

队伍编号	MCB2301959
赛道	A

基于卷积神经网络 CNN 图像特征提取的 SVM 坑洼道路检测与识别

摘 要

啥也不说了，先占个坑，等我有空再写。

关键词：计算机视觉；卷积神经网络；支持向量机；特征提取；模型融合；可视化多维评估

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 初赛总结	2
2.3 各问逐一分析	3
2.4 行文思路	3
三、 模型的假设	3
四、 符号说明	4
五、 模型的建立与求解	4
5.1 初赛研究相关结论、坑洼特征分析	4
5.1.1 原图、热力图对比	5
5.1.2 RGB、灰度直方图	5
5.1.3 边缘、轮廓检测	6
5.1.4 阈值分割	7
5.2 图像文件预处理	7
5.2.1 人为再分类	7
5.2.2 文件夹分类	8
5.2.3 原数据集的保留与拓充	9
5.2.4 测试集数据预读取	10
5.3 图像数据预处理	10
5.3.1 opencv 读取图像文件	10
5.3.2 划分训练集与测试集	10
5.3.3 训练集数据增强	10
5.4 坑洼特性提取模型的训练	12
5.4.1 1	12
5.5 测试集图像坑洼特性预测	12
5.6 模型的评估	12
5.6.1 三种模型的效果逐一分析	14
5.6.2 模型对比	17
5.7 未知数据集的预测	17
六、 模型的评价与推广	18
6.1 模型的评价	18
6.2 模型的推广	19
参考文献	21

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

- 初赛：
 - **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，并展示分析结果的合理性。
- 复赛：
 - **问题一：**建立可以更精准地识别坑洼边缘的模型，以满足道路避障、道路修复等工作的需求。
 - **问题二：**依据识别的坑洼边缘，对其面积进行估算，计算各坑洼占其图像面积的百分比。

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、检测类问题。

从分析目的看，本题需要分析道路坑洼图像并提取其边缘、面积等特性，建立一个识别率高、速度快、准确的普适性模型。因此本题需要完成以下两方面任务：其一，分析、研究道路坑洼的边缘特性，检测其边缘，为后续模型的建立提供支撑；同时，用以完成道路避障、道路修复等工作。其二，根据上述的提取的边缘特性，估算图像中坑洼的像素占比。

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。因此优先考虑深度学习的方法, 利用其强大的特征提取与表示能力, 对道路图像进行特征提取。

从模型的选择看, 本题图像数量多, 特征复杂, 且需要对道路中的坑洼边缘进行检测, 同时估算其像素占比。因此, 我们考虑图像增强技术, 建立基于 Ultralytics 平台的 YOLO (You Only Look Once, YOLO) 算法, 对道路的坑洼特性进行学习。

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, 对坑洼道路的边缘进行提取, 估算其像素占比, 因此我们选择使用基于 Pycharm 内核的 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 初赛总结

针对**问题一**, 需要提取给定图像的特征, 建立一个用于识别其为正常或是坑洼道路的识别率高、速度快、分类准确的模型。首先, 对图像文件**预处理**, 包括**人为再分类**, 对标记错误的进行**修正**, 或**剔除异常图像**; 再将两类图像置于各子文件中。其次, 从**机理层**面对正常与坑洼道路进行比较分析, 包括**原图、热力图、RGB 与灰度直方图的对比、边缘与轮廓检测、阈值分割**, 从而有效地提取出**图像特征**。之后, 利用 **opencv-python** 读取图像, 并加入 **try-except 异常处理语块, 防止破损文件污染模型**; 此外, 计算出正常与坑洼道路的个数比值为 **6.32**, 即**数据集类别不平衡**, 因此采用**基于抽样的方法平衡化**; 之后, 以 **8 : 2** 的比例划分数据集为**训练集与测试集**; 考虑到现有的图像未能较全面地展现特征, 因此**对训练集采用高斯、双边滤波, 旋转处理进行图像增强**。随后, 建立支持向量机、卷积神经网络的单一模型, 但由于 **CNN 中 softmax 函数的局限性**, 因而, **利用卷积神经网络对图像信息的特征进行提取**, 依据上述分析, 设定其输入层、卷积层、激活层、池化层、全连接层参数; 在此基础上, 建立**支持向量机分类模型**, 对道路类别进行分类。

针对**问题二**, 需要对问题一的模型进行训练, 并进行多维度的评估分析, 保证模型的准确性、快速性及普适性。因此, 对问题一所建立的模型进行训练与测试, 并从**模型耗时、准确率、分类报告、五折交叉验证, 混淆矩阵、ROC/AUC 曲线**进行多维度评价。模型准确率在 **95 %** 的置信水平下达到 **(90 ± 8) %**; 其精确率、召回率、F1 分数值在各维度下均于 **88 % 及以上**; **ROC/AUC 曲线靠近图的左上角, 曲线下面积达到 0.90**。此外, 还将其与单一模型进行多维度对比, **准确率最高可提升 3 %**, 模型提升效果良好。

针对**问题三**, 需要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。经过读取模型、读取数据、数据集预处理、预测得出**模型在 4942 张图像中, 识别出坑洼道路为 3765 张, 识别为正常道路为 1177 张**。

最后, 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。

2.3 各问逐一分析

- **问题一：**核心目的在于建立更精准的识别道路坑洼边缘特性的模型，用于完成道路避障、道路修复等工作。因此考虑对图像文件进行分割，有效地提取坑洼的边缘。首先，需要对图像文件进行预处理，包括人为再分类、文件夹分类，对标记错误的图像进行修正，并剔除异常的图像、原数据集保留与拓充、测试集数据预读取。之后从机理层面对道路的坑洼处进行分析，包括原图、热力图、RGB 直方图、灰度直方图的对比、边缘及轮廓检测、以及阈值分割，从而有效地从机理层面分析特性。在此基础上建立基于 YOLOv8s 的 segment 图像分割及 YOLOv8n 的 detect 图像检测模型，对道路的坑洼特性进行学习、训练。在此之前需要对图像进行标注，标注文件的格式转换，训练集、测试集、验证集三者的划分，对训练集进行数据增强，配置训练文件。待模型训练完成后，再读取迭代过程中最优模型对测试集数据的坑洼边缘进行提取。
- **问题二：**核心目的在于依据上述识别的坑洼边缘，估算坑洼面积占比。因此考虑结合上述建立的检测与分割模型，得到其边缘像素坐标，估算其像素面积，进而计算坑洼面积占比。这里需要考虑到预测的坑洼的多样性，可能存在多个坑洼、坑洼边缘重叠等情况，因此需要对上述情况进行对应的处理，得到其坑洼面积，从而求得坑洼面积占比。

2.4 行文思路

为了便于理清思路，绘制出本文解决问题的流程图，如图 1 所示。

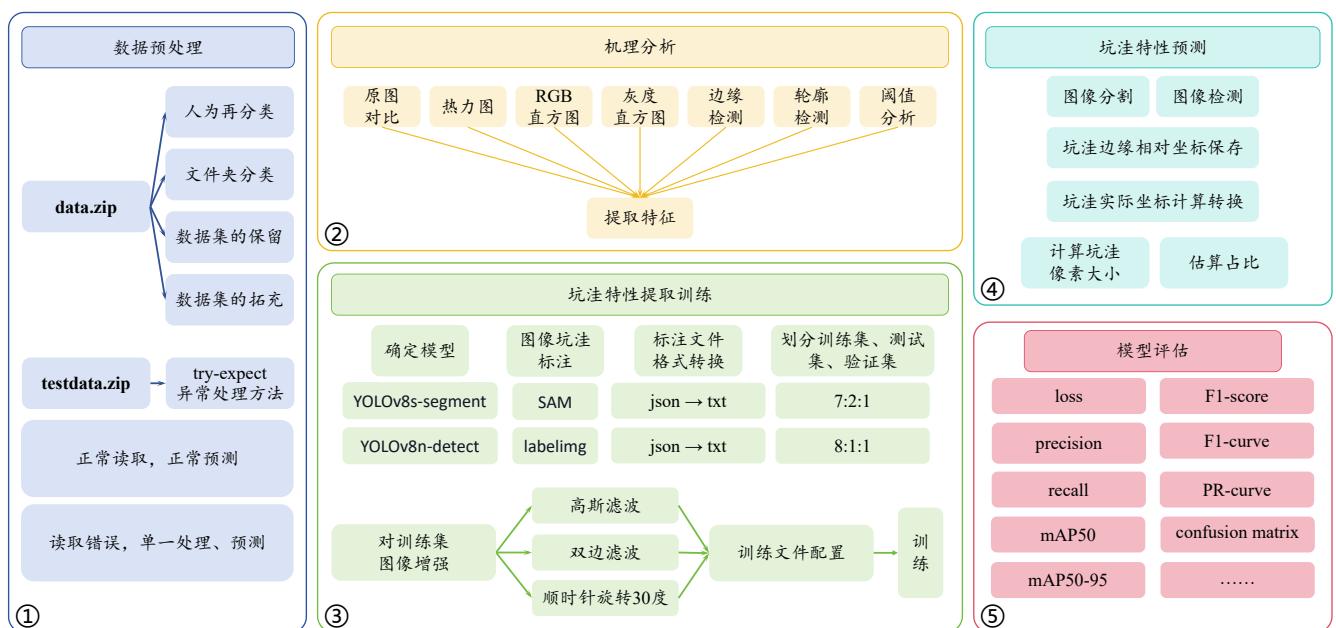


图 1 流程图

三、模型的假设

- **假设一：**所给数据集存在因某些原因而标注异常的图像数据的现象；
- **假设二：**所给数据集未能完整地表现出分类的特征。

四、符号说明

符号	符号说明
X	图像横轴像素点的像素值
Y	图像纵轴像素点的像素值
I_f	滤波后的图像像素值
I	原始图像像素值
G_s	归一化权重
Ω	滤波邻域窗口
F_s	空间域核函数
F_f	像素值域核函数
$E(x)$	数据均值
λ_n	协方差矩阵特征值
\mathbf{u}	协方差矩阵特征向量
C	SVM 惩罚系数
S	池化区域面积
$N_{\text{TruePredict}}$	预测正确样本数
N_{Predict}	预测样本总数

