

队伍编号	MCB2301959
赛道	A

## 基于卷积神经网络图像特征提取的 SVM 坑洼道路检测与识别 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
1.1 问题背景	1
1.2 问题要求	1
二、问题的分析	1
2.1 问题的整体分析	1
2.2 各问题逐一分析	2
三、符号说明	2
四、模型的假设	2
五、模型的建立与求解	2
5.1 图像文件预处理	2
5.1.1 人为再分类	3
5.1.2 文件夹分类	3
5.2 正常与坑洼道路的比较分析、图像特征提取	4
5.2.1 原图、热力图对比	4
5.2.2 RGB、灰度直方图	5
5.2.3 边缘、轮廓检测	6
5.2.4 阈值分割	6
5.3 图像数据预处理	7
5.3.1 opencv 读取图像文件	7
5.3.2 图像数据两类平衡化	7
5.3.3 划分训练集与测试集	8
5.3.4 训练集数据增强	8
5.4 分类模型的建立	10
5.4.1 基于主成分分析降维、网格调优的支持向量机	10
5.4.2 深度学习：CNN 卷积神经网络	10
5.4.3 CNN-SVM 的分类模型	10
5.5 模型的评估	10
5.5.1 三种模型的效果逐一分析	10
5.5.2 模型对比	10
5.6 未知数据集的预测	10
六、模型的评价与推广	10
6.1 模型的评价	10
6.2 模型的推广	10
参考文献	11
附录	12

# 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<sup>[1]</sup>。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，并展分析结果的合理性。

# 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**，本题需要分析道路图像并提取其特征，并建立一个识别率高、速度快、分类准确的模型，用以准确判断道路是正常或是坑洼。因此本题需要完成以下两方面任务：**其一**，分析、研究不同道路图像的特征并归类总结。**其二**，根据上述分析，建立合适的模型，通过对道路图像特征识别，判断道路是否正常。确保模型的准确性、快速性及普适性。

**从图像数据特征看**，本题的道路图像具有数量多，环境等情况复杂，以及坑洼形态多样等特征，极大地增加的模型学习的难度。

**从模型的选择看**，本题图像数量多，特征复杂，且需要对道路图像做出准确分类。同时结合图像数据特征的分析，我们考虑结合图像增强技术建立深度学习的特征提取与分类的机器学习模型。

**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。

## 2.2 各问题逐一分析

- **问题一：**其核心目的在于结合给定的图像数据集，对其进行特征的提取，从而建立一个用于识别其为正常道路或是坑洼道路的模型。此外，该模型需要具有识别率高、速度快、分类准确的特点。
- **问题二：**其核心目的在于对问题一的模型进行训练，并对其进行多维度的评估分析，以保证模型的准确性、快速性及普适性。
- **问题三：**其核心目的在于利用上述已建立的模型，对未知数据集图像进行分类，并保存结果。

## 三、符号说明

符号	符号说明
$\mu$	样本平均值
$\sigma$	样本方差
$x_{\text{standard}}$	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
$\rho$	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
$\Omega$	叶节点正则项惩罚系数
$P$	某事件发生的概率
$\omega$	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

### 5.1 图像文件预处理

该题附件 1 为“data.zip”，共包含 301 张图片，且均为“jpg”格式，且在文件名中进行了标注。其中，文件名中包含“normal”字符的表示正常道路，共 266 张，包含“potholes”字符的为坑洼道路，共 35 张。考虑到数据的标注严谨性，我们首先对其进行人为再分类，将标注异常的进行重新标注或删除。此外，为方便后续模型的分析、建立与求解，我们将“normal”与“potholes”图像文件置于“DATA”文件下的“normal”及“potholes”文件夹中。

#### 5.1.1 人为再分类

考虑到数据的标注严谨性，我们首先对其进行人为再分类，将标注异常的进行重新标注或删除。对于所给数据集，我们筛选出了以下几张图片，其标注存在问题，需要进行重新标注或删除，如图 1 所示，图中所标注的为该图像的原始文件名。

