Multiclass classification:

(a) Download the CIFAR-10 [2] and load the pickled data into your program.

(b) Build a basic convolutional neural networks with several convolution, pooling, and normalization layers. Flatten the output of the convolution layers and pass it to a single dense layer that will produce the output using softmax activation.

(c) Test the performance of the model you built and tune hyper parameters as needed.

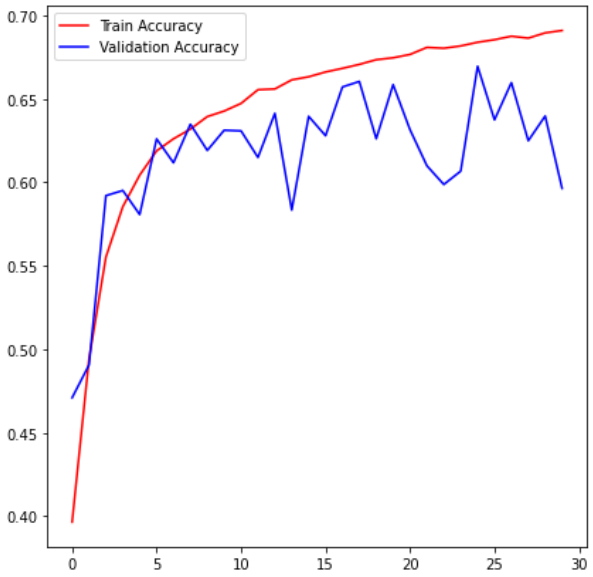
(d) Add one or two inception blocks and test performance.

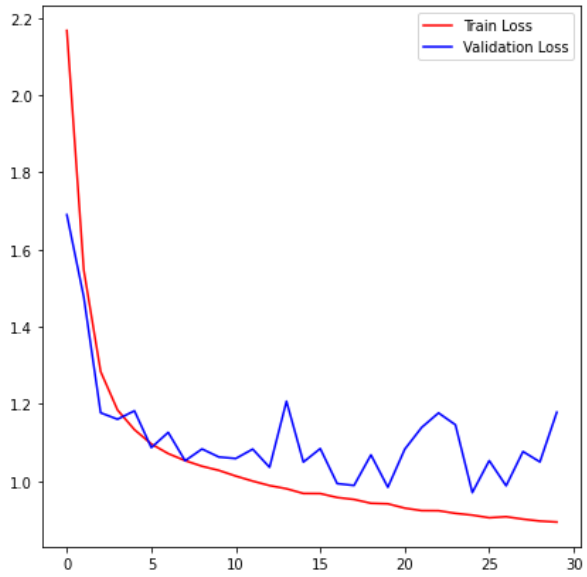
(e) Remove the inception blocks and add one or two residual blocks instead. Test performance and compare to previous results.

1. **Hyperparameter tuning:**

   i. Initial model with 2 convolution layers, max pooling layers, Adding dropout rate 0.2: Model summary:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         (None, 32, 32, 3)         0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 32, 32, 64)        256
_________________________________________________________________
leaky_re_lu_6 (LeakyReLU)    (None, 32, 32, 64)        0
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 16, 16, 64)        0
_________________________________________________________________
dropout_7 (Dropout)          (None, 16, 16, 64)        0
_________________________________________________________________
batch_normalization_6 (Batch (None, 16, 16, 64)        256
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 16, 16, 128)       73856
_________________________________________________________________
leaky_re_lu_7 (LeakyReLU)    (None, 16, 16, 128)       0
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 8, 8, 128)         0
_________________________________________________________________
dropout_8 (Dropout)          (None, 8, 8, 128)         0
_________________________________________________________________
batch_normalization_7 (Batch (None, 8, 8, 128)         512
_________________________________________________________________
flatten_2 (Flatten)          (None, 8192)              0
_________________________________________________________________
dropout_9 (Dropout)          (None, 8192)              0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                81930
=================================================================
Total params: 156,810
Trainable params: 156,426
Non-trainable params: 384
_________________________________________________________________
```





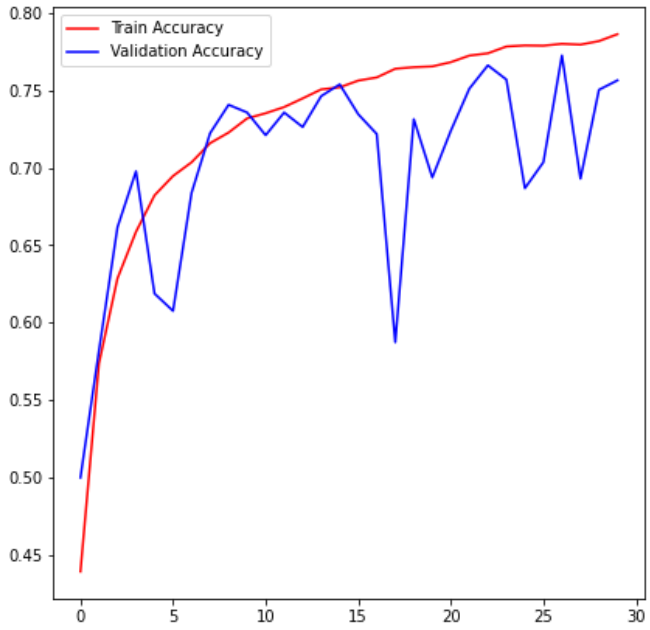
2. Adding inception blocks:

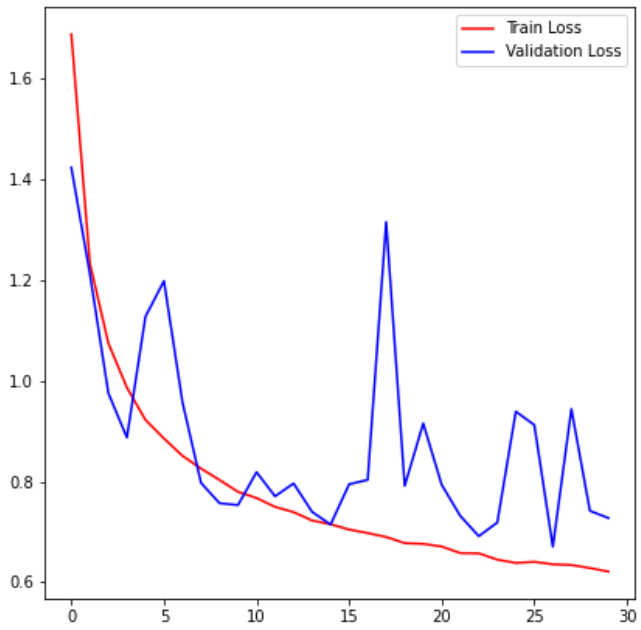
```
inception1 = Conv2D(32, kernel_size=1, strides=1, input_shape=self.img_shape, padding
= 'same')(img)
   hid1 = LeakyReLU()(inception1)
   hid1 = MaxPooling2D((2,2))(hid1)
   hid1 = Dropout(0.2)(hid1)
   hid1 = BatchNormalization()(hid1)
   print(hid1)

