50.039 Small Project Report

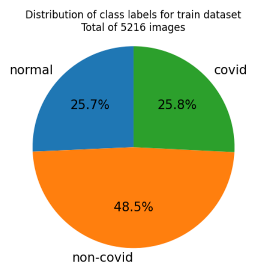
Notation

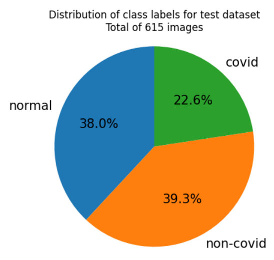
Before we begin, we would like to define the class labels that we are using within this project. There are 3 distinct classes within this context:

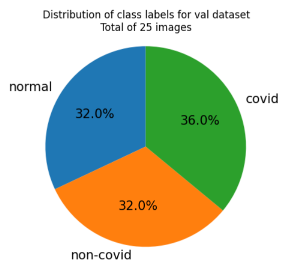
- normal: patients without pneumonia at all
- non-Covid-19: patients with pneumonia, but do not have Covid-19
- Covid-19: patients with pneumonia and Covid-19

Dataset

We started by looking at the datasets given. We visualised the 3 datasets into separate pie charts to show the distribution of class labels within each dataset.







We can immediately see the differences between the distributions. Most importantly is that there is a large class imbalance in the training dataset with the non-covid label having almost twice the number of samples as the other two labels. This would pose a challenge to a naive application of a deep learning model as the model can achieve high performance by simply biasing towards the majority class which is non-covid in this case. This would be the first challenge that we have to tackle.

Another challenge that has been posed to us is the relatively small dataset that we have on hand with the restriction against transfer learning. Most established deep learning models have a large number of parameters (e.g. VGG16 has 138 million parameters (Simonyan, K., & Zisserman, A. (2014)), which tends to result in severe overfitting especially in such a small dataset.

The main challenges that we have to tackle are these two:

- Adjusting for class imbalance
- Generalisation to unseen data by reducing overfitting

Data Pre-Processing and Augmentation

Class Imbalance

There are two popular options in tackling class imbalance: Data sampling, or Class weighing. We can adjust the sampling rate of a class label to ensure an equal balance of class labels within the training dataset. On the other hand, we can also assign weights to the penalties for misclassification of class labels so that we can penalize more heavily the misclassification of the minority classes to make up for the lack of samples in the dataset.

The resultant effect is the same for either method: each class label has roughly equivalent weight during training. We have implemented both methods:

- WeightedRandomSampler for Data Sampling: helps the training DataLoader to sample equal representation of each class label
- Passing weights to our loss functions for Class weighing: helps to place additional weights to the minority classes

Since both methods have the same effect, we have opted to just use the WeightedRandomSampler.

Generalisation

There are multiple ways in which we can attempt to generalize our model which we will cover later on. In this section, we are focusing on data augmentation. Typically data augmentation during training helps the model to generalize to unseen data better. Taking a look at multiple samples and after some experimentation, we have decided on using two data augmentations: rotation and affine transformation.

We have looked at several other popular options, but they do not seem appropriate or effective. For example, horizontal or vertical flipping of the image is definitely not applicable in our model's context because of the nature of the problem that we are attempting to solve. Xrays are generally asymmetrical and in fact, we can see a R on the top right corner to denote which is the right side. This suggests that horizontal flipping is definitely not appropriate.

Model Architecture

Differences between the two architectures: Cascading Architecture and 3-class Classifier

Dataset
A multinomial classifier is a good choice if the dataset is are points that belong exclusively to one class. For example, belonging to either Summer, Winter, Autumn or Spring. However, when the data can belong to more than one class, multiple binary classifiers would be more useful. Hence for our given data set and class labels, we have decided to go with the cascading architecture of 2 binary classifiers. This is because we recognise that the classes of infected, non-Covid-19 and infected, Covid-19, would share similar features of the lungs being infected. Additionally, it is likely that the detection of infection would be easier than the detection of Covid-19 in the lungs.

