A Multi-class Classification Tool for COVID-19 and Bacterial Pneumonia Diagnosis using X-ray Imaging Data

Vivian Ngo
Department of Statistical Sciences
University of Toronto
100 St. George St
Toronto, Ontario
M5S 3G3, Canada
viv.ngo@mail.utoronto.ca

Andrew Hoan Tran
Department of Biostatistics
Dalla Lana School of Public Health
155 College St
Toronto, Ontario
M5T 3M7, Canada
andrewhoan.tran@mail.utoronto.ca

Abstract

As of July 2020, the global pandemic COVID-19 has infected over 15 million people and caused major social and economic change. Due to the novelty and rapid spread of the virus, the healthcare system has experienced significant strain in its ability to diagnose, treat, and manage patients who become infected. Currently, there is a limited capacity for diagnostic testing in Canada. Tests are recommended to individuals who are at high risk or may have been in contact with someone else with the infection. Given the importance of early detection for mitigating the spread of COVID-19, we propose alternative methods of diagnostic testing. Pneumonia may be present in severe cases of COVID-19, and can be identified by radiologists on x-rays. This project focuses on the classification of bacterial pneumonia, COVID-19, and healthy x-ray images using convolutional neural networks. A strong classification method will help to reduce the need for COVID-19 specific testing kits and reduce the strain on healthcare workers. Related code for this study is available on GitHub at COVID-19-X-ray-Classifier.

1 Introduction

In December 2019, early outbreaks of a novel coronavirus were reported in Wuhan, the capital of Central China's Hubei province (Sun et al., 2020). In just a few short months, this virus has been transmitted to over 15 million people and resulted in at least 300,000 mortalities worldwide (Rafiq et al., 2020). It has caused global isolation, placed unprecedented burdens on healthcare systems, and incapacitated world trade and travel. Today, we know this epidemic as coronavirus disease 2019 (COVID-19).

The official name of the virus which causes COVID-19 is severe acute respiratory syndrome coronavirus-2 (SARS-CoV-2) (Mayr et al., 2020). This virus belongs to a family of coronaviruses that caused severe acute respiratory syndrome (SARS) and Middle East respiratory syndrome (MERS). Given the transmission dynamics of COVID-19, the World Health Organization has urged countries to implement stringent infection control procedures including physical distancing, isolation, and quarantine (Aylward, Bruce (WHO); Liang, 2020).

In these harrowing times, the global preparedness of countries to respond to pandemics has been tested. In particular, the efficiency and resource allocation of hospitals and healthcare providers have become crucial. One critical step in the recovery process for a COVID-19 patient is the initial diagnosis. Testing and diagnosis are currently not recommended to all citizens and an incubation period of 14 days makes it difficult for people to make informed decisions about their health (Ontario, 2020).

Given the current shortage of resources, the ability to leverage existing resources can play a large role in protecting citizens. In particular, by leveraging imaging data and using computer-based methods to perform diagnoses, we can lessen the burden on health care workers and decrease the need for specialized COVID-19 tests.

Pneumonia is a severe and relatively uncommon complication of COVID-19 (Singhal, 2020). However, viral pneumonia can be difficult for clinical experts to distinguish from bacterial pneumonia based on x-ray imaging. Coming to a correct diagnosis using chest x-rays is important because the course of treatment for these two types

of pneumonia are very different (Kermany et al., 2018). The primary objective of this project is to use deep learning techniques to differentiate between medical imaging of healthy individuals, individuals with bacterial pneumonia, and individuals with COVID-19.

2 Related Works

To date, there are only two other peer-reviewed papers which address x-ray based COVID-19 classification. Researchers in Hong Kong used deep-learning based decision-tree classifiers to identify COVID-19 cases (Yoo et al., 2020). However, their study design was slightly different from our own as they defined only binary classifiers that differentiated healthy x-rays from those with tuberculosis or COVID-19. Their dataset contained 162 images of COVID-19 and 585 healthy images. Similar to our approach, these researchers also decided to perform image augmentation such as horizontal flipping and random rotations to increase their sample size. They were successful in their approach and the average accuracy of the binary tree that identifies COVID-19 was 95%.