注：这里并未列出其余变量，这是由于它们在不同小节处有不同的含义，同时该表中也未列出专有定义的变量，这些变量在使用时会在相应位置进行详细说明。

五、模型的建立与求解

5.1 初赛研究相关结论、坑洼特征分析

在初赛中，为了方便模型的分析、建立与求解，因此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。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-C.11，Comparative Analysis of Normal and Potholes [正常与坑洼道路的比较分析]。

5.1.1 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¹。

¹本文所有图示、表格均已交叉引用，读者阅读 PDF 时可点击对应图表，进行跳转。



(a) normal133.jpg

(b) potholes1.jpg

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。

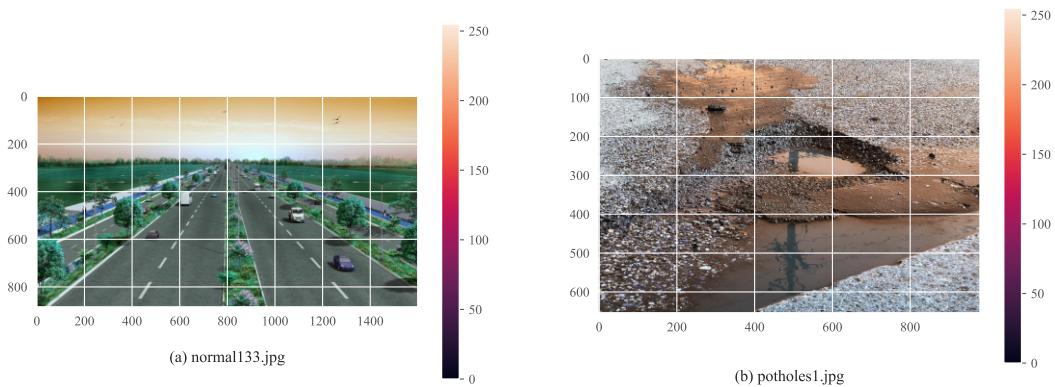


图 3 热力图对比

5.1.2 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。

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。

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。

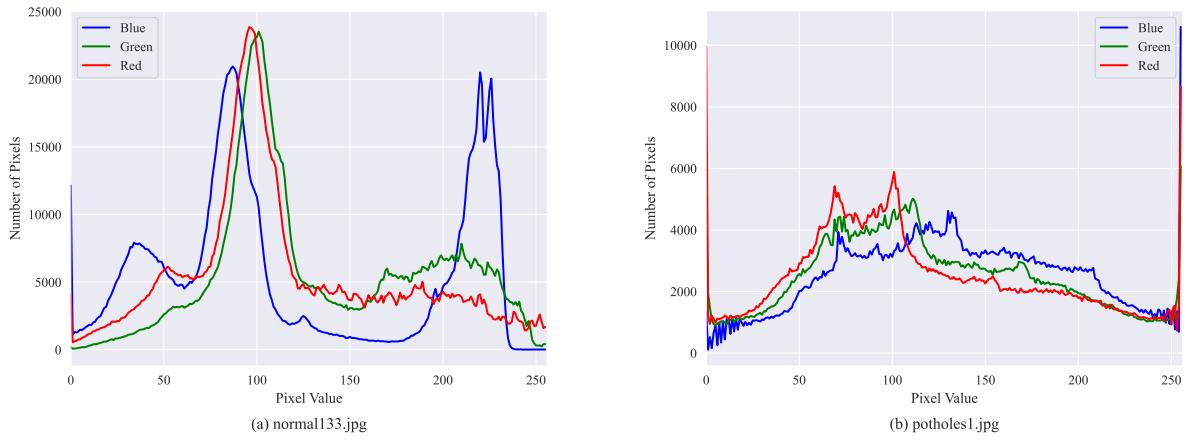


图 4 RGB 直方图对比

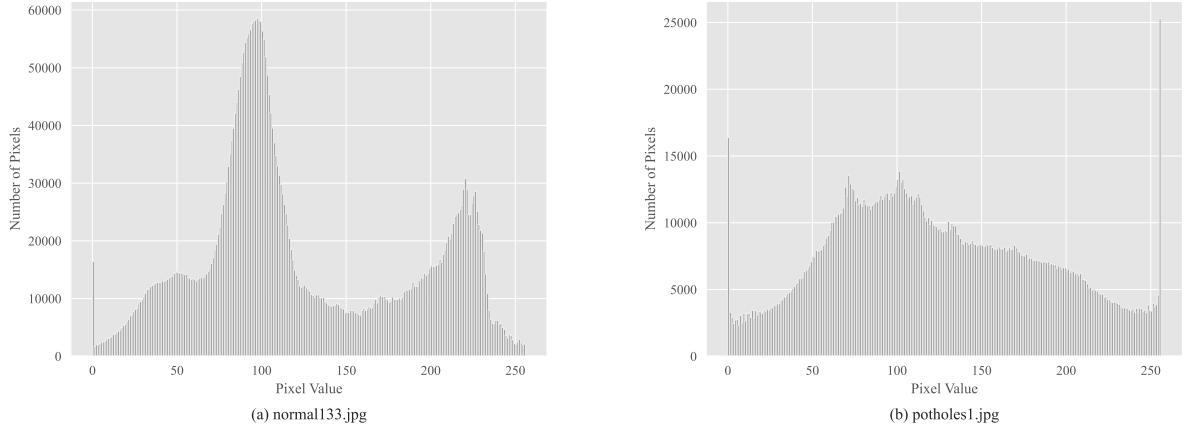


图 5 灰度直方图对比

5.1.3 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。

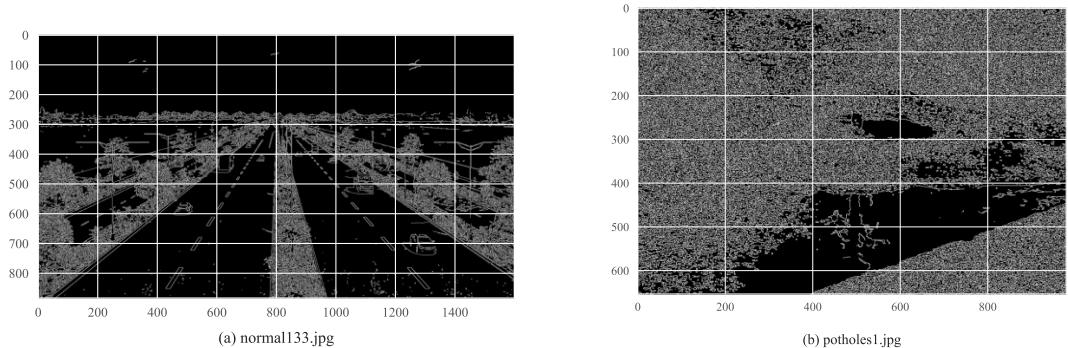


图 6 边缘检测

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。

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。

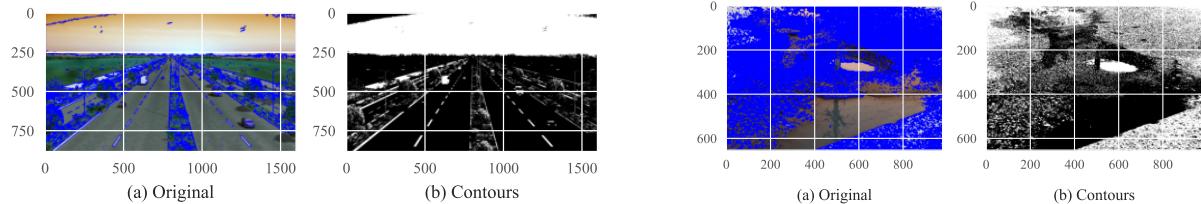


图 7 轮廓检测

5.1.4 阈值分割

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

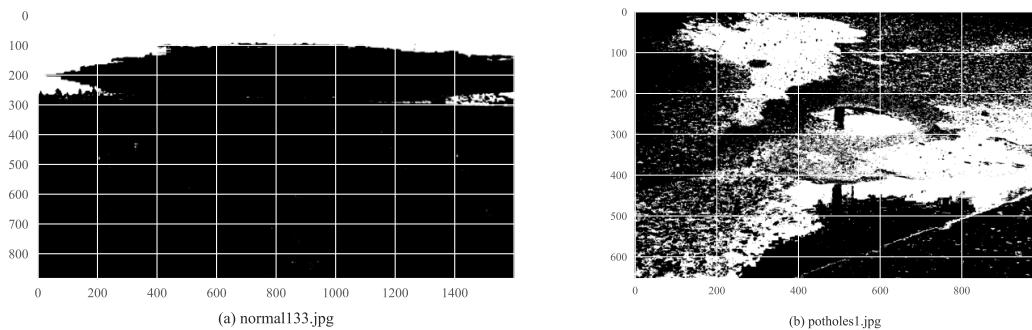


图 8 阈值分割

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。

5.2 图像文件预处理

该题原数据集为 “data.zip”，包含 301 张图片，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，首先对其进行人为再分类，将标注异常的进行重新标注或剔除。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。

考虑到图像文件较少，用于训练的特征不足，因此对原数据集进行拓充，方法为从网络上获取坑洼道路的图像。

此外，还需要预先读取测试集图像，以防破损文件影响预测过程。

5.2.1 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剔除。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剔除，如图 9 所示，图中所标注的为该图像的原始文件名。