   inception2 = Conv2D(32, kernel_size=3, strides=1, input_shape=self.img_shape,
padding = 'same')(img)
   hid2 = LeakyReLU()(inception2)
   hid2 = MaxPooling2D((2,2))(hid2)
   hid2 = Dropout(0.2)(hid2)
   hid2 = BatchNormalization()(hid2)
   print(hid2)

   inception3 = Conv2D(32, kernel_size=5, strides=1, input_shape=self.img_shape,
padding = 'same')(img)
   hid3 = LeakyReLU()(inception3)
   hid3 = MaxPooling2D((2,2))(hid3)
   hid3 = Dropout(0.2)(hid3)
   hid3 = BatchNormalization()(hid3)
   print(hid3)
```

```
Model: "model_3"
_________________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
=========================================================================================
input_4 (InputLayer)         (None, 32, 32, 3)         0
_________________________________________________________________________________________
conv2d_9 (Conv2D)            (None, 32, 32, 32)        128         input_4[0][0]
_________________________________________________________________________________________
conv2d_10 (Conv2D)           (None, 32, 32, 32)        896         input_4[0][0]
_________________________________________________________________________________________
conv2d_11 (Conv2D)           (None, 32, 32, 32)        2432        input_4[0][0]
_________________________________________________________________________________________
leaky_re_lu_8 (LeakyReLU)    (None, 32, 32, 32)        0           conv2d_9[0][0]
_________________________________________________________________________________________
leaky_re_lu_9 (LeakyReLU)    (None, 32, 32, 32)        0           conv2d_10[0][0]
_________________________________________________________________________________________
leaky_re_lu_10 (LeakyReLU)   (None, 32, 32, 32)        0           conv2d_11[0][0]
_________________________________________________________________________________________
max_pooling2d_8 (MaxPooling2D)  (None, 16, 16, 32)     0           leaky_re_lu_8[0][0]
_________________________________________________________________________________________
max_pooling2d_9 (MaxPooling2D)  (None, 16, 16, 32)     0           leaky_re_lu_9[0][0]
_________________________________________________________________________________________
max_pooling2d_10 (MaxPooling2D) (None, 16, 16, 32)     0           leaky_re_lu_10[0][0]
_________________________________________________________________________________________
dropout_10 (Dropout)         (None, 16, 16, 32)        0           max_pooling2d_8[0][0]
_________________________________________________________________________________________
dropout_11 (Dropout)         (None, 16, 16, 32)        0           max_pooling2d_9[0][0]
_________________________________________________________________________________________
dropout_12 (Dropout)         (None, 16, 16, 32)        0           max_pooling2d_10[0][0]
_________________________________________________________________________________________
```

