Gil Lotzky
CS 440
Class Challenge

CS 440 Class Challenge: Image Classification of COVID-19 X-rays

Architecture of Models:

Task 1: Classify normal vs. COVID-19 Xrays using the data

VGG16
Optimizer: Adam, Learning Rate = .0005
Loss Function: Binary Cross Entropy
Parameters: 21,137,729 (Trainable params: 6,423,041, Non-trainable params: 14,714,688)
Regularization: None

Architecture:
For Task 1, I used a VGG16 based model. The VGG16 model uses pre-trained weights on Imagenet. The model contains 16 total layers. VGG16 is a convolutional neural network model. It replaces large kernel-sized filters with multiple 3x3 kernel-sized filters. VGG16 contains three fully-connected layers, which follow a stack of convolutional layers. The last layer is the softmax layer.

```
vggmodel.summary()
```

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 0
Non-trainable params: 14,714,688
_________________________________________________________________
```

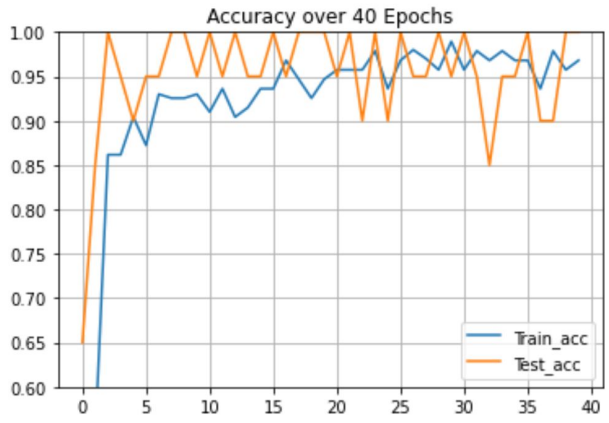
The architecture of my model for task 1 consists of the VGG16 Model layer, a flatten layer, and two dense layers.

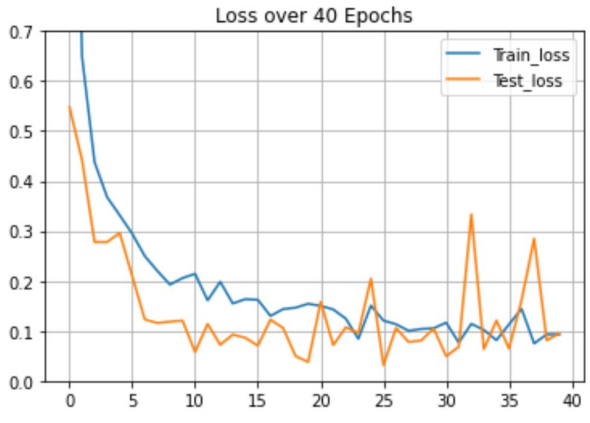
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 7, 7, 512)         14714688
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
dense_feature (Dense)        (None, 256)               6422784
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 257
=================================================================
Total params: 21,137,729
Trainable params: 6,423,041
Non-trainable params: 14,714,688
```

There were 6,423,041 trainable parameters, 14,714,688 non trainable parameters for a total of 21,137,729 parameters. The first dense layer was used as the feature layer, with the relu activation function. This fully connected layer had 256 nodes. It was found that dropout layers had not improved the accuracy of the model, so they were not included in the architecture. The final dense layer, the fully connected layer was incorporated using sigmoid activation function. This had 1 node for simplification since we were only comparing two classifications of images.

After training and testing, this architecture had proven to perform the best using the VGG16 Model. A learning rate of .0005, a batch size of 10, a batch size of 10, and 40 epochs were used to fit the model. Additionally, I had used Adam for the optimization of the model. The Binary Cross Entropy loss function was also used to conduct the training and testing of the model.

As a result the test accuracy was around .95 and 1, and the training accuracy was between .9 and .95.





Task 2: Classify normal vs. COVID-19 X Rays using the data
For task 2, I had attempted to use three different models. All models were sequential based architectures.

