Name: Melissa, Sean Lew Teng Siong, Teng Jun Yuan
Student ID: 1003850, 1002751, 1003319

Deep Learning Mini Project
Theory Questions

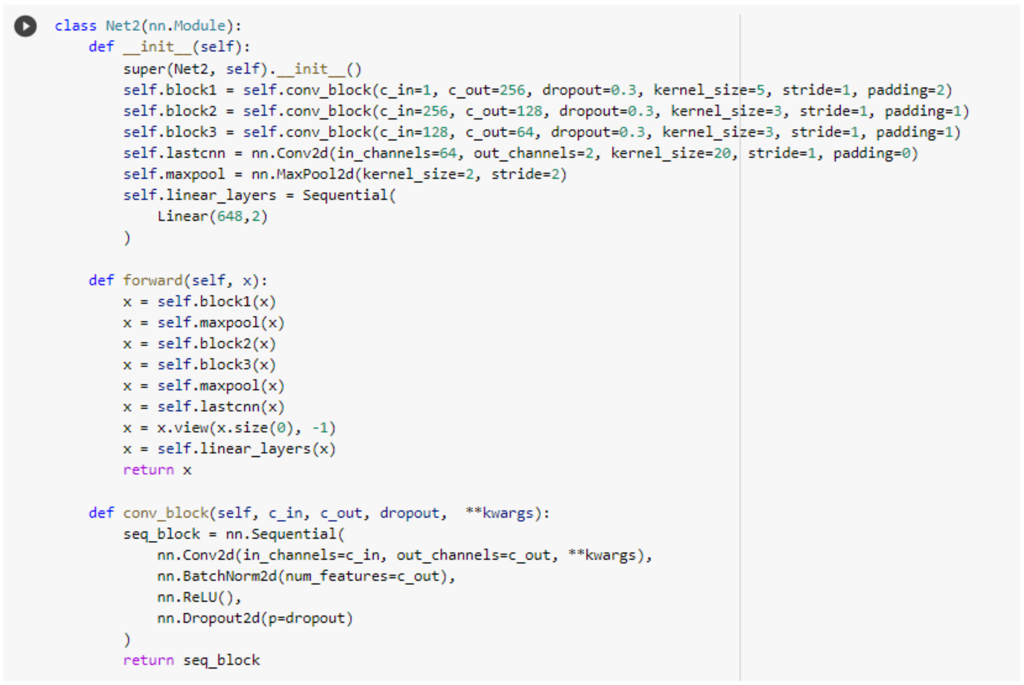
1.Discuss typical data processing operations to be applied to your images.Typically, you have seen in the Demo Notebook, that our images needed some normalization was one, but why did we need it anyway? Are there other pre-processing operations that are needed for our images?
In our custom Dataset class, we return our input image as a normalized array. In particular we converted the image to a numpy array and normalize the pixel values by dividing it by 255 before returning the image from the open_img() function. The reason why we do this is because when training a Deep Neural Network with images, the computation of high numeric values may become complex, as pixel values range from 0 to 256, apart from 0 the range is 255, so we divide it by that value and now our computation works with values ranging from 0 to 1, making it easier and thus faster. The normalizing of these values also prevents exploding gradients as the product of multiple large values ranging from 0 to 255 would lead to exponentially large values.

2.Discuss the differences between the two architectures and defend which architecture you decided to go for and why?
Our initial direction was to go with a multi-class classifier as it was easily implementable and seeing as we have 3 classes in total, the multi-class classifier would be able to identify each of the patterns of the Normal, Infected (non-covid) and Infected (covid) individually and tell them apart. However, after training, our multi-class classifier did not perform as well as we had hoped and had an accuracy of 58%.

After trying to understand the problem we had at hand better, we moved to the 2-layer Binary classifier approach, first to determine whether an image is normal or infected, and then determining whether it is covid or non-covid. We hypothesised that determining Normal vs. Infected cases would be easier as an infected person's chest x-ray would reveal areas of opacity that is white/densely colored, marking a clear distinction between normal vs infected cases. We decided that the binary classification would be better for this problem as it is more crucial in determining a healthy chest x-ray vs. an infected chest x-ray. The second step in determining covid vs. non-covid cases is actually not fully diagnosed from chest x-rays images, but from further diagnostics/tests the doctor would carry out, thus the second classification test is not well represented in our dataset consisting of only chest x-rays.

