队伍编号	MCB2301959
赛道	A

基于图形增强的 CNN-PCA-SVM 模型在图像分类中的应用

摘　　要

最后，本文对所建立的模型的优缺点进行了中肯的评价、提出了模型的改进措施以及对模型进行了一定推广。

关键词：特征工程；计算机视觉；支持向量机；评分预测；可视化评估

目录

一、问题的提出

1.1 问题背景

坑洼道路的检测与识别工作是推动自动无人驾驶、地质勘探、航天科学及自然灾害等领域研究和应用的不可或缺的计算机视觉任务。然而传统的分类算法往往因坑洼图像的复杂及多变性，不能取得较好的效果。近年来，深度学习邻域技术的发展为坑洼道路的检测、为当前存在的亟待解决的问题提供了新的思想。

在实际工程应用中，非结构化的坑洼道路检测将遇到复杂多样的环境，例如，道路周围环境存在差异、光照条件变化等，这无疑增加了基于视觉检测道路的难度[1]。道路坑洼的检测是实现智能车辆导航的基础，只有快速准确地检测出道路的可行驶区域才能够真正实现自动驾驶。

深度学习拥有很强的特征提取与表示能力，可从图像中提取重要特征。于坑洼道路检测和识别而言，其可识别坑洼的轮廓、纹理、形态等特征，并将上述特征转换为更易分类的表现形式。在深度学习的基础上，迁移学习、知识蒸馏等技术可进一步提升分类性能以更加准确对新出现的道路图像自动识别。

1.2 问题要求

- **问题一：**基于图像文件，提取图像特征，以“正常”和“坑洼”为特征建立一个识别率高、速度快且分类准确的模型。
- **问题二：**对问题一中所建立的模型进行训练，并对其进行多维度的评估分析。
- **问题三：**利用训练模型识别测试集中的坑洼图像，并展示结果。

二、问题的分析

2.1 问题的整体分析

该题是一个基于计算机视觉的坑洼道路的图像数据分析、预测类问题。

从分析目的看，

从数据来源、特征看，

从模型的选择看，

从编程软件的选择看，本题为图像大数据分析类，需要对大量的图片数据进行分析，并依据设问建立合适的模型，对正常及坑洼道路的图像进行分类预测，因此我们选择使用 Python Jupyter 对问题进行求解，其交互式的编程范式及轻量化，方便且高效。

从 Python 第三方库的选择看，

2.2 问题一的分析

问题一的核心目的在于**对原数据集进行重采样，并进行更深层次的分析**。

2.3 问题二的分析

问题二的核心目的在于**为移动公司撰写一份非技术性报告，为其提供合理性建议，从而为客户提供更好的服务**。

三、符号说明

符号	符号说明
μ	样本平均值
σ	样本方差
x_{standard}	经过标准化后的数据
$R(x)_{m\times n}$	经过某项处理后的数据特征集
ρ	皮尔逊相关系数
x'	经过某项处理后的数据
$Gini$	样本集合基尼系数
$\hat{y}$	预测值
$L^{(t)}$	目标函数
Ω	叶节点正则项惩罚系数
P	某事件发生的概率
ω	权重

四、模型的假设

本文对于模型的假设与初赛假设一致，如下：

- **假设一**：语音与上网业务的八项评分中，存在个别用户乱评、错评现象；
- **假设二**：除个别用户的部分评分外，其余所有数据真实且符合实际情况；
- **假设三**：用户评分还受到除附件中因素之外的因素的影响；
- **假设四**：给定的数据集可全面体现用户整体情况；
- **假设五**：对于同一业务，学习数据与预测数据的内在规律是一致的。

五、模型的建立与求解

参考文献

[1] 曹江华. 复杂背景下非结构化道路可行驶区域检测研究 [D]. 浙江科技学院,2021.

[2] CSDN.【数据预处理】sklearn 实现数据预处理（归一化、标准化）[EB/OL]. https://blog.csdn.net/weixin_44109827/article/details/124786873.

[3] 王殿武, 赵云斌, 尚丽英, 王凤刚, 张震. 皮尔逊相关系数算法在 B 油田优选化学防砂措施井的应用 [J]. 精细与专用化学品,2022,30(07):26-28.DOI:10.19482/j.cn11-3237.2022.07.07.

[4] 饶雷, 冉军, 陶建权, 胡号朋, 吴沁, 熊圣新. 基于随机森林的海上风电机组发电机轴承异常状态监测方法 [J]. 船舶工程,2022,44(S2):27-31.DOI:10.13788/j.cnki.cbgc.2022.S2.06.

