Standard CNN

In model one, standard 2D convolutional layer was used. As shown in figure 12, this model consists of 621057 trainable parameters. In convolution layer, the number of parameters is determined by multiplying kernel size with the number of channels of image and adding one bias for every filter. Hence, each will have 3*3*3+1=28 params and 32 filters have 896 params. Next, the flattened output of second max-pooling layer yields a vector with 200,704 units. With the 3 units in fully-connected layer, the number of params is 200,704*3+3= 602,115 (Ding, 2020).

```
#Traditional CNN Base
from tensorflow import keras
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization, Resizing, Activation
from keras import models

cnn= models.Sequential()
cnn.add(Resizing(224, 224))
cnn.add(Conv2D(32, (3, 3), padding='same',
               input_shape=(224, 224, 3)))
cnn.add(MaxPooling2D((2,2)))
cnn.add(Activation('relu'))
cnn.add(Conv2D(64, (3, 3), padding='same'))
cnn.add(MaxPooling2D((2,2)))
cnn.add(Activation('relu'))
cnn.add(Flatten())
cnn.add(Dense(3, activation='Softmax'))

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

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

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

 conv2d (Conv2D)             (32, 224, 224, 32)        896

 max_pooling2d (MaxPooling2D  (32, 112, 112, 32)       0
 )

 activation (Activation)     (32, 112, 112, 32)        0

 conv2d_1 (Conv2D)           (32, 112, 112, 64)        18496

 max_pooling2d_1 (MaxPooling  (32, 56, 56, 64)         0
 2D)

 activation_1 (Activation)   (32, 56, 56, 64)          0

 flatten (Flatten)           (32, 200704)              0

 dense (Dense)               (32, 3)                   602115

=================================================================
Total params: 621,507
Trainable params: 621,507
Non-trainable params: 0
_________________________________________________________________
```

Performance

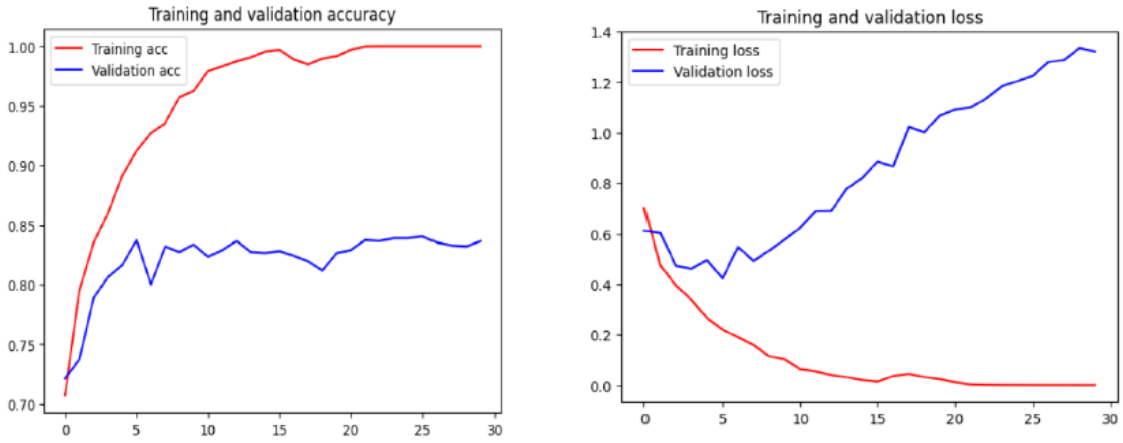


Figure 1 Accuracy and Loss of training and validation set for CNN base model.

Throughout the 30 epochs, the accuracy and loss graphs show that overfitting occurs, it is observed that the accuracy and loss in training set was improving. At epoch 30, the model achieved 100% accuracy and 0.0003 loss. On the other hand, the validation accuracy increased from 73% to 83.67% from epoch 0 to 5 but remained almost stagnant thereafter despite showing some small fluctuations. Similarly, training loss decreased from 0.7 to almost 0 at from epoch 0 to 10. However, validation loss increased to more than twice the amount of loss at epoch 0. This model does not generalize well on unseen data even though training accuracy and loss showed good performance. Even though the accuracy is still within an acceptable range, but the validation loss is high, and it moves towards the opposite direction. Hence, regularization was applied in the next section. On the other hand, as the training accuracy of the model has achieved 100%, this model was not tuned further.

