[redacted]

毕业设计说明书

毕业设计题目：病菌图像自动分析与识别处理的系统设计

学院（部） [redacted]

专　　业 [redacted]

学　　号 [redacted]

学 生 姓 名 [redacted]

指 导 教 师 [redacted]

完 成 日 期 2023 年 05 月 28 日

[redacted]

摘 要

病菌是能导致人畜植物生病的微生物。微生物学要检测和识别病菌，这能让我们了解病菌的特征、数量、形状、变化以及分布，为防治疾病提供依据。目前常规的鉴别方法主要用显微镜培养、染色、观察等，此类方法既费时费力，又容易受人为因素干扰，结果不可靠。然而，随着计算机视觉技术的进步，使得用深度学习进行病菌图像的自动分析和识别正在成为一种新的方案。深度学习方法能够利用大量的已标注的图像或文本数据来训练一个高效的神经网络模型，从而快速地完成病菌图像的识别处理。

因此，本文提出了一种用 YOLOv8 模型分析和识别病菌图像的系统。系统输入是病菌图像，输出是标注了病菌种类和置信度的图像。系统有两个流程：检测识别、展示统计。第一个流程用 YOLOv8 模型检测和分类病菌，得到病菌的边界框、种类和置信度，并用 sqlite 数据库存储识别信息，避免重复检测。第二个流程是用 matplotlib 库可视化检测结果，画出病菌检测框的图片，显示每种病菌的数量、平均置信度等，并把结果保存为 PNG 图片和 sqlite 数据库。系统可以加密识别结果。本系统旨在设计一个自动化的识别流程，以一定程度地缓解传统病菌识别领域的难点，痛点，并为病菌图像处理业务提供一种深度学习的可行的设计方案，向有关领域的后来者提供一个参考。

关键词：病菌图像；YOLOv8；自动检测；深度学习

Abstract

Pathogens are microorganisms that can cause diseases in humans, animals and plants. Microbiology aims to detect and identify pathogens, which can help us understand their characteristics, numbers, shapes, changes and distributions, and provide a basis for disease prevention and treatment. The current conventional identification methods mainly use microscope culture, staining, observation and so on. These methods are time-consuming, labor-intensive, and prone to human interference, resulting in unreliable results. However, with the advancement of computer vision technology, using deep learning to automatically analyze and identify pathogen images is becoming a new solution. Deep learning methods can use a large amount of labeled image or text data to train an efficient neural network model, and quickly complete the recognition and processing of pathogen images.

Therefore, a system that used YOLOv8 model to analyze and identify pathogen images was proposed by this paper. A pathogen image was the input of the system, and an image annotated with the pathogen type and confidence was the output. The system had two processes: detection and identification, display and statistics. The YOLOv8 model was used to detect and classify pathogens in the first process, and the bounding box, type and confidence of the pathogens were obtained. The identification information was stored by using sqlite database to avoid repeated detection. The second process was to visualize the detection results by using matplotlib library, draw pictures of pathogen detection boxes, display the number and average confidence of each pathogen, etc., and save the results as PNG images and sqlite databases. The identification results were able to be encrypted by the system. An automated identification process was designed by this system to alleviate some of the difficulties and pain points in the traditional pathogen identification field to some extent, and a feasible design scheme for pathogen image processing business using deep learning was provided by this system, and a reference for later comers in related fields was provided by this paper.

Keywords: pathogens image; YOLOv8; automatic detection; deep learning

目　录

第 1 章 绪 论

1.1 研究背景、目的和意义

生物医学技术与图像识别技术的不断进步，使得病菌图像的自动化的分析与识别处理日益成为一个重要的研究方向,而病菌的准确识别和分类有助于医生确定疾病的致病原因，并采取相应的治疗措施。对许多疾病来说，早期诊断至关重要。通过快速而准确的病菌识别，医生能够尽早给出治疗方案，降低治疗事故并提高患者的生存率。病菌图像的识别和分类结果，经过医疗人员的处理，还可以有效监控公共疫情事件。因此，知晓病原微生物的类型和分布对于预防措施和有效干预至关重要，在流行病学调查和疫情控制中具有重要意义。此外，病菌图像的识别和分类还能为为研究人员提供相关致病微生物信息，有助于深入了解其特点、变异和传播机制。用于研究疾病的发病机制和开发抗菌药物以及防控策略的制定的意义重大。然而，这一领域也面临着许多挑战，如低质量的图像、多噪声、低分辨率、多样的病菌形态和类别失衡等，非现代的图像处理方法对此无法轻松应对。

针对上述解决这些问题，近年来，基于深度学习的目标检测方法具有高效、准确和鲁棒的优点，因为它们可以直接从图像中定位和识别目标，而不需要复杂的预处理和特征提取。其中，目前最先进的物体检测方法之一是 YOLO 系列算法。它以端到端的方式通过单次前向传播实现目标检测任务，在拥有 YOLOv5 的基本优点上，还加入了许多新功能和改进点。YOLOv8 支持图像分类、目标检测和实例分割任务，亦可运行在各种硬件平台上。具有速度快、准确率高、具有较强的泛化能力等特点。因此本次设计中使用的 YOLOv8，可以有效提升本课题设计的模型的表达和泛化性能。此外，还采用了一些数据增强、训练策略和推理技术来增强模型优化的有效性。

本文的目的是探讨 YOLOv8 在病原体图像的自动分析和识别中的部署与工作情况。以大肠杆菌以及结核分枝杆菌为例，这两种病原体都是常见的致病因子，对公共健康构成了严重的危害。因此，本课题利用 YOLOv8 算法，针对大肠杆菌和结核分枝杆菌图像进行了检测与自动化的统计，并对识别结果进行了可视化展示。希望本课题能够展示 YOLOv8 在病原体图像识别处理中的优秀性能和广泛的应用潜力，为相关的发展研究提供一些借鉴与参考。