[5] 陈振宇, 刘金波, 李晨, 季晓慧, 李大鹏, 黄运豪, 狄方春, 高兴宇, 徐立中. 基于 LSTM 与 XGBoost 组合模型的超短期电力负荷预测 [J]. 电网技术,2020,44(02):614-620.DOI:10.13335/j.1000-3673.pst.2019.1566.

[6] 杨贵军, 徐雪, 赵富强. 基于 XGBoost 算法的用户评分预测模型及应用 [J]. 数据分析与知识发现,2019,3(01):118-126.

[7] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16). Association for Computing Machinery, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785.

[8] 张著英, 黄玉龙, 王翰虎. 一个高效的 KNN 分类算法 [J]. 计算机科学,2008(03):170-172.

[9] 汪海燕, 黎建辉, 杨风雷. 支持向量机理论及算法研究综述 [J]. 计算机应用研究,2014,31(05):1281-1286.

[10] 马晓君, 沙靖岚, 牛雪琪. 基于 LightGBM 算法的 P2P 项目信用评级模型的设计及应用 [J]. 数量经济技术经济研究,2018,35(05):144-160.DOI:10.13653/j.cnki.jqte.20180503.001.

[11] 唐敏, 张宇浩, 邓国强. 高效的非交互式隐私保护逻辑回归模型 [J/OL]. 计算机工程:1-11[2023-01-04].DOI:10.19678/j.issn.1000-3428.0065549.

[12] 史佳琪, 张建华. 基于多模型融合 Stacking 集成学习方式的负荷预测方法 [J]. 中国电机工程学报,2019,39(14):4032-4042.DOI:10.13334/j.0258-8013.pcsee.181510.

[13] A.Tharwat, Applied Computing and Informatics (2018). https://doi.org/10.1016/j.aci.2018.08.003.

附　录

[A] 图示

[B] 支撑文件列表

支撑文件列表如下（列表中不包含原始数据集）：

文件（夹）名	描述
附件 6：result.xlsx	用户评分预测结果
语音业务高分组描述.csv	语音业务高分组数据描述
语音业务低分组描述.csv	语音业务低分组数据描述
上网业务高分组描述.csv	上网业务高分组数据描述
上网业务低分组描述.csv	上网业务低分组数据描述
语音业务 Sample.csv	语音业务剔除不合理评分后样本数据
上网业务 Sample.csv	上网业务剔除不合理评分后样本数据
语音业务词云.txt	语音业务词云图文本内容
上网业务词云.txt	上网业务词云图文本内容
词云.txt	语音及上网业务综合词云图文本内容
词云图.py	语音及上网业务综合词云图代码文件
语音业务用户分析.ipynb	语音业务评分特征分析 Jupyter 文件
上网业务用户分析.ipynb	上网业务评分特征分析 Jupyter 文件
语音业务数据分析.ipynb	语音业务模型建立 Jupyter 文件
上网业务数据分析.ipynb	上网业务模型建立 Jupyter 文件
语音业务用户分析.html	语音业务评分特征分析运行结果
上网业务用户分析.html	上网业务评分特征分析运行结果
语音业务数据分析.html	语音业务模型建立运行结果
上网业务数据分析.html	上网业务模型建立运行结果
bg.jpg	词云底图
wordcloud.png	语音及上网业务综合词云图
figuresOne	语音业务所有图示文件夹
figuresTwo	上网业务所有图示文件夹

[C] **使用的软件、环境**

为解决该问题，我们所使用的主要软件有：

- TeX Live 2022
- Visual Studio Code 1.83.1
- WPS Office 2023 秋季更新（15398）
- Python 3.10.4 [MSC v.1929 64 bit (AMD64)] on win32
- Pycharm 2023.2.3 (Professional Edition)

Python 环境下所用使用到的库及其版本如下：

库	**版本**	**库**	**版本**
copy	内置库	missingno	0.5.1
jieba	0.42.1	mlxtend	0.20.2
jupyter	1.0.0	numpy	1.22.4+mkl
jupyter-client	7.3.1	openpyxl	3.0.10
jupyter-console	6.4.3	pandas	1.4.2
jupyter-contrib-core	0.4.0	pyecharts	1.9.1
jupyter-contrib-nbextensions	0.5.1	scikit-learn	0.22.2.post1
jupyter-core	4.10.0	seaborn	0.11.2
jupyter-highlight-selected-word	0.2.0	sklearn	0.0
jupyterlab-pygments	0.2.2	snapshot_phantomjs	0.0.3
jupyterlab-widgets	1.1.0	warnings	内置库
jupyter-latex-envs	1.4.6	wordcloud	1.8.1
jupyter-nbextensions-configurator	0.5.0	xgboost	1.6.1
matplotlib	3.5.2	yellowbrick	1.4