Another group of researchers in Turkey used a transfer learning approach to multiclass classification of healthy, COVID-19, and pneumonia x-rays (Ozturk et al., 2020). They used the DarkNet model as their starting point and trained their model using the same set of bacterial and healthy x-ray images which we obtained from Kaggle. To avoid class imbalance problems, these researchers only randomly selected 500 healthy and 500 pneumonia x-ray images for training. A key difference between our approach and theirs is that they did not perform any form of data augmentation on their dataset. However, they were still able to generate a multi-class classification accuracy of 87.02%.

We will use these papers as the benchmark for model performance that we hope for our model to exceed.

3 Data

The data used throughout this project was collected from several sources. The first data source is (Cohen et al., 2020), from which we obtained x-rays and CT scans of roughly 200 patients with COVID-19. This dataset is updated regularly with the approval of the University of Montreal's Ethics Committee.

We also required medical images of healthy individuals and individuals with bacterial pneumonia. For this, we utilized a dataset from Kaggle of chest x-ray images for individuals with and without bacterial pneumonia (Mooney, 2018). This dataset contains more than 5,000 x-ray images all together.

3.1 Data Pre-processing

Before any analysis was completed, work had to be done to ensure consistency across the datasets that were used. To begin, the dataset of COVID-19 images contained a variety of radiographic images including CT scans and chest x-rays. To ensure consistent comparisons of images, we manually filtered the dataset to include only x-ray images. This was possible due to the relatively small size of our dataset. Moreover, the COVID-19 dataset also contained images of other viral pneumonias such as MERS and SARS. Images from patients with illnesses other than bacterial infection and COVID-19 were excluded from our analysis.

Another issue with the COVID-19 dataset was the presence of annotations (i.e. arrows or text) indicating anomalies present in the image. We believed this would introduce bias in our classifier, so these images were manually reviewed and removed from the dataset. For all images, a universal crop was applied to remove annotations or text that were present on the border or edges of the image. The ideal universal crop was determined through experimentation. It was performed by resizing the images to 1500 by 1500 pixels and then applying a centred cropping to 1200 by 1200 pixels. The images were then resized to 160 by 160 pixels, as this is the size required by many pre-trained models available in Keras. Upon inspection, we found that most of the images retained their quality and no longer had annotations after the crop.

4 Approach and Methods

4.1 Data Augmentation

Two issues that we encountered with this dataset were low sample size and class imbalance. Since the COVID-19 epidemic is still in its infancy, there is currently a paucity of x-ray images available from patients with the disease. In our dataset, images are classified as bacterial (N = 2,780), COVID-19 (N = 391), or healthy (N = 1,583). These are also referred to as class 0, 1, or 2,

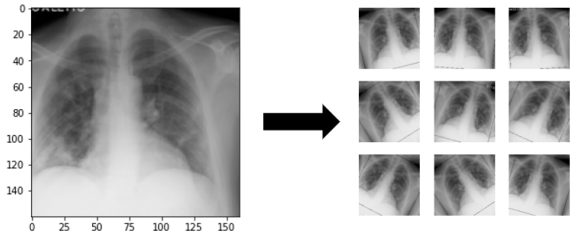

Figure 1: Data Augmentation on a single x-ray image in the dataset. X-ray images were horizontally flipped or rotated by up to 36 degrees clockwise or counter-clockwise.

respectively, in our model. The smallest class is nearly approximately 7 times smaller than the largest class and comprises less than 10 percent of the entire dataset. As a result, we decided to apply data augmentation techniques to artificially inflate the sample size and rebalance the classes. Data augmentation is commonly utilized for deep learning model image classification to improve classification accuracy and reduce overfitting (Singhal, 2020).

There are a variety of data augmentation methods available to machine learning researchers. The two types that we used in this project were basic image manipulation and oversampling. Examples of basic image manipulation include flipping, cropping, translation, color space transformation, or rotation. We decided to use only horizontal flipping and rotation because these manipulations were the most appropriate for our dataset while still preserving the image labels. The images were either flipped horizontally or randomly rotated up to 36 degrees. An example of these manipulations applied to a single image in our dataset is shown in Figure 1.

