实验3  
CNN图片分类任务

Due: 5月22日

# 任务描述

实现卷积神经网络CNN，并使用CIFAR-10数据集进行进行图片分类任务。

CIFAR-10是计算机视觉领域中的一个重要的数据集[[1]](#footnote-0)。 原始数据集分为训练集和测试集，其中训练集包含50000张、测试集包含300000张图像。 在测试集中，10000张图像将被用于评估，而剩下的290000张图像将不会被进行评估，包含它们只是为了防止手动标记测试集并提交标记结果。这些图片共涵盖10个类别：飞机、汽车、鸟类、猫、鹿、狗、青蛙、马、船和卡车，高度和宽度均为32像素并有三个颜色通道（RGB）。图1的左上角显示了数据集中飞机、汽车和鸟类的一些图像。本实验使用部分的CIFAR-10数据集，其中训练集共包含5000张png格式的图像，每个类别包含500张；测试集共包含5000张jpg格式的图像。

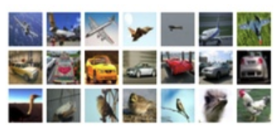


图1 CIFAR-10数据示例

# 教学要求

1. 掌握CNN模型的原理与构建
2. 了解PyTorch库，并掌握本实验内容相关的部分
3. 将CNN模型用于图片分类任务
4. 进行参数分析实验

# 实验要求

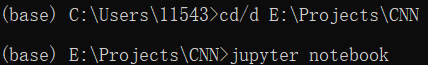
在掌握CNN原理的基础上，使用CNN在CIFAR-10数据集上进行图片分类任务。在此实验中，要求掌握以下内容：

1. 图片数据的加载和预处理，熟悉PyTorch中对数据集的处理
2. 使用PyTorch实现CNN网络架构
3. 掌握深度学习模型的训练过程，正确地进行tensor的运算
4. 计算损失函数和准确率，对模型性能进行评估
5. 调整参数设置，进行参数分析实验

# 实验步骤

1. 在Jupyter Notebook中打开实验文件（本实验在自己的电脑上即可运行）

打开方法：在Anaconda Prompt中进入实验文件所在目录，再输入jupyter notebook即可。



1. 实验环境配置

本实验需要用到torch及torchvision库，模型架构搭建及模型训练的相关操作都将使用torch库中的函数来进行，请根据自己的电脑环境（是否有GPU）进行安装并熟悉torch的基本操作。

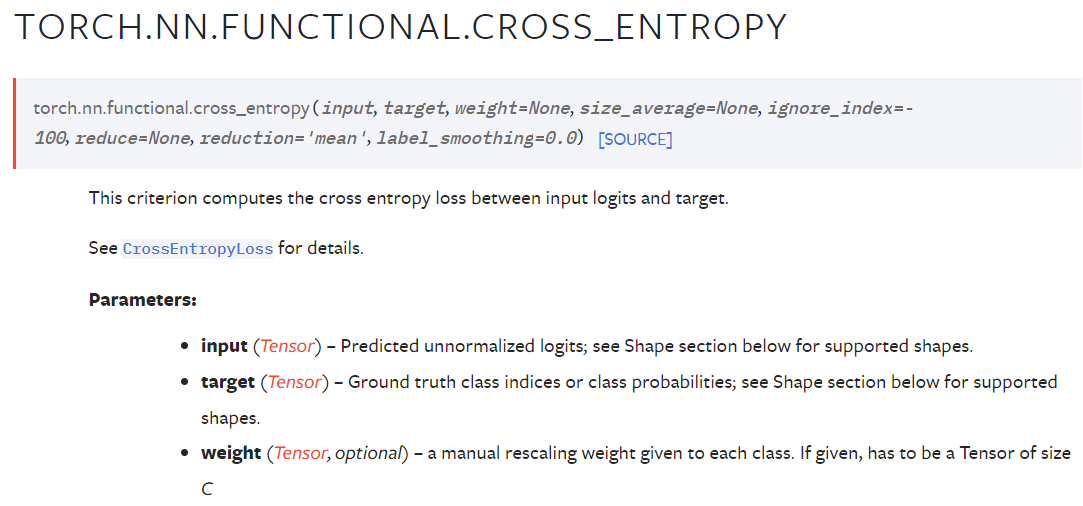
官方文档：[Quickstart — PyTorch Tutorials 2.0.1+cu117 documentation](https://pytorch.org/tutorials/beginner/basics/quickstart_tutorial.html)