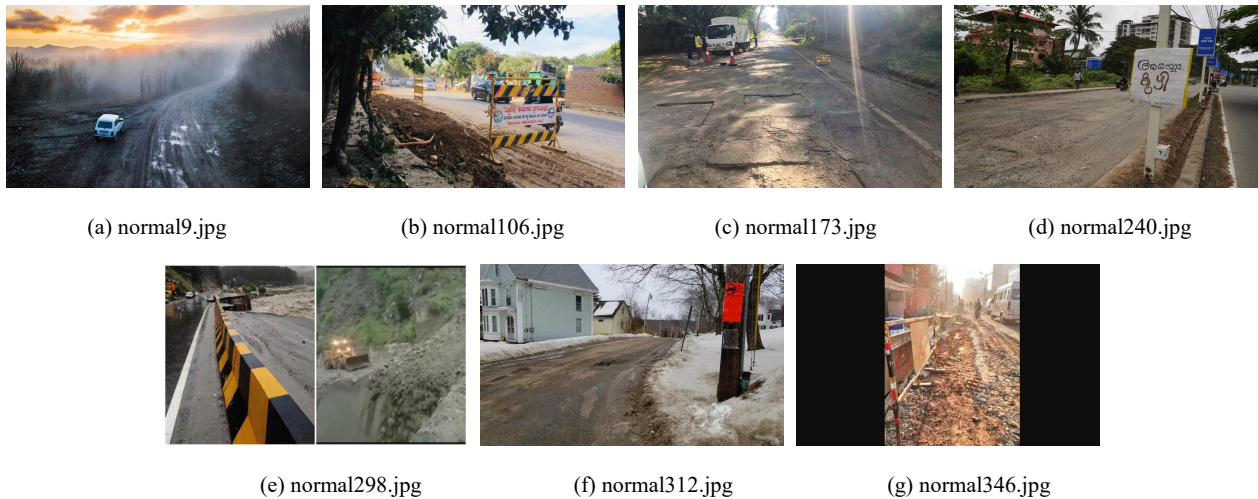


图 1 标注异常图片

对于上述“(e) normal298.jpg”文件，我们选择剔除，其余划分为“potholes”类。具体处理见表 1。

表 1 异常标注图片的处理

编号	原文件名	处理方式
a	normal9.jpg	potholes
b	normal106.jpg	potholes
c	normal173.jpg	potholes
d	normal240.jpg	potholes
e	normal298.jpg	删除
f	normal312.jpg	potholes
g	normal346.jpg	potholes

### 5.1.2 文件夹分类

为了后续模型的分析、建立与求解,我们将“normal”与“potholes”图像文件置于“DATA”文件下的“normal”及“potholes”文件夹中,作为二分类的数据集。该处理我们使用 Python 的 os 及 shutil 库进行处理,首先指定根目录为“DATA”,再在其下创建“normal”与“potholes”文件夹,之后读取“DATA”文件夹下的所有文件,若文件名中含有“normal”字符,则将其放置于“normal”文件夹中,否则放置于“potholes”文件夹中。具体处理代码见附录-D.1, **Data Preprocessing 数据预处理**。

至此,我们已将所有数据进行了人为再分类,并将数据集分别放于“normal”及"potholes"文件夹中,方便后续的处理。

### 5.2 正常与坑洼道路的比较分析、图像特征提取

为了方便后续模型的分析、建立与求解,我们还需要对图像数据进行机理分析,将标记为正常及坑洼的道路进行比较分析,并提取图像的特征。因此,我们从以下几个方面进行分析:

- **原图、热力图对比:** 分析坑洼与正常道路的表层区别。
- **RGB、灰度直方图:** 分析坑洼与正常道路的图像信息分布。
- **边缘、轮廓检测:** 分析坑洼与正常道路的边缘、轮廓特征。
- **阈值分割:** 分析坑洼与正常道路的阈值分割特征。

但由于图片较多,故我们选择“normal133.jpg”及“potholes1.jpg”进行对比分析。以下是具体分析内容,该部分具体处理代码见附录-D.2, **Comparative Analysis Of Normal And Potholes 正常与坑洼道路的比较分析**。

#### 5.2.1 原图、热力图对比

“normal133.jpg”及“potholes1.jpg”原图及热力图对比,如图 2及图 3所示。

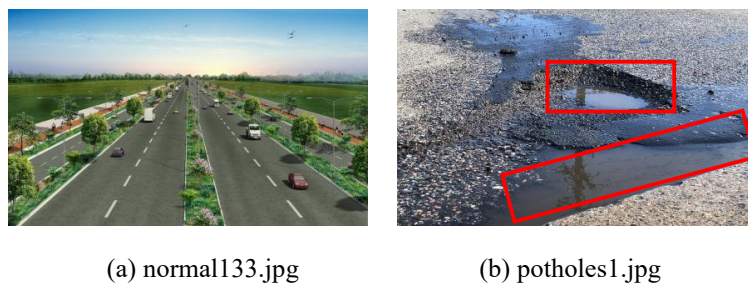


图 2 原图对比

对比正常道路和坑洼道路的热力图,我们发现若整个道路的状况相同,没有坑洼,则那一片区域的热力值近似相同,热力图呈现的颜色相仿;若道路中的某一块状况与周围不同,如出现坑洼,则出现坑洼的区域热力值发生变化,使在热力图中呈现的颜色与周围存在差异。正

