50.039 Small DL Project Report

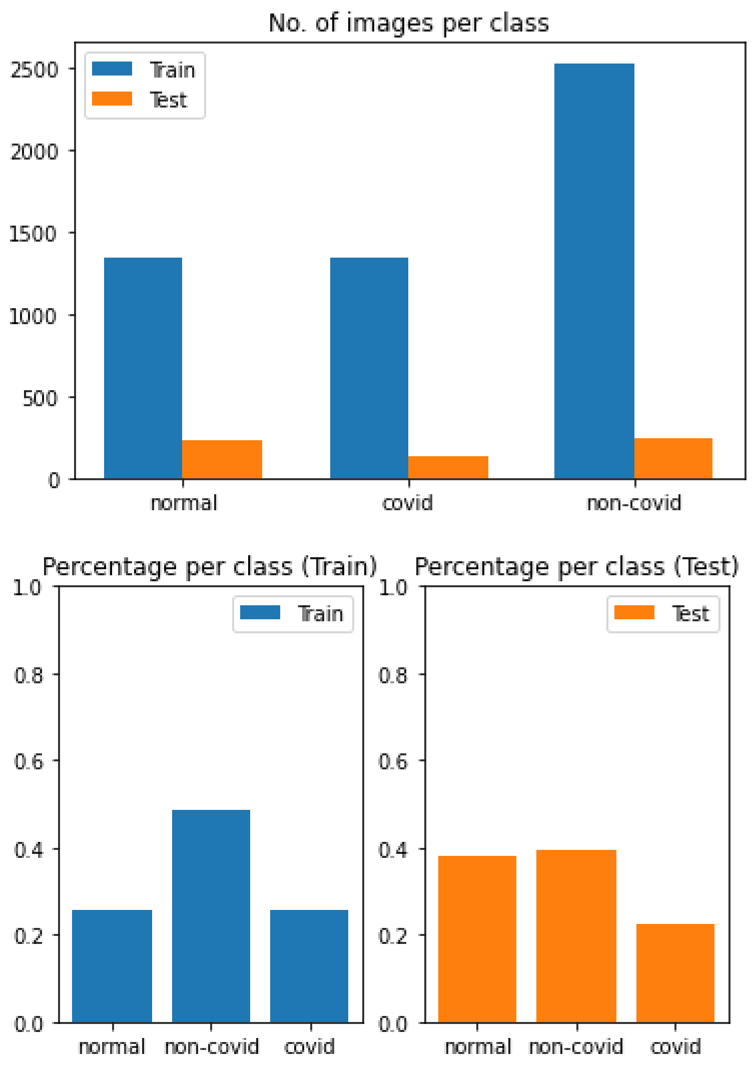
Loh De Rong (1003557), Koh Hui Wen (1003593)

Dataset and Dataloader

Provide graphs showing the distribution of images among classes and discuss whether or not the dataset is balanced between classes, uniformly distributed, etc.

From Figure 1, there are much more infected than normal images for both train and test set. In particular, for the train set, there are much more non-COVID images than normal or COVID images, thus the train dataset is not very balanced. As a result, the model may be more biased toward predicting the infected and non-COVID classes.

On the other hand, for the test set, there are much less infected COVID images than normal or infected non-COVID images. The percentage of normal cases is higher than in the train set, so some normal cases may be wrongly predicted as infected and non-COVID.



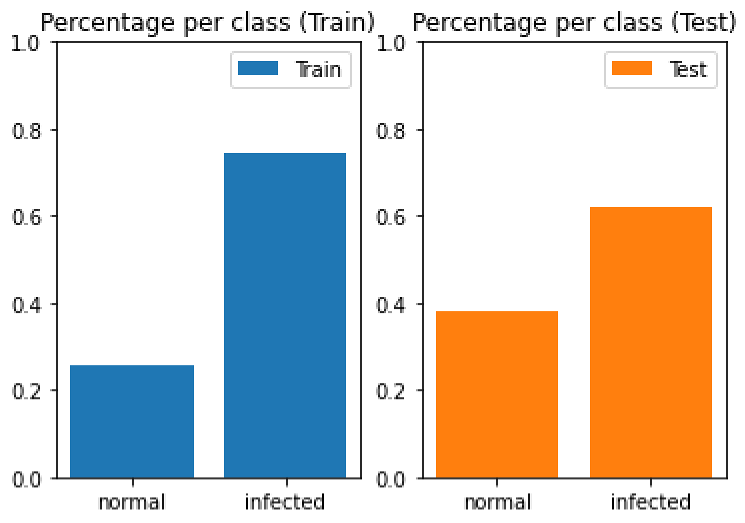


Figure 1. Class distribution graphs

Discuss the typical data processing operations applied to your images (e.g. normalization). Why did we need it? Are there other preprocessing operations that are needed for our images?

Normalization

The pixel values of an image range between 0 and 255, which is very large. Normalization ensures that this range is between 0 and 1, which is smaller. When the normalized values are being fed as inputs to the neural network, it ensures that the outputs are more uniform across the neurons, thereby speeding up learning and leading to faster convergence.

The normalization operation is applied to both the train, val and test images during data preparation.

Resizing

The image size given in the train set is already (150, 150). However, we still implemented the resize function during the preprocessing step to account for any new images that are added to the dataset which do not fit the dimensions.

The resizing operation is applied to both the train, val and test images during data preparation.

Data Augmentation

The transforms include random brightness and contrast, as well as small rotations. It is only applied to the original train images at every batch generation, which means only the batch images are copied and transformed every iteration.

By modifying the train samples in each epoch, data augmentation can help provide different views of the same sample and thus reduce model overfitting. The results of the data augmentation are discussed in the Bonus section, with reference to Figures 7 and 8.

Loh De Rong (1003557), Koh Hui Wen (1003593)

Proposed Model

Discuss the differences between the two architectures above and defend which architecture you decided to go for and why.