Regularization

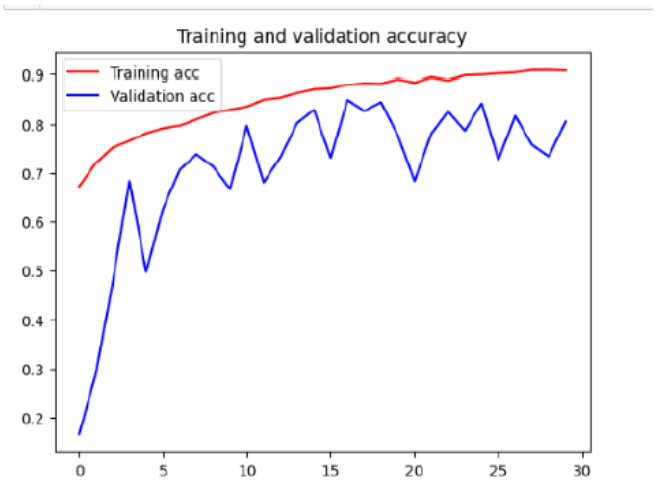
The occurrence of overfitting in the base model is due to the large weights in the network, which causes an unstable network. Regularization can be applied to update the learning algorithm to keep the weights small. In this section, dropout and batch normalization layers were added to the base model. After the first and second convolution layer, a 0.5 dropout rate was applied followed by batch normalization. The images were also resized to 90*90 to reduce the number of parameters.

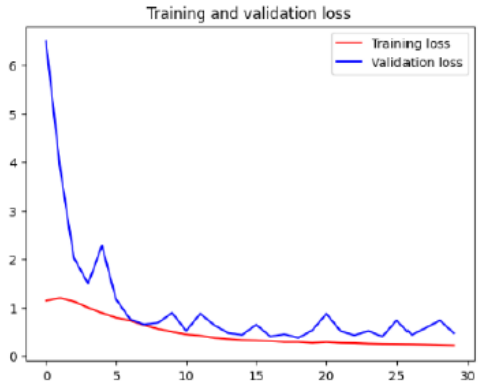
When applying dropout layer, some number of layer outputs are dropped at a given rate. As a result, the interdependent weights amongst the neurons are decreased (Kothiya, 2020). Batch normalization helps to improve the stability and speed of the network by normalizing the inputs, so that they are normally distributed with 0 mean and unit variance. As shown in the model summary, with regularization, there were 192 non-trainable parameters and 112515 trainable parameters.

```
from tensorflow import keras
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization, Resizing, Activation, Input
from keras import models

cnn_reg= models.Sequential()
cnn_reg.add(Resizing(90, 90))
cnn_reg.add(Input((224,224,3)))
cnn_reg.add(Conv2D(32, (3, 3), padding='same'))
cnn_reg.add(MaxPooling2D((2,2)))
cnn_reg.add(Activation('relu'))
cnn_reg.add(Dropout(0.5))
cnn_reg.add(BatchNormalization())
cnn_reg.add(Conv2D(64, (3, 3), padding='same'))
cnn_reg.add(MaxPooling2D((2,2)))
cnn_reg.add(Activation('relu'))
cnn_reg.add(Dropout(0.5))
cnn_reg.add(BatchNormalization())
cnn_reg.add(Flatten())
cnn_reg.add(Dense(3, activation='Softmax'))

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

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

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

 input_1 (InputLayer)        multiple                  0

 conv2d (Conv2D)             (32, 90, 90, 32)          896

 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)

 conv2d_1 (Conv2D)           (32, 45, 45, 64)          18496

 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: 112707 (440.26 KB)
Trainable params: 112515 (439.51 KB)
Non-trainable params: 192 (768.00 Byte)
_________________________________________________________________
```

