Deep Learning Small Project Report

He Yuhang 1003775
Zhang Jiazheng 1003772
Zhang Shaozuo 1003756

Dataset and DataLoader

Data Distributions

We observe that the classes are not equally distributed. For three classes architecture, the number of samples labeled as ‘infected non covid’ class is almost twice of the samples labeled as ‘normal’ class and ‘infected covid’ class in the training set, while in the testing set three classes are almost equally distributed.

For two classes architecture, the number of samples labeled as ‘infected’ is twice of the samples labeled as ‘normal’ in both the training set and testing set. For all infected samples, ‘non covid’ class dominates the ‘covid’ class in both the training set and testing set.

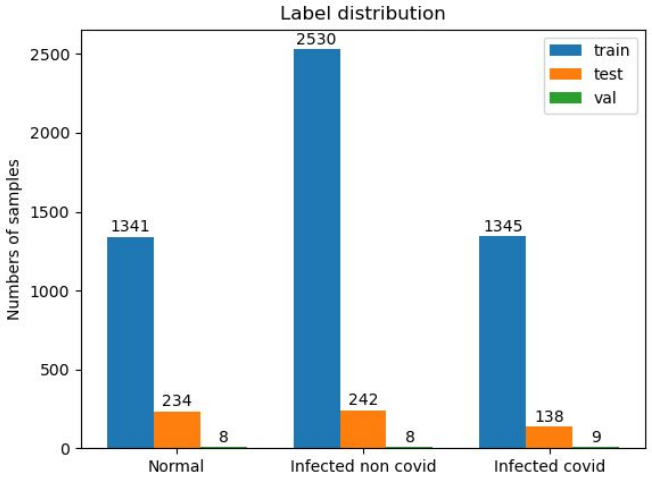


Figure 1.1.1: Three Class Distribution

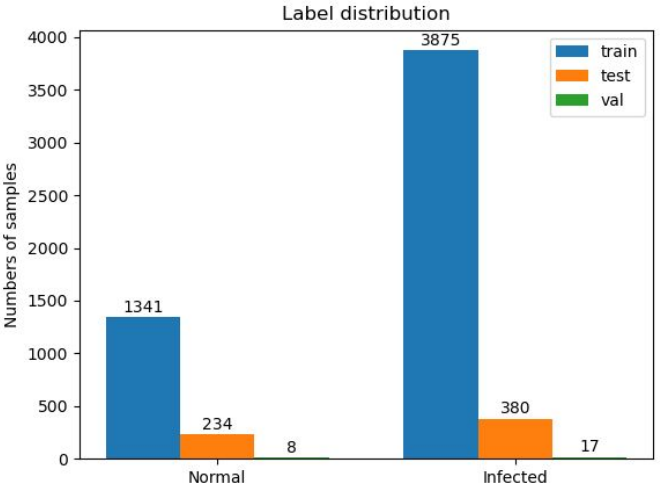


Figure 1.1.2: Normal and Infected Class Distribution

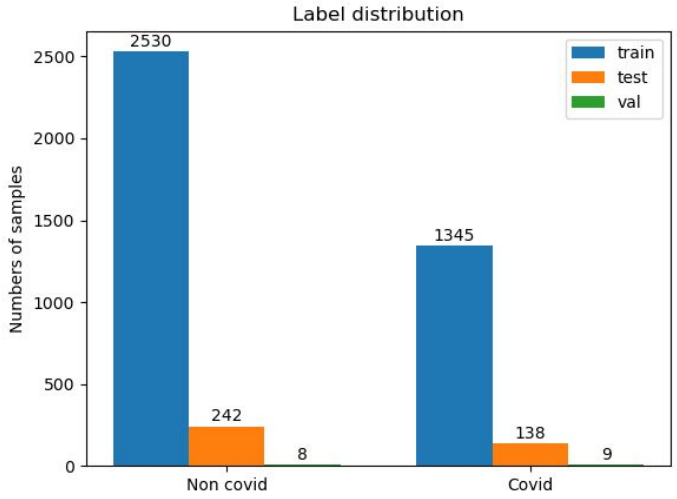


Figure 1.1.3: Non-COVID and COVID Class Distribution

Data Preprocessing and Augmentation

The code below shows the preprocessing steps and data augmentation steps we use.

```
self.transforms = transforms.Compose([
                transforms.Grayscale(),
                transforms.ColorJitter(0.1, 0.1, 0.1),
                transforms.RandomRotation(5, resample=Image.BILINEAR),
                transforms.RandomResizedCrop((150, 150), (0.7, 1.0)),
                transforms.ToTensor(),
                ])
```

Preprocessing

transforms.Grayscale: For an image, we first return the Grayscale version of the input since the original image is in gray style with three identical channels, Grayscale() reduces the output channel number to 1.

transforms.ToTensor(Normalization): Another important preprocess step is normalization. We normalize the pixel value [0,255] to [0,1]. Normalization process puts the data on the same scale. Scaling to [0,1] improves convergence speed and accuracy.

Augmentation

Motivations of Augmentation

- Image rotation and crop, brightness and contrast variation might occur in real world situations when X-ray is generated. In other words, the input images are not always orientated and scaled in the standard way. Our model should be robust to those changes.
- In addition, our dataset is relatively small. Data augmentation helps us enlarge our dataset so that the performance is better.
- Last but not least, data augmentation potentially prevents overfitting by introducing some noise.

Methodology

transforms.ColorJitter(0.1,0.1,0.1):
Change the brightness, contrast and saturation of the image. In our case, the brightness factor, contrast factor and saturation factor will all be chosen uniformly from [0.9, 1.1]. The reason such parameters are chosen is that: the transformed image will diverge too much from the original if we choose the parameter greater than 0.1. Here we didn't jitter hue because color is not a feature to learn.

transforms.RandomRotations(5, resample = Image.BILINEAR):
Randomly rotate the image with the degree [-5,+5]. The rotation degrees are kept to small values, otherwise much information will be lost.

transform.RandomResizedCrop((150,150),(0.7,1.0)):
Crop the image with the scale range from (0.7,1.0), then resize the shape to (150,150) which is the shape of original images.

Performance with Augmentation

With the optimized parameters epoch=35, learning rate=0.001, batch size=64 and optimizer=Adam, we compare the model's performance training on augmented data and non-augmented data.
To run the experiment, we plot the training loss, validation loss, training accuracy and testing accuracy over epochs. For the options of not doing augmentation, we simply return the grayscale of the input image in tensor format (value from [0,1]). The results are displayed below:

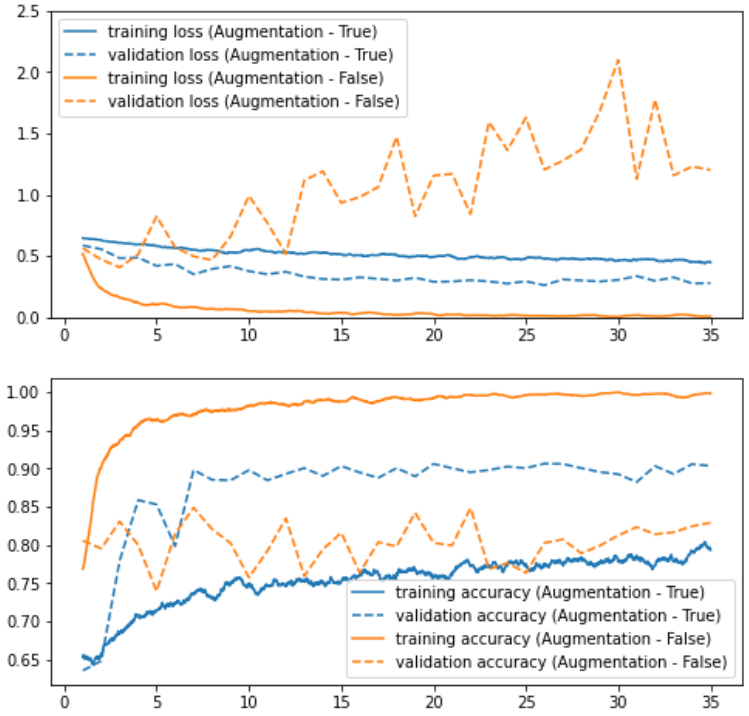


Figure 1.2.1:Loss and Accuracy Curve for Normal-Infected Classifier with and without Augmentation