[D] 问题解决源程序

D.1 Data Preprocessing 数据预处理

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import os
import shutil

# 指定目录"DATA"
path = "DATA"

# 在"DATA"文件夹中创建"normal"和"potholes"文件夹
os.mkdir(os.path.join(path, "normal"))
os.mkdir(os.path.join(path, "potholes"))

# 读取"DATA"文件夹，若文件名中含有"normal"，则将其放置于"normal"文件夹中，否则放置于"
    potholes"文件夹中
files = os.listdir(path)
for file in files:
    if "normal" in file:
        shutil.move(os.path.join(path, file), os.path.join(path, "normal"))
    else:
        shutil.move(os.path.join(path, file), os.path.join(path, "potholes"))
```

D.2 Comparative Analysis Of Normal And Potholes 正常与坑洼道路的比较分析

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import cv2
import numpy as np
import matplotlib.pyplot as plt

# In[2]:


normalImg = cv2.imread('DATA\\normal\\normal133.jpg')
potholesImg = cv2.imread('DATA\\potholes\\potholes1.jpg')

```

```
# In[3]:


plt.rcParams['font.sans-serif'] = ['Times New Roman']
plt.rcParams['axes.unicode_minus'] = False


# In[4]:


def cv_show(img):
    b, g, r = cv2.split(img)
    img = cv2.merge([r, g, b])
    plt.imshow(img)


# In[5]:


cv_show(normalImg)

# In[6]:


cv_show(potholesImg)

# In[7]:


color = ('b', 'g', 'r')

for i, col in enumerate(color):
    histr = cv2.calcHist([normalImg], [i], None, [256], [0, 256])
    plt.plot(histr, color=col)

plt.legend(['Blue', 'Green', 'Red'])
plt.xlim([0, 256])
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.title('(a) normal133.jpg', y=-0.2, fontsize=12)
plt.xlabel('Pixel Value', fontsize=11)
plt.ylabel('Number of Pixels', fontsize=11)
plt.savefig('Figures\\normal133RGB直方图.pdf', bbox_inches='tight')