Learning Process
There is a simpler learning process involved with the 2 binary classifiers as the classifiers can focus on classifying just between 2 classes (either infected/not or Covid-19/not). It might also allow the model to understand better the features associated with that class. We can also model the learning process like a pseudo decision tree. This allows us to incorporate existing knowledge into the models (ie someone who has Covid-19 would definitely have a lung infection, at least in the point of view of this problem statement and dataset).

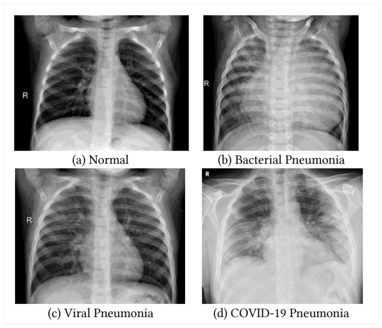
Balanced dataset
However, with the binary classifier, it is then important to have a balanced dataset for the classes. As shown in the dataset distribution graph above, our dataset is not very balanced. We deal with it using the WeightedRandomSampler as mentioned above.

Computation Time
The binary classifiers will be faster to train and also allows for parallelizability.

Ease of use
The multiclass classifier would however be easier to set up, especially if there is a large amount of classes. For example, if later on, we are tasked to identify more forms of pneumonia with our chained binary classifiers, we would need to train more models and chain them accordingly, which is less developmental intensive as compared to a single multiclass classifier.

Implications of using Cascading Architecture
While it is able to focus more on the characteristics per class, it relies on the assumption that the infection of lungs for a person with Covid-19 is the same as the infection of lungs without Covid-19. As the Covid-19 is still a novel virus, this is not something that we can claim for sure. The figure below may suggest that the infection caused by different diseases may spread

differently. This is something we might not be able to pick up as well using the Cascading Architecture.



(Katsamenis et al., 2020, p. 1)

Architecture Choices

Training

For our training, we decided to train the two models separately. The model for prediction infection will be trained on all training samples, and only compared to the target_label_infected for loss. For the model for predicting Covid-19, we only train on samples that are infected from the training set, and we train it to predict whether there is Covid-19 amongst the infected samples.

PreliminaryModel

- 2 Convolutional Layers
  While designing the model, we emphasised on the rule that the number of parameters to be trained should correspond accordingly to the number of training samples available. As we only have under 6000 training samples, which is a small amount (ImageNet for example has around 14 million) and in addition to the fact that we are not allowed to use transfer learning, we decided to keep the parameters to minimum (experimenting with a maximum of three convolutional layers).
- ReLU Layer
  After each convolutional layer, we added a non-linear activation function to help the model learn more complex patterns.
- Max Pooling layer
  In order to keep the number of parameters low, as well as to inject some noise to avoid overfitting, we added a max pooling layer after each convolutional layer

OneModel

- Only 1 convolutional layer
  This was an edit after implementing the Preliminary Model for both classifying infection and classifying Covid-19. We found that while the test losses for classifying Covid-19 were decreasing relatively smoothly, the test losses for classifying infection had very high variance, jumping to and fro from high value to low values. A likely possible reason

for this was overfitting, especially because we think that the prediction of infection is quite easy, so there are not a lot of parameters needed to learn it.

This is because in radiology, “the definition of pneumonia is the presence of an abnormal opacity on chest X-ray”(Mieghem, 2005, p 66). The X-ray images are largely similar, so it might be very easy for the machine to tell the difference between an infected lung and a non-infected one.

So we decided to decrease the convolutional layer to just one layer. Results will be discussed in the Results section.

ThreeModel

- 3 conv layer
  Lastly, for experimentation, we also decided to increase the number of layers to see the effect on the learning. To deal with the increase in layers, and thus the increase in parameters, we implemented a bigger max pooling layer (?)

