This code demonstrates a basic approach to training a Convolutional Neural Network (CNN) on the CIFAR-10 dataset using TensorFlow and Keras. It covers two key aspects of training: building a baseline model and then enhancing the model's performance using data augmentation.

1. Model Training (Baseline) Dataset:

The CIFAR-10 dataset is loaded using the tf.keras.datasets module. It consists of 60,000 32x32 color images across 10 classes, with 50,000 images for training and 10,000 for testing.

Preprocessing:

Both the training and testing images are normalized by scaling the pixel values to the range [0, 1]. This step helps the model to converge faster.

CNN Architecture:

The model consists of three convolutional layers (Conv2D), each followed by a pooling layer (MaxPooling2D) to reduce the spatial dimensions.After flattening the output of the convolutional layers, it passes through a fully connected layer (Dense) with 64 units.The final layer is a dense output layer with 10 units (one for each class), without activation, as the model uses logits and applies softmax during the loss calculation.

Compilation:

The Adam optimizer is used with default settings.The loss function is SparseCategoricalCrossentropy, which is suited for integer-labeled classes.Accuracy is used as the evaluation metric.

Training:

The model is trained for 5 epochs using the train\_images dataset and validated on test\_images.

Result:

The baseline accuracy is printed, showing the validation accuracy after 5 epochs.

2. Data Augmentation Purpose:

Data augmentation is applied to improve the model's generalization by artificially increasing the diversity of the training data.

Technique:

The ImageDataGenerator is used to apply real-time data augmentation. The augmentation includes:

Small random rotations (rotation\_range=10),Shifting the image horizontally and vertically (width\_shift\_range and height\_shift\_range=0.1),

Random horizontal flips (horizontal\_flip=True),

Zooming in and out (zoom\_range=0.1).

Model Adjustment:

The learning rate of the optimizer is reduced to 0.0001 to make the model more stable during training on augmented data.The same model is recompiled and retrained using the augmented images for 5 more epochs.

Training with Augmentation:

The model is retrained using augmented data generated by the datagen.flow() function, which applies augmentation on the fly. The model is still validated on the original test\_images set.

Result:

The new validation accuracy after applying data augmentation is printed, allowing comparison with the baseline model.

Conclusion:

The code demonstrates how to apply