```

```
# In[8]:


color = ('b', 'g', 'r')

for i, col in enumerate(color):
    histr = cv2.calcHist([potholesImg], [i], None, [256], [0, 256])
    plt.plot(histr, color=col)

plt.legend(['Blue', 'Green', 'Red'])
plt.xlim([0, 256])
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.title('(b) potholes1.jpg', y=-0.2, fontsize=12)
plt.xlabel('Pixel Value', fontsize=11)
plt.ylabel('Number of Pixels', fontsize=11)
plt.savefig('Figures\\potholes1RGB直方图.pdf', bbox_inches='tight')

# In[9]:


plt.style.use('ggplot')
plt.hist(normalImg.ravel(), 256, [0, 256], color='grey')
plt.title('(a) normal133.jpg', y=-0.2, fontsize=12)
plt.xlabel('Pixel Value', fontsize=11)
plt.ylabel('Number of Pixels', fontsize=11)
plt.savefig('Figures\\normal133灰度直方图.pdf', bbox_inches='tight')

# In[10]:


plt.style.use('ggplot')
plt.hist(potholesImg.ravel(), 256, [0, 256], color='grey')
plt.title('(b) potholes1.jpg', y=-0.2, fontsize=12)
plt.xlabel('Pixel Value', fontsize=11)
plt.ylabel('Number of Pixels', fontsize=11)
plt.savefig('Figures\\potholes1灰度直方图.pdf', bbox_inches='tight')

# In[11]:


# 边缘检测
gray = cv2.cvtColor(normalImg, cv2.COLOR_BGR2GRAY)
edges = cv2.Canny(gray, 100, 200)
```

```
105 plt.imshow(edges, cmap='gray')
106 plt.title('(a) normal133.jpg', y=-0.2, fontsize=12)
107 plt.savefig('Figures\\normal133边缘检测.pdf', bbox_inches='tight')
108
109 # In[12]:
110
111
112 # 边缘检测
113 gray = cv2.cvtColor(potholesImg, cv2.COLOR_BGR2GRAY)
114 edges = cv2.Canny(gray, 100, 200)
115 plt.imshow(edges, cmap='gray')
116 plt.title('(b) potholes1.jpg', y=-0.2, fontsize=12)
117 plt.savefig('Figures\\potholes1边缘检测.pdf', bbox_inches='tight')
118
119 # In[13]:
120
121
122 plt.imshow(normalImg)
123 plt.colorbar()
124 plt.title('(a) normal133.jpg', y=-0.3, fontsize=12)
125 plt.savefig('Figures\\normal133热力图.pdf')
126
127 # In[14]:
128
129
130 plt.imshow(potholesImg)
131 plt.colorbar()
132 plt.title('(b) potholes1.jpg', y=-0.3, fontsize=12)
133 plt.savefig('Figures\\potholes1热力图.pdf')
134
135 # In[15]:
136
137
138 # 阈值分割
139 hsv = cv2.cvtColor(normalImg, cv2.COLOR_BGR2HSV)
140 lower_blue = np.array([90, 50, 50])
141 upper_blue = np.array([130, 255, 255])
142 mask = cv2.inRange(hsv, lower_blue, upper_blue)
143 plt.imshow(mask, cmap='gray')
144 plt.title('(a) normal133.jpg', y=-0.2, fontsize=12)
145 plt.savefig('Figures\\normal133阈值分割.pdf', bbox_inches='tight')
146
147 # In[16]:
148
```

```
149
150 # 阈值分割
151 hsv = cv2.cvtColor(potholesImg, cv2.COLOR_BGR2HSV)
152 lower_blue = np.array([90, 50, 50])
153 upper_blue = np.array([130, 255, 255])
154 mask = cv2.inRange(hsv, lower_blue, upper_blue)
155 plt.imshow(mask, cmap='gray')
156 plt.title('(b) potholes1.jpg', y=-0.2, fontsize=12)
157 plt.savefig('Figures\\potholes1阈值分割.pdf', bbox_inches='tight')
158
159 # In[17]:
160
161
162 # 转换为灰度图像
163 gray = cv2.cvtColor(normalImg, cv2.COLOR_BGR2GRAY)
164 # 二值化
165 ret, binary = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)
166 # 轮廓检测
167 contours, hierarchy = cv2.findContours(binary, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
168 # 绘制轮廓
169 cv2.drawContours(normalImg, contours, -1, (0, 0, 255), 3)
170
171 plt.subplot(1, 2, 1)
172 plt.title('(a) Original', y=-0.4, fontsize=12)
173 plt.imshow(normalImg)
174 plt.subplot(1, 2, 2)
175 plt.imshow(binary, cmap='gray')
176 plt.title('(b) Contours', y=-0.4, fontsize=12)
177 plt.savefig('Figures\\normal133轮廓检测.pdf', bbox_inches='tight')
178
179 # In[18]:
180
181
182 # 转换为灰度图像
183 gray = cv2.cvtColor(potholesImg, cv2.COLOR_BGR2GRAY)
184 # 二值化
185 ret, binary = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)
186 # 轮廓检测
187 contours, hierarchy = cv2.findContours(binary, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
188 # 绘制轮廓
189 cv2.drawContours(potholesImg, contours, -1, (0, 0, 255), 3)
190
191 plt.subplot(1, 2, 1)
192 plt.title('(a) Original', y=-0.4, fontsize=12)
```

```
plt.imshow(potholesImg)
plt.subplot(1, 2, 2)
plt.imshow(binary, cmap='gray')
plt.title('(b) Contours', y=-0.4, fontsize=12)
plt.savefig('Figures\\potholes1轮廓检测.pdf', bbox_inches='tight')
```

D.3 Random Plot Images 随机展现正常与坑洼道路图像

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import pathlib
import matplotlib.pyplot as plt
from keras.preprocessing.image import ImageDataGenerator


# In[2]:


dataDirectory = pathlib.Path('DATA\\')
classNames = [item.name for item in dataDirectory.glob('*')][:2]
classNames


# In[3]:


dataAdd = 'DATA'
normalAdd = 'DATA\\normal'
potholesAdd = 'DATA\\potholes'

