# 模式识别及应用

# 大作业报告


院　　系：信息与通信工程学院

班　　级：2019211112

学　　号：2021523016

类　　别：访学交流生

姓　　名：徐川峰

指导老师：赵志诚 齐勇刚

2022 年 06 月 06 日

## 1、研究背景与问题描述

### ①数据集简介

狗作为宠物与人类生活密切相关，日益成为一些家庭的重要成员。与此同时，与狗相关的伤害事件和不文明行为频繁见诸于新闻报道，对城市治理提出了新的挑战。这迫切需要采用现代视觉技术来识别狗以及细粒度的品种信息，以加强城市中狗的管理。物体细分类是一个重要的视觉问题，要求从细微的类间差异中区分不同的子类。对于狗的细分类而言，而由于地理隔离或杂交的缘故，同一品种的狗之间可能存在较大的差异。该数据集中共包含 5 个品种的狗，鉴于真实情况中不同种类狗的数量分布不均匀，数据集中的类别也存在分布不均匀的情况。

### ②VGG16 网络

图像分类是指根据各自在图像信息中所反映的不同特征，把不同类别的目标区分开来的图像处理方法。它利用计算机对图像进行定量分析，把图像或图像中的每个像元或区域划归为若干个类别中的某一种，以代替人的视觉判读。在本次大作业中，我使用了 pytorch 编程框架实现经典且强大的 VGG16 网络来进行狗狗的图像分类任务。

VGG 是 Oxford 的 Visual Geometry Group 的组提出的。该网络是在 ILSVRC 2014 上的相关工作，主要工作是证明了增加网络的深度能够在一定程度上影响网络最终的性能。VGG 有两种结构，分别是 VGG16 和 VGG19，两者并没有本质上的区别，只是网络深度不一样。

VGG16 相比 AlexNet 的一个改进是采用连续的几个 3x3 的卷积核代替 AlexNet 中的较大卷积核（11x11，7x7，5x5）。对于给定的感受野（与输出有关的输入图片的局部大小），采用堆积的小卷积核是优于采用大的卷积核，因为多层非线性层可以增加网络深度来保证学习更复杂的模式，而且代价还比较小（参数更少）。

VGG16 包含了 16 个隐藏层（13 个卷积层和 3 个全连接层），如下图中的 D 列所示。

| ConvNet Configuration | | | | | |
|---|---|---|---|---|---|
| A | A-LRN | B | C | D | E |
| 11 weight layers | 11 weight layers | 13 weight layers | 16 weight layers | 16 weight layers | 19 weight layers |
| input (224 × 224 RGB image) | | | | | |
| conv3-64 | conv3-64<br>**LRN** | conv3-64<br>**conv3-64** | conv3-64<br>conv3-64 | conv3-64<br>conv3-64 | conv3-64<br>conv3-64 |
| maxpool | | | | | |
| conv3-128 | conv3-128 | conv3-128<br>**conv3-128** | conv3-128<br>conv3-128 | conv3-128<br>conv3-128 | conv3-128<br>conv3-128 |
| maxpool | | | | | |
| conv3-256<br>conv3-256 | conv3-256<br>conv3-256 | conv3-256<br>conv3-256 | conv3-256<br>conv3-256<br>**conv1-256** | conv3-256<br>conv3-256<br>**conv3-256** | conv3-256<br>conv3-256<br>**conv3-256** |
| maxpool | | | | | |
| conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512<br>**conv1-512** | conv3-512<br>conv3-512<br>**conv3-512** | conv3-512<br>conv3-512<br>**conv3-512** |
| maxpool | | | | | |
| conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512<br>**conv1-512** | conv3-512<br>conv3-512<br>**conv3-512** | conv3-512<br>conv3-512<br>**conv3-512** |
| maxpool | | | | | |
| FC-4096 | | | | | |
| FC-4096 | | | | | |
| FC-1000 | | | | | |
| soft-max | | | | | |