1.2 国内外研究现状

YOLOv8 算法的核心是对人工特征工程或先验知识的零需求，从大量混合数据中

学习非线性的复杂特征的深度学习，它也因此得到了细菌图像处理和识别中的广泛应用，并能够表现出超越传统方法的性能。而国内的研究者，已有将深度学习的方法应用于细菌图像处理领域，例如：

张新峰等回顾了模式识别在图像处理中的应用，特别是对细菌图像分析，他们介绍了一些常用的细菌图像分割、特征提取、分类和识别方法，也讨论了一些细菌图像分析使用模式识别面临的挑战和未来方向[1]。

黄煜研究了基于广义结构元的动物病菌图像分形和识别方法，他使用改进的盒计数法计算了细菌图像的分形维数，还使用基于广义结构元的形态学滤波器增强了细菌图像的对比度和边缘，并应用支持向量机分类器根据细菌的分形特征识别不同类型的细菌[2]。

刘友江介绍了一种细菌显微图像自动识别技术，是使用自适应阈值法、形态学操作、水平集算法和区域生长算法将细菌图像分割为单个细胞区域,并从每个细胞区域中提取几何特征、纹理特征和颜色特征，最后使用 k 最近邻分类器根据细胞特征识别不同种类的细菌[3] 。

邓佳丽等人提出了一种基于目标检测的医学影像分割算法，他们使用 Mask R-CNN 对 CT 影像中不同器官进行检测和分割，还使用卷积神经网络对分割后的器官根据外观特征进行了正常或异常类别的分类[4]，

在国外领域，也有一些研究是关于细菌图像识别预处理的，也是使用深度学习，例如：

Hung 等人应用 Faster R-CNN 对疟原虫图像进行目标检测，他们使用迁移学习对预训练的 Faster R-CNN 模型在自己收集的血涂片图像数据集上进行了微调，使得在测试集上达到了 96%的平均精度和 95%的平均召回率[5]。

Yoshihara 等人开发了一个针对革兰氏染色涂片中弯曲杆菌和白血球吞噬活性的目标检测系统，此系统使用 YOLOv3 检测并定位图像中的弯曲杆菌和白血球，还使用卷积神经网络对检测到的对象进行多类分类如：阳性白血球、阴性白血球、仅弯曲杆菌[6]。

Ma 等人演示了一种利用人工智能和光学成像加速食品中细菌检测的方法，他们使用基于智能全息显微镜捕捉琼脂板上生长中的细菌菌落的相干图像，也是使用了深度神经网络分析这些时间序列全息图，以快速检测细菌生长和分类相应的物种[7]。

这些研究说明，在细菌图像分析与识别方面，深度学习优势巨大。但是，还有一些问题和限制，需要在今后的研究中改善，比如：现有的深度学习的图像训练集缺少规模大、质量高的数据集，用来训练和测试不同工作目标的深度学习模型。因此难以

把深度学习模型应用到不同种类的细菌、成像方法、噪音水平、照明条件等。目前需要更多研究资源投入来开发更先进的深度学习方法，以优化这些问题和限制，并探索更广的应用场景。

1.3 需求分析与设计内容

为了设计本课题的病菌图像自动分析与识别处理系统，需要深刻理解用户的需求，并通过实地考察医疗机构的生产环境作为参考。用户需求分析主要包括以下内容：

（1）快速响应的病菌图像处理能力：用户需要系统能够快速准确地对病菌图像进行分析和识别，找出并计算病菌的位置与数量，并将结果可视化。

（2）可靠的识别准度：用户期望系统有较高的识别性能和准确度，能够有效地区别不同类型的病菌，并准确地识别病变区域，以便后续进行医学诊断和治疗。

（3）简洁干净的界面和流畅的操作体验：用户希望设计的系统用户界面是操作方便易学，能够面向大众的，能偶降低人工操作的复杂性的。

（4）可扩展性和自我学习：用户需要的系统要具有良好的可扩展性和处理多目标的适应性，能够自动处理不同规模、类型的病菌图像

（5）数据加密服务：用户可能需要在某些特定场景下对识别结果进行基本的加密处理，来保护劳动成果和数据安全。

通过细致的需求分析，与用户密切合作，进行需求的持续调研并继续收集用户反馈，通过这样的方法，可以确定课题设计的工作方向，并分析行业的总体趋势，紧跟市场需求，最大程度地满足用户需求全性。本课题通过上一步的进展，在系统设计中尝试添加以下内容：

（1）实时目标检测方面，采纳了 YOLOv8 这一种将检测任务简化为一次前向传播的最新的识别算法，，提升了业务工作效率。

（2）利用 SQLite 这一嵌入式数据库引擎，即免去配置步骤，降低磁盘 I/O 消耗，加快读写速度。

（3）借助 PyQt5 制作了图形界面，用户友好，点击按钮就能激活相应功能。

（4）程序代码可以更换识别模型，医疗机构提供新的图像数据集给开发者，可以通过在 Colab 上在线训练来获取最新模型，并传输给用户一键更换，即可扩大病菌识别范围，并加快识别速度。

（5）程序使用 Fernet 的加密方法，应用 AES 对称加密以及 HMAC 消息认证算法，利用同一个密钥进行加密和解密，并保证消息的完整性和身份验证。

1.4 本章小结

本课题综合考虑了病菌图像自动分析与识别处理系统研究背景是基于对疾病检测和医疗领域的需求，旨在提供高效准确的自动化解决方案。在国内外研究现状中，该领域已成为研究热点，但仍面临挑战。在本课题系统的设计上，预先确定了需求分析，制定对应设计内容，针对用户的需求核对系统要求和功能的开发。在考虑到课题的综合影响时，从经济性分析得出该系统能提高工作效率、降低成本的优点，而在社会影响方面，本课题的设计有助于提高疾病早期检测和治疗效果，推动医疗技术发展，具有一定研究价值和应用潜力。

