**📌 Stage A: Loading and Preprocessing the Image Data**

# -------------------- Step 1: Import Required Libraries --------------------

import tensorflow as tf # Main deep learning library

from tensorflow.keras.datasets import cifar10 # CIFAR-10 dataset (images of 10 classes)

from tensorflow.keras.models import Sequential # For creating sequential model

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

import matplotlib.pyplot as plt # For plotting graphs

# -------------------- Step 2: Load CIFAR-10 Dataset ------------------------

(x\_train, y\_train), (x\_test, y\_test) = cifar10.load\_data() # 50,000 train + 10,000 test images

# -------------------- Step 3: Normalize the Images -------------------------

x\_train = x\_train / 255.0 # Pixel values from [0,255] → [0,1]

x\_test = x\_test / 255.0

print("Training data shape:", x\_train.shape)

print("Testing data shape:", x\_test.shape)

**✅ 📌 Stage B: Defining the Model’s Architecture (Convolutional Neural Network)**

# -------------------- Step 4: Build CNN Model --------------------

model = Sequential([

# 1st Convolution Layer + Max Pooling

Conv2D(32, (3,3), activation='relu', input\_shape=(32,32,3)),

MaxPooling2D((2,2)),

# 2nd Convolution Layer + Max Pooling

Conv2D(64, (3,3), activation='relu'),

MaxPooling2D((2,2)),

# Flatten output to feed into Dense Layers

Flatten(),

# Fully Connected Layer

Dense(128, activation='relu'),

# Dropout to reduce overfitting

Dropout(0.5),

# Output Layer - 10 classes

Dense(10, activation='softmax')

])

# Display model summary (structure)

model.summary()

**✅ 📌 Stage C: Compile and Train the Model**

# -------------------- Step 5: Compile the Model --------------------

model.compile(optimizer='adam', # Optimizer: Adam (smarter SGD)

loss='sparse\_categorical\_crossentropy', # Suitable for integer labels

metrics=['accuracy']) # Track training accuracy

# -------------------- Step 6: Train the Model --------------------

history = model.fit(x\_train, y\_train,

epochs=10,

batch\_size=64,

validation\_data=(x\_test, y\_test))

**✅ 📌 Stage D: Estimating the Model’s Performance (Accuracy & Loss Graph)**

# -------------------- Step 7: Evaluate on Test Data --------------------

test\_loss, test\_acc = model.evaluate(x\_test, y\_test)

print("Test Loss:", test\_loss)

print("Test Accuracy:", test\_acc)

# -------------------- Step 8: Plot Accuracy & Loss --------------------

# Plot Accuracy

plt.plot(history.history['accuracy'], label='Training Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.title("Model Accuracy")

plt.xlabel("Epochs")

plt.ylabel("Accuracy")

plt.legend()

plt.show()

# Plot Loss

plt.plot(history.history['loss'], label='Training Loss')

plt.plot(history.history['val\_loss'], label='Validation Loss')

plt.title("Model Loss")

plt.xlabel("Epochs")

plt.ylabel("Loss")

plt.legend()

plt.show()

**✅ 📘 Viva Questions & Answers for Practical 3**

| **Question** | **Answer** |
| --- | --- |
| What is CIFAR-10 dataset? | A dataset of 60,000 color images (32×32) in 10 classes like airplane, car, bird, cat, etc. |
| Why do we normalize images? | To bring pixel values to range [0,1], which helps model learn faster and improves performance. |
| What is a Convolutional Layer? | A layer that detects features like edges, shapes from images by applying filters. |
| What is MaxPooling? | It reduces image size by selecting only the maximum value from a group of pixels – decreases computation & avoids overfitting. |
| Why do we use Dropout? | Dropout randomly disables some neurons during training to prevent overfitting. |
| Why softmax in output layer? | Softmax converts outputs into probability values for multiple classes. |
| What is the difference between training and validation accuracy? | Training accuracy is performance on training data, validation accuracy is performance on unseen test data. |
| What is Adam optimizer? | It is an advanced optimizer which combines RMSprop and Momentum, faster than basic SGD. |