https://blog.csdn.net/qdulij

2. 数据分析与代码实现

①数据预处理

```
import numpy as np
import pandas as pd
import os
from PIL import Image


def luoData(path):
    lis =
os.listdir('C:\\Users\\xuduoduo\\Desktop\\dog_divide\\data')
    print(lis)
    labels = []
    imgData = []
    for ele in lis:
        sample = os.listdir(path + '/' + ele)
        for i in sample:
            print(path + "/" + ele + '/' + i)
            if ele == "n000001-Shiba_Dog":
                img = Image.open(path + "/" + ele + '/' + i)
                img = img.resize((128, 128), Image.ANTIALIAS)
                img = np.array(img)
                img = img / 255.0
                imgData.append(img)
                labels.append(0)
            if ele == 'n000002-French_bulldog':
                img = Image.open(path + "/" + ele + '/' + i)
                img = img.resize((128, 128), Image.ANTIALIAS)
                img = np.array(img)
                img = img / 255.
                imgData.append(img)
                labels.append(1)
            if ele == 'n000003-Siberian_husky':
                img = Image.open(path + "/" + ele + '/' + i)
                img = img.resize((128, 128), Image.ANTIALIAS)
                img = np.array(img)
                img = img / 255.
                imgData.append(img)
                labels.append(2)
            if ele == 'n000004-Pomeranian':
                img = Image.open(path + "/" + ele + '/' + i)
                img = img.resize((128, 128), Image.ANTIALIAS)
                img = np.array(img)
```

```
                img = img / 255.
                imgData.append(img)
                labels.append(3)
            if ele == 'n000005-golden_retriever':
                img = Image.open(path + "/" + ele + '/' + i)
                img = img.resize((128, 128), Image.ANTIALIAS)
                img = np.array(img)
                img = img / 255.
                imgData.append(img)
                labels.append(4)

    imgData = np.array(imgData)
    labels = np.array(labels)

    return imgData, labels
```

②模型搭建（基于 VGG16 架构进行预测）

```
class VGG16(Model):
    def __init__(self):
        super(VGG16, self).__init__()
        self.c1 = Conv2D(filters=64, kernel_size=(3, 3),
padding='same')  # 卷积层 1
        self.b1 = BatchNormalization()  # BN 层 1
        self.a1 = Activation('relu')  # 激活层 1
        self.c2 = Conv2D(filters=64, kernel_size=(3, 3),
padding='same', )
        self.b2 = BatchNormalization()  # BN 层 1
        self.a2 = Activation('relu')  # 激活层 1
        self.p1 = MaxPool2D(pool_size=(2, 2), strides=2,
padding='same')
        self.d1 = Dropout(0.2)  # dropout 层

        self.c3 = Conv2D(filters=128, kernel_size=(3, 3),
padding='same')
        self.b3 = BatchNormalization()  # BN 层 1
        self.a3 = Activation('relu')  # 激活层 1
        self.c4 = Conv2D(filters=128, kernel_size=(3, 3),
padding='same')
        self.b4 = BatchNormalization()  # BN 层 1
        self.a4 = Activation('relu')  # 激活层 1
        self.p2 = MaxPool2D(pool_size=(2, 2), strides=2,
padding='same')
        self.d2 = Dropout(0.2)  # dropout 层

        self.c5 = Conv2D(filters=256, kernel_size=(3, 3),
```

```
padding='same')
        self.b5 = BatchNormalization()  # BN层1
        self.a5 = Activation('relu')  # 激活层1
        self.c6 = Conv2D(filters=256, kernel_size=(3, 3),
padding='same')
        self.b6 = BatchNormalization()  # BN层1
        self.a6 = Activation('relu')  # 激活层1
        self.c7 = Conv2D(filters=256, kernel_size=(3, 3),
padding='same')
        self.b7 = BatchNormalization()
        self.a7 = Activation('relu')
        self.p3 = MaxPool2D(pool_size=(2, 2), strides=2,
padding='same')
        self.d3 = Dropout(0.2)

        self.c8 = Conv2D(filters=512, kernel_size=(3, 3),
padding='same')
        self.b8 = BatchNormalization()  # BN层1
        self.a8 = Activation('relu')  # 激活层1
        self.c9 = Conv2D(filters=512, kernel_size=(3, 3),
padding='same')
        self.b9 = BatchNormalization()  # BN层1
        self.a9 = Activation('relu')  # 激活层1
        self.c10 = Conv2D(filters=512, kernel_size=(3, 3),
padding='same')
        self.b10 = BatchNormalization()
        self.a10 = Activation('relu')
        self.p4 = MaxPool2D(pool_size=(2, 2), strides=2,
padding='same')
        self.d4 = Dropout(0.2)

        self.c11 = Conv2D(filters=512, kernel_size=(3, 3),
padding='same')
        self.b11 = BatchNormalization()  # BN层1
        self.a11 = Activation('relu')  # 激活层1
        self.c12 = Conv2D(filters=512, kernel_size=(3, 3),
padding='same')
        self.b12 = BatchNormalization()  # BN层1
        self.a12 = Activation('relu')  # 激活层1
        self.c13 = Conv2D(filters=512, kernel_size=(3, 3),
padding='same')
        self.b13 = BatchNormalization()
        self.a13 = Activation('relu')
        self.p5 = MaxPool2D(pool_size=(2, 2), strides=2,
```

```
padding='same')
        self.d5 = Dropout(0.2)

        self.flatten = Flatten()
        self.f1 = Dense(512, activation='relu')
        self.d6 = Dropout(0.2)
        self.f2 = Dense(512, activation='relu')
        self.d7 = Dropout(0.2)
        self.f3 = Dense(5, activation='softmax')

    def call(self, x):
        x = self.c1(x)
        x = self.b1(x)
        x = self.a1(x)
        x = self.c2(x)
        x = self.b2(x)
        x = self.a2(x)
        x = self.p1(x)
        x = self.d1(x)

        x = self.c3(x)
        x = self.b3(x)
        x = self.a3(x)
        x = self.c4(x)
        x = self.b4(x)
        x = self.a4(x)
        x = self.p2(x)
        x = self.d2(x)

        x = self.c5(x)
        x = self.b5(x)
        x = self.a5(x)
        x = self.c6(x)
        x = self.b6(x)
        x = self.a6(x)
        x = self.c7(x)
        x = self.b7(x)
        x = self.a7(x)
        x = self.p3(x)
        x = self.d3(x)

        x = self.c8(x)
        x = self.b8(x)
        x = self.a8(x)
```

```
        x = self.c9(x)
        x = self.b9(x)
        x = self.a9(x)
        x = self.c10(x)
        x = self.b10(x)
        x = self.a10(x)
        x = self.p4(x)
        x = self.d4(x)

        x = self.c11(x)
        x = self.b11(x)
        x = self.a11(x)
        x = self.c12(x)
        x = self.b12(x)
        x = self.a12(x)
        x = self.c13(x)
        x = self.b13(x)
        x = self.a13(x)
        x = self.p5(x)
        x = self.d5(x)

        x = self.flatten(x)
        x = self.f1(x)
        x = self.d6(x)
        x = self.f2(x)
        x = self.d7(x)
        y = self.f3(x)
        return y
```