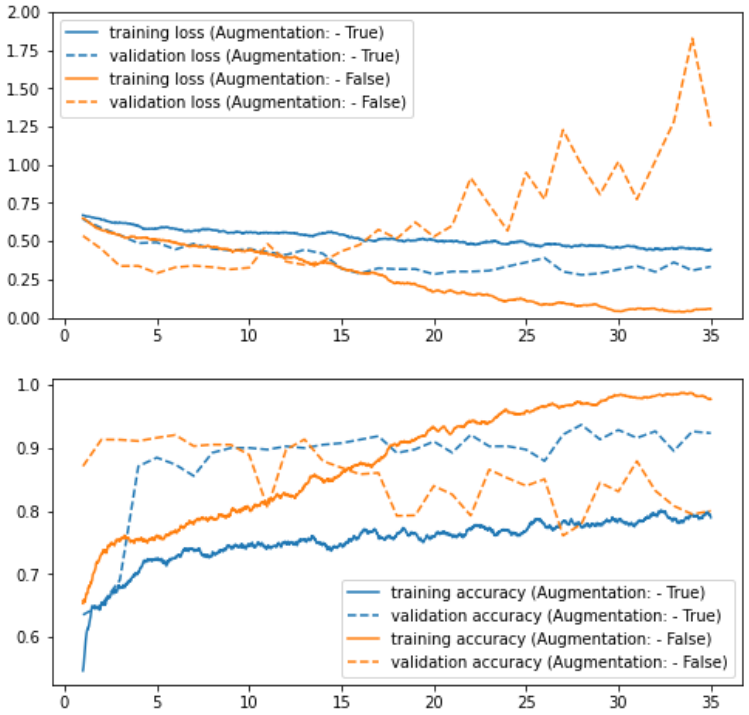


Figure 1.2.2:Loss and Accuracy Curve for COVID-nonCOVID Classifier with and without Augmentation

It is observed that for both tasks, the classifier starts to overfit quickly without data augmentation. And with the help of data augmentation, this problem is alleviated and the validation accuracy is higher, validation loss is smaller. Therefore, we can conclude that data augmentation prevents overfitting and helps improve the performance.

Model

Differences between Two Architectures

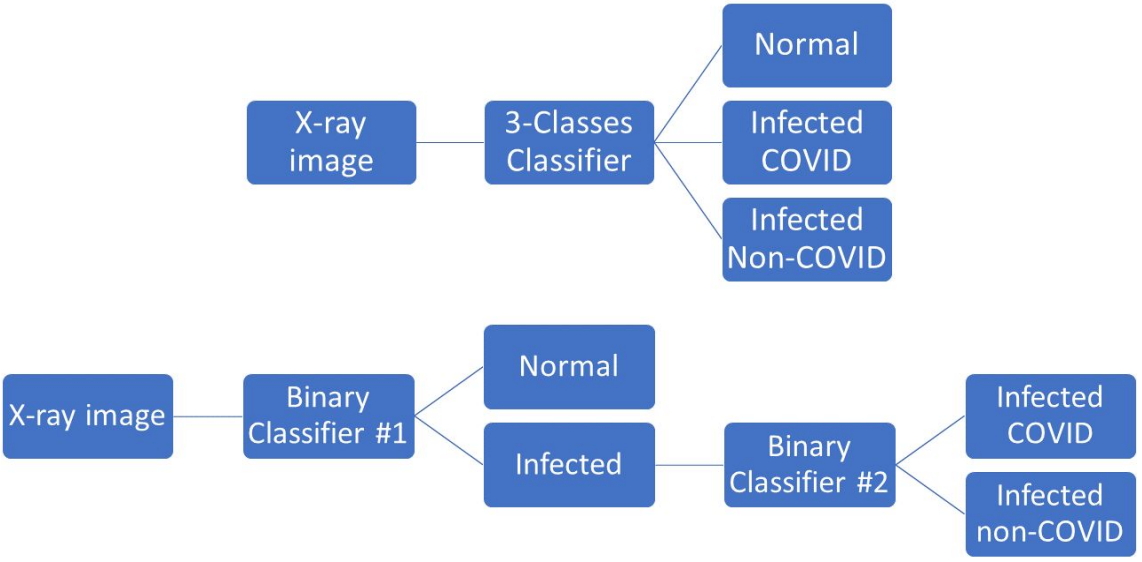


Figure 2.1.1 Two Model Strategies

- A three-classes classifier is a multi-class classifier, it has only one classifier and we can perform end-to-end training. For the combination of 2 binary classifiers, we cannot perform end to end training, we need to train the 2 binary classifiers separately and combine them in the end. As a result, the 2 binary models can be very different from each other.

- A three-classes use the same features to differentiate all 3 classes, whereas for combination of 2 binary classifiers, the logic of classification of normal and infected class and classification of COVID and nonCOVID class can be different.

Choice and Reasons

Choice:
Combination of 2 binary classifiers.

Reasons:

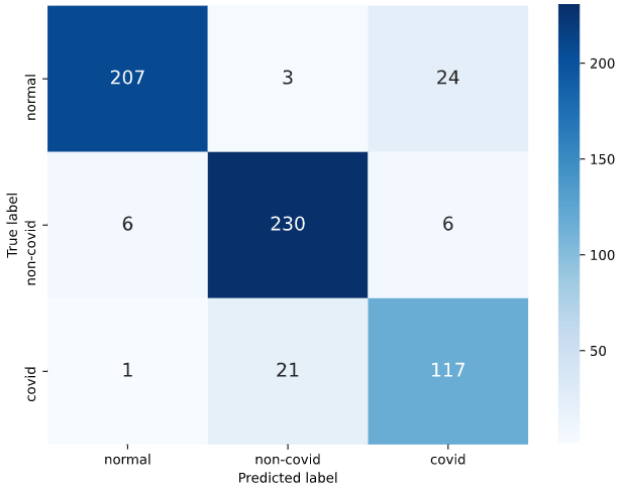
- Firstly, both COVID and nonCOVID classes fall under the infected (pneumonia) class. They do not really have the same hierarchy as the non-infected(non-pneumonia) class. As a result, it makes more sense to classify whether the image is from the infected class first, and then classify whether it is from COVID class.

