Chest X-Ray Scan Classification

I. Task 1 : Binary Classification

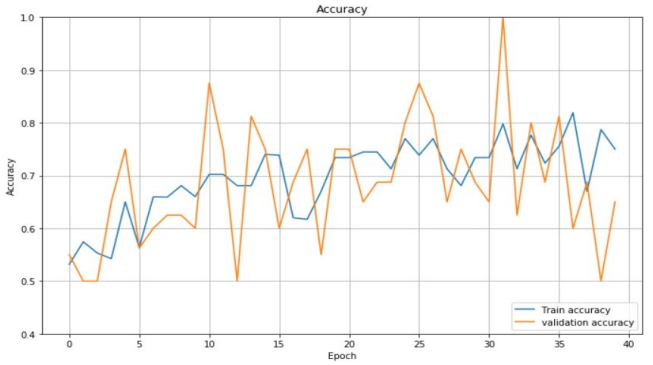
For this task, we decided to run three pretrained models along with our own model. we ran all four models on the training data with default hyperparameters to get baseline. Then based on these baselines decided to go ahead with VGG16 as it gave the best baseline results among all 4 models.

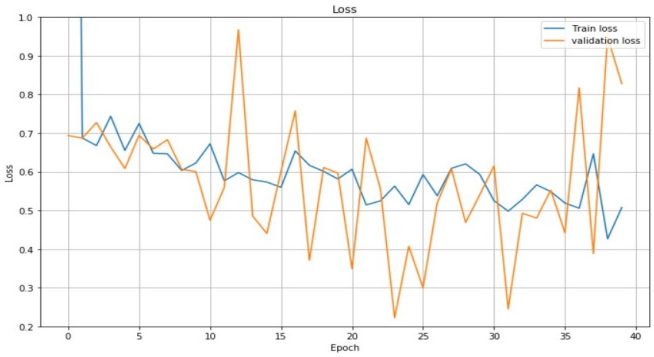
A. *Self Created Model*

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 222, 222, 64)      1792
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 111, 111, 64)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 109, 109, 64)      36928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 54, 54, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 186624)            0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               23888000
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 65
=================================================================
Total params: 23,935,041
Trainable params: 23,935,041
Non-trainable params: 0
_________________________________________________________________
```

1) Architecture:

- we used 2 convolution layers followed by pooling layers.
- For both the convolution layers, we used 64 filters with a kernel size of (3x3), stride of 1 and no padding.
- In the 1st dense layer, 256 nodes with the relu activation function followed by a dropout layer with a dropout probability of 0.25 is used.
- The final dense layer has 1 node with a sigmoid activation as we are dealing with a binary classification problem.
- The model is fitted using the Adam optimizer with the default learning rate of 0.005 for 40 epochs. The loss function used is binary crossentropy.





2) Results: On analysis we realized that this model did poorly because it have a large number of parameters(23,935,041) and only around 130 images to train. we tried a much shallow network as well but that did not improve the results.

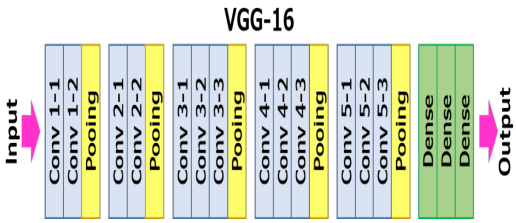
B. *VGG16 base feature extractor*

The second model that we used as a baseline is the VGG16 based model. we decided to use the pretrained weights trained on Imagenet and added our own fully connected layers at the top.

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 7, 7, 512)         14714688
_________________________________________________________________
flatten_2 (Flatten)          (None, 25088)             0
_________________________________________________________________
dense_3 (Dense)              (None, 256)               6422784
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 257
=================================================================
Total params: 21,137,729
Trainable params: 6,423,041
Non-trainable params: 14,714,688
_________________________________________________________________
```

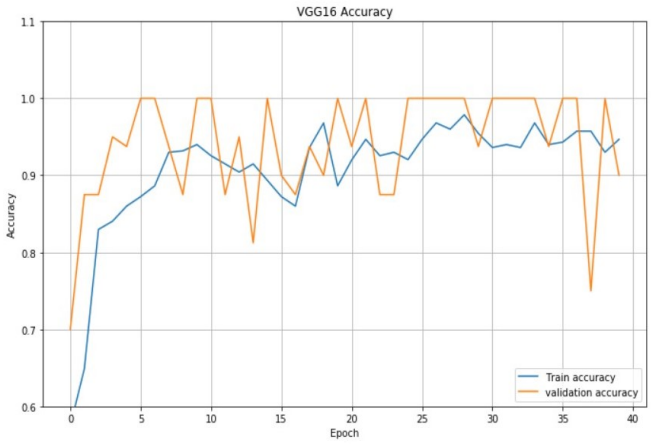
1) Architecture:

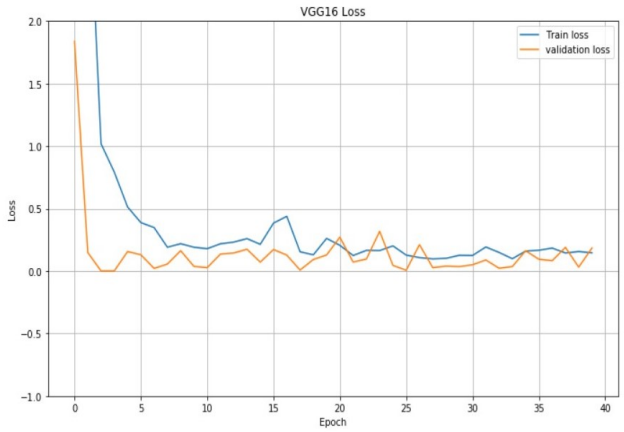
- VGG16 is a convolutional neural network that contains 16 layers and is pre-trained on Imagenet is used as base feature extractor.



