



## 50.039 Deep Learning Small Project Report

Chen Yan (1003620)

Goh Yi Lin, Tiffany (1003674)

Xavier Tan De Jun (1003376)

## Index

- Background
- Explore Dataset
- Data processing operations and application
- Choice of Loss Function
- Learning Rate Schemes
- Evaluation
- Different choice of the size of mini batch training
- Different choice of initialization parameters
- Try different CNN model architecture (normal\_infected\_model)
- Try different CNN model architecture (covid\_non\_model)
- Final model Architecture
- Result Analysis

## Background

### 1. Project briefing

In this project, we will design a deep learning model to assist with the diagnosis of pneumonia, for COVID and non-COVID cases, by using X-ray images of patients. We trained two binary classifiers using the full dataset that was given and designed two CNN deep learning models to classify the X-ray images of patients to three different classes (normal, infected covid and infected non-covid).

## 2. The advantages of X-ray

Chest X-ray is the first imaging technique that plays an important role in the diagnosis of COVID-19 disease. Due to the high availability of large-scale annotated image datasets, great success has been achieved using convolutional neural networks (CNNs) for image recognition and classification.

## 3. The Comparison of X-ray images characteristic for different classes

The X-ray of images have different characteristics for different classes. The COVID-19 was more likely than non-COVID-19 pneumonia to have a high-grade ground glass opacities ( $P = .01$ ), extensive lesion distribution ( $P < .001$ ), mixed lesions of varying sizes (27.7% vs 57.0%,  $P = .001$ ), subpleural prominence (23.4% vs 86.7%,  $P < .001$ ), and lower lobe prominence (48.9% vs 82.0%,  $P < .001$ ). However, peribronchial interstitial thickening was more likely to occur in non-COVID-19 viral pneumonia (36.2% vs 19.5%,  $P = .022$ ). The statistically significant differences from multivariable analysis were the degree of ground glass opacities ( $P = .001$ ), lesion distribution ( $P = .045$ ), lesion size ( $P = .020$ ), subpleural prominence ( $P < .001$ ), and lower lobe prominence ( $P = .041$ ). The sensitivity and specificity of the model were 94.5% and 76.6%, respectively, with an AUC of 0.91.

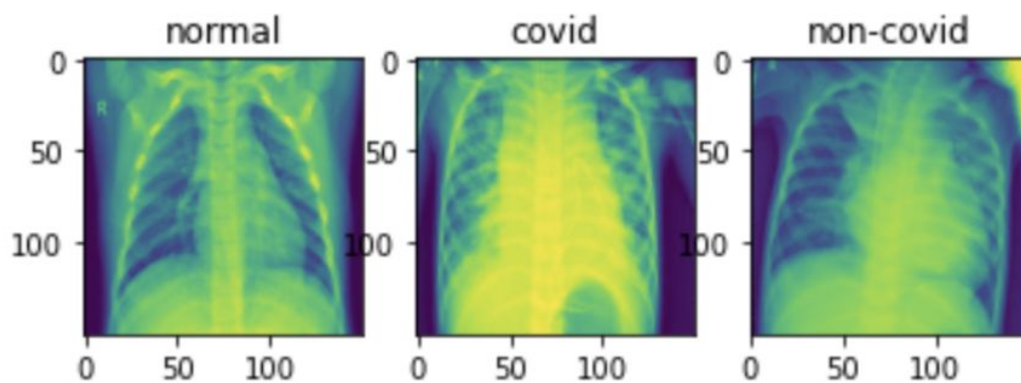
However, a point to note, using X ray for detection may be insufficient as other methods such as swab test may be required to accurately diagnose a patient with COVID.

## Explore Dataset

### 1. Explore dataset in details

The full dataset has three different sub-datasets(training, validation,test) for each different class, namely normal, infected and non covid and infected and covid. It contains 1341 images for the train dataset, normal class, 2530 images for the train dataset, infected and non-covid class,1345 images for the train dataset, infected and covid class,234 images for the test dataset, normal class,242 images for the test dataset, infected and non-covid class,139 images for the test dataset, infected and covid class,8 images for the val dataset, normal class,8 images for the val dataset, infected and non-covid class, 9 images for the val dataset, infected and covid class.

Here is the X-ray images(0-0) of three different classes:



0-0

### 2. Check the balance between classes (uniformly distributed)

Balanced dataset means the dataset has positive values which are approximately the same as negative. Imbalanced dataset means there is a very high difference between the

positive values and negative values. When we are looking through the full dataset(0-1-1), we can see there is a high difference between the normal, infectedcovid and infectednon classes in the training set. In the test and validation set, the difference between three classes is so small that we can ignore. Therefore, the training dataset is slightly unbalanced with around 1000 more infected but non-covid images, the test and validation dataset are both well-balanced between classes.