Thus, given the context of our problem, the binary classifier should be chosen as the model's ability to determine Normal vs. Infected is a better represented classification task compared to Covid vs. Non-Covid, and thus would be a better classifier compared to a multi-class classifier. It would also be more important to determine a healthy vs. infected person, as the infected person would have to undergo further testing to determine whether it is covid.

3.Discuss the reasons behind our architecture (our model)



```
class Net2(nn.Module):
    def __init__(self):
        super(Net2, self).__init__()
        self.block1 = self.conv_block(c_in=1, c_out=256, dropout=0.3, kernel_size=5, stride=1, padding=2)
        self.block2 = self.conv_block(c_in=256, c_out=128, dropout=0.3, kernel_size=3, stride=1, padding=1)
        self.block3 = self.conv_block(c_in=128, c_out=64, dropout=0.3, kernel_size=3, stride=1, padding=1)
        self.lastcnn = nn.Conv2d(in_channels=64, out_channels=2, kernel_size=20, stride=1, padding=0)
        self.maxpool = nn.MaxPool2d(kernel_size=2, stride=2)
        self.linear_layers = Sequential(
            Linear(648,2)
        )

    def forward(self, x):
        x = self.block1(x)
        x = self.maxpool(x)
        x = self.block2(x)
        x = self.block3(x)
        x = self.maxpool(x)
        x = self.lastcnn(x)
        x = x.view(x.size(0), -1)
        x = self.linear_layers(x)
        return x

    def conv_block(self, c_in, c_out, dropout,  **kwargs):
        seq_block = nn.Sequential(
            nn.Conv2d(in_channels=c_in, out_channels=c_out, **kwargs),
            nn.BatchNorm2d(num_features=c_out),
            nn.ReLU(),
            nn.Dropout2d(p=dropout)
        )
        return seq_block
```

4.Multiclass Model and Binary Model A (Infected vs Non-infected)
We built 4 blocks of convolutional layers, each block consisting of a **Convolution** + **BatchNorm** + **ReLU** + **Dropout** layers. At the end of the convolution we will use a Fully Connected layer (output=2 for Binary and output =3 for multiclass) . Even though the final FC can be replaced by a Convolutional layer, the benefits of using a FC is that we provide our model the ability to mix signals, since every neuron has a connection to every single one in the next layer, now there is flow of information between each input dimension to the final output class.

Some of the considerations when designing the parameters is to ensure that the network is wide enough in width to be able to capture more features of the x-ray images. That being said it was important to also have sufficient dropout between each layer to help reduce overfitting. The model did not had to be too deep since we observed overfitting with multiple previous models that were much deeper

```
class Net3(nn.Module):
    def __init__(self):
        super(Net3, self).__init__()
        self.block1 = self.conv_block(c_in=1, c_out=256, dropout=0.1, kernel_size=5, stride=1, padding=2)
        self.block2 = self.conv_block(c_in=256, c_out=128, dropout=0.1, kernel_size=3, stride=1, padding=1)
        self.block3 = self.conv_block(c_in=128, c_out=64, dropout=0.1, kernel_size=3, stride=1, padding=1)
        self.lastcnn = nn.Conv2d(in_channels=64, out_channels=2, kernel_size=20, stride=1, padding=0)
        self.maxpool = nn.MaxPool2d(kernel_size=2, stride=2)
        self.linear_layers = Sequential(
            Linear(6272,2)
        )

    def forward(self, x):
        x = self.block1(x)
        # x = self.maxpool(x) removed one pooling layer for model to capture more details to differentiate between covid and non-covid
        x = self.block2(x)
        x = self.block3(x)
        x = self.maxpool(x)
        x = self.lastcnn(x)
        x = x.view(x.size(0), -1)
        x = self.linear_layers(x)
        return x

    def conv_block(self, c_in, c_out, dropout,  **kwargs):
        seq_block = nn.Sequential(
            nn.Conv2d(in_channels=c_in, out_channels=c_out, **kwargs),
            nn.BatchNorm2d(num_features=c_out),
            nn.ReLU(),
            nn.Dropout2d(p=dropout)
        )
        return seq_block
```

5.Binary model B

Given our hypothesis that distinguishing between Infected (covid) and Infected (non-covid) x-ray images will be a much more daunting task for a neural network. We tried to re-use as much of the network that we can from model A but what we alter here is the dropouts between each layer and also removing the final maxpooling layer. Lowering/removing dropouts will allow our network to capture even more specific details that can help us distinguish between the 2 classes. But at the same time this can cause the model to overfit.

