Report for Small Project

Ryan Gen Zi Qiang (1003479)
Wilbert Aristo (1003742)

Instructions to run

1) Make sure the “Small Project Final.ipynb” file is in the same directory as “model.py” and “train.py”
2) Run all cells in the .ipynb file

Instructions to load and evaluate model without training

1) Make sure the “Small Project Final.ipynb” file is in the same directory as “model.py”
2) Open the .ipynb file and run all cells above Section 8 (“Running the Model”)
3) Uncomment the code snippet in the cell above Section 8. It should look like the following:

```
from model import load_model, evaluate_without_training

healthy_model = load_model("health_model.pt")
covid_model = load_model("covid_model.pt")
evaluate_without_training(healthy_model, device, first_test_loader)
evaluate_without_training(covid_model, device, infected_test_loader)
```

4) Run the cell, this will report the model’s accuracy on a testing dataset.

Data processing? Why normalization for the images needed? Any other pre-processing operations that are needed for our images?

We did not do any pre-processing operations for our images apart from normalization of images (dividing each pixel value by 255). We normalize the images such that the pixels in the images have a similar data distribution and help speed up convergence when training the network. There was also no need to resize the images as we were using our own model.

Differences between the 2 architectures and why binary?

The first architecture is a 3-class classifier model that will get the difference between normal, COVID-infected, and non-COVID-infected lung X-ray images. Meanwhile, the second architecture is to construct 2 separate classifiers where the first binary classifier is to differentiate between normal and infected lung X-ray images (COVID-infected and non-COVID infected combined) and the second binary classifier is to differentiate between COVID-infected and non-COVID infected lung X-ray images. This is essentially a discussion between a one vs one (2 binary) and a one vs all(multiclass) model architecture.

We decided to go ahead with the second architecture which is to create 2 binary classifiers. We felt this was definitely the better option because intuitively the model should be to

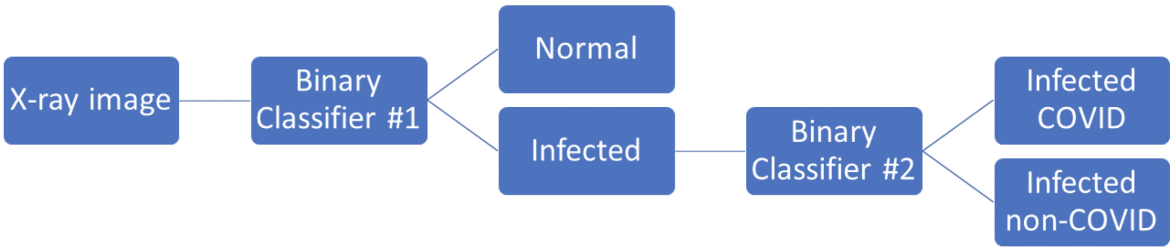
differentiate between the 2 categories and more importantly learn the features that differentiate between the 2 categories. When training a 3 class classifier for this scenario, it is trying to learn the decision boundaries between all 3 classes that separate the examples of each class from examples of all other classes.

If images of different classes resemble each other quite a lot, learning such decision boundaries can be especially complicated and unnecessary. If you look at this particular example, we do not need to learn the boundaries between normal lung images and COVID-infected lung images as well as normal lung images and non-COVID lung images. Instead, we want to find out what features are unique to an infected lung image (be it COVID or non-COVID) and not in normal lung images.

Moving forward with an infected image, it is also important to be able to differentiate COVID/non-COVID pneumonia for diagnosis. In this case, it is essential to have a high-accuracy differentiator and also learn the boundaries between COVID-infected X-ray and non-COVID-infected X-ray. By using a 2 binary classifier architecture, each model is able to focus on differentiating and learning these unique features and hypothetically lead to better recognition performance.

Choice of architecture

As concluded previously, we will go ahead with the 2 binary classifier architecture. The system will look like the image shown below(taken from the pdf provided)



We start by training Binary Classifier 1 which is our Health_Model which differentiates between normal and infected/sick patients. We will also train Binary Classifier 2 which is our Covid_Model which separates the infected/sick patients between Infected Covid and infected non-Covid Patients.