0-1-1

### Data processing operations and application

1. The reasons of normalization for images

Normalization is a process that changes the range of pixel intensity values. It also is called contrast stretching or histogram stretching sometimes. The purpose of normalization in the images is usually to bring the image into a range that is more

familiar or normal to the senses. Our dataset consists of X-ray images. Medical images like X-ray images are often corrupted by noise and affected by artifacts. Some of the texture-based features that should describe the structure of the tissue under examination in the X-ray image may also reflect. It may lead to an inappropriate description of the tissue or incorrect classification. To limit these phenomena, we need to do image normalization. Normalization also can help to update weights converge fast.

## 2. Other preprocessing operations that are needed for images.

For image preprocessing, we still can do image cropping, noise removal and histogram equalization. The size of our image is  $150 \times 150$  and there is still 10% of the image comprising the background with a lot of noise. We can crop the images and guarantee that all the regions of interest contain the lesion areas meanwhile avoid the useless information. Moreover, these images are scanned at different illumination conditions, so some images appear too bright and some are too dark. Therefore, we can use noise removal and histogram equalization to increase the contrast range in an image by increasing the dynamic range of grey levels. It is utilized to enhance the image for demeaning the effects of over-brightness and over-darkness in images.

## **Choice of Loss Function**

### 1. The difference between different loss function and parameters

We tried to use four different loss functions(Cross-entropy, Hinge loss, BCE loss and Squared hinge-loss) to evaluate our analysis results. Cross-Entropy Loss is the preferred loss function under the inference framework of maximum likelihood. It is the loss function to be evaluated first and only changed if you have a good reason. Cross-entropy will calculate a score that summarizes the average difference between the actual and predicted probability distributions for predicting class 1. The score is minimized and a perfect cross-entropy value is 0. Hinge Loss is intended for use with binary classification where the target values are in the set  $\{-1, 1\}$ . The hinge loss function encourages examples to have the correct sign, assigning more error when there is a difference in the sign between the actual and predicted class values. Squared Hinge-Loss simply calculates the square of the score hinge loss. It has the effect of smoothing the surface of the error function and making it numerically easier to work with. If using a hinge loss does result in better performance on a given binary classification problem, it is likely that a squared hinge loss may be appropriate. BCE loss is a softmax activation plus a Cross-Entropy loss.

## 2. The reasons of our choice for loss function(BCE loss)

BCE computes sigmoid predictions independently for each class, it is a softmax activation plus a Cross-Entropy loss. For BCE, an instance is allowed to be both class-A and class-B at the same time, which is better for multi-label tasks (e.g. OpenImage dataset). From our performance results, we can see that the accuracy, precision and recall are much higher with BCE loss function than other loss functions.

## Choice of Optimiser

### 1. The difference between different optimiser and parameters

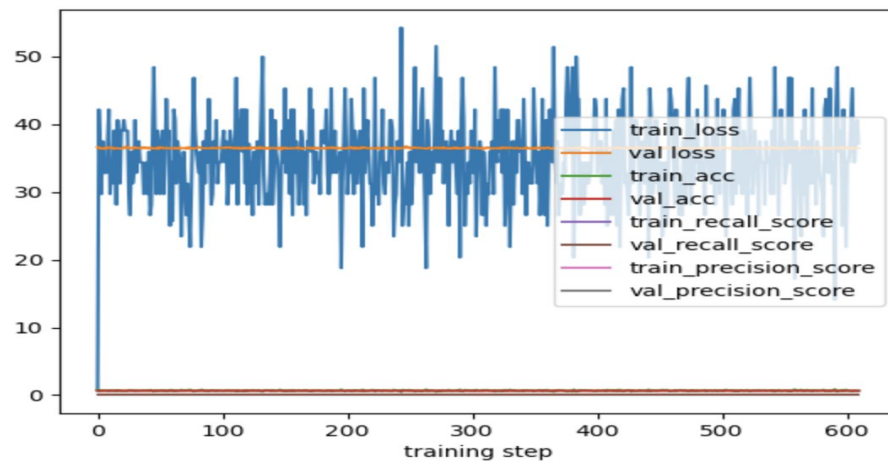
We tried to use three different optimisers( SGD,RMSprop, Adam) to change the attributes of our neural network such as weights and learning rate in order to reduce the loss. SGD is a variant of Gradient Descent, it tries to update the model's parameters more frequently. The parameters in SGD are altered after computation of loss on each training example. RMSprop tries to dampen the osciations and restricts the oscillations in the vertical direction, but in a different way than momentum. It takes away the need to adjust learning rate and does it automatically and choses a different learning rate for each parameter. Adam is an algorithm that combines the heuristics of both Momentum and RMSProp.

