机器学习实验报告

实验名称： CNN 图片分类任务

学生姓名： 唐梓烨

学生学号： 58122310

完成日期： 2024/5/22

目录

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。

数据集简介

CIFAR-10 是计算机视觉领域中的一个重要的数据集。 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 数据集示例

目标

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

(1) 数据加载

本次实验中由于原本目录中的 test 文件夹用于测试，发现测试误差异常低，为 0.08 左右。初步判断为电脑读取路径中文件的顺序不一样导致 test dataset 的标签不匹配的情况。(同样的模型在使用 torchvision.datasets 下载的数据集下测试误差正常在 0.78 左右)。因此我选择将目录中的 train 文件夹中的数据集按 9：1 分为训练集和验证集，其中验证集还同时用作测试集。在加入数据增强的情况下，我对验证集进行了数据增强，但未对测试集进行数据增强。

```
dataset = ImageFolder(data_dir+'/train')
random_seed = 42
torch.manual_seed(random_seed)
val_size = 500
train_size = len(dataset) - val_size
train_ds, val_ds = random_split(dataset, [train_size, val_size])
test_dataset = val_ds
len(train_ds), len(val_ds)
```

✓ 0.0s

```
(4500, 500)
```

```
# 定义数据增强的操作
transform = transforms.Compose([
    transforms.RandomHorizontalFlip(),  # 随机水平翻转图片
    transforms.RandomVerticalFlip(),  # 随机垂直翻转图片
    transforms.RandomResizedCrop((32, 32), scale=(0.8, 1), ratio=(0.5, 2)), # 随机裁切图片
    transforms.ToTensor()  # 将图片转换为Tensor，并归一化至[0, 1]
])
```

✓ 0.0s

```
train_ds.transform = transform
val_ds.transform = transform
dataset.transform = transform
```

✓ 0.0s

(2) 数据预处理

在数据预处理部分，我对图片进行了**数据增强**。

图片数据增强是一种常用的技术，用于扩展机器学习模型训练数据集的大小和多样性。通过对原始图像应用一系列随机变换来生成视觉上可信的新图像，数据增强可以帮助提高模型的泛化能力，即在新、未见过的数据上的表现。这对于图像识别、分类和其他视觉任务尤其重要，因为这些任务通常需要大量的数据来训练稳健的模型。

