```python
# --- Install & Import Dependencies ---
!pip install torch torchvision matplotlib pillow opencv-python scikit-learn pandas seaborn

import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader
import torchvision
from torchvision import transforms, models, datasets
from torchvision.models import resnet18, ResNet18_Weights
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import cv2
from google.colab import files
from sklearn.metrics import confusion_matrix, classification_report
import pandas as pd
import seaborn as sns
from torch.utils.data import random_split

# --- Step 1: Load CIFAR-10 Dataset (from Untitled3) ---
transform_cifar = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])

trainset = datasets.CIFAR10(root='./data', train=True, download=True,
transform=transform_cifar)
testset = datasets.CIFAR10(root='./data', train=False, download=True,
transform=transform_cifar)

# Split train into train/val (85%/15%)
train_size = int(0.85 * len(trainset))
val_size = len(trainset) - train_size
train_dataset, val_dataset = random_split(trainset, [train_size,
val_size])

trainloader = DataLoader(train_dataset, batch_size=64, shuffle=True)
valloader = DataLoader(val_dataset, batch_size=64, shuffle=False)
testloader = DataLoader(testset, batch_size=64, shuffle=False)

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

# --- Step 2: Define Custom CNN (translated from Untitled3) ---
class CustomCNN(nn.Module):
    def __init__(self):
        super(CustomCNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 32, kernel_size=3, padding=1)
```

```
        self.relu1 = nn.ReLU()
        self.conv2 = nn.Conv2d(32, 32, kernel_size=3, padding=0)
        self.relu2 = nn.ReLU()
        self.pool1 = nn.MaxPool2d(kernel_size=2, stride=2)
        self.dropout1 = nn.Dropout(0.25)

        self.conv3 = nn.Conv2d(32, 64, kernel_size=3, padding=1)
        self.relu3 = nn.ReLU()
        self.conv4 = nn.Conv2d(64, 64, kernel_size=3, padding=0)
        self.relu4 = nn.ReLU()
        self.pool2 = nn.MaxPool2d(kernel_size=2, stride=2)
        self.dropout2 = nn.Dropout(0.25)

        self.flatten = nn.Flatten()
        self.fc1 = nn.Linear(64 * 6 * 6, 512)  # Adjusted based on 
feature map size
        self.relu5 = nn.ReLU()
        self.dropout3 = nn.Dropout(0.5)
        self.fc2 = nn.Linear(512, 10)

    def forward(self, x):
        x = self.relu1(self.conv1(x))
        x = self.relu2(self.conv2(x))
        x = self.pool1(x)
        x = self.dropout1(x)

        x = self.relu3(self.conv3(x))
        x = self.relu4(self.conv4(x))
        x = self.pool2(x)
        x = self.dropout2(x)

        x = self.flatten(x)
        x = self.relu5(self.fc1(x))
        x = self.dropout3(x)
        x = self.fc2(x)
        return x

model_cnn = CustomCNN()
