Neural Networks and Deep Learning (ECS659P/ECS7026P)

Name: Addalin Lariena Rose Devasahaya Rajan

Student No: 230187746

Neural Network Architecture:

The code implements ResNet architecture called ResNet18 from week 8 ResNet lab.

ResNetBlock:

- It is a residual block in the ResNet architecture
- Each block has 2 convolutional layers with batch normalization along with RELU activation.
- A 3rd optional convolutional layer is used to adjust the dimensions incase the input channel, output channel or stride size differs
- The output is obtained by adding the 2nd convolutional layer with the 3rd, if the 3rd layer is present or to the input incase no adjustments required

ResNetMacroBlock:

- It is a macroblock in the architecture
- It consists of 2 ResNetBlocks

ResNet18:

- Main ResNet architecture and consists of following:
    - Initial convolution layer: takes an input image with 3 channels and applies a 3x3 convolution with 64 output channels.
    - Batch Normalization and ReLU activation
    - Max Pooling Layer: To reduce the spatial dimensions, a 3x3 max pooling layer with stride 2 is applied
    - 4 ResNetMacroblocks with each having output channel increase from one another (64, 128, 256, and 512)
    - Adaptive Average Pooling: to get a fixed-size representation, this layer calculates the average value of each channel over the spatial dimensions.
    - Flattening: The output of the previous layer is flattened to a 1D tensor
    - Fully Connected Layer: the flattened is mapped to the output classes
- Initialization: using Xavier uniform initialization, the weights of convolutional and linear layers are initialized

Difference between the architectures (This ResNet architecture with the basic architecture):

- The intermediate Block has 2 convolutional layers while the basic architecture has many L independent convolutional layers. Moreover, combining multiple convolutional layers with different coefficients is also not present.
- The output block flattens the output tensor and runs it through a single fully connected layer to produce the logits. But is different from the basic architecture, due to the fact that it omits the series of fully connected layers that calculate the logits vector o using the input picture x's average channel values.

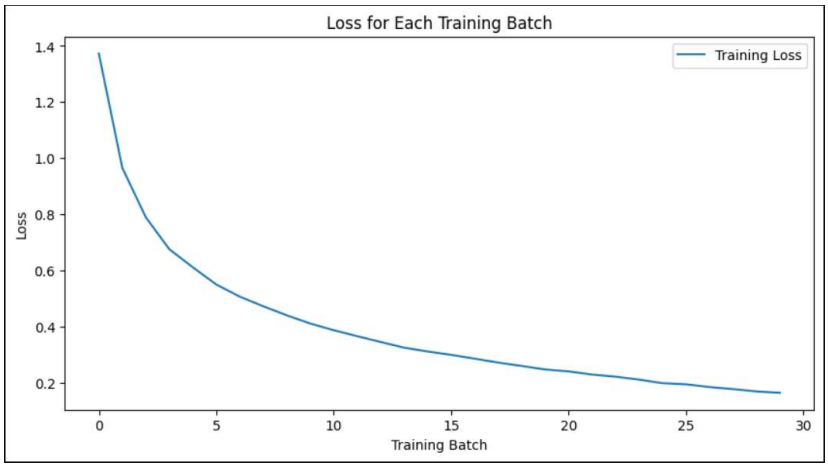
Hyperparameters employed for training:

- Learning rate: The learning rate is 0.02
- Batch size: batch size is 128 for training
- Weight initialization: uses Xavier uniform initialization
- Optimizer: Adam optimizer is used
- Loss function: Cross Entropy Loss
- Learning Rate Scheduler: learning rate scheduler with OneCycleLR
- Number of epochs: 30
- Shuffle: shuffle is set to True
- Num_workers: set to 3
- Data transformations: Uses RandomCrop, RandomHorizontalFlip, ToTensor and Normalize

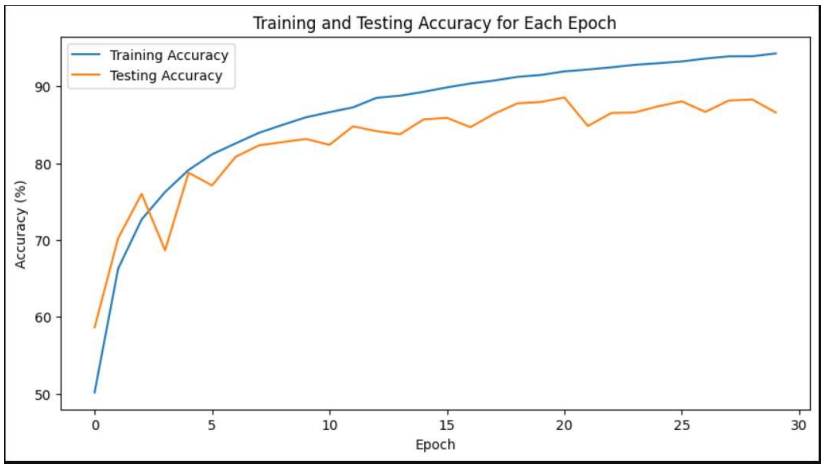
Techniques employed for training:

- Weight initialization: uses Xavier uniform initialization
- Batch normalization
- ReLU activation
- Max pooling
- Adaptive average pooling
- Data Augmentation
- Normalization
- Dataset Splitting
- Data Loading
- GPU Utilization

Loss for each training batch:



Training and Testing Accuracy for Each Epoch:



The highest accuracy obtained is **88.55%**

Improving Initial Implementation:

The original code was taken from Week 8 ResNet lab which was meant for FashionMNIST dataset. It was then implemented for Cifar10 dataset but the accuracy for it was extremely low, to increase its accuracy I took inspiration from a code on Kaggle titled “CIFAR10 ResNet: 90+% accuracy;less than 5 min” by Kamal Das (Das, 2020). Data normalization, Data augmentation, learning rate scheduling and Adam optimizer were the key ideas that I learned from it. Along with these ideas and tuning and experimenting with other hyperparameters I finally achieved the accuracy that I wanted.

References

Das, K. (2020). CIFAR10 ResNet: 90+% accuracy;less than 5 min. Kaggle. Retrieved April 9, 2024, from https://www.kaggle.com/code/kmldas/cifar10-resnet-90-accuracy-less-than-5-min