64 and 128×128 are probably close to optimal choices.

The other strong tool we have to cope with the paucity of training data is image augmentation. As mentioned earlier, we use a variety of typical perturbations to the images that will encourage our models to generalize better. In general, perturbations such as zoom level and brightness are generally things we want our classifiers to ignore, so by feeding models with many augmented images, we may train them to not weigh these features heavily in class predictions.

2 Plan Overview

Since convolutional neural networks are well known to perform well for image classification and I've had experience with them, I decided to apply them to this project. I've found Keras to be the easiest library to use, while offering sufficient flexibility for projects of this scope, and so I committed early to Keras. The main plan was to train Keras CNNs and grid-search a reasonable hyperparamter space to narrow down what works best for this dataset and metric. Importantly, while there are about 4.5 times fewer `covid` training images than there are for each of the other classes, the test scoring metric weighs `covid` 5 times higher. To compensate for this, I used Keras class weights during model fitting to adjust the loss function appropriately. I then split the training data into 85% train and 15%

validation and tracked the `weighted class accuracy` metric while training. Training with the final metric of interest was a large part of my success because it allowed me to realistically evaluate models without having to dip into the test data. In addition, the use of early stopping allowed me to save weights during training that performed best on the test metric.

3 Training

To start, I decided to work with 64×64 pixel images for quicker results over a large grid of hyperparameters. To keep things simple, I used the same number of filters in each Conv2D layer. Across all experiments, I also used max pooling and dropout.

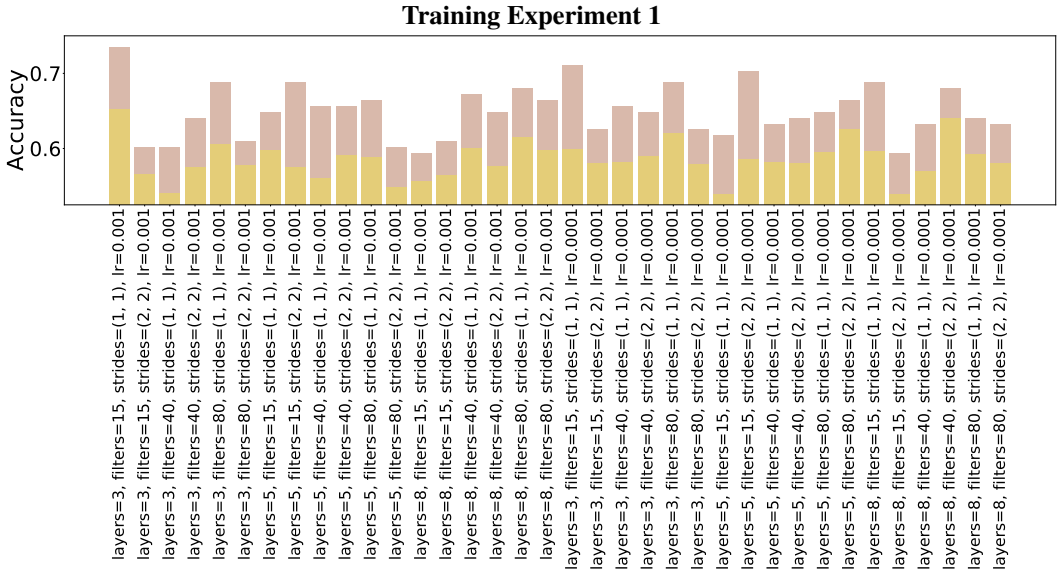

Figure 3: The effect of numbers of filters, layers, strides and learning rate on training for 64 x 64 image size. Accuracy weighted so that `covid` weighs 5 times more than the other classes. Training took place over 100 epochs and the batch size was 32. Yellow bars are best scores from a 7 epoch moving average, sienna bars are max scores across all epochs.

In figure 3 we can see that no single hyperparameter makes or breaks training performance. Across all learning rates, layers, filters and strides we see some of the lowest performance and some of the best performance, depending on the combination. The clear winner out of the bunch was the simplest of models: 3 Conv2D layers with 15 filters each, a single stride length and a standard learning rate of 0.001. This result makes intuitive sense; with such a paucity of training data, the stronger models introduce too much variance and cannot generalize as well. So from these observations, I continued by exploring the effects of other hyperparamters, edging on the side of simpler models all around. Never again would I try to throw 80 filters into each layer.