Algorithms	Advantages	Disadvantages
SGD	<ul style="list-style-type: none"><li>- Frequent updates of model parameters hence, converges in less time.</li><li>- Requires less memory as no need to store values of loss functions.</li><li>- May get new minima.</li></ul>	<ul style="list-style-type: none"><li>- High variance in model parameters.</li><li>- May shoot even after achieving global minima.</li><li>-To get the same convergence as gradient descent needs to slowly reduce the value of learning rate.</li></ul>
RMSprop	<ul style="list-style-type: none"><li>- RMSProp converge faster than GD or SGD</li><li>- RMSProp might be able to converge to solutions with better quality (e.g. better local minima)</li></ul>	<ul style="list-style-type: none"><li>- Thelearning rate is too early and the excess is reduced</li></ul>
Adam	<ul style="list-style-type: none"><li>- The method is too fast and converges rapidly.</li><li>- Rectifies vanishing learning rate, high variance.</li></ul>	<ul style="list-style-type: none"><li>- Computationally costly.</li></ul>



## 2. The reasons of our choice( Adam optimizer)

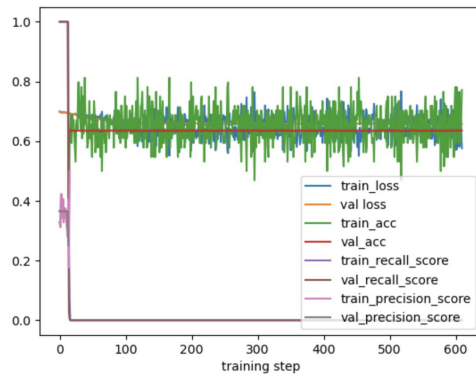
We are performing experiments on the full dataset using networks with RMSProp, Adam and SGD. We set three different models to judge the performance of these three optimizers(Model2\_2,Model2\_3,Model2\_4).We achieved 87% accuracy with SGD(learning rate of 0.1) and training loss almost remains constant. When testing the same exact configuration with RMSProp and Adam as well as the initial learning rate of 0.001,we achieved the highest accuracy of 86% and 89% and a significantly less smooth



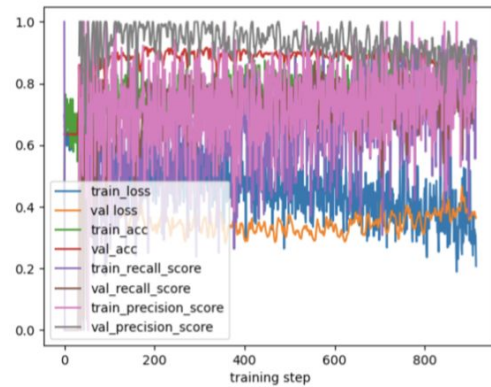
training  
curve with  
gradually  
decreasing  
training

loss(G0-3).Therefore, the Adam optimizer is the best optimizer for our model. (The result is on Table 1-3 )

RMSprop Optimizer G0-1



SGD Optimizer G0-2



Adam Optimizer G0-3

### Choice of Learning Rate

- In some optimization literature, it shows that increasing the learning rate can compensate for larger batch sizes. Therefore, we increase the learning rate for our model with increasing batch size to see if we can recover the asymptotic test accuracy we lost and improve the performance of our model. In a word, the model will achieve better accuracy with a lower batch size or higher learning. After we tried different learning rates, we decided to set the learning rate to be 0.1.

Learning rate	Evaluate metric
0.1	Training Loss: 0.594 - Validation Loss: 0.603 - Validation Accuracy: 0.620
0.15	Training Loss: 0.569 - Validation Loss: 0.685 - Validation Accuracy: 0.620
0.20	Training Loss: 0.572 - Validation Loss: 0.679 - Validation Accuracy: 0.620
0.25	Training Loss: 0.609 - Validation Loss: 0.675 - Validation Accuracy: 0.620
0.50	Training Loss: 0.609 - Validation Loss: 0.673 - Validation Accuracy: 0.620

## **Evaluation**

### **1. Accuracy**

Accuracy is the simple ratio between the number of correctly classified points to the total number of points. However, it has some limitations of accuracy. For accuracy, the probability of the predictions of the model can be derived. It is used when the True Positives and True negatives are more important. So if we only use accuracy, we can not measure how good the predictions of the model are.

