Implementing ResNet Architecture for Image Classification

1. **Model Architecture:**

   For the ResNet-18 model architecture, I have defined a **"Buiding_block"** class. This class defines the basic architecture template present within each ResNet layer. It consists of 2 Conv2D layers; each layer followed by a batch normalization layer.

   Here, the stride can be 1 or 2.

   If Stride is 1:

   For the conv2D layers, the kernel size is 3X3 and the padding is always 1.

   However, if stride is 2:

   For the conv2D layers, we have kernel size of 1X1 and no padding. Then we add a batch normalization layer.

   Additionally, I have defined a helper function that does zero padding to our input to maintain its dimensions. This is done prior to the skip connection that adds the output of the previous layer with stride = 1 with the output of the layer with stride = 2.

   The "helper_function" is only used if x1.shape != x2.shape, before performing x1+ x2.

   The activation function "ReLU" is used throughout the "Building_block" architecture.

   Now, we finally define the **"ResNet18"** class, which is built upon the "Building_block" class. Here, we first define our first Conv2D layer that uses kernel size of 7X7, stride of 2 and padding of 3. This is followed by the Batch Normalization layer, ReLU activation function and Max-Pool layer with kernel size of 3X3, stride = 2, padding = 1. The in_channel is 3 and out_channel is 64.

   Then we define each layer one by one.

   The first layer is a sequence of layers consisting of 2 Building_block architecture with in_channels and out_channels of 64 and stride=2.

   The 2nd layer is a sequence of layers consisting of 2 Building_block architecture with in_channels and out_channels of 64 and 128 and 128 and 128 respectively and stride=2.

   The 3rd layer is a sequence of layers consisting of 2 Building_block architecture with in_channels and out_channels of 128 and 256 and 256 and 256 respectively and stride=2.

The 4th layer is a sequence of layers consisting of 2 Building_block architecture with in_channels and out_channels of 128 and 256 and 256 and 256 respectively and stride=2.

Following the 4th layer, we have added an Average_pooling layer.

Finally, we have a fully connected (Linear) layer that takes in_channel of 512 and transforms it into the number of classes (Here, 3), followed by Softmax activation function.

2. **Techniques Used:**
    a. L2 Regularization: L2 regularization has been applied to reduce overfitting of the model. After applying L2 regularization, the test accuracy has increased to 88.72%. L2 regularization can be applied by setting a weight decay that is greater than 0. We have chosen the value of 1e-5.

    b. Batch Normalization: Batch Normalization has been applied after layers to combat overfitting of the ResNet architecture model. After applying BN layers, the test accuracy has increased to approximately 90%. So, the model performance has increased a lot by adding BN.

3. **Result Analysis:**
    After training the model, the train, validation and test Accuracies and Losses are as follows:

Train Accuracies: 84.88095238095238%

Validation Accuracies: 81.69444444444444%

Test Accuracies: 81.72222222222221%

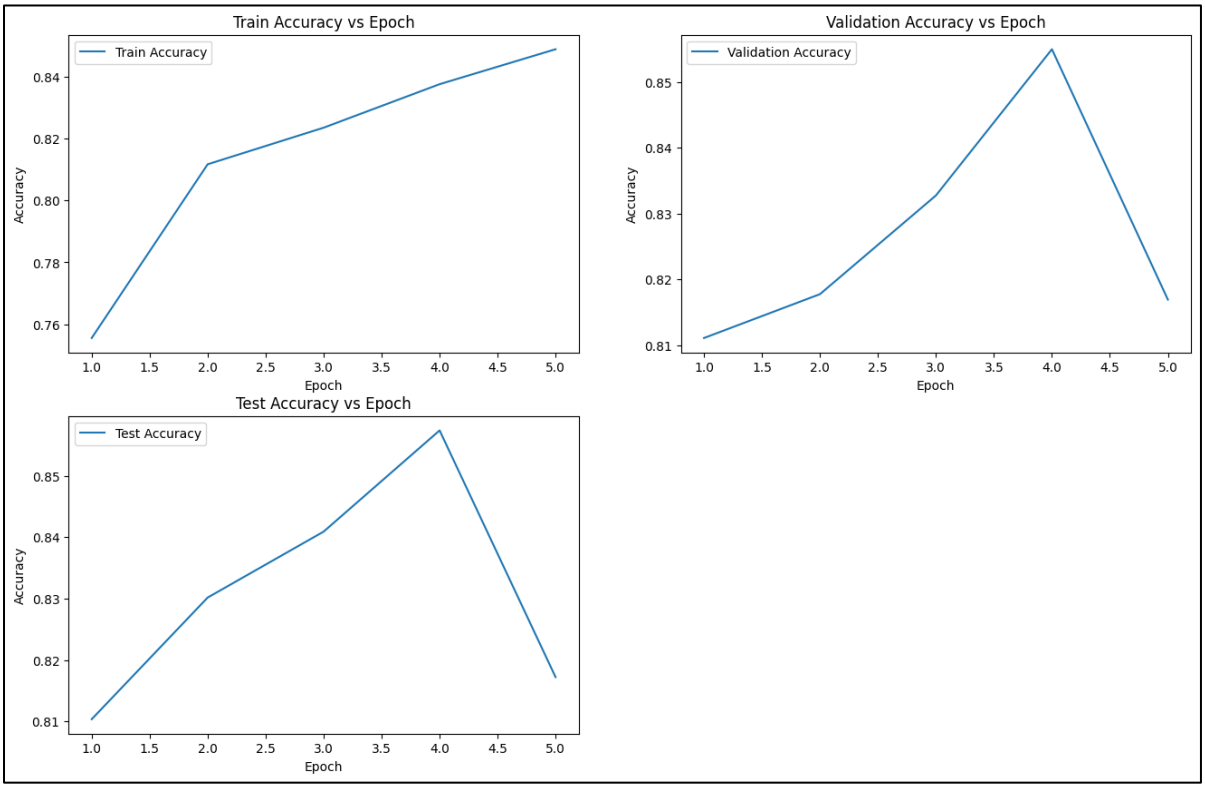
Train Loss: 0.699107855842227

Validation Loss: 0.73182090388404

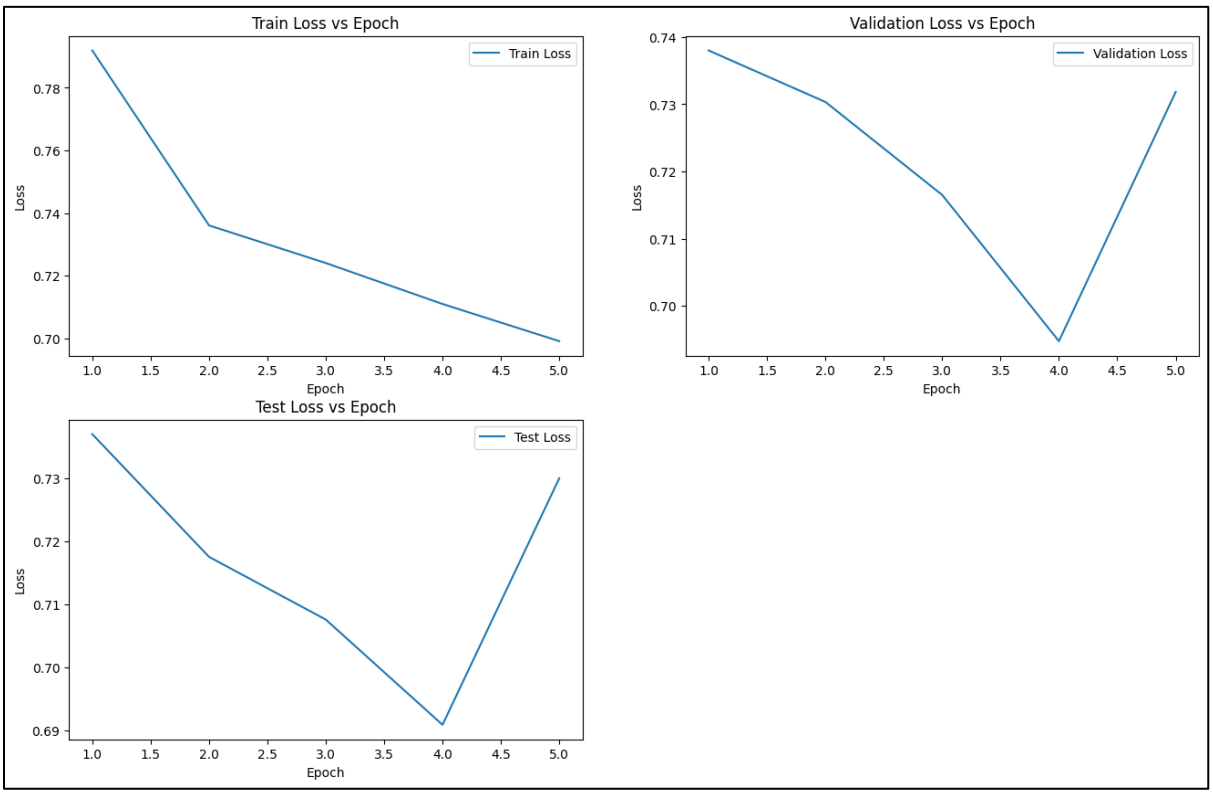
Test Loss: 0.7300489725889983

The graphs showing the above results:

Training, Validation, and Test Accuracy plot:



Training, Validation, and Test Losses plot:



After evaluating the model on test dataset, we have got the following result:

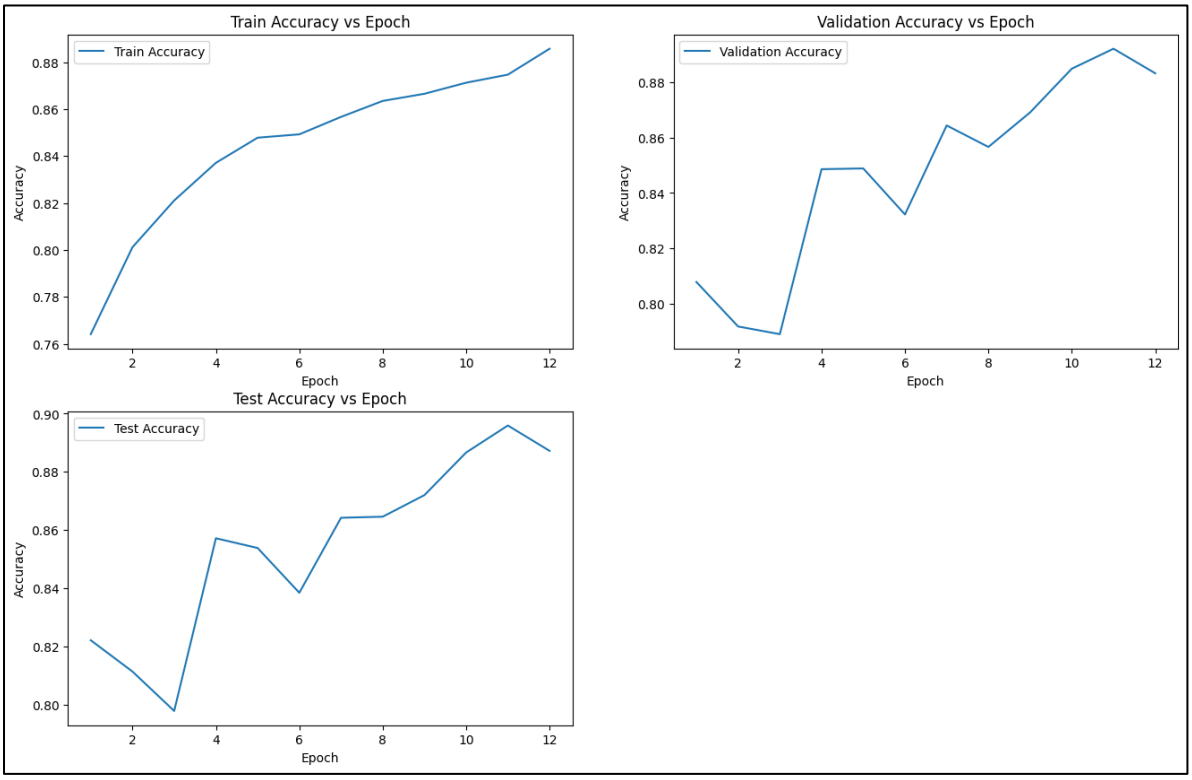
```
Confusion Matrix:
[[1400  291   91]
 [ 114 1555  150]
 [ 225  116 1458]]

Precision: 0.8185
Recall: 0.8172
F1 Score: 0.8172
```