Figure 4: The effect of kernel size and image resolution on training. The lighter, lower mark, in each bar was best out of a 7 epoch moving average, the higher mark is best out of all epochs. The accuracy score is, as in figure 3, the weighted average we're also using in the Kaggle test. Learning rate was 0.0001 across tests.

Unfortunately, no image resolution will significantly boost performance above what we saw in figure 3 for 64×64 images. As expected, going to an image size of 256×256 does not seem to add much relevant detail that helps to classify images, and the high dimensions cause overfitting. On the other end, going all the way down to 64×64 may in fact start to blur some distinguishing features of the images. 128×128 seems to be the right middle ground, but we'll further probe this. As an additional benefit to using smaller resolutions, training is sped up significantly. Even at 256×256 training is quite slow, and much worse at 512×512. We can also see in figure 4 that the larger kernel sizes may be beneficial. We will continue by exploring more the effect of kernel size, filters, image size and now also optimizer. Perhaps with larger kernels and more filters per layer, 256×256 resolution could still have a shot at working well.

Figure 5: Experimentation with different optimizers, kernel size, image size and filters. The lighter lower mark in each bar was best out of a 7 epoch moving average, the higher mark is best out of all epochs. The accuracy score is, as in figure 3, the weighted average we're also using in the Kaggle test. Learning rate was 0.0001 across tests.

For the training with results shown in figure 5, we started with the kernel size shown in the top Conv2D layer and then gradually reduced kernel size down to 3. Unfortunately, more exploration of hyperparamters doesn't seem to significantly benefit performance. Earlier we used the Adam optimizer across experiments, now trying RMSProp in addition makes little difference. The results are also little changed by number of filters per layer. However, at this point it's fairly clear that an image size of 128×128 is near optimal, so we'll stick with this.

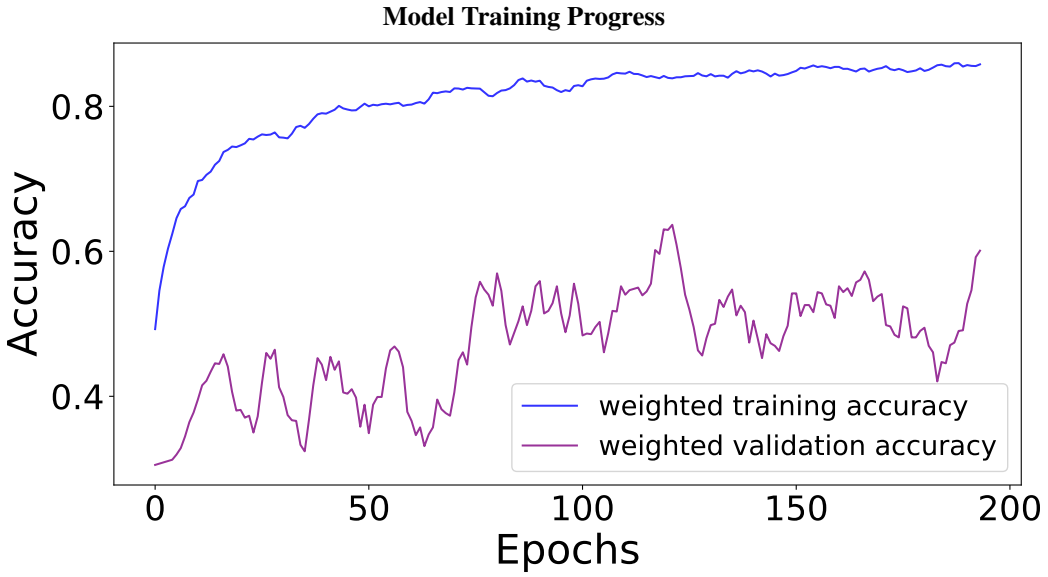

Figure 6: Final model training with 4 Conv2D layers, with 10 filters in each, kernel sizes 9,7,5,5 and 128×128 image resolution.

From grid searching hyperparamters, we learned that our choice does not have a huge impact on performance. Still, it became clear that a resolution of 128×128 is near optimal, higher kernel sizes can be helpful in the top layers, and we don't need an excessive amount of layers or filters, likely because we don't have enough data to support a model with too many parameters. We can already see above that training accuracy is substantially better than validation, so we can confirm that overfitting is a strong threat here.