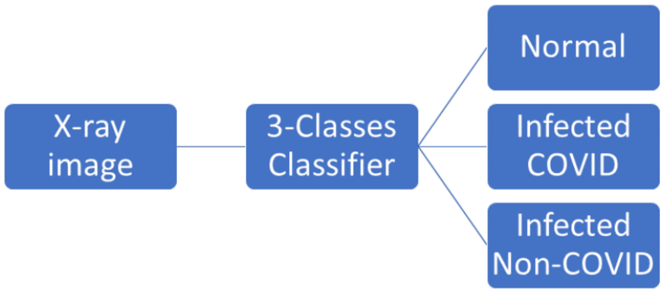


Figure 2. Architecture of a 3-classes classifier model

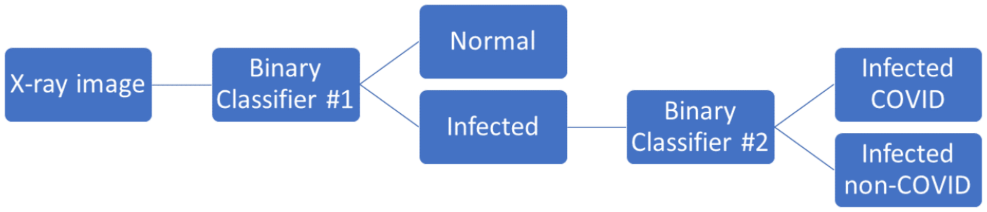


Figure 3. Architecture of a 2 binary classifiers model

Differences between the 2 architectures

2 binary classifiers model	**3-classes classifier model**
It consists of 2 separate 2-class classifiers. Binary Classifier #1 predicts an input image as either Normal or Infected. Binary Classifier #2 predicts an input image as either Infected COVID or Infected non-COVID.	It is a single 3-classes classifier model, which simply predicts if the input image is Normal, Infected COVID, or Infected Non-COVID.
Both the classifiers have to be trained separately. The trained classifiers can then be stacked together as a combined model for prediction.	Model training and prediction can be done end-to-end.
The prediction is a two-step process, where Binary Classifier #2 only takes in images which are predicted as Infected by Binary Classifier #1, for more fine-grained classification.	The prediction is a one-step process, where it outputs the class with the highest softmax probability.

Why may choosing the 2 binary classifiers model be better?

The advantage of implementing this model builds on a reasonable assumption that there exists some pattern for the COVID and non-COVID samples belonging to the infected class.

A three-classes classifier would have to pick up distinguishing features that separate the normal, COVID and non-COVID classes. However, based on the earlier assumption, features of the COVID and non-COVID samples might be similar. Hence, a three-classes classifier might not learn these important features that characterize an infected patient because they do not help the model differentiate the infected COVID and infected non-COVID from the normal samples.

On the other hand, using a 2 binary classifiers model allows for binary classifier #1 to focus on the features that can distinguish between normal and infected samples, and for binary classifier #2 to focus on the more subtle differences between the COVID and non-COVID samples. Such an approach could possibly yield higher performance.

Why may choosing the 3-classes classifier model be better?

Choosing a 3-classes classifier might work better due to class imbalance of the dataset (refer to Exploration of Dataset.ipynb). In terms of distribution, 25.7% are normal images, 25.7% are infected COVID images, 48.6% are infected non-COVID images.

The upper bound performance of the 2 binary classifiers model is limited by the performance of the first binary classifier, which classifies samples into either normal or infected. In the train set, there are significantly more infected images (74.3%) than normal images (25.7%). Hence, the model might be biased toward predicting infected classes, leading to poor model performance. Furthermore, the performance will be further reduced by the second binary classifier, which classifies samples into either COVID or non-COVID samples. This is because there is also class imbalance between the infected COVID samples (34.6%) and infected non-COVID samples (65.4%) in the train set, which likely leads to prediction results favouring infected COVID samples.

Final choice

However, we decided to overcome the class imbalance by using weighted loss functions. Thus, there are more advantages to using the 2 binary classifiers, since each model can focus on specific features that distinguishes between two classes instead of three.

Discuss the reasons behind the choice of evaluation metrics used

In addition to loss and accuracy, we decided to include F1 score in our evaluation metrics. This is because the dataset is imbalanced (refer to Figure 1), but the detection of infected and COVID classes is very important. It is possible for the model to have high accuracy (predict most samples as non-COVID, since majority of the test samples are non-COVID), but doing so may result in misclassification of COVID samples.

It is dangerous for COVID samples to be labelled as non-COVID, i.e. the cost of False Negatives for the COVID class is high. Hence, accuracy alone is not enough and the recall score should be taken into account. However, it would be bad to only rely on the recall score, since the model could simply predict every input sample as infected and COVID and get a 100% recall score. Thus, the precision score should also be considered. As such, we decided to use the F1 score, which is a balance between the recall and precision scores. The goal is to achieve a high F1 score, since it would mean that our model is generally not misclassifying COVID samples as non-COVID and at the same time, is rather precise such that it does not label everything as COVID.

Discuss the reasons behind the choice of architecture (e.g. types of layers, and their parameters).

To obtain good results for the 2 binary classifiers model, we need to train the two classifiers individually. Thus, finding the optimal architecture for each classifier will result in an optimal overall model.

<u>More convolutional layers and increasing channel depth</u>
The choice of using convolutional layers is intentional because we want to preserve the spatial features that are often clustered in the same vicinity. In the case of pneumonia, areas of white spots will be of interest.

We experimented with the number of convolutional layers to use, recording the test performance of each architecture in Table 1.1 (for the Normal vs Infected classifier) and Table 2.1 (for the COVID vs Non-COVID classifier) below. For all convolutional layers, there is padding of size 1 and the kernel size is fixed to be 3x3 so that the shape of the image is maintained. Each convolutional layer is followed by a ReLU activation function to introduce non-linearity whilst maintaining learning speed. The first convolutional layer is fixed to have 8 filters, and any subsequent convolutional layer will have double the number of kernels as the previous layer. The number of filters is ascending because at the bottom layers, the model extracts low level features while at higher layers, the model extracts more high level features which uses these low level features. All models contain a max pooling layer and one hidden fully-connected layer with 128 units (as explained in the following section), and are trained for 10 epochs each.

We also attempted to use VGG blocks (conv-relu-conv-relu-maxpool), inspired by the state-of-the-art VGG model. The results are shown in Table 1.2 and 2.2.

Table 1.1. Convolutional Layers experiment (Normal vs Infected)

No. of Conv Layers	Test Loss	Test Accuracy	Test F1
1	0.5705	81.1%	86.4%
2	0.4463	82.9%	87.5%
3	**0.4538**	**83.7%**	**88.0%**

Table 1.2. VGG blocks experiment (Normal vs Infected)

No. of VGG blocks	Test Loss	Test Accuracy	Test F1

1	0.5269	80.8%	86.2%
2	0.6530	81.8%	86.9%
3	**0.4507**	**85.7%**	**89.4%**
4	0.5805	84.0%	88.3%

Table 2.1. Convolutional Layers experiment (COVID vs Non-COVID)

No. of Conv Layers	Test Loss	Test Accuracy	Test F1
1	0.3631	88.4%	84.4%
2	0.3908	88.9%	84.2%
3	**0.3398**	**90.0%**	**85.8%**

Table 2.2. VGG blocks experiment (COVID vs Non-COVID)

No. of VGG blocks	Test Loss	Test Accuracy	Test F1
1	**0.3181**	**91.6%**	**87.6%**
2	0.3854	85.8%	80.7%
3	0.3844	87.4%	82.0%
4	0.3915	85.3%	80.3%

From the results in Tables 1.1, 1.2, and 2.1, we can see that the model performance generally increases with more convolutional layers or blocks. As the network gets deeper, the channel depth increases. This allows for more feature maps, each acting as a separate individual pattern detector, to be stacked together to combine neighbouring parts and learn more complex features. In general, the deeper the network, the greater is its learning capacity.

