CIFAR-10 Classification

Introduction:

CIFAR-10 is a benchmark dataset for image classification tasks. The dataset contains 60,000 32x32 color images in 10 different classes, with 6,000 images per class. The objective of this project is to build a convolutional neural network (CNN) to classify these images into one of the 10 classes. The model will be built on the training set and evaluated on the test set.

Data Preparation:

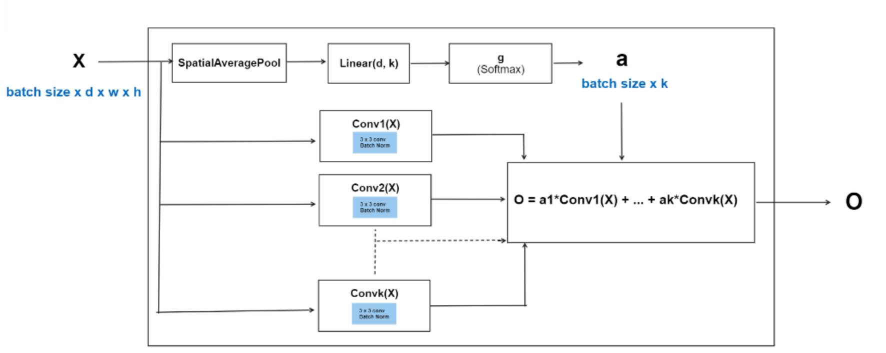
The CIFAR-10 dataset is readily available in PyTorch by using the torchvision.dataset. The data is divided into training and test sets, with 50,000 and 10,000 images respectively. Before feeding the images into the model, these images are transformed using the transforms module. The transform.Compose() function is used to apply a series of transformations to the images. These transformations include converting the images to PyTorch tensors and normalizing them.

Methodology:

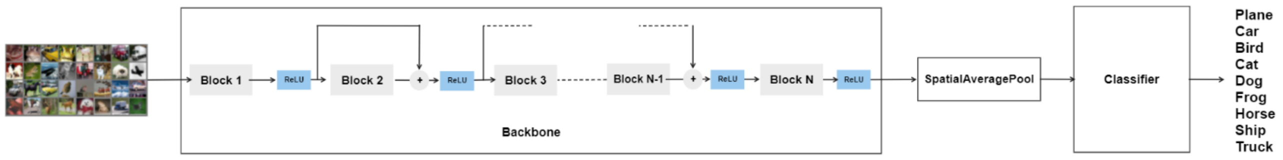
The following diagram explains the workflow:

Model: The model mainly consists of the Backbone and classifier. The model architecture is based on the BackboneBlock class, which combines the outputs of multiple convolutional operations to produce a single output. The model is trained for 30 epochs and evaluated on the test set. The accuracy achieved by the model is a metric to assess the performance of the model and can be further improved by tuning the hyperparameters of the model.

The architecture of a Block inside the backbone is defined as the class BackboneBlock in Jupyter Notebook.



The architecture of the model is defined as class Net in Jupyter Notebook.



The "BackboneBlock" and "Net" classes make up the neural network's architecture. One block of the neural network's "BackboneBlock" defines its layers. Each block has a linear layer that applies the softmax activation function to join a spatial average pool to a vector "a". The weights from "a" are used to stack the outputs of the convolutional layers, which are then combined into a single output. A neural network with "N" blocks is defined by "Net" using "BackboneBlock". An RGB image serves as the network's input and is processed by a succession of "N" blocks, each containing "k" convolutional layers. Each block's output is added to the output of the one before it using a ReLU activation function except the first and Nth block. The final output is routed through a spatial average pool and a classifier that contains 1 fully connected layer with 32 nodes before the output layer.

Model Training:

The model is trained on the training set for 30 epochs with a batch size of 32. During each epoch, the model is updated using the backpropagation algorithm with the Adam optimizer. The loss function used in this project is the cross-entropy loss. After training, the model is evaluated on the test set using the testloader.

