

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

坑洼道路的高效与精确检测可以有效地推进自动无人驾驶、地质勘探、航天科学等领域的研究与应用。由于道路图像的环境背景复杂及多变性，传统的分类算法难以提取有效特征，导致效果较差。因此，本文旨在建立基于卷积神经网络图像特征提取的 SVM 模型，对坑洼道路进行检测与识别。

针对问题一，主要需要结合给定的图像数据进行特征的提取，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。因此，本文首先

针对问题二，主要需要对问题一的模型进行训练，并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。

针对问题三，主要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。

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：计算机视觉；卷积神经网络；支持向量机；特征提取；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
1.3 本文思路	1
二、 问题的分析	2
2.1 问题的整体分析	2
2.2 各问题逐一分析	2
三、 符号说明	3
四、 模型的假设	3
五、 模型的建立与求解	3
5.1 图像文件预处理	3
5.1.1 人为再分类	4
5.1.2 文件夹分类	4
5.2 正常与坑洼道路的比较分析、图像特征提取	5
5.2.1 原图、热力图对比	6
5.2.2 RGB、灰度直方图	6
5.2.3 边缘、轮廓检测	7
5.2.4 阈值分割	8
5.3 图像数据预处理	8
5.3.1 opencv 读取图像文件	8
5.3.2 图像数据两类平衡化	9
5.3.3 划分训练集与测试集	9
5.3.4 训练集数据增强	10
5.4 分类模型的建立	11
5.4.1 基于主成分分析降维、网格调优的支持向量机	11
5.4.2 深度学习： CNN 卷积神经网络	13
5.4.3 CNN-SVM 的分类模型	15
5.5 模型的评估	15
5.5.1 三种模型的效果逐一分析	17
5.5.2 模型对比	19
5.6 未知数据集的预测	20
六、 模型的评价与推广	20
6.1 模型的评价	20
6.2 模型的推广	21
参考文献	22
附录	23

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，并分析结果的合理性。

1.3 本文思路

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。

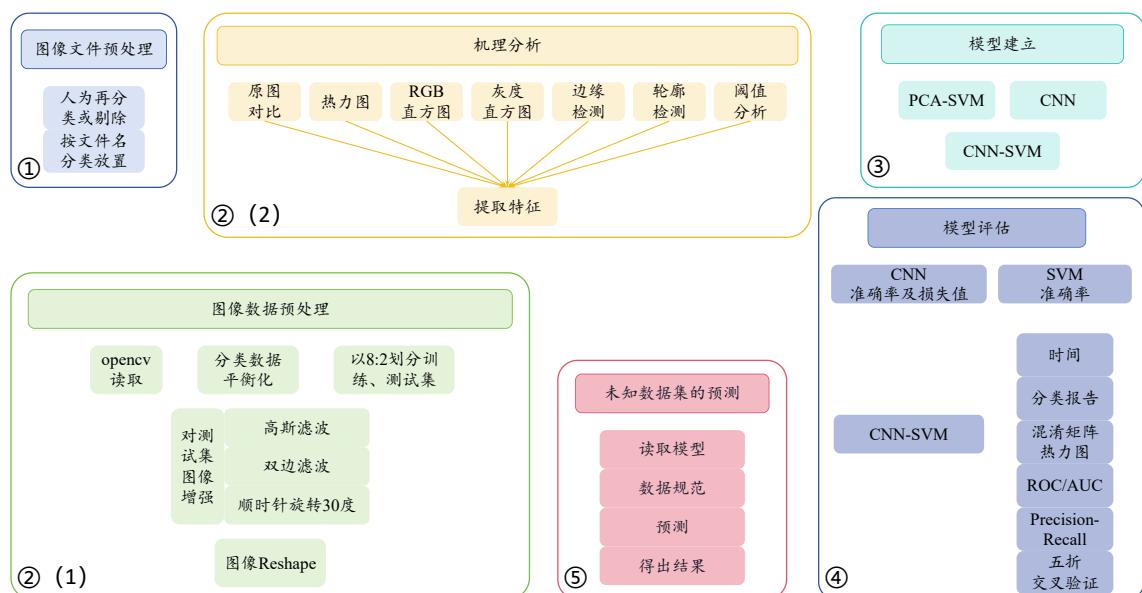


图 1 流程图

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, 为后续模型的建立提供支撑。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。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, 再建立合适的模型进行分类预测。

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。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。之后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 直方图、灰度直方图对比、边缘及轮廓检测、以及阈值分割, 从而有效地提取出图像特征。随后, 对图像数据进行预处理, 将划分标记好的文件读入内存中, 分析各类占比, 对数据集进行平衡化, 在此基础上划分数据集为训练集及测试集, 考虑到图像文件并未能很好地全面展现特征, 因此考虑对训练集数据进行增强, 增加模型的学习样本。完成上述步骤后, 即开始建立模型。我们从三方面考虑, 其一, 利用主成分分析法对数据进行降维, 再建立支持向量机模型, 同时加入网格调优。其二, 建立卷积神经网络深度学习模型, 设定其输入层、卷积层、激活层、池化层、全连接层参数。其三, 考虑到单一模型存在一定局限性, 因此建立基于卷积神经网络图像特征提取的 SVM 模型, 利用 CNN 网络对图像数据进行特征提取, 再利用支持向量机对其进行分类。
- 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。因此, 我们首先对问题一中所建立的前两个模型进行学习, 并对其进行评估分析, 主要从分类报告分析。之后将已训练好的 CNN 模型用于图像特征的提取, 再次基础上再训练 SVM 模型, 优化参数, 并对其进行评估分析。