However, in Table 2.1, we see a small dip in model performance from 3 VGG blocks to 2 VGG blocks. Also, in Table 2.2, we observe that the model performance decreases with more convolutional blocks, contrary to earlier results. We believe this is a result of performance degradation arising from difficulties in identity mapping, especially for convolutional models without skip connections like ours (He, Zhang, Ren, & Sun, 2016).

Our final choice of architecture is based on the highest test F1 score for each binary classifier. For Binary Classifier #1, we will use 3 VGG blocks, and for Binary Classifier #2, we will use 1 VGG block.

Hidden Fully-Connected layers

We made sure to have at least 1 or more hidden fully-connected (FC) layers. This is because if there is no no hidden FC layer, the network is only capable of modelling only a linear function, which is inadequate for most image classification tasks.

By the Universal Approximation Theorem, we believe a single FC layer is sufficient as a feature extractor in modelling functions that are just complex enough to perform our simple classification task well. We did not go for more layers to avoid overfitting as our dataset size and number of classes are not significantly large.

Additionally, since most of the parameters are contributed by the FC layers, having less layers also means less number of parameters to be updated during backpropagation and hence faster training time. This assumes that a reasonable number of parameters is set for each layer.

Discuss the choice of value for a mini-batch size for the training set.

The mini-batch size for the training set determines the number of training samples to be loaded during each iteration in the training process. With larger batch size, we can optimize the loss simultaneously over a larger set of images and potentially get more accurate gradients and updates. However, smaller batch size may introduce some noise to the model which may help improve generalisation. Thus, we need to find the appropriate value for batch size.

We conducted grid search over different combinations of parameters, including mini-batch size. Each combination is run for 3 epochs. The parameters and the values we searched are shown in Table 3 and Table 4 below. The most optimal combination of parameters which yielded the best test F1 score is also recorded. For Table 3 (Binary Classifier #1), the best test F1 score is 92.5% while for Table 4 (Binary Classifier #2), the best test F1 score is 66.3%.

For Binary Classifier #1 and Binary Classifier #2, a batch size of 64 and 16 respectively was determined to be the best value based on our grid search results.

Table 3. Grid Search for parameters (Normal vs Infected)

Parameters	Search values	Best value
Mini-batch size	[16,32,64]	64
Learning rate	[1e-4, 1e-3, 1e-2]	0.0001
Adam momentum term 1	[0.8,0.9]	0.9
Adam momentum term 2	[0.8,0.9]	0.9
Weight decay	[1e-5, 1e-4]	0.0001

Gamma	[0.95, 0.9]	0.9

Table 4. Grid Search for parameters (COVID vs Non-COVID)

Parameters	Search values	Best value
Mini-batch size	[16,32,64]	16
Learning rate	[1e-4, 1e-3, 1e-2]	0.001
Adam momentum term 1	[0.8,0.9]	0.8
Adam momentum term 2	[0.8,0.9]	0.8
Weight decay	[1e-5, 1e-4]	0.0001
Gamma	[0.95, 0.9]	0.95

Discuss the choice of a loss function and parameters, if any.

We used cross-entropy loss as our loss function, since this is a classification problem. The cross-entropy loss allows us to calculate the probability of the sample being in each class, which may be important in diagnosing COVID. For example, when the model predicts a non-COVID class with 51% probability, the patient still has a high probability of having COVID and therefore the appropriate measures should be taken.

Moreover, we explored using weighted loss functions to overcome class imbalance as seen from Figure 1. We give the minority classes heavier weights such that misclassification during training results in larger losses. The model will be more sensitive to the minority samples as the optimizer tries to minimise training loss. By doing so, the model will be less biased towards predicting the majority classes just because they are seen more often during training.

To assign each class weights, we took the inverse of the total number of samples from that class. For example, for the binary classifier which detects non-COVID and COVID, there are 2530 non-COVID training samples and 1345 COVID samples, thus the non-COVID class will get 1/2530 weight and the COVID class will get 1/1345 weight. As such, although the COVID class has a lower number of samples than the non-COVID class, it is assigned a larger weight.

Explain your choice of an optimizer and its parameters, if any (e.g. learning rate, momentum, etc).

We used the Adam optimizer, since it is an improvement over RMSProp (which can adapt its step-size to flat regions and steep regions), and uses two momentum terms: one for the gradient, and one for the element-wise squared gradient. The Adam optimizer also allows faster convergence during training, which is especially important to our project where we need to experiment many different combinations of parameters.

The choice of parameters (learning rate and the 2 momentum terms) is found through grid search, as seen in Table 3 and Table 4.

From the grid search, we determined learning rate to be 0.0001 and the momentum terms to be 0.9 each for Binary Classifier #1. On the other hand, Binary Classifier #2 has a learning rate of 0.001 and momentum terms of 0.8 each.

Explain your choice of initialization for your model parameters.

We used the Kaiming initialization for the convolutional layer parameters because it shows better stability than random initialization. This is because Kaiming works especially well for networks that implement ReLu activation functions, such as ours, as it allows for the mean to increment slowly with variance close to 1 in the feedforward phase, thereby avoiding vanishing or exploding gradient problems during backpropagation phase.

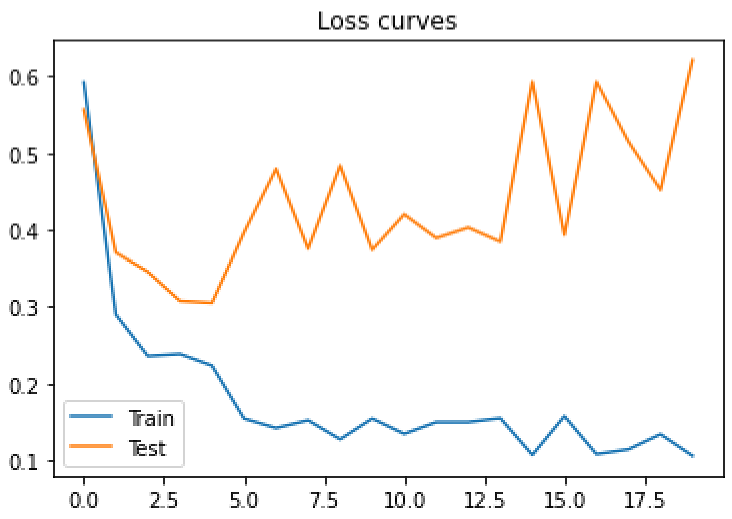
Additionally, Kaiming initialization is also able to break network symmetry, just like random initialization. This prevents the case where the weights are updated identically, leading to neurons in the model not being able to produce different outputs.

