## Separable CNN

As separable CNN transforms the image only once in depth-wise convolution and then elongate it to 64 channels, the number of trainable parameters is lesser than the standard CNN by 16837 parameters. Hence, the training session took shorter time to complete. On the hand, it is observed that the output shape of each layer in this model is similar to the respective layer in the standard CNN model.

```
#Separable CNN Base
from tensorflow import keras
from keras.layers import SeparableConv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization, Resizing, Activation
from keras import models

scnn= models.Sequential()
scnn.add(Resizing(224, 224))
scnn.add(SeparableConv2D(32, (3, 3), padding='same',
                 input_shape=(224, 224, 3)))
scnn.add(MaxPooling2D((2,2)))
scnn.add(Activation('relu'))
scnn.add(SeparableConv2D(64, (3, 3), padding='same'))
scnn.add(MaxPooling2D((2,2)))
scnn.add(Activation('relu'))
scnn.add(Flatten())
scnn.add(Dense(3, activation='Softmax'))

METRICS = [
      keras.metrics.CategoricalAccuracy(name='accuracy'),
      keras.metrics.AUC(name='auc')] # precision-recall curve

scnn.compile(loss='categorical_crossentropy', optimizer='Adam', metrics=METRICS)
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resizing_1 (Resizing)       (32, 224, 224, 3)         0

 separable_conv2d (Separable  (32, 224, 224, 32)       155
 Conv2D)

 max_pooling2d_2 (MaxPooling  (32, 112, 112, 32)       0
 2D)

 activation_2 (Activation)   (32, 112, 112, 32)        0

 separable_conv2d_1 (Separab  (32, 112, 112, 64)       2400
 leConv2D)

 max_pooling2d_3 (MaxPooling  (32, 56, 56, 64)         0
 2D)

 activation_3 (Activation)   (32, 56, 56, 64)          0

 flatten_1 (Flatten)         (32, 200704)              0

 dense_1 (Dense)             (32, 3)                   602115

=================================================================
Total params: 604,670
Trainable params: 604,670
Non-trainable params: 0
_________________________________________________________________
```

## Performance

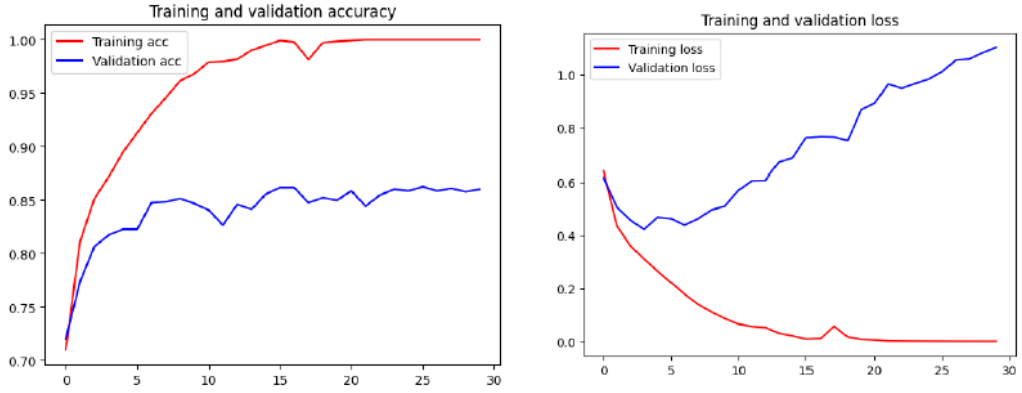



*Figure 1 Accuracy and Loss for Separable CNN Base Model*

Despite the training accuracy achieved 100% with relatively small loss 0.0003, the model fails to generalize on unseen data. Validation accuracy stays around 85% from epoch 5 to 30, while the validation loss increases gradually from epoch 5 to 30 until 1.10 loss. Hence, regularization techniques were added in the next section.