### **2. Other performance metrics(Precision , recall)**

Besides accuracy, we also will use precision and recall to evaluate the performance of our model. Precision helps us understand how useful the results are. It's implied as the measure of correctly identified positive cases from all the predicted positive cases. It's useful when the costs of False Positives are high. Recall helps us understand how complete the results are. It's the measure of the correctly identified positive cases from all the actual positive cases. It's important when the cost of False Negatives is high.

## **Different choice of initialization of the size of mini batch training**

Batch size is one of the most important hyperparameters to tune in modern deep learning systems. Larger batch sizes make larger gradient steps than smaller batch sizes for the same number of samples. Practitioners often want to use a larger batch size to train their model as it allows computational speedups from the parallelism of GPUs. However, it is well known that too large of a batch size will lead to poor generalization. For convex

functions that we are trying to optimize, there is an inherent tug-of-war between the benefits of smaller and bigger batch sizes. On the one extreme, using a batch equal to the entire dataset guarantees convergence to the global optima of the objective function. However, this is at the cost of slower, empirical convergence to that optima. On the other hand, using smaller batch sizes have been empirically shown to have faster convergence to “good” solutions. This is intuitively explained by the fact that smaller batch sizes allow the model to “start learning before having to see all the data.” The downside of using a smaller batch size is that the model is not guaranteed to converge to the global optima. It will bounce around the global optima, staying outside some  $\epsilon$ -ball of the optima where  $\epsilon$  depends on the ratio of the batch size to the dataset size. Therefore, under no computational constraints, it is often advised that one starts at a small batch size, reaping the benefits of faster training dynamics, and steadily grows the batch size through training, also reaping the benefits of guaranteed convergence. Therefore, we set the batch size to 3 for two binary classifiers.

### **Different choice of initialization of model parameters**

1. It is quite important that we consider the initialization of parameters when we build a neural network. We will be achieved in the least time to converge to a minima if we do initialization correctly. For SGD the weights are initialized to approximately the magnitude you want them to be and most of the learning is shuffling the weights along the hyper-sphere of the initial radius. As for ADAM, the model completely ignores the initialization.

2. There are four techniques that we use to initialize parameters, namely zero initialization, random initialization, He-et-al initialization and Xavier Initialization .

- Zero initialization sets all the weights to zeros. It will cause neurons to perform the same calculation in each iteration and produce the same outputs. The derivatives will remain the same for every  $w$  in  $W[l]$ . Therefore, neurons will learn the same features in each iteration. Zero initialization is no more powerful than linear model or logistic regression, it always causes a bad performance.
- Random initialization is generally used to break the symmetry and this process gives much better accuracy than zero initialization. Neurons will not learn the same features of its inputs but can learn different functions of its input. However, if the weights initialized randomly very high or low, it may have some limitations. If the weights are initialized very high, the term  $\text{np.dot}(W, X) + b$  becomes larger. Then applying a sigmoid function of our model architecture, maps the value to 1 which causes slower change in the slope of gradient descent. As a result, learning takes a lot of time. If the weights are initialized with much lower value, sigmoid function tends to map the value to zero and eventually it slows down the optimization.

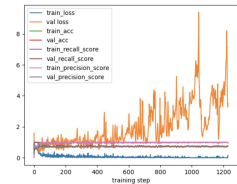
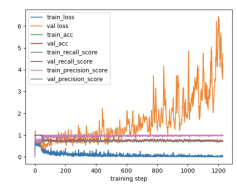
3. The reasons of our choice of Xavier initialization for model parameters

However, both zero initialization and random initialization have vanishing or exploding gradient problems. Therefore, we use new initialization techniques to initialize our model. We tried two more initialization methods(He-et-al and Xavier Initialization). He-et-al initialization is almost similar to Xavier initialization except that they use

different scaling factors for the weights. The scaling factor for Xavier initialization is  $\sqrt{1./\text{layers\_dims}[l-1]}$  and the scaling factor for He-et-al is  $\sqrt{2./\text{layers\_dims}[l-1]}$ . In Xavier initialization, we want the variance to remain the same for each passing layer. It helps to keep the signal from exploding to a high value or vanishing to zero. Xavier initialization is the same as He initialization but it is used for tanh() activation function. Based on the performance results, we can see the training loss of our model is quite smaller than other initialization and it gets the highest recall and precision value on two binary classifier models(Validation recall:0.991, Validation precision:0.747 for norm\_infected\_model; Validation recall:0.792, Validation precision:0.949 for covid\_non\_model). Hence we chose Xavier as our final initialization methods.