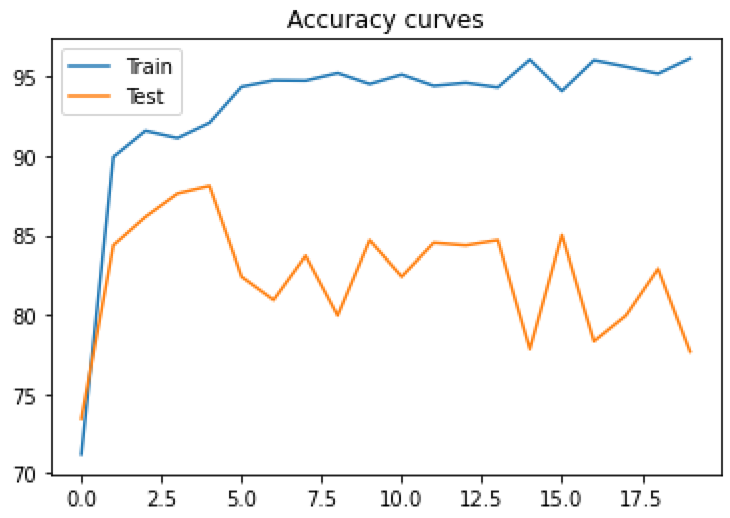
Provide and explain learning curves showing the evolution of your loss function and other performance metrics over successive training epochs.

The plots of the loss, accuracy and F1 score curves of Binary Classifier #1 and Binary Classifier #2 are shown in Figure 4 and Figure 5 respectively.

For Binary Classifier #1, train loss generally decreases while train accuracy and F1 score increases. However, as seen from Figure 4, the test loss is higher than train loss, and the test accuracy and F1 scores are lower than that of the train set. Moreover, we can see that for the test curves, the test performance stops improving after around the 4th epoch. This could be a case of overfitting, where the model is too fitted on the train set and does not learn the generalised features. If given more time, it is possible to explore the use of dropout layers during training to reduce the overfitting and improve generalisation of the model. For now, we have saved the model based on the best test F1 score, thus even after 20 epochs, only the model from the 4th epoch (with the best test performance) is saved. Thus, the resulting model would not be as overfitted.

For Binary Classifier #2, the test curves have generally the same shapes as the train curves. The loss curves are generally decreasing while the accuracy and F1 score curves are generally increasing. This means that given more epochs, this classifier’s performance would continue to improve. From Figure 5, we also see that test loss is lower than train loss, and test accuracy and F1 is higher than that of the train set. This shows that the model is not overfitted on the train set, since it predicts better on the test set than the train set.





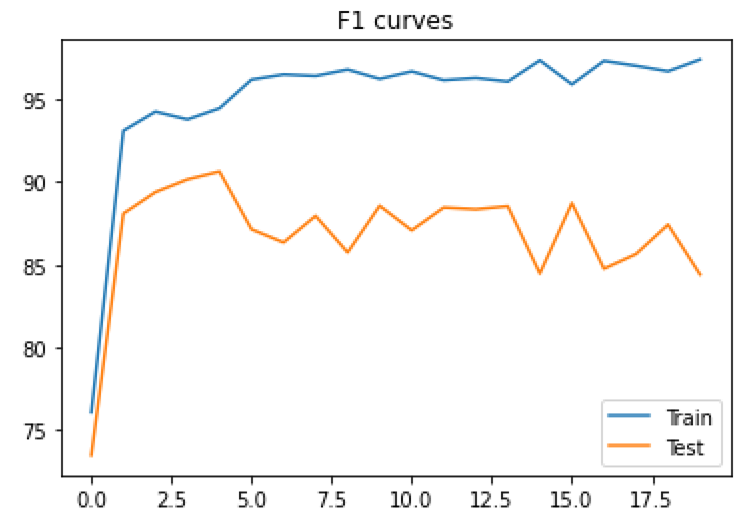
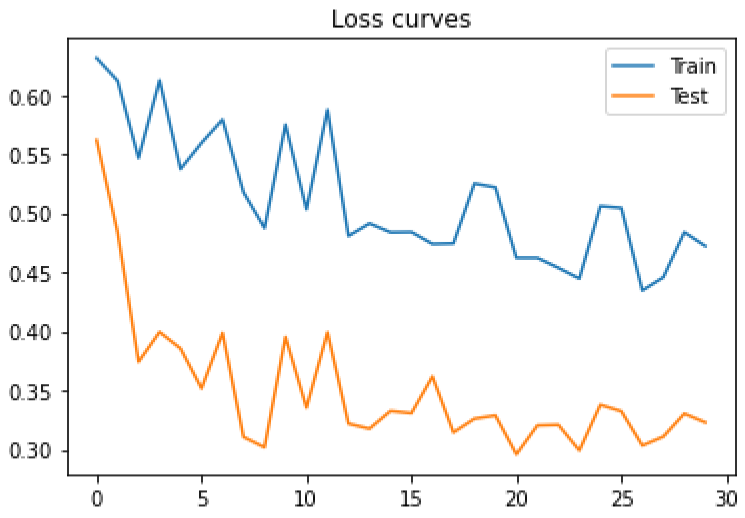
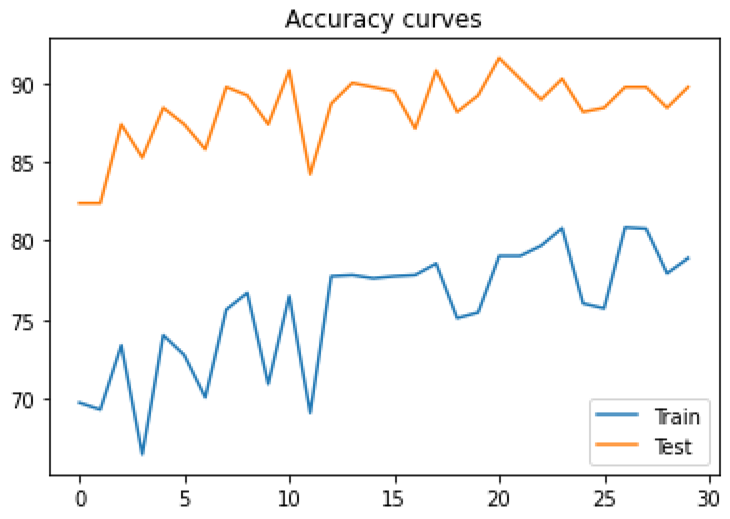