常见的图片数据增强方式有：

- **随机裁剪**（Random Cropping）

功能：随机选取图像中的一部分；作用：帮助模型学习从部分视图中识别对象。

- **水平翻转**（Horizontal Flipping）

功能：将图像水平翻转；作用：当对象的方向变化对分类结果无影响时，这可以有效增

加数据多样性。

• 垂直翻转（Vertical Flipping）

功能：将图像垂直翻转；作用：通常用于天空、海洋等自然场景的图像，但不适用于所有类型的图像，如文本图像。

• 旋转（Rotation）

功能：将图像围绕中心旋转一个随机角度；作用：增强对图像方向的鲁棒性。

• 色彩变换（Color Jittering）

功能：随机改变亮度、对比度和颜色等；作用：提高模型在不同光照和色彩条件下的性能。

• 尺度变换（Rescaling）

功能：调整图像的大小或缩放图像的某一部分；作用：模仿物体距离相机远近的不同情况。

• 噪声注入（Noise Injection）

功能：在图像中添加随机噪声；作用：提高模型对图像质量变差的容忍度。

• 透视变换（Perspective Transform）

功能：模拟从不同视角观察图像的效果；作用：增加视角的多样性，提高模型的适应性。使用数据增强的好处

• 防止过拟合：增加数据的多样性可以帮助模型在训练过程中不过度依赖于训练数据的特定特征，从而减少过拟合。

• 增强模型鲁棒性：通过训练模型处理多样化的输入，模型能更好地处理实际应用中的变化和噪声。

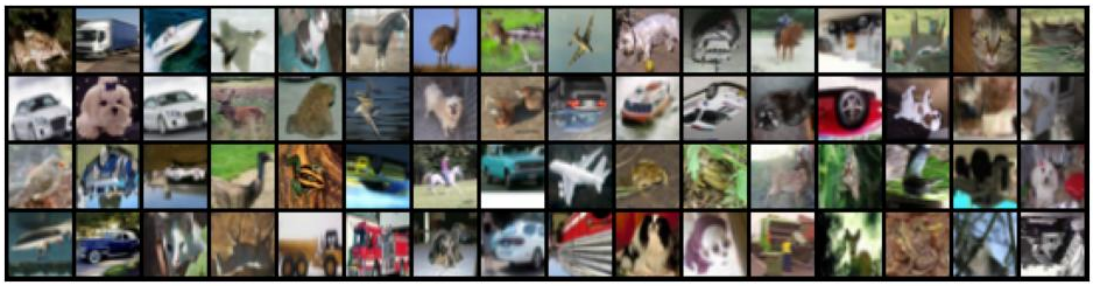
• 提高数据效用：对于数据集较小的情况，数据增强是一个非常有效的方法来扩展数据集，提高模型训练的有效性。

上述数据增强操作可以单独使用，也可以结合使用。但在使用数据增强技术时也要注意其带来的过拟合风险，避免模型过于注意与变换过后的图片。

```
# 定义数据增强的操作
transform = transforms.Compose([
    transforms.RandomHorizontalFlip(),  # 随机水平翻转图片
    transforms.RandomVerticalFlip(),  # 随机垂直翻转图片
    transforms.RandomResizedCrop((32, 32), scale=(0.8, 1), ratio=(0.5, 2)), # 随机裁切图片
    transforms.ToTensor()  # 将图片转换为Tensor，并归一化至[0, 1]
])
```

具体来说，本次实验中我采用了随机水平/垂直翻转图片、随机在 0.8~1 倍下裁切图片组合用于数据增强。

经过数据增强的数据示例：



2. 使用 PyTorch 实现 CNN 网络架构

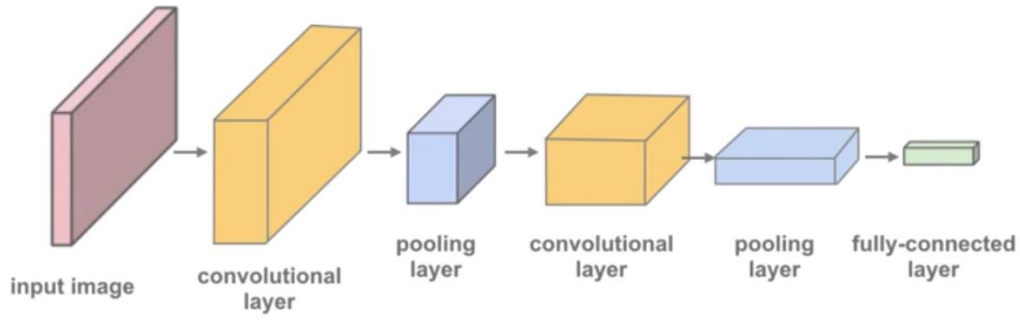
本实验使用 PyTorch 来搭建 CNN 模型，利用 torch.nn 中的函数可以添加卷积层 nn.Conv2d、池化层 nn.MaxPool2d、线性层 nn.Linea 以及 ReLU 函数 nn.ReLU。

具体来说本次实验搭建了如下图所示的 CNN 模型：

```
Cifar10CnnModel(
  (network): Sequential(
    (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU()
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (6): ReLU()
    (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): ReLU()
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU()
    (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): ReLU()
    (14): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (15): Flatten(start_dim=1, end_dim=-1)
    (16): Linear(in_features=4096, out_features=1024, bias=True)
    (17): ReLU()
    (18): Linear(in_features=1024, out_features=512, bias=True)
    (19): ReLU()
    (20): Linear(in_features=512, out_features=10, bias=True)
  )
)
```