- It has a total of 21,137,729 parameters (Trainable: 6,423,041 and Non Trainable: 14,714,688).
- In the first fully connected dense layer, we used 256 nodes with the relu activation function followed by a dropout layer with a dropout probability of 0.25.
- The final dense layer has 1 node with a sigmoid activation as we are dealing with a binary classification problem.
- The model is fitted using the Adam optimizer with the default learning rate of 0.005 for 40 epochs. we used binary crossentropy as a loss function.

2) Results: At the end of 40 epochs, VGG16 based model had a training accuracy of around 96% and validation accuracy of around 90%.





C. *VGG19 base feature extractor*

The third model that we used as a baseline was the VGG19 based model. Again, we decided to use the pretrained weights from imagenet and added our own fully connected layers at the top.

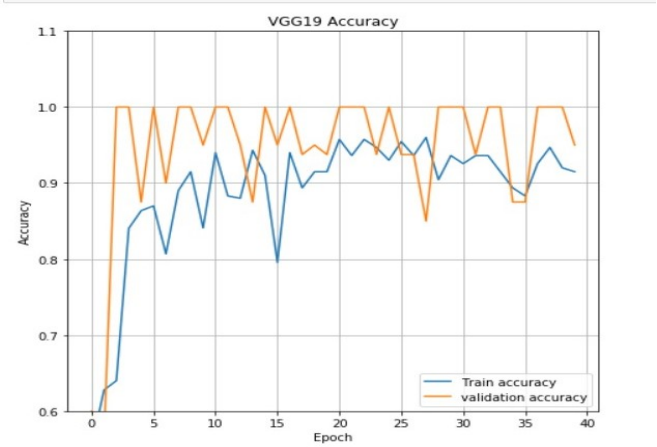
```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg19 (Model)                (None, 7, 7, 512)         20024384
_________________________________________________________________
flatten_3 (Flatten)          (None, 25088)             0
_________________________________________________________________
dense_5 (Dense)              (None, 256)               6422784
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 257
=================================================================
Total params: 26,447,425
Trainable params: 6,423,041
Non-trainable params: 20,024,384
_________________________________________________________________
```

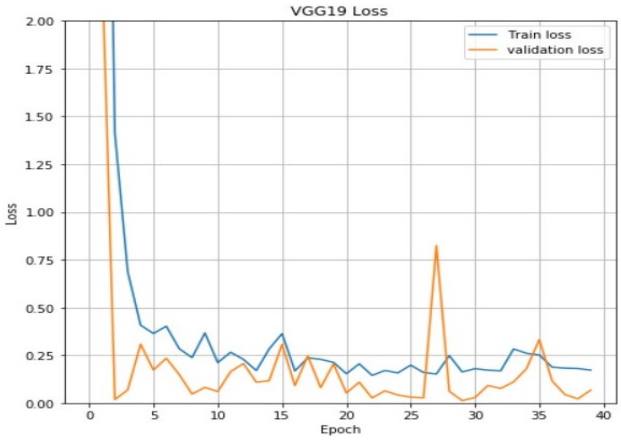
1) Architecture:

- The VGG19 is a convolution neural network with 19 layers having 3 more convolution layers than VGG16.

- It has a total of 26,447,425 parameters (Trainable: 6,423,041 and Non Trainable: 20,024,384)
- In the first fully connected dense layer, 256 nodes with the relu activation function followed by a dropout layer with a dropout probability of 0.25 is used.
- The final dense layer has 1 node with a sigmoid activation as we are dealing with a binary classification problem.
- The model is fitted using the Adam optimizer with the default learning rate of 0.005 for 40 epochs. The loss function used is binary crossentropy.

2) Results: The VGG19 model also did fairly well with a training accuracy of around 91% and valiation accuracy of around 95%.





*D. **ResNet50***

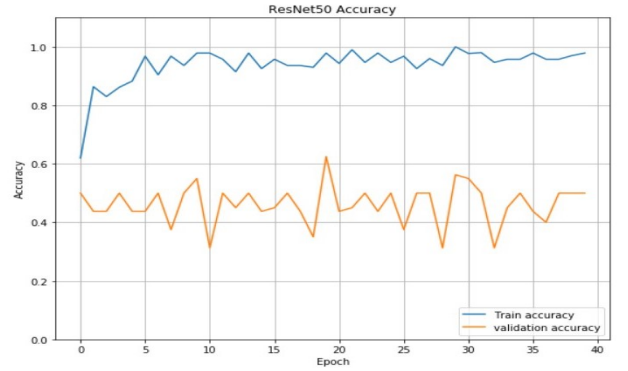
The last pretrained mdoel we used in this task is the ResNet50. As earlier, we used pretrained weights on the imagenet dataset and added our own dense layers at the top.