本文用于评估模型效果的主要方法有：时间、准确率、损失值、分类报告、混淆矩阵、ROC/AUC 曲线、精确率与召回率曲线、交叉验证。最后，对模型进行整体评价，选择出最优模型。

- **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。因此，我们需要读取问题一、二中所建立及训练好的模型，对未知数据集进行预测，注意分类标识的顶柜，同时也需注意避免文件名与被预测值错位，并将其文件名与对应的类别保存好。

三、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。

四、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。

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。

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 2 所示，图中所标注的为该图像的原始文件名。

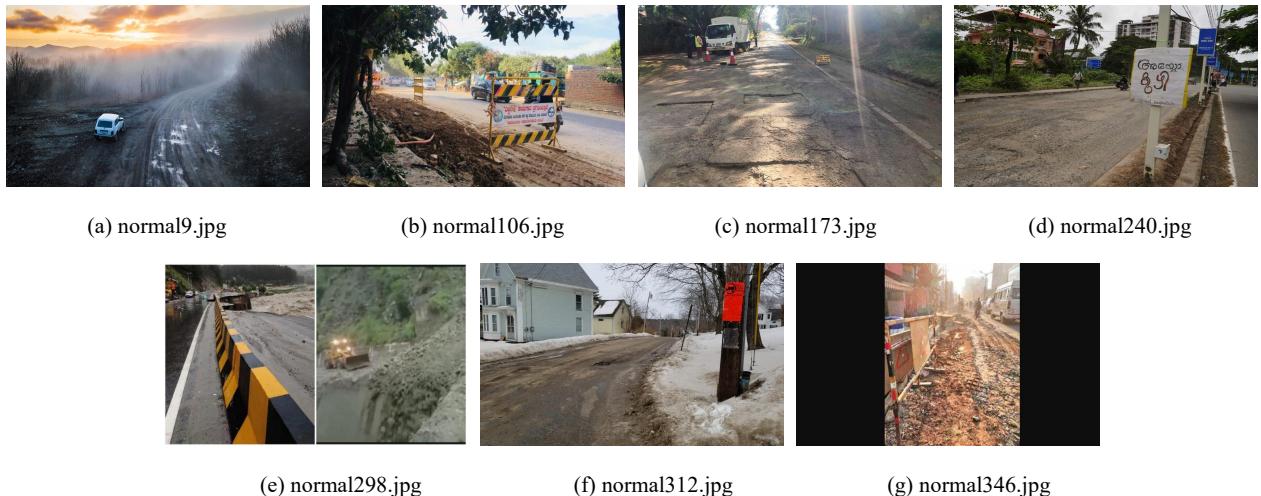


图 2 标注异常图片

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

5.1.2 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-C.1, Data Preprocessing [数据预处理]。

这里，我们可以大致观测数据集图像，如图 3 所示。

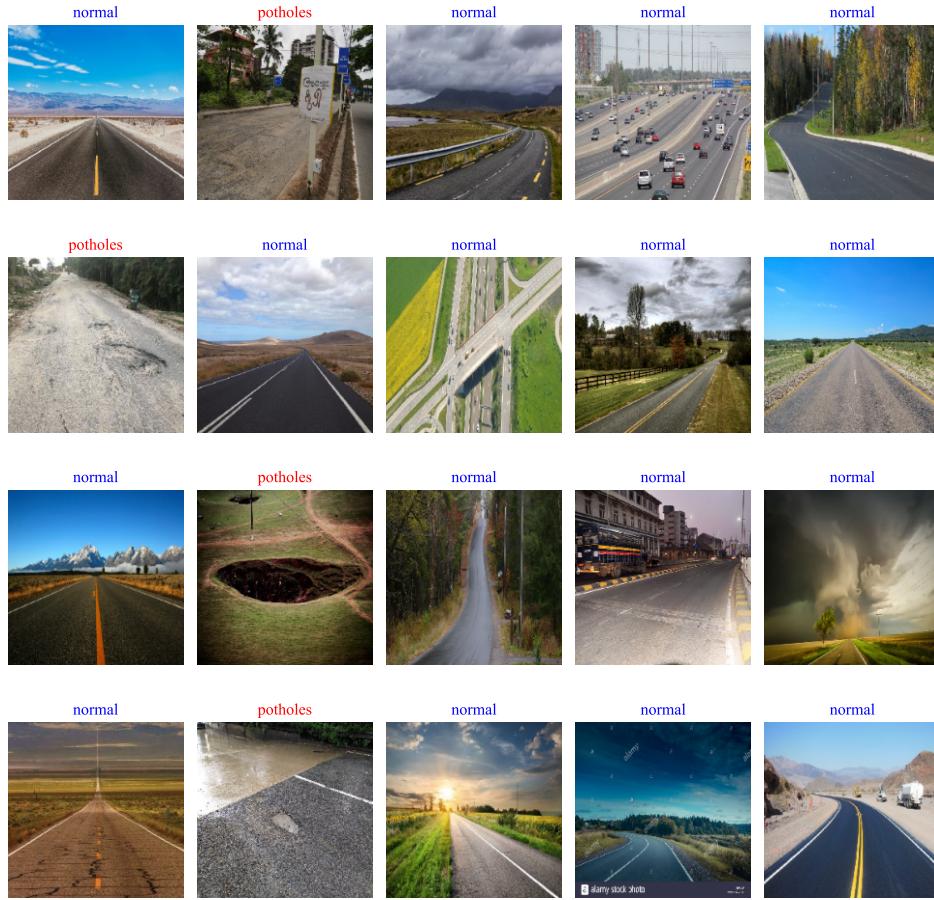


图 3 图像数据观测

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 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-C.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4及图5所示。