常道路热力值基本一致，其热力图呈现色彩差别小，而坑洼道路中非坑洼处与坑洼处热力值相差较大，坑洼处与非坑洼处颜色相比存在较大差异。

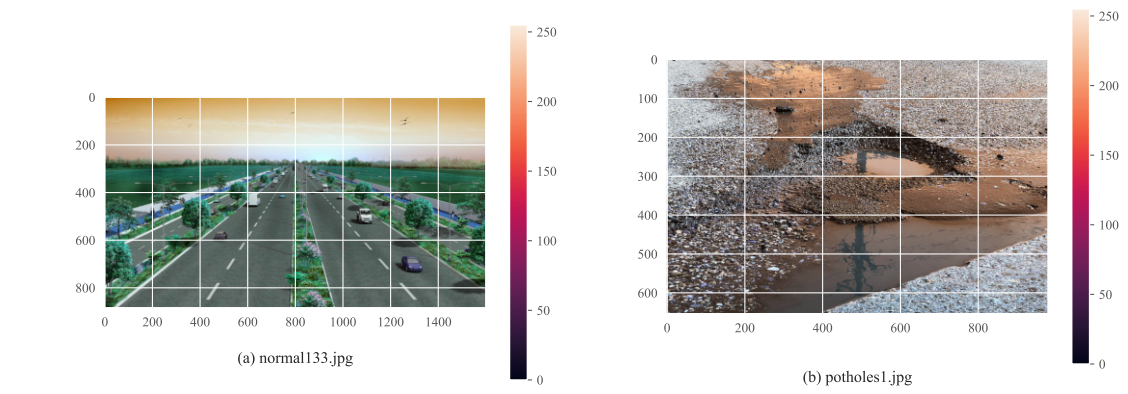


图 3 热力图对比

5.2.2 RGB、灰度直方图

“normal133.jpg”及“potholes1.jpg”RGB 及灰度直方图对比，如图 4 及图 5 所示。

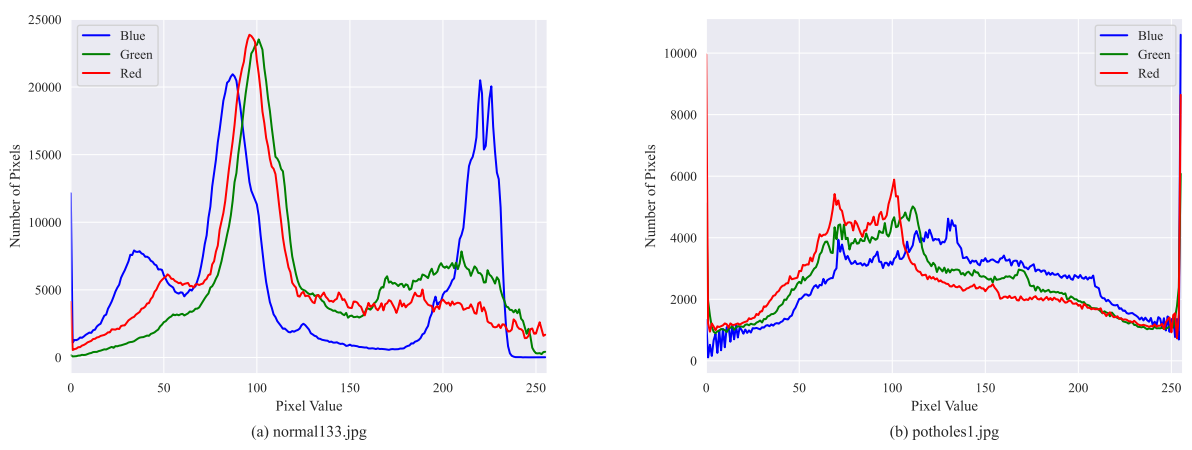


图 4 RGB 直方图对比

观察 RGB 直方图的对比，我们可以发现两者 RGB 三色整体变化具有一致性，但正常道路的 RGB 数值整体高于坑洼道路，且正常道路 RGB 峰值出现在像素值为 50~100 之间，而坑洼道路的 RGB 峰值出现在像素值为 0 与 250 处。

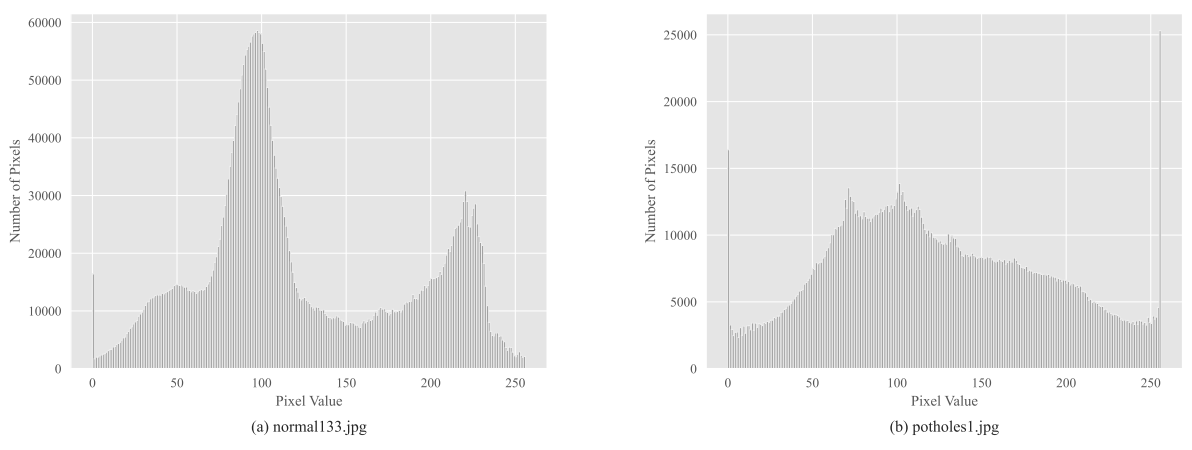


图 5 灰度直方图对比



观察灰度直方图的对比，我们可先发现灰度直方图中正常道路所呈现的峰值远高于坑洼道路，其中坑洼道路的峰值出现在图像两侧，而正常道路的峰值出现在图像的中心位置。此外正常道路数值波动明显大于坑洼道路。故我们认为灰度直方图所展现的数值大小、分布及波动情况可以较好体现道路的坑洼情况。

