Image Classification using the CIFAR-10

Problem Understanding:

The goal of the project is to build and evaluate a Convolutional Neural Network (CNN) for image classification using the CIFAR-10 dataset. CIFAR-10 consists of 60,000 color images in 10 classes, including airplanes, automobiles, birds, and more. The challenge is to accurately classify unseen test images by training the model on labeled training data and ensuring generalization.

Code:

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,
Dropout, BatchNormalization
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
import os
import cv2

class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog',
'frog', 'horse', 'ship', 'truck']

(x_train, y_train), (x_test, y_test) = cifar10.load_data()
x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)

data_gen = tf.keras.preprocessing.image.ImageDataGenerator(
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
    zoom_range=0.1
)
data_gen.fit(x_train)

model = Sequential([
```

```
    Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
    BatchNormalization(),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    BatchNormalization(),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    BatchNormalization(),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(10, activation='softmax')
])

model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
early_stopping = EarlyStopping(monitor='val_loss', patience=5,
restore_best_weights=True)
lr_reduction = ReduceLROnPlateau(monitor='val_loss', factor=0.5,
patience=3, verbose=1)

history = model.fit(
    data_gen.flow(x_train, y_train, batch_size=64),
    epochs=20,
    validation_data=(x_test, y_test),
    callbacks=[early_stopping, lr_reduction]
)

test_loss, test_accuracy = model.evaluate(x_test, y_test, verbose=2)
print(f"Test Accuracy: {test_accuracy * 100:.2f}%")

plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Loss')
plt.xlabel('Epoch')
```

```
plt.ylabel('Loss')
plt.legend()
plt.show()

y_pred = model.predict(x_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = np.argmax(y_test, axis=1)

conf_matrix = confusion_matrix(y_true, y_pred_classes)
disp = ConfusionMatrixDisplay(confusion_matrix=conf_matrix,
display_labels=class_names)
disp.plot(cmap=plt.cm.Blues)
plt.title("Confusion Matrix")
plt.show()

class_wise_accuracy = (conf_matrix.diagonal() / conf_matrix.sum(axis=1)) *
100
for i, acc in enumerate(class_wise_accuracy):
    print(f"Class {class_names[i]} Accuracy: {acc:.2f}%")

def process_and_predict_image(image_path):
    try:
        img = cv2.imread(image_path)
        if img is None:
            raise ValueError("Image cannot be read. Check the file format
or path.")
        img_resized = cv2.resize(img, (32, 32))
        img_normalized = img_resized.astype('float32') / 255.0
        img_normalized = np.expand_dims(img_normalized, axis=0)
        prediction = model.predict(img_normalized)
        predicted_class = prediction.argmax()
        predicted_class_name = class_names[predicted_class]
        print(f'Predicted class index: {predicted_class}')
        print(f'Predicted class name: {predicted_class_name}')
        plt.imshow(cv2.cvtColor(img_resized, cv2.COLOR_BGR2RGB))
        plt.title(f"Predicted: {predicted_class_name}
({predicted_class})")
        plt.axis('off')
        plt.show()
    except Exception as e:
        print(f"Error processing the image: {e}")

def select_and_predict_image():
    image_path = input("Please enter the full path of the image you want
to predict: ")
```

```
    if os.path.exists(image_path):
        process_and_predict_image(image_path)
    else:
        print("Invalid file path. Please try again.")

select_and_predict_image()
```

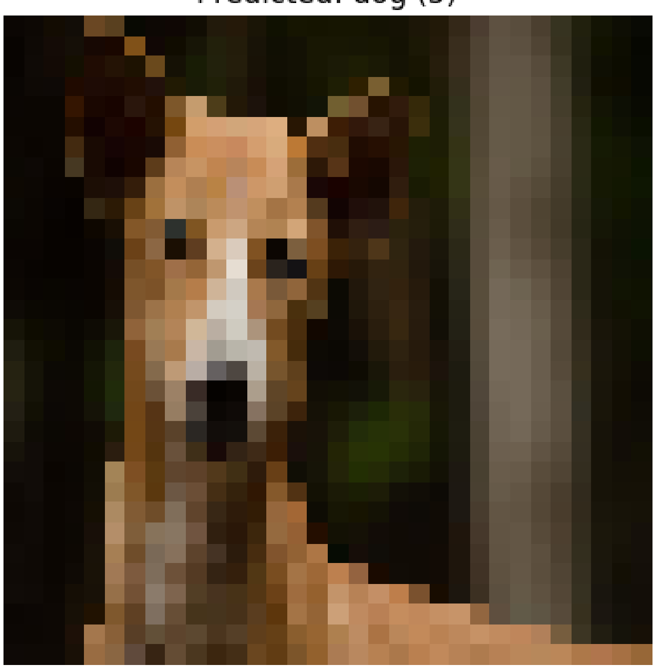
Model Design:

The model is a Convolutional Neural Network (CNN) with three convolutional blocks using increasing filters (32, 64, 128). Each block includes **Batch Normalization**, **ReLU activation**, and **Max Pooling** to extract features and reduce spatial dimensions. The flattened output is passed through a dense layer with **128 neurons** and **Dropout** for regularization, followed by a softmax-activated output layer for class probabilities. Training leverages **Adam optimizer**, **categorical cross-entropy**, and callbacks like **EarlyStopping** and **ReduceLROnPlateau** for efficient learning and overfitting control.

Result:

```
Please enter the full path of the image you want to predict: /content/stray_dog_513872.jpg
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
Predicted class index: 5
Predicted class name: dog
```

Predicted: dog (5)



Conclusion:

This CNN effectively classifies CIFAR-10 images with competitive accuracy, demonstrating the power of convolutional layers for feature extraction. While the model performs well overall, certain classes require refinement to minimize confusion. Future improvements could include exploring advanced architectures (e.g., ResNet) or fine-tuning hyperparameters for better performance. This work underscores the importance of preprocessing, model design, and evaluation in developing reliable machine learning solutions.