Figure 4. Learning curves over successive epochs for Binary Classifier #1





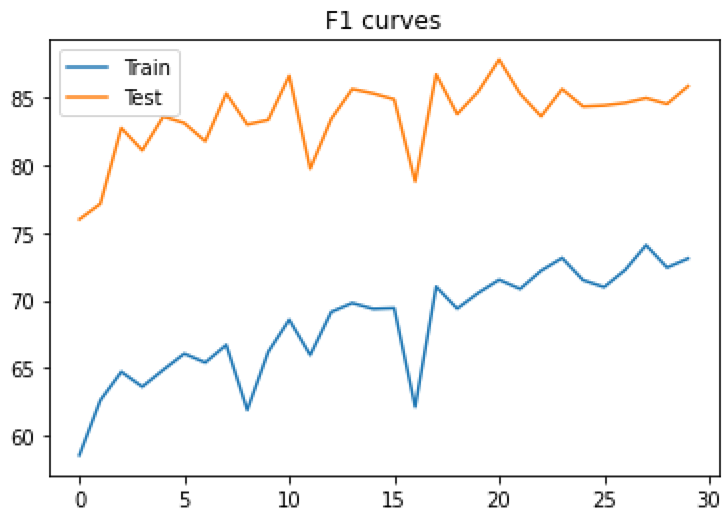


Figure 5. Learning curves over successive epochs for Binary Classifier #2

Create a loader function for instructors to reproduce and verify model performance

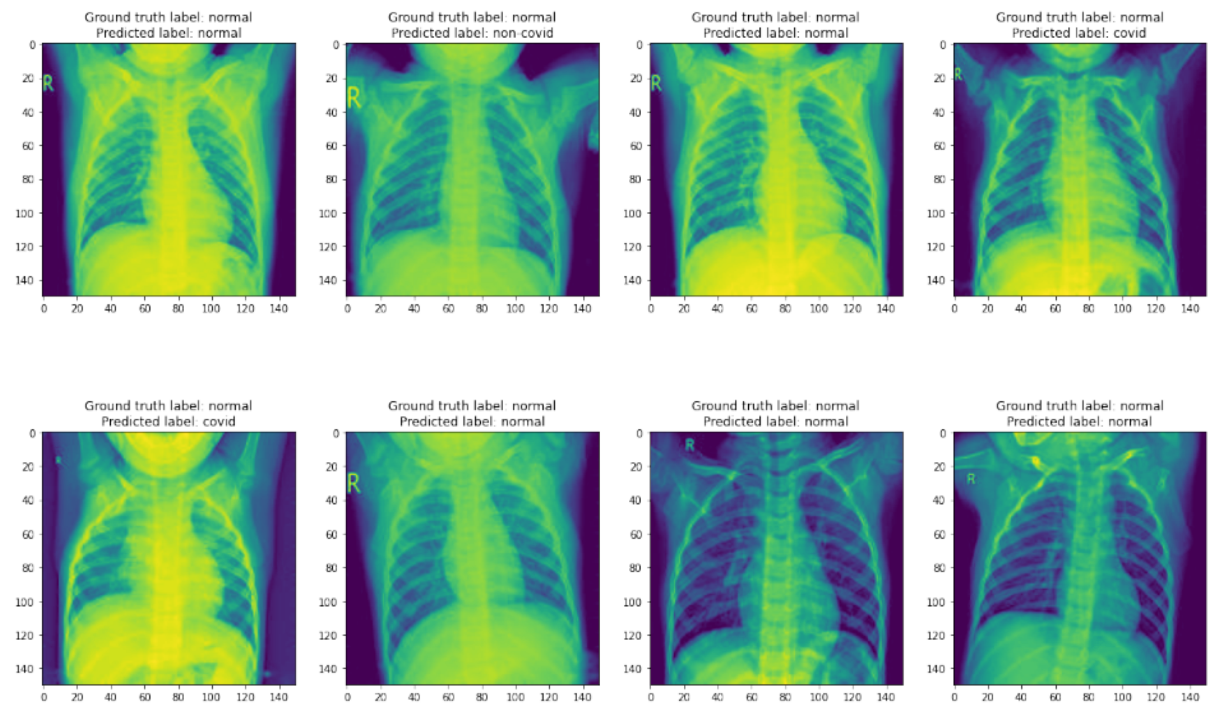
The loader function is written in model.py. An argument parser called --checkpoint has been added to both train.py and evaluation.py, which can be used to load the model path for training and evaluation respectively.

Display the performance of your trained model on the 24 images in the validation set. Include possible overall metrics too.

The validation accuracy is 70.8%. The validation F1 score is 72.7%. The performance is expected to be slightly lower than the individual scores of binary classifier #1 and binary classifier #2, because an infected sample would have to be correctly predicted in two passes instead of just one.

Loh De Rong (1003557), Koh Hui Wen (1003593)

Validation set pictures with predicted and ground truth labels
Average accuracy 17/24 = 70.8%
F1 score for COVID class = 72.7%


Ground truth label: normal
Predicted label: normal
Ground truth label: normal
Predicted label: non-covid
Ground truth label: normal
Predicted label: normal
Ground truth label: normal
Predicted label: covid
Ground truth label: normal
Predicted label: covid
Ground truth label: normal
Predicted label: normal
Ground truth label: normal
Predicted label: normal
Ground truth label: normal
Predicted label: normal