- Secondly, COVID and nonCOVID classes are much more difficult to differentiate because they share the same features for the infected class; the two classes look very similar and only have subtle differences. Using 1 dedicated binary classifier to differentiate these 2 classes allow us to focus on the subtle differences that are hard to differentiate.
- Last but not least, based on the research, radiologists diagnose pneumonia through X-rays by looking at the presence of white spots, then examine whether it is COVID-19. This

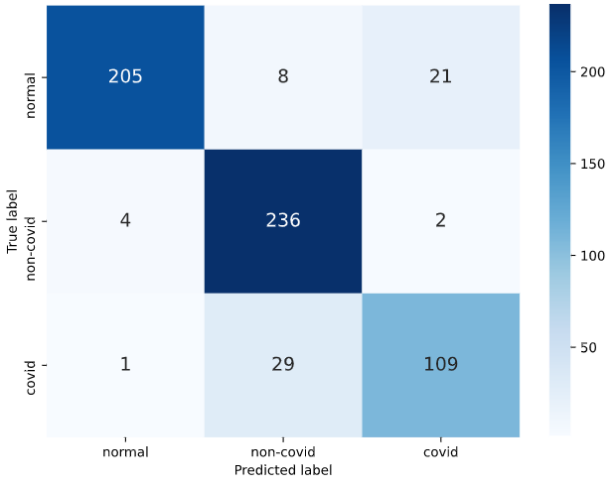
architecture simulates the thinking procedure of human beings. We expect the model can learn the features similar to human's observations. In this case the key feature is the white spots. After successfully classifying the X-rays as normal or infected, we can explore specifically what is the potential feature of covid-19 and even visualize it in feature maps.

- We tested both architectures with similar classifier structure. The results shown below suggest that 2 binary classifiers architecture did achieve slightly better performance.

	f1 - normal	f1 - non-covid	f1 - covid	accuracy
2 binary classifiers	**0.924107**	**0.927419**	**0.818182**	**90.08%**
3 classes classifier	0.923423	0.916505	0.804428	89.43%



Confusion matrix of 2 binary classifier architecture on validation set



Confusion matrix of 3 classes classifier architecture on validation set

Choice of Model Architecture

General strategy: Convolutional Neural Network(CNN) is good at extracting features for images, so CNN and its variation are our primary choices.

Binary classifier for normal and infected classes:

Architecture: 5 iterated CNN-ReLU-MaxPooling layers, followed by 2 fully connected layers as shown in the figure below.

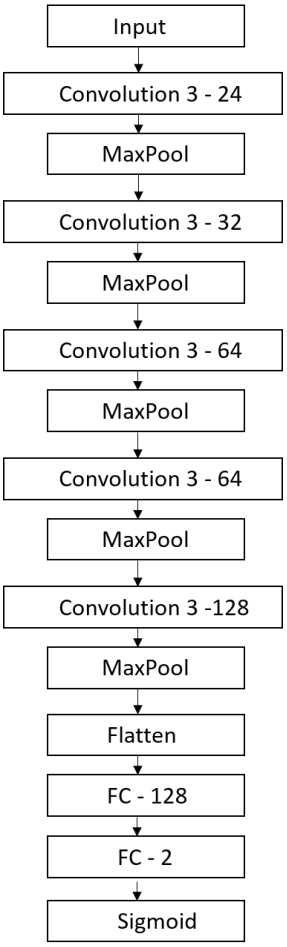


Figure 2.2.1 Normal-Infected Classifier Architecture
*Note: "convolution k - o", means kernel size k*k, o output channels*

Reasons:

- Iterated CNN-ReLU-MaxPooling is a traditional structure used in the VGG models, which achieves high performance on ImageNet. We gained inspiration from VGG to adopt the Iterated CNN-ReLU-MaxPooling layers.
- It turns out 5 Iterated CNN-ReLU-MaxPooling layers performs better than 3 or 4 iterated Iterated CNN-ReLU-MaxPooling layers. The feature map after 5 Iterated CNN-ReLU-MaxPooling is already as small as 2*2, so we cannot add any more CNN feature extraction layers after it. So we adopted 5 layer Iterated CNN-ReLU-MaxPooling as our model architecture for classifying normal and infected images. The below figure shows the loss and accuracy curve using different numbers of CNN layers, we can see that 5 CNN layers give better performance.

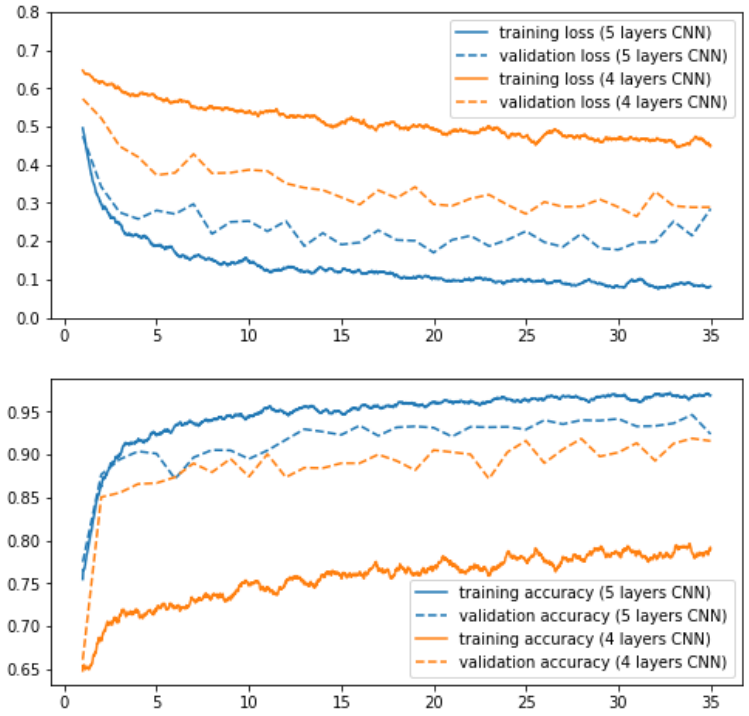


Figure 2.2.2 Loss and Accuracy Curve for Normal-Infected Classifier with Different CNN Layers

Binary classifier for COVID and nonCOVID classes:

Architecture: 10 CNN layers with residual connection, followed by 2 fully connected layers as shown in the figure below.

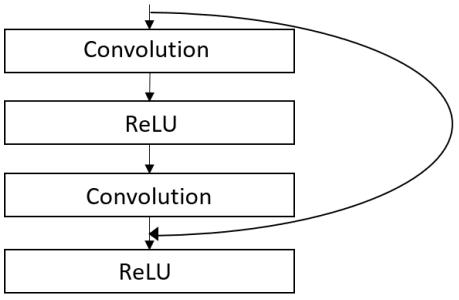


Figure 2.2.3 Residual Block

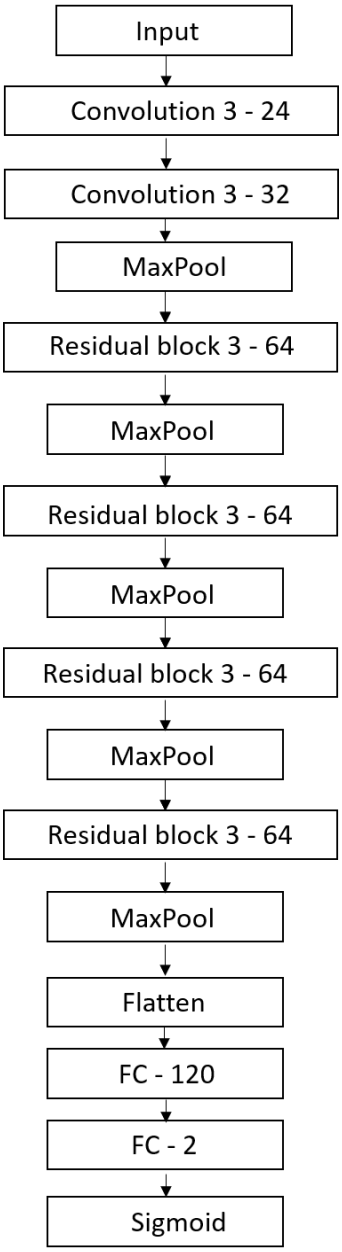


Figure 2.2.4 COVID-nonCOVID Classifier Architecture
*Note: "convolution k - o", means kernel size k*k, o output channels; "Residual block k - o" means residual block shown in figure 2.2.3, both convolution layers with kernel size k*k, and o output channels*

Reasons:

- Firstly, it is a more difficult problem compared to differentiating the normal and infected classes. As a result, it may require deeper networks to extract more complex features.
- Secondly, we experimented on models with 10 CNN layers and it converges very slowly on COVID and nonCOVID tasks, this is because it may have the vanishing gradient issue.
- To have better gradient flow and solve the vanishing gradient issue, we add residual connection to deal with this issue. And it turns out this gives better gradient flow and the vanishing gradient issue is alleviated.

