دانشکده مهندسی کامپیوتر

گزارش درس مبانی هوش محاسباتی

عنوان گزارش:

تمرین سوم CNN

ارائه‌دهندگان:

فرزانه آقازاده **زینب جنتی** **فاطمه نجفی**

استاد درس:

دکتر حسین کارشناس

دستیاران درس:

رضا برزگر

علی شاه زمانی

آرمان خلیلی

نیم سال دوم ۱۴۰۳۱۴۰۴

فهرست

شرح کد

```
import torch
import torchvision
import torchvision.transforms as transforms
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
```

کتابخانه‌های اصلی PyTorch برای پیاده‌سازی شبکه عصبی کانولوشنی ایمپورت شده‌اند.

```
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])
```

ToTensor() تصاویر را به تانسورهای PyTorch تبدیل می‌کند.

Normalize() مقادیر پیکسل‌ها را از محدوده [۰,۱] به [-۱,۱] نرمال می‌کند.

```
trainset =
torchvision.datasets.CIFAR10(root="D:\\_frzna_\\Projects\\PycharmProjects\\CNN\\cifar-10-batches-py", train=True, download=True, transform=transform)

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

مجموعه داده CIFAR-10 را دانلود و بارگذاری می‌کند (۶۰۰۰۰ تصویر x32۳۲ رنگی در ۱۰ کلاس).

```
trainloader = torch.utils.data.DataLoader(trainset, batch_size=64,
shuffle=True)

testloader = torch.utils.data.DataLoader(testset, batch_size=64,
shuffle=False)
```

داده‌ها را به بچ‌های ۶۴ تایی تقسیم می‌کند و هر بار به صورت تصادفی مخلوط می‌شود.

تعریف معماری شبکه CNN

```
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 32, 3)
        self.pool = nn.MaxPool2d(2, 2)
```

```
        self.conv2 = nn.Conv2d(32, 64, 3)
        self.fc1 = nn.Linear(64 * 6 * 6, 128)
        self.fc2 = nn.Linear(128, 10)
    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))  # Conv1 → ReLU → Pooling
        x = self.pool(F.relu(self.conv2(x)))  # Conv2 → ReLU → Pooling
        x = torch.flatten(x, 1)  # Flatten
        x = F.relu(self.fc1(x))  # FC1 → ReLU
        x = self.fc2(x)
        return x
```

یک شبکه کانولوشنی با دو لایه کانولوشن و دو لایه تمام‌متصل تعریف شده است.

و مسیر پردازش داده از ورودی تا خروجی را مشخص شده:

۱. کانولوشن → ReLU → Pooling

۲. کانولوشن → ReLU → Pooling

۳. تبدیل به بردار یک‌بعدی (Flatten)

۴. لایه تمام‌متصل → ReLU

۵. لایه خروجی

ایجاد مدل، تابع هزینه و بهینه‌ساز

```
model = CNN()
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

مدل CNN ساخته می‌شود.

از loss function کراس آنتروپی برای طبقه‌بندی چندکلاسه استفاده شده است.

بهینه‌ساز Adam با نرخ یادگیری ۰.۰۰۱ تنظیم شده است.

فرآیند آموزش

```
for epoch in range(10):
    running_loss = 0.0
    for i, data in enumerate(trainloader, 0):
        inputs, labels = data

        optimizer.zero_grad()

        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
```

```
        running_loss += loss.item()

    print(f'Epoch {epoch + 1}, Loss: {running_loss / len(trainloader)}')

correct = 0
total = 0
with torch.no_grad():
    for data in testloader:
        images, labels = data
        outputs = model(images)
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

print(f'Test Accuracy: {100 * correct / total}% in {total} images')
```

مدل روی داده‌های تست ارزیابی می‌شود.
بدون محاسبه گرادیان‌ها (`torch.no_grad`) برای صرفه‌جویی در حافظه.
تعداد پیش‌بینی‌های صحیح شمرده می‌شود.
دقت نهایی روی مجموعه تست محاسبه و چاپ می‌شود.