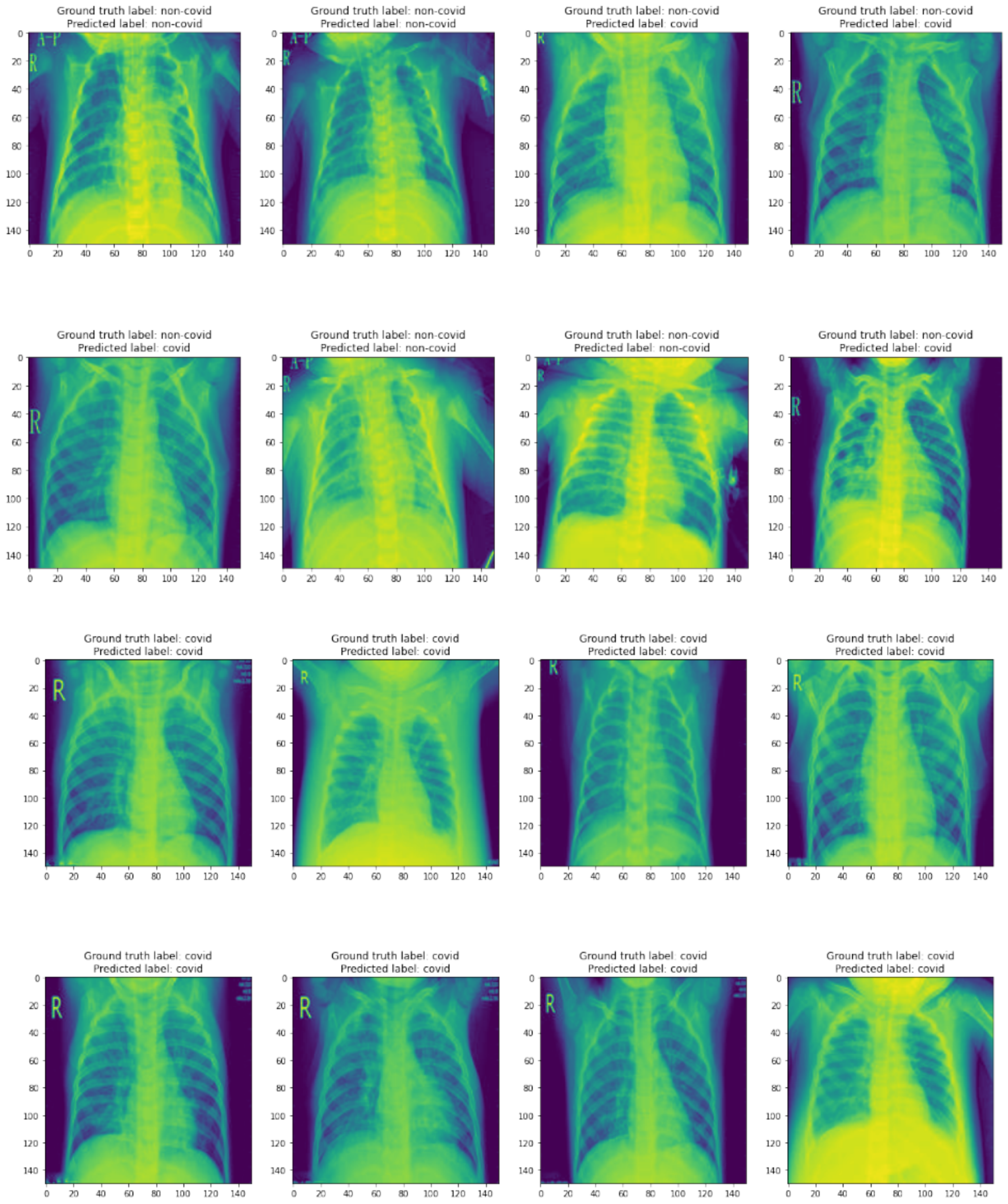

Ground truth label: non-covid
Predicted label: non-covid
Ground truth label: non-covid
Predicted label: non-covid
Ground truth label: non-covid
Predicted label: covid
Ground truth label: non-covid
Predicted label: covid
Ground truth label: non-covid
Predicted label: covid
Ground truth label: non-covid
Predicted label: non-covid
Ground truth label: non-covid
Predicted label: non-covid
Ground truth label: non-covid
Predicted label: covid
Ground truth label: covid
Predicted label: covid
Ground truth label: covid
Predicted label: covid
Ground truth label: covid
Predicted label: covid
Ground truth label: covid
Predicted label: covid
Ground truth label: covid
Predicted label: covid
Ground truth label: covid
Predicted label: covid
Ground truth label: covid
Predicted label: covid
Ground truth label: covid
Predicted label: covid

Figure 6. Performance of trained model on validation set.

You might find it more difficult to differentiate between non-covid and covid x-rays, rather than between normal x-rays and infected (both covid and non-covid) people x-rays. Was that something to be expected? Discuss.

This is expected. The difference in normal x-rays and infected x-rays is the presence of pneumonia infection, which presents itself in the form of white spots. Such clinical features should be easily picked up by the convolutional model to be used for classification. However, there does not seem to be any obvious detectable difference between non-COVID and COVID x-rays, hence the model might struggle in extracting features that distinguish between non-COVID and COVID classes.

Would it be better to have a model with high overall accuracy or low true negatives/false positives rates on certain classes? Discuss.

It is preferred to achieve low true negative rates for the more “dangerous” classes (i.e. infected class and COVID class), and low false positive rates for the less “dangerous” classes (i.e. normal class and non-COVID class).

For example, rather than a model with high accuracy but classifies COVID samples to be non-COVID (due to the train and test set having more non-COVID samples), we want the model to predict COVID samples correctly despite potentially having non-COVID samples be misclassified as COVID. This is because it will be more severe to miss the COVID cases as the virus is lethal and these patients may unknowingly spread the disease to surrounding people instead of being quarantined, leading to increased disease transmission in the community. Even if non-COVID samples are mislabelled as COVID, it would at most cause a false alarm, which is less harm than if a COVID case went undetected.

Similarly, it is better for healthy, normal cases to be mislabelled as infected, rather than having the infected cases go undetected. The infected cases are more life-threatening and thus the consequences would be much more severe to miss such cases. The cost of initial unnecessary worry due to thinking that the patient is infected (even though they are not) is nothing compared to suffering from pneumonia or COVID due to the treatment not being given in time.

Bonuses

Explain in your report the reasons that motivated you to perform data augmentation and show proof of how it benefited your model.

The reason that motivated us to perform data augmentation is to generate more similar images for every new batch, and therefore increase the train dataset size to reduce overfitting. This is because when more data is added, the model is unable to overfit all the samples, and is forced to generalize.

For Binary Classifier #1, it can be observed that without data augmentation, the test loss increased much more quickly relative to the train loss (refer to Figure 7). Moreover, we see that using data augmentation generally decreases the test loss such that the difference between the train and test loss is not as large. Even though data augmentation did not prevent overfitting, it certainly reduced overfitting to a large extent.

For Binary Classifier #2, the benefits of data augmentation are more obvious. Without data augmentation, the test loss started to increase at the 4th epoch while the train loss continued to decrease, suggesting model overfitting (refer to Figure 8). However, with data augmentation, the train and test loss are decreasing in sync. In fact, the test loss is consistently lower than the train loss. This is evidence that data augmentation has helped to prevent overfitting.