大致框架为（2 层卷积+1 层池化）*3+三层全连接层。

卷积层让图片的维度提高（3→256）同时大小变小，池化层则起到大小减半的作用。卷积层堆叠让每一个维度的小范围 4x4 数据代表原图 32x32 的数据特征，最终再将这些数据特征展平输入到全连接神经网络，最终从 256*4*4 维度的特征中分类出 10 个类别。



实验结果

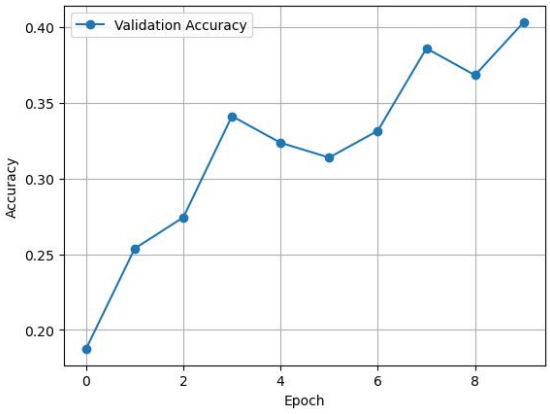
实验在 pytorch2.2.2 CUDA11.8 版本下进行

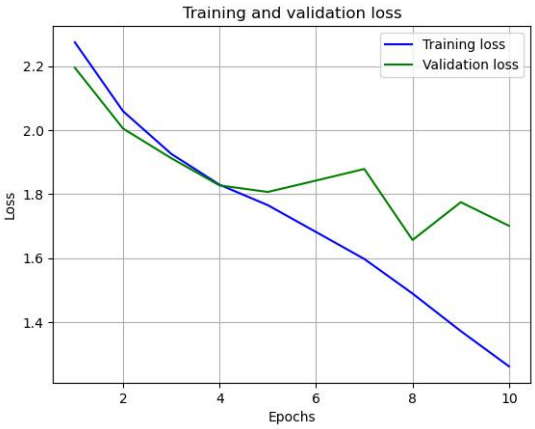
损失函数：交叉熵损失

评估方法：准确率

