HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BỘ MÔN PYTHON

BÁO CÁO BÀI TẬP LỚN 2

Giảng viên hướng dẫn : **KIM NGOC BACH**

Họ và tên sinh viên : **CHU MINH QUANG - B23DCVT354**

Lớp : **D23CQCE04 - B**

Hà Nội – 2025

I. Introduction

Image Classification (phân loại ảnh) là một bài toán nền tảng trong lĩnh vực thị giác máy tính (Computer Vision), trong đó mục tiêu là gán một nhãn (label) cho một ảnh đầu vào từ tập nhãn đã biết. Ví dụ, hệ thống có thể nhận diện một ảnh là "con chó", "xe hơi", "máy bay",...

Vai trò và ứng dụng:

- Nhận dạng chữ viết tay (digit recognition)
- Phân loại sản phẩm trong thương mại điện tử
- Nhận diện khuôn mặt, vật thể, y học (ảnh X-quang, MRI,...)
- Trợ lý lái xe, hệ thống camera thông minh

CNN – Mạng nơ-ron tích chập là một loại kiến trúc mạng nơ-ron sâu được thiết kế đặc biệt để xử lý dữ liệu dạng lưới – như ảnh 2D. CNN đã trở thành nền tảng cho hầu hết các tiến bộ trong lĩnh vực thị giác máy tính.

Cấu trúc chính của CNN:

1. **Lớp tích chập (Convolutional Layer)**: Tự động trích xuất đặc trưng (features) từ ảnh như cạnh, góc, họa tiết...
2. **Lớp kích hoạt (ReLU)**: Tăng tính phi tuyến cho mô hình.
3. **Lớp gộp (Pooling Layer)**: Giảm chiều kích thước không gian, giảm số tham số và tránh overfitting.
4. **Lớp fully connected**: Tích hợp đặc trưng và thực hiện phân loại.

Ưu điểm:

- Hiệu quả cao trong xử lý ảnh và video
- Tự động học đặc trưng mà không cần trích xuất thủ công
- Ứng dụng tốt cho cả nhận dạng đối tượng, phát hiện vật thể, segmentation,...

II. Các thư viện cần thiết

```python
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
import torchvision
import torchvision.transforms as transforms
import matplotlib.pyplot as plt
from torch.utils.data import DataLoader, random_split
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, accuracy_score

```

torch, torch.nn, torch.optim: Thư viện chính của PyTorch để xây dựng mô hình, huấn luyện và tối ưu.

torchvision: Dùng để tải và xử lý tập dữ liệu hình ảnh (CIFAR-10).

matplotlib.pyplot: Dùng để vẽ biểu đồ (loss và accuracy).

sklearn.metrics: Đánh giá hiệu suất mô hình qua confusion matrix và độ chính xác.

III. Chuẩn bị dữ liệu

```python
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])
train_dataset = torchvision.datasets.CIFAR10(root='./data', train=True,
                                             download=True, transform=transform)
test_dataset = torchvision.datasets.CIFAR10(root='./data', train=False,
                                            download=True, transform=transform)
train_size = int(0.8 * len(train_dataset))
val_size = len(train_dataset) - train_size
train_data, val_data = random_split(train_dataset, [train_size, val_size])
train_loader = DataLoader(train_data, batch_size=64, shuffle=True)
val_loader = DataLoader(val_data, batch_size=64, shuffle=False)
test_loader = DataLoader(test_dataset, batch_size=64, shuffle=False)
```

Transform: Biến đổi ảnh thành tensor và chuẩn hóa giá trị pixel về khoảng [-1, 1].

Tải CIFAR-10: Ảnh màu 32x32 thuộc 10 lớp (xe, mèo, chó...).

Random_split: Chia tập huấn luyện thành tập train (80%) và val (20%).

DataLoader: Tạo các lô dữ liệu (batches) để dễ huấn luyện.

IV. Xây dựng mô hình CNN

```
28  class CNN(nn.Module):
29      def __init__(self):
30          super(CNN, self).__init__()
31          self.conv1 = nn.Conv2d(3, 32, 3, padding=1)
32          self.conv2 = nn.Conv2d(32, 64, 3, padding=1)
33          self.conv3 = nn.Conv2d(64, 128, 3, padding=1)
34          self.fc1 = nn.Linear(128 * 4 * 4, 256)
35          self.fc2 = nn.Linear(256, 10)
36
37      def forward(self, x):
38          x = F.relu(self.conv1(x))
39          x = F.max_pool2d(x, 2)
40          x = F.relu(self.conv2(x))
41          x = F.max_pool2d(x, 2)
42          x = F.relu(self.conv3(x))
43          x = F.max_pool2d(x, 2)
44          x = x.view(-1, 128 * 4 * 4)
45          x = F.relu(self.fc1(x))
46          x = self.fc2(x)
47          return x
```

3 lớp tích chập: conv1, conv2, conv3 với số kênh tăng dần (32 → 64 → 128)

Max Pooling: Giảm kích thước đầu ra sau mỗi lớp tích chập.