5.2.3 边缘、轮廓检测

“normal133.jpg”及“potholes1.jpg”边缘、轮廓检测对比，如图 6 及图 7 所示。

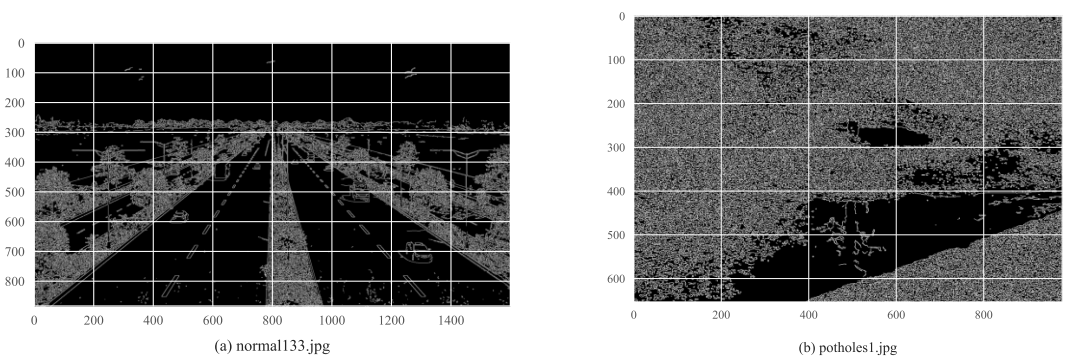


图 6 边缘检测

观察边缘检测的对比，我们可以发现边缘检测后正常道路色彩连续完整，而坑洼道路的坑洼部分在边缘检测后整体呈现深度黑色，与平整的砂石路面形成了强烈对比。故我们认为采用边缘检测可以较好判断道路状况。

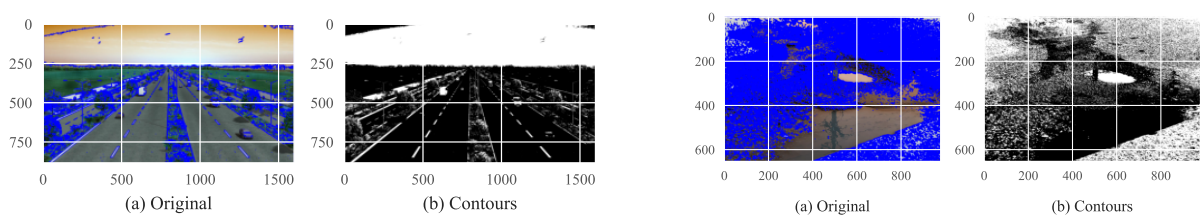


图 7 轮廓检测

对道路的图片进行轮廓检测后，我们可以发现检测后无坑洼道路的图片颜色相同，而对于坑洼道路，图片上有很明显的颜色差别。经过分析，我们认为不同的颜色代表了道路的平整与坑洼，进行轮廓检测后，可以通过道路颜色的异同判断道路是否坑洼。

5.2.4 阈值分割

“normal133.jpg”及“potholes1.jpg”阈值分割对比，如图 8 所示。



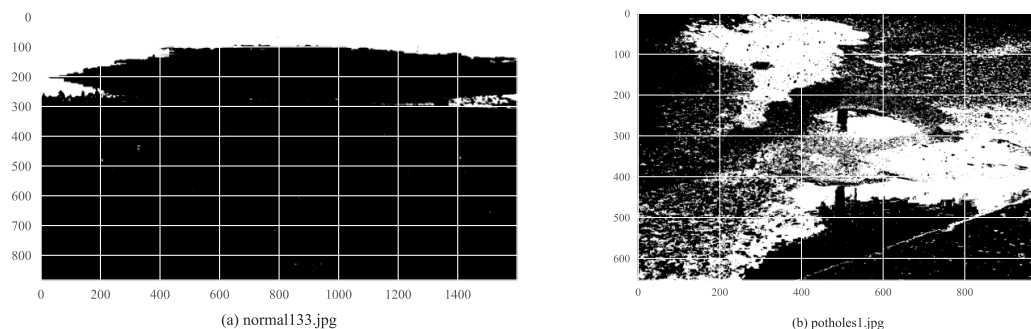


图 8 阈值分割

阈值分割将图片分为两类区域，对比正常道路与坑洼道路的阈值分割图，正常道路由于整个道路的状态相同，因此经阈值分割后，全部被划分为同一类，故颜色相同；而对于坑洼道路，道路中出现了与周围状态不同的区域，经过阈值分割后，呈现出不同的颜色，黑色为正常的道路，白色为道路中存在的坑洼，色彩对比强烈。

### 5.3 图像数据预处理

#### 5.3.1 opencv 读取图像文件

考虑到图像文件较多，为方便后续处理，我们利用 Python 的 opencv 及 os 库，将数据加载至内存中。并分析各类中包含的图像数据个数，如表 2 所示。这里罗列了原数据集与经过图像文件预处理部分处理后的情况，同时还计算初正常道路与坑洼道路数据个数的比值。

表 2 数据信息

类别	1: 正常道路	0: 坑洼道路	正常道路与坑洼道路图像数据个数比值	合计
原数据图像个数	266	35	7.60	301
处理后图像个数	259	41	6.32	300

#### 5.3.2 图像数据两类平衡化

我们需要重点关注表 2 中正常道路与坑洼道路数据个数的比值，发现正常道路图像个数为坑洼道路图像个数的 6.32 倍，这意味着正常道路图像数据的数量远大于坑洼道路图像数据的数量，即对于该二分类，其数据是不平衡的，这将会对后续模型的建立与求解造成一定的影响，易造成模型的偏向性学习。因此我们需要对现有的数据进行平衡化。即尽量使得正常道路图像数据与坑洼道路图像数据的数量相近，从而使得模型的学习均衡，减少模型的偏向性。