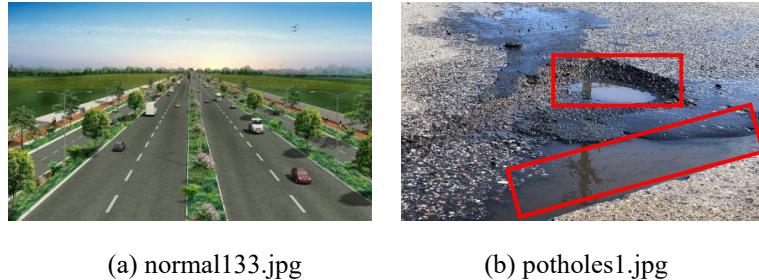


图4 原图对比

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。

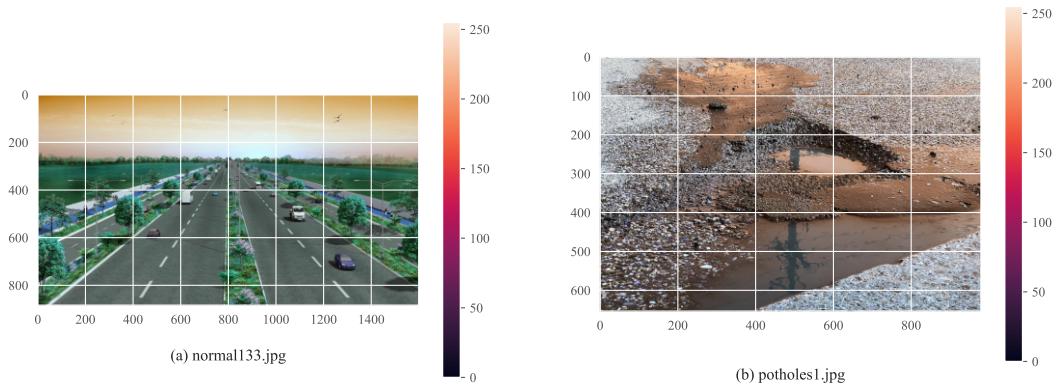


图5 热力图对比

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6及图7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。

