cifar10

2024 年 12 月 11 日

0.1 机器学习课程设计任务

1. 使用机器学习以及人工智能应用课程上学习到的机器学习相关理论与算法，使用 CIFAR-10 数据训练 10 分类的机器学习模型。
2. 将学习到的模型部署为一个 web-demo：在网页中上传新的图像，显示图像，使用学习到的模型进行预测，输出预测的分类类别。

0.2 机器学习课程设计要求

1. 熟悉并掌握机器学习全过程：数据准备、模型训练和模型选择、模型测试、模型部署
2. 熟悉 sklearn、pytorch 机器学习库的使用，掌握使用它们构建模型、解决实际问题的步骤和方法。
3. 要求尝试使用三类模型 (算法) 进行机器学习，其中至少一种使用 sklearn 库中的算法，每类模型选择最优参数。然后在三类模型中选择最优模型作为最终模型，最终模型在测试集上的准确率作为评分的依据。
4. 要求使用学习的最优模型，基于 streamlit 将模型部署为一个 web-demo，显示新上传的图像并显示分类结果。

说明（如果不做模型部署则对应相应的低档）：+ 三个模型全部使用 sklearn 构建（仅使用传统机器学习算法-对应 D 档、C 档）+ 分别使用 sklearn 和 pytorch 构建模型（结合使用深度学习算法，对应 B 档、A 档）

0.3 一、数据读取及显示

CIFAR-10 是一个包含 60000 张 32x32 彩色图片的数据集，其中包括 10 个类别的图片，每个类别有 6000 张图片。这个数据集通常用于图像分类任务的训练和测试。

CIFAR-10 数据集的 10 个类别分别是：飞机、汽车、鸟、猫、鹿、狗、青蛙、马、船和卡车。

关于 CIFAR-10 数据集的更多信息可以在以下链接找到：https://www.cs.toronto.edu/~kriz/cifar.html

[25]:
```
import pickle
import numpy as np
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

[ ]:
```
# 读取 CIFAR-10 数据集的函数
def unpickle(file):
    with open(file, 'rb') as fo:
        dict = pickle.load(fo, encoding='bytes')
    return dict
```

[63]:
```
## 读取 CIFAR-10 数据集
data_batch_1 = unpickle('cifar-10-batches-py/data_batch_1')
data_batch_2 = unpickle('cifar-10-batches-py/data_batch_2')
data_batch_3 = unpickle('cifar-10-batches-py/data_batch_3')
data_batch_4 = unpickle('cifar-10-batches-py/data_batch_4')
data_batch_5 = unpickle('cifar-10-batches-py/data_batch_5')
test_batch = unpickle('cifar-10-batches-py/test_batch')
```

[64]:
```
## 整合成所需要的数据
x_train = np.concatenate((data_batch_1[b'data'], data_batch_2[b'data'],␣
 ↪data_batch_3[b'data'], data_batch_4[b'data'], data_batch_5[b'data']), axis=0)
y_train = np.concatenate((data_batch_1[b'labels'], data_batch_2[b'labels'],␣
 ↪data_batch_3[b'labels'], data_batch_4[b'labels'], data_batch_5[b'labels']),␣
 ↪axis=0)
x_test = test_batch[b'data']
y_test = np.array(test_batch[b'labels'])
```

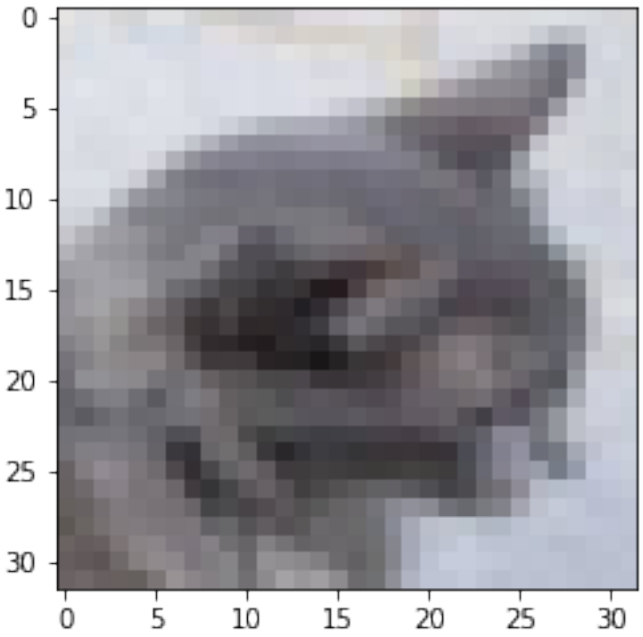
注意图像数据的组织： + NCHW 中，C 排列在外层，每个通道内像素紧挨在一起，即 'RRRRRRGGGGGGBBBBBB' 这种形式。+ NHWC 格式，C 排列在最内层，多个通道对应空间位置的像素紧挨在一起，即 'RGBRGBRGBRGBRGBRGB'

Nvidia cudnn 以及 pytorch 使用 NCHW 的形式，tensorflow 默认 NHWC 的形式（可以设为 NCHW 的形式）

[68]:
```
## 每个样本 3072 的数据 是按 （3, 32, 32）进行组织，即同 pytorch 中的 CHW 形式
img=x_train[7].reshape(3,32,32)
```