## Regularization

```
from tensorflow import keras
from keras.layers import SeparableConv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization, Resizing, Activation
from keras import models

scnn_reg= models.Sequential()
scnn_reg.add(Resizing(90, 90))
scnn_reg.add(SeparableConv2D(32, (3, 3), padding='same',
                input_shape=(224, 224, 3)))
scnn_reg.add(MaxPooling2D((2,2)))
scnn_reg.add(Activation('relu'))
scnn_reg.add(Dropout(0.5))
scnn_reg.add(BatchNormalization())
scnn_reg.add(SeparableConv2D(64, (3, 3), padding='same'))
scnn_reg.add(MaxPooling2D((2,2)))
scnn_reg.add(Activation('relu'))
scnn_reg.add(Dropout(0.5))
scnn_reg.add(BatchNormalization())
scnn_reg.add(Flatten())
scnn_reg.add(Dense(3, activation='Softmax'))

METRICS = [
      keras.metrics.CategoricalAccuracy(name='accuracy'),
      keras.metrics.AUC(name='auc')] # precision-recall curve

scnn_reg.compile(loss='categorical_crossentropy', optimizer='Adam', metrics=METRICS)
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resizing (Resizing)         (32, 90, 90, 3)           0

 separable_conv2d (Separabl  (32, 90, 90, 32)          155
 eConv2D)

 max_pooling2d (MaxPooling2  (32, 45, 45, 32)          0
 D)

 activation (Activation)     (32, 45, 45, 32)          0

 dropout (Dropout)           (32, 45, 45, 32)          0

 batch_normalization (Batch  (32, 45, 45, 32)          128
 Normalization)

 separable_conv2d_1 (Separa  (32, 45, 45, 64)          2400
 bleConv2D)

 max_pooling2d_1 (MaxPoolin  (32, 22, 22, 64)          0
 g2D)

 activation_1 (Activation)   (32, 22, 22, 64)          0

 dropout_1 (Dropout)         (32, 22, 22, 64)          0

 batch_normalization_1 (Bat  (32, 22, 22, 64)          256
 chNormalization)

 flatten (Flatten)           (32, 30976)               0

 dense (Dense)               (32, 3)                   92931

=================================================================
Total params: 95870 (374.49 KB)
Trainable params: 95678 (373.74 KB)
Non-trainable params: 192 (768.00 Byte)
_________________________________________________________________
```