```
batch_normalization_8 (BatchNor (None, 16, 16, 32)   128         dropout_10[0
][0]
__________________________________________________________________________________
__________________
batch_normalization_9 (BatchNor (None, 16, 16, 32)   128         dropout_11[0
][0]
__________________________________________________________________________________
__________________
batch_normalization_10 (BatchNo (None, 16, 16, 32)   128         dropout_12[0
][0]
__________________________________________________________________________________
__________________
concatenate_2 (Concatenate)     (None, 16, 16, 96)   0           batch_normal
ization_8[0][0]
                                                                 batch_normal
ization_9[0][0]
                                                                 batch_normal
ization_10[0][0]
__________________________________________________________________________________
__________________
conv2d_12 (Conv2D)              (None, 16, 16, 64)   6208        concatenate_
2[0][0]
__________________________________________________________________________________
__________________
leaky_re_lu_11 (LeakyReLU)      (None, 16, 16, 64)   0           conv2d_12[0]
[0]
__________________________________________________________________________________
__________________
max_pooling2d_11 (MaxPooling2D) (None, 8, 8, 64)     0           leaky_re_lu_
11[0][0]
__________________________________________________________________________________
__________________
dropout_13 (Dropout)            (None, 8, 8, 64)     0           max_pooling2
d_11[0][0]
__________________________________________________________________________________
__________________
batch_normalization_11 (BatchNo (None, 8, 8, 64)     256         dropout_13[0
][0]
__________________________________________________________________________________
__________________
conv2d_13 (Conv2D)              (None, 8, 8, 128)    73856       batch_normal
ization_11[0][0]
__________________________________________________________________________________
__________________
leaky_re_lu_12 (LeakyReLU)      (None, 8, 8, 128)    0           conv2d_13[0]
[0]
__________________________________________________________________________________
__________________
max_pooling2d_12 (MaxPooling2D) (None, 4, 4, 128)    0           leaky_re_lu_
12[0][0]
__________________________________________________________________________________
__________________
dropout_14 (Dropout)            (None, 4, 4, 128)    0           max_pooling2
d_12[0][0]
__________________________________________________________________________________
__________________
```

```
batch_normalization_12 (BatchNo (None, 4, 4, 128)     512         dropout_14[0
][0]
__________________________________________________________________________________
__________________________
flatten_3 (Flatten)             (None, 2048)          0           batch_normal
ization_12[0][0]
__________________________________________________________________________________
__________________________
dropout_15 (Dropout)            (None, 2048)          0           flatten_3[0]
[0]
__________________________________________________________________________________
__________________________
dense_3 (Dense)                 (None, 10)            20490       dropout_15[0
][0]
==================================================================================
======================
Total params: 105,162
Trainable params: 104,586
Non-trainable params: 576
__________________________________________________________________________________
__________________________
```

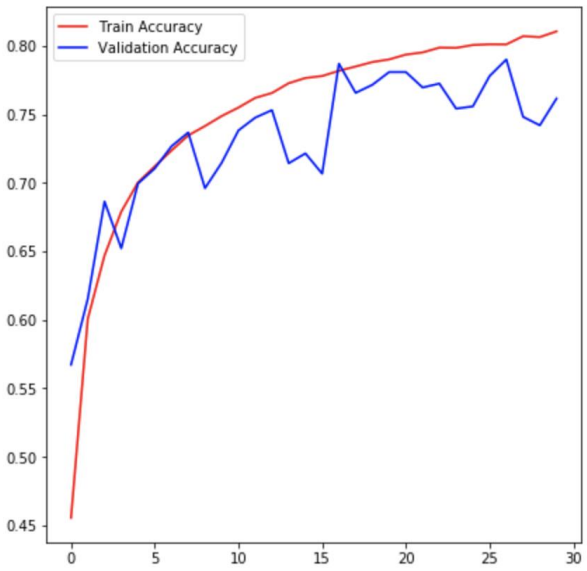


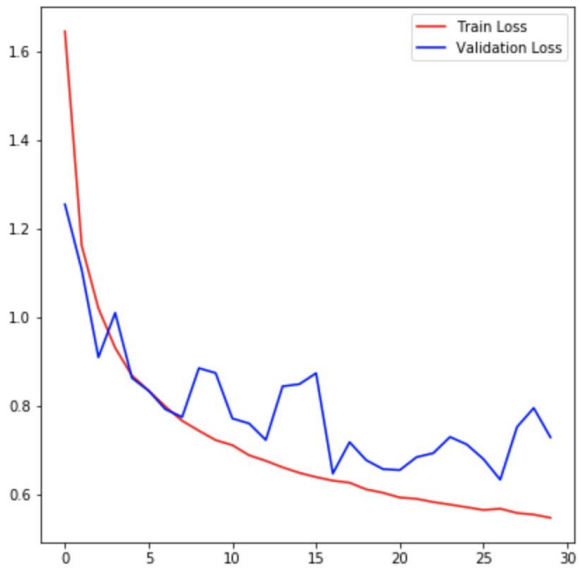


Adding inception blocks has increased the performance to 75% on the validation set. This is because inception blocks increase the dimensionality of the features extracted by the convolution blocks.

3. Residual Block

- Adding residual blocks has a better performance than the inception blocks.





Adding a second residual block

- A second residual block has not improved the performace, and is not necessary for our use case.