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model_cnn.parameters(), lr=0.001)

# --- Step 3: Train the Model (from Untitled3) ---
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model_cnn.to(device)

num_epochs = 50
train_losses, val_losses = [], []
train_accs, val_accs = [], []

for epoch in range(num_epochs):
```

```
    model_cnn.train()
    running_loss = 0.0
    correct = 0
    total = 0
    for inputs, labels in trainloader:
        inputs, labels = inputs.to(device), labels.to(device)
        optimizer.zero_grad()
        outputs = model_cnn(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item()
        _, predicted = outputs.max(1)
        total += labels.size(0)
        correct += predicted.eq(labels).sum().item()
    train_losses.append(running_loss / len(trainloader))
    train_accs.append(100. * correct / total)

    model_cnn.eval()
    val_loss = 0.0
    correct = 0
    total = 0
    with torch.no_grad():
        for inputs, labels in valloader:
            inputs, labels = inputs.to(device), labels.to(device)
            outputs = model_cnn(inputs)
            loss = criterion(outputs, labels)
            val_loss += loss.item()
            _, predicted = outputs.max(1)
            total += labels.size(0)
            correct += predicted.eq(labels).sum().item()
    val_losses.append(val_loss / len(valloader))
    val_accs.append(100. * correct / total)

    print(f'Epoch {epoch+1}/{num_epochs} - Train Loss: {train_losses[-
1]:.4f}, Val Loss: {val_losses[-1]:.4f}')

# --- Step 4: Plot Accuracy and Loss (from Untitled3) ---
fig, ax = plt.subplots(1, 2, figsize=(12, 4))
ax[0].plot(train_accs, label='Train Accuracy')
ax[0].plot(val_accs, label='Val Accuracy')
ax[0].set_title('Accuracy Over Epochs')
ax[0].legend()
ax[1].plot(train_losses, label='Train Loss')
ax[1].plot(val_losses, label='Val Loss')
ax[1].set_title('Loss Over Epochs')
ax[1].legend()
plt.show()

# --- Step 5: Evaluate on Test Set (from Untitled3) ---
```

```
model_cnn.eval()
y_true, y_pred = [], []
with torch.no_grad():
    for inputs, labels in testloader:
        inputs = inputs.to(device)
        outputs = model_cnn(inputs)
        _, predicted = outputs.max(1)
        y_true.extend(labels.cpu().numpy())
        y_pred.extend(predicted.cpu().numpy())

train_acc = train_accs[-1]
test_acc = sum(np.array(y_true) == np.array(y_pred)) / len(y_true) *
100

# Accuracy Table
accuracy_table = pd.DataFrame({
    'Metric': ['Training Accuracy', 'Testing Accuracy'],
    'Value': [train_acc, test_acc]
})
print("\nTable of Training and Testing Accuracy:")
print(accuracy_table)

# Confusion Matrix
conf_matrix = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
xticklabels=class_names, yticklabels=class_names)
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()

# Classification Report
print("\nClassification Report:")