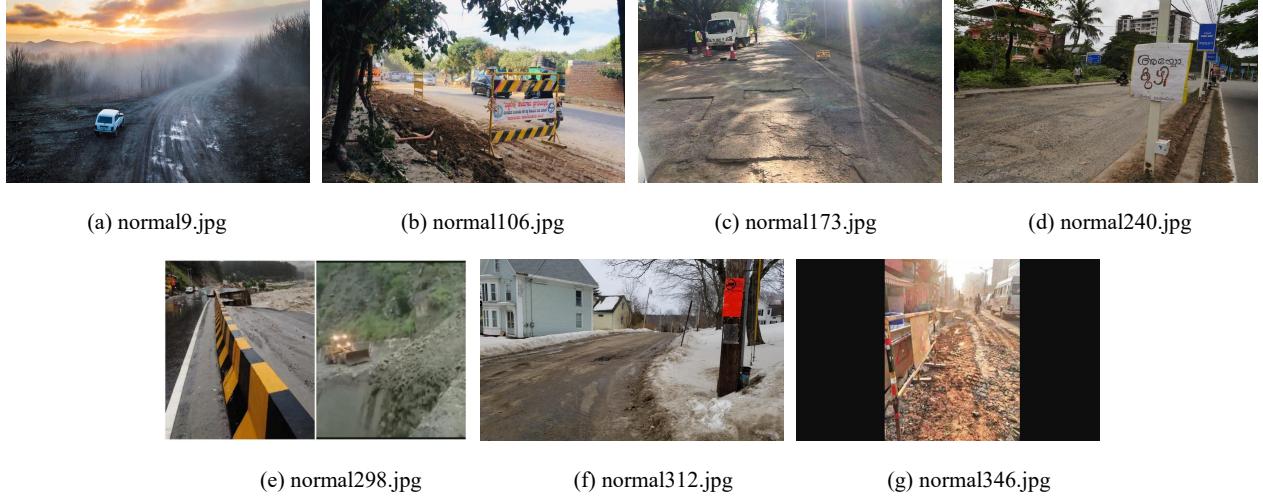


图 9 标注异常图片

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。表中“potholes”代表将该文件视为坑洼道路，而“删除”代表将该文件删除，不参与后续模型的分析、建立、训练等。

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

5.2.2 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-C.10, Data Preprocessing [数据预处理]。

这里，我们可以大致观测数据集图像，如图 10 所示。

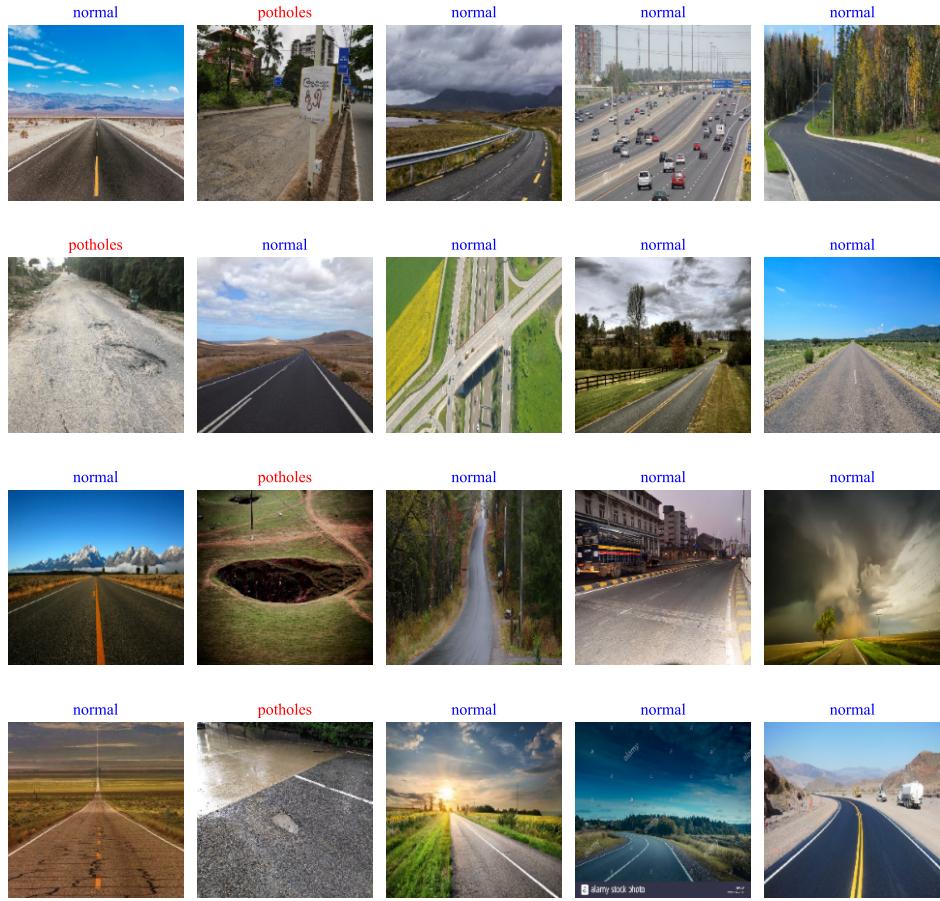


图 10 图像数据观测

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。

5.2.3 原数据集的保留与拓充

由于在复赛中，本题任务需要重点分析坑洼道路的特性，因此，这里舍弃“normal”类图像，保留“potholes”图像，共计 41 张。

考虑到保留的图像文件较少，学习特征不足，将会影响模型精度。同时，为了加强我们分析的可信度，我们从以下网站收集了坑洼图像，对“potholes”类图像数据进行拓充，来源见表 2 所示。²

表 2 拓充数据来源

数据源	网址
Kaggle	https://www.kaggle.com/
Github	https://github.com/
百度图片	https://image.baidu.com/

²注：拓充的图为原数据集中尚未出现过的图像，即原数据集为新数据集的子集。

5.2.4 测试集数据预读取

此外，还需要预先读取测试集图像，以防破损文件影响预测过程。这里我们使用 ultralytics 库，并加上 try-except 异常处理，将数据加载至内存中，方便后续进行预测。

经过读取，发现存在一个文件读取失败，为“guy7iodk.jpg”。因此，对于该张图片，我们对其进行单一读取、处理。而其余的 4941 张图正常读取，在后续可正常预测。

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 3 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，同时还计算出正常道路与坑洼道路数据个数的比值。

表 3 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 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。

5.3.3 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³。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：

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

³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：

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

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；
- $G_s(X, Y)$: 归一化权重；
- Ω : 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 11 所示。经过上述处理后，在一定程度上可以增加训练集数据的多样性，从而模型可以学习到更多关于正常或坑洼道路的特征，因而有利于提升模型的泛化能力。

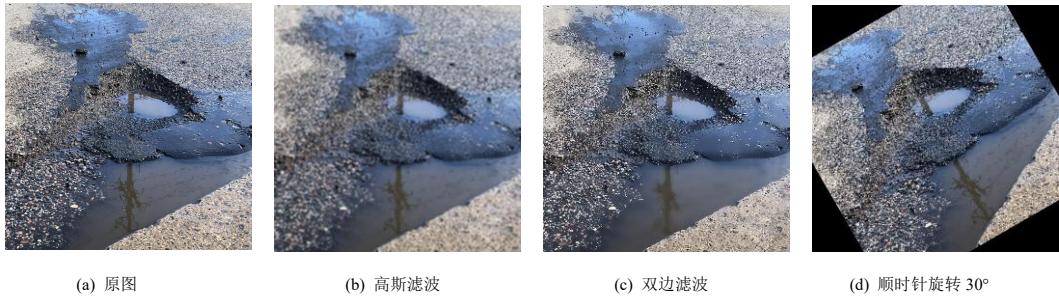


图 11 原图、高斯滤波、双边滤波、顺时针旋转 30 度

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 12 所示。



图 12 数据增强图像

5.4 坑洼特性提取模型的训练

5.4.1 1

5.5 测试集图像坑洼特性预测

5.6 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：

- **模型训练及预测时间和 (T)**
- **准确率 (Accuracy):** 即预测正确数占总数的比例，其计算公式如下：

$$\text{Accuracy} = \frac{N_{\text{TruePredict}}}{N_{\text{Predict}}} \quad (3)$$

其中, $N_{\text{TruePredict}}$ 为预测正确的样本数, N_{Predict} 为被预测的样本总数;

- **损失值 (Loss):** 这里我们采用 Categorical Crossentropy^[11], 即交叉熵损失函数, 其使用交叉熵 (Cross Entropy) 作为度量分类任务的差异, 从而来衡量两者之间的概率分布的相似性, 其公式如下:

$$\text{Loss} = - \sum_{i=1}^{\text{output size}} y_i \cdot \log(\hat{y}_i) \quad (4)$$

其中, y_i 为真实标签, \hat{y}_i 为预测标签;

- **分类报告 (Classification Report):** 其可以直观得到模型各项参数, 包括每一类别的精确率 (Precision), 召回率 (Recall), F1 分数值 (F1-Score)。对于这三项值, 其计算公式如下:

– 精确率

$$\text{Precision} = \frac{TP}{TP + FP} \quad (5)$$

– 召回率

$$\text{Recall} = \frac{TP}{TP + FN} \quad (6)$$

– F1 分数值

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \quad (7)$$

此外, 对于模型的精确率、召回率, 我们可以根据定义可以发现若这两项值较大, 则模型效果较好。同时根据定义, 我们可以发现模型的精确率、召回率在理想情况下是相差较小的, 我们可以根据图示结果验证, 符合预期效果。对于模型的 F1 分数值, 其为精确率与召回率的调和平均数^[12], 因此当精确率与召回率均有较好表现时, F1 分数值会有较优秀表现。我们也可对(7)式进行一定变换, 可以得到:

$$F1 = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} \quad (8)$$

根据该式, 我们可以得出上述结论。

- **混淆矩阵 (Confusion Matrix):** 矩阵每一行表示样本标签的实际类别, 在本题中表示道路类型: 为正常还是坑洼的实际标签; 每一列表示样本标签的预测类别, 在本题中表示道路类型: 为正常还是坑洼的预测标签。因此该图示的主对角线数据之和即为模型预测准确的样本数。这里此外还需要引入四项值, 分别为 TP 、 FN 、 FP 、 TN , 其中 T 为 True, F 为 False, 这两个字母表示预测值与实际值是否相同; P 为 Positive, N 为 Negative, 这两个字母表示预测出的是属于正类还是负类。而混淆矩阵可以直观地观察到预测准确与错误的情况, 以及模型对于每一类别的区分程度。
- **特征曲线及曲线下面积曲线 (Receiver Operating Characteristic/Area Under the Curve, ROC/AUC):** 首先我们需要引出模型的相关参数, 定义如下:

- 灵敏度 (Sensitivity)。灵敏度又被称为真阳性率，即 TP 率：

$$\text{Sensitivity} = TPR = \frac{TP}{TP + FN} \quad (9)$$

- 特异性 (Specificity)。特异性又被称为真阴性率，即 TN 率：

$$\text{Specificity} = TNR = \frac{TN}{TN + FP} \quad (10)$$

- 1-Specificity。称为假阳性率 (False Positive Rate, FPR)：

$$FPR = 1 - \text{Specificity} = \frac{FP}{FP + TN} \quad (11)$$

- 1-Sensitivity。称为假阴性率 (False Negative Rate, FNR)：

$$FNR = 1 - \text{Sensitivity} = \frac{FN}{FN + TP} \quad (12)$$

FPR 和 FNR 均对数据分布的变化不敏感^[13]，因此这两个指标可以用于在不平衡的数据上建立的模型效果的评价。

对于 ROC/AUC 曲线，其以每一类别的 1-Specificity 即 FPR 为横坐标，以 Sensitivity 即 TPR 为纵坐标，其可体现出模型的灵敏度与特异性之间的关系与差异。因此，该图的理想点位于左上角，即 $FPR = 0$ 且 $TPR = 1$ 。换言之，当曲线越靠近左上角，模型效果就越优。从而，我们可以得到另一项指标，即曲线下面积 (Area Under the Curve, AUC)，由上述分析可知，AUC 值越高，模型的整体效果也就越优。

- 精确率-召回率曲线 (Precision-Recall Curve): 该图像可表现出分类的预测精度与召回率之间的关系^[14]。对于预测精度与召回率的计算方式在“问题三”中已经叙述。因此图像的填充区域越大，分类效果越优。
- 交叉验证 (Cross Validation): 在该方法中，每一条数据用于训练的次数均相同，且恰好被检验一次。假设使用 $k(k = 2, 3, 4, \dots)$ 折交叉验证，将数据分为大小相等的 k 份，在每一次学习时，选择其中一份作为检验集，而余下的数据作为训练集，上述过程重复 k 次。此外，检验集是互斥的，并且能够有效地覆盖数据集^[15]。

5.6.1 三种模型的效果逐一分析

- PCA-SVM

通过查看 Jupyter 的代码执行记录：PCA 模型在训练集训练耗时 50 s + 281 ms，在测试集适应耗时 2 s + 968 ms；SVM 模型网格调优耗时 9 s + 948 ms，在训练集训练及在测试集测试共耗时 185 ms。通过在测试集的测试，经计算，该模型准确率为：86.67%。此外，我们还得出其分类报告表，如表 4 所示。观察上表，我们可以发现由于该模型尚未采用采样方法扩充数据集，虽然整体准确率较高，但对于各类别有明显的不平衡性。此

表 4 PCA-SVM 分类报告表

维度	precision	recall	f1-score	support
0: potholes	0.50	0.25	0.33	8
1: normal	0.89	0.96	0.93	52
accuracy			0.87	60
marco avg	0.70	0.61	0.63	60
weighted avg	0.84	0.87	0.85	60

外，该模型对于坑洼类别的精确率、召回率、F1 分数值均较低，即说明该模型效果较为一般。

- CNN

通过查看 Jupyter 的代码执行记录：CNN 模型在训练集训练 50 轮耗时 $5\text{m} + 26\text{s} + 888\text{ms}$ ，在测试集测试耗时 435 ms。通过在测试集的测试，经计算，该模型准确率为：88.97%。此外，我们还得出其分类报告表，如表 5 所示。

表 5 CNN 分类报告表

维度	precision	recall	f1-score	support
0: potholes	0.90	0.88	0.89	68
1: normal	0.88	0.90	0.89	68
accuracy			0.89	136
marco avg	0.89	0.89	0.89	136
weighted avg	0.89	0.89	0.89	136

观察该表，我们可以发现，该模型在新数据集上表现效果较优秀，各项指标均较为理想。此外，我们还绘制其在训练过程中在训练集及测试集上的准确率与损失值，如图 13 所示。通过观察该图，我们可以发现，随着迭代次数的增加，训练集上的准确率平稳而缓慢增加，测试集上的准确度经过起伏后也为上升趋势。而随着迭代次数增加，其训练集上的损失逐渐减少，而在测试集上其损失呈波动性，这可能是由于数据集过少，即缺乏代表性样本^[15]，易造成过拟合。

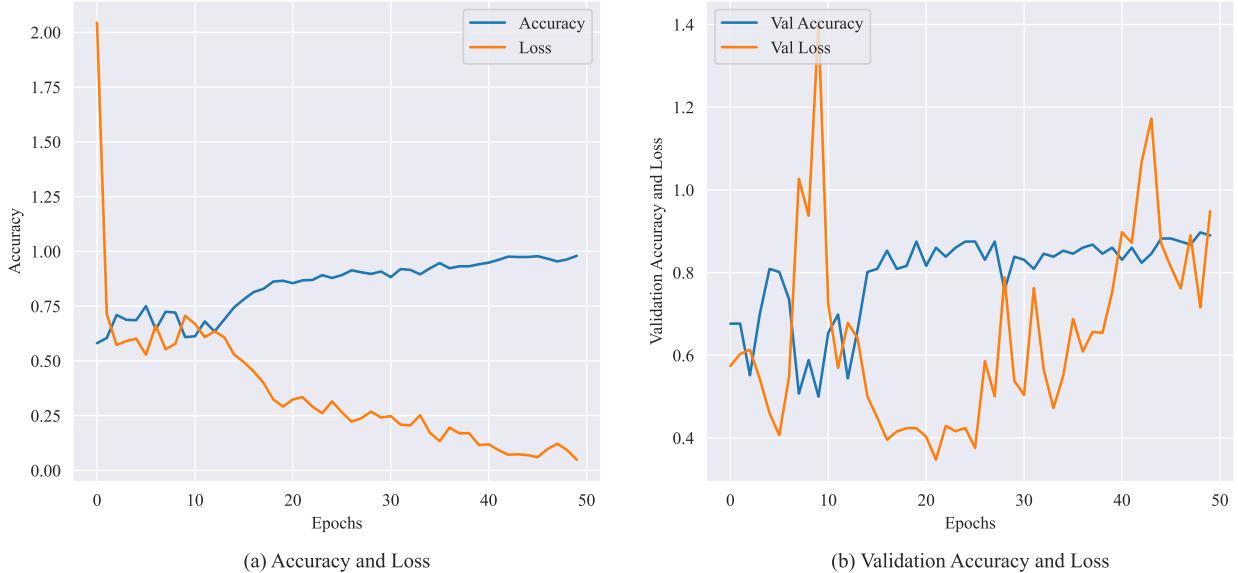


图 13 CNN 训练结果

- **CNN-SVM**

CNN 模型训练耗时 $5\text{ m} + 26\text{ s} + 888\text{ ms}$, 提取图像特征耗时 482 ms; SVM 分类器在所提取的特征上训练耗时 48 ms, 在测试集上得出结果耗时 28 ms。通过在测试集的测试, 经计算, 该模型准确率为: 90 %。该模型分类报告表如表 6 所示。此外, 为了方便查看, 我们还绘制其分类报告图, 如图 14 所示。观察其分类报告, 我们可以发现, 该模型无论是对于正常或者是坑洼道路, 其均能较好地识别, 精度较高, 并且召回率与 F1 分数值均较优秀。

表 6 CNN-SVM 分类报告表

维度	precision	recall	f1-score	support
0: potholes	0.89	0.93	0.91	68
1: normal	0.92	0.88	0.90	68
accuracy			0.90	136
macro avg	0.91	0.90	0.90	136
weighted avg	0.91	0.90	0.90	136

此外我们还根据模型在测试集上的预测结果与实际真实值进行比较, 绘制出模型的混淆矩阵热力图, 如图 15 所示。观察该图, 我们可以发现: 对于坑洼道路检测中, 有 63 张图像的检测结果为坑洼, 5 张的检测结果为正常, 即在 68 张图片为坑洼道路的图像中, 识别正确的为 63 张, 正确率达到 92.48 %; 对于正常道路的检测中, 有 60 张图像的检测结果为正常, 8 张图像的检测结果为坑洼, 即在 68 张为正常的道路图片中, 识别正确的有 60 张, 正确率为 88.24 %, 由此可见, 此模型的效果较好。

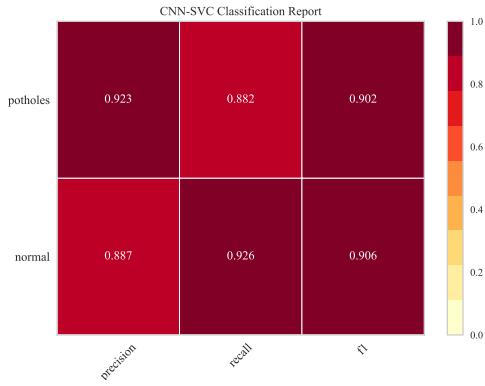


图 14 CNN-SVM 分类报告图

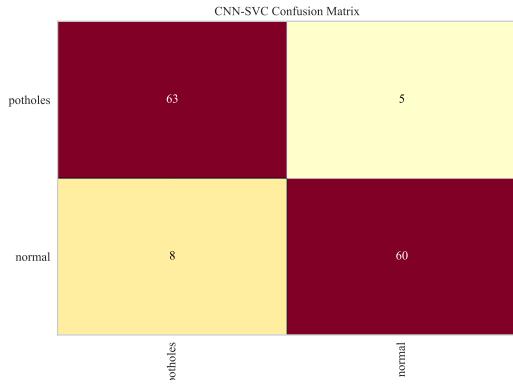


图 15 CNN-SVM 混淆矩阵热力图

同时, 我们还绘制模型的 ROC/AUC 曲线以及精确率-召回率曲线, 分别如图 16 与图 17 所示。由图可知, 该模型的 ROC/AUC 曲线主要分布在左上方, 曲线下面积较大, 即 AUC 值较高, 为 0.90; 同时, 该模型的精确率与召回率都位于较高的数值, 说明该模型在对道路是否坑洼的识别上准确度高, 模型效果较好。