For oversampling, we used a popular algorithm called Synthetic Minority Oversampling TEchniques (SMOTE) to generate more samples of the minority class. In brief, SMOTE works by generating augmented samples which share feature similarities with other samples of the same class (Chawla et al., 2002). Assuming that the data has points belonging to minority and majority classes, SMOTE works as follows: a) select a point from the minority class called the seed, b) compute the k-nearest neighbours from the same class by Euclidian distance, c) randomly select one of the computed neighbours, d) generate an "augmented" sample between the seed and selected neighbour, e) repeat steps a-d until the desired number of samples are generated. For our purposes, we used SMOTE to obtain equal sample sizes for all three classes.

4.2 Models

For our classification task, we considered two general types of models: from-scratch neural networks and pre-trained transfer learning models.

Although it is typically recommended to use pre-trained transfer learning models, especially for models with a small dataset, we wanted to try a more simplistic model from scratch to begin with. The first entry block of our from-scratch network architecture contains rescaling, convolutional 2D, batch normalization, and relu activation layers. This entry block is then set aside as a residual block. We then stacked another sequence of relu activation, separable convolutional 2D, batch normalization, and max pooling layers. The size of the convolutional 2D layers increased incrementally from 128 to 728 units. The residual layer is then added back to the model and an additional global average pooling layer is included. The final layer of our model is a dense layer with 3 units and softmax activation to make predictions about the probability of an image belonging to a particular class. The most notable features of the from-scratch neural network are that it contains "branches" and is not purely sequential. It also contains many convolutional 2D and batch normalization layers to improve the speed, performance, and stability of our model. The exact configuration of the from-scratch neural network is available at the project GitHub.

As an alternative to from-scratch neural networks, we also decided to experiment with pre-trained models. The rationale behind this decision was that because we have relatively little data to train the model, its capacity to learn from these few images may be limited (Raghu et al., 2019). Pre-trained models, on the other hand, are trained on large datasets and are already able to perform image classification tasks successfully, even if our dataset is small. One example of a popular dataset used to train pre-trained models is the ImageNet dataset. ImageNet contains more than 14 million images belonging to 1,000 categories. Pre-trained model architectures such as MobileNet and ResNet are available in Keras and have the op-

tion of being loaded with weights from training on these large datasets.

We chose to use the MobileNetV2 architecture for several reasons. Although larger architectures like VGG19 can perform better than smaller architectures like MobileNet, they also contain more parameters in the order of several magnitudes (Chollet et al.). We wanted to balance predictive power with simplicity to reach a parsimonious model. Computation-wise, a simpler model would also require less time and computational resources.

Given that our classification task is different from the task performed by the pre-trained model, we are using a transfer learning approach to generate a starting point for our model. We improved on the pre-trained MobileNetV2 model by adding our own global pooling and prediction layer, as well as experimenting with the hyperparameters to determine the optimal network configuration.

4.3 Parameter Tuning

During our experiments, it was important to try different parameters. Our goal was to find a set of hyperparameters that would give a well-performing and stable model. Some examples of parameters that we tuned include the learning rate, number of epochs, steps per epoch, and class weights.

First of all, the learning rate is important to tune because a learning rate that is too small can cause the model to require a lot of time and epochs to optimize the loss function. However, a learning rate that is too large may miss optimal solutions by proposing moves that are too large. Moreover, models with small datasets are especially sensitive to changes in learning rate, which is why we decided to experiment with this parameter (Mishra et al., 2019).

The number of epochs and steps per epochs are also important to tune because if these values are too low then the model may not have had enough time to reach an optimal solution. However, if these values are too large then the model would risk overfitting to the training data.

Finally, because of the class imbalance in our dataset, it was possible for the model to classify a lot of images incorrectly into one or two categories and still have a high overall accuracy simply because the last class had a small sample size. We ran into this issue even after implementing

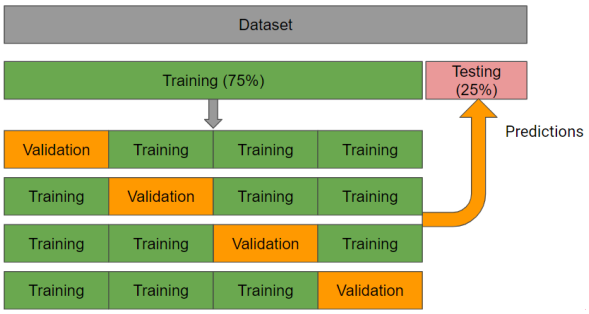

Figure 2: Cross-validation strategy for training and testing models. There was 75% of data set aside for training and 25% for testing.