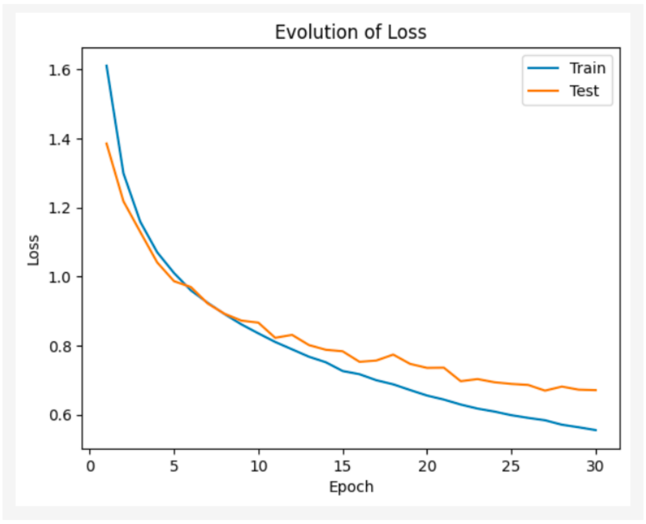
The hyperparameters used to train the CIFAR-10 classification model include the batch_size (size of mini batch), N (Number of blocks in the BackBone), k (Number of Convolutional layers in each block), num_filters (output channels in convolutional layers), epochs (Number of epochs), and LR (learning rate).

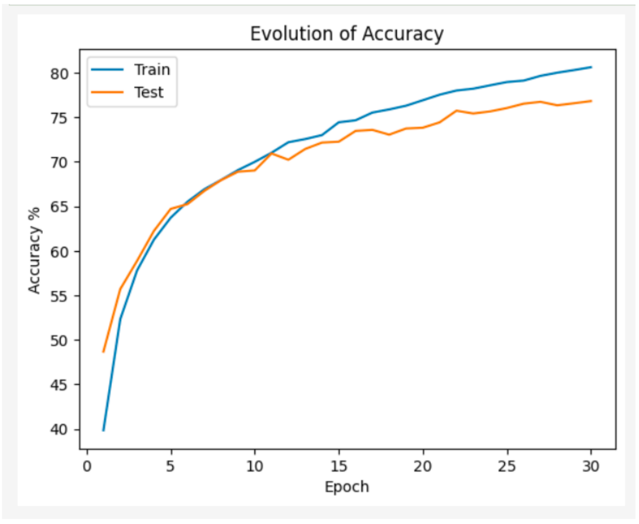
The best parameters are as follows:

The batch size is set to 32, which is the number of images that will be processed at once before the model updates its parameters. N is set to 16, which represents the number of blocks in the backbone. k is set to 3, which represents the number of convolutional layers in each block inside the backbone. The number of filters is set to 64, which is the number of feature maps in the convolutional layers. The model is trained for 30 epochs, which means that the entire training set is used 30 times. The learning rate is set to 0.0005, which determines the step size at each iteration of the optimization algorithm.

Model Evaluation:

Evaluation Plots: Curves for Evolution of Loss and Accuracy





The provided plots are used to analyze the performance of the trained model during the training and testing phases. By examining the plots, we can observe that the model's performance improves over time as it learns to make better predictions.