HighPass/LowPass Model

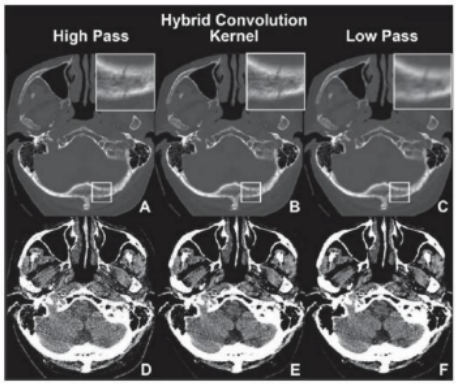
For this it takes on the same structure as PreliminaryModel(), ie 2 convolutional layers, but we added an additional layer at the start (that is not to be trained) to filter the high frequencies or low frequencies depending on the kernel set.

Low pass filtering retains low spatial frequency components and removes the high spatial frequency components (such as noise, texture). High pass filtering does the opposite.

High/Low pass filtering is often in medical imagings and proves useful even not in machine learning areas.

High pass and Low pass

Fig. 1—From left to right, axial 2.5-mm section from unenhanced head CT scan generated with bone (high pass), hybrid (hybrid convolution kernel), and standard (low pass) convolution kernels and displayed with preset bone (top row: width, 2,500 HU; level, 480 HU) and brain (bottom row: width, 80 HU; level, 40 HU) window settings. For depiction of bone, raters judged **A** and **B** equivalent but superior to **C**. Note trabeculae and nondisplaced fracture of left occipital bone (*square box with magnified inset*) are less distinct in **C**. For depiction of soft tissue, raters judged **E** and **F** equivalent but superior to **D**. Note increased noise within brain and extracranial soft tissue (**D** vs **E** and **F**).



(Weiss et al., 2011, p. 404)

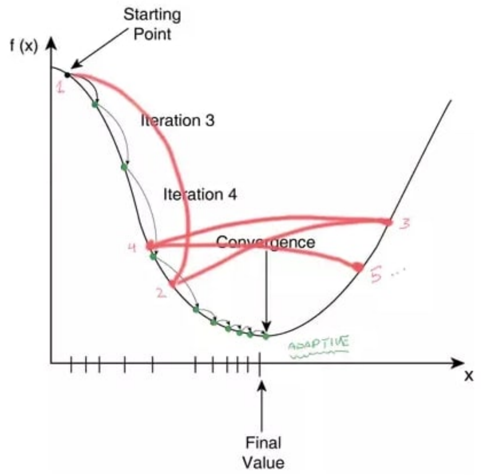
As seen from the figure above, the high pass filtering allowed for bone fractures to be more easily discerned, while low pass filter was decided by experts to be a better depiction of a soft tissue. Since our x-rays are of lungs which are soft tissue, we are expecting the low pass to do well here, but we implemented both passes for experimentation.

Minibatch size

The minibatch size is important for a variety of reasons. Some common minibatch sizes are 32, 64, 128 etc and can even go up to 512. For our small dataset, we experimented with smaller minibatches of 8,16,32,64, 128. For a collection of factors of mostly limited resources and for speed, we went with the standard size of 64.

The size of the minibatch will affect the:

- Training time until converge
  This requires a sweet spot, as a too big or too small minibatch size will result in long training time.
- Training Efficiency
  A larger batch size would imply faster training times, as we are processing less batches in total by processing more data points at one time.
- Quality in Model, measured by ability to generalize
  We also observe that for larger batch sizes, there is a usually a decrease in ability to generalize on the test set (Keskar, 2016). A research paper suggests that this due to the fact that large batches tend to converge to sharp minimizers and this causes poorer generalization.
- Relation to other hyperparameters
  The extent the batch size affects also depends on the other parameters, especially learning rate. With the combination of these two, it will adjust the speed of learning, and we should adjust both to find the sweet spot that will converge fast.



This diagram shows the effects of both small jumps and big jumps (affected by batch size and learning rate) in missing the convergence point, resulting in longer time to convergence.

Loss function

We chose to use the Cross Entropy Loss Function, one of the most commonly used loss functions in classification problems, combining the NLLLoss with the soft max. A good reason to choose a probabilistic loss function is so that we can also output the probability of the prediction. The probability would be useful in the area of prediction in medical fields. This is because healthcare professionals might want to know the probability in the case of limited healthcare resources to allocate corresponding resources to a patient, or to suggest different treatments depending on how probable the person has infection/Covid-19.