1. 加载CIFAR-10数据集和数据预处理

本实验使用10%的CIFAR-10数据集，共包含10个类别的6000张图片。这里使用torchvision.datasets中的ImageFolder函数来处理数据集，将数据集处理为tensor的形式，并将像素值归一化到[0, 1]范围内。

注：

1. Tensor是深度学习中常用的数据形式，官方文档参考：[Tensors — PyTorch Tutorials 2.0.1+cu117 documentation](https://pytorch.org/tutorials/beginner/basics/tensorqs_tutorial.html)
2. CNN分类标签通常使用One-Hot编码，这里的ImageFolder函数将标签处理为0-9，是因为之后用到的交叉熵损失torch.nn.funtional.cross\_entropy函数内置了One-Hot操作（以及softmax操作）。  
   可以在官方文档中查阅函数的参数设置（这里的target为ground truth class indices）：[torch.nn.functional.cross\_entropy — PyTorch 2.0 documentation](https://pytorch.org/docs/stable/generated/torch.nn.functional.cross_entropy.html" \l "torch.nn.functional.cross_entropy)



（其他函数若有参数输入格式不明时，都可以查看官网）

1. 数据增强（该部分选做）

对图片进行数据增强，包括缩放、旋转等操作。

图像增强在对训练图像进行一系列的变化之后，生成相似但不同的训练样本，从而扩大了训练集的规模。 此外，图像增强还可以减少模型对某些属性的依赖，从而提高模型的泛化能力。 例如，可以以不同的方式裁剪图像，使感兴趣的对象出现在不同的位置，减少模型对于对象出现位置的依赖。 对图像进行缩放与旋转操作，可以使模型对不同大小与角度的模式都具有足够的经验数据。

具体说明与示例代码请见附录部分。

1. 将数据划分为训练集、验证集和测试集

1）数据文件data/cifar10中包含train和test两个文件夹，这里将train文件夹中包含的图片划分为训练集和验证集两部分。

该部分代码已给出。

1. 生成DataLoader：DataLoader是PyTorch常用的数据加载器，作用是把数据集按照一定的batch size进行打包，并允许使用多进程来加速数据读取，官方参考[torch.utils.data — PyTorch 2.0 documentation](https://pytorch.org/docs/stable/data.html)

请将代码补充完整。

1. 实现CNN架构，指定损失函数和性能指标

本实验使用PyTorch来搭建CNN模型，利用torch.nn中的函数可以添加卷积层、池化层、线性层以及ReLU函数。

1. 使用一个3x3的kernel（fileter）实现卷积运算。

请将代码补充完整。

1. 构建CNN模型：利用nn.Sequantial()，先通过一个简单模型熟悉卷积和池化运算，再构建完整的CNN模型。

在ImageClassificationBase图像分类基类中将损失和准确率的运算补充完整，在最终的CNN模型Cifar10CnnModel中将缺失的模块补充完整。