When running inference on this model. We will pass the images into both models but for the images that have been classified as infected from the Health_model binary classifier, these will be relabelled according to the predicted classes from the Covid_Model binary classifier.

Our architecture is a subclass of the NN.Module.

For our Health_model, for each layer we have a group of functions of <Conv -> Relu -> Batch Normalisation> for each convolutional group layer. Following that we will have a max pool layer of size 2 and apply a dropout layer after. We then have 2 fully connected layers with Relu operation applied and one final dropout layer before the last fully connected layer (classifier) is used. Then we apply a softmax to get the probability for each class.

Similarly, for the COVID model it follows the same model architecture. However, we have left out the dropout layers because we felt that the images between the Non-covid and Covid are very similar. It is likely that the differences are minute and thus we have decided not to include the dropout layers in the COVID model.

For the kernel sizes that we have chosen, we decided to use 3x3 kernels. This is because for these x-ray images the differences would be less obvious and require a smaller kernel size to learn these less obvious differences.

For the number of kernels in each convolutional layer. We decided to multiply them by 2 as we go to each layer. The common practice is to have a smaller number of filters at the start and increase it as you go deeper into the model because of more features needing to be extracted as more combinations arise in the later layers. In addition, for a complex task such as this, we would need more filters to find the minute differences.

We included Relu layers to introduce non-linearity to the model. We also included batch normalisation to reduce internal covariate shift so that the model can be trained in less epochs. It also offers some regularisation impact, which will lessen generalisation error and overfitting.

Finally, dropout is used in the Health_Model in order to reduce overfitting by randomly switching off some neurons in the deep learning model.

Number of Layers

We decided to run some experiments to decide for the appropriate number of convolutional group layers. We have limited this to 3-4 because we felt that 2 layers was too simple a model and when we tried 5 layers, we were met with a “CUDA out of memory error”. We felt that this was not too important as well because due to the limited dataset that was provided, a deep neural network would not be the best idea due to overfitting.

The results of our experiments can be seen below:

	Health_Model				Covid_Model			
Metric	train acc	val acc	train loss	val loss	train acc	val acc	train loss	val loss
3 Layers	89	82	0.2458	1.6801	78	83	0.5444	0.436
4 Layers	85	79	0.3199	0.9689	84	79	0.7691	0.8578

	Label	Precision	Recall	F1
3 Layers	Healthy	0.875	0.875	0.875
	Covid	0.5	0.333	0.4
	Non-Covid	0.364	0.5	0.421
4 Layers	Healthy	0.875	0.875	0.875
	Covid	0.643	1	0.783
	Non-Covid	1	0.375	0.545

According to our results, 4 layers performed much better especially when we take a look at the metrics such as Recall for the Covid label. While the 3 layers model only identified 33% of the Covid images, the 4 layers modeled managed to identify all of them. Thus, we decided to go ahead with a model with 4 convolutional group layers.

Why batch size of 16 chosen?

We tried increasing the batch size to 32 but was always met with “CUDA out of memory error”. The highest batch size we can go is thus 16. Batch size should be in powers of 2 because for best performance, we want to align these mini batches to the page boundary in the GPU which are usually a power of 2.

Choice of loss function and its parameters if any

We have chosen to go with the cross-entropy loss function because cross-entropy loss generally works quite well in a classification problem. This is because cross-entropy loss minimizes the distance between the predicted probability distribution and the actual probability distribution. This is consistent with what we have been taught in class and from researching multiple online sources for effective loss functions for classification problems.

Explain your choice of optimizer and its parameters, if any

We have chosen to go ahead with the Adam optimiser as it seems to be the most effective SGD optimiser that takes into account a true weighted average as well as normalising every dimension of the update separately. The main parameter for Adam that is interchangeable is the learning rate which we have set to 0.001.

Explain choice of initialization for model parameters

We tried 4 different weight initialization techniques, which are Kaiming Uniform, Kaiming Normal, Xavier Uniform, and Xavier Normal. From these different initializations, we noted the performance of the model in terms of accuracy and their confusion matrix.