第 2 章 相关技术简介

2.1 Python 简介

最出名的解释型脚本编程语言之一就是 Python，它是开源的，有这些优势：语法接近自然语言，简单明了，不用担心语法细节，可以专注于解决问题；跨平台的语言，Python 可以在不同的操作系统上直接运行 Python 代码，不需要编译或者修改；支持面向对象的编程范式，Python 可以使用类和对象来组织代码，实现数据和功能的封装、继承和多态；与其他语言（如 C/C++）进行混合编程，Python 可以利用其它语言高效的性能来优化关键代码的同时，拥有丰富易上手的第三方库和框架，轻松实现各种功能，如数据分析、人工智能、网络编程、用户图形界面等；开源免费的语言，Python 可以自由地使用、修改和发布 Python 代码，也不用担心版权问题，其开源性也促进了社区的发展，吸引了很多优秀的开发者和机构参与到 Python 的改进和维护中；内置数据类型强大，Python 拥有列表、元组、字典等易于处理各种数据结构，同时还支持如过程式、函数式、面向对象等的多种编程范式，可根据不同的需求来选择。因此，Python 是一种功能强大、简单易用、高可扩展性、开源免费的编程语言，在各个领域都有广泛的应用，是值得学习和使用的一种语言。

2.2 PyQt5 简介

PyQt5 是一种功能强大的 Python 绑定，它基于 Qt 的 GUI 控件集。不仅可以在 Linux、Windows 和 Mac OS 系统上跨平台运行，还采用了信号槽机制进行通信。此外，PyQt5 完全封装了 Qt 库，并提供了一整套多样化的窗口控件。成熟的集成开发环境可以加快开发人员的界面设计，并一键生成可用的 Python 代码。这种特性使得界面设计变得方便快捷，开发人员可以通过拖放控件、设置属性和连接信号槽来创建用户友好的界面。同时，自动生成的 Python 代码可直接用于开发和部署应用程序。

PyQt5 还具备方便调用 C/C++库的能力，从而提高性能和功能。Qt 作为一个强大的跨平台应用程序框架，提供了许多高级功能和工具，如图形渲染、数据库访问、网络通信等。作为 Qt 的 Python 绑定，PyQt5 可以直接调用这些功能和工具，充分利用 Qt 的优势来扩展和增强应用程序。因此，PyQt5 也适用于快速开发原型和小工具。

PyQt5 提供了丰富的窗口控件和便捷的 GUI 开发工具。开发人员可以快速创建具有基本功能的应用程序原型，以便进行用户测试和反馈。控件涵盖了各种常见的 GUI 元素，如按钮、文本框、复选框、下拉列表等。这些控件属性和方法丰富，可灵活定制

和交互。此外，PyQt5 还支持布局管理器，如垂直布局、水平布局和网格布局，便于开发人员对控件进行组织和排列，提高工作效率。

PyQt5 的实现原理如下：作者开发了一个工具，用于生成 Qt 和 Python 的绑定。通过解析 Qt 的头文件并生成对应的 Python 模块，实现了 Qt 库与 Python 的连接。sip 工具用于将 C++的头文件转换为 Python 模块，生成一系列的包装代码，实现 Python 与 Qt 库之间的接口。借助这些包装代码，PyQt5 可以在 Python 环境下使用 Qt 的功能，与 Qt 库进行交互，创造强大的 GUI 应用程序。

总结起来，PyQt5 作为强大的 GUI 开发工具，拥有丰富的窗口控件、强大的绘图功能、多线程支持，并与 Qt 生态系统无缝集成。它广泛的功能和工具，使得开发人员能够轻松创建出功能完善的应用程序。

2.3 SQLite 简介

本课题采用 SQLite 作为本地图像识别的数据库解决方案。它可以存储图像的 MD5、识别结果数据以及图片的创建和修改时间。

SQLite 是一种占用内存小的嵌入式关系型数据库管理系统（RDBMS），在各个领域都有广泛应用。它旨在提供简单、快速、可靠的本地数据库解决方案，并适用于嵌入式设备和应用程序。它的工作原理基于文件系统，将整个数据库存储在单个文件中，无需独立的数据库服务器进程。通常将该文件命名为数据库文件，使用.db 扩展名。

SQLite 使用 B 树这种自平衡的数据结构来组织和管理数据，能够高效地进行数据检索和插入操作。

在 SQLite 中，数据以表的形式组织，每个表由多个列组成，每个列定义了数据的类型。每一行代表一个记录，包含一组与列对应的值。SQLite 可接受的数据类型广泛，如整数、浮点数、文本和日期。数据库文件由多个页组成，每个页的大小通常为 4KB，每个页可包含数据行、索引和元数据。SQLite 使用页来存储表和索引数据，以及事务和回滚日志等辅助结构。同时，它采用了 ACID（原子性、一致性、隔离性和持久性）事务特性，确保数据的完整性和一致性。事务是一系列数据库操作的逻辑单元，要么全部执行成功，要么全部不执行。

SQLite 提供了 SQL（Structured Query Language）接口，用户可以使用 SQL 语句进行数据查询和操作。用户可以通过 SQL 语句创建表、插入数据、更新数据、删除数据，以及执行复杂的查询操作。大部分标准的 SQL 语法，包括 SELECT、INSERT、UPDATE、DELETE 等语句是在 SQLite 里获得支持的。借助这些优势，SQLite 在许多

应用领域都得到了推广与应用。

2.4 Fernet 加密技术简介

在本课题的数据安全设计中，为了加密识别结果，借鉴了黄峰、冯勇等人提出的一篇论文，该论文介绍了一种新颖的图像加密算法，基于二维混沌映射和图像分割思想。这个算法通过对图像进行拉伸和折叠处理，实现了图像的混乱。然后，利用 Logistic 混沌序列作为参数对混乱后的图像进行异或运算，以增强加密算法的安全性。这种算法具有密钥空间大、密钥敏感度高、扩散性能好、加密速度快等优点。相比于其他混沌映射算法，如 Baker 映射和 Cat 映射，它具有明显的优势[8]。

然而，由于在 Python 编程方面的熟练程度有限，本课题设计无法成功实现这个加密算法。遇到了一些困难，例如如何将图像分割为等腰三角形、如何将二维混沌映射转化为可逆的计算公式、如何生成 Logistic 混沌序列等等。尝试了一些方法，但都没有得到满意的结果。因此，为了完成本课题，采用了 Python 第三方包中的 Fernet 加密算法来对数据进行加密和解密。

