Name – Samrudhi Bhosale
StudentID – 230854907
Module Identifier - ECS659P

Neural Network & Deep Learning Assignment-

Basic Architecture-

The image initially goes through an Intermediate Block, which we'll call B. In my model, I've designed it to have three of these blocks: B1, B2, and B3. Each Intermediate Block contains three separate convolution layers. After passing through each Intermediate Block, including the final B3, the image then moves through the Output Block, denoted as O, to produce a vector of logits. I have explained the basic architecture below and later what enhancements I have done to increase the accuracy.

Data Loading –

1. First, I have prepared the data to feed into the model by using transforms to tweak our images , that is converting them into tensors and normalizing them. This helps my model understand and learn from these images better.
2. I have used two datasets: one for training and other for testing.
3. We load these images from CIFAR10 dataset with images of planes, cars, birds and more. Then I have split the images into batches because it's easier for the model to learn bit by bit **batch_size = 64**

Intermediate Block-

1. The intermediate block takes number of input channels , output channels and num of convolution layers.
2. Each block contributes to the hierarchical feature extraction process, enhancing the classifier's capacity to understand the patterns within the CIFAR10 dataset.
3. Then depending upon the number of convolution layers we need in a block, we loop through and create these layers using **ModuleList** class.
4. Then we create the fully connected layer using **Linear** class by passing the initial input channels and number of convolution layers.
5. The forward method of the Intermediate Block is responsible for executing the computations necessary for feature transformation.
6. Initially, the mean of each channel is computed using the **mean()** function, with parameters including the batch size and number of input channels.
7. The **sigmoid** activation function is applied to compute vector a in basic architecture.
8. This vector 'a' is then broadcasted across all convolutional layers to compute the transformed feature representation denoted as X' (Xprime).

Output Block-

1. The Output Block class represents the final stage of the CIFAR10 Classifier model, responsible for generating the output based on the transformed features extracted by the preceding Intermediate Blocks.

2. The constructor initializes the OutputBlock by taking two parameters: in_channels and num_classes.
3. The **forward()** executes the forward pass through the OutputBlock, by taking input tensor x.
4. The mean of each channel across spatial dimensions is computed using **torch.mean()** with dim=[2, 3]. This operation results in a tensor with shape **[batch_size, in_channels],** where each element represents the mean value of a channel for a particular sample in the batch.
5. The mean tensor is then passed through a fully connected layer (self.fc) to perform the classification. The linear transformation performed by the fully connected layer maps the input features to the output classes, producing logits for each class, which represents the predicted scores for each class.

CIFAR10Classifier Class –

1. The CIFAR10Classifier specifies the number of intermediate blocks we need to train the model.
2. We have used **CrossEntropyLoss**() class to set a loss function and to keep track of errors.
3. It also calculates the gradient of the loss with respect to the model's parameters, which is essential for training the model using backpropagation.
4. Stochastic Gradient Descent optimizer is used in the basic architecture to improve the model based on errors.
5. The model parameters are optimized through iterative training loops, where batches of data are processed, and the model parameters are updated to minimize the defined loss function.

Results-

1. The accuracy was very less. Training accuracy was about 40% and that of testing was 38%.

New architecture - Enhancements

To increase the accuracy of the model I have made below enhancements in the basic architecture.

Data Augmentation –

1. Data Augmentation technique is used to increase the diversity of training data by applying random transformations.
2. I have executed random horizontal flips and rotations up to 10 degrees to the input images, followed by converting them to tensors and normalizing their pixel values **RandomHorizontalFlip()** , **RandomRotation(10)**.

Normalization -

1. I have used Batch Normalisation **nn.BatchNorm2d(out_channels)** to adjust and make the input images consistent.
2. It stabilizes the learning process by normalizing the input of each layer.

Activation function –

1. In the new architecture I have used ReLU activation function instead of sigmoid **nn.ReLU(inplace=True)**

2. Softmax is applied to the output of the fully connected layer to compute coefficients for combining convolutional layer outputs in each intermediate block

Regularization –

1. I have applied regularization techniques like Dropout **nn.Dropout(p=0.5)** can prevent the model from overfitting on the training data, thus improving performance on unseen data.

Hyperparameter Tunning-

1. In the basic architecture I tried to keep the learning rate **lr** = 0.001 it gave good result but to enhance the accuracy more I tried with learning rate = 0.0005 but this did not give any effective results. So, in my new architecture I have used learning rate **lr = 0.001**
2. In the new architecture the number of intermediate blocks are **num_blocks = 3**
3. The number of convolutional layers within each intermediate blocks are **num_convs_per_block = 3**
4. Regularization parameter that penalizes large weights in the model are **weight_decay** = 0.01
5. Maximum number of epochs is the number of times the dataset is passed forward and backward through the model epochs = 40

Optimizer –

1. In Basic architecture I have used SGD optimizer which did not give good results.
2. In the new architecture I have used adaptive learning rate optimizer like Adam to decrease over time and can yield better results.
3. **optimizer = optim.Adam(model.parameters(), lr=0.001)**

Accuracy- From the New Enhanced Architecture, I have achieved **82.05 % testing accuracy** and 85% training accuracy.

Plots –