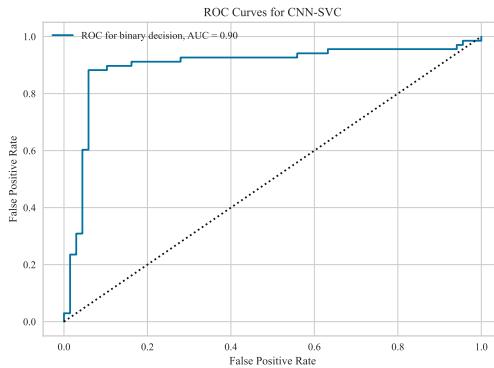


图 16 CNN-SVM ROC 曲线

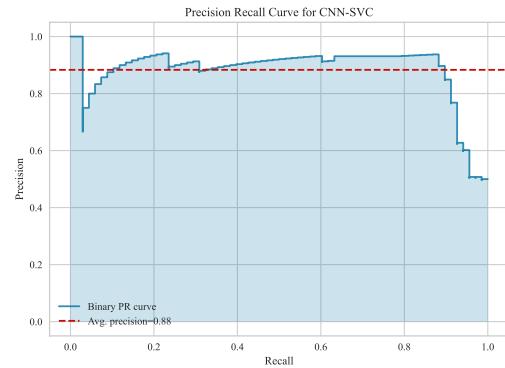


图 17 CNN-SVM 精确率-召回率曲线

此外, 我们还对该模型进行了五折交叉验证, 具体结果为:

[0.96428571, 0.88888889, 0.88888889, 0.85185185, 0.92592593]

因而, 可以得到该模型的准确率在置信度为 95 % 的水平下, 置信区间为 [82 %, 98 %]。

5.6.2 模型对比

未选择最优模型, 我们依据[三种模型的效果逐一分析](#)部分, 将各模型指标整理至一张表格中, 如表 7 所示。同时结合图 14~图 17, 我们可以发现, 在三者中, CNN-SVM 模型的效果最优, 其各项指标均较优秀。

5.7 未知数据集的预测

对于该问题, 具体处理代码见[附录-C.7, Predict \[预测\]](#)。我们按以下步骤进行处理:

表 7 模型效果对比

指标	计算维度	PCA-SVM	CNN	CNN-SVM
precision	0: potholes	0.50	0.90	0.89
	1: normal	0.89	0.88	0.92
	macro	0.70	0.89	0.91
	weighted	0.84	0.89	0.91
recall	0: potholes	0.25	0.88	0.93
	1: normal	0.96	0.90	0.88
	macro	0.61	0.89	0.90
	weighted	0.87	0.89	0.90
f1-score	0: potholes	0.33	0.89	0.91
	1: normal	0.93	0.89	0.90
	macro	0.63	0.89	0.90
	weighted	0.85	0.89	0.90
accuracy	整体	0.87	0.89	0.90
T	共计	6 m 8 s 197 ms	5 m 27 s 323 ms	6 m 15 s 37 ms

- **读取已建立、训练完成的 CNN-SVM 模型:** 将上述 CNN-SVM 模型利用 keras 及 pickle 库读取；
- **读取未知数据集:** 将未知数据集的图像数据及其文件名（包括后缀名）一并读入，方便后续结果的输出，共计 4942 张图像；
- **对未知数据集进行预处理:** 为了准确预测，需要将该数据集格式转换为前期训练模型输入的格式，避免预测错误。
- **对未知数据集进行预测:** 首先传入 CNN 卷积神经网络，对数据集的特征进行提取；之后将提取的特征数据集传入 SVM 支持向量机中，进行预测；
- **结果查看:** 经过上述步骤，将文件名与预测结果一一对应转为表格数据，方便查看；通过预测，模型在 4942 张图像中，识别出坑洼道路为 3765 张，识别为正常道路为 1177 张。
- **保存结果:** 确认无误后，将上述表格输出为“test_result.csv”文件。

六、模型的评价与推广

6.1 模型的评价

• 模型的优点

1. 未完全套用深度学习及传统机器学习模型，而是将 CNN 与 SVM 结合，使得模型能够处理非线性且稀疏数据并且对于权重的处理更加准确；
2. 建立不同的模型进行对比分析，使结果更加准确；
3. 在图像文件预处理时，对数据进行了人为再分类，使数据更准确有效；

4. 在图像数据预处理时，对训练集进行高斯、双边滤波与旋转增强，提升模型的泛化能力；
5. 由于所给原数据集分类样本不平衡，我们使用了采样的方法，适当增加数据集，使得原数据集为新数据集的子集，且分类样本平衡化。

- **模型的缺点**

1. 在对训练集进行数据增强时，仅使用了三种增强方法，在对特征进行学习时可能会有所遗漏；
2. 由于所给原数据集分类样本不平衡，我们使用了采样的方法，适当增加数据集，使得原数据集为新数据集的子集，且分类样本平衡化。然后，该方法成本较高，对于数据集较少的任务，难以推广使用。

- **模型的改进**

1. 由于所给原数据集分类样本不平衡，除本文方法外，还可在模型中使用可选度量法和代价敏感学习进行解决 [15]；
2. 对于 CNN 神经网络，可选择其他更适合本题的架构使模型更加精确；
3. 在进行预测时，可以结合优化算法，如贝叶斯调优进一步优化模型超参数，以提高预测精度。其主要方法为：利用已知的超参数 x 和模型结果 y 来拟合一个代理模型，再利用采集函数选择下一个最优 (x, y) ，并用此新的 (x, y) 来优化代理模型，重复上述过程，最终得到调优后的参数，其算法伪代码如 Algorithm 1 所示。
4. 对于本文的 SVM 模型，我们可用其他分类效果更好的模型，如 Stacking 多模型融合集成学习。

Algorithm 1: 贝叶斯优化框架

Input: 初始点个数 n_0 , 最大迭代次数 N , 代理模型 $g(x)$, 采集函数 $\alpha(x|D)$

Output: 最优候选评估点: $\{x^*, y^*\}$

```

1 Step 1: 随机初始化  $n_0$  点  $X_{\text{init}} = \{x_0, x_1, \dots, x_{n_0-1}\}$ 
2 Step 2: 获取其对应的函数值  $f(X_{\text{init}})$ , 初始点集  $D_0 = \{X_{\text{init}}, f(X_{\text{init}})\}$ , 令  $t = n_0$ ,
    $D_{t-1} = D_0$ 
3 while  $t < N$  do
4   Step 3: 根据当前获得的点集  $D_{t-1}$ , 构建代理模型  $g(x)$ 
5   Step 4: 基于代理模型  $g(x)$ , 最大化采集函数  $\alpha(x|D_{t-1})$ , 获得下一个评估点:
      $x_t = \operatorname{argmin} \alpha(x|D_{t-1})$ 
6   Step 5: 获得评估点  $x_t$  的函数值  $f(x_t)$ , 将其加入到当前评估点集合中:
      $D_t = D_{t-1} \cup \{x_t, f(x_t)\}$ , 转 Step 3
7 end
Result: 最优候选评估点:  $\{x^*, y^*\}$ 

```

6.2 模型的推广

此模型具备处理具有复杂特征图像的能力，且能够较好地进行分类，因此除了判别道路坑洼与否外，我们也可将此其应用于重大基础设施建设前的勘探过程中，对工程建设地区的

形进行判别，准确且快速地确定该地区是否适合建设该工程。以此类推，除了工程方面，此模型还可应用于交通预警，此模型适用于各地交管局及城市建设市政管理部门。在一定程度上改善了当前的预警机制，为减少民众财产损失做出了贡献。

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] 刘建新, 史志仙. 概率论与数理统计 [M]. 北京: 高等教育出版社,2016:115.
- [3] 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.
- [4] 张松兰, 王鹏, 徐子伟. 基于统计相关的缺失值数据处理研究 [J]. 统计与决策,2016,No.456(12):13-16.
- [5] 李莎. 基于改进卷积神经网络的上市企业财务风险预测研究 [J]. 现代信息科技,2023,7(20):111-115.
- [6] 孟琪琳, 窦燕. 基于 EMD-CNN-LSTM 模型的铁路客运量短期预测研究 [J/OL]. 铁道运输与经济:1-9[2023-10-30].
- [7] ZHANG P,FANG Y.Research on text classification algorithm based on machine learning[J].Journal of Physics Conference Series,2020,1624:042010.
- [8] 李毅泉. 基于注意力机制的显著区域提取研究和实现 [D]. 北京交通大学,2007.
- [9] 何铠, 管有庆, 龚锐. 基于深度学习和支持向量机的文本分类模型 [J]. 计算机技术与发展,2022,32(07):22-27.
- [10] 刘幸倩. 基于注意力增强卷积神经网络的滚动轴承故障诊断方法研究 [D]. 东北石油大学,2023.
- [11] CSDN. 损失函数: categorical_crossentropy[EB/OL].https://blog.csdn.net/qq_40661327/article/details/107034575.
- [12] 知乎. 模型评测:PRECISION、RECALL、F1-score[EB/OL].<https://zhuanlan.zhihu.com/p/519982682>.
- [13] A.Tharwat, Applied Computing and Informatics (2018). <https://doi.org/10.1016/j.aci.2018.08.003>.
- [14] Yellowbrick.Precision-Recall Curves - Yellowbrick v1.5 documentation[EB/OL].<https://www.scikit-yb.org/en/latest/api/classifier/prcurve.html>.
- [15] Tan P N, Steinbach M, Karpatne A, et al. Introduction to Data Mining[M]. Pearson, 2019:389.