```
plt.imshow(img.transpose((1,2,0))) ## 由于 plt 显示要求图像是 hwc 的格式，所以作一下通道变换，便于显示
plt.show()
print(x_train[7].shape) ## 每个图像维度
print(x_train[0:64])    ## 每个图像的数据示例，注意是 0-255 的整数值。
                        ## 机器学习时一般会处理为 [0,1] 或者 [-1,1] 范围的数据（归一化处理）
print(y_train[0:64])
```

[68]: <matplotlib.image.AxesImage at 0x1e9c6a052c8>




```
(3072,)
[[ 34  37  44 … 49  48  58]
 [ 91  95  92 … 74 121 134]
 [208 198 164 … 163 165 157]
 …
 [255 254 254 … 99  99  97]
 [202 202 202 … 81  94 101]
 [  4   5   7 … 233 234 238]]
```

```
[6 2 5 6 3 3 8 3 3 0 0 1 7 0 8 3 4 0 2 5 8 6 5 3 9 8 5 3 5 3 8 4 3 9 3 9 6
 6 2 0 0 7 8 2 6 1 6 3 5 1 0 5 7 6 3 4 2 1 6 8 3 9 8 7]
```

0.4 二、数据预处理

[ ]:
```
## 根据所选择的机器学习算法或模型的需要进行数据预处理：如将 X 转换成所需的维度，以及归一化处理等，对 Y 作 one-hot 编码等.
##{根据需要，自行完成}
```

0.4.1 机器学习可以使用的数据集

[66]:
```
## 将训练集分成训练集和验证集
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.2, random_state=42)
print('训练集大小:', x_train.shape, y_train.shape)
print('验证集大小:', x_val.shape, y_val.shape)
print('测试集大小:', x_test.shape, y_test.shape)
```

```
训练集大小: (40000, 3072) (40000,)
验证集大小: (10000, 3072) (10000,)
测试集大小: (10000, 3072) (10000,)
```

0.4.2 具体类别信息

[67]:
```
batches_meta=unpickle('cifar-10-batches-py/batches.meta')


label_names= batches_meta[b'label_names']
label_names= [name.decode('utf-8') for name in label_names]
label_names=np.array(label_names)
print('分类类别数:', label_names.shape[0])
for index in range(0, label_names.shape[0]):
    print(index, label_names[index])
```

```
分类类别数: 10
0 airplane
1 automobile
2 bird
3 cat
4 deer
```

```
5 dog
6 frog
7 horse
8 ship
9 truck
```

0.5 三、模型训练与模型选择

每个人自由选择传统机器学习算法（ML 课程中讲到的如）或者人工神经网络（深度学习）的模型, 可以使用 skearn 和 pytorch 库。+ K 近邻、logistic 回归、SVM、决策树、随机森林、梯度 boosting 等方法 + MLP，CNN，Resnet 等

1. 选择不同的学习算法或模型，或者结合多种算法，完成模型的训练 2. 使用验证集选择不同的模型或者不同的参数

[ ]:
```
##{自行选择模型，自行编程实现，至少三个模型/算法，选择最优模型}
##{pytorch 数据准备：从 numpy 中创建 dataset，自行查阅 pytorch 文档}
```

0.5.1 模型 1 的数据准备 (按需进行)，模型训练与（模型）参数选择

[ ]:
```
##{模型 1，在此编程}
```

0.5.2 模型 2 的数据准备（按需进行），模型训练与（模型）参数选择

[ ]:
```
##{模型 2，在此编程}
```

0.5.3 模型 3 的数据准备（按需进行），模型训练与（模型）参数选择

[ ]:
```
##{模型 3，在此编程}
```

0.6 四、模型测试

- 使用上面的模型在分别在测试集上测试，输出准确率，找到最优模型。
- 将最优模型保存为文件，方便后续的模型部署。sklearn 参考：https://scikit-learn.org/stable/model_persistence.html pytroch 参考：https://pytorch.org/tutorials/beginner/saving_loading_models.html
- 输出最优模型的准确率，作为评分依据之一。

[ ]:
```
#{模型测试-自行编程实现}
```

0.7 五、模型部署

自行学习 streamlit 的使用以及编程方法

使用上面训练得到的模型，采用streamlit构建一个 web demo, 上传图像，输出图像对应的分类名称。参考：+ Streamlit documentation + https://stevensmiley1989.medium.com/train-deploy-yolov7-to-streamlit-5a3e925690a9