③模型训练

```
import tensorflow as tf
import os
import numpy as np
from matplotlib import pyplot as plt
from tensorflow.keras.layers import Conv2D, BatchNormalization,
Activation, MaxPool2D, Dropout, Flatten, Dense
from tensorflow.keras import Model
from makedata import luoData
from sklearn.model_selection import train_test_split

np.set_printoptions(threshold=np.inf)

data, label =
luoData('C:\\Users\\xuduoduo\\Desktop\\dog_divide\\data')
# 数据集打乱
```

```
np.random.seed(116)
np.random.shuffle(data)
np.random.seed(116)
np.random.shuffle(label)

x_train, x_test, y_train, y_test = train_test_split(data, label,
test_size=0.3, random_state=116)


class VGG16(Model)...

model = VGG16()

model.compile(optimizer='adam',

loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False)
,
            metrics=['sparse_categorical_accuracy'])

checkpoint_save_path = "./checkpoint/VGG16.ckpt"
if os.path.exists(checkpoint_save_path + '.index'):
    print('-------------load the model-----------------')
    model.load_weights(checkpoint_save_path)

cp_callback =
tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint_save_path,
                                                save_weights_only=True,
                                                save_best_only=True)

history = model.fit(x_train, y_train, batch_size=32, epochs=50,
validation_data=(x_test, y_test), validation_freq=1,
                    callbacks=[cp_callback])
model.summary()
```

### 3. 模式验证与结果保存

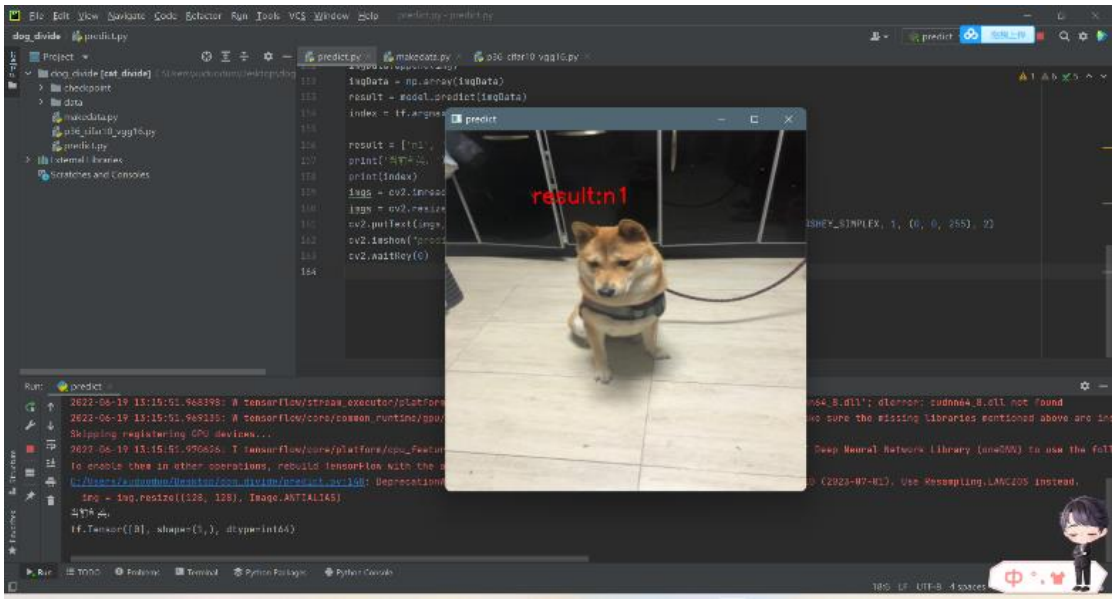
```
# 显示训练集和验证集的 acc 和 loss 曲线
acc = history.history['sparse_categorical_accuracy']
val_acc = history.history['val_sparse_categorical_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

plt.subplot(1, 2, 1)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.title('Training Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.title('Training Loss')
plt.legend()
plt.show()
```