Optimizer

- Weight_decay as regularisation
  We also decided to use L2 regularisation, seeing as the test losses for predicting infection was fluctuating a lot after each epoch. This regularisation is implemented using pytorch's implementation of weight_decay on the Adam optimizer, as set out originally by (Loshchilov and Hutter, 2017).
  After some experimentation, we have gone with the weight_decay value of 1e-2.

- Learning rates
  We explored using learning rates of 0.01, 0.001 and 0.0001. With 0.01 rate, we initially hit what seems like an early local minimal point with the Covid-19 model at around 0.694 training loss, which was surprising as we did not expect it to be an issue for a relatively large learning rate since the other smaller rates did not face such an issue. We later discover that it could be the result of oscillating between two gradients and not actually a minimum point. We note that if given more epochs, it is possible that Adam's adaptive learning rate is able to break out of this oscillation and have the training loss decrease further.

  We ended up using the learning rate of 0.001 since it did not face the issue above and it converges faster than 0.0001. The specific experiment can be found in the annex.

Evaluation Metrics: Is Accuracy Enough?

Evaluation metrics are the criterion that we use to understand how respective models perform in different contexts. We consider precision and recall as potential metrics alongside the typical accuracy metric due to the nature of the problem at hand. The highly contagious nature of Covid-19 puts into question whether just having a good accuracy score is sufficient.

It is obvious that the correct identification of the Covid-19 label would be more helpful in the context of this global pandemic outbreak than perhaps the other two labels so that healthcare officials can take the necessary precautions to prevent further infection. Hence we can choose to prioritise this specific label. However, there are multiple facets to this task, which is why we are considering the precision and recall metrics.

Precision measures how accurate our prediction of the positive class is whereas recall measures how well the positive class is predicted. High precision would imply that for any positive prediction our model makes, it has a high probability that the patient is indeed infected. Having high recall would imply that for any patient who is infected, our model has a high probability of identifying him as infected.

Recall is important since Covid-19 patients are highly contagious and therefore, it is vital that we identify them if they are indeed infected so that proper precautions can be taken. However, similarly, we do not want to achieve high recall at the expense of precision (since recall can be attained by simply predicting all samples as positive) since we do not want to necessarily alarm normal and well patients if they do not have any conditions.

Hence we arrive at using the F-score to combine the two metrics and balance them. However, as we deem that recall is more important in this current context to minimize false negatives, we would want to choose a higher beta score to put more emphasis on recall. In conclusion, we end up using the F2-score as our main evaluation metrics (alongside accuracy for a quick comparison). We specifically use the F2-score of the Covid-19 class label as the main criterion

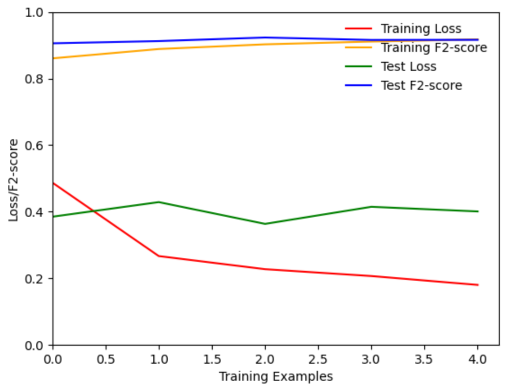
Discussions

Standard Model

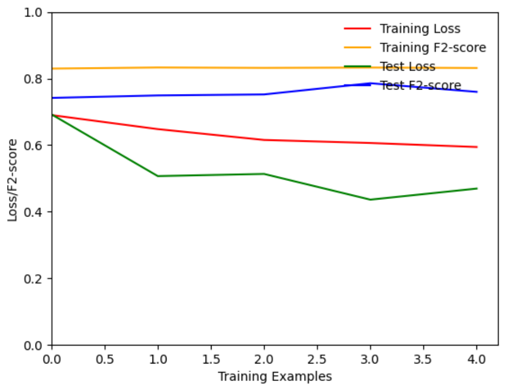
We chose as our standard model to implement both the model for predicting infected and Covid-19 labels as PreliminaryModel() (as described in above).