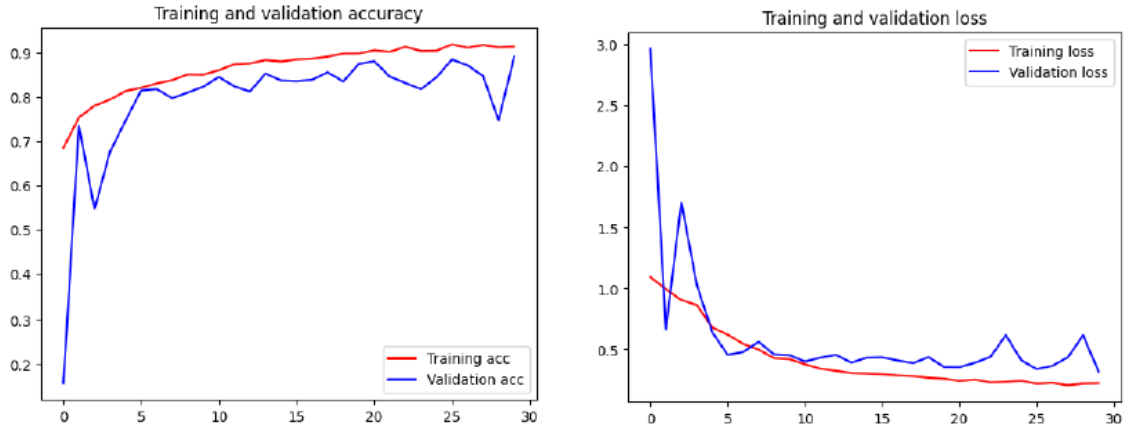



*Figure 2 Accuracy and Loss for Separable CNN Regularized Model*

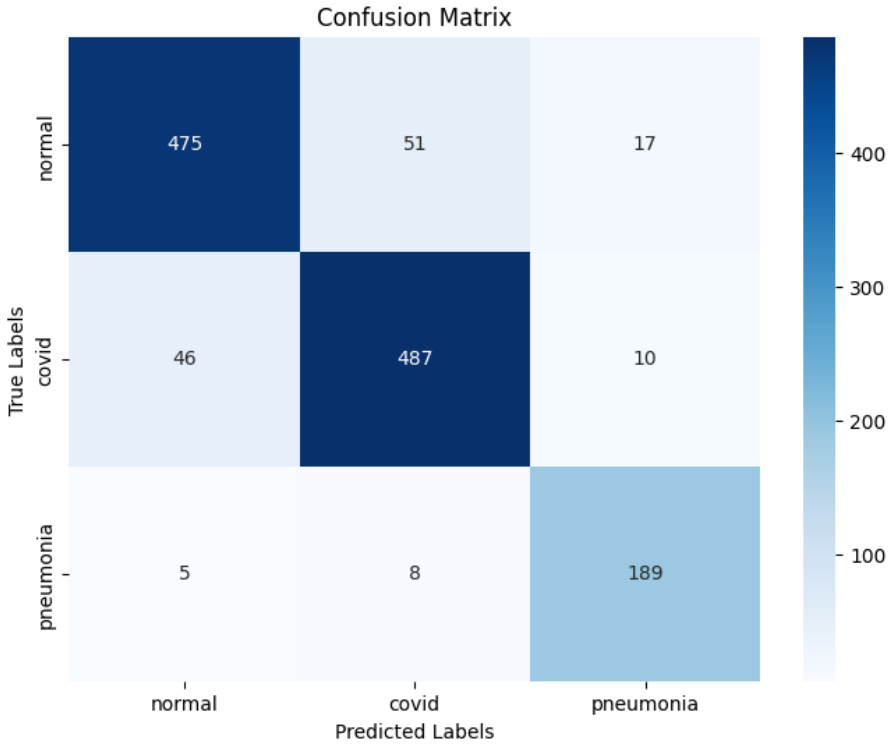
It is observed that overfitting is resolved after regularization. Both training and validation accuracy as well as loss moves towards the same direction. Throughout the 30 epochs, the training accuracy and loss improve gradually. On the other hand, there is a sharp increase in validation accuracy between epoch 0 to 3. It is observed that the validation loss experienced a steep fall at epoch 1, in which the

loss in validation set approaches 0.31 at the end of training. At epoch=30, the training accuracy was 91.32% with 0.2207 loss, whereas for validation set, the accuracy was 89.04% with 0.3161 loss.

**Test Set**

```
import matplotlib.pyplot as plt
import seaborn as sns

# Visualize the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_scnn, annot=True, fmt='d', cmap='Blues',xticklabels=classLst, yticklabels=classLst)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```




Based on the confusion matrix, the model correctly classifies 475 normal, 487 Covid-19 and 189 pneumonia CXR. On the contrary, 68 normal, 56 covid and 12 pneumonia CXR were misclassified. Similarly, it is observed that the model struggles to differentiate between normal and Covid-19 CXR.

```
from sklearn.metrics import classification_report
print(classification_report(true_labels, predicted_classes, target_names=classLst))
```

```
              precision    recall  f1-score   support

      normal       0.90      0.87      0.89       543
       covid       0.89      0.90      0.89       543
   pneumonia       0.88      0.94      0.90       202

    accuracy                           0.89      1288
   macro avg       0.89      0.90      0.90      1288
weighted avg       0.89      0.89      0.89      1288
```

The macro average precision, recall and f1-score are 0.89, 0.90 and 0.89 respectively. Among all classes, the model shows the highest precision in normal class while highest recall in pneumonia class. In other words, when the model classifies the CXR as normal, it is correct 90% of the time. Besides that, the model correctly identifies 94% of all pneumonia CXR. The macro average f1-score which consider all classes as equally important is close to 1.