Here are our findings:

	Health_Model				Covid_Model			
Metric	train acc	val acc	train loss	val loss	train acc	val acc	train loss	val loss
Kaiminguniform	85	79	0.3199	0.9689	84	79	0.7691	0.8578
KaimingNormal	85	72	0.2834	0.8127	89	80	0.2836	0.5532
XavierNormal	84	72	0.5628	0.5058	75	85	1.5768	0.4173
XavierUniform	90	83	1.3626	2.8635	96	87	0.4134	2.1356

		Label	Precision	Recall	F1
Kaiming	uniform	Healthy	0.875	0.875	0.875
		Covid	0.643	1	0.783
		Non-Covid	1	0.375	0.545
	normal	Healthy	1	0.875	0.933
		Covid	0.615	0.889	0.727
		Non-Covid	0.6	0.375	0.462
Xavier	uniform	Healthy	1	0.25	0.4
		Covid	0.421	0.889	0.571
		Non-Covid	0.5	0.25	0.333
	normal	Healthy	0.8	0.5	0.615
		Covid	0.444	0.889	0.592
		Non-Covid	0.5	0.125	0.2

As you can see, when our model is initialized with He Initialization (Kaiming Uniform), the accuracy and scores of the model are generally better compared to that of other initializations. We also paid extra attention to the **recall score** of COVID, because this is the metric that quantifies the number of correct COVID predictions made out of all COVID predictions that could have been made. This metric provides an indication of missed COVID predictions, which is very undesirable since the impact of diagnosing a COVID-infected person as healthy could be devastating.

Discussion about the evaluation of the metrics to determine which is the best indicator

Loss is the average binary cross-entropy loss between the ground truth label and the predicted label

Accuracy is the percentage of images that have been correctly classified

Recall is the ability to find all relevant instances in a dataset

Precision is the proportion of the data points that our model says was relevant and is indeed relevant.

F1-Score provides a balance between the recall and the precision.

In this situation, our group feels that the most important metric to judge how well our model does is Recall for the COVID label. This is because the consequences will be disastrous if we clear a patient of COVID when he or she actually has COVID.

The precision in this case is not as important because it is better to falsely identify someone for having COVID when they are not compared to falsely identifying someone for NOT

having COVID when they actually do. People who are falsely identified as having COVID can have more tests and they should be cleared after that.

Meanwhile, the more common metrics such as accuracy might not be good. This is especially because this is a binary classification and the likelihood of getting the correct class at random is already 50%, so it's not the best indicator of whether the model is good. Loss is also not the best metric because a model with low loss is useless if it does not act according to what is important in the use case. A model with low loss is just a general guide on how good the model is but it is not necessarily the final decision metric. In addition, accuracy and loss in our project are made on the individual models and thus might not depict completely the performance of the system as a whole.

F1 score is also a useful metric as it provides a balance between the recall and the precision and is often a very accurate metric to see if a model is performing well. However, in this case since there is a focus on the recall especially for Covid label, this will not be the most important metric.

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.

Logically speaking, we hypothesise that distinguishing between non-COVID and COVID X-rays would be harder as they are both images of infected lungs.

These infected lung images (both COVID and non-COVID) should have much more noticeable differences to the healthy lung images (e.g. infected lung images are generally more bloated, etc.). However, the difference between COVID-infected lung images and non-COVID lung images would not be as obvious. Thus, it would be more difficult to differentiate between COVID infected lung vs. non-COVID infected lung compared to differentiating between infected lung (Both COVID & non-COVID) vs. healthy lung.

As seen from the images in the PDF handout, we can see that the images for COVID and non-COVID X-rays are indeed quite similar whereas images for infected lungs (both COVID and non-COVID) are generally quite different from that of healthy lungs. This supports our hypothesis above.

However, as seen from our results (accuracy, recall scores, etc.), our model seems to be able to differentiate healthy lung from infected lung and differentiate COVID-infected lung from non-COVID-infected lung equally well. This might mean that the model was able to detect some subtle difference effectively that might not be so obvious to a human eye.