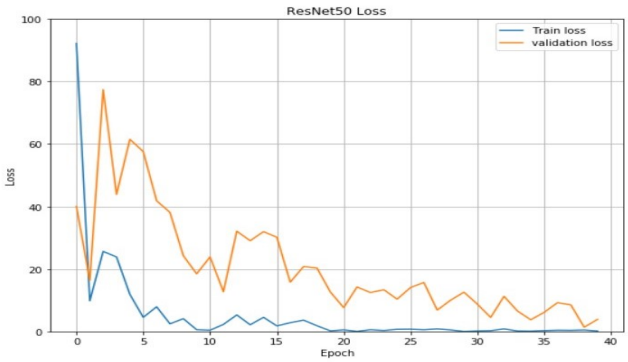
```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Model)             (None, 7, 7, 2048)        23587712
_________________________________________________________________
flatten_4 (Flatten)          (None, 100352)            0
_________________________________________________________________
dense_7 (Dense)              (None, 256)               25690368
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_8 (Dense)              (None, 1)                 257
=================================================================
Total params: 49,278,337
Trainable params: 25,690,625
Non-trainable params: 23,587,712
_________________________________________________________________
```

1) Architecture:

- ResNet50 is a variant of ResNet model which has 48 Convolution layers along with 1 Max-Pool and 1 Average Pool layer.
- It uses the concept of Residual blocks which uses activations from previous layers using skip connections in an attempt to avoid vanishing gradients and keep lowe r layers of the network relevant.
- In the 1st dense layer that we added on top of the ResNet5feature extractor, 256 nodes with the relu activation function followed by a dropout layer with a dropout probability of 0.25 is used.
- The final dense layer has 1 node with a sigmoid activation as we are dealing with a binary classification problem.
- The model is fitted using the Adam optimizer with the default learning rate of 0.005 for 40 epochs. The loss function used is binary crossentropy.

2) Results: The ResNet50 model has very good training accuracy at around 98% but very poor generalization at 50%. Overall the model had low bias and high variance.





E. *Tuned VGG16*

In this section, we decided to go ahead with the VGG16 model as it had encouraging baselines. It is also a smaller model than VGG19 and hence we decided to further tune the VGG16 based model. For parameter tuning, we used sklearns GridsearchCV function. Hyper parameters which are tuned are Learning rate, Dropouts and Activation function and epochs. we ran the model using 3-folds for accurate and better results. Below are the values used for hyper parameters tuning:

- learning_rate: [0.001,0.005,0.01,0.05,0.1]
- epochs: [40,80,120]
- activation:['sigmoid','relu']
- dropout: [0.15,0.20,0.25]

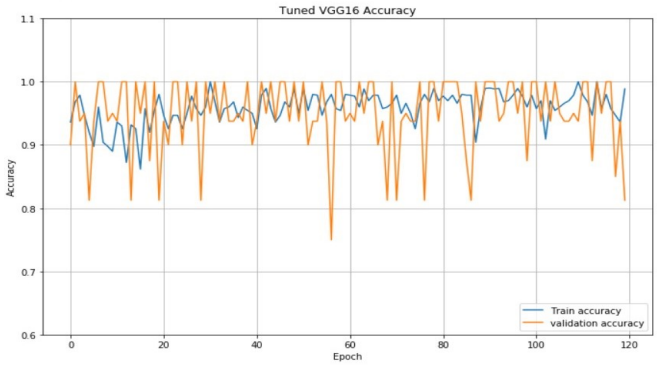
Through GridsearchCV, we found the best hyper parameters to be:

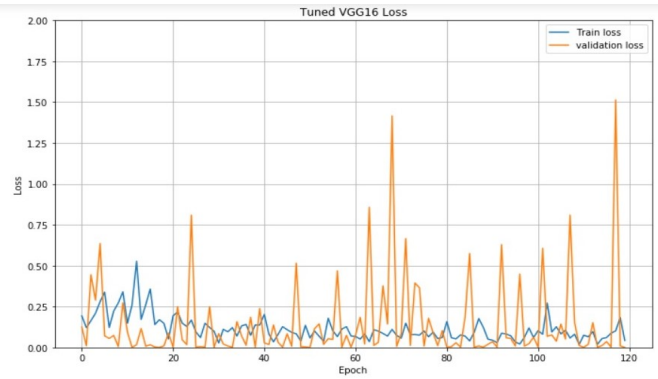
- learning_rate: '0.001'
- epochs: '120'
- activation:'relu'
- dropout: '0.2'

1) Architecture:

- we used VGG16 pretrained on Imagenet as base feature extractor.
- In the first fully connected dense layer, 256 nodes with the relu activation function followed by a dropout layer with a dropout probability of '0.20' is used which is optimized though GridsearchCV.
- The final dense layer had 1 node with a sigmoid activation as we are dealing with a binary classification problem.
- The model is fitted using the Adam optimizer with a learning_rate of 0.001 for 120 epochs. The loss function used is binary crossentropy.

2) Results: After tuning and training the model, we got a validation accuracy of 100% and a training accuracy of 96%. The validation loss converged to close to 0 and the traning loss was very low at 0.1068. Overall, we can see that the model performance improved from that of the baseline VGG16.

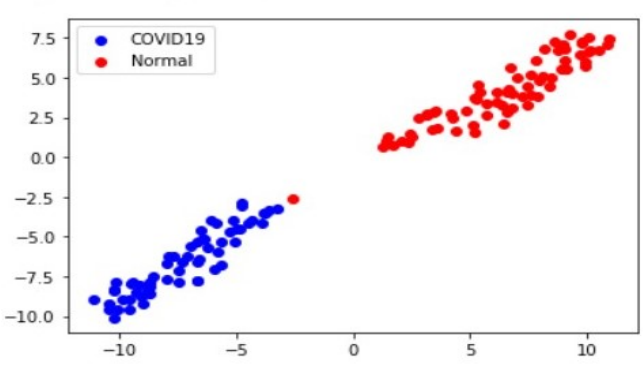




F. *TSNE Plot*

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

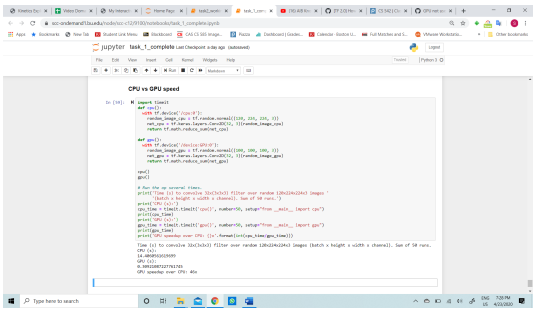
We decided to extract features from our 1st dense layer and plot the features. We can see that our model performed very well except one false positive between the 2 classes.