# 定义一个数据生成器，用于处理图像数据，并对数据进行归一化处理。同时，将数据集的20%作为验证数据，而其余80%用于训练
dataImageDataGenerator = ImageDataGenerator(rescale = 1/255., validation_split = 0.2)
dataTrain = dataImageDataGenerator.flow_from_directory(dataAdd, target_size = (224, 224), batch_size = 32, subset = 'training', class_mode = 'binary')
dataVal = dataImageDataGenerator.flow_from_directory(dataAdd, target_size = (224, 224), batch_size = 32, subset = 'validation', class_mode = 'binary')


# In[4]:


```

```
35
36 def random_plot_images():
37     images, labels = dataTrain.next()
38     labels = labels.astype('int32')
39     i = 1
40
41     plt.figure(figsize = (10, 10))
42
43     for image, label in zip(images, labels):
44         plt.subplot(4, 5, i)
45         plt.imshow(image)
46         if label == 0:
47             plt.title(classNames[label],fontname='Times New Roman', fontsize=12, color='
                   blue')
48         else:
49             plt.title(classNames[label],fontname='Times New Roman', fontsize=12, color='
                   red')
50         plt.axis('off')
51         i += 1
52         if i == 21:
53             break
54
55     plt.tight_layout()
56     plt.savefig('Figures\\图像数据观测.pdf')
57
58
59 random_plot_images()
```

D.4 PCA-SVM PCA 降维、网格调优的支持向量机

```
1  #!/usr/bin/env python
2  # coding: utf-8
3
4  # In[1]:
5
6
7  import os
8  import cv2
9  import numpy as np
10 import collections
11 from sklearn import svm
12 import matplotlib.pyplot as plt
13 from sklearn.decomposition import PCA
14 from sklearn.metrics import accuracy_score
15 from keras.preprocessing.image import ImageDataGenerator
```

```
16
17
18  # In[2]:
19
20
21  # 加载图像数据
22  def load_img():
23      inputImg, inputLabel = [], []
24      resize = (224, 224)
25      for dirname, _, filenames in os.walk('DATA\\'):
26          for filename in filenames:
27              photo_path = os.path.join(dirname, filename)
28              photo_class = dirname.split('\\')[-1]
29              try:
30                  read_im = cv2.imread(photo_path)
31                  inputImg.append(cv2.resize(read_im, resize))
32                  # potholes == 0
33                  if photo_class == 'potholes':
34                      inputLabel.append(0)
35                  # normal == 1
36                  elif photo_class == 'normal':
37                      inputLabel.append(1)
38              except:
39                  print(photo_path)
40      return inputImg, inputLabel
41
42
43  inputImg, inputLabel = load_img()
44
45  # In[3]:
46
47
48  # 计算inputLabel中各类别的数量
49  counter = collections.Counter(inputLabel)
50  counter
51
52
53  # In[4]:
54
55
56  # 随机划分训练集和测试集，比例为test_prop，x为图像数据，y为标签
57  def train_test_split(test_prop, inputImg, inputLabel):
58      test_size = int(np.floor(test_prop * len(inputLabel)))
59      # 随机数
```

```
60      np.random.seed(202310)
61      test_index = np.random.choice(len(inputLabel), size=test_size, replace=False)
62      # 划分
63      train_x, test_x, train_y, test_y = np.delete(inputImg, test_index, axis=0), np.take
            (inputImg, test_index, axis=0), np.delete(inputLabel, test_index, axis=0), np.
            take(inputLabel, test_index, axis=0)
64      # 返回图像和标签的训练集和测试集
65      return train_x, test_x, train_y, test_y, test_index
66
67
68  train_x, test_x, train_y, test_y, test_index = train_test_split(0.2, inputImg,
        inputLabel)
69
70
71  # In[5]:
72
73
74  # opencv滤波增加训练集
75  def opencv_blur(inputImg, inputLabel):
76      inputLabelNew = inputLabel.copy()
77      inputImgNew = inputImg.copy()
78      for i in range(len(inputImg)):
79          im = inputImg[i]
80          im = im.astype('uint8')
81          imLbl = [inputLabel[i]]
82
83          # 高斯滤波
84          imgGaussian = cv2.GaussianBlur(im, (5, 5), 0)
85          # 双边滤波
86          imgBilateral = cv2.bilateralFilter(im, 9, 75, 75)
87
88          # 添加到训练集中
89          inputImgNew = np.append(inputImgNew, [imgGaussian, imgBilateral], axis=0)
90          inputLabelNew = np.append(inputLabelNew, imLbl * 2, axis=0)
91      return inputImgNew, inputLabelNew
92
93
94  inputImgNew, inputLabelNew = opencv_blur(train_x, train_y)
95
96
97  # In[6]:
98
99
100 # 图像增强，在原有的训练集上进行图像增强
```