Results

Graph of Losses and F2-Score (of the positive label) over the Learning



Infection Classifier (Preliminary Model)



Covid-19 Classifier (Preliminary Model)

We tested the combined models with the test datasets to obtain the following results:
Overall accuracy: 0.776
F2-scores for:

- normal: 0.767,
- non-covid: 0.871,
- covid: 0.616

Evaluation
We observe that the test loss fluctuates more than the other curves. This is likely due to overfitting on the model for the training set, resulting in loss in the test set. However, despite this fluctuation of magnitude of loss, it seems like largely, the prediction of the labels do not change, resulting in a pretty stable F2-Score.

We also observe that the Infection Classifier has a much higher F2-score, which is to be expected, as we explained how the model learns with respect to the infection dataset and the Covid-19 dataset above.

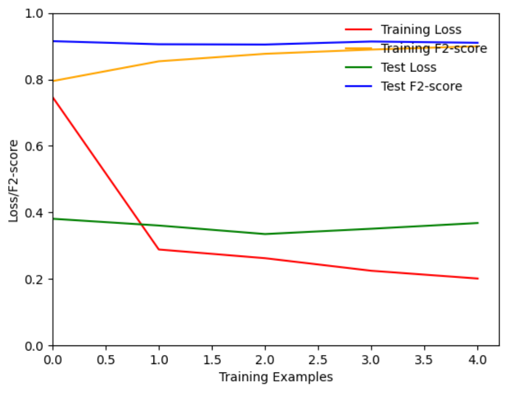
Variations

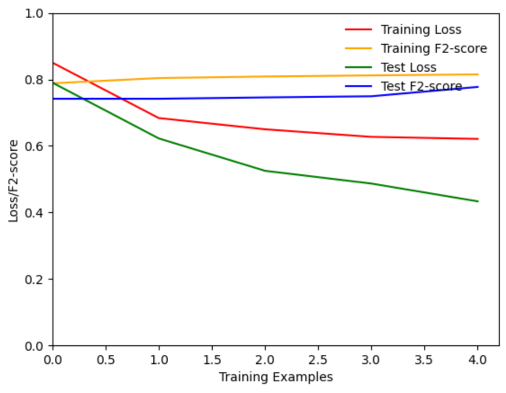
We have 4 variations, which are OneModel(), ThreeModel(), HighPass(), LowPass(). In the following section, we will present the learning curves (in terms of loss and F2-score) of each model as an infection classifier and Covid-19 classifier. We will also discuss the evaluation in terms of how the models measure up to the standard model, or with other combinations of other models, as well as their effects on the F2 scores of infection, Covid-19, and overall scores.

For the sake of convenience, we shall denote the models we use as [InfectionClassifier, Covid-19Classifier]. For example, if we were to say [OneModel,PreliminaryModel], it would mean we used the OneModel for the Infection Classifier and the PreliminaryModel for Covid-19 Classifier.

1. Replacing with OneModel()

Results





Infection Classifier (One Model) Covid-19 Classifier (One Model)