Fernet 加密算法是一种对称加密算法，它使用 AES 算法和 HMAC 认证码对数据进行加密和解密。其原理如下：首先，随机生成一个 256 位长度的密钥，其中前 128 位用于 AES 加密，后 128 位用于 HMAC 认证。紧接着，对待加密的数据进行填充，以确保长度成为 16 的倍数，并采用 PKCS7 填充方式进行处理。接下来，采用 AES 算法和前 128 位密钥对填充后的数据进行加密，生成密文。此后，生成一个时间戳，用于表示加密的时间，并用于验证数据的有效期。然后，利用 HMAC 算法和后 128 位密钥对时间戳和密文进行认证，得到认证码。最后，将时间戳、密文和认证码拼接在一起，形成最终的加密数据。

解密过程是加密过程的逆过程：首先，将加密数据分割为时间戳、密文和认证码三个部分。接着，使用 HMAC 算法和后 128 位密钥对时间戳和密文进行认证，以验证其与认证码是否一致，若不一致，则抛出异常。随后，检查时间戳是否在有效期内，若超过有效期，则抛出异常。然后，利用 AES 算法和前 128 位密钥对解密密文，得到填充后的数据。最终，去除数据的填充部分，即可得知原始数据。

尽管 Fernet 加密算法不如混沌算法那么高级和复杂，但它的优点也不错，比如简单易用、安全可靠、无信息损失等。希望通过不断的学习，未来能够掌握更多编程知识和信息安全相关内容，让更加出色的数据加密功能的实现成为可能。

2.5 YOLOv8 目标检测简介

2.5.1 YOLOv8 的识别原理

YOLOv8 是一种基于卷积神经网络（CNN）的目标检测模型。它的核心思想是将输入图像划分为多个网格，对每个网格进行预测，输出目标的类别和位置。实现原理包含多个步骤。首先，可以通过图像分割得到所需的训练数据集[9]。然后，对数据集进行预处理，包括图像缩放、归一化和数据增强等操作。接下来，使用深层 CNN 进行特征提取，生成不同尺度的多个特征图。为了综合各个尺度的特征信息，采用多尺度特征融合（MFF）技术。这种技术利用上采样和下采样、相加或拼接等方式，得到更加丰富和细粒度的特征图。然后，采用预测头模块对每个特征图进行预测，输出每个网格的目标类别、置信度和边界框坐标。最后，通过后处理步骤对预测结果进行筛选和优化，例如非极大值抑制（NMS），来去除重叠和低置信度的边界框，从而获得最终的检测结果。

2.5.2 YOLOv8 的模型优势

YOLOv8 对目标检测进行了改进，引入了新的技术，如注意力机制、自适应卷积核和多尺度特征融合[10]，以提升性能和效率。

YOLOv8 的主要特点包括：支持多项任务，如目标检测、多目标跟踪、实例分割、姿态估计和图像分类。有多种模型大小可选，从 YOLOv8n 到 YOLOv8x，以满足不同的精度和速度需求。支持多种数据集，包括 VOC、COCO 等，还可以使用自定义数据集进行训练。适用于各种部署环境，如 Amazon Web Services (AWS)、Google Cloud (GCP)、Docker Image 等。此外，YOLOv8 可以生成多种导出格式，如 TFLite、ONNX、CoreML 和 TensorRT，便于模型的部署和集成。与 Roboflow、ClearML、Comet 和 Neural Magic 等多种集成工具兼容，进一步扩展了模型的功能和性能。

YOLOv8 的优势在于快速、准确和易用。它采用一种高效的卷积神经网络架构，在 CPU 或 GPU 上均可实现高速推理。通过多尺度特征融合、自适应锚框和注意力机制等技术，提高了检测和分割的精确率。此外，YOLOv8 的官方提供了简单的命令行界面和 Python API，使得训练、验证、预测和部署更加便捷。

YOLOv8 的一些缺点也是不可忽视的。首先，此算法涉及多种任务、模型、数据集和环境时，相对复杂，需要投入时间和精力进行掌握。此外，由于使用大量参数和计算量，需要较高的硬件配置和内存空间[11]。与 YOLOv5 相比，YOLOv8 的权重文件

更大，复杂度更高，需要更多存储空间和内存，并且训练时间更长。因此，对于快速迭代的项目、设备或平台，可能不太适用。

2.6 Colab 简介

Colab（Google Colaboratory）是谷歌提供的在线工作平台，开发者只需要一个浏览器，无需环境配置即可执行 Python 代码，并与他人分享和合作。Colab 的优势在于它是免费且便捷的，无需安装软件或配置环境，只需谷歌账号和网络连接。此外，Colab 提供谷歌云端服务器，包括 GPU 和 TPU，以提升计算速度和效率。它支持多种 Python 库，如 TensorFlow、PyTorch、Pandas 等，可用于数据科学和机器学习任务。Colab 具备灵活性，可以导入自定义数据集或使用云端硬盘、Github 等其他数据来源。此外，平台还提供文档、教程和示例等资源，以帮助用户快速上手和学习。

Colab 的优势在于使得 Python 数据分析和机器学习变得轻松，不论用户的硬件配置、操作系统或编程水平如何。通过省去安装、更新、调试等繁琐步骤，Colab 提高了工作效率，让用户能够直接在浏览器中编写和运行代码，从而节省时间和精力。此外，Colab 还推动了协作，允许用户与他人共享和编辑同一文档，交流和反馈可以无缝进行，从而提升工作效率。

Colab 亦有其不足之处。它设有使用限制，如每日仅可使用 12 小时的 GPU 或 TPU，文档保存期限为 90 天，每篇文档最多允许 100 名协作者等。由于 Colab 乃在线平台，故可能受网络连接、服务器负载以及系统更新等诸多因素影响，致使速度减慢、出现错误或中断等问题。此外，某些库如 PySpark、OpenCV 可能不受支持或兼容，需由用户自行安装或修改。总的来看，尽管 Colab 具有使用限制、不稳定性和不完善性，但对于内存占用不超过 12.7G 的一般模型训练而言，它是一个便利、实用且强大的在线工作平台，可于浏览器中执行各类数据科学和机器学习任务。