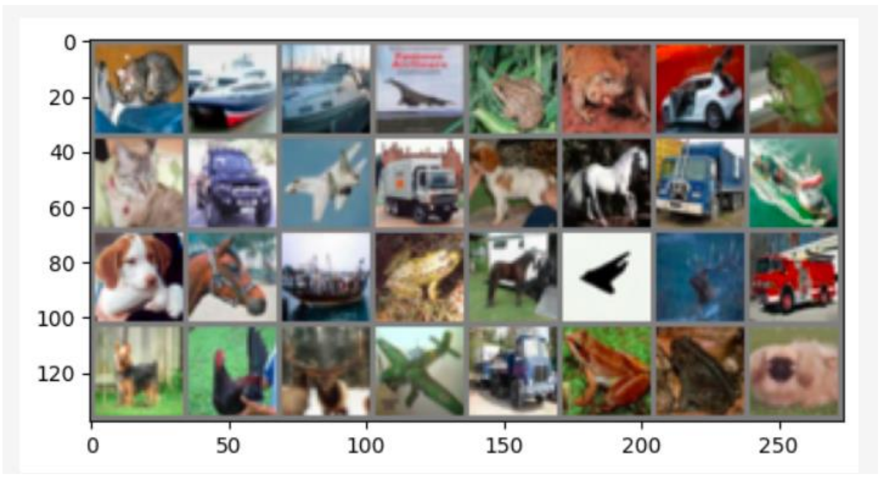
Evolution of Loss: The plot of the evolution of loss shows a significant decrease in loss during the first 5 epochs. After the 5th epoch, the training and test loss continue to decrease, but at a slower rate. This indicates that the model is gradually learning to make better predictions.

Evolution of Accuracy: The plot of the evolution of accuracy shows a sharp increase in accuracy during the first 5 epochs. Between the 5th and 26th epochs, the accuracy increases gradually. After the 26th epoch, the test accuracy remains almost constant. This suggests that the model has learned to make accurate predictions on the test data, and further training is unlikely to significantly improve its performance.

Analysis Of the Predictions:

The accuracy of the network on the 10000 test images is **76.8 %.**

Let's look at predictions of some images from the test set.

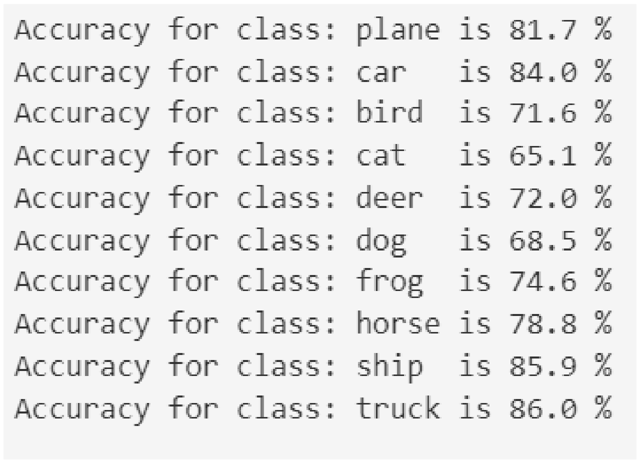


GroundTruth: cat, ship, ship, plane, frog, frog, car, frog, cat, car, plane, truck, dog, horse, truck, ship, dog, horse, ship, frog, horse, plane, deer, truck, dog, bird, deer, plane, truck, frog, frog, dog

Predicted Classes: cat, plane, ship, plane, bird, cat, car, deer, cat, plane, plane, truck, dog, horse, truck, frog, dog, horse, ship, frog, horse, plane, ship, truck, deer, cat, bird, plane, truck, deer, frog, dog

Per Class Accuracy:

The prediction can be analysed by analyzing the accuracy of each class.



```
Accuracy for class: plane is 81.7 %
Accuracy for class: car   is 84.0 %
Accuracy for class: bird  is 71.6 %
Accuracy for class: cat   is 65.1 %
Accuracy for class: deer  is 72.0 %
Accuracy for class: dog   is 68.5 %
Accuracy for class: frog  is 74.6 %
Accuracy for class: horse is 78.8 %
Accuracy for class: ship  is 85.9 %
Accuracy for class: truck is 86.0 %
```

The reported accuracies for each class in the CIFAR-10 dataset offer insights into the model's ability to distinguish between different types of images. The highest accuracy is achieved for the 'truck' class, with an accuracy of 86%, followed closely by 'ship' and 'car' with accuracies of 85.9% and 84.0%, respectively. However, some classes, such as 'cat' and 'dog,' have lower accuracies, indicating that the model may struggle to accurately classify certain types of images. From the accuracies of the 'plane', 'horse', 'deer', 'bird', and 'frog', we can observe that the performance of the model for these classes is better than the cat/dog class.