### Try different CNN model architecture (norm\_infected\_model)

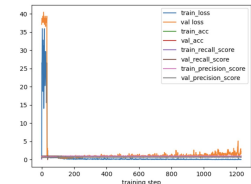
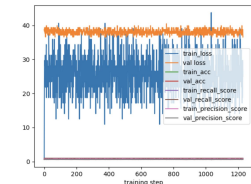
Adam Optimiser (Table 1-1)

	Dropout	Layer	Initialisation	Evaluate metrics	Graph
Model1_2	25%	Added 25% dropout layer on fc1 and fc2	default	test_loss 5.301449453830719 acc tensor(0.7451) recall tensor(0.9975) precision tensor(0.7099)	
Model1_3	40%	Added 40% dropout layer on fc1 and fc2	default	test_loss 3.6674819946289063 acc tensor(0.7381) recall tensor(1.) precision tensor(0.7026)	

Modell_4	40%	normal + dropout 40%, decrease nodes from 100 to 60 in fc1 layers	default	test_loss 3.6674819946289063 acc tensor(0.7381) recall tensor(1.) precision tensor(0.7026)	
Modell_5	40%	remove conv3, decrease nodes in fc layers	default	test_loss 4.421181488037109 acc tensor(0.6900) recall tensor(0.9948) precision tensor(0.6675)	
Modell_8	35%	Nodes Fc1 200 Nodes Fc2 100	default	test_loss 2.098197114467621 acc tensor(0.7606) recall tensor(0.9949) precision tensor(0.7241)	
Modell_9	35%	Original architecture	Xavier Initialization	test_loss 3.560723733901977 acc tensor(0.7883) recall tensor(0.9913) precision tensor(0.7470)	
Modell_10	35%	Original architecture	Xavier Initialization	test_loss 2.33152197599411 acc tensor(0.7572) recall tensor(0.9949) precision tensor(0.7198)	
Modell_11	35%	Original architecture	Kaiming_normal Initialization	Training loss:0.034 Validation loss: 2.258 Validation Accuracy:0.725 Validation recall:0.998 Validation precision:0.700	

Model1_12	35%	Original architecture	Kaiming_normal Initialisation	test_loss 3.4483689188957216 acc tensor(0.7768) recall tensor(0.9842) precision tensor(0.7403)	
-----------	-----	-----------------------	-------------------------------	---	--

RMSprop Optimiser (Table 1-2)

	Dropout	Layer	Initialisation	Validation Accuracy	Graph
Model1_6	35%	remove conv3, decrease nodes in fc layers	default	test_loss 2.2705860257148744 acc tensor(0.6844) recall tensor(1.) precision tensor(0.6624)	
Model1_7	35%	Nodes Fc1 200 Nodes Fc2 100	default	Training loss:25.00 Validation loss: 36.56 Validation Accuracy:0.634 Validation recall:1.0 Validation precision:0.634	

Try different CNN model architecture (covid\_non\_model)(Table 1-3)

	Optimizer	Initialisation	Evaluate metrics
Model2_2	Adam	default	Training loss:0.387 Validation loss: 0.331 Validation Accuracy:0.893 Validation recall:0.784 Validation precision:0.910
Model2_3	RMSprop	default	Training loss:0.389 Validation loss: 0.339 Validation Accuracy:0.862



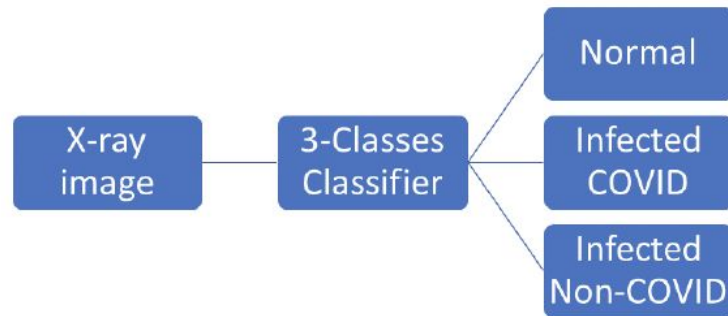
			Validation recall:0.798 Validation precision:0.885
Model2_4	SGD	default	Training loss:0.432 Validation loss: 0.339 Validation Accuracy:0.874 Validation recall:0.707 Validation precision:0.903
Model2_5	Adam	He-initialisation	Training loss:0.314 Validation loss: 0.383 Validation Accuracy:0.913 Validation recall:0.831 Validation precision:0.925
Model2_6	Adam	Xavier Initialisat ion	Training loss:0.501 Validation loss: 0.319 Validation Accuracy:0.906 Validation recall:0.792 Validation precision:0.949