2.7 本章小结

本章简要介绍了几个本课题设计用到的工具或语言：首先是一种广泛应用的解释型脚本编程语言 Python，其次提到了用于创建用户友好界面的 Python 库 PyQt5，还有存放图片数据库用的 SQLite，另外，本课题设计用到的 Fernet 加密解密技术，它提供了数据加解密的功能。而本文的核心工作模块是由基于卷积神经网络的 YOLOv8 算法负责实现的。最后本章还提到了用于本课题设计用于训练模型的 Colab，是一种可提供免费算力在线工作平台。

第 3 章 病菌识别的系统设计

3.1 开发环境和框架

本系统面向生物，医疗机构，及其上下游的检测机构的 Windows 端用户，用户界面使用 PyQt5 编写，数据库使用 SQLite 以存放相关数据。整套程序在 Vscode 的 Python 运行环境下调试，数据的可视化调试用 Navicat 进行。此系统在 Windows10 的操作系统下进行开发，处理器为 AMD Ryzen 7 5800H，内存是 16GB DDR4 3200MHz(8GB + 8GB)。主要使用的开发工具如下表 3.1 所示

表 3.1 识别处理系统的开发使用的主要工具

工具	版本号	用途
SQLite	3.35.5	关系型数据库
PyQt5	5.15.9	现代化交互式图形用户界面开发
Navicat	12.0.16(64-bit)-Premium	数据库 GUI 管理工具
Vscode	1.78.2	轻量级的跨平台 Python IDE
Colab	5.15.107	集成 Google Drive 的 Jupyter 笔记本环境

3.2 功能的程序设计与实现

在本课题代码中，运行逻辑为先导入所需的库并预定义一些全局变量，包括文件路径、密钥长度和 YOLO 使用的模型等，并在默认的当前的工作路径下生成 keyfname，一段随机的加密解密用的密钥，并将其保存在一个以当前日期和时间命名的文本文档中。代码在点击“启动”按钮后，会与数据库建立链接，并遍历默认的工作文件夹下的所有 JPG，JPEG，PNG 的图片文件，或者用户可以选择新的工作路径文件夹；对遍历路径下的图片 MD5 比对数据库内的 MD5 值，若没有结果，比对失败，则对于已扫描的每个图片文件执行以下操作：首先，用 model()函数对图片识别并处理，使用 cv2 库的 imread 函数读取传入的图片文件并将其转换一下 RGB 格式，避免显色异常，并将检测结果保存在默认工作文件夹下的"BD_" + date_time + ".png"的文件中，再将这里的文件图象传递到 pixmap 上，在图形界面中展示；用户可以点击“加密图”的按钮对选择的文件夹路径下所有 PNG 图片进行加密，这时，加密后的图片则无法打开，解密时只需点击“解密图”按钮，选择正确的密钥文本即可；本设计的总体工作流程图如图 3.2 所示。