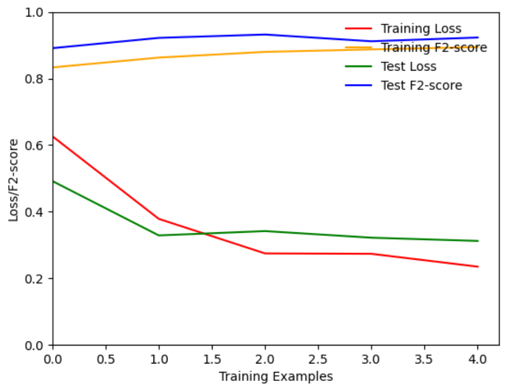
*more results can be found in Appendix

Evaluation

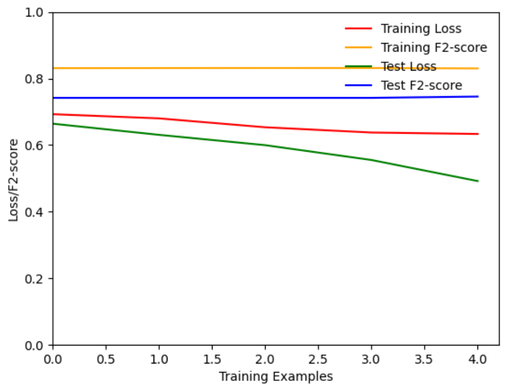
We find that this generally increases in the F2 Score for Covid-19 Classifications when it is used as a Covid-19 classifier. However, the increase is not very drastic, as we can see from the comparison of [PreliminaryModel,PreliminaryModel] and [Preliminary,OneModel], the Covid-19 F2 score increased from 0.616 to 0.645. For [OneModel, PreliminaryModel], we can see that there is also an increase for Normal F2 score from 0.767 to 0.837. However, when we look at [OneModel, OneModel], while the Normal F2 score still remains relatively high at 0.837, the Covid-19 F2 score drops to 0.569.

2. Replacing with ThreeModel()

Results



Infection Classifier (Three Model)



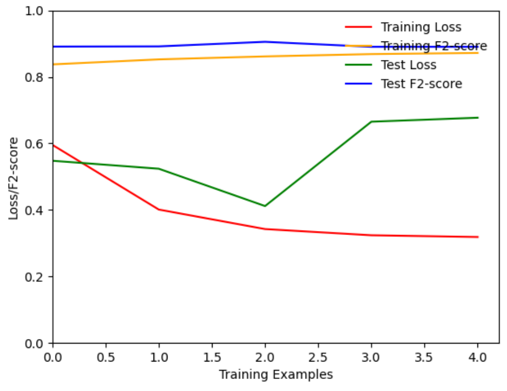
Covid-19 Classifier (Three Model)

Evaluation

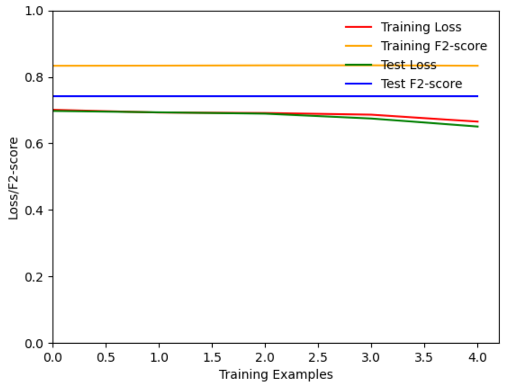
The three convolutional layer model (ThreeModel) worked much better for predicting infection than predicting Covid-19. We can see for the [ThreeModel,PreliminaryModel] there is a pretty good increase of F2 Normal score to 0.881 from 0.767 from the Standard (ie [Preliminary,Preliminary]) , while for [PreliminaryModel,ThreeModel] there is a drastic decrease of F2 Covid-19 Score from 0.616 to 0.218.

3. Using HighPassModel()

Results



Infection Classifier (HighPass Model)



Covid-19 Classifier (HighPass Model)

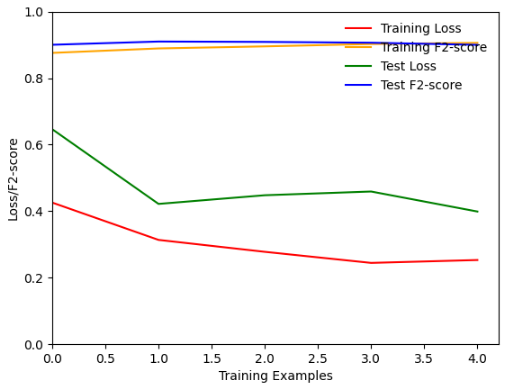
Evaluation

The HighPass model performed badly for both F2 Normal Scores and F2 Covid-19 Scores. This is expected as we mentioned in the introduction of this model that the High Pass filter is expected to depict medical images better for bones.