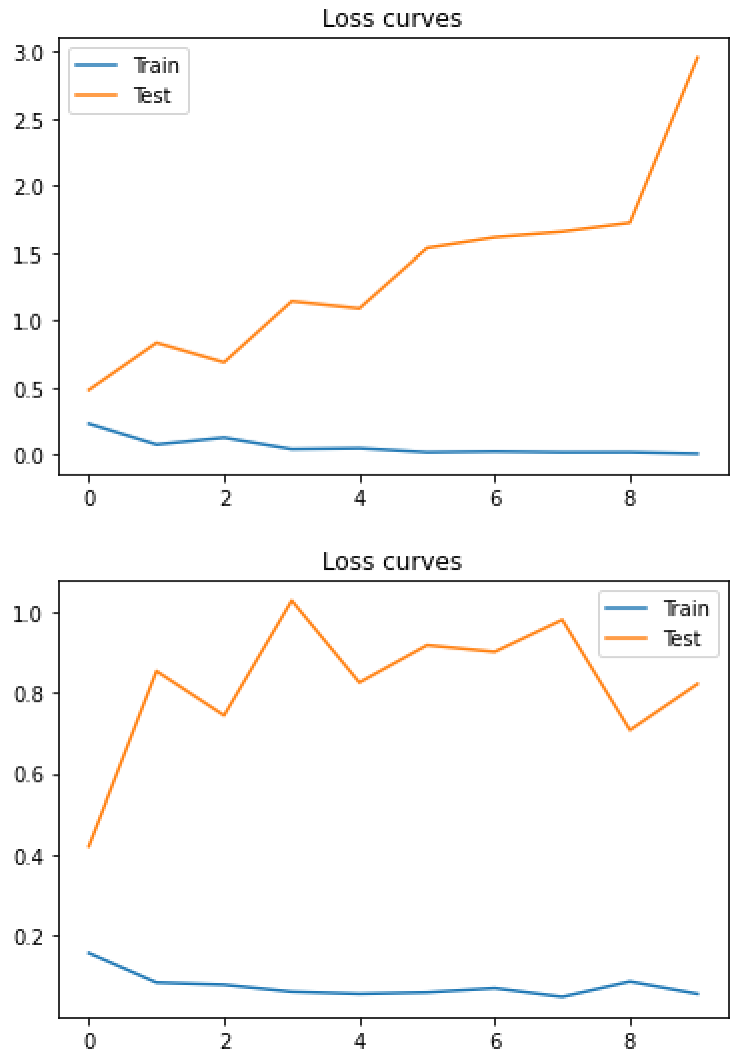
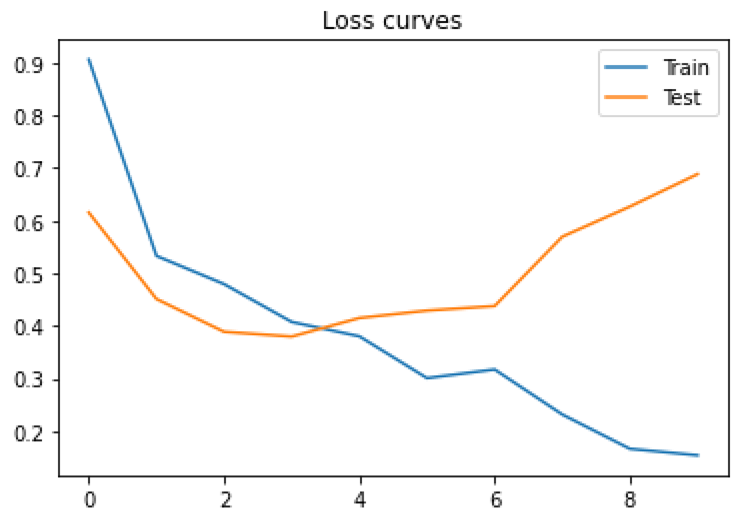


Figure 7. No data augmentation (top) vs data augmentation (bottom) for Binary Classifier #1



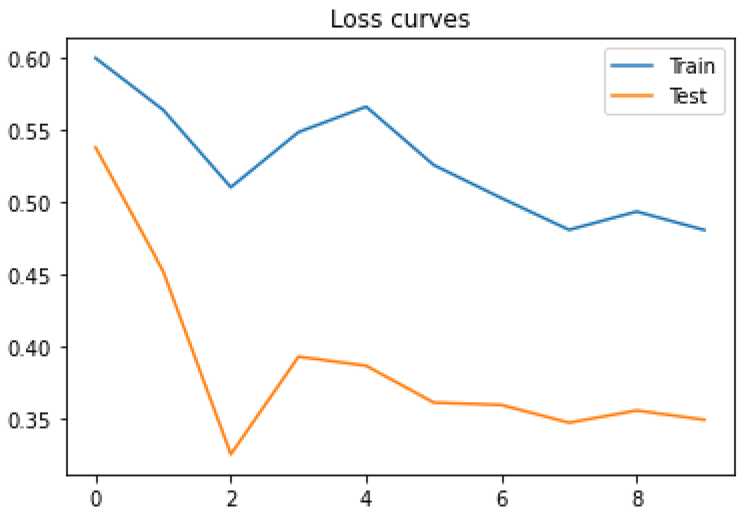


Figure 8. No data augmentation (top) vs data augmentation (bottom) for Binary Classifier #2

Implement and discuss a learning rate scheduler, and discuss its appropriate choice of parameters and benefits for your model.

We used the stepwise learning rate reduction scheduler. For every step, the learning rate scheduler decreases the learning rate using the gamma parameter. As the training epochs increase, the model becomes closer and closer to the local minimum. There is a possibility that it will take too large a step and overshoot the local minimum. Thus, we need to decrease the learning rate when it is close to the local minimum. By using the learning rate scheduler, we can decrease the learning rate as the training epochs increases, effectively allowing us to reach the local minimum.

The choice of parameters (gamma) is found through grid search, as seen in Table 3 and Table 4. From the grid search, the value of gamma was found to be 0.9 for Binary Classifier #1 and 0.95 for Binary Classifier #2.

Loh De Rong (1003557), Koh Hui Wen (1003593)

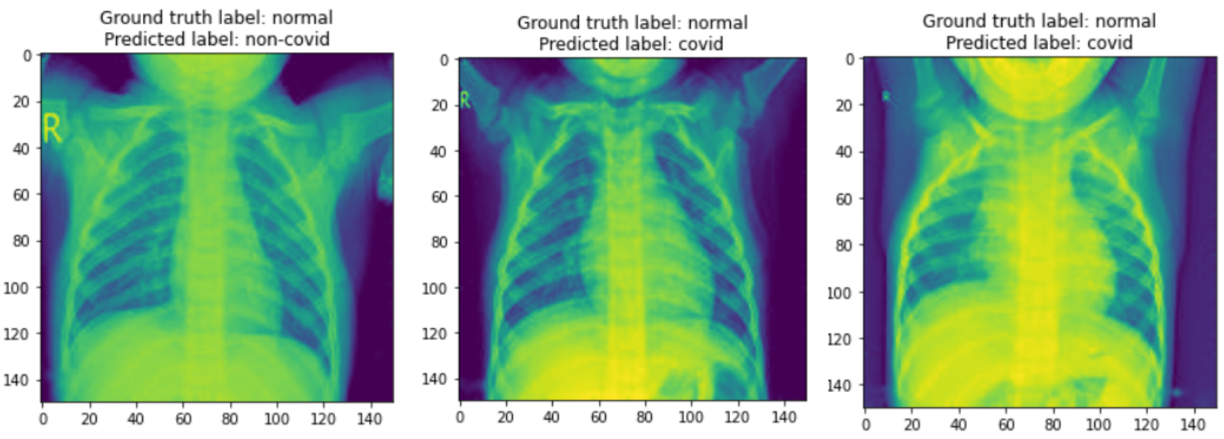
Implement a regularization on your loss function and discuss its appropriate choice of parameters and benefits for your model.