该处我们采用的方法是对数据集扩充，扩充的图为原数据集中尚未出现过的图像，新图像数据来源于<https://www.kaggle.com/>及<https://github.com/>。即原数据集为新数据集的子集。具体数据见表 3。

这里，我们将随机选择 10 张图像进行展示，如图 9 所示。

表 3 新旧数据集信息

	1: normal	0: potholes	正常道路与坑洼道路图像数据个数比值	总计
经预处理的数据集	259	41	6.32	300
新数据集	352	329	1.07	681

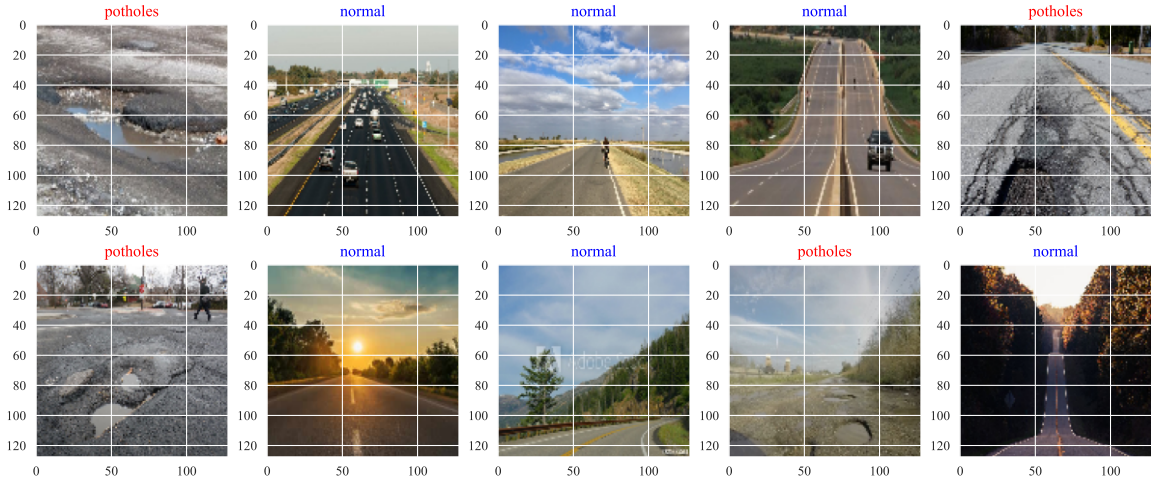


图 9 新旧数据集图像

### 5.3.3 划分训练集与测试集

对于模型的学习，我们需要对数据集进行划分，划分为训练集与测试集。训练集用于模型的训练，测试集用于对模型性能的评估分析，检验模型效果。由于该数据集样本较少，我们设置训练集占比为 80%，测试集占比为 20%，且上述划分为采用 Python 的 sklearn 库中 `model_selection.train_test_split` 方法及自定义地客观地随机划分。

### 5.3.4 训练集数据增强

观察数据集，发现其特征多数存在相似性；同时，数据集样本较少。因此，为避免模型的过拟合与欠拟合，这里我们需要对训练集进行数据增强<sup>1</sup>。从而增强模型对于未知数据集的泛化能力，提升模型的稳健性。

这里，我们选用线性滤波中的**高斯滤波 (Gauss Blur)**、非线性滤波中的**双边滤波 (Bilateral Blur)**、以及对图像进行**顺时针旋转  $\frac{\pi}{6}$**  处理。

- **高斯滤波**是一种线性平滑滤波。其基本原理是使用高斯核对图像进行卷积操作，进行加权平均的过程。每一个像素点的值，都由其邻域内的其他像素值和本身经过加权平均后得到。高斯核的标准差和大小决定了滤波器的效果，标准差越小，滤波器的效果越不明显，但是不会导致图像的细节信息丢失。其权值随着距离中心像素点的距离增加而逐渐减小，从而保留了图像的边缘信息。使用公式如下：

$$G(X, Y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{X^2 + Y^2}{2\sigma^2}\right) \quad (1)$$

<sup>1</sup>注意：这里数据增强是在划分训练集与测试集之后的，并且仅对训练集数据进行增强，并不对测试集进行划分。这是由于，若在划分训练集与测试集之前进行增强，则会造成数据泄露，影响模型的准确性，以及对未知数据的泛化能力

- **双边滤波**是一种基于高斯滤波的非线性滤波方法，目的是解决高斯滤波造成的边缘模糊。结合图像的空间邻近度和像素值相似度，同时考虑空域信息和灰度相似性，实现保边去噪。双边滤波器比高斯滤波多一个高斯核。它是基于像素颜色分布的高斯滤波函数，所以在边缘附近，当两个像素距离很近时，只有同时当颜色很接近时影响才会较大，反之，虽然距离很近，但颜色差距较大，那么平滑权重也会很小。

为实现双边滤波，我们首先定义滤波器的参数，即空间域核函数和像素值域核函数；然后计算出每个像素在空间域和像素值域上的权重；最后根据计算得到的权重，对每个像素的周围像素进行加权平均，以得到滤波后的像素值。其公式可表示为：

$$I_f(X, Y) = \frac{1}{G(X, Y)} \sum_{(i, j) \in \Omega} I(X + i, Y + j) \cdot F_s(i, j) \cdot F_f[I(X, Y), I(X + i, Y + j)] \quad (2)$$