G. *CPU vs GPU speedup*

For the speed up, we decided to compare the speeds using a toy problem. we compared the time required by the CPU and GPU to run a convolution task of convolving 32 filters of size(3x3x3) over 120 images of size (224x224x3). we ran this entire process 50 times and compared the speed. we did so in an attempt to replicate the size of image we used in this challenge, the number of images we had and the size of the filter we used in our architecture.

It takes the CPU around 14.5 seconds to perform this task while it takes the GPU only 0.31 seconds. The GPU is roughly 46x faster than the CPU.



II. Task 2 : Multi Class Classification

For this task, we decided to run four pretrained models along with our own model. we ran VGG16, InceptionResnetv2, Resnet50, Xception and own model all on default hyper parameters to get baselines. Based on the baseline results we decided to tune three of these models to evaluate on the test set.

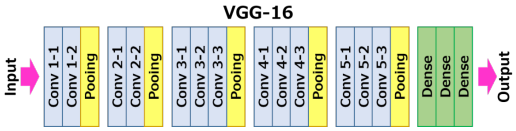
A. *VGG16*

The first model that we used as a baseline was the VGG16 based model. we decided to use the pretrained weights trained on Imagenet and added our own fully connected layers at the top.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 7, 7, 512)         14714688
_________________________________________________________________
flatten_1 (Flatten)          (None, 25088)             0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               6422784
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 1028
=================================================================
Total params: 21,138,500
Trainable params: 6,423,812
Non-trainable params: 14,714,688
_________________________________________________________________
```

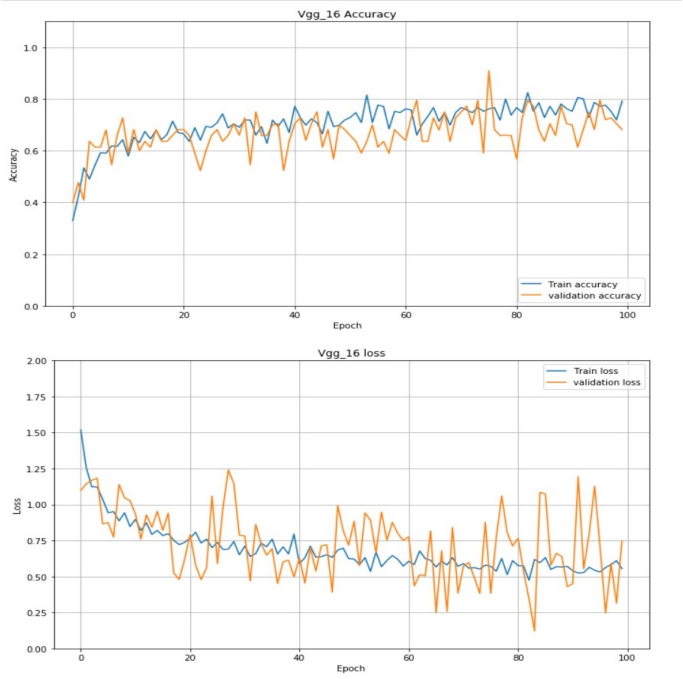
1) *Architecture:*

- we used a VGG16 pretrained on Imagenet as our base feature extractor.



- In our first fully connected Dense layer, we used 256 nodes with the relu activation function followed by a dropout layer with a dropout probability of 0.25.
- Our final dense layer had 4 node with the 'softmax' activation as we are dealing with a multi class classification problem.
- The model is fitted using the Adam optimizer with the default learning rate of 0.0001 for 100 epochs.The loss function used is binary crossentropy.

2) Results: The VGG16 model after 100 epochs had a training accuracy of 79% and a validation accuracy of 68%. The training loss was 0.55 and the validation loss was about 0.74



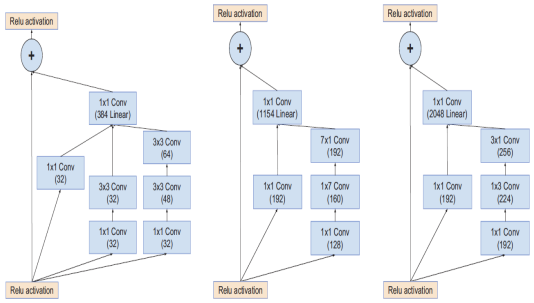
B. *InceptionResnetV2*

The Second model that we used for a baseline was the InceptionResnetV2 . we decided to use the pretrained weights trained on Imagenet and added our own fully connected layers at the top.

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_resnet_v2 (Model)  (None, 5, 5, 1536)        54336736
_________________________________________________________________
flatten_2 (Flatten)          (None, 38400)             0
_________________________________________________________________
dense_3 (Dense)              (None, 256)               9830656
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_4 (Dense)              (None, 4)                 1028
=================================================================
Total params: 64,168,420
Trainable params: 9,831,684
Non-trainable params: 54,336,736
_________________________________________________________________
```