After Applying L2:

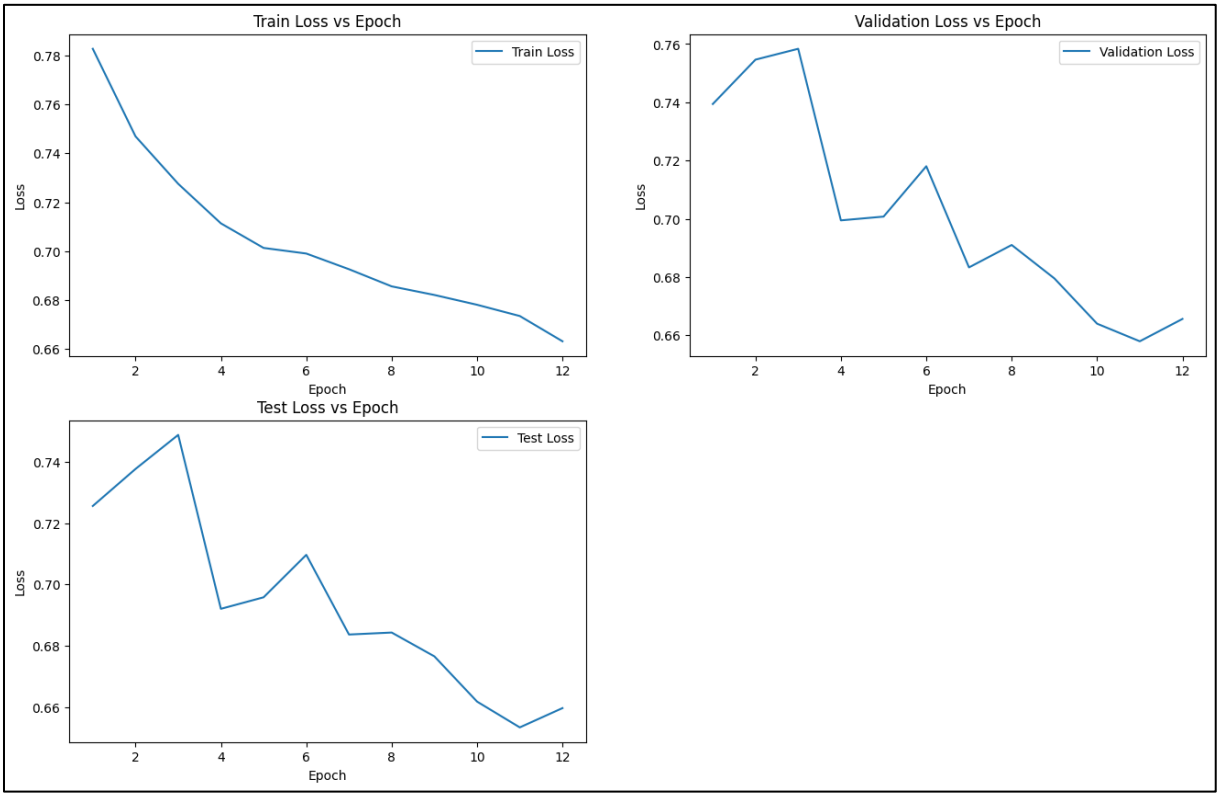
The result obtained are as follows:

```
Confusion Matrix:
[[1480  135  167]
 [ 132 1615   72]
 [  46   57 1696]]

Precision: 0.8876
Recall: 0.8872
F1 Score: 0.8867
```