VGG16
Optimizer: Adam, Learning Rate = .0001
Loss Function: Binary Cross Entropy
Parameters: 21,138,500 (Trainable params: 6,423,812, Non-trainable params: 14,714,688)
Regularization: None

Architecture:
The first model used was the same VGG16 based model from task 1. The differences in task 2 was that the last fully connected layer had 4 nodes due to its multi classification model. The same VGG16 trained on imagenet was used with 16 layers and the last softmax layer.

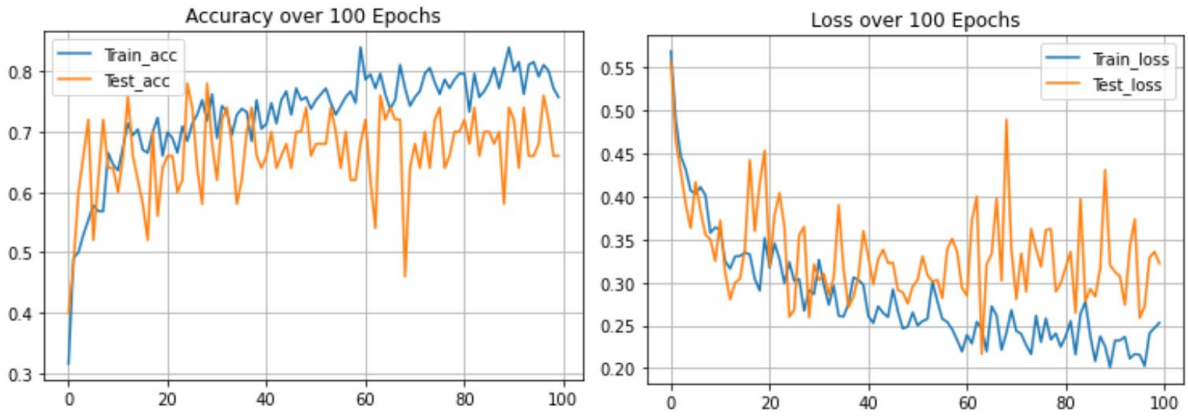
The architecture of the VGG16 model for task 2 is as follows:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 7, 7, 512)         14714688
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
feature_dense (Dense)        (None, 256)               6422784
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 1028
=================================================================
Total params: 21,138,500
Trainable params: 6,423,812
Non-trainable params: 14,714,688
```

There were 6,423,812 trainable parameters, 14,714,688 non trainable parameters for a total of 21,138,500 parameters. The first dense layer was used as the feature layer, with the relu activation function. This fully connected layer had 256 nodes. It was found that dropout layers had not improved the accuracy of the model, so they were not included in the architecture. The final dense layer, the fully connected layer was incorporated using softmax activation function. This had 4 nodes since we needed to compare four classifications of images.

After training and testing, this architecture had proven to perform the best using the VGG16 Model. A learning rate of .0001, a batch size of 10, a batch size of 10, and 100 epochs were used to fit the model. Additionally, I had used Adam for the optimization of the model. The Binary Cross Entropy loss function was also used to conduct the training and testing of the model.

As a result the test accuracy was .69, and the training accuracy was between .7 and .8.



AlexNet
Optimizer: Adam
Loss Function: Categorical Cross Entropy
Parameters: 11,325,964 (Trainable params: 11,325,964, Non-trainable params: 0)