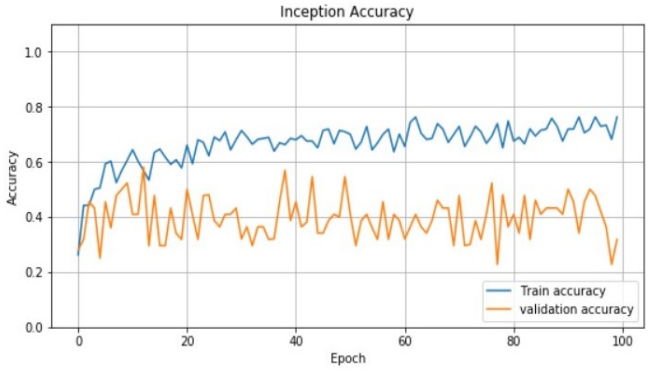
1) Architecture:

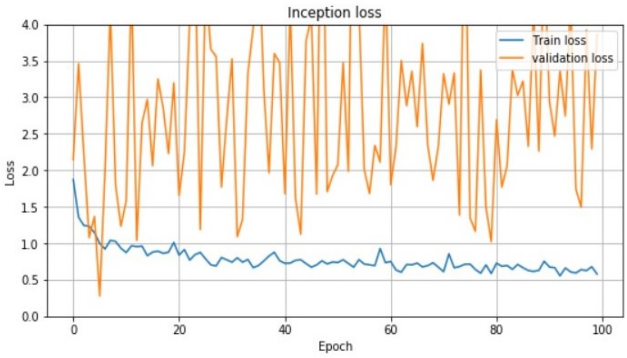
- The InceptionResnetV2 is a model that has 164 layers. It is an enhancement of the previous Inception v2.
- The InceptionResnetV2 brings back the idea of residual blocks by converting the three inception blocks to 'residual inception' blocks.
- The InceptionResNetV2 model has a total of 64,168,420 parameters (Trainable: 9,831,684 and Non Trainable: 54,336,736).



- In the first fully connected Dense layer, we used 256 nodes with the relu activation function followed by a dropout layer with a dropout probability of 0.25.
- The final dense layer had 4 node with the 'softmax' activation as we are dealing with a multi class classification problem.
- The model is fitted using the Adam optimizer with the default learning rate of 0.0001 for 100 epochs. The loss function used is categorical crossentropy.

2) Results: The InceptionResNetV2 model after 100 epochs had a training accuracy of 76% and a validation accuracy of 32%. The training loss was 0.57 and the validation loss was about 3.86. This models performance was comparable to the VGG16 on the training set but had very poor generalisation. This could be because it uses residual blocks and those models are generally very data heavy.





C. *ResNet50*

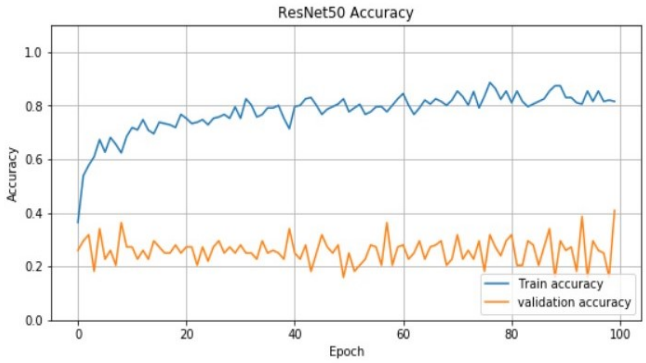
The third model that we used for a baseline was the Resnet50. we decided to use the pretrained weights trained on Imagenet and added our own fully connected layers at the top.

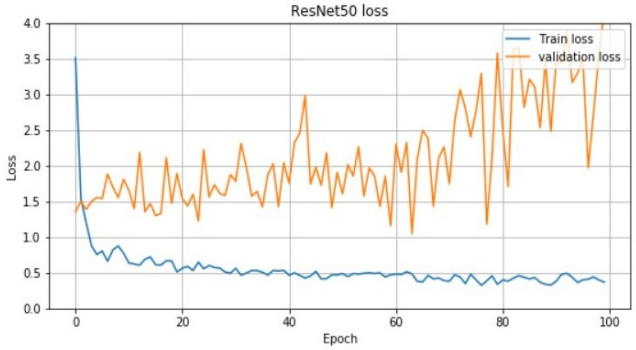
1) Architecture:

- In the 1st dense layer that we added on top of the ResNet50 feature extractor,256 nodes with the relu activation function followed by a dropout layer with a dropout probability of 0.25 is used.
- Our final dense layer had 4 node with the 'softmax' activation as we are dealing with a multiclass classification problem.
- The model is fitted using the Adam optimizer with the default learning rate of 0.0001 for 100 epochs. The loss function used is categorical crossentropy.