Batch Size Choice

Choice: We chose 64 as our mini-batch size.

Reasons:
64 is the biggest batch size we can use on our machine. We experimented on batch size 32 and 64. The graphs below show the loss and accuracy curves for normal and infected classification as well as COVID and nonCOVID classification respectively. From the graphs, we can observe that using batch size 32 or 64 have similar performance on both tasks, and there are no significant differences.

Using batch size 64 gives faster training speed. It also has the least step size compared to smaller batch sizes. As a result, we choose batch size 64 for faster training and better utilization of our GPU resources.

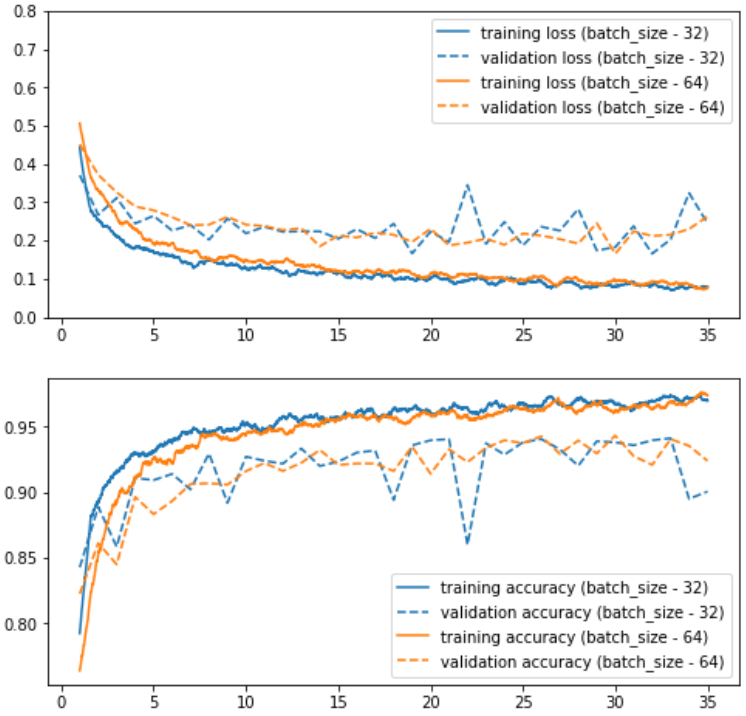


Figure 2.3.1 Loss and Accuracy Curve for Normal-Infected Classifier with Different Batch Sizes

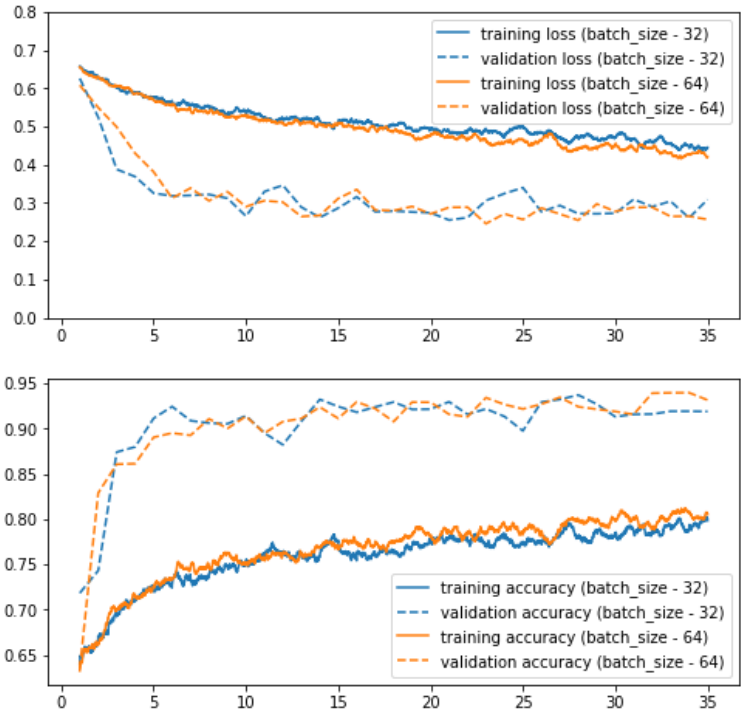


Figure 2.3.2 Loss and Accuracy Curve for COVID-nonCOVID Classifier with Different Batch Sizes

Loss Function and Parameter Choice

We use binary cross entropy as our loss functions since we are training binary classifiers.

Using binary cross entropy is equivalent to fitting the model using maximum likelihood estimation. Maximum Likelihood estimators have nice asymptotic properties (they are the best estimators in terms of convergence rates), they are consistent and statistically efficient.

Optimizer and Parameter Choices

Optimizer Choice:

We choose Adam as the optimizer.

Reasons:
We experimented on SGD, Adam, RSMprop optimizers using the default parameters for both classification tasks. The graphs below show the loss and accuracy curves for both classification tasks with different optimizers. It turns out Adam gives the best performance. As a result, we chose Adam as our optimizer.

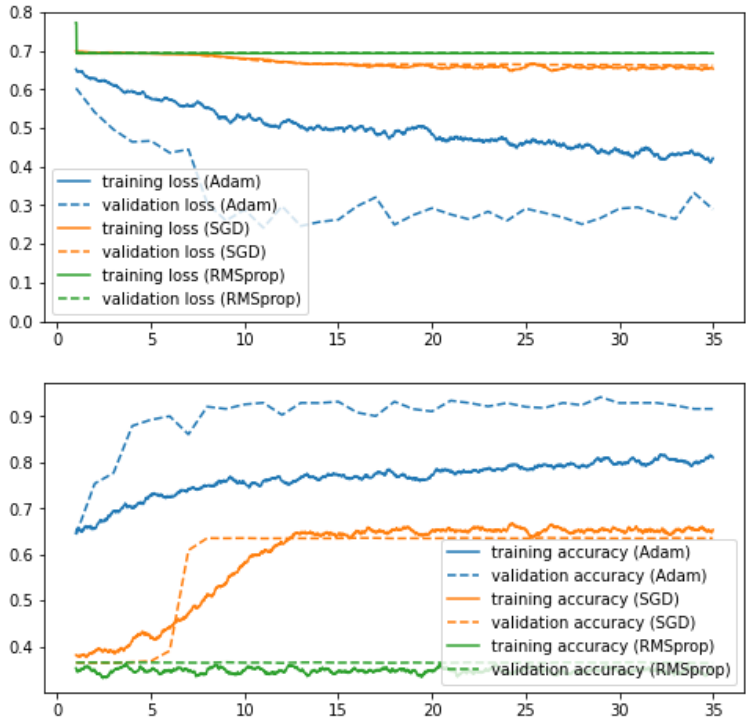


Figure 2.5.1 Loss and Accuracy Curve for COVID-nonCOVID Classifier with Different Optimizers

