图像分类

数据集

- 首先设置超参数，批次大小、学习率以及训练轮次。
- 使用 `transforms.Compose[]` 定义一组变换，将图像转换为张量以及归一化。
- 然后构建训练数据集与测试数据集
- 再使用 `torch.utils.data.DataLoader` 类对数据集构建一个数据加载器来进行批处理。该加载器可以在训练过程中对数据集采样，并一批次一批次的应用变换。

In [1]:
```
import torch
import torchvision
import torchvision.transforms as transforms

batch_size = 16
learning_rate = 0.001
num_epochs = 20

# 数据预处理转换器
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize(
        mean=(0.4914, 0.4822, 0.4465), # 每个通道的均
        std=(0.2023, 0.1994, 0.2010) # 每个通道的标准
    )
])

# 构建训练集数据，使用transform处理数据集
train_set = torchvision.datasets.CIFAR10('./data', t
```

```python
# 构建测试集数据，使用transform处理数据集
test_set = torchvision.datasets.CIFAR10('./data', tr

# 数据批处理
train_loader = torch.utils.data.DataLoader(train_set
test_loader = torch.utils.data.DataLoader(test_set,

# 标签
print(train_set.class_to_idx)
```

```
{'airplane': 0, 'automobile': 1, 'bird': 2, 'cat': 3,
'deer': 4, 'dog': 5, 'frog': 6, 'horse': 7, 'ship':
8, 'truck': 9}
```

构建网络

- 构建卷积神经网络类 `CNN`，网络结构分为特征提取器(features)和分类器(classifier)。
  - features中包含三个卷积层、三个批归一化层、ReLU激活函数层和最大池化层。
  - classifier包含了两个Dropout层以及全连接层，中间进行了一个ReLU激活函数操作。最后一个全连接层输出为10对应CIFAR-10的10个类别。
- 构建前向传播函数进行数据处理：首先经过卷积层和池化层，然后将特征展平之后经过全连接层输出。
- 进行模型实例化，检测GPU是否可用，如果GPU可用将模型参数移动到GPU上，否则还是在CPU上。

In [2]:
```python
import torch.nn as nn
# import torch.nn.functional as F

class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
```

```
        self.features = nn.Sequential(
            nn.Conv2d(3, 64, kernel_size=3, padding=
            nn.BatchNorm2d(64), # 批归一化层
            nn.ReLU(inplace=True), # 激活函数ReLU
            nn.MaxPool2d(2),  # 最大池化层，池化核大
            nn.Conv2d(64, 128, kernel_size=3, paddin
            nn.BatchNorm2d(128), # 批归一化层
            nn.ReLU(inplace=True), # 激活函数ReLU
            nn.MaxPool2d(2),  # 最大池化层，池化核大
            nn.Conv2d(128, 256, kernel_size=3, paddi
            nn.BatchNorm2d(256), # 批归一化层
            nn.ReLU(inplace=True), # 激活函数ReLU
            nn.MaxPool2d(2),  # 最大池化层，池化核大
        )
        self.classifier = nn.Sequential(
            nn.Dropout(0.5), # Dropout层，防止过拟合
            nn.Linear(256*4*4, 1024), # 全连接层1，
            nn.ReLU(inplace=True), # 激活函数ReLU
            nn.Dropout(0.5), # Dropout层，防止过拟合
            nn.Linear(1024, 10), # 全连接层2，输入特
        )

    def forward(self, x):
        x = self.features(x) # 前向传播，经过卷积层
        x = x.view(x.size(0), -1) # 将特征图展平，-1
        x = self.classifier(x) # 前向传播，经过全连
        return x

# 实例化模型
device = torch.device('cuda' if torch.cuda.is_availa
net = CNN().to(device)
```

模型训练

选择优化器和损失函数

- 使用交叉熵损失函数。
- 优化器使用Adam优化器可以快速收敛。

In [3]:
```
import torch.optim as optim
criterion = nn.CrossEntropyLoss() # 交叉熵损失函数
optimizer = optim.Adam(net.parameters(), lr=learning
```