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Model)             (None, 7, 7, 2048)        23587712
_________________________________________________________________
flatten_3 (Flatten)          (None, 100352)            0
_________________________________________________________________
dense_5 (Dense)              (None, 256)               25690368
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_6 (Dense)              (None, 4)                 1028
=================================================================
Total params: 49,279,108
Trainable params: 25,691,396
Non-trainable params: 23,587,712
_________________________________________________________________
```

2) Results: The ResNet50 model had very good training accuracy at around 81.5% but poor generalization at 41%. The model had a training loss of 0.3685 and a validation loss of 4.1713 after 100 epochs. however, if we look at the plots, the loss never looked like converging. Accuracy and Loss:





D. *Xception*

The fourth model that we used for a baseline was Xception . We decided to use the pretrained weights trained on Imagenet and added our own fully connected layers at the top.

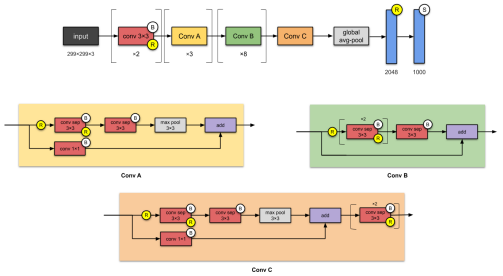
Model: "sequential_4"

Layer (type)	Output Shape	Param #
xception (Model)	(None, 7, 7, 2048)	20861480
flatten_4 (Flatten)	(None, 100352)	0
dense_7 (Dense)	(None, 256)	25690368
dropout_4 (Dropout)	(None, 256)	0
dense_8 (Dense)	(None, 4)	1028

Total params: 46,552,876
Trainable params: 25,691,396
Non-trainable params: 20,861,480

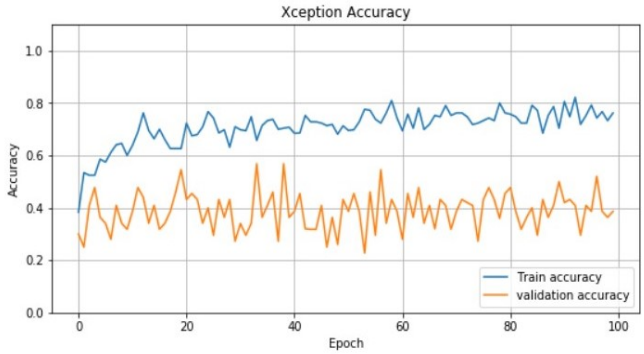
1) Architecture:

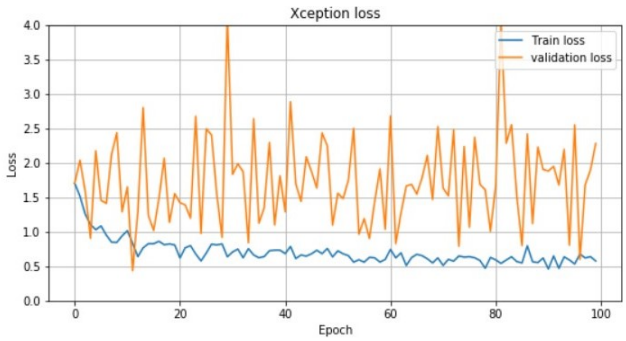
- Xception is an adaptation from Inception, where the Inception modules have been replaced with depth wise separable convolutions.
- Firstly, cross-channel (or cross-feature map) correlations are captured by 11 convolutions.
- Consequently, spatial correlations within each channel are captured via the regular 33 or 55 convolutions.
- The model does this entirely based on depth-wise separable convolution layers.



- The Xception model has a total of 46,552,876 parameters (Trainable: 25,691,396 and Non Trainable: 20,861,480).
- In the 1st dense layer that we added on top of the Xception feature extractor, we used 256 nodes with the relu activation function followed by a dropout layer with a dropout probability of 0.25.
- Our final dense layer had 4 node with the 'softmax' activation as we are dealing with a multiclass classification problem.
- We fit the model using the Adam optimizer with the default learning rate of 0.0001 for 100 epochs. The loss function we used was categorical crossentropy.

2) Results: The Xception model had very good training accuracy at around 76% but poor generalization at 38.64%. The model had a training loss of 0.5768 and a validation loss of 2.2823 after 100 epochs.





E. *Self Created Model*

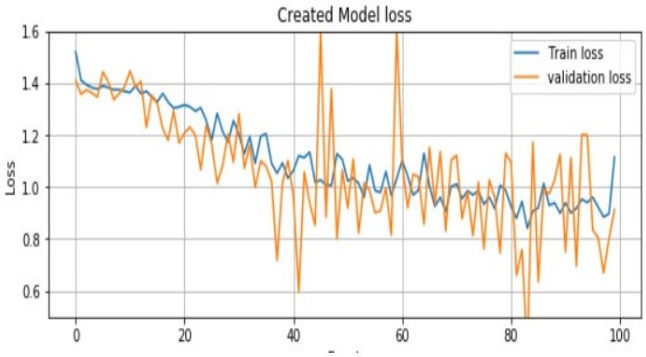
The last model that we used was one that we created on our own. We trained it on the given training data.

1) Architecture:

- We used 2 convolution layers followed by pooling layers.
- For both our convolution layers, we used 64 filters with a kernel size of (3x3), stride of 1 and no padding.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 222, 222, 64)	1792
max_pooling2d_1 (MaxPooling2	(None, 111, 111, 64)	0
conv2d_2 (Conv2D)	(None, 109, 109, 64)	36928
max_pooling2d_2 (MaxPooling2	(None, 54, 54, 64)	0
flatten_1 (Flatten)	(None, 186624)	0
dense_1 (Dense)	(None, 128)	23888000
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 64)	8256
dropout_2 (Dropout)	(None, 64)	0
dense_3 (Dense)	(None, 4)	260

Total params: 23,935,236
Trainable params: 23,935,236
Non-trainable params: 0



- In the 1st dense layer, we used 128 nodes with the relu activation function followed by a dropout layer with a dropout probability of 0.25.
- In our intermediate dense layer, we used 64 nodes with the 'relu' activation fucntion followed by a dropout layer with a dropout of 0.25.
- The final dense layer had 4 node with the 'softmax' activation as we are dealing with a multi class classification problem.