Regularization: Three dropout layers applied(Dropout(.4), Dropout(.4), Dropout(.4))

Architecture:
The second model used to classify these images was an AlexNet based model.

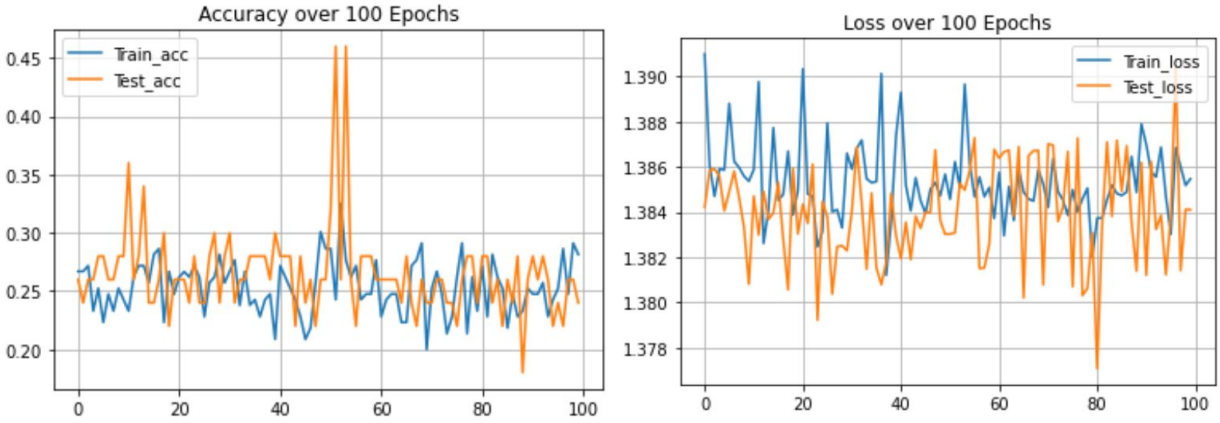
```
alexnet.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 54, 54, 96)        34944
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 27, 27, 96)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 17, 17, 256)       2973952
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 256)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 6, 6, 384)         885120
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 4, 4, 384)         1327488
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 2, 2, 256)         884992
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 1, 1, 256)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 256)               0
_________________________________________________________________
feature_dense (Dense)        (None, 256)               65792
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense (Dense)                (None, 4096)              1052672
_________________________________________________________________
dropout_1 (Dropout)          (None, 4096)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1000)              4097000
_________________________________________________________________
dropout_2 (Dropout)          (None, 1000)              0
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 4004
=================================================================
Total params: 11,325,964
Trainable params: 11,325,964
Non-trainable params: 0
```

AlexNet architecture consists of eight layers and an output layer. There were 11,325,964 trainable parameters. The first layer passes an image through a convolutional layer with 96 filters with 11x11 size and 4 strides, and relu activation function. This will change the images dimensions. It then passes it through a maximum pooling layer to get its dimensions to 27x27x96. The second layer of this model consists of a second convolutional