注：利用torch.nn.functional中的cross\_entropy来计算交叉熵损失。

3）编译模型

1. 使用训练集训练模型

1）批次大小（batch size）设为64（也可设为其他合适的值）。

2）设置合适数量的训练周期（epochs）和学习率（learning rate）：请将代码补充完整。

3）展示验证集的准确率，同时监控训练集与验证集损失：计算损失并绘制成折线图，请将代码补充完整，并在实验报告中较详细地说明。

1. 在测试集上评估模型性能

1）给出测试集上的标签输出

2）给出单个图片的示例性展示

3）评估模型性能

请将这部分代码补充完整，并请在实验报告中详细说明。

1. 进行参数分析实验

调整batch\_size, epochs, learning rate以及CNN模型架构的参数，在测试集上观察模型性能的变化，并给出最优的方案。

测试集的正确标签已给出，请在实验报告中详细说明这部分。

1. 可视化学习到的过滤器、激活图和误分类的例子，以便更好地了解模型的行为。

（该部分选做）

# 模型架构

以下是一个模型架构示例，具体以.ipynb中架构为基准。

1. 输入层：32x32彩色图像
2. 卷积层1：输出通道数32，过滤器大小3x3，激活函数：ReLU
3. 卷积层2：输出通道数64，过滤器大小3x3，激活函数：ReLU
4. 最大池化层1：池化大小2x2
5. 卷积层3：输出通道数128，过滤器大小3x3，激活函数：ReLU
6. 卷积层4：输出通道数128，过滤器大小3x3，激活函数：ReLU
7. 最大池化层2：池化大小2x2
8. 卷积层5：输出通道数256，过滤器大小3x3，激活函数：ReLU
9. 卷积层6：输出通道数256，过滤器大小3x3，激活函数：ReLU
10. 最大池化层3：池化大小2x2
11. 展平输出
12. 全连接层1（稠密层）：1024个神经元，激活函数：ReLU
13. 全连接层2（稠密层）：512个神经元，激活函数：ReLU
14. 输出层（稠密层）：10个神经元，激活函数：Softmax
15. 损失函数：分类交叉熵

优化器：Adam

性能指标：准确率

# 提交要求

1. 在.ipynb文件中将所有空出的代码补充完整
2. 撰写实验报告，报告内容需包含以下几部分：
3. CNN模型原理的简单阐述
4. CIFAR-10数据集的内容和数据预处理
5. CNN模型构建
6. 实验设置，包括运行环境（GPU/CPU）、参数设置和评估方法
7. 实验结果，包括模型性能和可视化结果展示，并进行相应的分析
8. 参数分析实验，调整参数设置，报告并分析模型性能
9. 提交.ipynb文件和实验报告，打包为压缩文件，并命名为“CNN实验\_学号\_姓名”

# 附录 图像增强说明[[2]](#footnote-1)

在对常用图像增广方法的探索时，我们将使用下面这个尺寸为400×500的图像作为示例。

d2l.set\_figsize()

img = d2l.Image.open('../img/cat1.jpg')

d2l.plt.imshow(img);

大多数图像增广方法都具有一定的随机性。为了便于观察图像增广的效果，定义辅助函数apply。 此函数在输入图像img上多次运行图像增广方法aug并显示所有结果。

**def** apply(img, aug, num\_rows=2, num\_cols=4, scale=1.5):

Y = [aug(img) **for** \_ **in** range(num\_rows \* num\_cols)]

d2l.show\_images(Y, num\_rows, num\_cols, scale=scale)

左右翻转图像通常不会改变对象的类别。这是最早且最广泛使用的图像增广方法之一。 接下来，我们使用transforms模块来创建RandomFlipLeftRight实例，这样就各有50%的几率使图像向左或向右翻转。

apply(img, torchvision.transforms.RandomHorizontalFlip())

另外，我们可以通过对图像进行随机裁剪，使物体以不同的比例出现在图像的不同位置。 这也可以降低模型对目标位置的敏感性。下面的代码将随机裁剪一个面积为原始面积10%到100%的区域，该区域的宽高比从0.5～2之间随机取值。 然后，区域的宽度和高度都被缩放到200像素。

shape\_aug = torchvision.transforms.RandomResizedCrop(

(200, 200), scale=(0.1, 1), ratio=(0.5, 2))

apply(img, shape\_aug)

另一种增广方法是改变颜色。 我们可以改变图像颜色的四个方面：亮度、对比度、饱和度和色调。 在下面的示例中，我们随机更改图像的亮度，随机值为原始图像的50%（1−0.5）到150%（1+0.5）之间。

apply(img, torchvision.transforms.ColorJitter(

brightness=0.5, contrast=0, saturation=0, hue=0))

我们还可以创建一个RandomColorJitter实例，并设置如何同时随机更改图像的亮度（brightness）、对比度（contrast）、饱和度（saturation）和色调（hue）。

color\_aug = torchvision.transforms.ColorJitter(

brightness=0.5, contrast=0.5, saturation=0.5, hue=0.5)

apply(img, color\_aug)

在实践中，我们将结合多种图像增广方法。比如，我们可以通过使用一个Compose实例来综合上面定义的不同的图像增广方法，并将它们应用到每个图像。

augs = torchvision.transforms.Compose([

torchvision.transforms.RandomHorizontalFlip(), color\_aug, shape\_aug])

apply(img, augs)

1. https://www.kaggle.com/c/cifar-10 [↑](#footnote-ref-0)
2. 《动手学深度学习》13.13节。 [↑](#footnote-ref-1)