Implementing ResNet Architecture

1. Describe the ResNet architecture you have defined.

⟶ Given below is the architecture of the Resnet18 model which we use:

Basic Block:

The BasicBlock class defines the basic building block of the ResNet architecture. It consists of two convolutional layers with batch normalization and ReLU activation functions.

Each convolutional layer is followed by batch normalization (nn.BatchNorm2d) and a ReLU activation function (nn.ReLU). The stride parameter controls the stride of the first convolutional layer within the block. The shortcut connection helps in maintaining the dimensions of the input and output of the block and helps in learning residual functions.

ResNet-18 Architecture:

The ResNet18 class defines the overall ResNet-18 architecture using the BasicBlock building block. It starts with a convolutional layer (conv1) followed by batch normalization and ReLU activation. It uses four stages, each containing multiple layers of BasicBlocks.

1. The number of blocks in each stage is specified by the num_blocks parameter.
2. The number of output channels for each stage doubles while reducing spatial dimensions by half (except for the first stage where the spatial dimensions remain the same).
3. The adaptive average pooling layer (avgpool) is used to reduce the spatial dimensions of the output to a fixed size before passing it to the fully connected layer.
4. Finally, there is a fully connected layer (fc) that produces the final output logits.

ResNet18_custom Function:

The ResNet18_custom function instantiates a ResNet-18 model with the specified configuration, using BasicBlock as the building block and [2, 2, 2, 2] as the number of blocks for each stage.

2. Describe how the techniques (regularization, dropout, early stopping) have impacted the model's performance.

⟶ Comparing the output of the base model and the best model which uses all the hyperparameter techniques to prevent overfitting is as follows:

1. **Regularization:**

Regularization techniques such as L2 regularization or weight decay are commonly used to prevent overfitting by adding a penalty term to the loss function.

In the base model (without regularization), we see some signs of overfitting. The training accuracy increases across epochs while the validation and test accuracies plateau or slightly decrease. This suggests that the model might be memorizing the training data and failing to generalize well to unseen data.

In the model using regularization, we can observe that the gap between training and validation/test accuracies has reduced. This indicates that regularization has helped in reducing

overfitting by penalizing large weights in the model. Consequently, the model's performance on unseen data (validation and test sets) improves.

2. **Dropout:**

Dropout is a regularization technique that randomly drops a proportion of neurons during training to prevent co-adaptation of feature detectors.
In the model using all techniques, dropout might have contributed to improved generalization by preventing over-reliance on specific features or neurons. This can be observed in stabilizing or improving validation and test accuracies over training epochs.
Dropout helps in creating a more robust model by encouraging the network to learn more redundant representations.

3. **Early Stopping:**

Early stopping is a technique where training is stopped when the performance on a validation set starts to degrade, thereby preventing overfitting.
In the base model, we see that the training accuracy keeps increasing while the validation accuracy stagnates or starts decreasing after a few epochs. This suggests that the model starts to overfit the training data.
In the model using early stopping, the training is stopped after six epochs. This indicates that the model stopped training before it could overfit the training data excessively. As a result, the model generalizes better to unseen data, as evidenced by the stable or improving validation and test accuracies.
In summary, regularization, dropout, and early stopping have collectively contributed to improving the model's performance by reducing overfitting and improving generalization to unseen data.