SMOTE. Therefore, class weights was useful to use in order to rebalance the classes so that the loss function would take all three classes into account equally during training. These are some of the main parameters that we tuned.

5 Evaluation

5.1 Stratified K-Fold Cross Validation Training Strategy

To evaluate the performance of our models, we employed a stratified cross-validation strategy because it helped us to identify issues of overfitting which tends to be a problem with smaller datasets (Kohavi, 1995). K-Fold cross validation can also lead to increased precision in performance metric estimations and give the experimenter a better idea of the generalizability of their model.

The reason for using specifically a stratified k-fold strategy is because we wanted our folds to have class proportions that were equal to the total class proportions of the data, creating representative samples of all our data. This was especially important for our dataset as our classes are imbalanced. The class imbalance makes it more probable to have folds that only contain two of the three classes. Stratification ensured that this issue would not arise so that we could perform data augmentation within each fold to oversample (or undersample) images of desired classes. We also made sure to split our data into k folds before performing data augmentation so that we would not have related images in the testing and validation set and potentially inflate the model performance metrics.

Before training, we split our entire dataset into 75% training and 25% test. The training set was further split into k folds, where the k-th fold was

used for validation and the remaining folds were used to train the model. The test set was set aside and not used at any point during the training and tuning of our models. We reserved the test set to measure our model performance after the final model was selected.

5.2 Performance Metrics

During model training, it was important to have metrics to help us gauge whether the changes that we were making were indeed improving the model performance. In order to evaluate the performance of the model after tuning, we used several popular metrics for multi-class classification including recall (Equation 1), precision (Equation 2), and F1-score, the harmonic mean of precision and recall (Equation 3) (Sokolova and Lapalme, 2009).

$$Recall_M = \left(\Sigma_{i=1}^{l=3} \frac{TP_i}{TP_i + FN_i}\right)/3 \quad (1)$$

$$Precision_M = \left(\Sigma_{i=1}^{l=3} \frac{TP_i}{TP_i + FP_i}\right)/3 \quad (2)$$

$$F1score_M = \frac{(\beta^2 + 1)Precision_M Recall_M}{\beta^2 Precision_M + Recall_M} \quad (3)$$

where:

- TP_i= true positives for class i
- FN_i = false negatives for class i
- FP_i = false positives for class i
- β is a non-negative real number

For the F1-score, we employed a β value of 1 as it is said to balance recall and precision. In our specific application, we valued recall, the proportion of positive cases that were correctly identified, because it is important to be able to diagnose illnesses such as COVID-19 and bacterial pneumonia. However, it was also important for us to be precise in our diagnoses because resources are scarce during a global pandemic and it is costly to allocate medical resources to people who are not truly ill. Therefore, the main metric that was used to determine the final model was the macro average F1-score (Sokolova and Lapalme, 2009). For every model, we took the average of all of its macro average F1-scores (there is one per fold) and used this final metric to compare with other models.

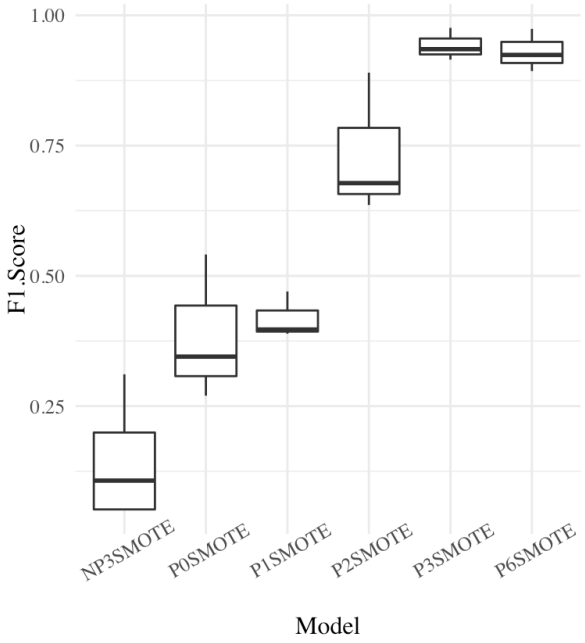

Figure 3: F1-scores obtained from cross-validation of 6 different models. NP refers to not pre-trained and P refers to pre-trained models. Models ending in SMOTE had Synthetic Minority Oversampling TEchnique applied to the training set.

