Tổng quan về Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNNs) là một loại mô hình học sâu (deep learning) được sử dụng rộng rãi trong việc phân tích hình ảnh và nhận diện mẫu. Dưới đây là một cái nhìn tổng quan và mã nguồn mẫu để minh họa cách sử dụng CNN.

1. Cấu trúc của CNN

CNN gồm các lớp chính:

- **Convolutional Layer**: Thực hiện phép tích chập để trích xuất đặc trưng từ hình ảnh đầu vào.
- **Activation Layer**: Thường dùng hàm ReLU để thêm tính phi tuyến vào mô hình.
- **Pooling Layer**: Thường dùng MaxPooling để giảm kích thước không gian và số lượng tham số.
- **Fully Connected Layer**: Kết nối toàn bộ các nút từ lớp trước và sử dụng để phân loại cuối cùng.

2. Nguyên lý hoạt động

- **Trích xuất đặc trưng**: Lớp tích chập (convolutional) trích xuất các đặc trưng cục bộ.
- **Giảm kích thước**: Lớp gộp (pooling) giúp giảm độ phức tạp và giữ lại các đặc trưng quan trọng.
- **Phân loại**: Lớp kết nối đầy đủ (fully connected) dựa trên các đặc trưng đã trích xuất để đưa ra kết quả phân loại.

3. Ưu điểm của CNN

- **Tự động học đặc trưng**: Giảm bớt sự can thiệp của con người trong việc xác định đặc trưng.
- **Tính bất biến dịch chuyển**: Khả năng nhận diện đối tượng bất kể vị trí của chúng trong hình ảnh.

- **Hiệu suất cao**: Thường đạt hiệu suất cao trong các bài toán nhận diện hình ảnh.

4. Ứng dụng của CNN

Nhận diện hình ảnh: Phân loại hình ảnh, nhận diện khuôn mặt, nhận diện vật thể.

Y tế: Chẩn đoán từ hình ảnh y khoa, phân đoạn ảnh y khoa.

Thị giác máy tính: Phát hiện và theo dõi đối tượng, phân loại hành động.

Xử lý ngôn ngữ tự nhiên: Nhận diện chữ viết tay, phân loại văn bản.

5. Mã nguồn mẫu

Dưới đây là mã nguồn Python mẫu sử dụng PyTorch để xây dựng và huấn luyện mô hình CNN trên bộ dữ liệu CIFAR-10.

```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
from sklearn.metrics import precision_score, recall_score

# Chuẩn bị dữ liệu
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])

trainset = torchvision.datasets.CIFAR10(root='./data', train=True,
                                        download=True, transform=transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=100,
                                          shuffle=True, num_workers=2)

testset = torchvision.datasets.CIFAR10(root='./data', train=False,
                                       download=True, transform=transform)
testloader = torch.utils.data.DataLoader(testset, batch_size=100,
                                         shuffle=False, num_workers=2)

# Định nghĩa mô hình CNN
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 16 * 5 * 5)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

```
net = Net()

criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(net.parameters(), lr=0.001, momentum=0.9)

# Huấn luyện mô hình
for epoch in range(2):
    running_loss = 0.0
    for i, data in enumerate(trainloader, 0):
        inputs, labels = data
        optimizer.zero_grad()
        outputs = net(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item()
        if i % 200 == 199:
            print(f'[Epoch: {epoch + 1}, Mini-batch: {i + 1}] loss: {runnin
            running_loss = 0.0

print('Finished Training')

# Dự đoán và tính toán Precision và Recall
all_preds = []
all_labels = []

with torch.no_grad():
    for data in testloader:
        images, labels = data
        outputs = net(images)
        _, predicted = torch.max(outputs, 1)
        all_preds.extend(predicted.numpy())
        all_labels.extend(labels.numpy())

precision = precision_score(all_labels, all_preds, average='macro') * 100
recall = recall_score(all_labels, all_preds, average='macro') * 100

print(f'Precision: {precision:.2f}%')
print(f'Recall: {recall:.2f}%')
```

Kết luận

CNN là một công cụ mạnh mẽ trong học sâu, đặc biệt hữu ích trong xử lý hình ảnh và nhận diện mẫu. Với các thư viện như PyTorch và TensorFlow, việc xây dựng và huấn luyện các mô hình CNN trở nên dễ dàng và hiệu quả hơn.