附录

[A] 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
html	包括所有解决问题的源程序运行结果
ipynb	包括所有解决问题的源程序源代码
models	已训练好的模型文件
process	包括所有模型训练、预测的过程结果
py	包括所有解决问题的源程序输出 python 文件
yaml	包括所有模型训练的配置文件
test_result2.csv	坑洼面积占比估算结果

[B] 使用的软件、环境

B.1 使用的软件及版本

- TeX Live 2022
- Visual Studio Code 1.85.0
- WPS Office 2023 冬季更新 (15990)
- Python 3.10.4 [MSC v.1929 64 bit (AMD64)] on win32
- Pycharm 2023.3 (Professional Edition)

B.2 模型训练所用计算机配置

- Intel(R) Core(TM) i5-10200H CPU @ 2.40GHz
- NVIDIA GeForce GTX 1650 Ti
- NVIDIA CUDA 11.7.102 driver
- 16.0 GB RAM
- Windows 10 家庭中文版 22H2

Python 环境下所用使用到的库及其版本

库	版本	库	版本
collections	内置库	jupyterlab-widgets	3.0.9
opencv-python	4.8.1.78	keras	2.14.0
h5py	3.10.0	matplotlib	3.8.0
jupyter	1.0.0	numpy	1.26.1
jupyter_client	8.5.0	os	内置库
jupyter-console	6.6.3	pandas	2.1.2
jupyter_core	5.4.0	pickle	内置库
jupyter-events	0.8.0	random	内置库
jupyter-lsp	2.2.0	shutil	内置库
jupyter_server	2.9.1	sklearn	1.3.2
jupyter_server_terminals	0.4.4	tensorflow	2.14.0
jupyterlab	4.0.7	warnings	内置库
jupyterlab-pygments	0.2.2	yellowbrick	1.50
jupyterlab_server	2.25.0		

[C] 问题解决源程序

C.1 ista2txt [ista 转 txt]

```
1 import json
2 import os
3 import shutil
4
5 category_mapping = {"potholes": 0}
6 # ISAT格式的实例分割标注文件
7 ISAT_FOLDER = "./annotations"
8 # YOLO格式的实例分割标注文件
9 YOLO_FOLDER = "./labels"
10
11 # 创建YoloV8标注的文件夹
12 if not os.path.exists(YOLO_FOLDER):
13     os.makedirs(YOLO_FOLDER)
14
15
16 # 载入所有的ISAT的JSON文件
17 for filename in os.listdir(ISAT_FOLDER):
18     if not filename.endswith(".json"):
19         # 不是json格式，跳过
20         continue
21     # 载入ISAT的JSON文件
22     with open(os.path.join(ISAT_FOLDER, filename), "r") as f:
23         isat = json.load(f)
24     # 提取文件名(不带文件后缀)
25     image_name = filename.split(".")[0]
26     # Yolo格式的标注文件名，后缀是txt
27     yolo_filename = f"{image_name}.txt"
28     # 写入信息
29     with open(os.path.join(YOLO_FOLDER, yolo_filename), "w") as f:
30         # 获取图像信息
31         # - 图像宽度
32
33         image_width = isat["info"]["width"]
34
35         # try:
36             #     image_width = isat["info"]["width"]
37         # except KeyError:
38             #     break
39         # - 图像高度
40         image_height = isat["info"]["height"]
41     # 获取实例标注数据
42     for annotation in isat["objects"]:
```

```

43     # 获取类别名称
44     category_name = annotation["category"]
45     # 如果不在类别名称字典里面，跳过
46     if category_name not in category_mapping:
47         continue
48     # 从字典里面查询类别ID
49     category_id = category_mapping[category_name]
50     # 提取分割信息
51     segmentation = annotation["segmentation"]
52     segmentation_yolo = []
53     # 遍历所有的轮廓点
54     for segment in segmentation:
55         # 提取轮廓点的像素坐标 x, y
56         x, y = segment
57         # 归一化处理
58         x_center = x / image_width
59         y_center = y / image_height
60         # 添加到segmentation_yolo里面
61         segmentation_yolo.append(f"{x_center:.4f} {y_center:.4f}")
62     segmentation_yolo_str = " ".join(segmentation_yolo)
63     # 添加一行Yolo格式的实例分割数据
64     # 格式如下: class_id x1 y1 x2 y2 ... xn yn\n
65     f.write(f"{category_id} {segmentation_yolo_str}\n")

```

C.2 xml2txt [xml 转 txt]

```

1 import xml.etree.ElementTree as ET
2 import os, cv2
3 import numpy as np
4 from os import listdir
5 from os.path import join
6
7 # 将给定的XML格式的标注文件转换为YOLOv8所需的TXT格式，并记录数据集中的类别信息
8 classes = []
9
10 def convert(size, box):
11     dw = 1. / (size[0])
12     dh = 1. / (size[1])
13     x = (box[0] + box[1]) / 2.0 - 1
14     y = (box[2] + box[3]) / 2.0 - 1
15     w = box[1] - box[0]
16     h = box[3] - box[2]
17     x = x * dw
18     w = w * dw
19     y = y * dh

```

```

20     h = h * dh
21     return (x, y, w, h)
22
23
24 def convert_annotation(xmlpath, xmlname):
25     with open(xmlpath, "r", encoding='utf-8') as in_file:
26         txtname = xmlname[:-4] + '.txt'
27         txtfile = os.path.join(txtpath, txtname)
28         tree = ET.parse(in_file)
29         root = tree.getroot()
30         filename = root.find('filename')
31         img = cv2.imdecode(np.fromfile('{}/{}/{}'.format(imgpath, xmlname[:-4], postfix)
32                                     , np.uint8), cv2.IMREAD_COLOR)
33         h, w = img.shape[:2]
34         res = []
35         for obj in root.iter('object'):
36             cls = obj.find('name').text
37             if cls not in classes:
38                 classes.append(cls)
39             cls_id = classes.index(cls)
40             xmlbox = obj.find('bndbox')
41             b = (float(xmlbox.find('xmin').text), float(xmlbox.find('xmax').text), float(
42                                         xmlbox.find('ymin').text),
43                                         float(xmlbox.find('ymax').text)))
44             bb = convert((w, h), b)
45             res.append(str(cls_id) + " " + ".join([str(a) for a in bb]))
46         if len(res) != 0:
47             with open(txtfile, 'w+') as f:
48                 f.write('\n'.join(res))
49
50 if __name__ == "__main__":
51     postfix = 'jpg'
52     imgpath = 'VOCdevkit/JPEGImages'
53     xmlpath = 'VOCdevkit/Annotations'
54     txtpath = 'VOCdevkit/txt'
55
56     if not os.path.exists(txtpath):
57         os.makedirs(txtpath, exist_ok=True)
58
59     list = os.listdir(xmlpath)
60     error_file_list = []
61     for i in range(0, len(list)):
62         try:

```

```

62         path = os.path.join(xmlpath, list[i])
63         if ('.xml' in path) or ('.XML' in path):
64             convert_annotation(path, list[i])
65             print(f'file {list[i]} convert success.')
66         else:
67             print(f'file {list[i]} is not xml format.')
68     except Exception as e:
69         print(f'file {list[i]} convert error.')
70         print(f'error message:\n{e}')
71         error_file_list.append(list[i])
72     print(f'this file convert failure\n{error_file_list}')
73     print(f'Dataset Classes:{classes}')

```

C.3 Segment Spilt Data [图像分割数据集划分]

```

1 import os
2 import random
3 from tqdm import tqdm
4
5 # 指定 images 文件夹路径
6 image_dir = "./images"
7 # 指定 labels 文件夹路径
8 label_dir = "./labels"
9 # 创建一个空列表来存储有效图片的路径
10 valid_images = []
11 # 创建一个空列表来存储有效 label 的路径
12 valid_labels = []
13
14 # 遍历 images 文件夹下的所有图片
15 for image_name in os.listdir(image_dir):
16     # 获取图片的完整路径
17     image_path = os.path.join(image_dir, image_name)
18     # 获取图片文件的扩展名
19     ext = os.path.splitext(image_name)[-1]
20     # 根据扩展名替换成对应的 label 文件名
21     label_name = image_name.replace(ext, ".txt")
22     # 获取对应 label 的完整路径
23     label_path = os.path.join(label_dir, label_name)
24     # 判断 label 是否存在
25     if not os.path.exists(label_path):
26         # 删除图片
27         os.remove(image_path)
28         print("deleted:", image_path)
29     else:
30         # 将图片路径添加到列表中

```

```

31     valid_images.append(image_path)
32     # 将label路径添加到列表中
33     valid_labels.append(label_path)
34     # print("valid:", image_path, label_path)
35 # 遍历每个有效图片路径
36 for i in tqdm(range(len(valid_images))):
37     image_path = valid_images[i]
38     label_path = valid_labels[i]
39     # 随机生成一个概率
40     r = random.random()
41     # 判断图片应该移动到哪个文件夹
42     # train: valid: test = 7:2:1
43     if r < 0.1:
44         # 移动到 test 文件夹
45         destination = "./datasets/test"
46     elif r < 0.3:
47         # 移动到 valid 文件夹
48         destination = "./datasets/valid"
49     else:
50         # 移动到 train 文件夹
51         destination = "./datasets/train"
52     # 生成目标文件夹中图片的新路径
53     image_destination_path = os.path.join(destination, "images", os.path.basename(
54         image_path))
55     # 移动图片到目标文件夹
56     os.rename(image_path, image_destination_path)
57     # 生成目标文件夹中 label 的新路径
58     label_destination_path = os.path.join(destination, "labels", os.path.basename(
59         label_path))
60     # 移动 label 到目标文件夹
61     os.rename(label_path, label_destination_path)
62     print("valid images:", valid_images)
63     # 输出有效label路径列表
64     print("valid labels:", valid_labels)