layer with 256 features of size 5x5 and a stride of 1. Then it's passed through a maximum pooling layer to reduce the image to 17x17x256. The third, fourth and fifth layers pass the image through more convolutional layers with relu activation function, and then a maximum pooling layer with stride 2. The sixth layer of the model then flattens the output through a fully connected layer. Following this, dropout layers are applied with a value of .4 to help prevent overfitting. Two more fully connected layers are applied with relu activation with the 4096 units. The output layer is then applied with a softmax function and 4 nodes for the 4 variations of the classification.

After training and testing, this architecture had proven to perform the worst using the AlexNet Model. A learning rate of .0001, a batch size of 10, a batch size of 10, and 100 epochs were used to fit the model. Additionally, I had used Adam for the optimization of the model. The Binary Cross Entropy loss function was also used to conduct the training and testing of the model.

As a result the test accuracy was .25, and the training accuracy was around .25 which implies that its accuracy can be attributed to nearly random guessing.



ResNet50
Optimizer: Adam
Loss Function: Categorical Cross Entropy
Parameters:49,279,108 (Trainable params: 49,225,988, Non-trainable params: 53,120)
Regularization: One dropout layer applied (Dropout(.25))

Architecture:
The third model used to classify these images was a ResNet based model. ResNet50 is a variation of the ResNet model with 48 convolutional layers, 1 Max-pooling layer, and 1 average pooling layer.

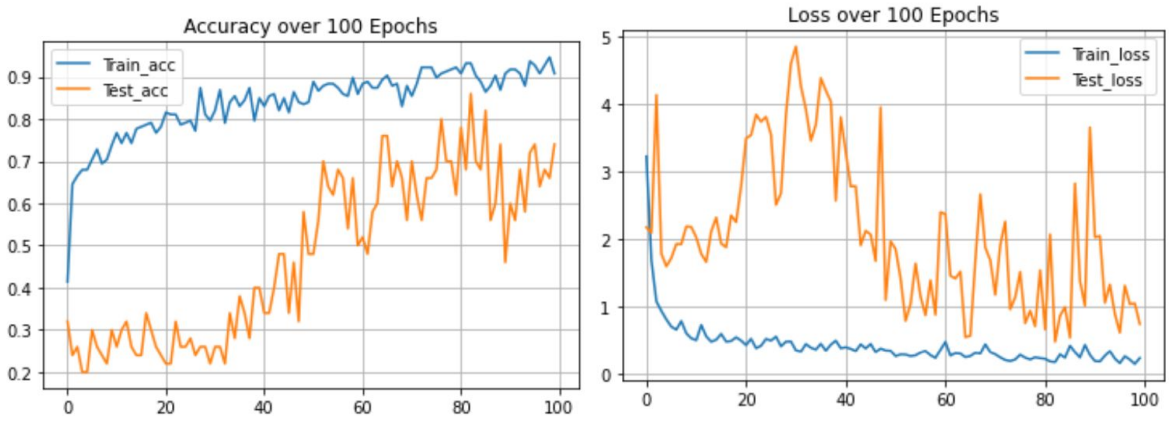
```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Functional)        (None, 7, 7, 2048)        23587712
_________________________________________________________________
flatten_2 (Flatten)          (None, 100352)            0
_________________________________________________________________
feature_dense (Dense)        (None, 256)               25690368
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 4)                 1028