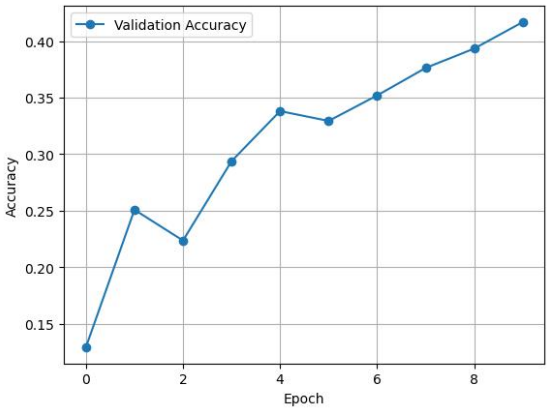
1. 计算损失函数和准确率，对模型性能进行评估

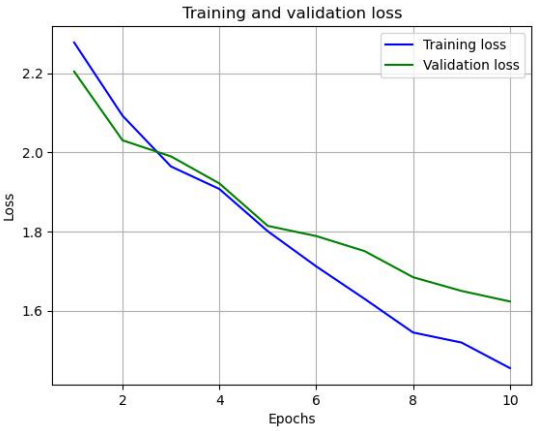
(1) lr=0.001, epochs=10, 无数据增强



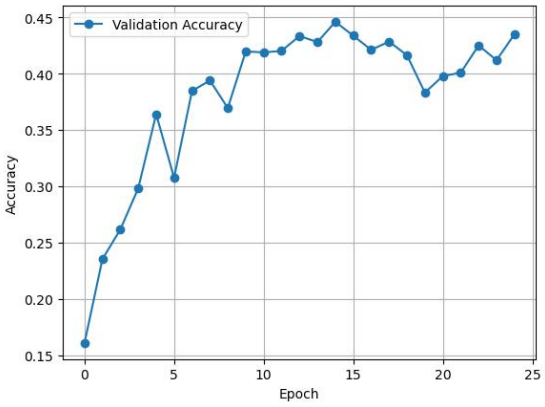


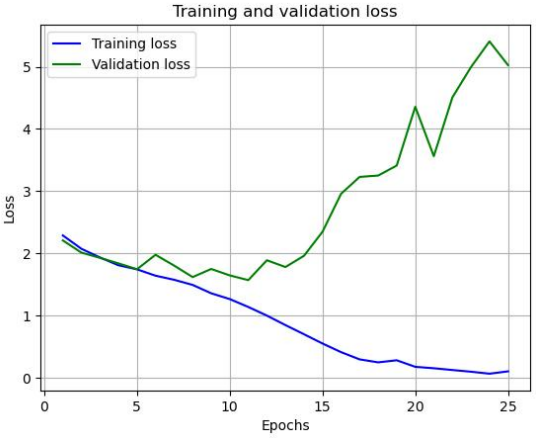
(2) lr=0.0001, epochs=10, 无数据增强



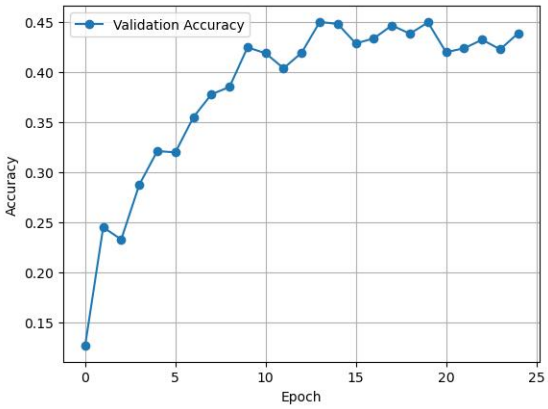


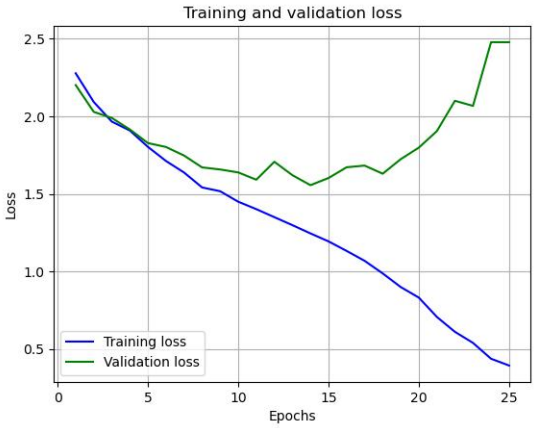
