Capstone Project: CIFAR-10 Image Classification with CNN + Streamlit App

Deep Learning (TensorFlow/Keras) — Google Colab Ready

1. Project Overview

This project builds, trains, and evaluates a Convolutional Neural Network (CNN) on the CIFAR-10 dataset using TensorFlow/Keras. The trained model is then deployed through a Streamlit web application that allows users to upload an image and receive a predicted class label.

2. Objectives

• Load and preprocess the CIFAR-10 dataset
• Build a CNN using TensorFlow/Keras
• Train the model and track learning curves
• Evaluate performance using accuracy, classification report, and confusion matrix
• Save the model and deploy it in a Streamlit app inside Google Colab using ngrok

3. Dataset

CIFAR-10 contains 60,000 RGB images (32×32) across 10 classes. It includes 50,000 training images and 10,000 test images.

Classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck

4. Tools & Frameworks

• Python
• TensorFlow / Keras
• NumPy
• Matplotlib
• Scikit-learn
• Streamlit
• pyngrok

5. Step-by-Step Implementation (Colab)

Step 1 — Install dependencies

```
!pip install -q streamlit pyngrok
```

Step 2 — Import libraries

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras import layers, models
from sklearn.metrics import classification_report, confusion_matrix
```

```
import seaborn as sns
import os
```

Step 3 — Load dataset

```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.cifar10.load_data()

print("Training data shape:", x_train.shape)
print("Test data shape:", x_test.shape)
```

Step 4 — Normalize

```
x_train = x_train.astype("float32") / 255.0
x_test = x_test.astype("float32") / 255.0
```

Step 5 — Class labels

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

Step 6 — Visualize samples

```
plt.figure(figsize=(10, 5))
for i in range(10):
    plt.subplot(2, 5, i+1)
    plt.imshow(x_train[i])
    plt.title(class_names[y_train[i][0]])
    plt.axis("off")
plt.show()
```

Step 7 — Build CNN model

```
model = models.Sequential([
    layers.Conv2D(32, (3,3), activation="relu", input_shape=(32, 32, 3)),
    layers.MaxPooling2D((2,2)),

    layers.Conv2D(64, (3,3), activation="relu"),
    layers.MaxPooling2D((2,2)),

    layers.Conv2D(128, (3,3), activation="relu"),
    layers.MaxPooling2D((2,2)),

    layers.Flatten(),
    layers.Dense(128, activation="relu"),
    layers.Dropout(0.5),
    layers.Dense(10, activation="softmax")
])

model.summary()
```

Step 8 — Compile model

```
model.compile(
    optimizer="adam",
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"]
)
```

Step 9 — Train model

```
history = model.fit(
    x_train, y_train,
    epochs=10,
    batch_size=64,
    validation_split=0.2
)
```

Step 10 — Plot learning curves

```
plt.figure(figsize=(12,5))

# Accuracy
plt.subplot(1,2,1)
plt.plot(history.history["accuracy"], label="Train Accuracy")
plt.plot(history.history["val_accuracy"], label="Val Accuracy")
plt.legend()
plt.title("Accuracy")

# Loss
plt.subplot(1,2,2)
plt.plot(history.history["loss"], label="Train Loss")
plt.plot(history.history["val_loss"], label="Val Loss")
plt.legend()
plt.title("Loss")

plt.show()
```

Step 11 — Evaluate on test set

```
test_loss, test_acc = model.evaluate(x_test, y_test)
print("Test Accuracy:", test_acc)
```

Step 12 — Classification report

```
y_pred_probs = model.predict(x_test)
y_pred = np.argmax(y_pred_probs, axis=1)
y_true = y_test.flatten()

print(classification_report(y_true, y_pred, target_names=class_names))
```

Step 13 — Confusion matrix

```
cm = confusion_matrix(y_true, y_pred)

plt.figure(figsize=(10,8))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=class_names, yticklabels=class_names)
plt.xlabel("Predicted")
plt.ylabel("True")
plt.title("Confusion Matrix")
plt.show()
```

Step 14 — Save model

```
model.save("cifar10_cnn_model.h5")
print("Model saved as cifar10_cnn_model.h5")
```

6. Streamlit App (Deployment in Colab)

The following Streamlit application loads the saved model and predicts the class of an uploaded image. Because Colab does not expose ports publicly by default, ngrok is used to create a public URL.

Step A — Create app.py

```
%%writefile app.py
import streamlit as st
import numpy as np
import tensorflow as tf
from PIL import Image

model = tf.keras.models.load_model("cifar10_cnn_model.h5")

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

st.title("CIFAR-10 Image Classifier")
st.write("Upload an image and the model will predict its class.")

uploaded_file = st.file_uploader("Choose an image...", type=["jpg", "jpeg", "png"])

if uploaded_file is not None:
    img = Image.open(uploaded_file).convert("RGB")
    st.image(img, caption="Uploaded Image", use_column_width=True)

    img_resized = img.resize((32, 32))
    img_array = np.array(img_resized).astype("float32") / 255.0
    img_array = np.expand_dims(img_array, axis=0)

    prediction = model.predict(img_array)
    predicted_class = class_names[np.argmax(prediction)]

    st.subheader("Prediction:")
    st.write(f"### {predicted_class}")
```

Step B — Run Streamlit

```
!streamlit run app.py &>/content/logs.txt &
```

Step C — Expose Streamlit using ngrok

```
from pyngrok import ngrok

public_url = ngrok.connect(8501)
print("Streamlit App URL:", public_url)
```

7. Evaluation Summary

Model performance is measured using test accuracy and per-class metrics. A confusion matrix is used to visualize which classes are most frequently confused. Typical CNN accuracy on CIFAR-10 for this architecture is around 65–75% depending on training settings.

8. Conclusion

This capstone demonstrates a complete deep learning workflow: data preprocessing, model design, training, evaluation, model persistence, and deployment in a lightweight interactive application.