6 Results

The results of our experimentation can be seen in Table 1 and Figure 3. In total, we report the results of six models which vary in their base model architecture or hyperparameters. Table 1 shows the F1-scores across k folds for each model. As a note, additional models were trained but their results have been omitted for the sake of clarity and conciseness. The adjustments made between subsequent models were motivated by weaknesses identified in preceding models.

6.1 From-Scratch Neural Network (NP3SMOTE)

One of the best-performing from-scratch models that we experimented with is a model that we denote as NP3SMOTE. NP3SMOTE was trained using 100 epochs and a learning rate of 0.005. For this model, we used SMOTE to oversample the minority classes until all classes had an equal sample size in the training set but did not implement class weights. The specific architecture for the non-pretrained models that we ran is described in further detail in the methods section. We tested it

Table 1: Model parameters and F1-scores obtained from cross-validation of 6 different models. NP refers to not pre-trained and P refers to pre-trained model. P models ending in SMOTE had Synthetic Minority Oversampling TEchnique applied to the training set.

Model	Change	Macro Average F1-Score for Fold				Average F1-Score
		1	2	3	4	
NP3SMOTE		0.311	0.052	0.162	0.051	0.144
P0SMOTE	+pretrained +decreased lr	0.541	0.345	0.27	-	0.385
P1SMOTE	+class weights	0.47	0.389	0.397	-	0.419
P2SMOTE	+adjusted class weights	0.89	0.678	0.636	-	0.735
P3SMOTE	+increased steps per epoch	0.935	0.976	0.915	-	0.942
P6SMOTE	-decreased steps per epoch -decreased epochs	**0.924**	**0.893**	**0.974**	**-**	**0.930**

with 4-fold cross validation and measured performance using the validation set in each fold. Evidently, from Table 1, the performance appears to be very poor across all of the folds. The macro average F1-scores ranged from 0.052 in the worst fold to 0.311 in the best fold, and the average of the macro average F1-scores was 0.144. The poor performance is likely due to the limited size of our dataset. After experimenting with parameter tuning and not seeing much improvement, we decided to use transfer learning.

6.2 Pre-trained Model and SMOTE

P0SMOTE was the next model tested, and it was a pre-trained model with a MobileNetV2 architecture and weights initialized from ImageNet. We chose to use a pre-trained model to do transfer learning because we believed that our dataset was too small to train a reasonable model. MobileNetV2 offered adequate accuracy on the ImageNet validation set while also being a smaller model with fewer parameters. This model, as well as all subsequent models, were trained and validated using 3-fold cross validation because there was concern that low sample size would cause too few COVID-19 images to be in each fold. Compared to NP3SMOTE, this model also featured SMOTE but used a decreased learning rate because previous experiments had suggested that a smaller learning rate led to better performance. The performance in this model was better than NP3SMOTE. However, the overall performance is still poor with F1-score ranging from 0.27 to 0.541 and an average F1-score of 0.385 across the folds.

6.3 Pre-trained Model and SMOTE: Class weights

Upon investigating the confusion matrices generated by the predictions of the model P0SMOTE, we found that nearly all of the images were classified as class 0 (bacterial) or class 1 (COVID-19). This is likely caused by two factors. The first factor is that class 0 has a larger sample size, likely causing the loss function to weigh this class more heavily. The second factor is that class 1 is a minority class so SMOTE caused the model to also prioritize this class. Meanwhile, no images would be classified into class 2 because the cost of misclassifying into class 0 or 1 was not very large.