## Model Architecture

1. The reason of our choice(two binary classifiers) architecture

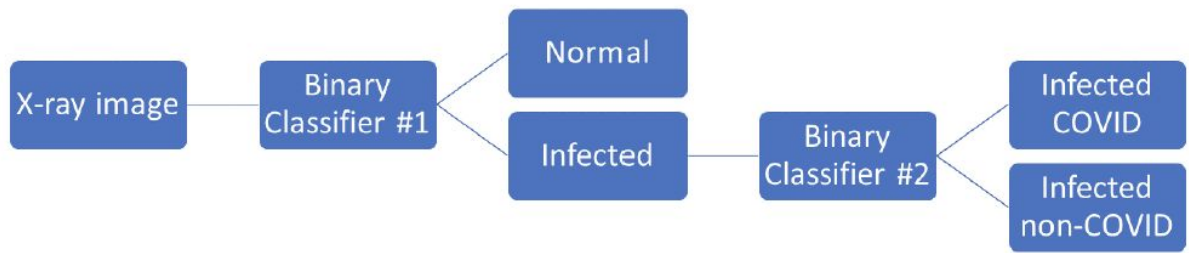
Architecture 1 uses a multiclass neural network with a softmax output layer to determine which label out of the three does each lung x-ray picture correlate to. Having more output nodes will cause our model to have higher complexity, it will also cause distinct features

between certain classes to become obsolete, and thus give poorer results.



While architecture 2 uses multiple binary classifiers, one One-vs-Rest classifier and one common binary classifier, to determine each pictures' label. Although approach number 2 can be computationally more intensive in terms of classification speed and resource consumption for large datasets, multiple & slow models (i.e. neural networks), having several binary classifiers might give more robust and increased classification accuracy as it might be better at learning specific features pertaining to its class. Binary classifiers are also more robust against poor hyperparameter selection and unoptimized network structures which is helpful since we will train our models from scratch. Similar study also suggested that “using an OvR type system, i.e.,” can yield better accuracy and general improved classification performance. (Riegler et al., 2018). Thus, we chose to follow architecture 2 since there is no need for real-time diagnosis (faster computations), such

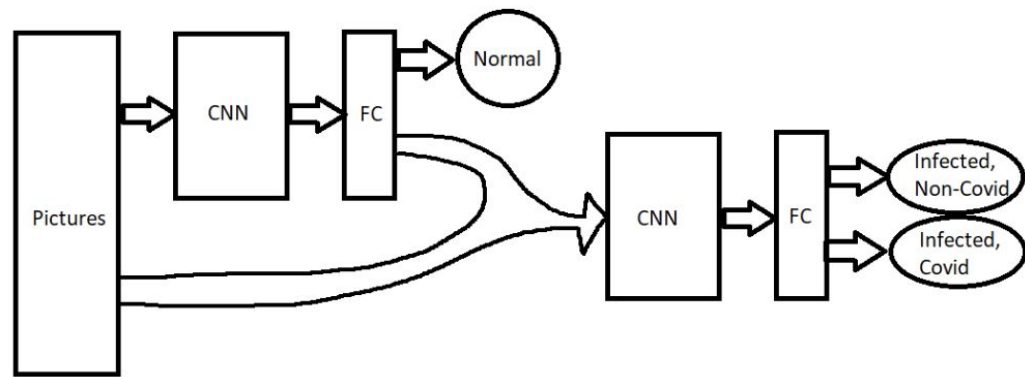
that we can generate more accurate results.



## 2. Our CNN architecture

- The whole architecture for our projects.

We trained two binary classifiers using the full dataset that was given as inputs and designed two CNN deep learning models to classify the X-ray of patients to three different classes( normal, infected covid and infected non-covid). We trained the 1st binary classifier using the full dataset as inputs and classified normal and infected classes. We trained the 2nd binary classifier using the infected dataset as inputs and classified infected covid and infected non covid classes. During the test process, we passed the test dataset to the 1st binary model to classify the normal and infected classes firstly, then we use the output of 1st binary classifier as the input of 2nd classifier to pass through the 2nd classifier to classified the infected covid and infected non covid. Finally, we classify three different classes successfully.