print(classification_report(y_true, y_pred, target_names=class_names))

# --- Step 6: Pretrained Model for Grad-CAM (from Untitled9) ---
model = resnet18(weights=ResNet18_Weights.DEFAULT)
model.eval()

final_conv = None
gradients = None

def forward_hook(module, input, output):
    global final_conv
    final_conv = output

def backward_hook(module, grad_in, grad_out):
    global gradients
    gradients = grad_out[0]
```

```
target_layer = model.layer4[-1].conv2
target_layer.register_forward_hook(forward_hook)
target_layer.register_full_backward_hook(backward_hook)

# Preprocessing for Grad-CAM
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229,
0.224, 0.225])
])

# --- Step 7: Upload and Process Images for Grad-CAM (from Untitled9,
with limit like Untitled3) ---
uploaded = files.upload()
image_paths = list(uploaded.keys())[:3]  # Limit to 3 like Untitled3

def generate_gradcam(img_path, model):
    img = Image.open(img_path).convert("RGB")
    input_tensor = transform(img).unsqueeze(0)

    output = model(input_tensor)
    pred_class = output.argmax().item()

    model.zero_grad()
    class_loss = output[0, pred_class]
    class_loss.backward()

    pooled_grads = torch.mean(gradients, dim=[0, 2, 3])
    activations = final_conv[0]
    for i in range(len(pooled_grads)):
        activations[i, :, :] *= pooled_grads[i]
    heatmap = torch.mean(activations, dim=0).detach().cpu().numpy()
    heatmap = np.maximum(heatmap, 0)
    heatmap /= np.max(heatmap) if np.max(heatmap) != 0 else 1

    heatmap = cv2.resize(heatmap, (img.size[0], img.size[1]))
    heatmap = np.uint8(255 * heatmap)
    heatmap = cv2.applyColorMap(heatmap, cv2.COLORMAP_JET)

    superimposed = cv2.addWeighted(np.array(img), 0.6, heatmap, 0.4,
0)

    fig, ax = plt.subplots(1, 2, figsize=(10, 5))
    ax[0].imshow(img)
    ax[0].set_title("Original Image")
    ax[0].axis("off")

    ax[1].imshow(superimposed)
```

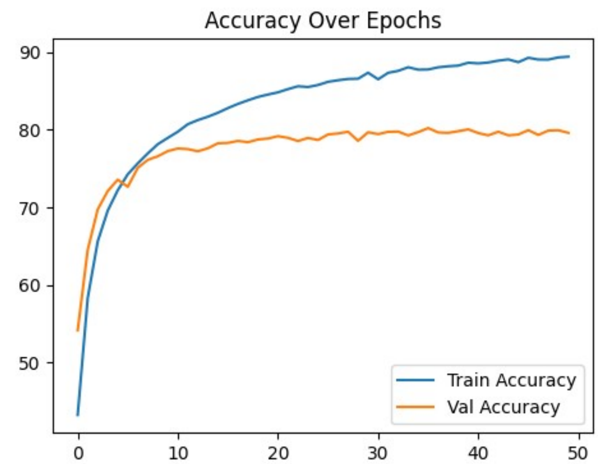
```python
    ax[1].set_title("GradCAM Visualization")
    ax[1].axis("off")

    plt.show()
    print(f"Predicted class index: {pred_class}")

for img_path in image_paths:
    print(f"\nProcessing: {img_path}")
    generate_gradcam(img_path, model)
```

```
Requirement already satisfied: torch in
/usr/local/lib/python3.12/dist-packages (2.8.0+cu126)
Requirement already satisfied: torchvision in
/usr/local/lib/python3.12/dist-packages (0.23.0+cu126)
Requirement already satisfied: matplotlib in
/usr/local/lib/python3.12/dist-packages (3.10.0)
Requirement already satisfied: pillow in
/usr/local/lib/python3.12/dist-packages (11.3.0)
Requirement already satisfied: opencv-python in
/usr/local/lib/python3.12/dist-packages (4.12.0.88)
Requirement already satisfied: scikit-learn in
/usr/local/lib/python3.12/dist-packages (1.6.1)