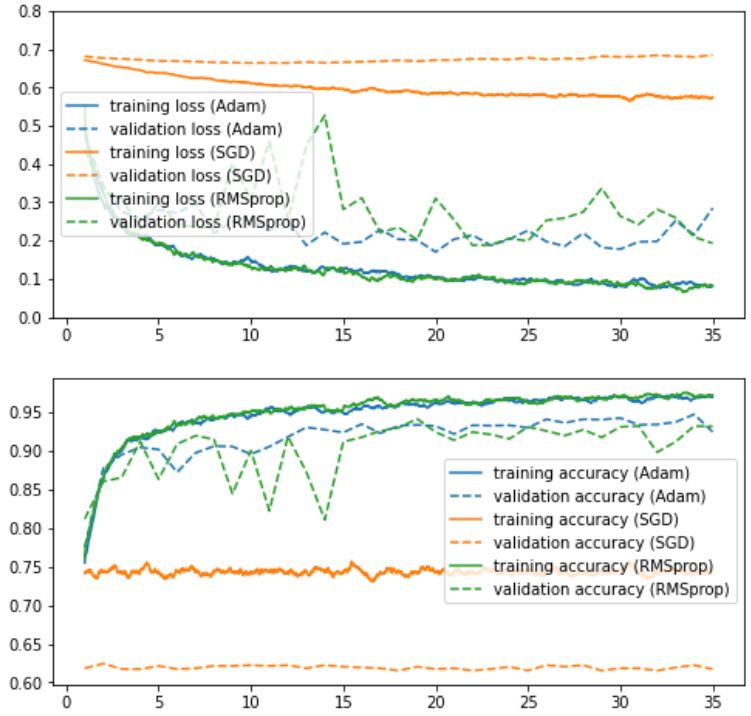


Figure 2.5.2 Loss and Accuracy Curve for Normal-Infected Classifier with Different Optimizers

Adam Parameter Choice

Learning Rate

We choose 0.001 as our learning rate.

Reason:
We experimented on different learning rates: 0.0001, 0.001, 0.01, 0.1, using Adam optimizer. The graphs below show the loss and accuracy curves for both classification tasks with different learning

rates. It turns out learning rate 0.001 gives the best performance, so we choose 0.001 as our learning rate.

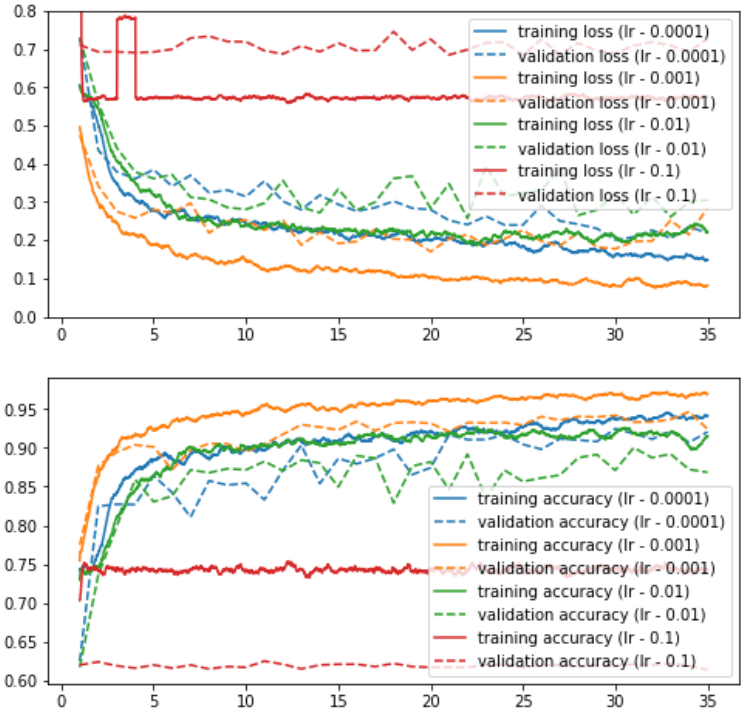


Figure 2.5.3 Loss and Accuracy Curve for Normal-Infected Classifier with Different Learning Rate(Adam)

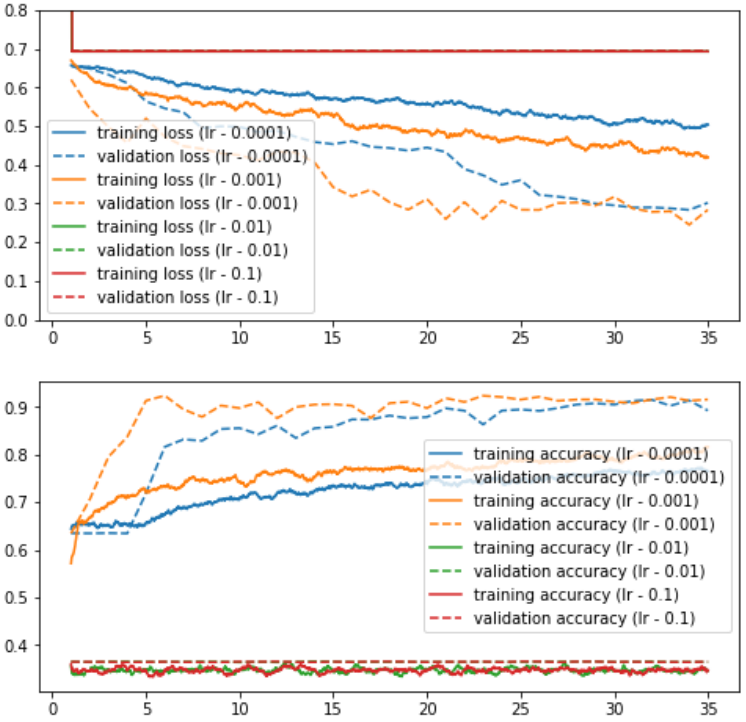


Figure 2.5.4 Loss and Accuracy Curve for COVID-nonCOVID Classifier with Different Learning Rate(Adam)

Weight Decay (Regularization)

We choose not to add weight decay to Adam optimizer.

Reason:
We experimented on different weight decay parameters: 0, 0.0005, 0.001, 0.01, using Adam optimizer and the chosen learning rate 0.001. The graphs below show the loss and accuracy curves for both classification tasks with different weight decay parameters. It turns out that weight decay 0.01 performs the worst, and the other 3 weight decay have similar performance. As a result, for simplicity, we choose 0 as our weight decay parameter, that means no weight decay is applied.