```
101 def append_img(inputImg, inputLabel, imgIterator):
102     inputLabelNew = inputLabel.copy()
103     inputImgNew = inputImg.copy()
104     for i in range(len(imgIterator)):
105         im = imgIterator[i]
106         im = im.astype('uint8')
107         imLbl = [inputLabel[i]]
108         inputImgNew = np.append(inputImgNew, im, axis=0)
109         inputLabelNew = np.append(inputLabelNew, imLbl, axis=0)
110     return inputImgNew, inputLabelNew
111
112
113 # In[7]:
114
115
116 # 旋转 + 30 deg
117 rotate_data_generartor = ImageDataGenerator(rotation_range=30)
118 imgIterator = rotate_data_generartor.flow(train_x, batch_size=1, shuffle=False)
119 inputImgNew, inputLabelNew = append_img(inputImgNew, inputLabelNew, imgIterator)
120
121 # In[8]:
122
123
124 # 计算inputLabelNew中各类别的数量
125 counter = collections.Counter(inputLabelNew)
126 counter
127
128
129 # In[9]:
130
131
132 def plot_img(inputImgNew, inputLabelNew):
133     plt.figure(figsize=(12, 5))
134     i = 1
135     for image, label in zip(inputImgNew, inputLabelNew):
136         if i <= 5:
137             if label == 1:
138                 plt.subplot(2, 5, i)
139                 plt.imshow(image)
140                 plt.title('normal', fontname='Times New Roman', fontsize=12, color='blue'
                        )
141                 plt.axis('off')
142                 i += 1
143         elif 5 < i < 11:
```

```
            if label == 0:
                plt.subplot(2, 5, i)
                plt.imshow(image)
                plt.title('potholes', fontname='Times New Roman', fontsize=12, color='red
                    ')
                plt.axis('off')
                i += 1
        else:
            break
    plt.savefig('Figures\\PCA-SVM训练样本.pdf', bbox_inches='tight')


plot_img(inputImgNew, inputLabelNew)

# In[10]:


nx, ny, nz = train_x.shape[1], train_x.shape[2], train_x.shape[3]
train_x_nn, test_x_nn = inputImgNew, test_x
train_x = inputImgNew.reshape((inputImgNew.shape[0], nx * ny * nz)) / 255
test_x = test_x.reshape((test_x.shape[0], nx * ny * nz)) / 255
train_y = inputLabelNew.reshape((inputLabelNew.shape[0], 1))
test_y = test_y.reshape((test_y.shape[0], 1))

# In[11]:


im_pca = PCA()
im_pca.fit(train_x)
variance_explained_list = im_pca.explained_variance_ratio_.cumsum()

# In[12]:


test_x_pca = im_pca.transform(test_x)
train_x_pca = im_pca.transform(train_x)