上式中各参量含义如下：

- $I_f(X, Y)$ ：滤波后的图像像素值；
- $I(X, Y)$ ：原始图像像素值；
- $G(X, Y)$ ：归一化权重；
- $\Omega$ ：滤波邻域窗口；
- $F_s(i, j)$ ：空间域核函数；
- $F_f[I(X, Y), I(X + i, Y + j)]$ ：像素值域核函数。

这里，我们随机选择一张图片，将其经过上述处理后的图像一一展示，如图 10 所示。



图 10 原图、高斯滤波、双边滤波、顺时针旋转 30 度

经过上述滤波及旋转处理后，意味着将训练集增加 3 倍，即现有训练集为  $681 \times 0.8 \times 4 = 2179$  张图像。这里，我们将随机选择 10 张图像进行展示，如图 11 所示。



图 11 数据增强图像

至此，模型的数据前期准备已经完成。

## 5.4 分类模型的建立

### 5.4.1 基于主成分分析降维、网格调优的支持向量机

### 5.4.2 深度学习：CNN 卷积神经网络

### 5.4.3 CNN-SVM 的分类模型

## 5.5 模型的评估

为更好地评估模型，我们针对上述三种模型进行多维度评估，并针对各模型进行针对性分析。这里我们用到的有：

### 5.5.1 三种模型的效果逐一分析

### 5.5.2 模型对比

## 5.6 未知数据集的预测

# 六、模型的评价与推广

## 6.1 模型的评价

## 6.2 模型的推广



## 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](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.cbge.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	pycharts	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

---

```
1  #!/usr/bin/env python
2  # coding: utf-8
3
4  # In[1]:
5
6
7  import os
8  import shutil
9
10 # 指定目录"DATA"
11 path = "DATA"
12
13 # 在"DATA"文件夹中创建"normal"和"potholes"文件夹
14 os.mkdir(os.path.join(path, "normal"))
15 os.mkdir(os.path.join(path, "potholes"))
16
17 # 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
18 files = os.listdir(path)
19 for file in files:
20     if "normal" in file:
21         shutil.move(os.path.join(path, file), os.path.join(path, "normal"))
22     else:
23         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

---

```
1  #!/usr/bin/env python
2  # coding: utf-8
3
4  # In[1]:
5
6
7  import cv2
8  import numpy as np
9  import matplotlib.pyplot as plt
10
11 # In[2]:
12
13
14 normalImg = cv2.imread('DATA\\normal\\normal133.jpg')
15 potholesImg = cv2.imread('DATA\\potholes\\potholes1.jpg')
16
```

```

17 # In[3]:
18
19
20 plt.rcParams['font.sans-serif'] = ['Times New Roman']
21 plt.rcParams['axes.unicode_minus'] = False
22
23
24 # In[4]:
25
26
27 def cv_show(img):
28     b, g, r = cv2.split(img)
29     img = cv2.merge([r, g, b])
30     plt.imshow(img)
31
32
33 # In[5]:
34
35
36 cv_show(normalImg)
37
38 # In[6]:
39
40
41 cv_show(potholesImg)
42
43 # In[7]:
44
45
46 color = ('b', 'g', 'r')
47
48 for i, col in enumerate(color):
49     histr = cv2.calcHist([normalImg], [i], None, [256], [0, 256])
50     plt.plot(histr, color=col)
51
52 plt.legend(['Blue', 'Green', 'Red'])
53 plt.xlim([0, 256])
54 plt.xticks(fontsize=10)
55 plt.yticks(fontsize=10)
56 plt.title('(a) normal133.jpg', y=-0.2, fontsize=12)
57 plt.xlabel('Pixel Value', fontsize=11)
58 plt.ylabel('Number of Pixels', fontsize=11)
59 plt.savefig('Figures\\normal133RGB直方图.pdf', bbox_inches='tight')
60

```

```

61 # In[8]:
62
63
64 color = ('b', 'g', 'r')
65
66 for i, col in enumerate(color):
67     histr = cv2.calcHist([potholesImg], [i], None, [256], [0, 256])
68     plt.plot(histr, color=col)
69
70 plt.legend(['Blue', 'Green', 'Red'])
71 plt.xlim([0, 256])
72 plt.xticks(fontsize=10)
73 plt.yticks(fontsize=10)
74 plt.title('(b) potholes1.jpg', y=-0.2, fontsize=12)
75 plt.xlabel('Pixel Value', fontsize=11)
76 plt.ylabel('Number of Pixels', fontsize=11)
77 plt.savefig('Figures\\potholes1RGB直方图.pdf', bbox_inches='tight')
78
79 # In[9]:
80
81
82 plt.style.use('ggplot')
83 plt.hist(normalImg.ravel(), 256, [0, 256], color='grey')
84 plt.title('(a) normal133.jpg', y=-0.2, fontsize=12)
85 plt.xlabel('Pixel Value', fontsize=11)
86 plt.ylabel('Number of Pixels', fontsize=11)
87 plt.savefig('Figures\\normal133灰度直方图.pdf', bbox_inches='tight')
88
89 # In[10]:
90
91
92 plt.style.use('ggplot')
93 plt.hist(potholesImg.ravel(), 256, [0, 256], color='grey')
94 plt.title('(b) potholes1.jpg', y=-0.2, fontsize=12)
95 plt.xlabel('Pixel Value', fontsize=11)
96 plt.ylabel('Number of Pixels', fontsize=11)
97 plt.savefig('Figures\\potholes1灰度直方图.pdf', bbox_inches='tight')
98
99 # In[11]:
100
101
102 # 边缘检测
103 gray = cv2.cvtColor(normalImg, cv2.COLOR_BGR2GRAY)
104 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)

```