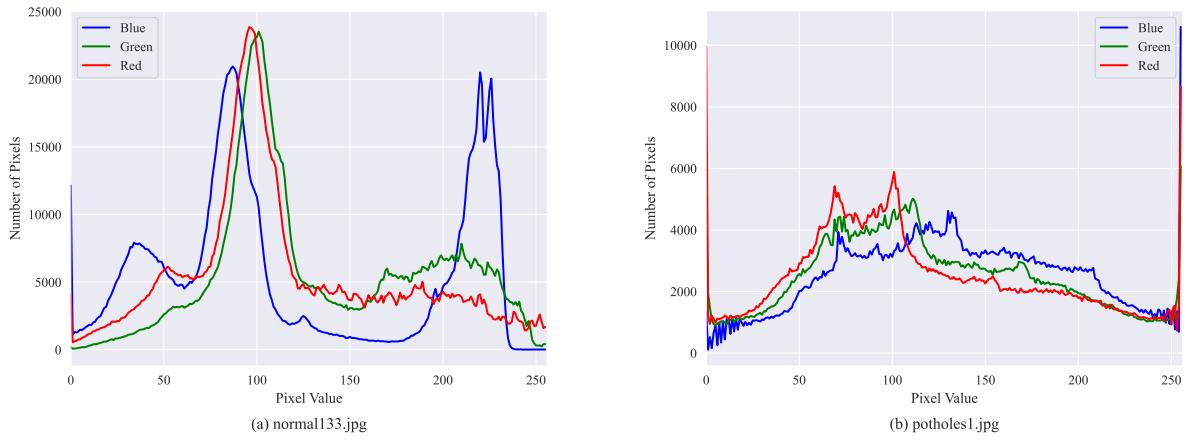


图 6 RGB 直方图对比

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。

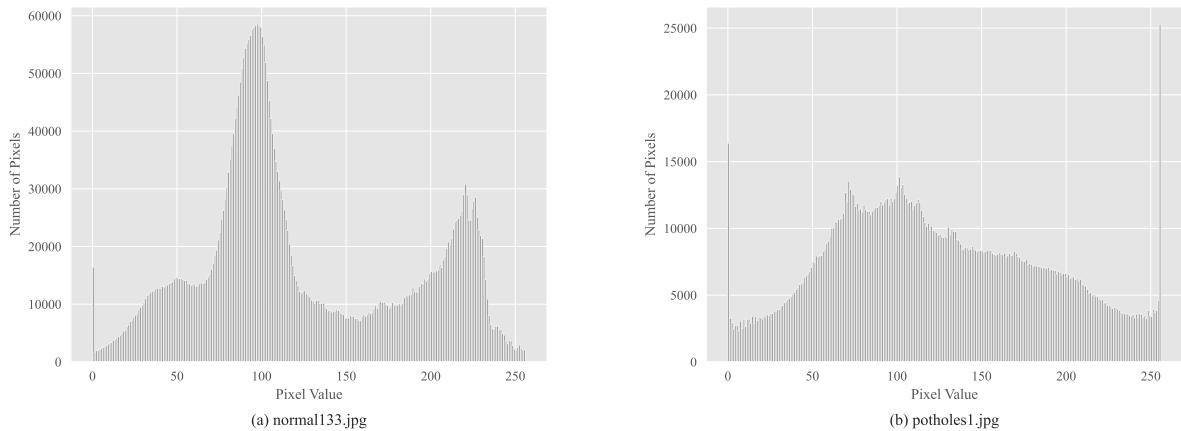


图 7 灰度直方图对比

5.2.3 边缘、轮廓检测

“normal133.jpg” 及 “potholes1.jpg” 边缘、轮廓检测对比，如图 8 及图 9 所示。

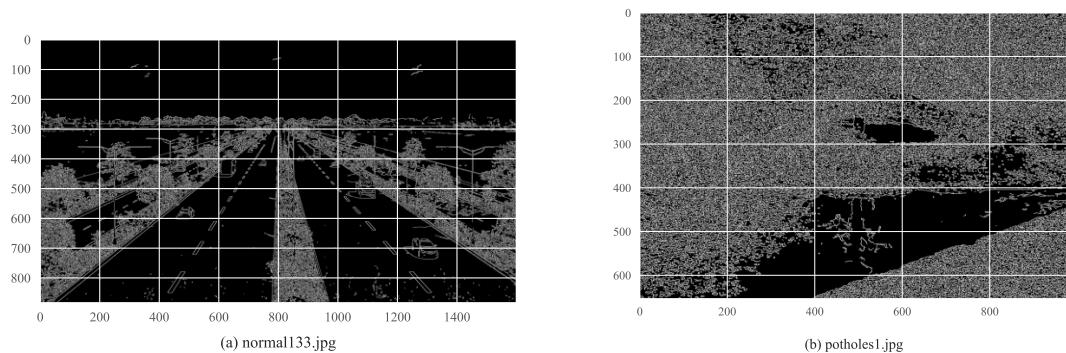


图 8 边缘检测

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。

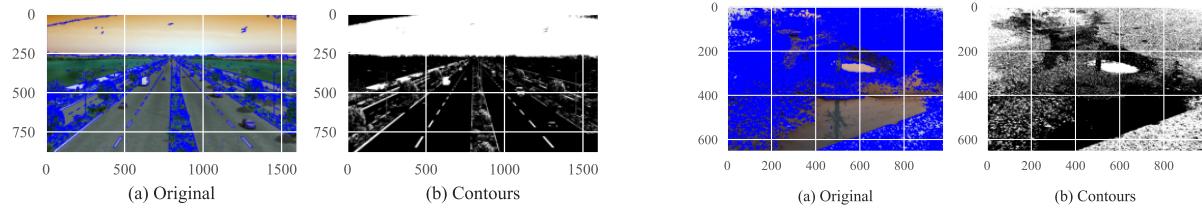


图 9 轮廓检测

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 10 所示。

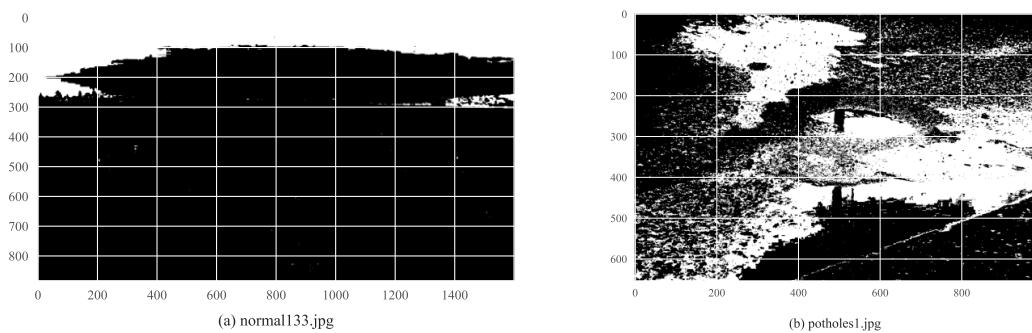


图 10 阈值分割

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拓充，拓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。