Requirement already satisfied: pandas in
/usr/local/lib/python3.12/dist-packages (2.2.2)
Requirement already satisfied: seaborn in
/usr/local/lib/python3.12/dist-packages (0.13.2)
Requirement already satisfied: filelock in
/usr/local/lib/python3.12/dist-packages (from torch) (3.19.1)
Requirement already satisfied: typing-extensions>=4.10.0 in
/usr/local/lib/python3.12/dist-packages (from torch) (4.15.0)
Requirement already satisfied: setuptools in
/usr/local/lib/python3.12/dist-packages (from torch) (75.2.0)
Requirement already satisfied: sympy>=1.13.3 in
/usr/local/lib/python3.12/dist-packages (from torch) (1.13.3)
Requirement already satisfied: networkx in
/usr/local/lib/python3.12/dist-packages (from torch) (3.5)
Requirement already satisfied: jinja2 in
/usr/local/lib/python3.12/dist-packages (from torch) (3.1.6)
Requirement already satisfied: fsspec in
/usr/local/lib/python3.12/dist-packages (from torch) (2025.3.0)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.6.77 in
/usr/local/lib/python3.12/dist-packages (from torch) (12.6.77)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.6.77 in
/usr/local/lib/python3.12/dist-packages (from torch) (12.6.77)
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.6.80 in
/usr/local/lib/python3.12/dist-packages (from torch) (12.6.80)
Requirement already satisfied: nvidia-cudnn-cu12==9.10.2.21 in
/usr/local/lib/python3.12/dist-packages (from torch) (9.10.2.21)
Requirement already satisfied: nvidia-cublas-cu12==12.6.4.1 in
/usr/local/lib/python3.12/dist-packages (from torch) (12.6.4.1)
```

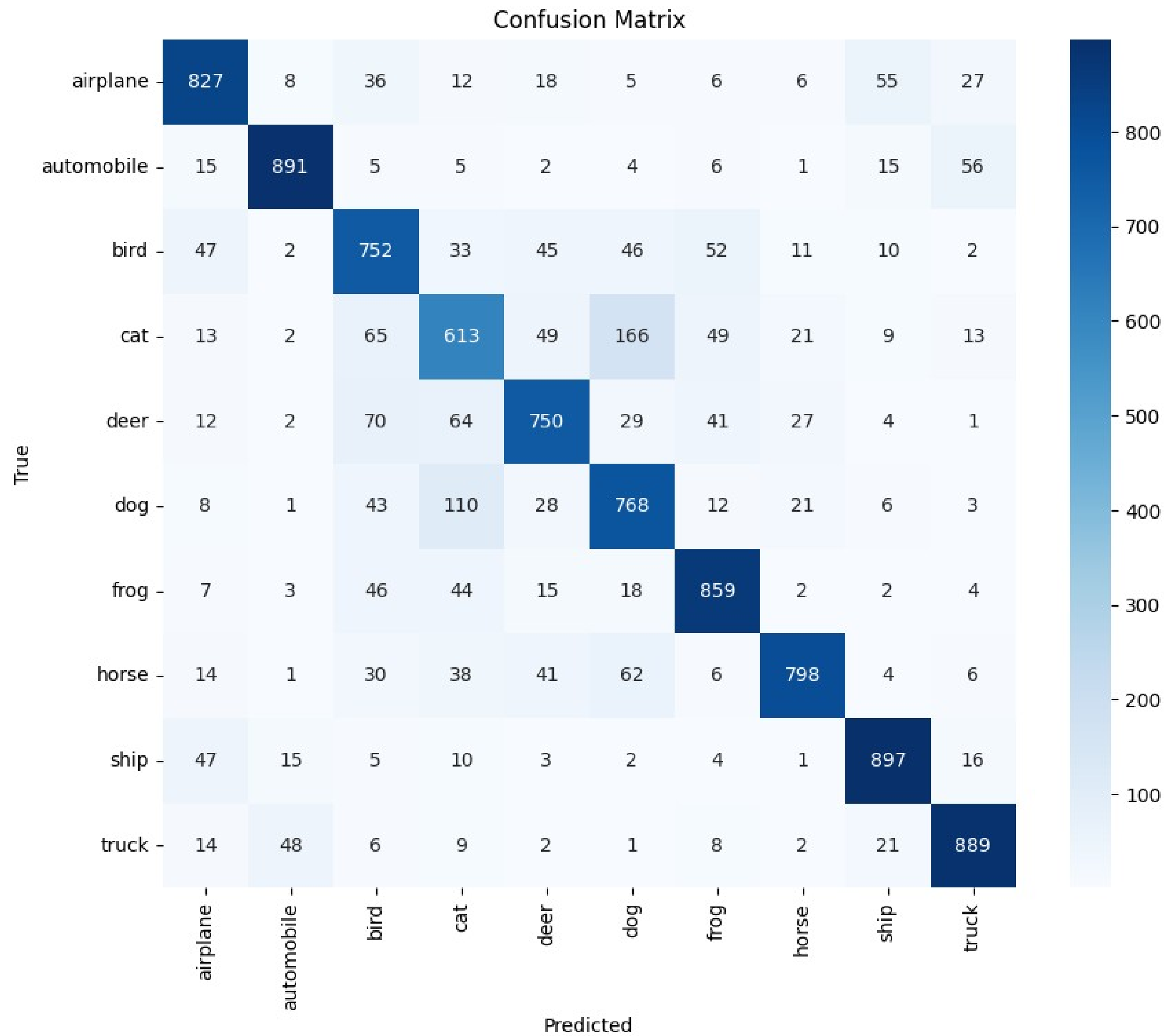
```
Requirement already satisfied: nvidia-cufft-cu12==11.3.0.4 in
/usr/local/lib/python3.12/dist-packages (from torch) (11.3.0.4)
Requirement already satisfied: nvidia-curand-cu12==10.3.7.77 in
/usr/local/lib/python3.12/dist-packages (from torch) (10.3.7.77)
Requirement already satisfied: nvidia-cusolver-cu12==11.7.1.2 in
/usr/local/lib/python3.12/dist-packages (from torch) (11.7.1.2)
Requirement already satisfied: nvidia-cusparse-cu12==12.5.4.2 in
/usr/local/lib/python3.12/dist-packages (from torch) (12.5.4.2)
Requirement already satisfied: nvidia-cusparselt-cu12==0.7.1 in
/usr/local/lib/python3.12/dist-packages (from torch) (0.7.1)
Requirement already satisfied: nvidia-nccl-cu12==2.27.3 in
/usr/local/lib/python3.12/dist-packages (from torch) (2.27.3)
Requirement already satisfied: nvidia-nvtx-cu12==12.6.77 in
/usr/local/lib/python3.12/dist-packages (from torch) (12.6.77)
Requirement already satisfied: nvidia-nvjitlink-cu12==12.6.85 in
/usr/local/lib/python3.12/dist-packages (from torch) (12.6.85)
Requirement already satisfied: nvidia-cufile-cu12==1.11.1.6 in
/usr/local/lib/python3.12/dist-packages (from torch) (1.11.1.6)
Requirement already satisfied: triton==3.4.0 in