Training, Validation, and Test Accuracy plot:



Training, Validation, and Test Losses plot:



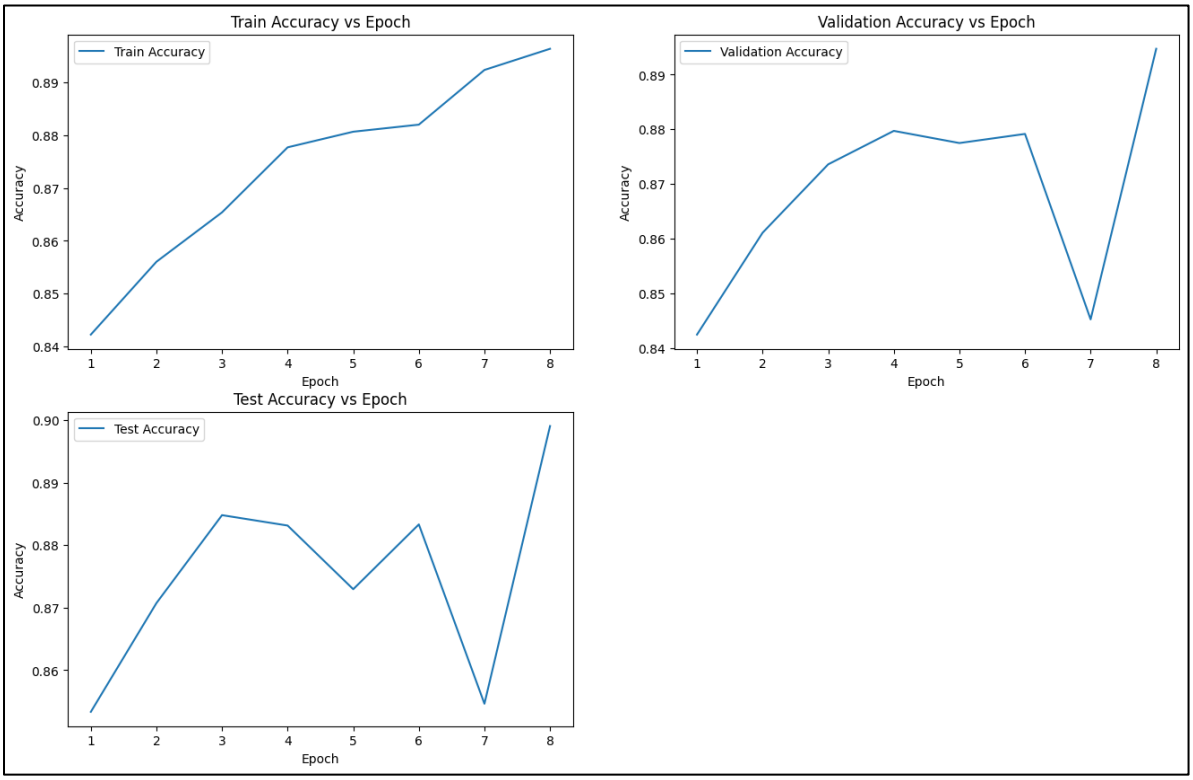
```
Train Accuracies: 88.57619047619048%
Validation Accuracies: 88.33333333333333%
Test Accuracies: 88.72222222222223%

Train Loss: 0.6632020957583473
Validation Loss: 0.6656107680002848
Test Loss: 0.6597307134557653
```