In figure 6 we plot the training of our final model over 200 epochs with a learning rate of 0.0001. Again, our accuracy plot is really weighted just as the final test performance is on Kaggle so we can see real time results as the model trains. Performance on the training set continues to improve, but validation weighted accuracy levels off around 150 epochs. We did try other sessions with more epochs to no benefit. Interestingly, the validation scores are not stable; the model wavers quite a bit during training. Because of this, it was a great benefit to be able to use a callback in Keras to bring back the model weights from the highest validation score to finally submit predictions from.

4 Model Interpretability

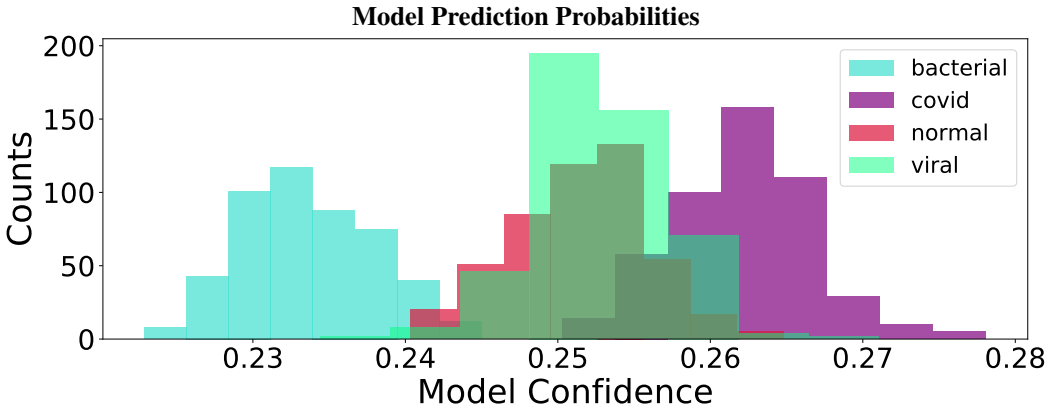

Figure 7: Histogram of model confidence

For the histogram above, for each test image we've asked the final model for the probability of each class. Most strikingly, all model probabilities are *really* close to 25%. That is to say, the model is very shy to claim it knows the class of any image much better than a random guess. For none of the images could the model even rule out a single class with confidence! Yet, when we color the confidences by class as in figure 7, we can see some of the model's proclivities. In particular, the model does generally assign high confidences to `covid`. This makes sense; the test metric weighs `covid` accuracy 5 times higher than anything else, even equalizing for class imbalance in quantity. To the extent that `viral` and `normal` overlap with the `covid` histogram, we can see how frequently the model would predict one of these classes over `covid`. It appears that more than 50% of images are predicted as `covid`. Because the `bacterial` histogram is completely below 25%, to the resolution visible in our bin size, the model never predicts this class, which leaves room for improvement since we expect this class to be as common as `normal` and `viral`.

The fact that our model tends to bias predictions towards covid-19 can certainly cause problems in the clinical setting if not properly understood. Clinicians would need to understand that in addition to artificially weighing `covid` predictions 5 times heavier, our model also compensates for their being far fewer `covid` cases in the training data. This is to the *detriment* of overall accuracy. In fact, patients in the real world do generally have a very low prior probability of having covid-19, so clinicians would need to understand how to apply Bayes rule to fully understand patients' posteriors after weighing all appropriate evidence. It's an unfortunate fact that when questioned, most doctors do not understand Bayes rule and often have the wrong intuition about how evidence from scans and models affects patients' likelihood of having certain diseases.

4.1 What does `covid` look like?

Although neural networks are notoriously difficult to interpret, we can have a look at which images the model is most confident are *not* `covid` and the images the model is most confident are `covid`. In this way, we can get a sense of what features the CNN picked up on to make reasonable predictions.