训练函数

训练过程主要包括前向传播计算输出、计算损失、反向传播计算梯度、参数更新。

In [4]:
```
def train(epoch):
    net.train() # 设置模型为训练模式
    running_loss = 0.0 # 初始化损失值

    # 遍历训练数据集
    for batch_idx, (inputs, targets) in enumerate(tr
        inputs, targets = inputs.to(device), targets
        optimizer.zero_grad() # 清空梯度(防止梯度累加
        outputs = net(inputs) # 前向传播(计算模型输
        loss = criterion(outputs, targets) # 计算损
        loss.backward() # 反向传播(计算梯度)
        optimizer.step() # 更新参数
        running_loss += loss.item() # 累加本批次的损
    print(f'Epoch {epoch}: Loss = {running_loss/len(
```

模型保存与评估函数

该部分包括模型保存函数和模型评估函数。其中模型评估函数会计算并返回每个轮次的准确率和每个类别的精确率。

In [5]:
```
def save_models(epoch):
    torch.save(net.state_dict(), "cifar10model_{}.mo
print("Chekcpoint saved")


from sklearn.metrics import precision_score
def evaluate():
    net.eval() # 设置模型为评估模式
    correct = 0 # 初始化正确预测的数量
    total = 0 # 初始化总样本数量
    all_preds = [] # 初始化所有预测结果列表
    all_targets = [] # 初始化所有真实标签列表

    # 遍历测试数据集
    # 这里使用torch.no_grad()来关闭梯度计算，节省内存
    with torch.no_grad():
        for inputs, targets in test_loader: # 从test
            inputs, targets = inputs.to(device), tar
            outputs = net(inputs) # 前向传播(计算模型
            _, predicted = outputs.max(1) # 获取预测
            total += targets.size(0) # 累加本批次样本
            correct += (predicted == targets).sum().
            all_preds.extend(predicted.cpu().numpy()
            all_targets.extend(targets.cpu().numpy()
    acc = correct / total # 计算准确率
    precisions = precision_score(all_targets, all_pr
    print(f'Test Accuracy: {acc:.4f}')
    for i, p in enumerate(precisions):
        print(f'Class {i} Precision: {p:.4f}')
    return acc, precisions # 返回整体准确率和每类的精
```

```
Chekcpoint saved
```