- We fit the model using the Adam optimizer with the default learning rate of 0.0001 for 100 epochs. The loss function we used was categorical crossentropy.

2) Results: Our model had a training accuracy at around 51% validation accuracy at around 52%. The model had a training loss of 1.1029 and a validation loss of 0.9131 after 100 epochs. Our model performed equally for for the training set and the validation set suggesting that our model did not overfit. It however had high bias which is not encouraged.

F. Tuned VGG16

In this section, we decided to tune VGG16 model hyperparameters as it had encouraging baselines. For parameter tuning, we decided to use sklearns GridsearchCV function.We tried different values for Learning rate, Dropouts and Activation function and did these over 3 folds. we tried using early stoppings to make the process faster. The values that we tried are:

- learning_rate: [0.0001,0.001,0.00001]
- activation:['sigmoid','relu']
- dropout: [0.15,0.20,0.25]

Following resutls from the GridsearchCV, we found the best hyperparemeters to be:

- learning_rate: '0.0001'
- activation:'sigmoid'
- dropout: '0.15'

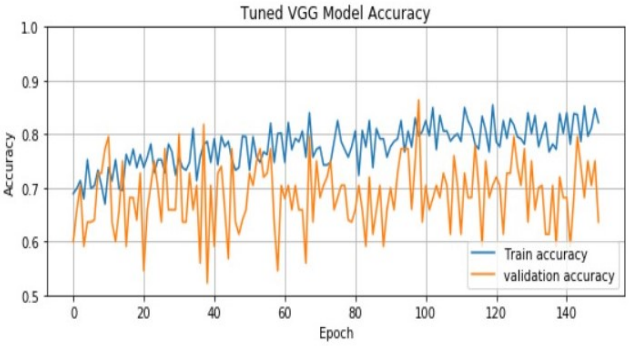
1) Architecture:

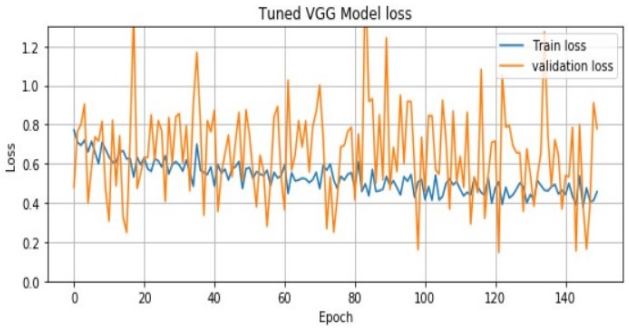
- We used the same VGG16 pretrained on Imagenet as our base feature extractor.
- In our first fully connected Dense layer, we used 256 nodes with the sigmoid activation function followed by a dropout layer with

a dropout probability of '0.15' which we optimized though GridsearchCV.
- Our final dense layer had 4 node with a 'softmax' activation function.
- The model is fitted using the Adam optimizer with a learning_rate of 0.0001 for 120 epochs. The loss function we used was categorical crossentropy.
- We used model checkpoints to save only the weights which reduced validation loss.
- We then loaded these weights while evaluating on the test set.

2) Results: After tuning and training the model, we got a validation accuracy of 63% and a training accuracy of 82%. The validation loss converged to 0.7785 and the training loss was very low at 0.4535.

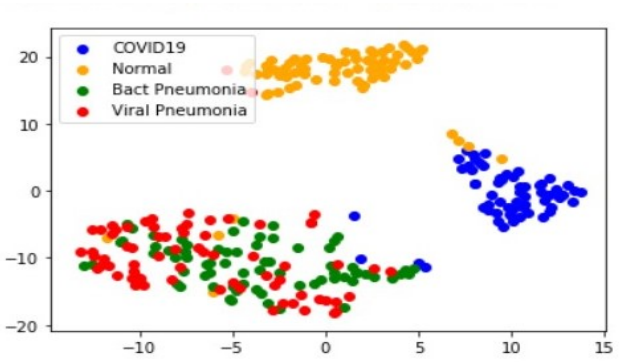
On the test set, we got a accuracy of 72.22% and a loss of 0.587





*3) **TSNE Plot**:* We decided to extract features from our 1st dense layer and plot the features. Overall, we can see that the 'COVID19' and 'Normal' classes are well separated with a few Normal patients being classified a s COVID19 positive. We feel this is still acceptable but not the other way around for this problem. Both the Pneumonia classes cannot be differentiated well which probably means they have very similar features.

The model does a satisfactory job at classifying between Normal, COVID19 and Pneumonia patients.



G. *Tuning our own model*

In this section, we decided to tune our own models hyperparameters. For parameter tuning, we decided to use sklearn's GridsearchCV function. We tried different values for Learning rate, Dropouts and Activation function and the number of nodes in our dense layer. We did these over 3 folds. The values that we tried are:

- learning_rate: [0.0001,0.001,0.01]
- activation:['sigmoid','relu']
- dropout: [0.15,0.20,0.25]
- units : [32,64]

Following resutls from the GridsearchCV, we found the best hyperparemeters to be:

- learning_rate: '0.001'
- activation:'relu'
- dropout: '0.15'
- units: '32'

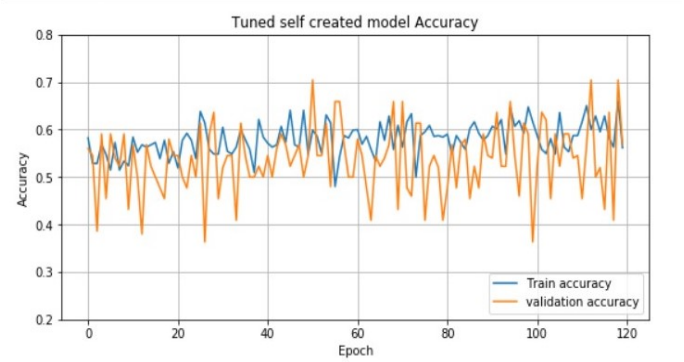
1) Architecture:

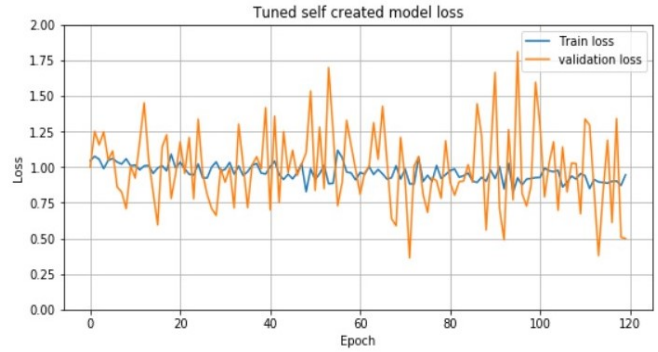
- We used the same architecture as we did for our baseline. We used 2 convolution layers with 64 filters and filter size of (3x3). We used a stride of 1 and no padding.
- we followed up our convolution layers with Max Pooling layers.

- In our first fully connected Dense layer, we used 128 nodes with the relu activation function followed by a dropout layer with a dropout probability of '0.15' which we optimized though GridsearchCV.
- In our second dense layer we had 32 nodes and a 'relu' activation function followed by a dropout layer with a dropout probability of '0.15'.