6.Explain the choice of a loss function and its parameters, if any.
Since the output of our model was a probability of the image being classified in each of the binary classes. It made sense for us to work with CrossEntropyLoss as our loss function as it measures the performance of classification models whose output is between 0 and 1. The loss increases as the predicted probability diverges from the actual label and penalizes the model respectively.

7.Explain your choice of an optimizer and its parameters, if any (e.g. learning rate, momentum, etc.)
We initially initialized our optimizer to be Adam but our models were exhibiting a large extent of overfitting behaviour. We did some research on comparing between SGD and ADAM, this particular research paper submitted by Pan Zhou titled “Towards Theoretically Understanding Why SGD Generalizes Better Than ADAM in Deep Learning” (https://arxiv.org/abs/2010.05627) gave us a little more clarity and confidence in utilizing SGD as an optimizer for better generalization. To summarize our learnings, SGD is more locally unstable than ADAM at sharp minima and can better escape from them to flatter ones, allowing it to have better generalization performance.

We also set a really small learning_rate of **0.00005** which might be unusually small, but the reason for this is also after observing the relatively good performance on the first epochs of our models, we decided to reduce the learning rate and maintain a high epoch of about ~30. A few of our previous models had fluctuating validation accuracy and we wanted to make sure that a high learning rate was not a cause of it.

For the multi-class and binary model B we adjusted the epochs to 50 and learning rate to 0.0001 because we observed that at the end of 30 epochs the accuracy and losses were still not slowing down. One of the reasons can be due to the much more complex task at hand compared to model A, they require more time to pick up the features for distinction.

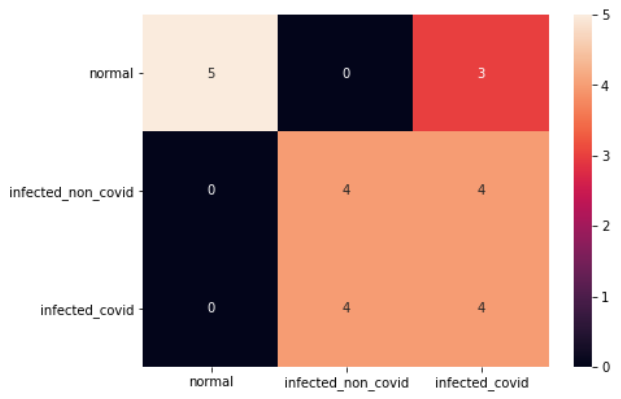
8.Explain your choice of initialization for your model parameters
For our model, only 2 particular layers contain initialization of model parameters (bias and weight):
nn.Conv2d
nn.Linear

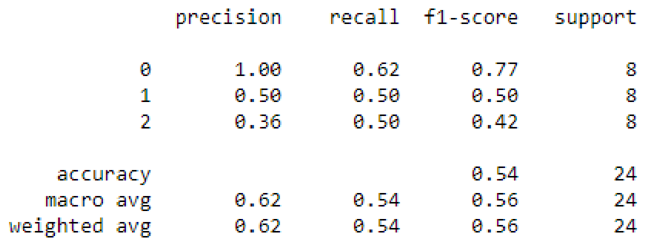
For both these layers, we have used Pytorch’s default weights and biases. For biases, the default initialization is to 0, and for weights, both are initialized to use the kaiming_uniform distribution, also known as the ‘He Initialization’.

As our convolution blocks are made up of ReLU layers, this initialization method would be preferred, as compared to the other frequently used Xavier initialization which is more suited for tanh functions, which is why we used the default initializations Pytorch has in their library.

9.Some of the other metrics that we used to understand our prediction

Some of the additional metrics that we implemented to help us better understand how our final models were predicting between the 3 classes were a heatmap visualization of a 3x3 confusion matrix as well as the built-in classification report.