主流程

该流程包括训练神经网络若干轮次、每轮次评估模型性能、记录准确率和每一类的精确率、保存最佳模型以及结果可视化。

In [6]:

```python
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline

# 记录每轮的 accuracy 和每类 precision
acc_list = [] # 存放每一轮的准确率
precision_list = [] # 存放每一轮的每个类别的精确率

if __name__ == '__main__':
    best_acc = 0.0 # 初始化最佳准确率为0.0
    # 训练和评估模型
    for epoch in range(1, num_epochs+1): # 遍历每个e
        train(epoch) # 执行训练函数
        acc, precisions = evaluate() # 执行评估函数i
        acc_list.append(acc) # 保存每一轮的准确率
        precision_list.append(precisions) # 保存每一
        # 保存最佳模型
        if acc > best_acc:
            best_acc = acc
            torch.save(net.state_dict(), 'best_cifar
    print(f'Best Test Accuracy: {best_acc:.4f}')

    # 绘制准确率和每一类精确率曲线
    epochs = list(range(1, num_epochs+1))

    plt.figure(figsize=(12, 5))
    plt.subplot(1, 2, 1)
    plt.plot(epochs, acc_list, marker='o')
    plt.title('Test Accuracy Over Epochs')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.grid(True)

    # 绘制每一类精确率曲线
    plt.subplot(1, 2, 2)
    precision_array = np.array(precision_list)  # sh
    for cls_idx in range(10):
        plt.plot(epochs, precision_array[:, cls_idx]
    plt.title('Per-Class Precision Over Epochs')
    plt.xlabel('Epoch')
```

```
    plt.ylabel('Precision')
    plt.legend()
    plt.grid(True)

    plt.tight_layout()
    plt.show()
```

```
Epoch 1: Loss = 1.5189
Test Accuracy: 0.6088
Class 0 Precision: 0.7103
Class 1 Precision: 0.7824
Class 2 Precision: 0.3527
Class 3 Precision: 0.4961
Class 4 Precision: 0.5890
Class 5 Precision: 0.5668
Class 6 Precision: 0.6037
Class 7 Precision: 0.7592
Class 8 Precision: 0.6805
Class 9 Precision: 0.6602
Epoch 2: Loss = 1.1722
Test Accuracy: 0.6694
Class 0 Precision: 0.8355
Class 1 Precision: 0.7723
Class 2 Precision: 0.5847
Class 3 Precision: 0.5195
Class 4 Precision: 0.5200
Class 5 Precision: 0.6706
Class 6 Precision: 0.7622
Class 7 Precision: 0.5776
Class 8 Precision: 0.7563
Class 9 Precision: 0.8256
Epoch 3: Loss = 1.0053
Test Accuracy: 0.7043
Class 0 Precision: 0.7801
Class 1 Precision: 0.9089
Class 2 Precision: 0.6538
Class 3 Precision: 0.4272
Class 4 Precision: 0.7035
Class 5 Precision: 0.7917
Class 6 Precision: 0.6889
Class 7 Precision: 0.8020
Class 8 Precision: 0.8091
Class 9 Precision: 0.7088
Epoch 4: Loss = 0.8988
Test Accuracy: 0.7548
Class 0 Precision: 0.7805
Class 1 Precision: 0.8984
Class 2 Precision: 0.6540
```

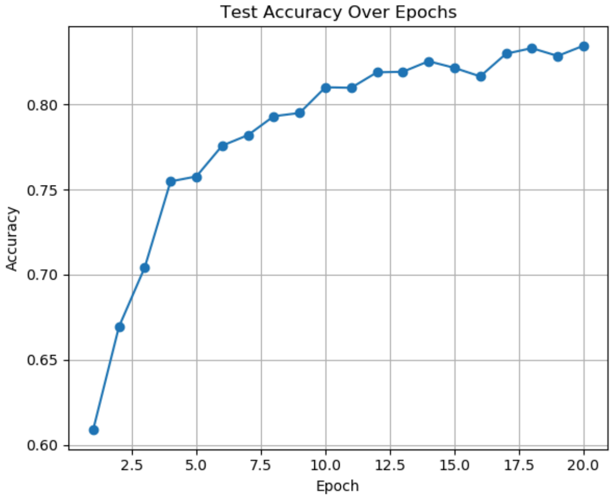
```
Class 3 Precision: 0.5458
Class 4 Precision: 0.7073
Class 5 Precision: 0.6928
Class 6 Precision: 0.7674
Class 7 Precision: 0.8109
Class 8 Precision: 0.8944
Class 9 Precision: 0.8165
Epoch 5: Loss = 0.8182
Test Accuracy: 0.7577
Class 0 Precision: 0.7089
Class 1 Precision: 0.8894
Class 2 Precision: 0.5548
Class 3 Precision: 0.6444
Class 4 Precision: 0.6911
Class 5 Precision: 0.7395
Class 6 Precision: 0.7694
Class 7 Precision: 0.8782
Class 8 Precision: 0.9116
Class 9 Precision: 0.8937
Epoch 6: Loss = 0.7571
Test Accuracy: 0.7758
Class 0 Precision: 0.7775
Class 1 Precision: 0.9159
Class 2 Precision: 0.7163
Class 3 Precision: 0.5147
Class 4 Precision: 0.7633
Class 5 Precision: 0.8877
Class 6 Precision: 0.8504
Class 7 Precision: 0.8054
Class 8 Precision: 0.8220
Class 9 Precision: 0.8578
Epoch 7: Loss = 0.7012
Test Accuracy: 0.7820
Class 0 Precision: 0.7408
Class 1 Precision: 0.8923
Class 2 Precision: 0.6260
Class 3 Precision: 0.6079
Class 4 Precision: 0.7358
Class 5 Precision: 0.8438
Class 6 Precision: 0.8267
Class 7 Precision: 0.8750
```