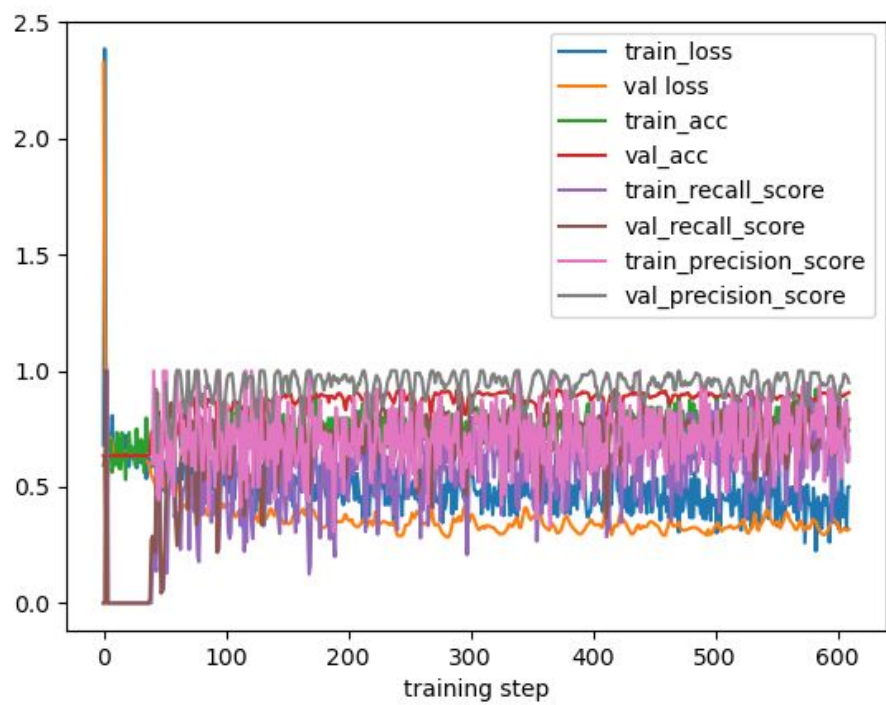
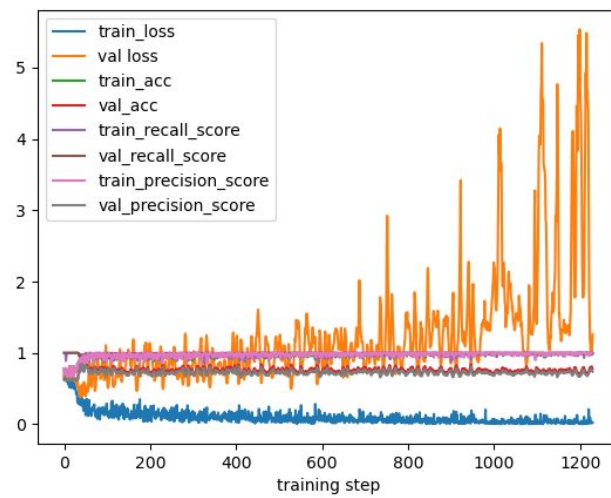