In order to balance the importance of predictions made for each class, we introduced class weights which were inversely proportional to the sample sizes of each class. These class weights were used in the P1SMOTE model. The weightings were 4, 26, and 7 for class 0, 1, and 2 respectively. After specifying the class weights, the performance of P1SMOTE increased in comparison to P0SMOTE with an average F1-score of 0.419. It is also interesting to note that there was less variance in the F1-score across the k folds compared to previously tested models, as can be seen in Figure 3.

Although P1SMOTE showed an improvement from the previous model, class 2 (healthy) was still being under-weighted as very few images were being classified in to this group. Thus, we adjusted

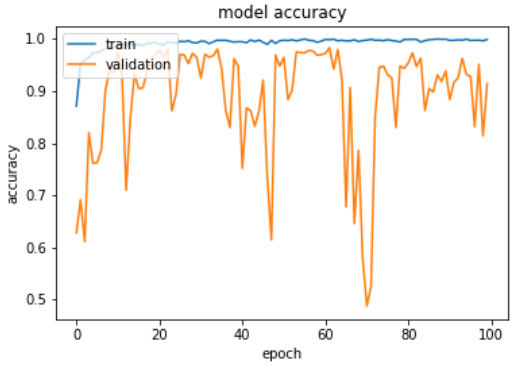

Figure 4: Accuracy of P3SMOTE for one example fold. It can be seen that the accuracy fluctuates and does not have an overall increasing trend.

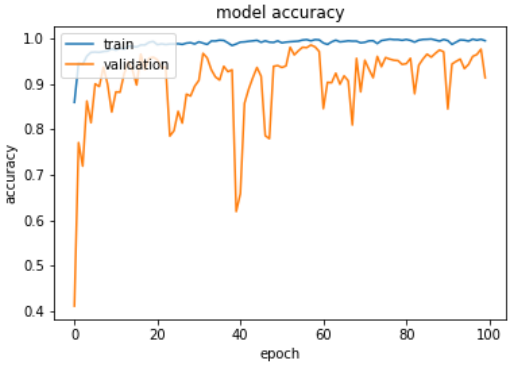

Figure 5: Accuracy of P6SMOTE for one example fold. It can be seen that the accuracy is much more stable compared to P3SMOTE.

the class weights further to give more weighting to this class. The weightings were adjusted to 4, 26, and 12 for class 0, 1, and 2 respectively, as opposed to 4, 26, and 12. We called this model P2SMOTE, and it had the most dramatic improvement in performance yet, with an average F1-score of 0.735. The reason behind this performance increase is likely because the adjusted class weights incentivized correct predictions of class 2, thus leading to a higher F1-score. At this point, we were satisfied with the class weight adjustment.

6.4 Pre-trained Model and SMOTE: Steps per epoch

In all of the models discussed so far, the steps per epoch for each model was arbitrarily set to 2. After consulting educational deep learning resources, we discovered that the steps per epoch is commonly calculated as the size of the training set divided by the batch size. This equated to approximately 130 steps per epoch for our dataset with a batch size of 32. This model is referred to as P3SMOTE.

After this proposed increase in the steps per epoch, the average F1-score rose to 0.942. This is by all means excellent performance. However, examination of the training history revealed unusual behavior. We found that the training and validation accuracy in P3SMOTE was highly fluctuating over the epochs, which can be seen in Figure 4. This was alarming because it was not an issue with previous models.

We determined that the issue was likely due to the increased number of steps per epoch since this was the only change from the previous model. So, the steps per epoch was decreased from 130 to 100. The number of training epochs was also reduced from 250 in the preceding models to only 100 epochs in order to prevent possible overfitting. This adjusted model is referred to as P6SMOTE. As shown in Table 1 and Figure 3, the performance of P6SMOTE is comparable to P3SMOTE with an average F1-score of 0.930 compared to 0.942. In contrast, the training history of P6SMOTE was a lot more stable and had fewer fluctuations than what was observed in P3SMOTE, as can be seen in Figure 5.

6.5 Final Model Selection

The final model that we selected was P6SMOTE because of its high macro average F1-scores as well as its stable increasing accuracy. This model was trained with a learning rate of 0.0001, 100 epochs, 100 steps per epoch, SMOTE, and class weights of 0: 4, 1: 26, 2: 12.