表 3 新旧数据集信息

数据集	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

这里，我们将随机选择 10 张图像进行展示，如图 11 所示。

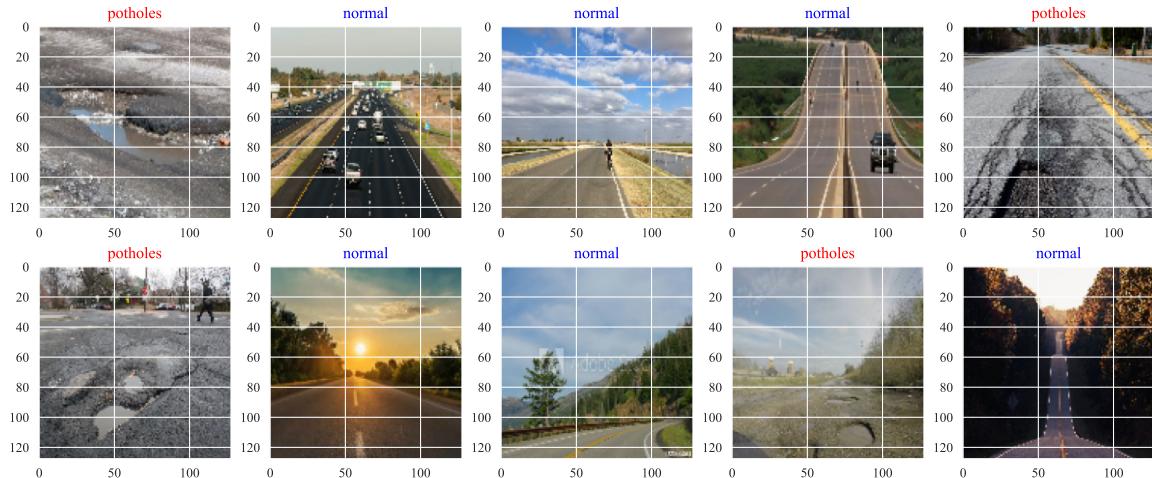


图 11 新旧数据集图像

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¹。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 (Bi-lateral 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：

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

- **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 12 所示。

¹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



图 12 原图、高斯滤波、双边滤波、顺时针旋转 30°

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 13 所示。



图 13 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

由于读取的数据本质上为特征向量，其维度较高，对于模型的训练时长、复杂度将会造成一定影响，因此这里我们首先使用**主成分分析 (Principal Component Analysis, PCA)**对数据集进行降维，其实质上是将 k 维特征映射到 n 维上，这 n 维是新的正交特征，即主成分。PCA 的工作就是从原始的空间中找一组相互正交的坐标轴。其中，第一个新坐标轴选取是原始数据中方差最大的方向，第二个新坐标轴选取是与第一个坐标轴正交的平面中使得方差最大的方向，第三个轴是与第一、二个轴正交的平面中方差最大的方向。依次类推，可以得到 k 个坐标轴。通过这种方式获得的新的坐标轴大部分方差都均含在上述的 n 个坐标轴中，后面的坐标轴所含的方差近似于 0。因此，我们可以忽略之后的坐标轴，只保留前面 k 个含有绝大部分方差的坐标轴。这相当于只保留包含绝大部分方差的维度特征，而忽略包含方差几乎为 0 的特征维度，从而对数据特征的降维处理。我们通过一组数据的样本均值和样本方差计算出两个样本之间的协方差，根据所求的协方差矩阵计算其特征值与特征向量，再将特征向量单位化得到基向量变换矩阵，最后利用基向量变换矩阵与原始矩阵做内积，得到降

维后的矩阵。对数据集计算协方差矩阵，公式如下：

$$\mathbf{C} = \begin{bmatrix} \text{Cov}(X_1, X_2) & \text{Cov}(X_1, X_2) & \cdots & \text{Cov}(X_1, X_n) \\ \text{Cov}(X_2, X_1) & \text{Cov}(X_2, X_2) & \cdots & \text{Cov}(X_2, X_n) \\ \vdots & \vdots & \ddots & \vdots \\ \text{Cov}(X_n, X_1) & \text{Cov}(X_n, X_2) & \cdots & \text{Cov}(X_n, X_n) \end{bmatrix} \quad (3)$$

其中，每一个元素定义如下^[2]

$$\text{Cov} = (X, Y) = E \{ [X - E(X)] [Y - E(Y)] \} = E(XY) - E(X)E(Y) \quad (4)$$

其中 $E(X)$ 为该列数据的均值。

之后，计算上述协方差矩阵的特征值 λ_n 及其对应的特征向量 \mathbf{u}_{nj} ，这里我们不妨令 $\lambda_1 \geq \lambda_2 \geq \dots \lambda_n \geq 0$ ，由特征向量组成 n 和新的指标变量

$$\left\{ \begin{array}{l} y_1 = \mathbf{u}_{11}x_1 + \mathbf{u}_{21}x_2 + \cdots + \mathbf{u}_{n1}x_n \\ y_2 = \mathbf{u}_{12}x_1 + \mathbf{u}_{22}x_2 + \cdots + \mathbf{u}_{n2}x_n \\ \vdots \\ y_3 = \mathbf{u}_{1n}x_1 + \mathbf{u}_{2n}x_2 + \cdots + \mathbf{u}_{nn}x_n \end{array} \right. \quad (5)$$

其中 y_n 是第 n 个主成分。

在 PCA 的基础上，我们建立**支持向量机（Support Vector Machine, SVM）**二分类模型。其是建立在结构风险最小原理及 Vapnik-Chervonenkis 理论基础之上的^[3]，以有限的数据信息，在数据样本中找出合适区分类别的决策分界面，与合适的边界分界面。实现分类误差最小的问题实质即求解下列方程^[4]：

$$\begin{aligned} \min \quad & L(\omega) = \frac{1}{2}\omega + C \sum_{i=1}^n \xi \\ \text{s.t.} \quad & y_i (\omega^T \varphi(x_i) + b) \geq 1 - \xi \end{aligned} \quad (6)$$