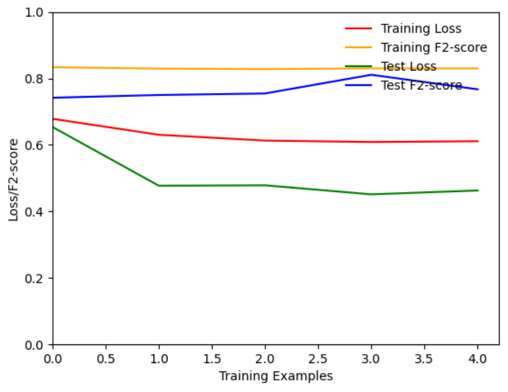
For all HighPassModels() used as Covid-19 classifiers, we see that there is a F2 Covid-19 Score of 0.0, while if it was used as an Infection Classifier, the F2 Normal score drops to 0.474.

4. Using LowPassModel()

<u>Results</u>



Infection Classifier (LowPass Model)



Covid-19 Classifier (LowPass Model)

<u>Evaluation</u>

The LowPass model proved to be more effective as an Infection Classifier, as we hypothesized in the introduction of the model due to the lung being a soft tissue. When implemented as Infection Classifier, the F2 Normal Score is increased up to 0.824 (from the original 0.767 as PreliminaryModel).

This however, did not have the same effects as a Covid-19 Classifier, where F2 Covid-19 Scores dropped to values ranging from 0.1 to 0.4.