Images the model is *least* confident are `covid`

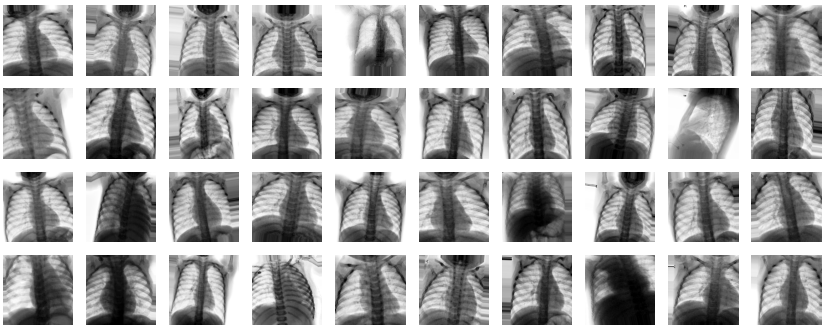

Figure 8: The images are mostly bright and relatively normal looking.

Images the model is *most* confident are `covid`

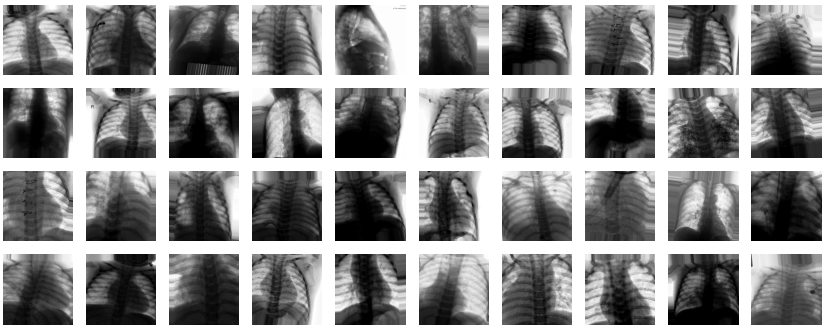

Figure 9: The images are much darker with large black patches.

Although our model's `covid` predictions' probabilities fell within a tight range between about 25% and 28%, the images it assigned near 25% confidence of `covid` make for a striking juxtaposition with the images that were assigned near 28%. We can see that the images the model predicts `covid` for tend to have large black patches, whereas the images that are predicted as the other classes largely do not. This gives us a pretty strong sense of what the model is actually focusing on to make predictions.

It's reasonable to assume that covid-19 may cause dark areas in the lungs in x-rays. It's likely that the virus causes fluid to build up, which could be the cause of the darkness we see. Because our model can pick up on these dark areas in the lungs, it could be useful to a clinician. For example, a doctor could use our model to warn patients with dark lung spots that they have higher than usual probability of covid-19, in the absence of more concrete testing availability. Models such as ours may also be useful in warning patients about increased risk while they await test results. The immediacy of x-ray results could be especially important since covid-19 often spreads while patients await slow test results. X-ray machines are becoming smaller and more available; in the future we can hope that virtually everyone in the world has access to x-ray imaging in their homes, at which point machine learning models will be particularly useful in extracting information from the myriad images taken everyday.

5 Conclusion

5.1 Lessons Learned

Having had significant experience with Keras and image classification in the past, I did not run into any hard to pin down bugs. I was, however, a bit over eager to start getting results. Consequently, I spent less time in the beginning looking at the data than I should have. For example, I started out training models with the image resolution at 512×512, whereas my plot of different resolutions in this report makes clear that this amount of detail is far more than needed to distinguish whatever features `covid` likely manifests, such as darkened lungs. Generally speaking, I find that I do get useful insight from very clean, organized, and systematic plots, such as my plot of images by class in the very beginning of this report. Yet, I often wait until towards the end of a project to make things more presentable. I also spent significant time training models with somewhat random hyperparameters before systematically grid searching and automating the process such that I could train many models and save them in an organized way with just a click. I also learned that while grid searching parameters, it makes more sense to use a higher learning rate and fewer epochs and to be very targeted about what hyperparameters to try, since run times can exponentially blow up over the space of parameters.

5.2 Final Thoughts

The final model predictions submitted to Kaggle scored about 77.5% on the competition's weighted accuracy metric. Considering a baseline model that always predicts the covid class scores about 26%, this is reasonable performance. The success of the model rests on good data preprocessing and augmentation techniques, a weighted loss function aligned to the final metric, grid-search to optimize hyperparameters and the power of CNNs to classify images. Albeit, image augmentation can only help to expand our training data by so much, and with such a low ratio of observations to features, neural networks struggle to make impressively confident and accurate predictions.