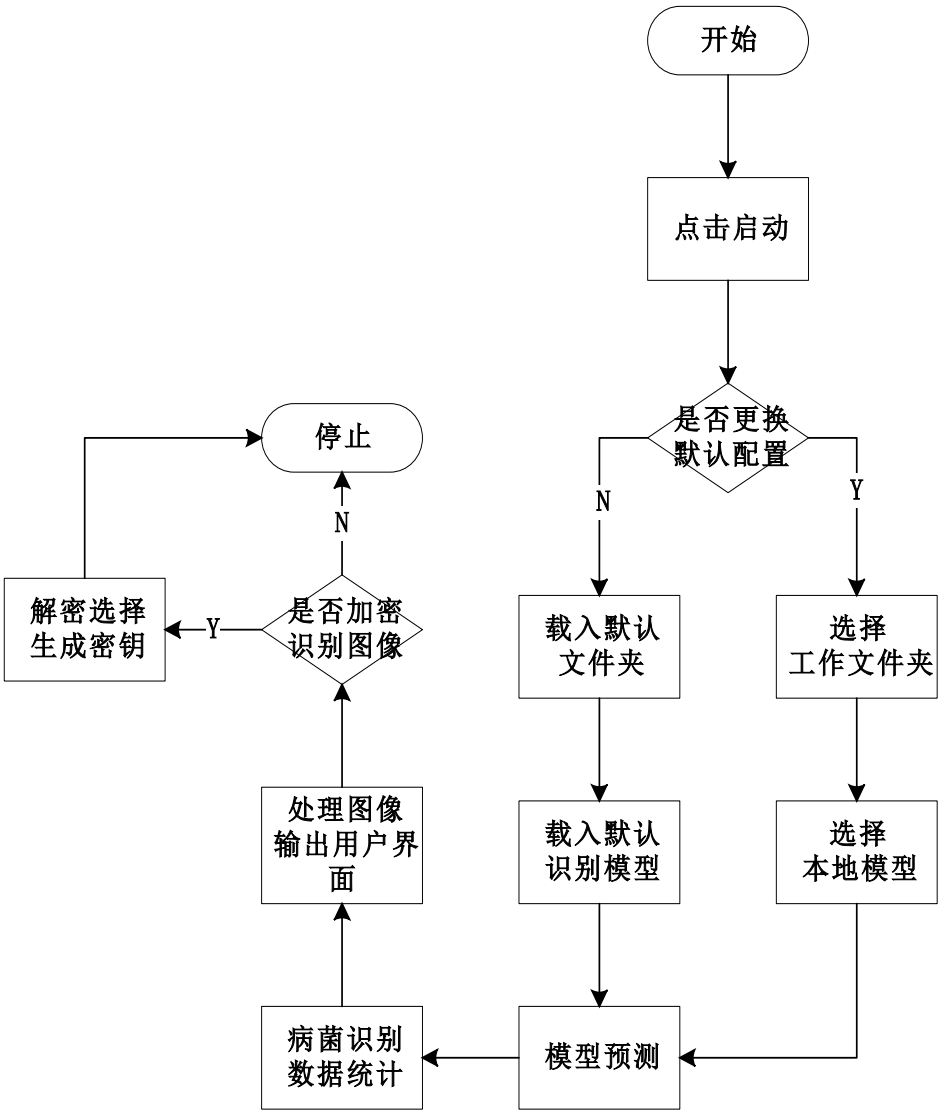


图 3.2 系统工作的流程图

为了避免浪费计算资源，根据课题设计的需求，针对工作件夹中已存在的病菌图片，实施遍历时不重复处理，因此采用了 SQLite 数据库，并设计了图片表，其中包括 ID、图片 MD5、识别种类以及检测数量等字段。图片表中的记录包含了自增主键 ID、创建时间和修改时间等信息。数据库表设计如图 3.3 所示。

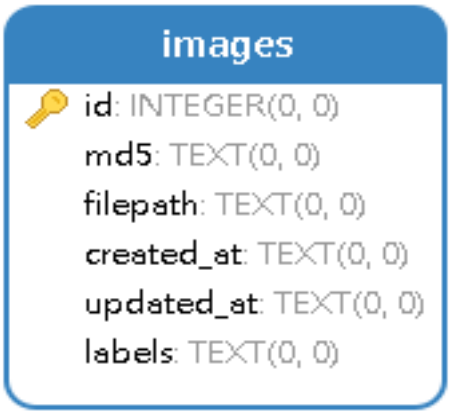


图 3.3 数据库的关系模型图

3.2.1 病菌图像识别模型的训练

在免费的 colab 上的 Jupyter 笔记本环境下，使用免费的 T4 GPU 加速模型的训练，150epoch 的训练只需要 1 小时左右即可完成。此部分代码可自动执行，无需修改；先下载网上搜索的大肠杆菌和结核分枝杆菌的图像数据集，在下载 YOLOv8 的运行库，再用 Linux 命令行代码修改 data.yaml 的种类名称和种类数量，修改第二个数据集的标注序号，再将两个数据集的图像文件合并，之后就可以训练多个病菌的识别模型了。只要在训练完毕后，下载 runs/detect/weights/下的权重文件 best.pt 即可。

图 3.4 Colab 训练识别模型的代码

3.2.2 用户界面的设计

为了实现用户界面，首先定义了一个 __init__ 方法，该方法用于初始化 MainWindow 类的实例，并设置窗口的标题、大小和图标等属性。此外，还创建了一个菜单栏，并添加了 6 个按钮：加密图片、解密图片、选择密钥文件、更换模型、更改工作路径和更改保存位置。针对每个菜单项，绑定了相应的方法；接下来，定义了一个名为 main 的函数，用于运行程序。在该函数中，创建了一个接受系统参数的 QApplication 对象，并将 sys.argv 参数传入其中。然后，创建了一个名为 MainWindow 对象，并调用 show 方法，以显示窗口。最后，调用 app.exec_ 方法，进入事件循环，从而使程序在运行后展示一个带有灰色网格布局的界面。