其中， $\xi > 0$ 为松弛变量， $\varphi(x_i)$ 为核函数， C 为惩罚系数。

此外，可针对不同的数据集的特征选择不同的**核函数（Kernel Function）**，从而达到较优秀的效果。对于该数据集，我们选择**高斯核函数（Radial Basis Function, RBF）**，其定义公式如下：

$$K(x_i, x_j) = \exp \left(-\frac{\|x_i - x_j\|^2}{2\sigma^2} \right) = \exp(-\gamma \|x_i - x_j\|^2) \quad (7)$$

同时，为了将模型效果优化，我们将对 SVM 模型的超参数进行调优。这里我们选用**网格调优（Grid Search）**，对惩罚系数 C 进行调优，最终选定的 $C = 1.90$ 。具体实现代码见附录-C.4 PCA-SVM [PCA 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

卷积神经网络是一类包含卷积计算且具有深度结构的前馈神经网络。它主要由输入层、卷积层、激活层、池化层和全连接层构成^[5]。其具体框架如图 14 所示。

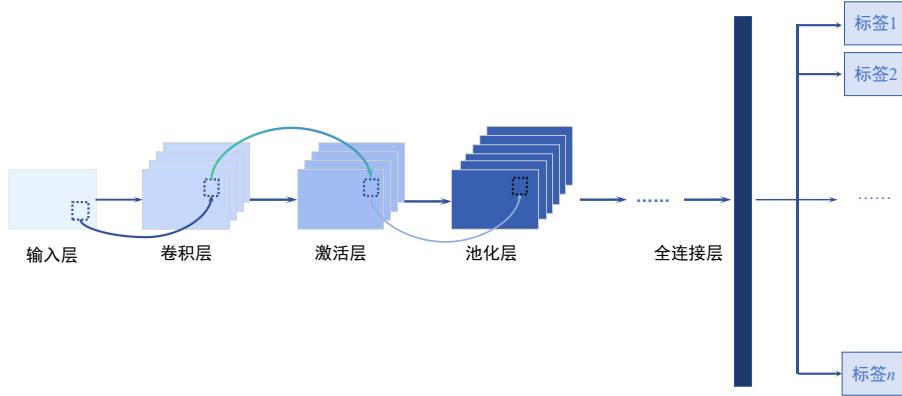


图 14 CNN 原理图

其首先于输入层对数据进行去均值处理，而后在卷积层进行特征提取，将特征通过激活函数映射到数学空间形成特征图并将输出结果于激活层进行非线性映射^[6]，接着在池化层中将特征降维压缩，降低复杂程度。池化的常见方法有平均池化和最大池化两种，最大池化是将输入图像分割为多个子区域，然后从每个子区域中选择最大像素值作为子区域的输出值，以此对输入图像的尺寸和特征进行采样和减少，以减少计算量。平均池化是将输入图像分割为多个子区域后计算每个子区域所有像素值的平均值，将平均值作为子图像的输出值，也可以减少计算量^[10]。最后传递至全连接层，将全连接层各节点与上一层所有的节点结合起来。平均池化计算公式及最大池化计算公式为：

$$p_l^{(i,j)} = \frac{1}{S} \sum_{x=id}^{id+d} \sum_{y=jd}^{jd+d} a_l^{(x,y)} \quad (8)$$

最大池化计算公式为：

$$p_l^{(i,j)} = \max_{(id, jd) \leqslant (x, y) \leqslant (id+d, jd+d)} a_l^{(x,y)} \quad (9)$$

其中， $p_l^{(i,j)}$ 为第一层中坐标为 $p_l^{(i,j)}$ 的特征值， S 为池化区域的面积， $a_l^{(x,y)}$ 为输入特征途中坐标为 (x, y) 的特征值， d 为步长。

$$p_l^{(i,j)} = \frac{1}{S} \sum_{x=id}^{id+d} \sum_{y=jd}^{jd+d} a_l^{(x,y)} \quad (10)$$

根据正常与坑洼道路的比较分析、图像特征提取部分的分析，对于该分类问题及数据集的特征，我们对 CNN 卷积神经网络的结构进行架构，具体如下：

- **卷积层：** 输入形状为 128×128 像素，3 个通道（RGB 图像），32 个卷积核，卷积核大小为 5×5 ，激活函数为 ReLU
- **最大池化层：** 池化窗口大小为 2×2

- **卷积层**: 输入形状为 64 个通道, 卷积核大小为 3×3 , 激活函数为 ReLU
- **最大池化层**: 池化窗口大小为 2×2
- **卷积层**: 输入形状为 128 个通道, 卷积核大小为 3×3 , 激活函数为 ReLU
- **最大池化层**: 池化窗口大小为 2×2
- **卷积层**: 输入形状为 128 个通道, 卷积核大小为 3×3 , 激活函数为 ReLU
- **最大池化层**: 池化窗口大小为 2×2
- **展平层**: 将卷积层的输出展平为一维数组
- **随机丢弃层**: 丢弃比例为 40 %
- **全连接层**: 输入为展平后的数据, 128 个神经元, 激活函数为 ReLU
- **全连接层**: 输入为 128 个神经元, 输出为 2 个类别 (类别标签为 0 和 1), 激活函数为 softmax