6.6 Final Model Evaluation

We used the final model to make predictions on the reserved test set, which comprised 25% of our total dataset. The x-rays in the test set were not used in any form during the training and validation process. The precision, recall, F1-score, accuracy and confusion matrix of our model is displayed in Figure 6. The results are positive, with a macro average precision, recall, and F1-score of 0.972, 0.959, and 0.965 respectively. The confusion matrix also shows that no classes are ignored when making predictions. This informs us that our

	precision	recall	f1-score	support	pred_0	pred_1	pred_2
class 0	0.975	0.971	0.973	687	667	1	19
class 1	0.989	0.937	0.962	95	5	89	1
class 2	0.952	0.970	0.961	406	12	0	394
accuracy	0.968	0.968	0.968		-	-	-
macro avg	0.972	0.959	0.965	1188	-	-	-
weighted avg	0.968	0.968	0.968	1188	-	-	-

Figure 6: Classification report and contingency table of the final P6SMOTE model on the reserved test set. The precision, recall, F1-score, and accuracy are reported for each class and the confusion matrix is shown in the last 3 columns. The confusion matrix is interpreted with true labels in the rows and predictions in the columns. (pred_i refers to images that were predicted into class i). Class 0, 1, and 2 refer to bacterial, COVID-19, and healthy cases respectively.

model appears to be generalizable to new unseen data.

Our final accuracy of 96.8% also outperforms other recent studies that use x-ray imaging to detect COVID-19. With a small sample size of 4,754 and incredibly imbalanced classes (2,780 bacterial, 391 COVID-19, and 1,583 healthy images), it is quite remarkable that our model was able to perform a multi-class classification task with such high performance.

7 Discussion

In this section, we will discuss some main takeaways of this project. As mentioned above, our model performed very well considering the data limitations that we faced and the multi-class classification task that we required the model to perform. This is due to several important factors.

First of all, it was essential to use transfer learning to aid in model training. Although attempts were made to build a model from scratch, all of those models performed very poorly. It was quickly learned that with such a small dataset, it is unfeasible to train an image classifier without having to run the model for a substantial amount of time and epochs, which can become incredibly costly. Alternatively, it is much easier and computationally less costly to implement pre-trained models and perform transfer learning.

Another important takeaway is that accuracy is not always the best measure of model performance, especially when the classes are imbalanced. When the size of one class is much greater than another, the model can optimize the loss function by categorizing all items into the larger class at very little cost. In this case, it is useful to build a custom loss function or use class weights to rebalance the loss function to put more emphasis into the minority classes.

Finally, it is not sufficient to simply assign class weights the values of the inverted counts per class. This can be especially problematic when additional techniques are combined with the reweighted loss function. For example, the final model in this project used both SMOTE and class weights. However, the implementation of SMOTE affected the prioritization of classes as well so that class weights that may have worked well before may no longer do so. Therefore, it is important to assess the effects of these techniques in unison and always look at the core metrics such as the classification report and contingency table.

Overall, this project has shown that well-performing deep learning models can definitely be built even with a small dataset. However, care must be taken to choose a good model to start with, ensure that the performance metrics are a good measure of the model performance and verify that the loss function is weighing all classes well.

8 Limitations

We acknowledge that there are some limitations to our project. First is the heterogeneity of our dataset. Given that all of the COVID-19 images belong to a dataset that is different from the bacterial and healthy images, there is a chance that there is unaddressed bias in our analysis that is caused by the different sources. For example, visual inspection of the dataset shows that the bacterial and control images look more uniform and have a dis-

tinctive color gradient compared to COVID-19 images. As a result, the model may be picking up on these non-biological differences rather than the detection of pneumonia to classify the images.

Another limitation that we face is the issue of small sample size and limited data sources, particularly for COVID-19 images. Although we have used many methods to mitigate the effects of a small sample size, this issue could still have an impact on the generalizability of our model. If our small dataset is unrepresentative of all x-ray images then the test set could also be biased and overestimate the performance of the model even if the model performs poorly when classifying new images.

Although these issues are inherently due to the privacy concerns in the healthcare industry, if this data becomes more publicly available in the future, it would be useful to train the model again on a larger dataset with a larger variety of data sources for all classes.