تحلیل نتایج

```
Epoch 1, Loss: 1.412473019126736
Epoch 2, Loss: 1.0414021530419664
Epoch 3, Loss: 0.8871514912304062
Epoch 4, Loss: 0.7716826744320447
Epoch 5, Loss: 0.6751528580094237
Epoch 6, Loss: 0.592101705539257
Epoch 7, Loss: 0.5118970723293931
Epoch 8, Loss: 0.44109894948847156
Epoch 9, Loss: 0.3722288070241814
Epoch 10, Loss: 0.3221115250607281
Test Accuracy: 71.79% in 10000 images
```

۱. Epoch 1: Loss = 1.412

مقدار اولیه نسبتاً بالا، نشان‌دهنده شروع آموزش از نقطه تصادفی

۲. Epoch 5: Loss = 0.675 (~52% کاهش نسبت به شروع)

پیشرفت سریع در یادگیری ویژگی‌های پایه

۳. Epoch 10: Loss = 0.322 (~77% کاهش نسبت به شروع)

مدل به خوبی همگرا شده و همچنان در حال بهبود است

الگوی کاهش Loss

کاهش سریع در اپوک‌های اول (یادگیری ویژگی‌های کلی)

کاهش آهسته‌تر در اپوک‌های بعدی (یادگیری ویژگی‌های ظریف)

نکته: اگر آموزش ادامه می‌یافت، ممکن بود مدل به Overfitting منجر شود

دقت ۷۱.۷۹%:

مدل توانسته است حدود ۷۲ درصد از تصاویر تست را به درستی طبقه‌بندی کند

مدل‌های پیشرفته‌تر معمولاً به دقت ۸۵-۹۵٪ می‌رسند

راه‌های بهبود:

افزایش عمق شبکه (اضافه کردن لایه‌های بیشتر)

استفاده از تکنیک‌هایRegularization

افزایش تعداد اپوک‌ها با نظارت برOverfitting

تنظیم نرخ یادگیری (Learning Rate Scheduling)

مقایسه

شبکه یک لایه (Perceptron): فقط یک لایه وزن بین ورودی و خروجی

شبکه دو لایه: یک لایه پنهان + لایه خروجی

اتصالات کامل: هر نورون با تمام نورون‌های لایه بعدی ارتباط دارد

محدودیت‌ها برای داده‌های تصویری:

۱. عدم حفظ ساختار مکانی: موقعیت پیکسل‌ها در نظر گرفته نمی‌شود

۲. پارامترهای بسیار زیاد: برای یک تصویر ۳۲ RGBx32:

ورودی: ۳۲×۳۲×۳ = ۳۰۷۲ نورون

لایه پنهان با ۵۱۲ نورون ⟶ ۳۰۷۲×۵۱۲ ≈ ۱.۵ میلیون پارامتر!

۳. عدم تشخیص الگوهای محلی: نمی‌تواند لبه‌ها، بافت‌ها و اشکال ساده را تشخیص دهد

شبکه‌های کانولوشنی (CNN)

ویژگی‌های کلیدی:

۱. کانولوشن (Convolution):

فیلترهایی که روی تصویر حرکت می‌کنند و ویژگی‌های محلی را استخراج می‌کنند

مثال: تشخیص لبه‌ها، رنگ‌ها، بافت‌ها

مزیت: پارامترهای مشترک (weight sharing) کاهش پارامترها

۲. لایه Pooling:

کاهش ابعاد با حفظ ویژگی‌های مهم

معمولاً Max Pooling: بیشترین مقدار در هر ناحیه را نگه می‌دارد

مزیت: کاهش حساسیت به جابجایی و چرخش جزئی

۳. سلسله مراتب ویژگی‌ها:

لایه‌های اول: لبه‌ها و اشکال ساده

لایه‌های میانی: ترکیب اشکال ساده

لایه‌های آخر: اشیاء پیچیده

مقایسه عملکرد:

ویژگی	پرسپترون چند لایه	CNN
حفظ ساختار مکانی	خبر	بله
تشخیص الگوهای محلی	ضعیف	عالی
تعداد پارامترها	بسیار زیاد	بهینه شده
عملکرد روی تصاویر	ضعیف (دقت حدود ۵۰٪)	قوی (دقت حدود ۷۰-۹۵٪)
حساسیت به جابجایی تصویر	بسیار حساس	مقاوم

همانطور که درنتایج بخش های قبلی مشخص است در پرسپترونچند لایهی بخش چهارم بعد از ۵۰ ایپاک به دقت حدود ۵۰ درصد دست یافته ایم که هم از نظر زمانی و هم از نظر فضایی از سی ان ان بدتر عمل کرده و در نهایت به دقت کافی هم دست نیافته است.

چرا CNN برای تصاویر بهتر است؟

۱. محلی‌نگری: هر فیلتر فقط یک ناحیه کوچک را می‌بیند

۲. اشتراک وزن: یک فیلتر برای کل تصویر استفاده می‌شود

۳. سلسله مراتب: یادگیری ویژگی‌ها از ساده به پیچیده

جمع‌بندی:

برای داده‌های ساختاریافته ساده (مثل جداول): شبکه‌های معمولی ممکن است کافی باشند

برای داده‌های فضایی (تصاویر، ویدیو): CNN انتخاب بهتری است

منابع

chat.deepseek.com

chatgpt.com