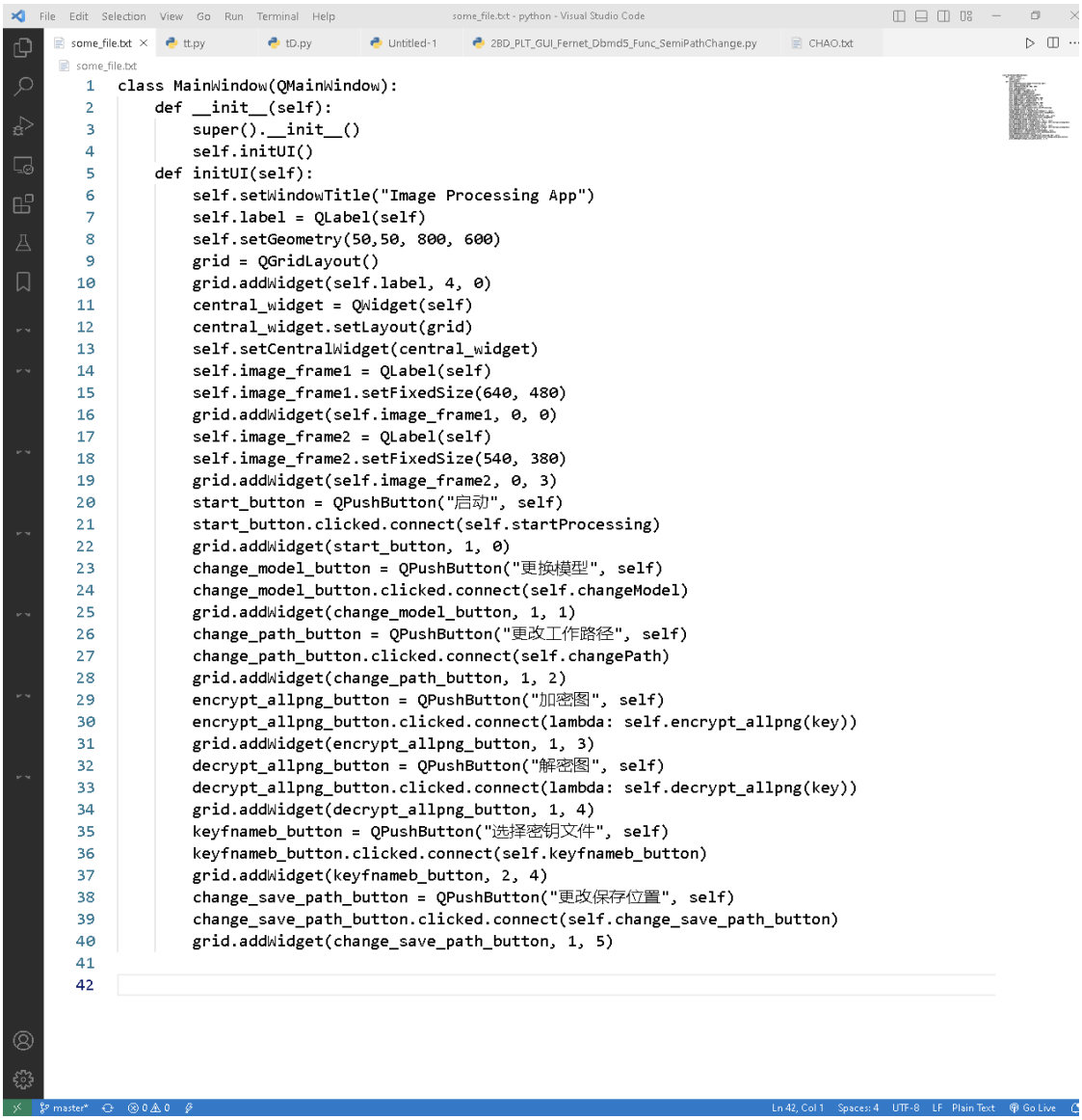


图 3.5 用户主界面代码示例

3.2.3 图像识别处理模块的设计

首先需要遍历所提供文件夹下的所有图片文件，包括 jpg、jpeg 和 png 格式。接着，在 image_data 数据库下创建一个名为 image 的表，其中包含图片的 MD5 值、图片 ID、处理信息、创建日期和修改日期等字段。然后，将选取的图片的 MD5 值与数据库中的 MD5 记录进行比对，如果没有对应记录，说明该图片未处理。接下来，使用 model.predict 方法处理输入的图片，并利用模型的 plot 方法绘制预测结果，包括置信度、边框和标签等，并返回一个 PIL 图像对象。获取预测结果中的边框信息，这是一个包含类别、置信度和坐标等属性的 Box 对象列表。为了获取病菌图像识别后的类别名称，初始化两个字典，分别用于累加每个类别的置信度之和和记录每个类别的数量。接下来，遍历边框列表，获取边框对应的整数类别编号和浮点数置信度。如果该类别在字典中不存在，则将其置信度初始化为 0，并分别累加该类别的置信度之和和数量。继续遍历字典中的类别编号，计算每个类别的平均置信度。

最后，将 PIL 图像对象转换为 OpenCV 图像对象 img，并将颜色通道顺序调整为 RGB。创建一个 matplotlib 图形对象 fig，然后在图形上展示 img 的图片，初始化一个字符串 label_text 来存储标签文本信息，并遍历类别名称和数量的字典。将类别名称、数量和平均置信度拼接在一起形成标签文本信息，并保留三位小数。为了显示中文字符，将 matplotlib 的字体参数设置为黑体。将标签文本信息作为标题文本，加上当前日期、时间和计数器值来构成保存文件名的一部分。在图形上添加标注，并设置标题文本的字体大小和粗细，同时对底下标注的文本进行颜色、字体大小、粗细和对齐方式的设置。为了保存识别的图片，将保存文件名以"BD_"开头，加上日期和时间信息，并以".png"结尾，将图形保存到指定的文件夹路径下，并使用保存文件名作为命名。接着打印提示信息，显示成功选择和保存的图片文件名。计算图片文件的 MD5 值，用于唯一标识图片内容的相同或不同性。然后执行一条 SQL 语句，向 image_data 数据库的 image 表中插入一条记录，包括 MD5 值、文件路径、创建时间、更新时间和标签信息等字段值，最后提交数据库更改。创建一个 QPixmap 对象，用于展示原始图片文件，并将其缩放到合适的大小，保持宽高比。将 QPixmap 对象设置为第一个图像框的内容，并更新图像框的显示，用于展示处理后的图片文件，同样将其缩放到合适的大小，保持宽高比。再将 QPixmap 对象设置为二个图像框的内容，操作相似，并进行图像框的显示更新。设置一个定时器，在延迟 2000 毫秒后执行一个空函数，用于用户界面的图片展示。最后用 plt.close 方法关闭 matplotlib 图形对象，释放资源。代码如下所示：

```
def process_images(self, image_files, folder_path):
    now = datetime.datetime.now()
    coun = 1
    conn = sqlite3.connect('image_data.db')
    cursor = conn.cursor()
    cursor.execute('''''
        CREATE TABLE IF NOT EXISTS images (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            md5 TEXT,
            filepath TEXT,
            created_at TEXT,
            updated_at TEXT,
            labels TEXT
        )
    ''')
    conn.commit()
    for file in image_files:
        filename = os.path.join(folder_path, file)
        filename = os.path.normpath(filename)
        cursor.execute(
            'SELECT md5 FROM images WHERE filepath = ?', (filename,))
        result = cursor.fetchone()
        if result is None:
            im1 = Image.open(filename)
            results = model.predict(source=im1)
            res_plotted = results[0].plot(
                line_width=5, conf=True, boxes=True, labels=False, pil=True)

            boxes = results[0].boxes
            names = model.names
            sums = {}
            counts = {}
            for box in boxes:
                cls = box.cls.item()
                conf = box.conf.item()
                if cls not in sums:
                    sums[cls] = 0
                    counts[cls] = 0
                sums[cls] += conf
                counts[cls] += 1
            for cls in sums:
                average = sums[cls] / counts[cls]
```

```
                img = cv2.cvtColor(res_plotted, cv2.COLOR_BGR2RGB)
                fig = plt.figure()
                plt.imshow(img)
                label_text = ''
                for namee, countt in counts.items():
                    label_text += f"{names[namee]}已识别: {countt} A_conf: {average:.3f} "
                plt.rcParams['font.sans-serif'] = ['SimHei']
                title_text = f"{label_text}"
                date_time = now.strftime("%Y-%m-%d___%H-%M-%S-") + str(coun)
                plt.title(title_text, fontsize=14, fontweight='bold')
                plt.figtext(0.5, 0.05, date_time, color='black', fontsize=10,
                            fontweight='bold', ha='center', va='center')
                save_fn = "BD_" + date_time + ".png"
                fig.savefig(os.path.join(SAVE_picPATH, save_fn))
                print("成功选择图片：", filename, "保存的文件:", save_fn)
                coun += 1
                md5 = self.calculate_md5(filename)
                cursor.execute('''''
                    INSERT INTO images (md5, filepath, created_at, updated_at, labels)
                    VALUES (?, ?, ?, ?, ?)
                ''', (md5, filename, date_time, date_time, label_text))
                conn.commit()
                pixmap=QPixmap(filename)
                pixmap = pixmap.scaled(640, 480, Qt.KeepAspectRatio)
                self.image_frame2.setPixmap(pixmap)
                self.image_frame2.update()
                pixmap=QPixmap(os.path.join(SAVE_picPATH, save_fn))
                pixmap = pixmap.scaled(640, 480, Qt.KeepAspectRatio)
                self.image_frame1.setPixmap(pixmap)
                self.image_frame1.update()
                QTimer.singleShot(2000, lambda: None)
                QApplication.processEvents()
                plt.close()
            else:
                print("未选择图片")
        cursor.close()
        conn.close()
        print("\n")
```