/usr/local/lib/python3.12/dist-packages (from torch) (3.4.0)
Requirement already satisfied: numpy in
/usr/local/lib/python3.12/dist-packages (from torchvision) (2.0.2)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.12/dist-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.12/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.12/dist-packages (from matplotlib) (4.59.1)
Requirement already satisfied: kiwisolver>=1.3.1 in
/usr/local/lib/python3.12/dist-packages (from matplotlib) (1.4.9)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.12/dist-packages (from matplotlib) (25.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.12/dist-packages (from matplotlib) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.12/dist-packages (from matplotlib)
(2.9.0.post0)
Requirement already satisfied: scipy>=1.6.0 in
/usr/local/lib/python3.12/dist-packages (from scikit-learn) (1.16.1)
Requirement already satisfied: joblib>=1.2.0 in
/usr/local/lib/python3.12/dist-packages (from scikit-learn) (1.5.1)
Requirement already satisfied: threadpoolctl>=3.1.0 in
/usr/local/lib/python3.12/dist-packages (from scikit-learn) (3.6.0)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.12/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in
/usr/local/lib/python3.12/dist-packages (from pandas) (2025.2)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.7-
```

```
>matplotlib) (1.17.0)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.12/dist-packages (from sympy>=1.13.3->torch)
(1.3.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.12/dist-packages (from jinja2->torch) (3.0.2)

100%|██████████| 170M/170M [00:04<00:00, 35.5MB/s]

Epoch 1/50 - Train Loss: 1.5525, Val Loss: 1.2804
Epoch 2/50 - Train Loss: 1.1719, Val Loss: 0.9988
Epoch 3/50 - Train Loss: 0.9778, Val Loss: 0.8328
Epoch 4/50 - Train Loss: 0.8669, Val Loss: 0.7805
Epoch 5/50 - Train Loss: 0.7936, Val Loss: 0.7291
Epoch 6/50 - Train Loss: 0.7387, Val Loss: 0.7764
Epoch 7/50 - Train Loss: 0.6991, Val Loss: 0.6900
Epoch 8/50 - Train Loss: 0.6555, Val Loss: 0.6677
Epoch 9/50 - Train Loss: 0.6240, Val Loss: 0.6599
Epoch 10/50 - Train Loss: 0.5989, Val Loss: 0.6455
Epoch 11/50 - Train Loss: 0.5739, Val Loss: 0.6476
Epoch 12/50 - Train Loss: 0.5447, Val Loss: 0.6513
Epoch 13/50 - Train Loss: 0.5303, Val Loss: 0.6427
Epoch 14/50 - Train Loss: 0.5146, Val Loss: 0.6504
Epoch 15/50 - Train Loss: 0.4976, Val Loss: 0.6202
Epoch 16/50 - Train Loss: 0.4831, Val Loss: 0.6278
Epoch 17/50 - Train Loss: 0.4686, Val Loss: 0.6054
Epoch 18/50 - Train Loss: 0.4594, Val Loss: 0.6491
Epoch 19/50 - Train Loss: 0.4456, Val Loss: 0.6268
Epoch 20/50 - Train Loss: 0.4362, Val Loss: 0.6333
Epoch 21/50 - Train Loss: 0.4246, Val Loss: 0.6322
Epoch 22/50 - Train Loss: 0.4128, Val Loss: 0.6209
Epoch 23/50 - Train Loss: 0.4067, Val Loss: 0.6508
