**# Load CIFAR-10 dataset**

(x\_train, y\_train), (x\_test, y\_test) = keras.datasets.cifar10.load\_data()

**# Normalize pixel values (0-255 → 0-1)**

X\_train, x\_test = x\_train / 255.0, x\_test / 255.0

**# Build CNN Model**

Model = keras.Sequential([

Keras.layers.Conv2D(32, (3,3), activation=’relu’, input\_shape=(32,32,3)),

Keras.layers.MaxPooling2D((2,2)),

Keras.layers.Conv2D(64, (3,3), activation=’relu’),

Keras.layers.MaxPooling2D((2,2)),

Keras.layers.Conv2D(64, (3,3), activation=’relu’),

Keras.layers.Flatten(),

Keras.layers.Dense(64, activation=’relu’),

Keras.layers.Dense(10, activation=’softmax’)

])

**# Compile the Model**

Model.compile(optimizer=’adam’, loss=’sparse\_categorical\_crossentropy’, metrics=[‘accuracy’])

**# Train the Model**

Model.fit(x\_train, y\_train, epochs=10, validation\_data=(x\_test, y\_test))

**# Evaluate the Model**

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

Print(f”Test Accuracy: {test\_acc:.2f}”)

**# Test Model on single image**

**# Load a sample image**

Sample\_image = x\_test[0]

Sample\_label = y\_test[0]

**# Predict the class**

Prediction = model.predict(np.expand\_dims(sample\_image, axis=0))

Predicted\_class = np.argmax(prediction)

**# Display the result**

Plt.imshow(sample\_image)

Plt.title(f”Predicted: {class\_names[predicted\_class]}, Actual: {class\_names[sample\_label[0]]}”)

Plt.show()

**#Build Model**

Import tensorflow as tf

From tensorflow import keras

**# Build CNN Model**

Model = keras.Sequential([

**# First Convolutional Layer**

Keras.layers.Conv2D(32, (3, 3), activation=’relu’, input\_shape=(32, 32, 3)),

Keras.layers.MaxPooling2D((2, 2)),

**# Second Convolutional Layer**

Keras.layers.Conv2D(64, (3, 3), activation=’relu’),

Keras.layers.MaxPooling2D((2, 2)),

**# Third Convolutional Layer**

Keras.layers.Conv2D(64, (3, 3), activation=’relu’),

**# Flatten Layer**

Keras.layers.Flatten(),

**# Fully Connected Layer**

Keras.layers.Dense(64, activation=’relu’),

**# Output Layer**

Keras.layers.Dense(10, activation=’softmax’) # 10 output classes

])

**# Compile the Model**

Model.compile(optimizer=’adam’,

Loss=’sparse\_categorical\_crossentropy’,

Metrics=[‘accuracy’])

**# Print Model Summary**

Model.summary()

**#Code to display sample images**

Import matplotlib.pyplot as plt

**# Class names in CIFAR-10**

Class\_names = [‘Airplane’, ‘Automobile’, ‘Bird’, ‘Cat’, ‘Deer’, ‘Dog’, ‘Frog’, ‘Horse’, ‘Ship’, ‘Truck’]

**# Display sample images**

Plt.figure(figsize=(10, 5))

For I in range(10):

Plt.subplot(2, 5, I + 1)

Plt.xticks([]) # Remove x-axis ticks

Plt.yticks([]) # Remove y-axis ticks

Plt.imshow(x\_train[i]) # Display the image

Plt.xlabel(class\_names[y\_train[i][0]]) # Display the label

Plt.show()

**#Combined code for building model and displaying sample images**

Import tensorflow as tf

From tensorflow import keras

Import numpy as np

Import matplotlib.pyplot as plt

**# Load CIFAR-10 dataset**

(x\_train, y\_train), (x\_test, y\_test) = keras.datasets.cifar10.load\_data()

**# Normalize pixel values (0-255 → 0-1)**

X\_train, x\_test = x\_train / 255.0, x\_test / 255.0

**# Class names in CIFAR-10**

Class\_names = [‘Airplane’, ‘Automobile’, ‘Bird’, ‘Cat’, ‘Deer’, ‘Dog’, ‘Frog’, ‘Horse’, ‘Ship’, ‘Truck’]

**# Display sample images**

Plt.figure(figsize=(10, 5))

For I in range(10):

Plt.subplot(2, 5, I + 1)

Plt.xticks([])

Plt.yticks([])

Plt.imshow(x\_train[i])

Plt.xlabel(class\_names[y\_train[i][0]])

Plt.show()

**# Build CNN Model**

Model = keras.Sequential([

Keras.layers.Conv2D(32, (3, 3), activation=’relu’, input\_shape=(32, 32, 3)),

Keras.layers.MaxPooling2D((2, 2)),

Keras.layers.Conv2D(64, (3, 3), activation=’relu’),

Keras.layers.MaxPooling2D((2, 2)),

Keras.layers.Conv2D(64, (3, 3), activation=’relu’),

Keras.layers.Flatten(),

Keras.layers.Dense(64, activation=’relu’),

Keras.layers.Dense(10, activation=’softmax’)

])

**# Compile the Model**

Model.compile(optimizer=’adam’,

Loss=’sparse\_categorical\_crossentropy’,

Metrics=[‘accuracy’])

**# Print Model Summary**

Model.summary()