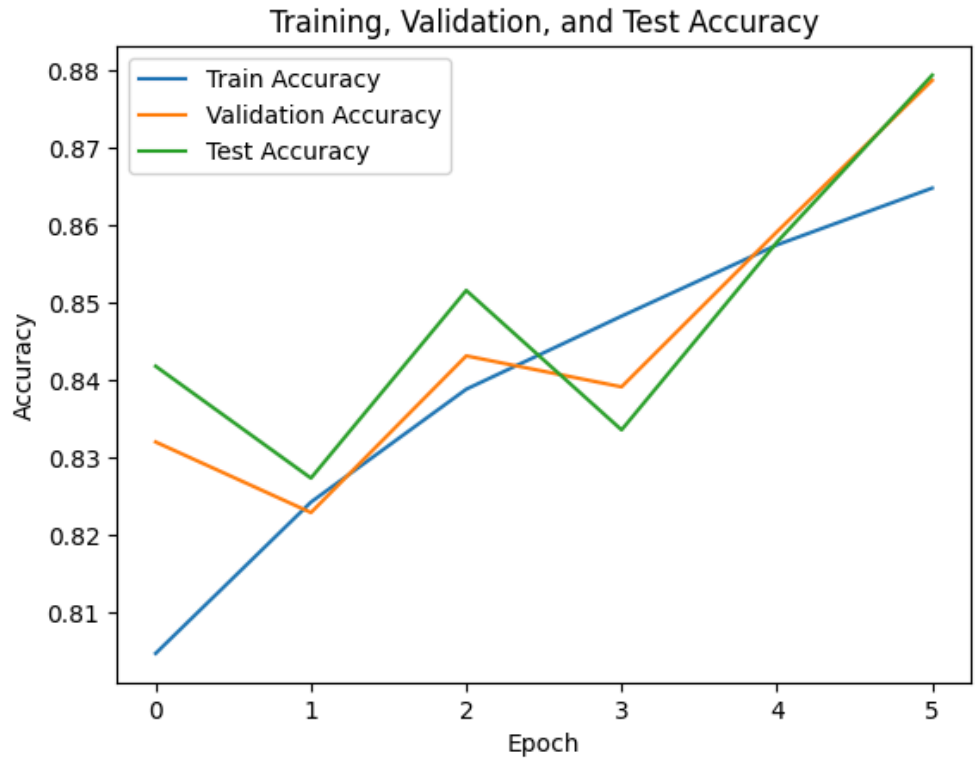
3. Discuss the results and provide relevant graphs:

a. Report training accuracy, training loss, validation accuracy, validation loss, testing accuracy, and testing loss.

→ For best model, the following were the results-
Train Loss: 0.3521
Train Acc: 0.8648,
Val Loss: 0.3312,
Val Acc: 0.8787,
Test Loss: 0.3229,
Test Acc: 0.8793

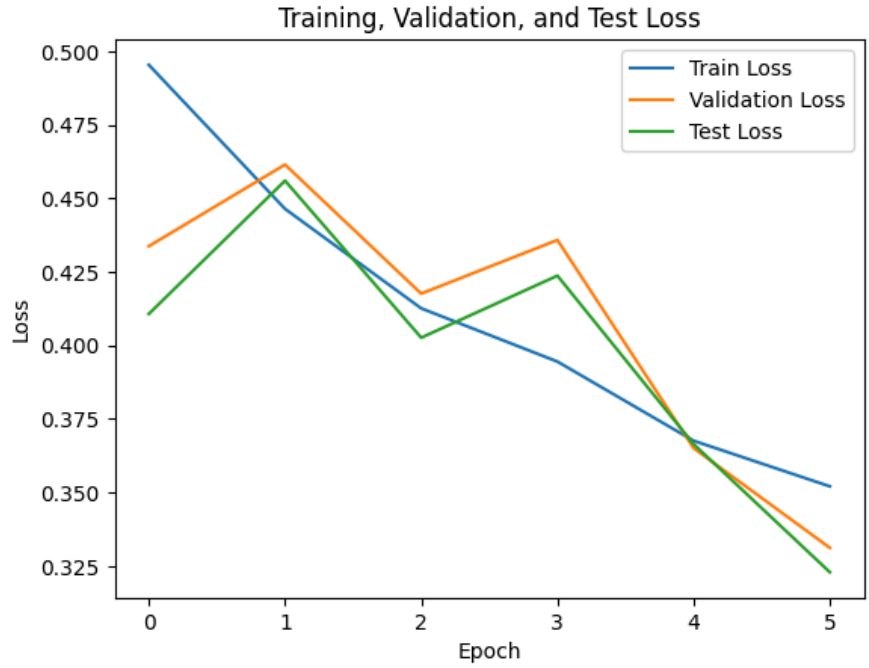
b. Plot the training and validation accuracy over time (epochs).

→



c. Plot the training and validation loss over time (epochs).

→



d. Generate a confusion matrix using the model's predictions on the test set.

→ For best model, below was the confusion matrix -

Confusion Matrix:

[[1375 51 50]
[257 1210 38]
[116 31 1372]]

e. Report any other evaluation metrics used to analyze the model's performance on the test set.

→ For best model, below were the other evaluation metrics -

Precision: 0.8884,
Recall: 0.8793,
F1 Score: 0.8801