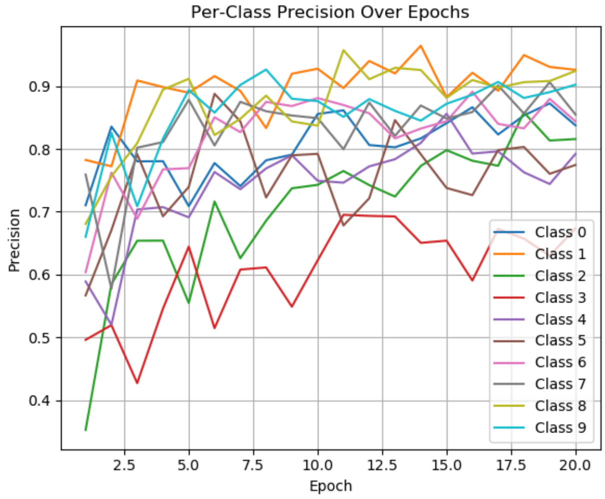
```
Class 8 Precision: 0.8480
Class 9 Precision: 0.9019
Epoch 8: Loss = 0.6550
Test Accuracy: 0.7931
Class 0 Precision: 0.7819
Class 1 Precision: 0.8332
Class 2 Precision: 0.6857
Class 3 Precision: 0.6113
Class 4 Precision: 0.7689
Class 5 Precision: 0.7228
Class 6 Precision: 0.8749
Class 7 Precision: 0.8600
Class 8 Precision: 0.8848
Class 9 Precision: 0.9264
Epoch 9: Loss = 0.6161
Test Accuracy: 0.7950
Class 0 Precision: 0.7909
Class 1 Precision: 0.9197
Class 2 Precision: 0.7373
Class 3 Precision: 0.5489
Class 4 Precision: 0.7900
Class 5 Precision: 0.7898
Class 6 Precision: 0.8682
Class 7 Precision: 0.8532
Class 8 Precision: 0.8438
Class 9 Precision: 0.8799
Epoch 10: Loss = 0.5744
Test Accuracy: 0.8101
Class 0 Precision: 0.8556
Class 1 Precision: 0.9277
Class 2 Precision: 0.7428
Class 3 Precision: 0.6227
Class 4 Precision: 0.7493
Class 5 Precision: 0.7922
Class 6 Precision: 0.8810
Class 7 Precision: 0.8489
Class 8 Precision: 0.8370
Class 9 Precision: 0.8761
Epoch 11: Loss = 0.5452
Test Accuracy: 0.8098
Class 0 Precision: 0.8614
```

```
Class 1 Precision: 0.8969
Class 2 Precision: 0.7648
Class 3 Precision: 0.6951
Class 4 Precision: 0.7462
Class 5 Precision: 0.6781
Class 6 Precision: 0.8699
Class 7 Precision: 0.7995
Class 8 Precision: 0.9572
Class 9 Precision: 0.8510
Epoch 12: Loss = 0.5213
Test Accuracy: 0.8189
Class 0 Precision: 0.8062
Class 1 Precision: 0.9400
Class 2 Precision: 0.7424
Class 3 Precision: 0.6934
Class 4 Precision: 0.7722
Class 5 Precision: 0.7214
Class 6 Precision: 0.8567
Class 7 Precision: 0.8738
Class 8 Precision: 0.9110
Class 9 Precision: 0.8793
Epoch 13: Loss = 0.4919
Test Accuracy: 0.8192
Class 0 Precision: 0.8026
Class 1 Precision: 0.9201
Class 2 Precision: 0.7241
Class 3 Precision: 0.6925
Class 4 Precision: 0.7836
Class 5 Precision: 0.8458
Class 6 Precision: 0.8168
Class 7 Precision: 0.8208
Class 8 Precision: 0.9291
Class 9 Precision: 0.8606
Epoch 14: Loss = 0.4698
Test Accuracy: 0.8254
Class 0 Precision: 0.8167
Class 1 Precision: 0.9642
Class 2 Precision: 0.7729
Class 3 Precision: 0.6505
Class 4 Precision: 0.8096
Class 5 Precision: 0.7915
```

```
Class 6 Precision: 0.8324
Class 7 Precision: 0.8691
Class 8 Precision: 0.9256