(3) lr=0.001, epochs=25, 无数据增强



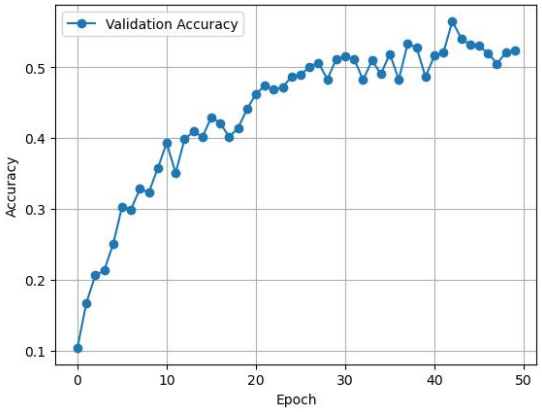


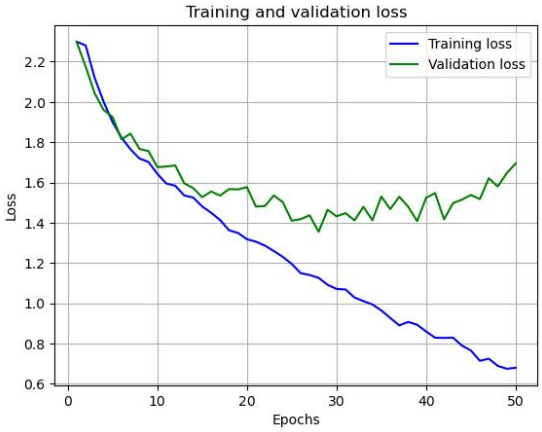
(4) lr=0.0001, epochs=25, 无数据增强



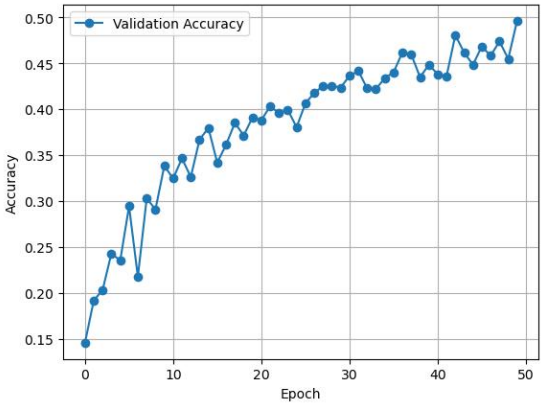


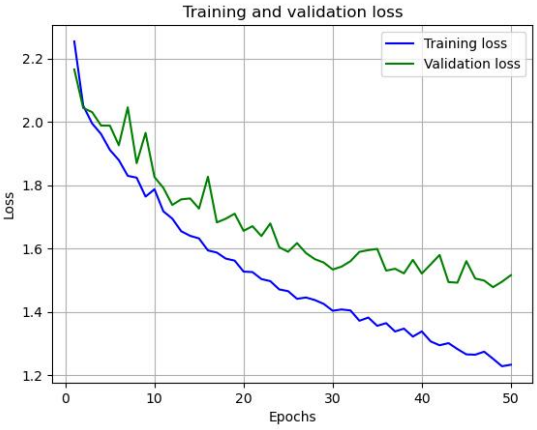
(5) lr=0.001, epochs=50, 有数据增强



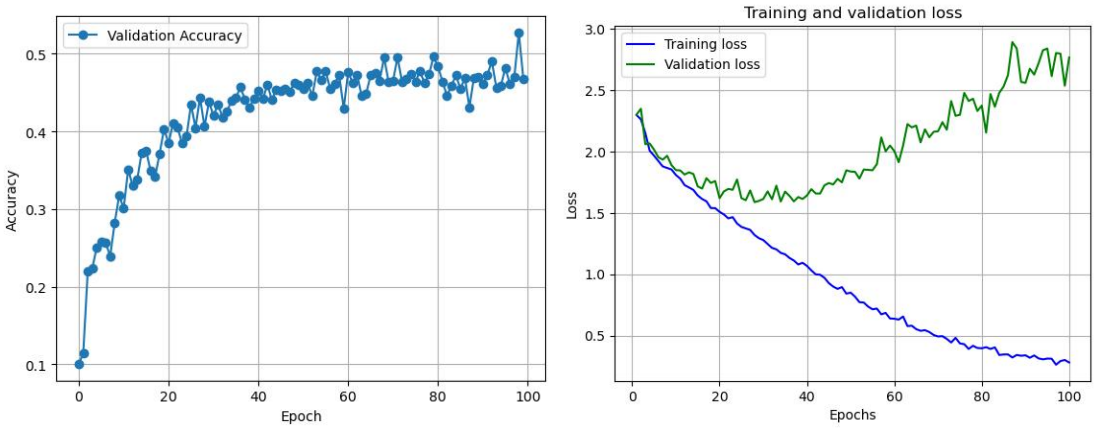


