در این بخش شما باید یک شبکه عصبی کانولوشنال (CNN) را برای دسته‌بندی تصاویر CIFAR-۱۰ با استفاده از PyTorch پیاده‌سازی کنید و عملکرد آن را با پرسپترون چندلایه قبلی مقایسه نمایید.

۱. تنظیم داده‌ها و بارگذاری CIFAR-۱۰

مجموعه‌داده CIFAR-۱۰ را با نرمال‌سازی مناسب بارگذاری کنید و هایDataLoader آموزش و تست را با اندازه بچ و شافل دلخواه ایجاد نمایید.

برنامه ۱: تعریف transforms و DataLoader

```
import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.functional as F
from torchvision import datasets, transforms
from torch.utils.data import DataLoader

# Device configuration
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

# Hyper-parameters
num_epochs = 20
batch_size = 128
learning_rate = 0.01

# CIFAR-10 dataset transforms
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.4914, 0.4822, 0.4465), (0.247, 0.243, 0.261))
])

# CIFAR-10 dataset
train_dataset = datasets.CIFAR10(root='./data', train=True, download=True,
    transform=transform)
test_dataset = datasets.CIFAR10(root='./data', train=False, download=True,
    transform=transform)
print(train_dataset.classes)
# Data loaders
train_loader = DataLoader(dataset=train_dataset, batch_size=batch_size,
    shuffle=True, num_workers=4)
test_loader = DataLoader(dataset=test_dataset, batch_size=batch_size,
    shuffle=False, num_workers=4)
```

۲. تعریف معماری ساده CNN

یک کلاس PyTorch از نوع nn.Module با ساختار زیر پیاده‌سازی کنید:

- Conv2d با ۳۲ فیلتر، کرنل 3×3، فعال‌سازی ReLU
- MaxPool2d با کرنل 2×2
- Conv2d با ۶۴ فیلتر، کرنل 3×3، فعال‌سازی ReLU
- MaxPool2d با کرنل 2×2
- Flatten() و دو لایه تمام‌متصل ۱۲۸ و ۱۰ نورونی Softmax) خروجی)

برنامه ۲: کلاس SimpleCNN

```python
class SimpleCNN(nn.Module):
    def __init__(self):
        super(SimpleCNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 32, kernel_size=3, padding=1)
        self.pool1 = nn.MaxPool2d(kernel_size=2, stride=2)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=3, padding=1)
        self.pool2 = nn.MaxPool2d(kernel_size=2, stride=2)
        self.fc1 = nn.Linear(64 * 8 * 8, 128)
        self.fc2 = nn.Linear(128, 10)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = self.pool1(x)
        x = F.relu(self.conv2(x))
        x = self.pool2(x)
        x = x.view(x.size(0), -1)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```

۳. حلقه آموزش مدل

حلقه آموزشی را با تابع هزینه CrossEntropyLoss و بهینه‌ساز SGD با Momentum پیاده کنید. در هر صد گام آموزشی، میانگین loss را چاپ نمایید.

برنامه ۳: حلقه آموزش train loop

```python
if __name__ == "__main__":
    model = SimpleCNN().to(device)

    # Loss and optimizer
    criterion = nn.CrossEntropyLoss()
    optimizer = optim.SGD(model.parameters(), lr=learning_rate, momentum=0.9)

    # Training loop
    for epoch in range(num_epochs):
        model.train()
        running_loss = 0.0
        for i, (images, labels) in enumerate(train_loader):
            images = images.to(device)
            labels = labels.to(device)

            # Forward pass
            outputs = model(images)
            loss = criterion(outputs, labels)

            # Backward and optimize
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()

            running_loss += loss.item()
            if (i + 1) % 100 == 0:
                print(f'Epoch [{epoch+1}/{num_epochs}], Step
    [{i+1}/{len(train_loader)}], Loss: {running_loss / 100:.4f}')
```

```
                running_loss = 0.0
```

۴. ارزیابی مدل و گزارش دقت

مدل آموزش‌دیده را روی مجموعه تست ارزیابی کنید و دقت نهایی را محاسبه و چاپ کنید. سپس وزن‌های مدل را ذخیره نمایید.

برنامه ۴: ارزیابی و ذخیره مدل

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

    print(f'Test Accuracy of the model on the 10000 test images: {100 *
    correct / total:.2f}%')

    # Save the model checkpoint
    torch.save(model.state_dict(), 'simple_cnn_cifar10.pth')
```

۵. گزارش معماری و تحلیل نتایج

یک گزارش مختصر بنویسید که شامل معماری نهایی، جزییات پیاده‌سازی، روند آموزش و نتایج ارزیابی باشد. همچنین مزایا و معایب استفاده از CNN را در مقایسه با پرسپترون چندلایه مورد بحث قرار دهید.

۶. مزایا و معایب استفاده از CNN در مقابل یک پرسپترون چندلایه

مزایا:

- **استخراج ویژگی‌های مکانی:** لایه‌های کانولوشنال با حفظ ساختار دوبعدی تصویر، الگوهای محلی مانند لبه‌ها و بافت‌ها را بهتر استخراج می‌کنند، در حالی که MLP تصویر را صاف‌شده دریافت می‌کند.
- **کاهش پارامترها:** با اشتراک وزن در فیلترها، تعداد پارامترها به‌طور قابل‌توجهی کمتر از MLP بوده و خطر بیش‌برازش کاهش می‌یابد.
- **مقیاس‌پذیری:** استفاده از pooling و فیلترهای محلی باعث می‌شود CNN بتواند روی تصاویر بزرگ‌تر یا پیچیده‌تر نیز کارایی مناسبی داشته باشد.
- **تعمیم بهتر:** CNN در برابر تغییرات کوچک در تصویر (جابجایی، چرخش، تغییر نور) مقاوم‌تر است و به همین دلیل در داده‌های بصری عملکرد بهتری دارد.

معایب:

- **پیچیدگی پیاده‌سازی:** طراحی معماری CNN اعم از انتخاب تعداد لایه‌ها، فیلترها و سایر ابرپارامترها نیازمند تجربه و آزمون‌وخطای بیشتری است.
- **زمان آموزش:** محاسبات کانولوشن هزینه‌برتر از ضرب‌های برداری ساده در MLP بوده و ممکن است زمان آموزش طولانی‌تری داشته باشد.

- **وابستگی به GPU:** برای آموزش سریع و مؤثر CNN معمولاً نیاز به شتاب‌دهنده‌هایی مانند GPU است، در حالی که MLP را می‌توان حتی روی CPU اجرا کرد.