We implemented weight decay as a parameter to the Adam optimizer. The weight decay specifies a regularization term which is added to the neural network's loss to compute backpropagation gradients during training. This regularization penalizes large weights and prevents the model from overfitting to any particular feature of the training samples, thus improving the generalization of the model.

The choice of parameters (weight decay) is found through grid search, as seen in Table 3 and Table 4. From the grid search, the value of weight decay was found to be 0.0001 for both binary classifiers.

Briefly look up online how doctors diagnose infections based on x-rays. Does our AI seem to be able to reproduce this behavior correctly? Show typical samples of the dataset on which your AI failed and discuss what might have been the reasons.

When doctors diagnose infections based on x-rays, doctors will look for white spots in the lungs. These white spots are called infiltrates and they identify an infection (Radiological Society of North America & American College of Radiology, 2019). Our AI does seem to reproduce this behavior correctly, since all our non-COVID and COVID samples are correctly predicted as infected by Binary Classifier #1 (refer to Figure 6).


Ground truth label: normal
Predicted label: non-covid
Ground truth label: normal
Predicted label: covid
Ground truth label: normal
Predicted label: covid

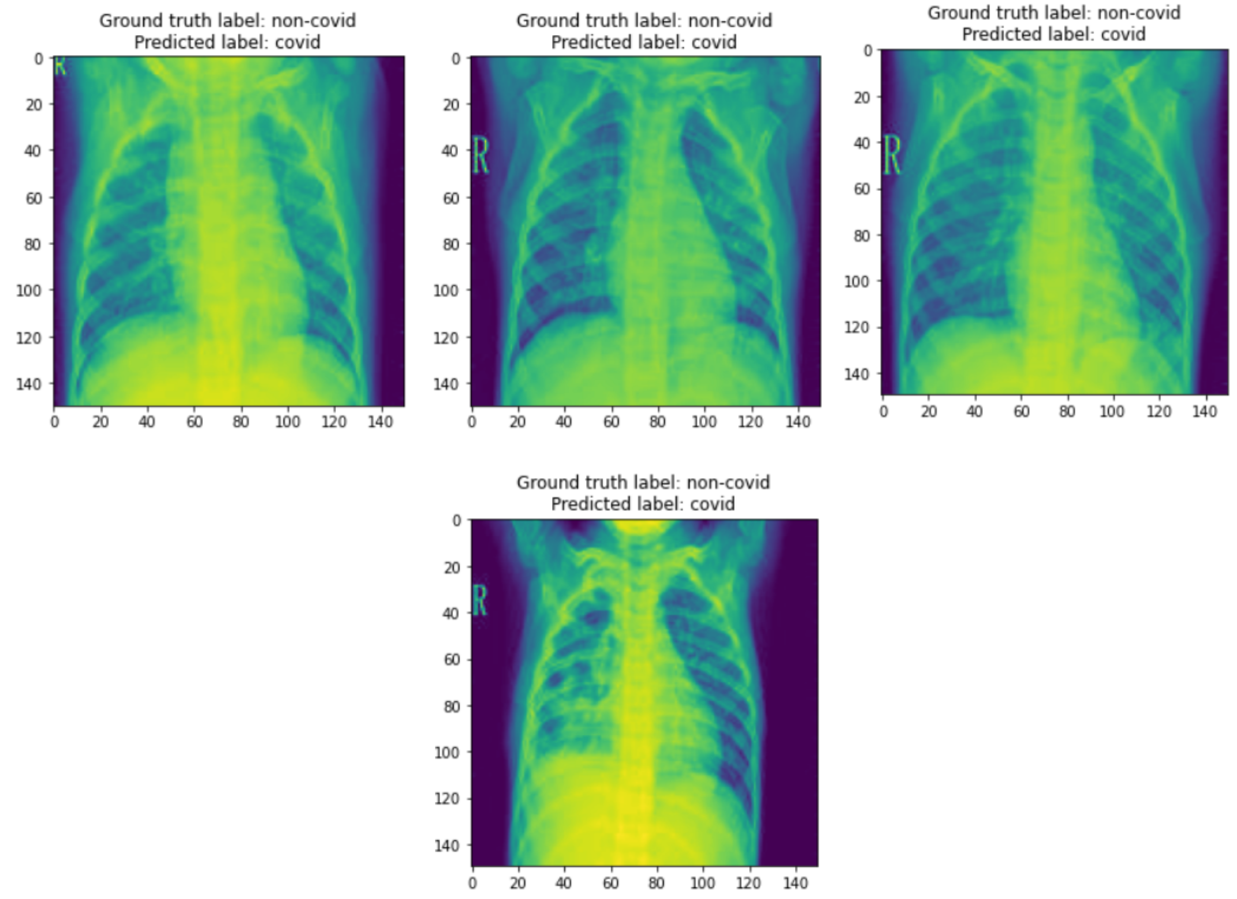

Ground truth label: non-covid
Predicted label: covid
Ground truth label: non-covid
Predicted label: covid
Ground truth label: non-covid
Predicted label: covid
Ground truth label: non-covid
Predicted label: covid

Figure 9. Typical samples of the dataset on which our AI failed.

These 7 images in Figure 9 are taken from the validation set, which have been misclassified by our model. Of all 3 normal samples, they are all predicted as infected; in particular, 2 of them are predicted as COVID. Of the remaining 4 non-COVID samples, they are all predicted as COVID.

It is evident that the model tends to predict a class that is more “dangerous” due to the way we chose our best model, which is by the highest F1 score. F1 metric places more emphasis on predicting the positive class correctly, so it is also expected that more true positives can be captured at the expense of having more false positives.

References

1. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. doi:10.1109/cvpr.2016.90
2. Radiological Society of North America, & American College of Radiology. (2019). Pneumonia. Retrieved March 19, 2021, from https://www.radiologyinfo.org/en/info.cfm?pg=pneumonia