```

C.4 Detect Spilt Data [图像检测数据集划分]

```

1 import os, shutil
2 from sklearn.model_selection import train_test_split
3
4 # 用于将图像和标注文件按照一定比例划分为训练集、验证集和测试集，并将它们分别复制到相应的文
5    件夹中
6 val_size = 0.1
7 test_size = 0.1
8 postfix = 'jpg'

```

```

8 imgpath = 'VOCdevkit/JPEGImages'
9 txtpath = 'VOCdevkit/txt'
10
11 os.makedirs('images/train', exist_ok=True)
12 os.makedirs('images/val', exist_ok=True)
13 os.makedirs('images/test', exist_ok=True)
14 os.makedirs('labels/train', exist_ok=True)
15 os.makedirs('labels/val', exist_ok=True)
16 os.makedirs('labels/test', exist_ok=True)
17
18 listdir = [i for i in os.listdir(txtpath) if 'txt' in i]
19 train, test = train_test_split(listdir, test_size=test_size, shuffle=True, random_state=0)
20 train, val = train_test_split(train, test_size=val_size, shuffle=True, random_state=0)
21 print(f'train set size:{len(train)} val set size:{len(val)} test set size:{len(test)}')
22
23 for i in train:
24     shutil.copy('{}/{}.{}'.format(imgpath, i[:-4], postfix), 'images/train/{}/{}'.format(i[:-4], postfix))
25     shutil.copy('{}/{}'.format(txtpath, i), 'labels/train/{}'.format(i))
26
27 for i in val:
28     shutil.copy('{}/{}.{}'.format(imgpath, i[:-4], postfix), 'images/val/{}/{}'.format(i[:-4], postfix))
29     shutil.copy('{}/{}'.format(txtpath, i), 'labels/val/{}'.format(i))
30
31 for i in test:
32     shutil.copy('{}/{}.{}'.format(imgpath, i[:-4], postfix), 'images/test/{}/{}'.format(i[:-4], postfix))
33     shutil.copy('{}/{}'.format(txtpath, i), 'labels/test/{}'.format(i))

```

C.5 YOLOv8s Segment [YOLOv8s 图像分割]

```

1 #!/usr/bin/env python
2 # coding: utf-8
3
4 # In[1]:
5
6
7 from ultralytics import YOLO
8
9 # In[3]:
10
11
12 model = YOLO('yolov8s-seg.pt')

```

```

13 results = model.train(data="dataset/VOCdevkit/datasets/potholes.yaml", epochs=300,
14                         device='cpu')
15
16 # In[2]:
17
18 model = YOLO('potholesSegment.pt')
19
20 # In[3]:
21
22
23 resultOne = model('potholes\\part1', save=True)
24
25 # In[4]:
26
27
28 resultTwo = model('potholes\\part2', save=True)
29
30 # In[5]:
31
32
33 resultThree = model('potholes\\part3', save=True)
34
35 # In[3]:
36
37
38 resultFour = model('potholes\\part4', save=True)
39
40 # In[4]:
41
42
43 resultFive = model('potholes\\part5', save=True)
44
45 # In[3]:
46
47
48 resultSix = model('potholes\\part6', save=True)
49
50 # In[4]:
51
52
53 resultSeven = model('potholes\\part7', save=True)
54
55 # In[3]:

```

```
56  
57  
58 resultEight = model('potholes\\part8', save=True)
```

C.6 YOLOv8n Detect [YOLOv8n 图像检测]

```
1 #!/usr/bin/env python  
2 # coding: utf-8  
3  
4 # In[1]:  
5  
6  
7 from ultralytics import YOLO  
8  
9  
10 # In[47]:  
11  
12  
13 model = YOLO("potholesDetect.pt")  
14 resultsOne = model.predict(source="potholes\\partOne", save=True, save_conf=True,  
    save_txt=True, imgsz=416)  
15  
16  
17 # In[48]:  
18  
19  
20 model = YOLO("potholesDetect.pt")  
21 resultsTwo = model.predict(source="potholes\\partTwo", save=True, save_conf=True,  
    save_txt=True, imgsz=416)
```

C.7 Coordinate Transformation [坐标转换]

```
1 #!/usr/bin/env python  
2 # coding: utf-8  
3  
4 # In[1]:  
5  
6  
7 import os  
8 import cv2  
9 import torch  
10 import numpy as np  
11  
12 # In[2]:  
13  
14
```

```

15 label_path = 'potholes\labels'
16 image_path = 'potholes\images'
17
18
19 def xywhn2xyxy(x, w=416, h=416, padw=0, padh=0):
20     y = x.clone() if isinstance(x, torch.Tensor) else np.copy(x)
21     y[:, 0] = w * (x[:, 0] - x[:, 2] / 2) + padw # top left x
22     y[:, 1] = h * (x[:, 1] - x[:, 3] / 2) + padh # top left y
23     y[:, 2] = w * (x[:, 0] + x[:, 2] / 2) + padw # bottom right x
24     y[:, 3] = h * (x[:, 1] + x[:, 3] / 2) + padh # bottom right y
25     return y
26
27
28 folder = os.path.exists('potholes\\labelsNew')
29 if not folder:
30     os.makedirs('potholes\\labelsNew')
31
32 folderlist = os.listdir(label_path)
33 for i in folderlist:
34     label_path_new = os.path.join(label_path, i)
35     with open(label_path_new, 'r') as f:
36         lb = np.array([x.split() for x in f.read().strip().splitlines()], dtype=np.
37                     float32) # labels
38         print(lb)
39         read_label = label_path_new.replace(".txt", ".jpg")
40         read_label_path = read_label.replace(label_path, image_path)
41         print(read_label_path, "ddd")
42         img = cv2.imread(str(read_label_path))
43         h, w = img.shape[:2]
44         lb[:, 1:] = xywhn2xyxy(lb[:, 1:], w, h, 0, 0)
45
46         for _, x in enumerate(lb):
47             class_label = int(x[0])
48
49             with open('potholes\\labelsNew\\' + i, 'a') as fw:
50                 fw.write(str(int(x[0])) + ' ' + str(x[5]) + ' ' + str(x[1]) + ' ' +
51                         str(x[2]) + ' ' + str(x[3]) + ' ' + str(
52                             x[4]) + '\n')

```

C.8 Proportional Calculation [面积占比估算]

```

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 pandas as pd
11
12 # In[2]:
13
14
15 labelsList = os.listdir('potholes\\labelsNew')
16 imagesList = os.listdir('potholes\\images')
17
18 # In[3]:
19
20
21 labelsDf = pd.DataFrame(labelsList, columns=['fnames'])
22 labelsDf
23
24 # In[4]:
25
26
27 Npotholes = pd.read_csv('potholes\\detectResults.txt', sep=' ', header=None)
28 Npotholes = Npotholes.drop([0, 1, 3, 5, 6], axis=1)
29 Npotholes.columns = ['fnames', 'Npotholes']
30 Npotholes['fnames'] = Npotholes['fnames'].str.split('\\').str[-1]
31 Npotholes['Npotholes'] = Npotholes['Npotholes'].str.replace('(', '')
32 Npotholes['Npotholes'] = Npotholes['Npotholes'].str.replace(')', '')
33 Npotholes
34
35 # In[5]:
36
37
38 Npotholes['Npotholes'] = Npotholes['Npotholes'].str.replace('no', '0')
39 Npotholes['Npotholes'] = Npotholes['Npotholes'].astype(int)
40 Npotholes
41
42 # In[6]:
43
44
45 data = pd.DataFrame(columns=['Type', '置信度', 'x1', 'y1', 'x2', 'y2', 'fnames'])
46 data
47
48 # In[7]:

```

```

49
50
51 for i in labelsList:
52     with open('potholes\\labelsNew\\' + i, 'r') as f:
53         content = f.readlines()
54         lb = np.array([x.strip().split() for x in content], dtype=np.float32) # labels
55         lb = pd.DataFrame(lb, columns=['Type', '置信度', 'x1', 'y1', 'x2', 'y2'])
56         lb['fnames'] = i
57
58     data = data._append(lb, ignore_index=True)
59
60 data
61
62 # In[8]:
63
64
65 data = data.drop('Type', axis=1)
66 data = data[['fnames', '置信度', 'x1', 'y1', 'x2', 'y2']]
67 data['fnames'] = data['fnames'].str.replace('.txt', '.jpg')
68 data
69
70 # In[9]:
71
72
73 data = pd.merge(data, Npotholes, on='fnames', how='right')
74 data
75
76 # In[10]:
77
78
79 potholesImages = pd.DataFrame(imagesList, columns=['fnames'])
80 potholesImages
81
82 # In[11]:
83
84
85 for i in imagesList:
86     img = cv2.imread('potholes\\images\\' + i)
87     height, width, channels = img.shape
88     potholesImages.loc[potholesImages['fnames'] == i, 'height'] = height
89     potholesImages.loc[potholesImages['fnames'] == i, 'width'] = width
90
91 potholesImages
92

```

```

93 # In[12]:
94
95
96 data = pd.merge(data, potholesImages, on='fnames', how='left')
97 data
98
99 # In[13]:
100
101
102 data['S'] = data['height'] * data['width']
103 data['X'] = data['x2'] - data['x1']
104 data['Y'] = data['y2'] - data['y1']
105 data['Spotholes'] = data['X'] * data['Y']
106 data['Spotholes/S'] = data['Spotholes'] / data['S'] * 100
107 data
108
109 # In[14]:
110
111
112 data['Spotholes/S'] = data['Spotholes/S'].fillna(data['Spotholes/S'].mean())
113 data['Spotholes/S'] = data['Spotholes/S'].apply(np.ceil)
114 data['Spotholes/S'] = data['Spotholes/S'].astype('int')
115 data
116
117 # In[15]:
118
119
120 dataNew = data.groupby('fnames').apply(lambda x: x.loc[x['Spotholes/S'].idxmax()])
121 dataNew
122
123 # In[16]:
124
125
126 dataNew = dataNew.reset_index(drop=True)
127 dataNew
128
129 # In[17]:
130
131
132 dataR = pd.read_csv('未知数据集预测效果.csv')
133 dataR = dataR.drop(['imgType', 'label', 'isTrue'], axis=1)
134 dataR
135
136 # In[18]:

```

```

137
138
139 Results = pd.merge(dataR, dataNew, on='fnames', how='left')
140 Results
141
142 # In[19]:
143
144
145 # 修改guy7iodk.jpg行, height=1587, width=1200, S=1587*1200=1904400, Spotholes=954792,
    Spotholes/S=50
146 Results.loc[Results['fnames'] == 'guy7iodk.jpg', 'height'] = 1587
147 Results.loc[Results['fnames'] == 'guy7iodk.jpg', 'width'] = 1200
148 Results.loc[Results['fnames'] == 'guy7iodk.jpg', 'S'] = 1904400
149 Results.loc[Results['fnames'] == 'guy7iodk.jpg', 'Spotholes'] = 954792
150 Results.loc[Results['fnames'] == 'guy7iodk.jpg', 'Spotholes/S'] = 50
151 Results
152
153 # In[20]:
154
155
156 Results.loc[Results['imgClass'] == 'normal', 'Spotholes/S'] = 0
157 Results
158
159 # In[21]:
160
161
162 Results['Spotholes/S'] = Results['Spotholes/S'].astype('int')
163 Results
164
165 # In[22]:
166
167
168 Results.to_csv('SpotholesS.csv', index=False)

```

C.9 Detect Segment Analyze [图像检测与图像分割效果分析]

```

1 #!/usr/bin/env python
2 # coding: utf-8
3
4 # In[1]:
5
6
7 import pandas as pd
8
9 dataDetect = pd.read_csv('detectResults.txt', sep=' ', header=None)

```

```

10 dataSegment = pd.read_csv('segmentResults.txt', sep=' ', header=None)
11
12 # In[2]:
13
14
15 dataDetect = dataDetect.drop([0, 1, 3, 5, 6], axis=1)
16 dataDetect.columns = ['fnames', 'Detect-Npotholes']
17 dataDetect['fnames'] = dataDetect['fnames'].str.split('\\').str[-1]
18 dataDetect['Detect-Npotholes'] = dataDetect['Detect-Npotholes'].str.replace('(', '')
19 dataDetect['fnames'] = dataDetect['fnames'].str.replace(':', '')
20 dataDetect['Detect-Npotholes'] = dataDetect['Detect-Npotholes'].str.replace('no', '0')
21 dataDetect['Detect-Npotholes'] = dataDetect['Detect-Npotholes'].astype('int')
22 dataDetect
23
24 # In[3]:
25
26
27 dataSegment = dataSegment.drop([0, 1, 3, 5, 6], axis=1)
28 dataSegment.columns = ['fnames', 'Segment-Npotholes']
29 dataSegment['fnames'] = dataSegment['fnames'].str.split('\\').str[-1]
30 dataSegment['Segment-Npotholes'] = dataSegment['Segment-Npotholes'].str.replace('(', '')
31 dataSegment['fnames'] = dataSegment['fnames'].str.replace(':', '')
32 dataSegment['Segment-Npotholes'] = dataSegment['Segment-Npotholes'].str.replace('no', '0')
33 dataSegment['Segment-Npotholes'] = dataSegment['Segment-Npotholes'].astype('int')
34 dataSegment
35
36 # In[4]:
37
38
39 data = pd.merge(dataDetect, dataSegment, on='fnames', how='left')
40 data
41
42 # In[5]:
43
44
45 # DetectFalse为Detect-Npotholes为0的个数
46 # DetectTrue为Detect-Npotholes不为0的个数
47 # SegmentFalse为Segment-Npotholes为0的个数
48 # SegmentTrue为Segment-Npotholes不为0的个数
49 DetectFalse = 0
50 DetectTrue = 0
51 SegmentFalse = 0

```

```

52 SegmentTrue = 0
53 for i in range(len(data)):
54     if data['Detect-Npotholes'][i] == 0:
55         DetectFalse += 1
56     else:
57         DetectTrue += 1
58     if data['Segment-Npotholes'][i] == 0:
59         SegmentFalse += 1
60     else:
61         SegmentTrue += 1
62
63 # In[6]:
64
65
66 DetectTrue / (DetectFalse + DetectTrue)
67
68 # In[7]:
69
70
71 SegmentTrue / (SegmentFalse + SegmentTrue)

```

C.10 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"))
```

C.11 Comparative Analysis of Normal and Potholes [正常与坑洼道路比较分析]

```
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(potheolesImg)
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([potheolesImg], [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) potheoles1.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\\potheoles1RGB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(potheolesImg.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(potheolesImg, 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(pothelesImg)
131 plt.colorbar()
132 plt.title(' (b) potheles1.jpg', y=-0.3, fontsize=12)
133 plt.savefig('Figures\\potheles1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(pothelesImg, 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) potheles1.jpg', y=-0.2, fontsize=12)
157 plt.savefig('Figures\\potheles1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(pothelesImg, 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(pothelesImg, 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(pothelesImg)
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\\potheles1轮廓检测.pdf', bbox_inches='tight')

```

C.12 Images Display [图像展示]

```

1 #!/usr/bin/env python
2 # coding: utf-8
3
4 # In[1]:
5
6
7 import cv2
8
9 img = cv2.imread("DATA\\potheles\\potheles1.jpg", cv2.IMREAD_COLOR)
10 img = cv2.resize(img, [256, 256])
11
12 Gaussian = cv2.GaussianBlur(img, (3, 3), 1)
13 Bilateral = cv2.bilateralFilter(img, 9, 75, 75)
14 Rotate = cv2.warpAffine(img, cv2.getRotationMatrix2D((img.shape[1] / 2, img.shape[0] /
15 2), 30, 1),
16 (img.shape[1], img.shape[0]))

```

```
17 cv2.imwrite("potholes1_Gaussian.jpg", Gaussian)
18 cv2.imwrite("potholes1_Bilateral.jpg", Bilateral)
19 cv2.imwrite("potholes1_Rotate.jpg", Rotate)
```

[D] 模型训练配置文件

D.1 segment.yaml

```
1 # Train/val/test sets as 1) dir: path/to/imgs, 2) file: path/to/imgs.txt, or 3) list: [
2   path/to/imgs1, path/to/imgs2, ...]
3 # path: ./datasets # dataset root dir
4 train: E:/University/PythonProject/UltralyticsPotholes/dataset/VOCdevkit/datasets/train
5   /images # train images (relative to 'path') 128 images
6 val: E:/University/PythonProject/UltralyticsPotholes/dataset/VOCdevkit/datasets/valid/
7   images # val images (relative to 'path') 128 images
8 test: E:/University/PythonProject/UltralyticsPotholes/dataset/VOCdevkit/datasets/test/
9   images # test images (optional)
10
11 # Classes
12 names:
13   0: potholes
```

D.2 detect.yaml

```
1 # dataset path
2 train: E:/University/PythonProject/UltralyticsMaster/dataset/images/train
3 val: E:/University/PythonProject/UltralyticsMaster/dataset/images/val
4 test: E:/University/PythonProject/UltralyticsMaster/dataset/images/test
5
6 # number of classes
7 nc: 1
8
9 # class names
10 names: ['potholes']
```

D.3 YOLOv8-seg.yaml

```
1 # Parameters
2 nc: 1 # number of classes
3 scales: # model compound scaling constants, i.e. 'model=yolov8n-seg.yaml' will call
4   yolov8-seg.yaml with scale 'n'
5   # [depth, width, max_channels]
6   n: [0.33, 0.25, 1024]
7   s: [0.33, 0.50, 1024]
8   m: [0.67, 0.75, 768]
9   l: [1.00, 1.00, 512]
10  x: [1.00, 1.25, 512]
11
12 # YOLOv8.0n backbone
13 backbone:
14   # [from, repeats, module, args]
15   - [-1, 1, Conv, [64, 3, 2]] # O-P1/2
```

```

15   - [-1, 1, Conv, [128, 3, 2]] # 1-P2/4
16   - [-1, 3, C2f, [128, True]]
17   - [-1, 1, Conv, [256, 3, 2]] # 3-P3/8
18   - [-1, 6, C2f, [256, True]]
19   - [-1, 1, Conv, [512, 3, 2]] # 5-P4/16
20   - [-1, 6, C2f, [512, True]]
21   - [-1, 1, Conv, [1024, 3, 2]] # 7-P5/32
22   - [-1, 3, C2f, [1024, True]]
23   - [-1, 1, SPPF, [1024, 5]] # 9
24
25 # YOLOv8.On head
26 head:
27   - [-1, 1, nn.Upsample, [None, 2, 'nearest']]
28   - [[-1, 6], 1, Concat, [1]] # cat backbone P4
29   - [-1, 3, C2f, [512]] # 12
30
31   - [-1, 1, nn.Upsample, [None, 2, 'nearest']]
32   - [[-1, 4], 1, Concat, [1]] # cat backbone P3
33   - [-1, 3, C2f, [256]] # 15 (P3/8-small)
34
35   - [-1, 1, Conv, [256, 3, 2]]
36   - [[-1, 12], 1, Concat, [1]] # cat head P4
37   - [-1, 3, C2f, [512]] # 18 (P4/16-medium)
38
39   - [-1, 1, Conv, [512, 3, 2]]
40   - [[-1, 9], 1, Concat, [1]] # cat head P5
41   - [-1, 3, C2f, [1024]] # 21 (P5/32-large)
42
43   - [[15, 18, 21], 1, Segment, [nc, 32, 256]] # Segment(P3, P4, P5)

```