Class 9 Precision: 0.8450
Epoch 15: Loss = 0.4540
Test Accuracy: 0.8215
Class 0 Precision: 0.8411
Class 1 Precision: 0.8821
Class 2 Precision: 0.7980
Class 3 Precision: 0.6540
Class 4 Precision: 0.8560
Class 5 Precision: 0.7378
Class 6 Precision: 0.8442
Class 7 Precision: 0.8483
Class 8 Precision: 0.8814
Class 9 Precision: 0.8719
Epoch 16: Loss = 0.4338
Test Accuracy: 0.8165
Class 0 Precision: 0.8663
Class 1 Precision: 0.9212
Class 2 Precision: 0.7812
Class 3 Precision: 0.5905
Class 4 Precision: 0.7926
Class 5 Precision: 0.7264
Class 6 Precision: 0.8910
Class 7 Precision: 0.8587
Class 8 Precision: 0.9097
Class 9 Precision: 0.8863
Epoch 17: Loss = 0.4166
Test Accuracy: 0.8298
Class 0 Precision: 0.8230
Class 1 Precision: 0.8926
Class 2 Precision: 0.7732
Class 3 Precision: 0.6726
Class 4 Precision: 0.7962
Class 5 Precision: 0.7982
Class 6 Precision: 0.8399
Class 7 Precision: 0.9011
Class 8 Precision: 0.8975
Class 9 Precision: 0.9066
Epoch 18: Loss = 0.3988
```

```
Test Accuracy: 0.8331
Class 0 Precision: 0.8534
Class 1 Precision: 0.9494
Class 2 Precision: 0.8580
Class 3 Precision: 0.6572
Class 4 Precision: 0.7628
Class 5 Precision: 0.8031
Class 6 Precision: 0.8327
Class 7 Precision: 0.8563
Class 8 Precision: 0.9062
Class 9 Precision: 0.8811
Epoch 19: Loss = 0.3808
Test Accuracy: 0.8285
Class 0 Precision: 0.8725
Class 1 Precision: 0.9305
Class 2 Precision: 0.8134
Class 3 Precision: 0.6299
Class 4 Precision: 0.7441
Class 5 Precision: 0.7604
Class 6 Precision: 0.8796
Class 7 Precision: 0.9062
Class 8 Precision: 0.9081
Class 9 Precision: 0.8904
Epoch 20: Loss = 0.3770
Test Accuracy: 0.8345
Class 0 Precision: 0.8375
Class 1 Precision: 0.9259
Class 2 Precision: 0.8157
Class 3 Precision: 0.6736
Class 4 Precision: 0.7921
Class 5 Precision: 0.7745
Class 6 Precision: 0.8438
Class 7 Precision: 0.8547
Class 8 Precision: 0.9242
Class 9 Precision: 0.9022
Best Test Accuracy: 0.8345
```


Test Accuracy Over Epochs
Accuracy
Epoch
0.60
0.65
0.70
0.75
0.80
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Per-Class Precision Over Epochs
Precision
Epoch
0.4
0.5
0.6
0.7
0.8
0.9
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Class 0
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Class 7
Class 8
Class 9