(6) lr=0.0001, epochs=50, 有数据增强

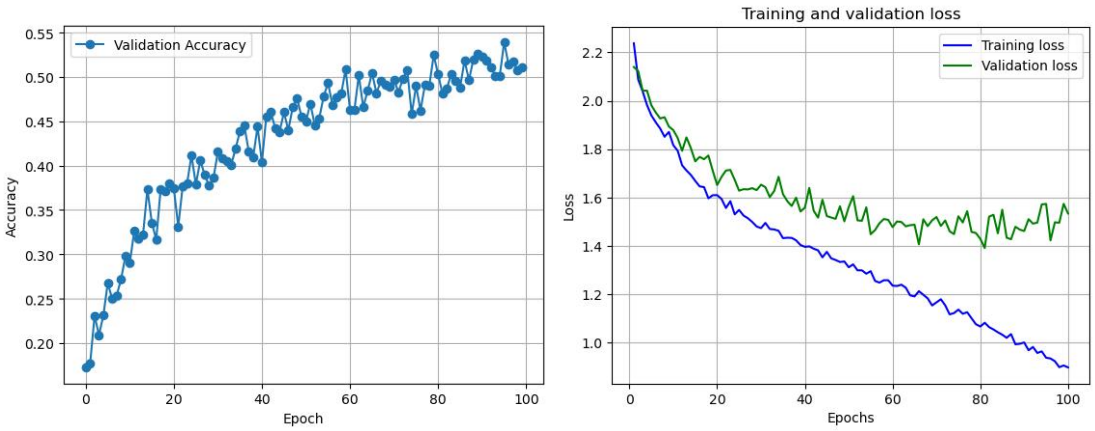




(7) lr=0.001, epochs=100, 有数据增强



(8) lr=0.0001, epochs=100, 有数据增强



2. 参数分析：

(1) 有无数据增强分析：

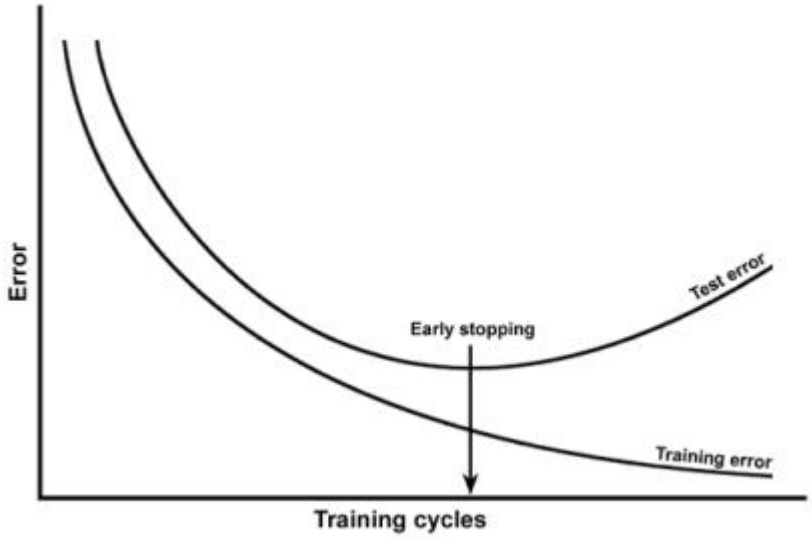
有数据增强的模型在训练时明显收敛速度更慢，在相同学习率下所需迭代次数更多。有数据增强的训练集训练的模型比没有数据增强的模型在测试集上的效果更好，能达到的最高准确率比没有数据增强的最高准确率高 5%左右。