Another consideration for future studies is to use larger pre-trained models. In this study, we opted for MobileNetV2 for purposes of parsimony. However, it could be worthwhile to pursue some more complicated models given the resources. It would be important to be careful of overfitting but a more complicated model does have the potential to increase performance. In this particular case, a small model improvement could help to save many lives.

References

Wannian (PRC) Aylward, Bruce (WHO); Liang. 2020. Report of the WHO-China Joint Mission on Coronavirus Disease 2019 (COVID-19). *The WHO-China Joint Mission on Coronavirus Disease 2019*, 2019(February):16–24.

NV Chawla, KW Bowyer, LO Hall, and WP Kegelmeyer. 2002. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 2002:321–357.

Francois Chollet et al. Keras Applications.

Joseph Paul Cohen, Paul Morrison, and Lan Dao. 2020. Covid-19 image data collection. *arXiv 2003.11597*.

Daniel S. Kermany, Michael Goldbaum, Wenjia Cai, Carolina C.S. Valentim, Huiying Liang, Sally L. Baxter, Alex McKeown, Ge Yang, Xiaokang Wu, Fangbing Yan, Justin Dong, Made K. Prasadha, Jacqueline Pei, Magdalena Ting, Jie Zhu, Christina Li, Sierra Hewett, Jason Dong, Ian Ziyar, Alexander Shi, Runze Zhang, Lianghong Zheng, Rui Hou, William Shi, Xin Fu, Yaou Duan, Viet A.N. Huu, Cindy Wen, Edward D. Zhang, Charlotte L. Zhang, Oulan Li, Xiaobo Wang, Michael A. Singer, Xiaodong Sun, Jie Xu, Ali Tafreshi, M. Anthony Lewis, Huimin Xia, and Kang Zhang. 2018. Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning. *Cell*, 172(5):1122–1131.e9.

Ron Kohavi. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, page 1137–1143, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

V Mayr, Dobrescu Ai, A Chapman, E Persad, I Klerings, G Wagner, U Siebert, C Christof, C Zachariah, G Gartlehner, V Mayr, Dobrescu Ai, A Chapman, E Persad, I Klerings, G Wagner, U Siebert, C Christof, and C Zachariah. 2020. Measures to control COVID-19 : a rapid review.

Sourav Mishra, Toshihiko Yamasaki, and Hideaki Imaizumi. 2019. Improving image classifiers for small datasets by learning rate adaptations. *Proceedings of the 16th International Conference on Machine Vision Applications, MVA 2019*.

Paul Mooney. 2018. Chest x-ray images (pneumonia).

Public Health Ontario. 2020. Coronavirus disease 2019 (covid-19) testing.

Tulin Ozturk, Muhammed Talo, Eylul Azra, Ulas Baran, and Ozal Yildirim. 2020. Automated detection of COVID-19 cases using deep neural networks with X-ray images. *Computers in Biology and Medicine*, (January).

Danish Rafiq, Asiya Batool, and M A Bazaz. 2020. Three months of COVID-19: A systematic review and meta-analysis. *Reviews in medical virology*, (April):e2113.

Maithra Raghu, Chiyuan Zhang, Jon Kleinberg, and Samy Bengio. 2019. Transfusion: Understanding Transfer Learning for Medical Imaging.

Tanu Singhal. 2020. A Review of Coronavirus Disease-2019 (COVID-19). *Indian Journal of Pediatrics*, 87:281–286.

Marina Sokolova and Guy Lapalme. 2009. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427 – 437.

Pengfei Sun, Xiaosheng Lu, Chao Xu, Wenjuan Sun, and Bo Pan. 2020. Understanding of COVID-19 based on current evidence. *Journal of Medical Virology*, 92(6):548–551.

Seung Hoon Yoo, Hui Geng, Tin Lok Chiu, Siu Ki Yu, Dae Chul Cho, Jin Heo, Min Sung Choi, Il Hyun Choi, Cong Cung Van, Nguen Viet Nhung,

Byung Jun Min, and Ho Lee. 2020. Deep Learning-Based Decision-Tree Classifier for COVID-19 Diagnosis From Chest X-ray Imaging. *Frontiers in Medicine*, 7(July):1–8.