- Our final dense layer had 4 node with a 'softmax' activation function.
- The model is fitted using the Adam optimizer with a learning_rate of 0.001 for 120 epochs. The loss function we used was categorical_crossentropy
- we used model checkpoints to save only the weights which minimized validation loss.
- we then loaded these weights while evaluating on the test set.

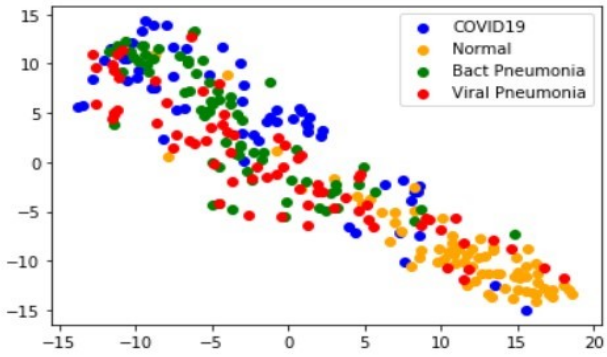
2) Results: After tuning and training the model, we got a validation accuracy of 56.82% and a training accuracy of 56.19%. The validation loss converged to 0.4992 and the training loss at 0.9462.

On the test set, we got a accuracy of 41.66% and a loss of 1.163 Accuracy and Loss:





3) TSNE Plot: We decided to extract features from our 1st dense layer and plot the features. There is a visible cluster for the 'Normal' cluster which can be differentiated from the other 3 classes. however, the model is not able to differentiate among the 'COVID19' class, the 'Bacterial Pneumonia' or the 'Viral Pneumonia' classes. This model may not be able to identify the differences in the features between these 3 classes. This is also the reason our model has such a poor performance on the test set.



III. Comparisons and Future Scope

A. TASK 1

For task 1, out of the 4 models that we used, the VGG16 had the best baseline. The VGG19 did had similar results comapared to VGG16 however, it had 26% more parameters than the VGG16. For this task, we decided to use pretrained weights and hence the complexity of the model did not affect our computation time too much. however, If we had to train these models from scratch, the VGG16 would have been much faster.

The Resnet50 had very good training accuracy, the best among all the models, but it had very low validation accuracy. The model had low bias and very high variance which is not desirable. Given more time, we believe this model could perform better if trained for more epochs and regularized more heavily. we would probably also try to use early stopping while tuning on the validation set to avoid over training the model. ResNet50 uses residual blocks and is generally very data hungry. The final model that we used was our own. It had the poorest performance among all the models

probably because we tried training it on the training set itself. It had a lot of parameters(23,935,041) that had to be learnt and only around 130 images which is very less for training a model this size. we believe if we had trained this model on imagenet and used those weights, it might have done better. A better architecture could have also helped.

B. TASK 2

For task2, out of the 5 models that we used, the VGG16 again had the best baselines overall. The Resnet50 again had a very good performance on the training set(better than the VGG16) but had poor generalization on the validation set similar to task1. We believe this model could be improved if regularized well. Maybe another layer of dropouts could have helped or maybe using early stopping could help avoiding the overfitting on the training set.

The InceptionResNetV2 and the Xception had a similar baseline performance. both of them had around 70%-75% accuracy on the training set and around 30%-40% accuracy on the validation set. The InceptionResNetV2 is an adaptation of the Inception models but they have been reinforced with residual blocks similar to the ResNet50. The Xception model uses a similar architecture as the Inception. Both these models suffer from overfitting. Given more time, we would like to do some additional reading about residual blocks and their impact on overfitting as all 3 models based on residual blocks that we used, overfit the data. We believe both these models could be optimized with stronger regularization. We would like to try BatchNormalization instead of dropouts in the future coupled with early stopping while optimizing these models.

The last model that we used was our own model. It had the worst performance among all the models. we suspect this was the case due to the limited data we had to train this model.