Epoch 24/50 - Train Loss: 0.4082, Val Loss: 0.6485
Epoch 25/50 - Train Loss: 0.4015, Val Loss: 0.6405
Epoch 26/50 - Train Loss: 0.3879, Val Loss: 0.6365
Epoch 27/50 - Train Loss: 0.3866, Val Loss: 0.6185
Epoch 28/50 - Train Loss: 0.3808, Val Loss: 0.6139
Epoch 29/50 - Train Loss: 0.3779, Val Loss: 0.6456
Epoch 30/50 - Train Loss: 0.3626, Val Loss: 0.6352
Epoch 31/50 - Train Loss: 0.3818, Val Loss: 0.6347
Epoch 32/50 - Train Loss: 0.3581, Val Loss: 0.6529
Epoch 33/50 - Train Loss: 0.3538, Val Loss: 0.6377
Epoch 34/50 - Train Loss: 0.3426, Val Loss: 0.6651
Epoch 35/50 - Train Loss: 0.3484, Val Loss: 0.6490
Epoch 36/50 - Train Loss: 0.3428, Val Loss: 0.6213
Epoch 37/50 - Train Loss: 0.3367, Val Loss: 0.6411
Epoch 38/50 - Train Loss: 0.3321, Val Loss: 0.6481
Epoch 39/50 - Train Loss: 0.3306, Val Loss: 0.6390
Epoch 40/50 - Train Loss: 0.3272, Val Loss: 0.6212
Epoch 41/50 - Train Loss: 0.3273, Val Loss: 0.6453
```

```
Epoch 42/50 - Train Loss: 0.3201, Val Loss: 0.6961
Epoch 43/50 - Train Loss: 0.3157, Val Loss: 0.6577
Epoch 44/50 - Train Loss: 0.3168, Val Loss: 0.6699
Epoch 45/50 - Train Loss: 0.3211, Val Loss: 0.6650
Epoch 46/50 - Train Loss: 0.3088, Val Loss: 0.6643
Epoch 47/50 - Train Loss: 0.3108, Val Loss: 0.6736
Epoch 48/50 - Train Loss: 0.3107, Val Loss: 0.6714
Epoch 49/50 - Train Loss: 0.3067, Val Loss: 0.6766
Epoch 50/50 - Train Loss: 0.3017, Val Loss: 0.6650
```




```
Table of Training and Testing Accuracy:
              Metric      Value
0  Training Accuracy  89.423529
1   Testing Accuracy  80.440000
```




```
Classification Report:
              precision    recall  f1-score   support

    airplane       0.82      0.83      0.83      1000
  automobile       0.92      0.89      0.90      1000
        bird       0.71      0.75      0.73      1000
         cat       0.65      0.61      0.63      1000
        deer       0.79      0.75      0.77      1000
         dog       0.70      0.77      0.73      1000
        frog       0.82      0.86      0.84      1000
       horse       0.90      0.80      0.84      1000
        ship       0.88      0.90      0.89      1000
       truck       0.87      0.89      0.88      1000

    accuracy                           0.80     10000
```

```
   macro avg       0.81      0.80      0.80     10000
weighted avg       0.81      0.80      0.80     10000

Downloading: "https://download.pytorch.org/models/resnet18-
f37072fd.pth" to /root/.cache/torch/hub/checkpoints/resnet18-
f37072fd.pth

100%|██████████| 44.7M/44.7M [00:00<00:00, 182MB/s]

<IPython.core.display.HTML object>

Saving airplane.jpg to airplane.jpg
Saving dog.jpg to dog.jpg
Saving ship.jpg to ship.jpg

Processing: airplane.jpg
```


```
Predicted class index: 908

Processing: dog.jpg
```


```
Predicted class index: 207

Processing: ship.jpg
```

```
Predicted class index: 628
```