Performance



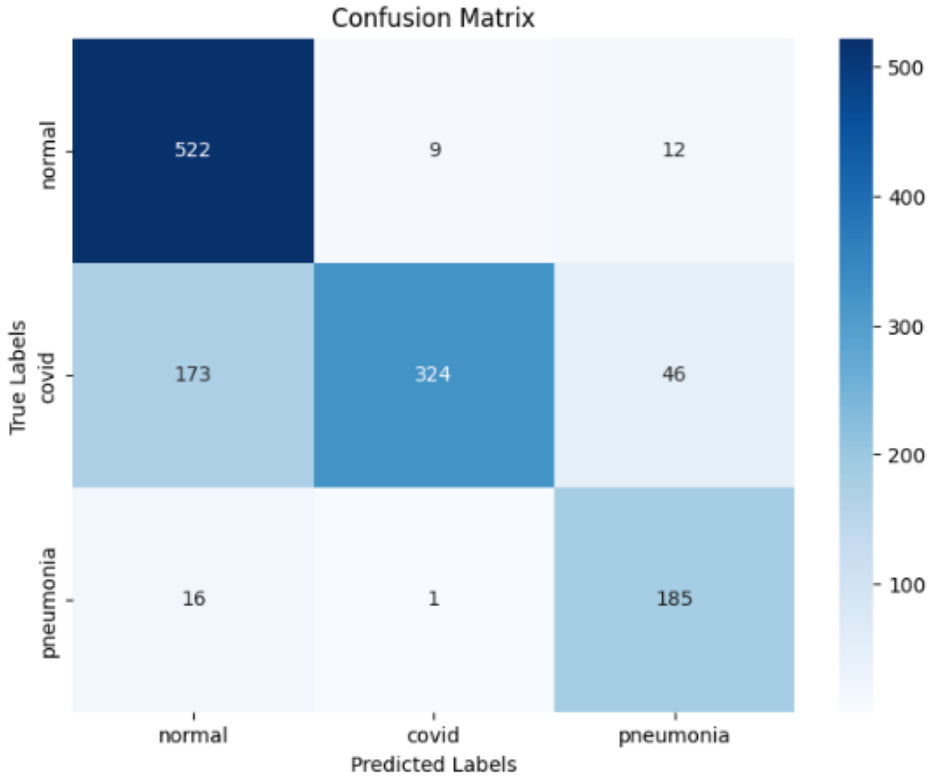


It is observed that by applying regularization techniques, the model performance in validation set has greatly improved. Throughout the 30 epochs, it is observed that both training and validation accuracy is improving gradually with higher accuracy in training set on average. At epoch=30, the training accuracy was close to 90.76% whereas validation accuracy was about 80.48%. On the other hand, the training loss is 0.2215 whereas validation loss is 0.4804.

Test Set

The model was further evaluated on test set with 1288 images, the model achieved 80.05% accuracy with 0.9405 loss. Based on the confusion matrix, the model correctly classifies 522 normal, 324 Covid-19 and 185 pneumonia CXR. On the contrary, 21 normal, 219 covid and 17 pneumonia CXR were misclassified. It is observed that the model struggles to differentiate between normal and Covid-19 CXR.

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

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



The precision, recall and f1-score for each class is as shown in the output below. Among all classes, the model shows the highest precision in covid class while highest recall in normal class. In other words, when the model classifies the CXR as covid, it is correct 97% of the time. Besides that, the model correctly identifies 83% of all normal CXR. The macro average f1-score which measures the balance between precision and recall for all classes is 0.8, which is close to 1.

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

```
              precision    recall  f1-score   support

      normal       0.73      0.96      0.83       543
       covid       0.97      0.60      0.74       543
   pneumonia       0.76      0.92      0.83       202

    accuracy                           0.80      1288
   macro avg       0.82      0.82      0.80      1288
weighted avg       0.84      0.80      0.79      1288
```