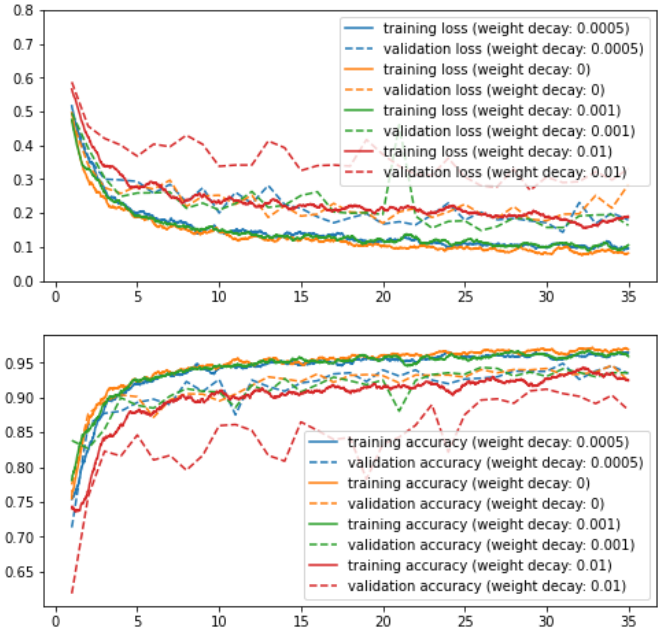


Figure 2.5.5 Loss and Accuracy Curve for Normal-Infected Classifier with Different Weight Decay Rate(Adam)

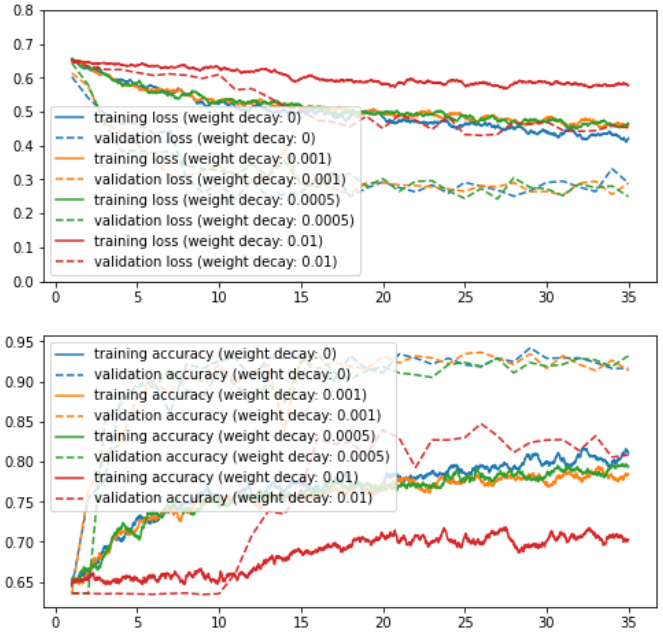


Figure 2.5.6 Loss and Accuracy Curve for COVID-nonCOVID Classifier with Different Weight Decay Rate(Adam)

Choice of Initialization for Model Parameter

In our model, default initialization methods are applied to layers. More specifically, weights for convolutional layers and fully connected layers are initialized by kaiming uniform distribution which is uniform distribution in range $[-gain \times \sqrt{3/fan_{in}}, gain \times \sqrt{3/fan_{in}}]$, and bias terms are initialized with uniform distribution in range $[-1/\sqrt{fan_{in}}, 1/\sqrt{fan_{in}}]$.

Loss Graph

Our final model and training strategy:
Model: combination of 2 binary classifiers explained in this section, applying dropout and batch normalization
Training Strategy:

- Normal-Infected classifier: Adam optimizer with learning rate 0.001, without weight decay, train 35 epochs
- COVID-nonCOVID classifier: Adam optimizer with learning rate 0.001, without weight decay, train 25 epochs

Loss graph for final model and training:

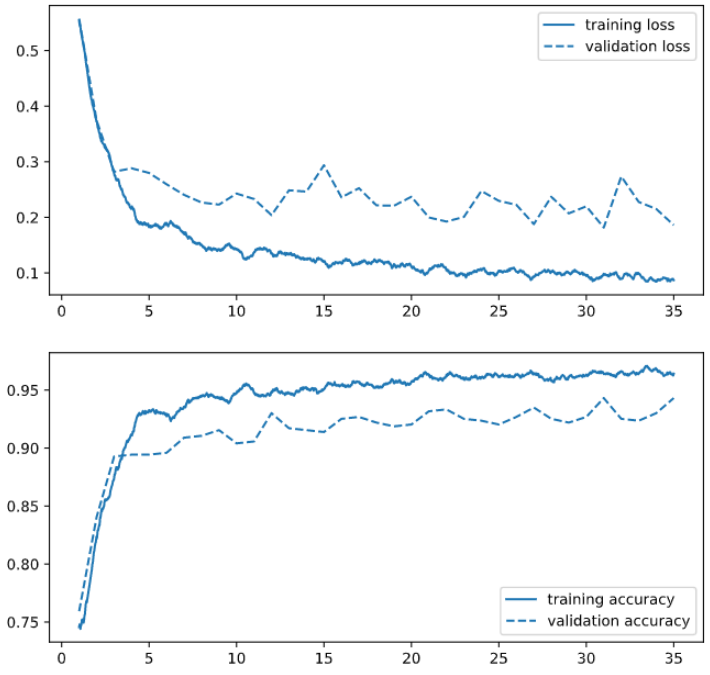


Figure 2.7.1 Loss and Accuracy Curve for Normal-Infected Classifier

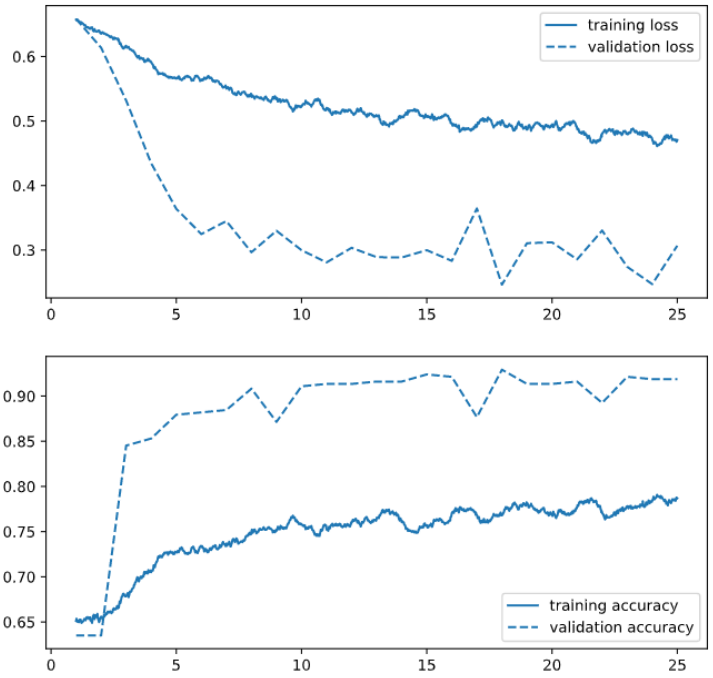


Figure 2.7.1 Loss and Accuracy Curve for COVID-nonCOVID Classifier

Testing Set Result and Visualization

Testing result on test set with cascade model

	recall	precision	f1 score	accuracy
normal	0.884615	0.967290	0.924107	
non-covid	0.950413	0.905512	0.927419	
covid	0.841727	0.795918	0.818182	
overall				0.9008

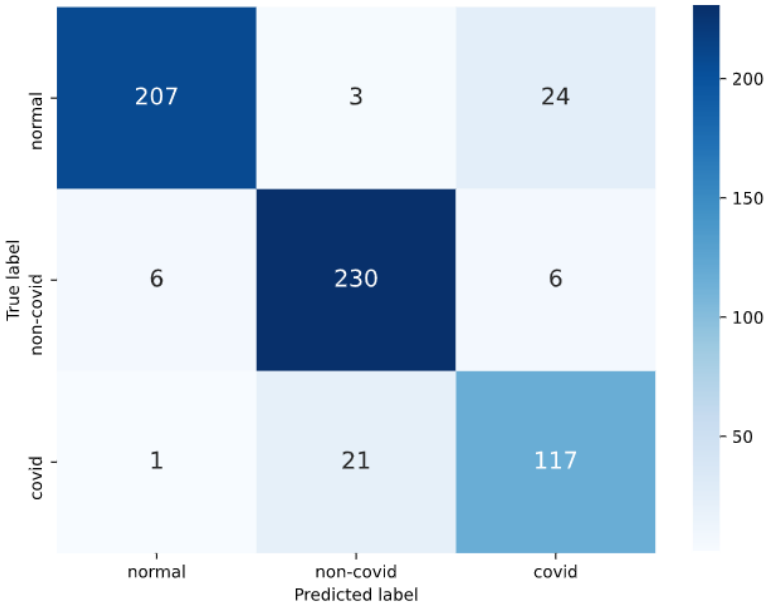


Figure 2.8.1 Confusion Matrix for Overall Model

Visualization on validation set (25 images)