=================================================================
Total params: 49,279,108
Trainable params: 49,225,988
Non-trainable params: 53,120
```

There were 49,225,988 trainable parameters, 53,120 non trainable parameters for a total of 49,279,108 parameters. The first layer consists of the ResNet feature extractor, the second layer is a flatten layer with 100352 nodes. The first dense layer contains 256 nodes with relu activation, a dropout layer was applied with a value of .25 to prevent overfitting. The output

layer was a dense layer with 4 nodes signifying the 4 different variations of the classification. A softmax activation function was applied on the last layer.

After training and testing, this architecture had proven to perform second best using the ResNet50 Model. A learning rate of .0001, a batch size of 10, a batch size of 10, and 100 epochs were used to fit the model. Additionally, I had used Adam for the optimization of the model. The Binary Cross Entropy loss function was also used to conduct the training and testing of the model.

As a result the test accuracy was .58, and the training accuracy was between .8 and .9. This implies that the model was overfitting the data.



Comparison of architectures for the second task:

In Task 2, I used 3 models; VGG16, AlexNet, Resnet50. The VGG16 model had the overall best performance. The AlexNet did not appear to have great results for the test or the training sets. The Resnet50 has a lot of potential if more regularization techniques were applied to the model.

The VGG16 model proved to have the best performance during training and testing without requiring any regularization techniques. It is possible, however, that introducing regularization techniques could have improved the overall performance of the model. The VGG16 model had the best overall combination compared to the other models. It had the second best time complexity overall, roughly one-fifth the amount of parameters trained and validated compared to ResNet, and achieved an accuracy of .69.

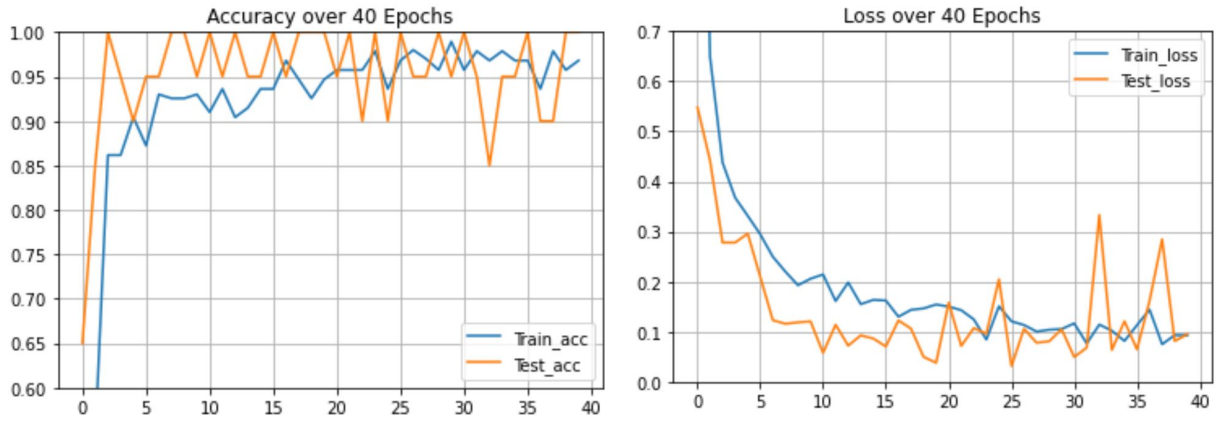
