《计算机视觉》实验报告

姓名：冯俊佳 学号：23122721

实验 10 卷积神经网络

一. 任务 1

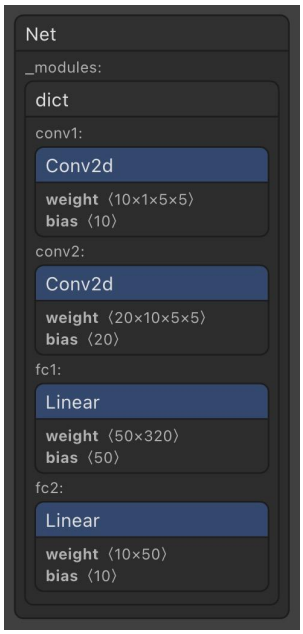
a) 核心代码：

```
# 定义转换操作，将图像数据转换为张量，并进行标准化
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.1307,), (0.3081,))])

# 下载并加载训练和测试数据集
train_dataset = datasets.MNIST(root='./data', train=True,
download=True, transform=transform)
test_dataset = datasets.MNIST(root='./data', train=False,
download=True, transform=transform)
train_loader = DataLoader(train_dataset, batch_size=64,
shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=1000,
shuffle=False)

# 定义一个简单的卷积神经网络
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 10, kernel_size=5)
        self.conv2 = nn.Conv2d(10, 20, kernel_size=5)
        self.fc1 = nn.Linear(320, 50)
        self.fc2 = nn.Linear(50, 10)
    def forward(self, x):
        x = F.relu(F.max_pool2d(self.conv1(x), 2))
        x = F.relu(F.max_pool2d(self.conv2(x), 2))
        x = x.view(-1, 320)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return F.log_softmax(x, dim=1)
# 实例化模型和优化器
```

```python
model = Net()
optimizer = optim.SGD(model.parameters(), lr=0.01, momentum=0.9)

# 定义训练函数
def train(model, device, train_loader, optimizer, epoch):
    model.train()
    epoch_loss = 0.0
    for batch_idx, (data, target) in enumerate(train_loader):
        data, target = data.to(device), target.to(device)
        optimizer.zero_grad()
        output = model(data)
        loss = F.nll_loss(output, target)
        loss.backward()
        optimizer.step()
        epoch_loss += loss.item()
    print(f'====> Epoch: {epoch} Average loss: {epoch_loss / len(train_loader):.4f}')

    # Evaluate training accuracy after epoch
    model.eval()
    correct = 0
    total = 0
    with torch.no_grad():
        for data, target in train_loader:
            data, target = data.to(device), target.to(device)
            output = model(data)
            pred = output.argmax(dim=1, keepdim=True)
            correct += pred.eq(target.view_as(pred)).sum().item()
            total += data.size(0)
    train_acc = 100. * correct / total
    print(f'Training Accuracy after Epoch {epoch}: {train_acc:.2f}%')

# 定义测试函数
def evaluate(model, device, test_loader):
    model.eval()
    test_loss = 0
    correct = 0
    with torch.no_grad():
        for data, target in test_loader:
            data, target = data.to(device), target.to(device)
            output = model(data)
            test_loss += F.nll_loss(output, target,
```

```
reduction='sum').item()  # 累计损失
            pred = output.argmax(dim=1, keepdim=True)  # 获取最大
可能的预测值
            correct +=
pred.eq(target.view_as(pred)).sum().item()

    test_loss /= len(test_loader.dataset)
    print(f'\nTest set: Average loss: {test_loss:.4f}, Accuracy:
{correct}/{len(test_loader.dataset)}'
          f' ({100. * correct /
len(test_loader.dataset):.0f}%)\n')

# 设置设备并运行训练
device = torch.device("cuda" if torch.cuda.is_available() else
"cpu")
model.to(device)

for epoch in range(1, 11):
    train(model, device, train_loader, optimizer, epoch)

# Evaluate on test set after all epochs
evaluate(model, device, test_loader)

# 保存模型参数到文件（权重）
torch.save(model.state_dict(), "mnist_cnn.pth")

# 保存整个模型结构（推荐用于 Netron 可视化）
torch.save(model, "mnist_cnn_model.pt")
```

b) 实验结果截图



如左图所示，为 CNN 的网络结构示意图，下表为示意图中的内容注释。

模块名称	类型	权重维度说明
conv1	Conv2d	weight: (10 × 1 × 5 × 5)：表示 10 个 5 × 5 卷积核，输入通道为 1
conv2	Conv2d	weight: (20 × 10 × 5 × 5): 20 个卷积核，输入通道为 10
fc1	Linear	weight: (50 × 320)：全连接层，输入展平后大小为 320
fc2	Linear	weight: (10 × 50)：输出为 10 类数字（0~9）

下图为训练后的准确度，以及测试一次时候的准确度。

```
====> Epoch: 1 Average loss: 0.2718
Training Accuracy after Epoch 1: 97.96%
====> Epoch: 2 Average loss: 0.0633
Training Accuracy after Epoch 2: 98.75%
====> Epoch: 3 Average loss: 0.0452
Training Accuracy after Epoch 3: 98.90%
====> Epoch: 4 Average loss: 0.0373
Training Accuracy after Epoch 4: 99.26%
====> Epoch: 5 Average loss: 0.0309
Training Accuracy after Epoch 5: 99.17%
====> Epoch: 6 Average loss: 0.0257
Training Accuracy after Epoch 6: 99.42%
====> Epoch: 7 Average loss: 0.0207
Training Accuracy after Epoch 7: 99.40%
====> Epoch: 8 Average loss: 0.0186
Training Accuracy after Epoch 8: 99.57%
====> Epoch: 9 Average loss: 0.0146
Training Accuracy after Epoch 9: 99.72%
====> Epoch: 10 Average loss: 0.0134
Training Accuracy after Epoch 10: 99.67%

Test set: Average loss: 0.0342, Accuracy: 9907/10000 (99%)
```

c) **实验小结**

本此实验基于 PyTorch 框架, 搭建了一个包含两层卷积和两层全连接层的卷积神经网络（CNN），并在经典的 MNIST 手写数字识别数据集上进行了训练和测试。经过 10 个 epoch 的训练，模型在训练集上的准确率达到了 99.67%，在测试集上达到了 99.07%的准确率，表现出良好的泛化能力。本实验还通过 Netron 可视化工具对 CNN 网络结构进行了可视化处理，进一步加深了我对卷积神经网络各层结构及其参数的理解，为后续更复杂的图像识别任务打下了基础。