为更好地展现 CNN 模型的架构, 我们将其可视化, 如图 15 所示。

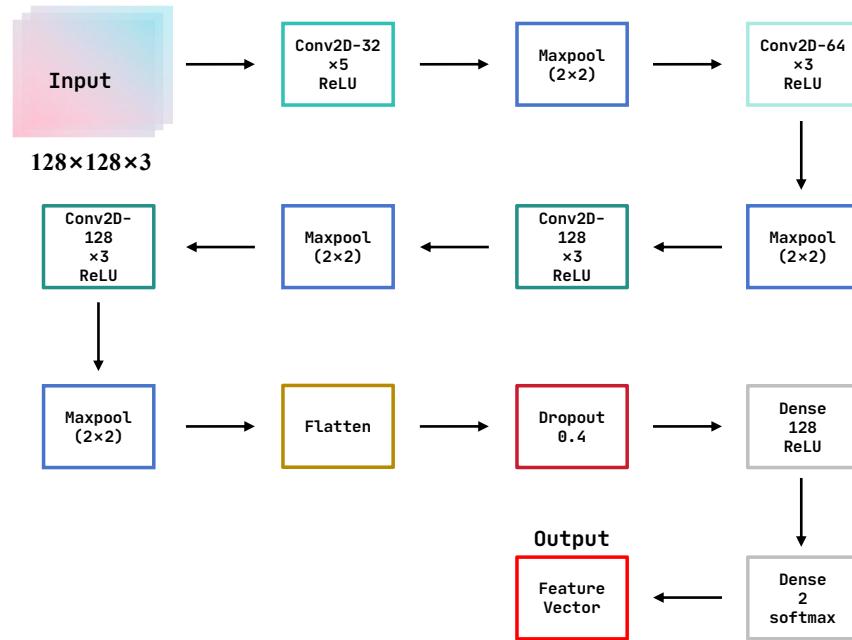


图 15 CNN 模型架构

5.4.3 CNN-SVM 的分类模型

传统的 CNN 模型由输入层、卷积层、激活层、池化层、全连接层构成，通过 softmax 进行输出。然而，由于 softmax 在针对非线性且稀疏的数据矩阵时，其效果不如传统机器学习算法^[7]。针对这一现象及避免单一模型的局限性，我们建立基于 CNN 与 SVM 组合的 CNN-SVM 模型，该模型利用基于监督学习的 SVM 模型实现分类，CNN 模型实现特征提取。除此之外，此模型相比于传统 CNN 模型还增加了注意力机制^[8]，此机制可理解为将更多的权重赋值在对图像识别有影响的特征上，同时减少对图像识别意义特征较小的权重。对于该模型：其一，将传统的 CNN 模型的 softmax 归一化函数更换为 SVM 模型，在一定程度上能够提升模型的泛化能力^[9]；其二，该模型还增加了注意力机制，能够更好地提取图像信息的特征，从而提升模型精度。

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：

- **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：

$$\text{Accuracy} = \frac{N_{\text{TruePredict}}}{N_{\text{Predict}}} \quad (11)$$

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：

$$\text{Loss} = - \sum_{i=1}^{\text{output size}} y_i \cdot \log(\hat{y}_i) \quad (12)$$

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

$$\text{Precision} = \frac{TP}{TP + FP} \quad (13)$$

– 召回率

$$\text{Recall} = \frac{TP}{TP + FN} \quad (14)$$

– F1 分数值

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \quad (15)$$

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(15) 式进行一定变换，可以得到：

$$F1 = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} \quad (16)$$

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：

$$\text{Sensitivity} = TPR = \frac{TP}{TP + FN} \quad (17)$$

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

$$\text{Specificity} = TNR = \frac{TN}{TN + FP} \quad (18)$$

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

$$FPR = 1 - \text{Specificity} = \frac{FP}{FP + TN} \quad (19)$$

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

$$FNR = 1 - \text{Sensitivity} = \frac{FN}{FN + TP} \quad (20)$$

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]。

5.5.1 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 $50\text{ s} + 281\text{ ms}$ ，在测试集适应耗时 $2\text{ s} + 968\text{ ms}$ ；SVM 模型网格调优耗时 $9\text{ s} + 948\text{ ms}$ ，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

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

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质。此外，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。

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 16 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

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

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 16 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 17 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
marco 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 18 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。

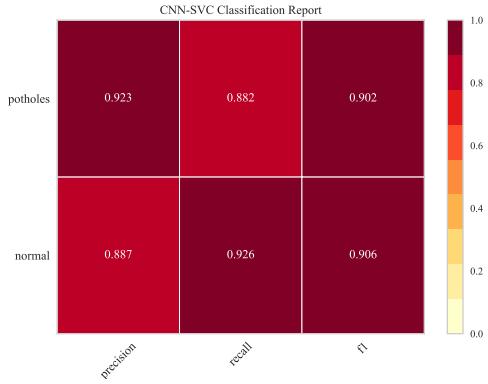


图 17 CNN-SVM 分类报告图

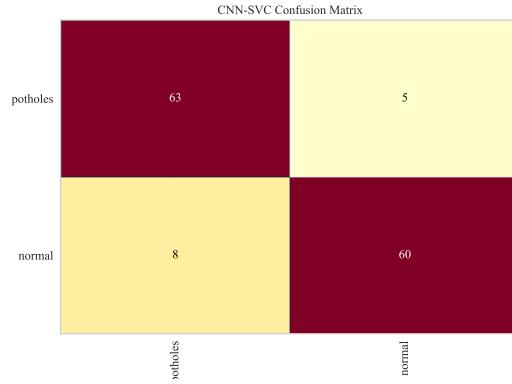


图 18 CNN-SVM 混淆矩阵热力图

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 19 与图 20 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。