The ResNet50 model proved to have the second best performance during training and testing. It had implemented Dropout layers as a regularization technique, but overfitting was present during its execution. It is probable that other regularization techniques should have been used in the model during training to prevent overfitting. It had the slowest time complexity compared to the other models, largest numbers of parameters, and achieved an accuracy of .58.

The AlexNet model proved to have the worst performance overall during training and testing. It had implemented Dropout layers as a regularization technique, but its accuracy was comparatively the same as random guessing. The AlexNet model did have the lowest time for execution and used the least parameters, but achieved the lowest accuracy of .25.

Additional modifications to each model could improve the accuracies, however, based on a combination of time-complexity, number of parameters, and overall performance, the VGG16 model had proved the best. The VGG16 model could be improved if regularized properly, however, due to computation time, it is very hard to accomplish better results given time constraints.

Accuracy of Tasks:

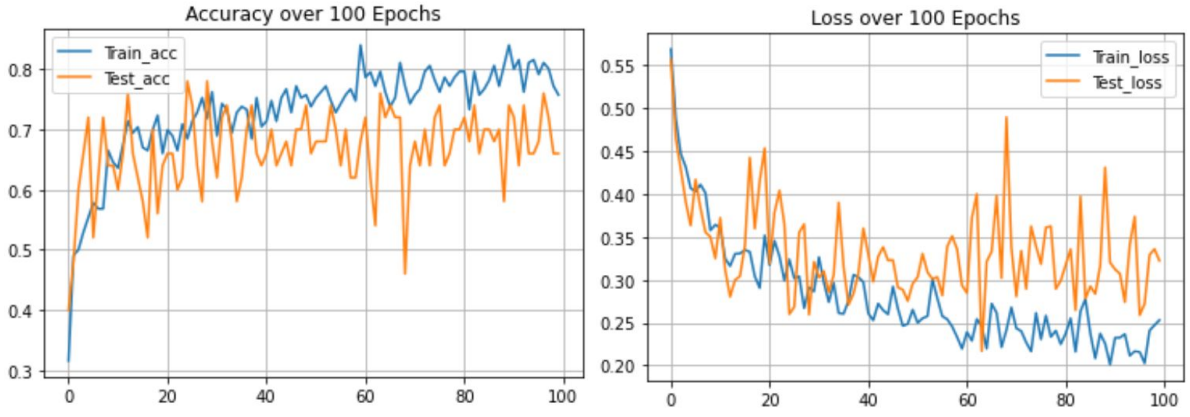
Task 1:
VGG16



The VGG16 model after 40 epochs had a training accuracy of .97 and a testing accuracy of .94. The training loss was .15 and the testing loss was about .19.

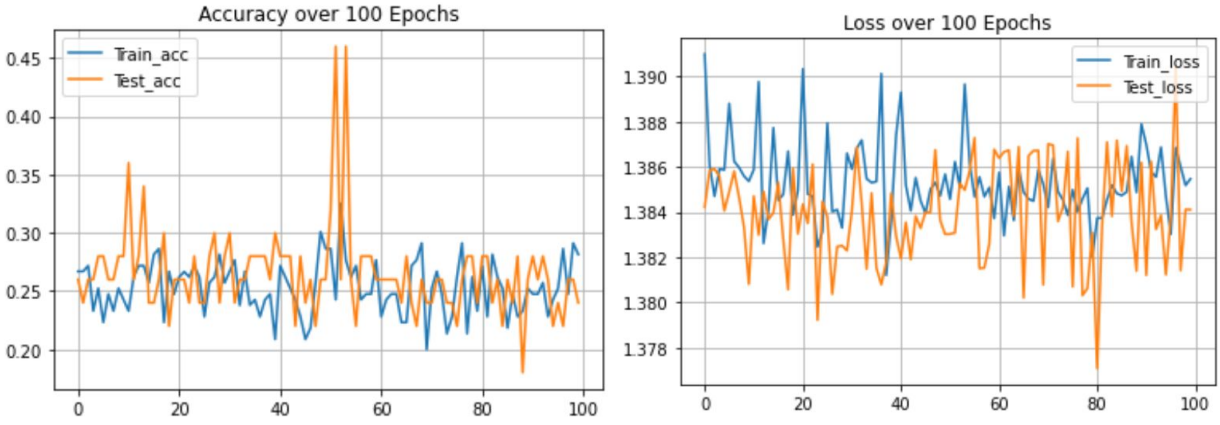
Task 2:

VGG16



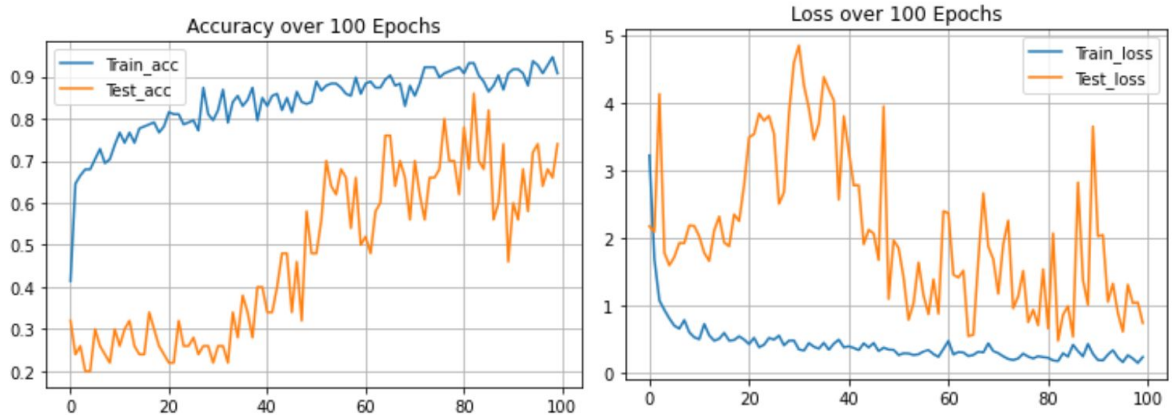
The VGG16 model after 100 epochs had a training accuracy of .79 and a test accuracy of .69. The training loss was .28 and the test loss was about .37.

Alexnet



The Alexnet model after 100 epochs had a training accuracy of .25 and a test accuracy of .25. The training loss was 1.388 and the test loss was about 1.388.

Resnet50

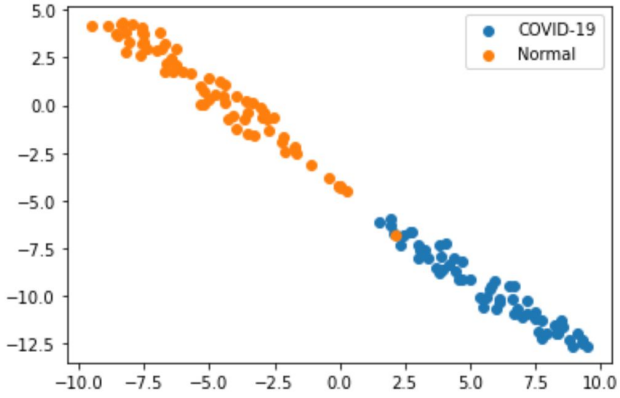


The Alexnet model after 100 epochs had a training accuracy of .88 and a test accuracy of .58. The training loss was .38 and the test loss was about 1.882.

t-SNE visualizations

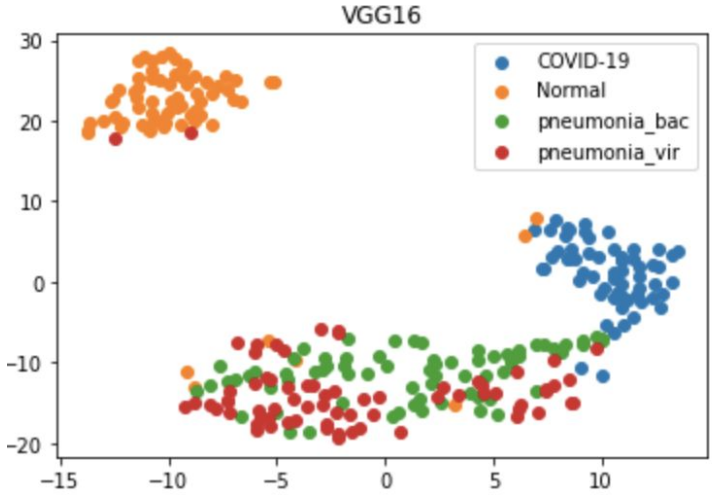