Flatten + Fully Connected: Biến tensor thành vector và đưa vào 2 lớp dense (fc1, fc2)

Số đầu ra: 10 lớp tương ứng 10 nhãn ảnh của CIFAR-10

V. Huấn luyện mô hình

```
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model = CNN().to(device)
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
train_losses, val_losses, train_accuracies, val_accuracies = [], [], [], []
num_epochs = 10

for epoch in range(num_epochs):
    model.train()
    train_loss, correct, total = 0, 0, 0
    for images, labels in train_loader:
        images, labels = images.to(device), labels.to(device)
        outputs = model(images)
        loss = criterion(outputs, labels)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        train_loss += loss.item()
        _, predicted = torch.max(outputs, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()
    train_losses.append(train_loss / len(train_loader))
    train_accuracies.append(correct / total)

    model.eval()
    val_loss, val_correct, val_total = 0, 0, 0
    with torch.no_grad():
        for images, labels in val_loader:
            images, labels = images.to(device), labels.to(device)
            outputs = model(images)
            loss = criterion(outputs, labels)
            val_loss += loss.item()
            _, predicted = torch.max(outputs, 1)
            val_total += labels.size(0)
            val_correct += (predicted == labels).sum().item()
    val_losses.append(val_loss / len(val_loader))
    val_accuracies.append(val_correct / val_total)
```

Chọn thiết bị: Sử dụng GPU nếu có (cuda), ngược lại dùng CPU.

CrossEntropyLoss: Hàm mất mát dùng cho bài toán phân loại nhiều lớp.

Adam optimizer: Phương pháp tối ưu hiệu quả và phổ biến.

Vòng lặp for epoch: Huấn luyện mô hình qua nhiều lần quét dữ liệu.

Trong mỗi epoch:

Huấn luyện (model.train()):

- Tính đầu ra, mất mát, gradient và cập nhật trọng số.
- Tính tổng loss và độ chính xác trên tập huấn luyện.

Đánh giá (model.eval()):

- Không cập nhật trọng số.
- Tính loss và độ chính xác trên tập validation.

VI. Vẽ biểu đồ (Learning Curve)

```
 89    plt.figure(figsize=(10, 4))
 90    plt.subplot(1, 2, 1)
 91    plt.plot(train_losses, label='Train Loss')
 92    plt.plot(val_losses, label='Val Loss')
 93    plt.title('Loss Curve')
 94    plt.xlabel('Epoch')
 95    plt.ylabel('Loss')
 96    plt.legend()
 97
 98    plt.subplot(1, 2, 2)
 99    plt.plot(train_accuracies, label='Train Acc')
100    plt.plot(val_accuracies, label='Val Acc')
101    plt.title('Accuracy Curve')
102    plt.xlabel('Epoch')
103    plt.ylabel('Accuracy')
104    plt.legend()
105    plt.show()
```

Giúp quan sát quá trình huấn luyện:

Loss giảm dần → mô hình học được.

Accuracy tăng dần → mô hình dần chính xác hơn.

Phát hiện overfitting nếu train tốt nhưng val kém.

VII. Đánh giá mô hình trên tập test và hiển thị Confusion Matrix

```python
model.eval()
all_preds, all_labels = [], []
with torch.no_grad():
    for images, labels in test_loader:
        images = images.to(device)
        outputs = model(images)
        _, predicted = torch.max(outputs, 1)
        all_preds.extend(predicted.cpu().numpy())
        all_labels.extend(labels.numpy())
test_acc = accuracy_score(all_labels, all_preds)
print(f"Test Accuracy: {test_acc * 100:.2f}%")

cm = confusion_matrix(all_labels, all_preds)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=train_dataset.classes)
disp.plot(cmap='Blues', xticks_rotation='vertical')
plt.title("Confusion Matrix - CNN")
plt.show()
```

Dự đoán trên dữ liệu chưa thấy (test set)

Tính độ chính xác tổng thể của mô hình sau khi huấn luyện hoàn tất.

Ma trận thể hiện **số lượng mẫu đúng/sai giữa các lớp.**

Dễ thấy mô hình **nhầm lẫn giữa lớp nào với lớp nào.**

Dùng để **phân tích chi tiết hiệu suất phân loại.**