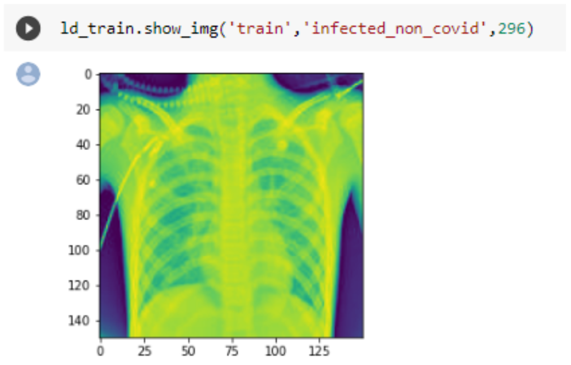




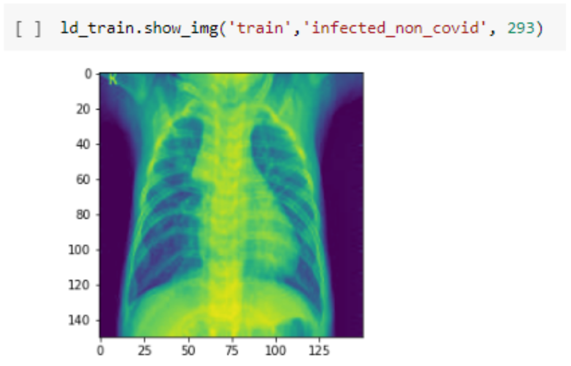
```
              precision    recall  f1-score   support

           0       1.00      0.62      0.77         8
           1       0.50      0.50      0.50         8
           2       0.36      0.50      0.42         8

    accuracy                           0.54        24
   macro avg       0.62      0.54      0.56        24
weighted avg       0.62      0.54      0.56        24
```

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

This is to be expected. From the x ray it's easy to tell if the patient is infected or not because an infected person's chest x ray will reveal areas of opacity on the x ray seen as white/densely colored. There is thus a more distinct difference between normal and infected x-rays.



```
ld_train.show_img('train','infected_non_covid',296)
```



```
[ ] ld_train.show_img('train','infected_non_covid', 293)
```

However, determining whether the infected person has covid is harder to tell because of the less visible differences in the xrays. In the x-rays of patients diagnosed with covid 19, lung markings are partially obscured by the increased whiteness, a ground glass pattern. This can be a very subtle difference. Hence our model is more likely to learn the bigger difference and might not be as good in identifying minute differences. This is also where we feel data augmentation can be crucial in helping us finetune our inputs.

One of the more effective and popular methods is to improve the clarity of grayscale images is Contrast Limited Adaptive Histogram Equalization (CLAHE), this is done by stretching the histogram of the image pixel values to either ends. Unfortunately, our group did not have the time to implement this. An example below show how difference in the contrast of the x-ray images can hinder the model's ability to recognize features clearly

10.Would it be better to have a model with high overall accuracy or low true negatives/false positives rates on certain classes? Discuss.
Low true negatives/false positive rate identifying covid-19 patients. When applying AI to medical situations, accuracy is not a sufficient metric to determine how well your model is. Diagnosing them as non-covid when they are in fact infected with covid has a more severe consequence than the reverse scenario. At the same time we also do not want to misdiagnose a normal person as being infected with covid. Hence we need to find an optimal threshold value for this trade off.

This is especially so in datasets with a lesser number of infected classes. Our model might predict all labels as normal simply because there are more normal cases than infected. We will achieve a high accuracy rate but we are not truly achieving the goal of our model - to predict whether the patient is infected.

Therefore we must include true negatives in our statistical analysis such as specificity or using an Roc Curve to determine the best threshold value

Final Thoughts:
After many iterations and changes of our models for both multi-class and cascading binary models. Our multi-class model ended up performing slightly better than the 2 models even though our single multi-class model achieved an average validation accuracy of **69-71%**, for model A **79-81%** and for model B **85-88%**. One of the main hypothesis we have is that we did not implement early stopping during our model training. That is to say that currently our “trained” model is the LAST epoch’s most updated weights, this mean that the model we use for evaluation might not have been the **BEST** performing model out of the different epochs. Unfortunately we did not have the time to implement early stopping and it will be the priority task for the project after submission.