```

193 plt.imshow(potholesImg)
194 plt.subplot(1, 2, 2)
195 plt.imshow(binary, cmap='gray')
196 plt.title('(b) Contours', y=-0.4, fontsize=12)
197 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

---

```

1  #!/usr/bin/env python
2  # coding: utf-8
3
4  # In[1]:
5
6
7  import pathlib
8  import matplotlib.pyplot as plt
9  from keras.preprocessing.image import ImageDataGenerator
10
11
12 # In[2]:
13
14
15 dataDirectory = pathlib.Path('DATA\\')
16 classNames = [item.name for item in dataDirectory.glob('*')][1:2]
17 classNames
18
19
20 # In[3]:
21
22
23 dataAdd = 'DATA'
24 normalAdd = 'DATA\\normal'
25 potholesAdd = 'DATA\\potholes'
26
27 # 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
28 dataImageDataGenerator = ImageDataGenerator(rescale = 1/255., validation_split = 0.2)
29 dataTrain = dataImageDataGenerator.flow_from_directory(dataAdd, target_size = (224,
    224), batch_size = 32, subset = 'training', class_mode = 'binary')
30 dataVal = dataImageDataGenerator.flow_from_directory(dataAdd, target_size = (224, 224),
    batch_size = 32, subset = 'validation', class_mode = 'binary')
31
32
33 # In[4]:
34

```

```

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='
48                 blue')
49         else:
50             plt.title(classNames[label], fontname='Times New Roman', fontsize=12, color='
51                 red')
52         plt.axis('off')
53         i += 1
54         if i == 21:
55             break
56
57     plt.tight_layout()
58     plt.savefig('Figures\\图像数据观测.pdf')
59
60 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_img = cv2.imread(photo_path)
31                 inputImg.append(cv2.resize(read_img, 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                     )
142                 plt.axis('off')
143                 i += 1
144             elif 5 < i < 11:

```



```

144         if label == 0:
145             plt.subplot(2, 5, i)
146             plt.imshow(image)
147             plt.title('potholes', fontname='Times New Roman', fontsize=12, color='red
148                     ')
149             plt.axis('off')
150             i += 1
151         else:
152             break
153     plt.savefig('Figures\\PCA-SVM训练样本.pdf', bbox_inches='tight')
154
155 plot_img(inputImgNew, inputLabelNew)
156
157 # In[10]:
158
159
160 nx, ny, nz = train_x.shape[1], train_x.shape[2], train_x.shape[3]
161 train_x_nn, test_x_nn = inputImgNew, test_x
162 train_x = inputImgNew.reshape((inputImgNew.shape[0], nx * ny * nz)) / 255
163 test_x = test_x.reshape((test_x.shape[0], nx * ny * nz)) / 255
164 train_y = inputLabelNew.reshape((inputLabelNew.shape[0], 1))
165 test_y = test_y.reshape((test_y.shape[0], 1))
166
167 # In[11]:
168
169
170 im_pca = PCA()
171 im_pca.fit(train_x)
172 variance_explained_list = im_pca.explained_variance_ratio_.cumsum()
173
174 # In[12]:
175
176
177 test_x_pca = im_pca.transform(test_x)
178 train_x_pca = im_pca.transform(train_x)
179
180
181 # In[13]:
182
183
184 # SVM网格调优
185 def svm_grid_search(C, kernel, train_x, train_y):
186     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_accuracy_id = accuracy_score_list.index(max(accuracy_score_list))
197     return C[max_accuracy_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: {:.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

---

```
1  #!/usr/bin/env python
2  # coding: utf-8
3
4  # In[1]:
5
6
7  import os
8  import cv2
9  import h5py
10 import warnings
11 import numpy as np
12 import collections
13 import random as rn
14 import matplotlib.pyplot as plt
15 from keras.models import Sequential
16 from keras.utils import to_categorical
17 from keras.layers import Conv2D, MaxPooling2D
18 from sklearn.preprocessing import LabelEncoder
19 from keras.layers import Dense, Flatten, Dropout
20 from sklearn.model_selection import train_test_split
21
22 warnings.filterwarnings('always')
23 warnings.filterwarnings('ignore')
24
25 # In[2]:
26
27
28 imagePath = []
29 for dirname, _, filenames in os.walk('DATAX\\'):
30     # 统计DATAX文件夹下子文件夹下的图片数量
31     print(dirname, len(filenames))
32     for filename in filenames:
33         path = os.path.join(dirname, filename)
34         imagePath.append(path)
35
36 len(imagePath)
37
38 # In[3]:
39
40
41 IMG_SIZE = 128
42 X = []
43 y = []
```