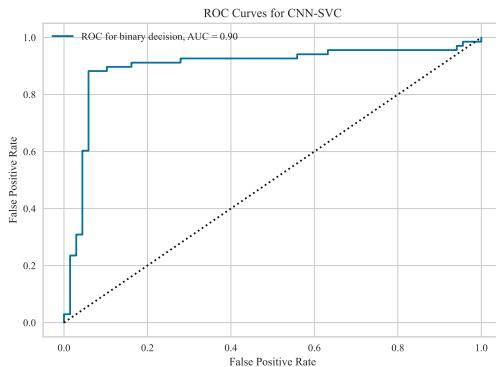


图 19 CNN-SVM ROC 曲线

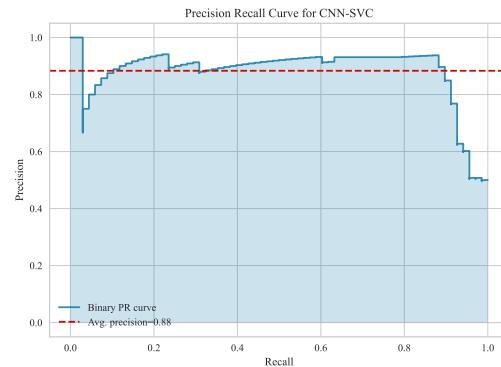


图 20 CNN-SVM 精确率-召回率曲线

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%]。

5.5.2 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 17~图 20, 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。

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

5.6 未知数据集的预测

对于该问题，我们按以下步骤进行处理：

- **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. 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。

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，对工程建设地区地形进行判别，准确且快速的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 文件	包括所有解决问题的源程序运行结果
ipynb 文件	包括所有解决问题的源程序源代码
py 文件	包括所有解决问题的源程序输出 python 文件
models 文件	已训练好的模型文件
test_result.csv	已训练模型识别测试集坑洼图像结果

[B] 使用的软件、环境

B.1 使用的软件及版本

- TeX Live 2022
- Visual Studio Code 1.83.1
- 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)

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 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.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(pothelesImg, 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) 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(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) 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(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(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')
```

C.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('*')] [: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
28 # 数据，而其余80%用于训练
29 dataImageDataGenerator = ImageDataGenerator(rescale = 1/255., validation_split = 0.2)
30 dataTrain = dataImageDataGenerator.flow_from_directory(dataAdd, target_size = (224,
31                                         224), batch_size = 32, subset = 'training', class_mode = 'binary')
32 dataVal = dataImageDataGenerator.flow_from_directory(dataAdd, target_size = (224, 224),
33                                         batch_size = 32, subset = 'validation', class_mode = 'binary')
34
35 # 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()

```

C.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
64         (inputImg, test_index, axis=0), np.delete(inputLabel, test_index, axis=0), np.
65         take(inputLabel, test_index, axis=0)
66     # 返回图像和标签的训练集和测试集
67     return train_x, test_x, train_y, test_y, test_index
68
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:

```

```

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
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()

```

C.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('DATA\\'):
30     # 统计DATA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))
140
141 # In[13]:
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\\CNNData.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')

```

C.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\\CNNDATA.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()

```

C.7 Predict [预测]

```

1 #!/usr/bin/env python
2 # coding: utf-8
3
4 # In[1]:
5
6
7 import os
8 import cv2
9 import pickle
10 import numpy as np
11 import pandas as pd
12 from keras.models import load_model
13
14 # In[2]:
15
16
17 CNNModel = load_model('Models\\CNN.h5')
18 with open('Models\\CNN-SVM.pkl', 'rb') as f:
19     model = pickle.load(f)
20
21 # In[3]:
22

```

```

23
24 imagePath = []
25 for dirname, _, filenames in os.walk('DATAPre\\'):
26     print(dirname, len(filenames))
27     for filename in filenames:
28         path = os.path.join(dirname, filename)
29         imagePath.append(path)
30
31 len(imagePath)
32
33 # In[4]:
34
35
36 IMG_SIZE = 128
37 X = []
38 y = []
39 imgName = []
40 for image in imagePath:
41     try:
42         img = cv2.imread(image, cv2.IMREAD_COLOR)
43         img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
44         X.append(np.array(img))
45         imgName.append(image.split('\\\\')[1])
46     except:
47         pass
48
49 # In[5]:
50
51
52 X = np.array(X)
53 X_train_features = CNNModel.predict(X)
54
55 # In[6]:
56
57
58 y_pred = model.predict(X_train_features)
59 df = pd.DataFrame({'fnames': imgName, 'label': y_pred})
60 df['label'] = df['label'].astype(int)
61 df
62
63 # In[7]:
64
65
66 df.to_csv('test_result.csv', index=False)

```

C.8 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\\potholes\\potholes1.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)
```