(2) 学习率分析：

学习率直接影响模型的收敛效率与模型能否找到更优解。模型学习率越大，收敛速度越快，但模型学习率越小，越有可能靠近最优解。

(3) 迭代次数分析：

在本次实验中，训练次数对与模型是否出现过拟合影响极大。当迭代次数增加时，可以明显观察到在训练集上损失不断减小，但在验证集上的损失不降反增，出现过拟合现象。加入数据增强能时过拟合出现所需的迭代次数增加，同时减小学习率也能使过拟合出现所需的迭代次数增加。

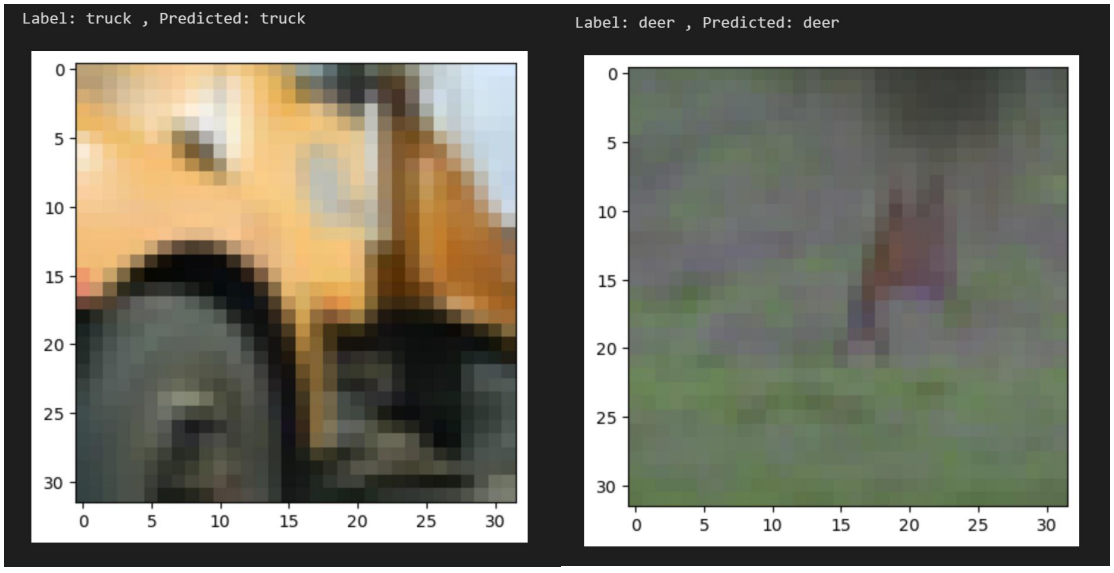


3. 最优参数：

在尝试的几种参数组合中，进行数据增强，lr=0.0001，epochs=100，batch_size=64 的训练效果最好，在测试集（未进行数据增强的验证集）准确率为 0.5457

```
test_loss: tensor(1.4984, device='cuda:0') , test_acc: tensor(0.5457)
```

正确分类结果示例：



总结

本次实验我实现了手动实现卷积核和卷积计算，加深了我对卷积的理解，为什么它能提取出图片中的特征，为什么堆叠卷积层可以扩大感受野。本次实验中我实现了调用 torch.nn 库搭建 CNN 模型，包括如何设计卷积层、池化层、全连接层、激活函数层，这锻炼了我搭建神经网络的能力。在本次实验中我实现了部分反向传播算法，交叉熵损失函数和准确度函数，加深了我对 BP 算法的理解。在本次实验中，我还对会影响模型的超参数进行了分析，加深了我对它们在模型中所起作用的理解。在本次实验中，我还自学了数据增强的相关知识，数据增强在数据预处理中是很有效的方法，可以提高我们的数据集质量和训练出来的模型性能，在后续实验和工程中也可以使用这个方法。