```

44 for image in imagePath:
45     try:
46         img = cv2.imread(image, cv2.IMREAD_COLOR)
47         img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
48         X.append(np.array(img))
49         if image.startswith('DATA\\normal'):
50             y.append('normal')
51         else:
52             y.append('potholes')
53     except:
54         pass
55
56 # In[4]:
57
58
59 fig, ax = plt.subplots(2, 5)
60 plt.subplots_adjust(bottom=0.3, top=0.7, hspace=0)
61 fig.set_size_inches(15, 12.5)
62
63 plt.rcParams['font.sans-serif'] = ['Times New Roman']
64 plt.rcParams['axes.unicode_minus'] = False
65
66 for i in range(2):
67     for j in range(5):
68         l = rn.randint(0, len(y))
69         ax[i, j].imshow(X[l][:, :, :-1])
70         if y[l] == 'normal':
71             ax[i, j].set_title(y[l], color='blue')
72         else:
73             ax[i, j].set_title(y[l], color='red')
74         ax[i, j].set_aspect('equal')
75
76 plt.savefig('Figures\\CNN训练样本.pdf', bbox_inches='tight')
77
78 # In[5]:
79
80
81 # 统计y中各类别的数量
82 collections.Counter(y)
83
84 # In[6]:
85
86
87 le = LabelEncoder()

```

```

88 Y = le.fit_transform(y)
89 Y = to_categorical(Y, 2)
90 X = np.array(X)
91
92 # In[7]:
93
94
95 y
96
97 # In[8]:
98
99
100 # Y中第一列: 1表示normal, 0表示potholes
101 Y
102
103 # In[9]:
104
105
106 # 划分训练集和测试集, 测试集占比20%
107 x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.20, random_state
    =5)
108
109 # In[10]:
110
111
112 model = Sequential()
113 model.add(Conv2D(32, (5, 5), activation='relu', input_shape=(128, 128, 3)))
114 model.add(MaxPooling2D((2, 2)))
115 model.add(Conv2D(64, (3, 3), activation='relu'))
116 model.add(MaxPooling2D((2, 2)))
117 model.add(Conv2D(128, (3, 3), activation='relu'))
118 model.add(MaxPooling2D((2, 2)))
119 model.add(Conv2D(128, (3, 3), activation='relu'))
120 model.add(MaxPooling2D((2, 2)))
121 model.add(Conv2D(128, (3, 3), activation='relu'))
122 model.add(MaxPooling2D((2, 2)))
123
124 model.add(Flatten())
125 model.add(Dropout(0.4))
126 model.add(Dense(128, activation='relu'))
127 model.add(Dense(2, activation='softmax'))
128
129 # In[11]:
130

```

```

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,
139                     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: {:.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

```



```

174
175 y_testA = y_test.astype(int)
176 y_testB = []
177 for i in y_testA:
178     a = 1
179     if i[0] == 1 and i[1] == 0:
180         a = 0
181     y_testB.append(a)
182
183 # In[17]:
184
185
186 # 分类报告
187 from sklearn.metrics import classification_report
188
189 print(classification_report(y_testB, y_pred))
190
191 # In[18]:
192
193
194 # 保存数据集为h5文件, 包括X_train, X_test, y_train, y_test
195 h5f = h5py.File('Models\\CNNDData.h5', 'w')
196 h5f.create_dataset('X_train', data=x_train)
197 h5f.create_dataset('X_test', data=x_test)
198 h5f.create_dataset('y_train', data=y_train)
199 h5f.create_dataset('y_test', data=y_test)
200 h5f.close()
201
202 # In[19]:
203
204
205 # 保存模型
206 model.save('Models\\CNN.h5')

```

---

## D.6 CNN-SVM 卷积神经网络与支持向量机

---

```

1 #!/usr/bin/env python
2 # coding: utf-8
3
4 # In[1]:
5
6
7 import h5py
8 from sklearn.svm import SVC
9 import matplotlib.pyplot as plt

```

```

10 from keras.models import load_model
11 from yellowbrick.classifier import ROCAUC
12 from sklearn.metrics import accuracy_score
13 from sklearn.metrics import classification_report
14 from yellowbrick.classifier import ConfusionMatrix
15 from sklearn.model_selection import cross_val_score
16 from yellowbrick.classifier import ClassificationReport
17 from yellowbrick.classifier import PrecisionRecallCurve
18
19 # In[2]:
20
21
22 plt.rcParams['font.sans-serif'] = ['Times New Roman']
23
24 # In[3]:
25
26
27 # 加载CNN模型
28 CNNModel = load_model('Models\\CNN.h5')
29
30 # 加载CNN数据
31 with h5py.File('Models\\CNNDData.h5', 'r') as file:
32     X_train = file['X_train'][:]
33     y_train = file['y_train'][:]
34     X_test = file['X_test'][:]
35     y_test = file['y_test'][:]
36
37 # 使用CNN提取特征
38 X_train_features = CNNModel.predict(X_train)
39 X_test_features = CNNModel.predict(X_test)
40
41 # In[4]:
42
43
44 # 使用SVM分类器
45 svm = SVC(kernel='rbf', random_state=0, gamma=2, C=1)
46 svm.fit(X_train_features, y_train[:, 0])
47
48 # In[5]:
49
50
51 y_pred = svm.predict(X_test_features)
52 print('Accuracy: %.2f%%' % (accuracy_score(y_test[:, 0], y_pred) * 100))
53 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')

```

```
98 plt.title('Precision Recall Curve for CNN-SVC')
99 plt.savefig('Figures\\[CNN-SVC]精确率召回率曲线.pdf')
100
101 # In[10]:
102
103
104 # 使用五折交叉验证
105 scores = cross_val_score(svm, X_test_features, y_test[:, 0], cv=5)
106 # 输出交叉验证分数
107 print(scores)
108 print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
109
110 # In[11]:
111
112
113 import pickle
114
115 with open('Models\\CNN-SVM.pkl', 'wb') as f:
116     pickle.dump(svm, f)
117 f.close()
```

---