t-SNE(t-Distributed Stochastic Neighbor Embedding) is a technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets.

Task 1:
VGG16



The first fully connected dense layer was extracted for the t-SNE plot. Evaluating the VGG16 model visually from the t-SNE graph, we can conclude that the model performed well. There is clustering present of the COVID-19 data points and the Normal data points, overall. This concludes that the model had a rather good performance on the testing set.

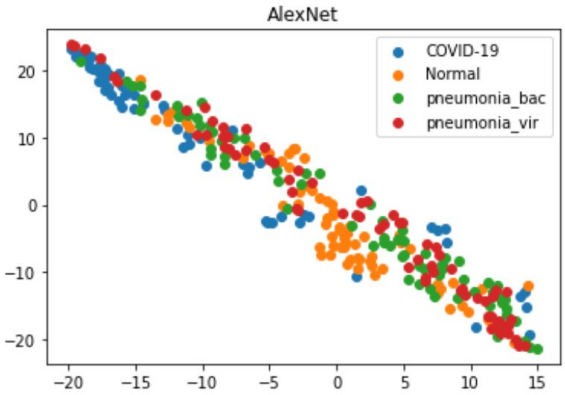
Task 2
VGG16



The first fully connected dense layer was extracted from the VGG16 model for the t-SNE plot. Evaluating the VGG16 model visually from the t-SNE graph, we can conclude that the model performed well for clustering COVID and Normal, but did not perform well for pneumonia identification. There are mixed data points for the two pneumonia groups in one cluster. This

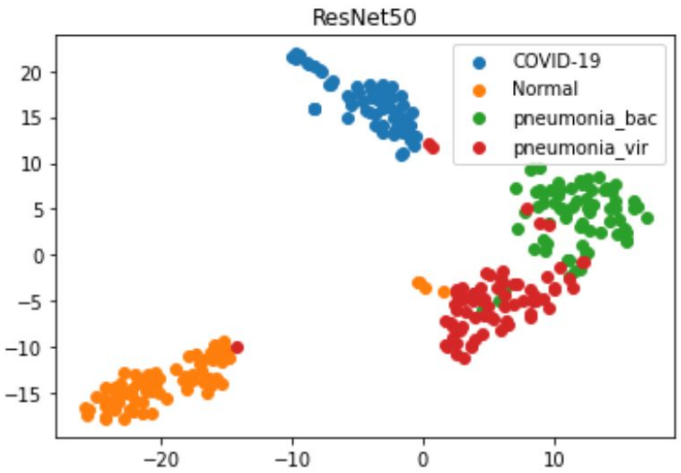
concludes that the model had a decent performance on the testing set for COVID and Normal, but struggled identifying pneumonia bac and pneumonia vir.

AlexNet



The first fully connected dense layer was extracted from the AlexNet model for the t-SNE plot. Evaluating the AlexNet model visually from the t-SNE graph, we can conclude that the model did not perform well for clustering any of the four groups. There are mixed data points for all points. This model was not able to identify the differences in the features between these four different classes. This concludes that the model had a terrible performance on the testing set.

Resnet50



The first fully connected dense layer was extracted from the Resnet50 model for the t-SNE plot. Evaluating the Resnet50 model visually from the t-SNE graph, we can conclude that the model was able to accurately clustering the four groups separately. This model was able to identify the differences in the features between these four different classes. This concludes that the model had a great performance on the testing set, which implied confusion for the poor performance. Many factors need to be reevaluated on the Resnet50 model to improve its performance.