# In[13]:


# SVM网格调优
def svm_grid_search(C, kernel, train_x, train_y):
    accuracy_score_list = []
```

```
187
188     for c in C:
189         svmClassifier = svm.SVC(C=c, kernel=kernel)
190         svmClassifier.fit(train_x, train_y.ravel())
191         pred_y = svmClassifier.predict(train_x)
192         accuracy = accuracy_score(train_y, pred_y)
193         accuracy_score_list.append(accuracy)
194         print('Regularization parameters: {:.2f}, Accuracy: {:.4f}'.format(c, accuracy))
195
196     max_accurarcy_id = accuracy_score_list.index(max(accuracy_score_list))
197     return C[max_accurarcy_id]
198
199
200 C, kernel = [0.1 * i for i in range(1, 30)], 'rbf'
201 opt_C = svm_grid_search(C, kernel, train_x_pca, train_y)
202
203 # In[14]:
204
205
206 svmClassifier = svm.SVC(C=opt_C, kernel='rbf')
207 svmClassifier.fit(train_x_pca, train_y.ravel())
208 pred_y = svmClassifier.predict(test_x_pca)
209 accuracy = accuracy_score(test_y, pred_y)
210 print('Test Accuracy: {:2.2f}%'.format(accuracy * 100))
211
212 # In[15]:
213
214
215 # 分类报告
216 from sklearn.metrics import classification_report
217
218 print(classification_report(test_y, pred_y))
219
220 # In[16]:
221
222
223 # 保存模型
224 import pickle
225
226 with open('Models\\PCA.pkl', 'wb') as f:
227     pickle.dump(im_pca, f)
228 with open('Models\\PCA-SVM.pkl', 'wb') as f:
229     pickle.dump(svmClassifier, f)
230 f.close()
```

D.5 CNN 卷积神经网络

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import os
import cv2
import h5py
import warnings
import numpy as np
import collections
import random as rn
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.utils import to_categorical
from keras.layers import Conv2D, MaxPooling2D
from sklearn.preprocessing import LabelEncoder
from keras.layers import Dense, Flatten, Dropout
from sklearn.model_selection import train_test_split

warnings.filterwarnings('always')
warnings.filterwarnings('ignore')

# In[2]:


imagePath = []
for dirname, _, filenames in os.walk('DATAX\\'):
    # 统计DATAX文件夹下子文件夹下的图片数量
    print(dirname, len(filenames))
    for filename in filenames:
        path = os.path.join(dirname, filename)
        imagePath.append(path)

len(imagePath)

# In[3]:


IMG_SIZE = 128
X = []
y = []
```

```
for image in imagePath:
    try:
        img = cv2.imread(image, cv2.IMREAD_COLOR)
        img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
        X.append(np.array(img))
        if image.startswith('DATAX\\normal'):
            y.append('normal')
        else:
            y.append('potholes')
    except:
        pass

# In[4]:


fig, ax = plt.subplots(2, 5)
plt.subplots_adjust(bottom=0.3, top=0.7, hspace=0)
fig.set_size_inches(15, 12.5)

plt.rcParams['font.sans-serif'] = ['Times New Roman']
plt.rcParams['axes.unicode_minus'] = False

for i in range(2):
    for j in range(5):
        l = rn.randint(0, len(y))
        ax[i, j].imshow(X[l][:, :, ::-1])
        if y[l] == 'normal':
            ax[i, j].set_title(y[l], color='blue')
        else:
            ax[i, j].set_title(y[l], color='red')
        ax[i, j].set_aspect('equal')

plt.savefig('Figures\\CNN训练样本.pdf', bbox_inches='tight')

# In[5]:


# 统计y中各类别的数量
collections.Counter(y)

# In[6]:


le = LabelEncoder()
```

```
Y = le.fit_transform(y)
Y = to_categorical(Y, 2)
X = np.array(X)

# In[7]:


y

# In[8]:


# Y中第一列：1表示normal，0表示potholes
Y

# In[9]:


# 划分训练集和测试集，测试集占比20%
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.20, random_state
    =5)

# In[10]:


model = Sequential()
model.add(Conv2D(32, (5, 5), activation='relu', input_shape=(128, 128, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

model.add(Flatten())
model.add(Dropout(0.4))
model.add(Dense(128, activation='relu'))
model.add(Dense(2, activation='softmax'))

# In[11]:


```

```
131
132 model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
133 model.summary()
134
135 # In[12]:
136
137
138 history = model.fit(x_train, y_train, epochs=50, batch_size=12, verbose=2,
        validation_data=(x_test, y_test))
139
140 # In[13]:
141
142
143 loss, accuracy = model.evaluate(x_test, y_test)
144 print('Test Accuracy: {:2.2f}%'.format(accuracy * 100))
145
146 # In[14]:
147
148
149 plt.figure(figsize=(12, 5))
150 plt.subplot(1, 2, 1)
151 plt.title('(a) Accuracy and Loss', y=-0.2)
152 plt.plot(history.history['accuracy'])
153 plt.plot(history.history['loss'])
154 plt.xlabel('Epochs')
155 plt.ylabel('Accuracy')
156 plt.legend(['Accuracy', 'Loss'], loc='upper right')
157 plt.subplot(1, 2, 2)
158 plt.title('(b) Validation Accuracy and Loss', y=-0.2)
159 plt.plot(history.history['val_accuracy'])
160 plt.plot(history.history['val_loss'])
161 plt.xlabel('Epochs')
162 plt.ylabel('Validation Accuracy and Loss')
163 plt.legend(['Val Accuracy', 'Val Loss'], loc='upper left')
164 plt.savefig('Figures\\CNN训练结果.pdf', bbox_inches='tight')
165
166 # In[15]:
167
168
169 prediction = model.predict(x_test)
170 y_pred = np.argmax(prediction, axis=1)
171
172 # In[16]:
173
```

```

y_testA = y_test.astype(int)
y_testB = []
for i in y_testA:
    a = 1
    if i[0] == 1 and i[1] == 0:
        a = 0
    y_testB.append(a)

# In[17]:


# 分类报告
from sklearn.metrics import classification_report

print(classification_report(y_testB, y_pred))

# In[18]:


# 保存数据集为h5文件，包括X_train, X_test, y_train, y_test
h5f = h5py.File('Models\\CNNData.h5', 'w')
h5f.create_dataset('X_train', data=x_train)
h5f.create_dataset('X_test', data=x_test)
h5f.create_dataset('y_train', data=y_train)
h5f.create_dataset('y_test', data=y_test)
h5f.close()

# In[19]:


# 保存模型
model.save('Models\\CNN.h5')
```