loss 为训练集损失值，acc 为训练集准确率；val_loss 为测试集损失值，val_acc 为测试集准确率。可以看到结果还是比较理想的，使用 accuracy 指标对模型进行评估，所训练的模型在测试集上可以高达 83.3%。

```
------------load the model-----------------
Epoch 1/20
  3/2615 [..............................] - ETA: 40:08 - loss: 1.5862 - sparse_categorical_accuracy: 0.8333
```

使用 Matplotlib 绘制以下训练集和测试集的准确率曲线，可以更清晰的看出，训练过程的变化。

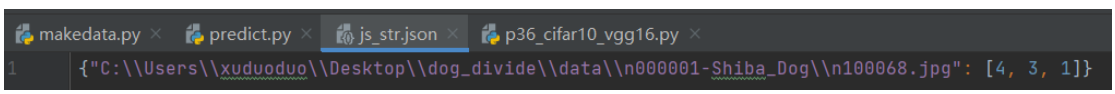
使用保存训练好的模型对真实数据进行测试，识别结果准确无误。




最后，将测试数据集的分类结果保存为一个 json 文件，json 文件包含一个字典，字典的 key 为图片相对路径及名称，字典的 value 为该图片分的类结果按置信度排序的前 3 个标签。预测的标签为 1-5 之间的数字。

```
result_s = ['n1', 'n2', 'n3', 'n4', 'n5']
dicts = {}
for loc,ele in enumerate(result_s):
    dicts[loc] = result[0][loc]
# 按照字典的值进行排序
a1 = sorted(dicts.items(), key=lambda x: x[1],reverse=True)
value_key = dict(a1).keys()
value_key = list(value_key)[0:3]
print(value_key)

res_dict = {path:value_key}
with open('js_str.json', 'w') as json_data:
    json.dump(res_dict, json_data, ensure_ascii=False)
print('当前种类：')
print(index)
imgs = cv2.imread(path)
imgs = cv2.resize(imgs, (500, 500))
cv2.putText(imgs, "result:{}".format(result[index[0]]), (120, 100),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
cv2.imshow("predict", imgs)
cv2.waitKey(0)
```

示例如下：



## 4. 实验总结与学习见解

本次大作业设计实现了一个基于 VGG16 网络的 CNN 方法，并使用 pytorch 进行实现。模型结构虽然较为复杂,但 VGG16 的优势和强大能使模型实现起来较为简单，参数较少。

这是我第一次接触深度学习，由于对 Python 语言掌握的不是很熟练，勉强算是刚刚入门的小白，所以我在实验过程中遇到了非常多的困难。从连续报错，到首次跑通程序，再到训练的过程中能够看到一个较高的 accuracy，每走一步对我来说都是新的尝试与挑战。我也通过查询各方面的资料，让自己头脑中模糊的概念逐渐清晰，使自己十分稚嫩的实验一步步完善起来，每一次改善都是我学习的收获。

在本次大作业中，虽然程序在验证集上能够正常预测，但训练集上正确率较低。可能是过多的预处理导致训练集的分布产生了变化，所以使得训练集的准确率低于验证集。因为实验环境有限，电脑扛不住多次调节参数并进行训练对比，所以很遗憾没能进一步寻找到针对本模型提高准确率的有效方法。如何提高图像分类的准确率，也是我接下来要去查缺补漏、认真学习研究的方向。若课后有条件，我也会去尝试使用其他语言，或无监督学习（聚类）与监督学习相融合的方法，进行实验去对比不同的结果。

最后非常感谢本学期的赵老师的教导。赵老师授课认真细致，见解独到深入，讲课时的激情感染到了我，让我对于模式识别这门课产生了浓厚的兴趣。

（期末大作业真的尽力而为且认真对待了 QAQ 希望和蔼可亲的赵老师能酌情给分 QAQ 谢谢老师！您辛苦了！）