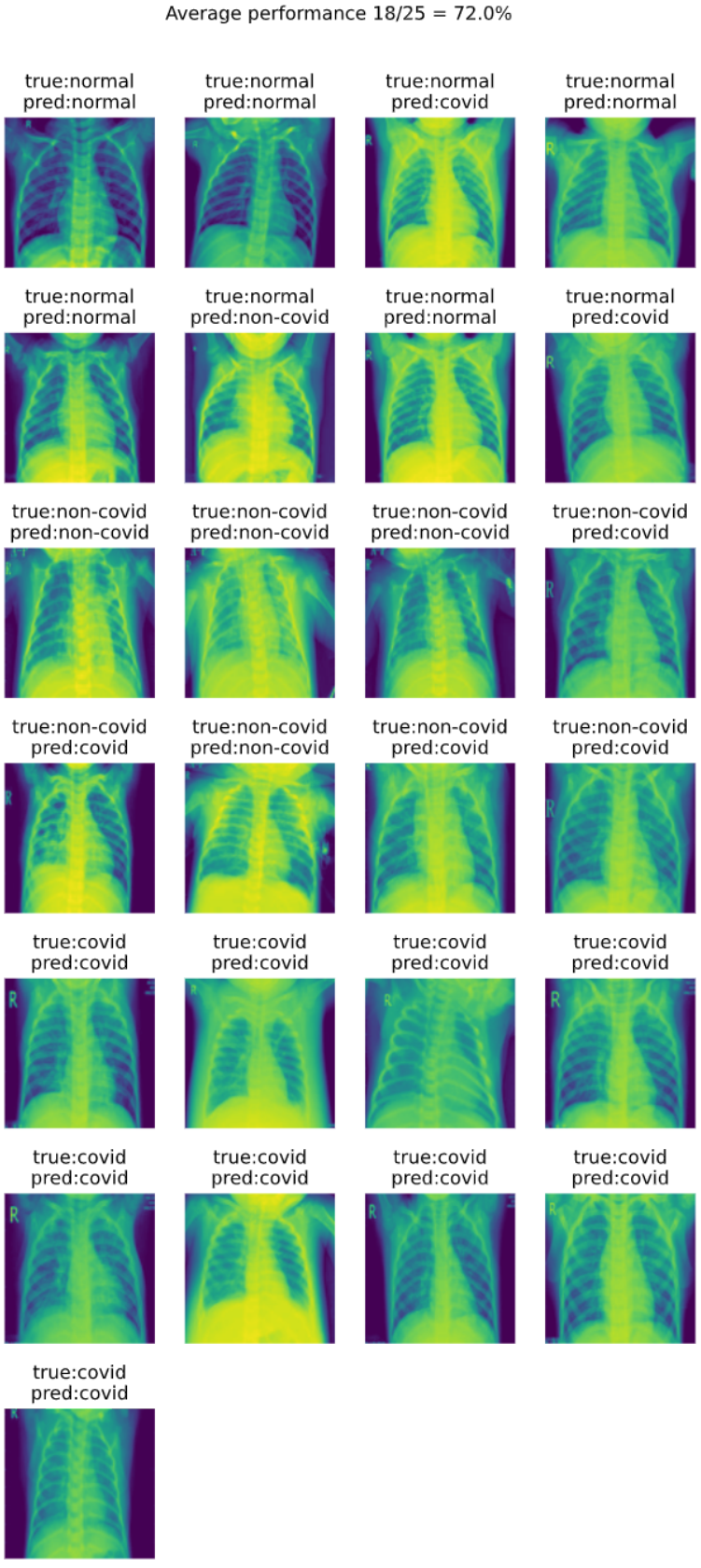


Figure 2.8.2 Visualization on Validation Set

Is it harder to differentiate covid and non-covid, was it expected?

Yes, and it is expected. No single feature on chest radiography is diagnostic of COVID-19 pneumonia, but a combination of multifocal peripheral lung changes of ground glass opacity and/or consolidation may present in covid-19 pneumonia, which increases the complexity of the features.

Is it better to have high overall accuracy or low false negatives/false positives rates on certain classes?

It depends on the use case of the classifier.

If our primary focus is one class, it is better to have a low negative and false positive rate. For example, if there are 2 classes A and B in the datasets, and 99% of the data are from class A and only 1% of the data are from class B, and the use case for the model is to identify class B from the datasets, for example, detects images with hate symbols images uploaded to Internet; in this case, we want to have low false negatives and false positive rates on class B as class B is our primary focus. And if we classify all images as class A, it has a very high accuracy of 99%, but it classifies all class B wrongly and the model does not help in our use case at all.

On the other hand, if the data is balanced distributed and we do not focus on 1 class, for example, classifying an image from ImageNet, it makes sense to prefer a high overall accuracy.

Bonus

Data Augmentation

We have discussed the data augmentation methods we use, the motivations and effects in this section.

Learning Rate Scheduler

Weight Decay

We experimented on weight decay strategy. In the end, we decided not to use it, and the details have been discussed in this section.

Regularization

Weight Decay

We experimented on weight decay strategy, which is equivalent to L2 regularization terms. In the end, we decided not to use it, and the details have been discussed in this section.

Dropout

We add dropout to our fully connected layers in both classifiers to help with the overfitting issue.

Batch Normalization

We apply batch normalization techniques in order to have better gradient flow in the network, and it helps to train the network faster.

Feature Maps

To understand how the model works, we visualized the output feature maps from every two convolutional layers of the COVID and nonCOVID classifier.

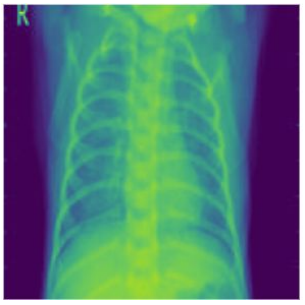


Figure 3.4.1 Original Image (COVID)