D.6 CNN-SVM 卷积神经网络与支持向量机

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import h5py
from sklearn.svm import SVC
import matplotlib.pyplot as plt
```

```
from keras.models import load_model
from yellowbrick.classifier import ROCAUC
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from yellowbrick.classifier import ConfusionMatrix
from sklearn.model_selection import cross_val_score
from yellowbrick.classifier import ClassificationReport
from yellowbrick.classifier import PrecisionRecallCurve

# In[2]:


plt.rcParams['font.sans-serif'] = ['Times New Roman']

# In[3]:


# 加载CNN模型
CNNModel = load_model('Models\\CNN.h5')

# 加载CNN数据
with h5py.File('Models\\CNNData.h5', 'r') as file:
    X_train = file['X_train'][:]
    y_train = file['y_train'][:]
    X_test = file['X_test'][:]
    y_test = file['y_test'][:]

# 使用CNN提取特征
X_train_features = CNNModel.predict(X_train)
X_test_features = CNNModel.predict(X_test)

# In[4]:


# 使用SVM分类器
svm = SVC(kernel='rbf', random_state=0, gamma=2, C=1)
svm.fit(X_train_features, y_train[:, 0])

# In[5]:


y_pred = svm.predict(X_test_features)
print('Accuracy: %.2f%%' % (accuracy_score(y_test[:, 0], y_pred) * 100))
print(classification_report(y_test[:, 0], y_pred))
```

```
54
55 # In[6]:
56
57
58 # 分类报告可视化
59 visualizer = ClassificationReport(svm, classes=['potholes', 'normal'])
60 visualizer.fit(X_train_features, y_train[:, 0])
61 visualizer.score(X_test_features, y_test[:, 0])
62 plt.title('CNN-SVC Classification Report')
63 plt.savefig('Figures\\[CNN-SVC]分类报告.pdf')
64
65 # In[7]:
66
67
68 # yellowbrick绘制学混淆矩阵热力图
69 cm = ConfusionMatrix(svm, classes=['potholes', 'normal'])
70 cm.fit(X_train_features, y_train[:, 0])
71 cm.score(X_test_features, y_test[:, 0])
72 plt.title('CNN-SVC Confusion Matrix')
73 plt.savefig('Figures\\[CNN-SVC]混淆矩阵热力图.pdf')
74
75 # In[8]:
76
77
78 # yellowbrick绘制ROC曲线
79 visualizer = ROCAUC(svm, classes=['potholes', 'normal'], binary=True)
80 visualizer.fit(X_train_features, y_train[:, 0])
81 visualizer.score(X_test_features, y_test[:, 0])
82 plt.legend()
83 plt.xlabel('False Positive Rate')
84 plt.ylabel('False Positive Rate')
85 plt.title('ROC Curves for CNN-SVC')
86 plt.savefig('Figures\\[CNN-SVC]ROC曲线.pdf')
87
88 # In[9]:
89
90
91 # yellowbrick绘制精度-召回率曲线
92 visualizer = PrecisionRecallCurve(svm, classes=['potholes', 'normal'])
93 visualizer.fit(X_train_features, y_train[:, 0])
94 visualizer.score(X_test_features, y_test[:, 0])
95 plt.legend()
96 plt.xlabel('Recall')
97 plt.ylabel('Precision')
```

```
plt.title('Precision Recall Curve for CNN-SVC')
plt.savefig('Figures\\[CNN-SVC]精确率召回率曲线.pdf')

# In[10]:


# 使用五折交叉验证
scores = cross_val_score(svm, X_test_features, y_test[:, 0], cv=5)
# 输出交叉验证分数
print(scores)
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))

# In[11]:


import pickle

with open('Models\\CNN-SVM.pkl', 'wb') as f:
    pickle.dump(svm, f)
f.close()
```