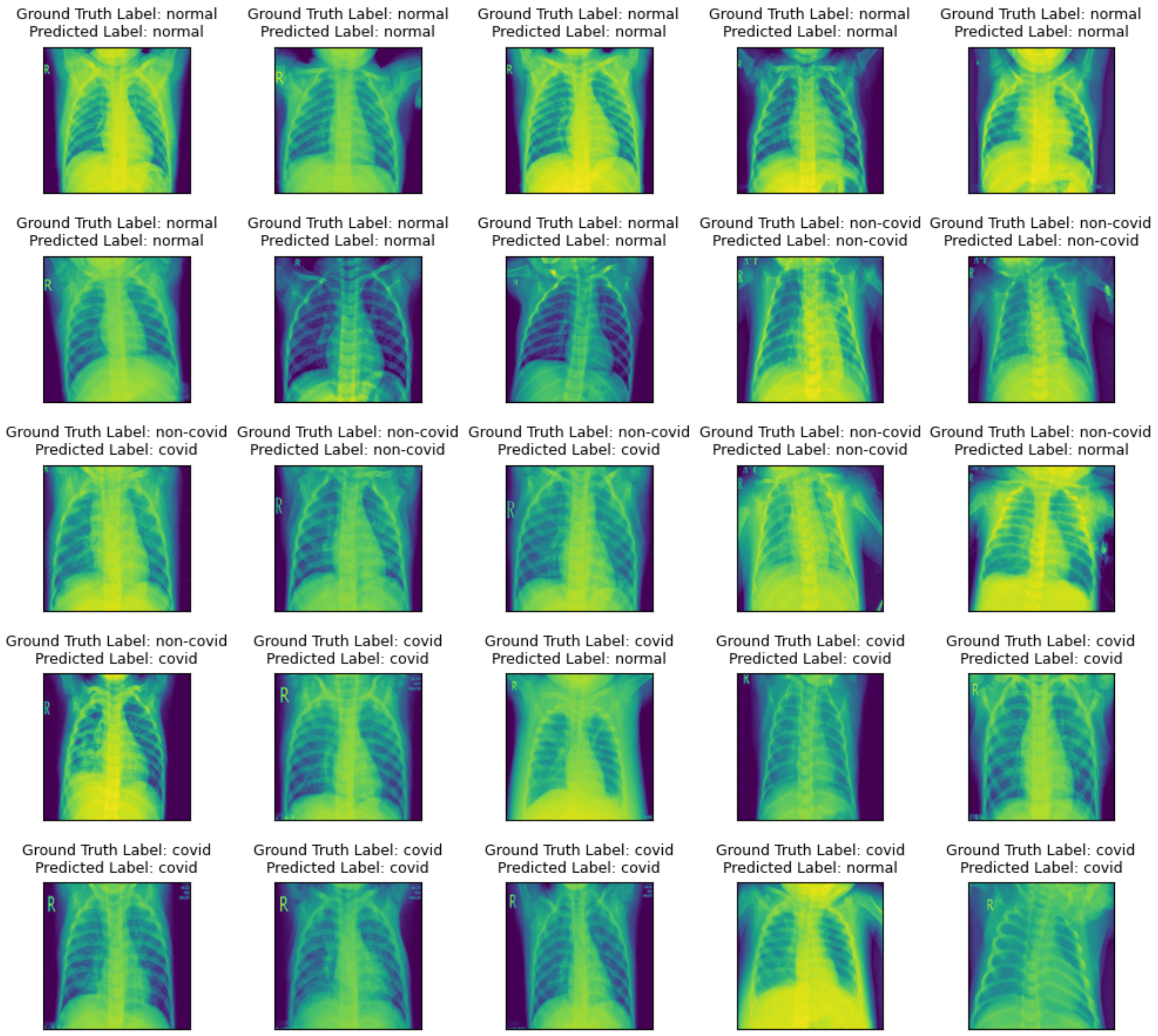
Conclusion

Amongst all the 25 combinations of models that we have tested out, we picked two that were respectively the highest overall accuracy and the highest F2 Covid-19 Score (that has a relatively high overall correctly predicted labels ie >0.77), which is the [ThreeModel, OneModel], and the [PreliminaryModel,OneModel].

Seeing the situation at hand globally, and for reasons mentioned before, we decided to proceed with the [PreliminaryModel,OneModel] which has a lower overall accuracy, though still quite high, but has a much higher F2 Covid-19 Score (0.645 as compared to 0.516).

Hence, we propose this cascaded model of the PreliminaryModel for the infected classifier, and the OneModel for the covid classifier as our final architecture, based on empirical evidence and all the conceptual reasoning that we went through before. We proceed to train it for 50 epochs and have uploaded the weights alongside the other trained models' weights. You can find them under "model/infected_PreliminaryModel_50.pt" and "model/covid_OneModel_50.pt". Below is the visualisation of the validation set.

Validation set pictures, with predicted and ground truth labels.
Covid F2-score: 0.761
Accuracy: 0.760


Ground Truth Label: normal
Predicted Label: normal
Ground Truth Label: normal
Predicted Label: normal
Ground Truth Label: normal
Predicted Label: normal
Ground Truth Label: normal
Predicted Label: normal
Ground Truth Label: normal
Predicted Label: normal
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: non-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: normal
Ground Truth Label: non-covid
Predicted Label: covid
Ground Truth Label: covid
Predicted Label: covid
Ground Truth Label: covid
Predicted Label: normal
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: normal
Ground Truth Label: covid
Predicted Label: covid

Citations

Katsamenis, I., Protopapadakis, E., Voulodimos, A., Doulamis, A., & Doulamis, N. (2020). Transfer Learning for COVID-19 Pneumonia Detection and Classification in Chest X-ray Images. Transfer Learning for COVID-19 Pneumonia Detection and Classification in Chest X-Ray Images, 1. https://doi.org/10.1101/2020.12.14.20248158

Keskar, N. S. (2016, September 15). On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. ArXiv.Org. https://arxiv.org/abs/1609.04836

Loshchilov, I., & Hutter, F. (2017). Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Van Mieghem, I. M., De Wever, W. F., & Verschakelen, J. A. (2005). Lung infection in radiology: a summary of frequently depicted signs. JBR-BTR : organe de la Societe royale belge de radiologie (SRBR) = orgaan van de Koninklijke Belgische Vereniging voor Radiologie (KBVR), 88(2), 66–71.

Weiss, K. L., Cornelius, R. S., Greeley, A. L., Sun, D., Chang, I.-Y. J., Boyce, W. O., & Weiss, J. L. (2011). Hybrid Convolution Kernel: Optimized CT of the Head, Neck, and Spine. American Journal of Roentgenology, 196(2), 403–406. https://doi.org/10.2214/ajr.10.4425

Appendix

The full results will be attached to this report as “appendix.pdf”