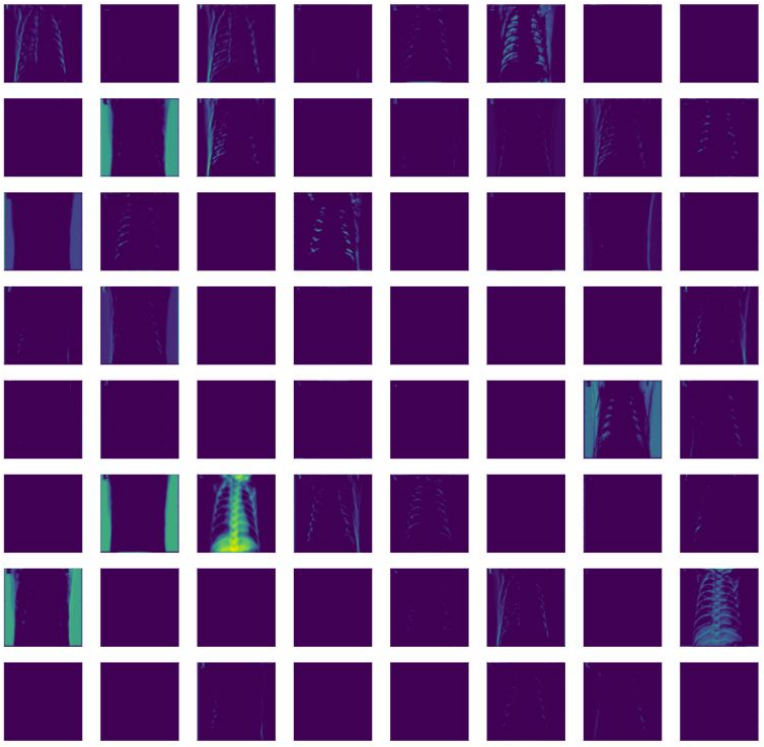


Figure 3.4.2 Feature Map of the 2nd Convolutional Layer

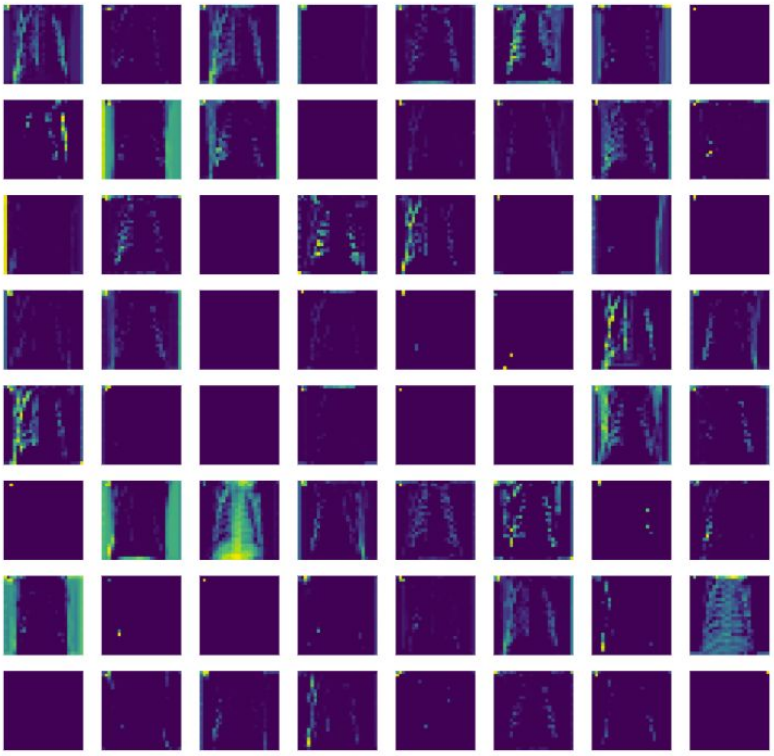


Figure 3.4.3 Feature Map of the 4th Convolutional Layer

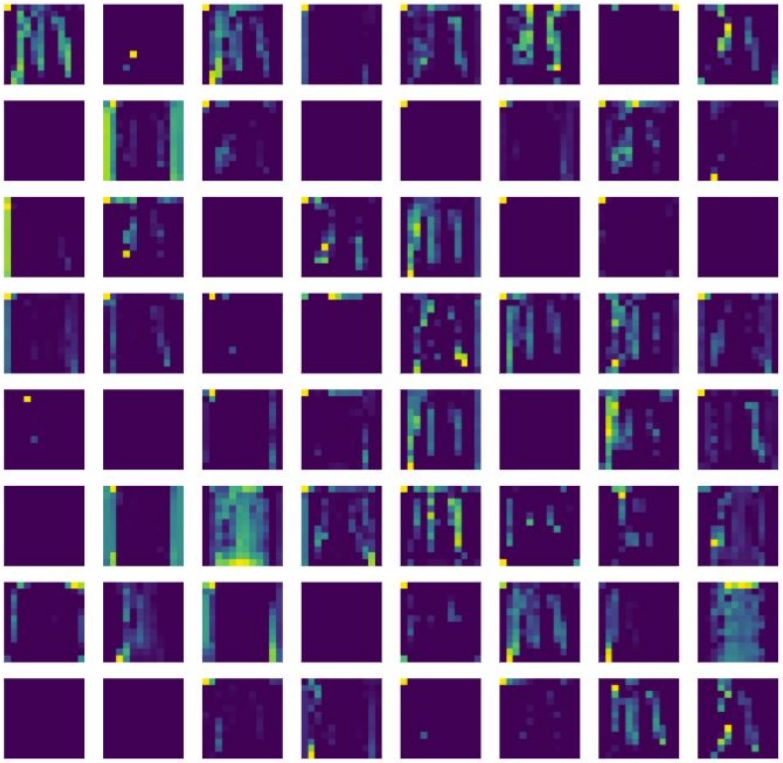


Figure 3.4.4 Feature Map of the 6th Convolutional Layer

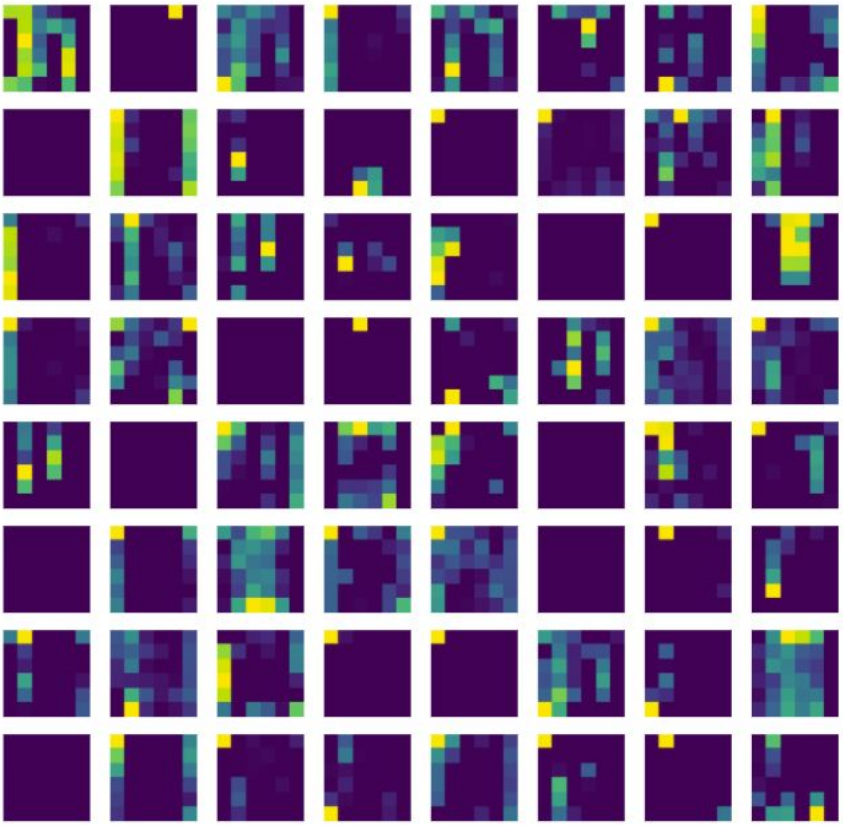


Figure 3.4.5 Feature Map of the 8th Convolutional Layer

References

Importance of Normalization: https://forums.fast.ai/t/images-normalization/4058/8
Diagnose covid-19 pneumonia with X-rays: https://www.bmj.com/content/370/bmj.m2426
Diagnose pneumonia with X-rays: https://www.radiologyinfo.org/en/info.cfm?pg=pneumonia
What makes binary cross entropy a better choice
https://datascience.stackexchange.com/questions/53400/what-makes-binary-cross-entropy-a-better-choice-for-binary-classification-than-o