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

We feel that low false negative rates on the COVID class or low true negative for a healthy is especially important. This is because we would not want to possibly miss out on a COVID case and clear them as healthy when they are sick, likewise we do not want to classify them as healthy when they are actually not. This is more important than wrongly classifying them

as COVID when they do not have it (False Positive) as they can take more tests and can be cleared by those. These might be more important than the baseline accuracy because the model can have a very high accuracy but if it misses some people with COVID and clear them, the consequences could be very dire. Therefore, although a model with high overall accuracy is desirable, in this particular case, we feel that a model with lower false negative for COVID and lower false positive for healthy class would be more important.

Implemented Early Stopping

We have implemented early stopping to prevent overfitting. Our early stopping rule is as such:

"**The minimum number of epochs to be run is 3.** After 3 epochs, if the validation loss of an epoch is smaller than that of its next 3 consecutive epochs, we will stop the training and use the model achieved from the above-mentioned epoch."

For example, let's say we have trained a model with the following statistics:

Epoch 1 Validation set: Average loss: 3.6188, Accuracy: 482/615 (78%)
Epoch 2 Validation set: Average loss: 1.8149, Accuracy: 465/615 (76%)
Epoch 3 Validation set: Average loss: 0.9689, Accuracy: 488/615 (79%)
Epoch 4 Validation set: Average loss: 1.0441, Accuracy: 440/615 (72%)
Epoch 5 Validation set: Average loss: 3.4567, Accuracy: 462/615 (75%)
Epoch 6 Validation set: Average loss: 3.7952, Accuracy: 490/615 (80%)

As you can see, since each validation loss of epoch 4, 5, and 6 is larger than that of the validation loss in epoch 3, we can say that the validation loss has gone worse for 3 consecutive epochs after epoch 3. Therefore, we stop the training and save the model achieved after epoch 3 and ignore the model achieved after epoch 4, 5, and 6.

Another example, let's say we have trained a model with the following statistics:

Epoch 1 Validation set: Average loss: 4.4662, Accuracy: 285/381 (75%)
Epoch 2 Validation set: Average loss: 2.9210, Accuracy: 299/381 (78%)
Epoch 3 Validation set: Average loss: 3.3938, Accuracy: 248/381 (65%)
Epoch 4 Validation set: Average loss: 0.8578, Accuracy: 300/381 (79%)
Epoch 5 Validation set: Average loss: 1.7284, Accuracy: 250/381 (66%)
Epoch 6 Validation set: Average loss: 1.6840, Accuracy: 289/381 (76%)
Epoch 7 Validation set: Average loss: 0.7267, Accuracy: 307/381 (81%)
Epoch 8 Validation set: Average loss: 1.2123, Accuracy: 249/381 (65%)
Epoch 9 Validation set: Average loss: 0.9764, Accuracy: 299/381 (78%)
Epoch 10 Validation set: Average loss: 0.8253, Accuracy: 302/381 (79%)

We can see that after epoch 4, we have 2 consecutive epochs (epoch 5 & 6), where each validation loss is lower than the validation loss in epoch 4. However, in epoch 7, the validation loss is lower than that of epoch 4, breaking the consecutivity. Therefore, we continue the training and after 3 consecutive epochs (epochs 8, 9, 10) where each validation

loss is larger than that of epoch 7, we stop the training. Finally, we save the model achieved after epoch 7 and ignore the model achieved after epoch 8, 9, and 10.

References:

Koehrsen, W. (2018, March 10). Beyond accuracy: Precision and recall. Retrieved March 21, 2021, from https://towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c

https://www.quora.com/When-should-you-use-cross-entropy-loss-and-why

B, N. (2019, January 18). Image data pre-processing for neural networks. Retrieved March 21, 2021, from https://becominghuman.ai/image-data-pre-processing-for-neural-networks-498289068258

Pawara, P., Okafor, E., Groefsema, M., He, S., Schomaker, L., & Wiering, M. (2020, July 01). One-vs-One classification for deep neural networks. Retrieved March 21, 2021, from https://www.sciencedirect.com/science/article/pii/S0031320320303319