- The architecture of two binary classifiers

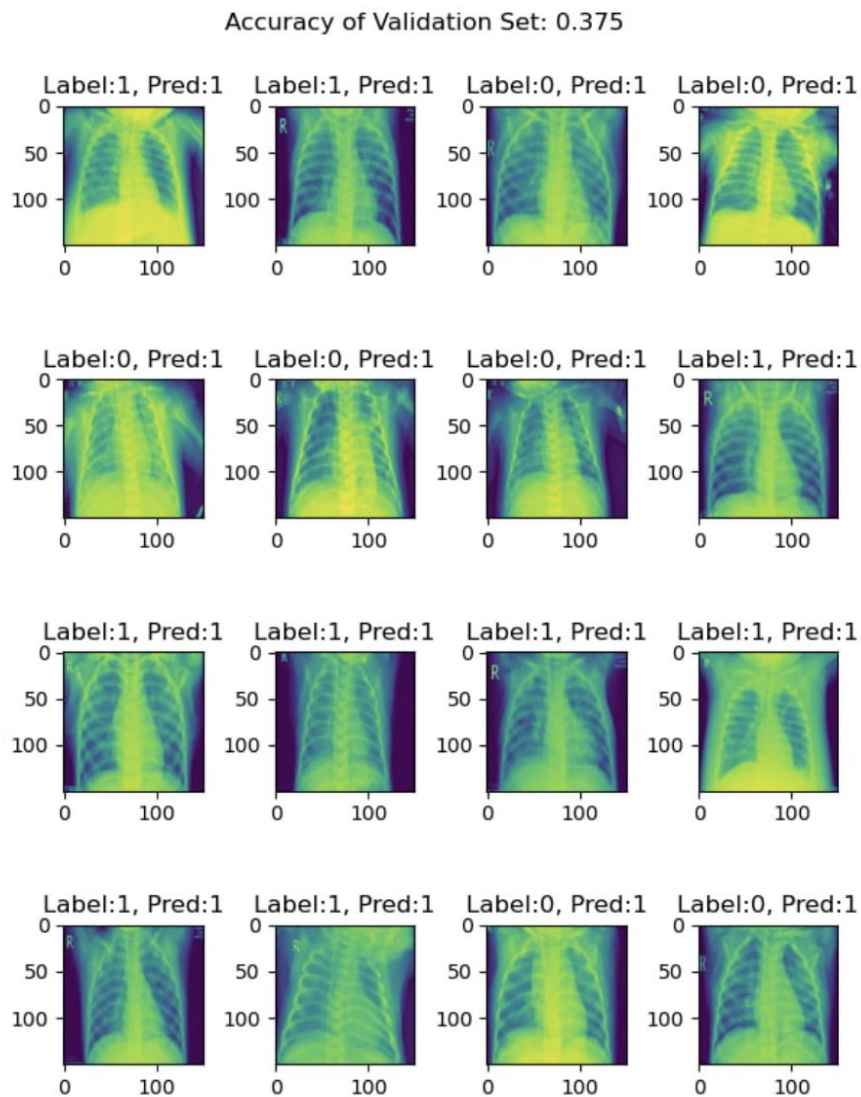
We use the same CNN architecture for two binary classifiers. During training, the input to our ConvNets is a fixed-size  $150 \times 150$  RGB image. Firstly, The image is passed through a convolutional (conv1) layer, where we use a kernel with a very small receptive field:  $3 \times 3$  (which is the smallest size to capture the notion of left/right, up/down, center). After that, we pass to Max-pooling, which is performed over a  $2 \times 2$  pixel window, with stride 2. Secondly, the image continues to pass through a stack of convolutional (conv2,conv3) layers with  $3 \times 3$  kernel. A stack of convolutional layers (which has a different depth in different architectures) is followed by three Fully-Connected (fc1,fc2,fc3) layers. The final layer is the sigmoid layer. The configuration of the fully connected layers is the same in all networks.

## Result Analysis

1. Here is the learning curves showing the evolution of loss function, precision and recall over successive training epochs for both train and test sets



- 
2. The performance of the trained model on the 24 images in the validation set.



- 
- 
- 3.

- 
- 
- 
4. The reasons why it is 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.

- The imaging characteristics of COVID-19 and non-COVID-19 viral pneumonia are a bit similar, but normal and infected classes have large differences in X-ray images.

Moreover, the features of covid and non-covid are more complex, the training process may be more difficult. Therefore, it is more difficult to differentiate between non-covid and covid x-rays, rather than between normal x-rays and infected (both covid and non-covid) patients x-rays.

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

Accuracy is the simple ratio between the number of correctly classified points to the total number of points. However, it has the limitations of accuracy. From accuracy, the probability of the predictions of the model can be derived. It is used when the True Positives and True negatives are more important. So from accuracy, we can not measure how good the predictions of the model are. Therefore, we still need precision and recall. Precision is implied as the measure of the correctly identified positive cases from all the predicted positive cases. Thus, it is useful when the costs of False Positives are high. Recall is the measure of the correctly identified positive cases from all the actual positive cases. It is important when the cost of False Negatives is high.

For our projects, if we misclassify normal to covid, it should be ok. However, if we misclassify covid to normal, it may cause big problems. Therefore we are better to evaluate a model not only using accuracy but also using low false positives rates on certain classes.

## References

Michael, A. R., Håvard, E., Pia, H. S., Konstantin, P., Pål, H., Tor Jan, D. B., Thomas de L. & Hakon, K. S. (2018).

[https://www.researchgate.net/publication/330256512\\_Tradeoffs\\_Using\\_Binary\\_and\\_Multiclass\\_Neural\\_Network\\_Classification\\_for\\_Medical\\_Multidisease\\_Detection](https://www.researchgate.net/publication/330256512_Tradeoffs_Using_Binary_and_Multiclass_Neural_Network_Classification_for_Medical_Multidisease_Detection)

<https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/>

<https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9>

<https://medium.com/analytics-vidhya/accuracy-vs-f1-score-6258237beca2>