After Applying Batch Normalization:

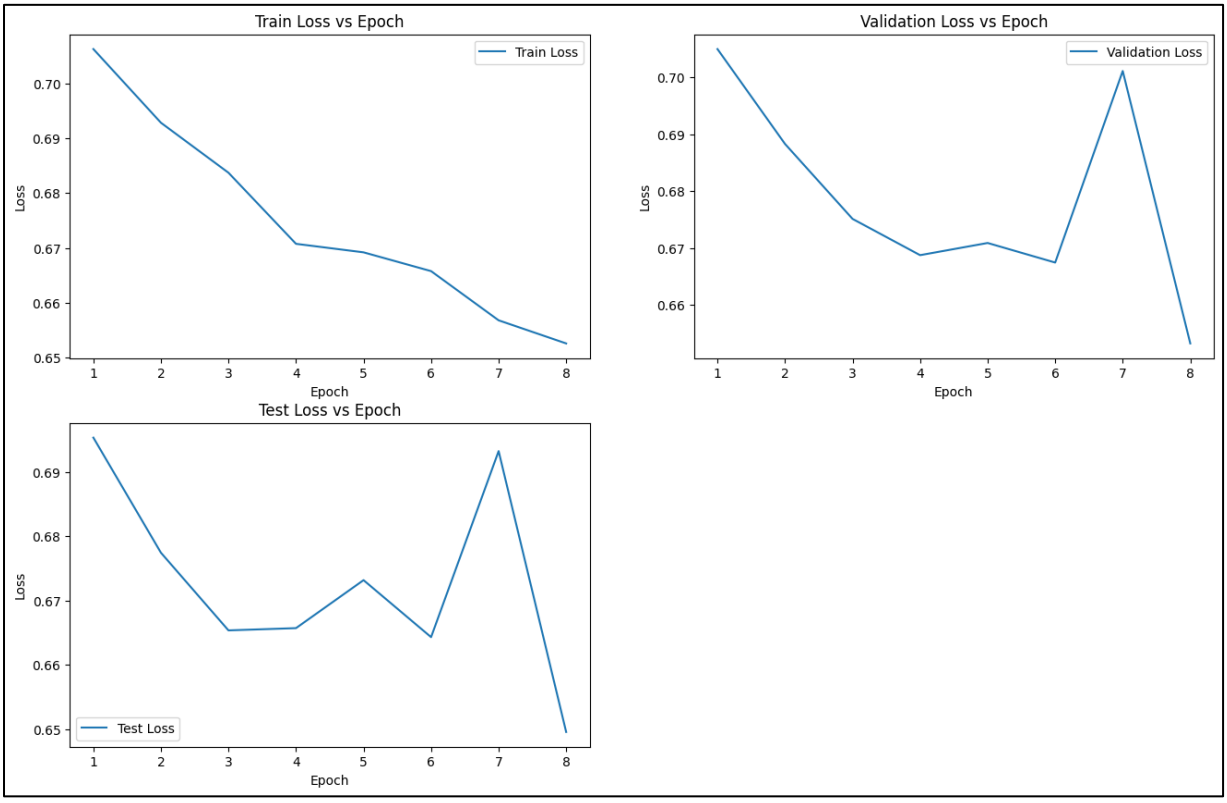
The result obtained are as follows:

```
Confusion Matrix:
[[1582  147   53]
 [ 158 1618   43]
 [  86   58 1655]]
Precision: 0.8998
Recall: 0.8991
F1 Score: 0.8993
```

Training, Validation, and Test Accuracy plot:



Training, Validation, and Test Losses plot:



Train Accuracies: 89.62857142857142%

Validation Accuracies: 89.47222222222221%

Test Accuracies: 89.9074074074074%

Train Loss: 0.6525589999925523

Validation Loss: 0.6532533815171984

Test Loss: 0.6495481295055813