为了计算图片的 MD5 值，提供了以下函数代码。使用`open`函数以二进制模式打开文件，并将其赋值给名为`file`的变量。接下来，使用`read`方法读取文件的全部内容，并将其赋值给名为`content`的变量。然后，创建了一个`hashlib.MD5`对象，并将其赋值给名为`MD5_hash`的变量。通过调用`update`方法，将文件内容传递给`MD5_hash`对象，以进行哈希计算。最后，使用`hexdigest`方法将哈希结果转换为十六进制字符串，并将其赋值给名为`MD5_value`的变量。这样，成功地完成了对文件内容进行哈希计算并转换为十六进制字符串的过程

```
def calculate_md5(self, file_path):
    with open(file_path, 'rb') as file:
        content = file.read()
    md5_hash = hashlib.md5()
    md5_hash.update(content)
    md5_value = md5_hash.hexdigest()
    return md5_value
```

图 3.6 计算图片 MD5 值的函数

3.2.4 用户配置更改的设计

本设计中，用户只需要点击按钮，即可实现对被处理文件夹路径，识别使用的模型，解密密钥文件的选取，更换，这主要是通过 PyQt5.QtWidgets 模块的 QFileDialog 类的 getExistingDirectory 或 getOpenFileName 的方法实现的。

```
def changeModel(self):
    file_path, _ = QFileDialog.getOpenFileName(
        self, "选择模型文件", "", "Model Files (*.pt)")
    if file_path:
        global model
        model_path = os.path.abspath(file_path)
        model = YOLO(model_path)
def change_save_path_button(self):
    global SAVE_picPATH
    SAVE_picPATH = QFileDialog.getExistingDirectory(self, "选择将要保存的文
    件夹", "")
def changePath(self):
    global folder_path
    folder_path = QFileDialog.getExistingDirectory(self, "选择将要处理的文
    件夹", "")
```

图 3.7 用户自定义运行配置的函数设计

3.2.5 数据安全模块的设计

加解密功能的实现方式如下：为了解密图片，定义了一个名 decrypt_allpng`的方法。该方法利用 Fernet 模块的 Fernet 类，创建了一个名为`fernet`的对象，并传入密钥。然后，使用 PyQt5.QtWidgets 模块的 QFileDialog 类的`getExistingDirectory`方法，弹出一个对话框，让用户选择一个文件夹。最后，遍历所选文件夹中的所有文件，并对每个 png 文件执行解密操作。

为了加密图片，定义了一个名为`encrypt_allpng`的方法。该方法利用 Fernet 模块的 Fernet 类，创建了一个名为`fernet`的对象，并传入密钥。然后，使用 PyQt5.QtWidgets 模块的 QFileDialog 类的`askDirectory`方法，弹出一个对话框，让用户选择一个文件夹。最后，遍历所选文件夹中的所有文件，并对每个 png 文件执行加密操作。

```
def decrypt_allpng(self,key):
    f = Fernet(key)
    folder_path = QFileDialog.getExistingDirectory()
    for file in os.listdir(folder_path):
        file_path = os.path.join(folder_path, file)
        if file_path.endswith(('.png')):
            with open(file_path, 'rb') as image_file:
                encrypted_data = image_file.read()
                decrypted_data = f.decrypt(encrypted_data)
            with open(file_path, 'wb') as image_file:
                image_file.write(decrypted_data)
        print(f'Decrypted {file_path}')
        self.label.setText(f'Decrypted {file_path}')
def encrypt_allpng(self,key):
    f = Fernet(key)
    folder_path = QFileDialog.getExistingDirectory()
    for file in os.listdir(folder_path):
        file_path = os.path.join(folder_path, file)
        if file_path.endswith(('.png')):
            with open(file_path, 'rb') as image_file:
                image_data = image_file.read()
            encrypted_data = f.encrypt(image_data)
            with open(file_path, 'wb') as image_file:
                image_file.write(encrypted_data)
            print(f'Encrypted {file_path}')
            self.label.setText(f'Encrypted {file_path}')
```

图 3.8 加解密模块的设计

3.3 本章小结

在开发环境和框架方面，为了实现整个功能，本章节内讨论选择了适合的工具。首先，进行了病菌图像识别模型的训练，确保系统的核心模块的运行。其次，设计了一个用户界面，以便用户能够方便地进行操作。接下来，进行了图像识别处理程序的设计，确保系统能够高效地处理图像。此外，还实现了用户配置更改的功能，让用户可以根据自己的需求自定义系统参数和设置。最后，为了保障用户数据的安全性和隐私保护，设计了一个数据安全模块。通过这些步骤的设计与实现，成功地开发了一个完全可运行的病菌图像自动分析与识别处理系统。。

第 4 章 系统测试

为了确保系统的稳定运行，系统测试是必不可少的。针对本课题设计中提到的各种功能，需要进行以下测试步骤。

4.1 用户界面显示的测试

对 def initUI(self):函数的测试，结果显示，用户图形界面可以正常显示，如图 4.1 所示

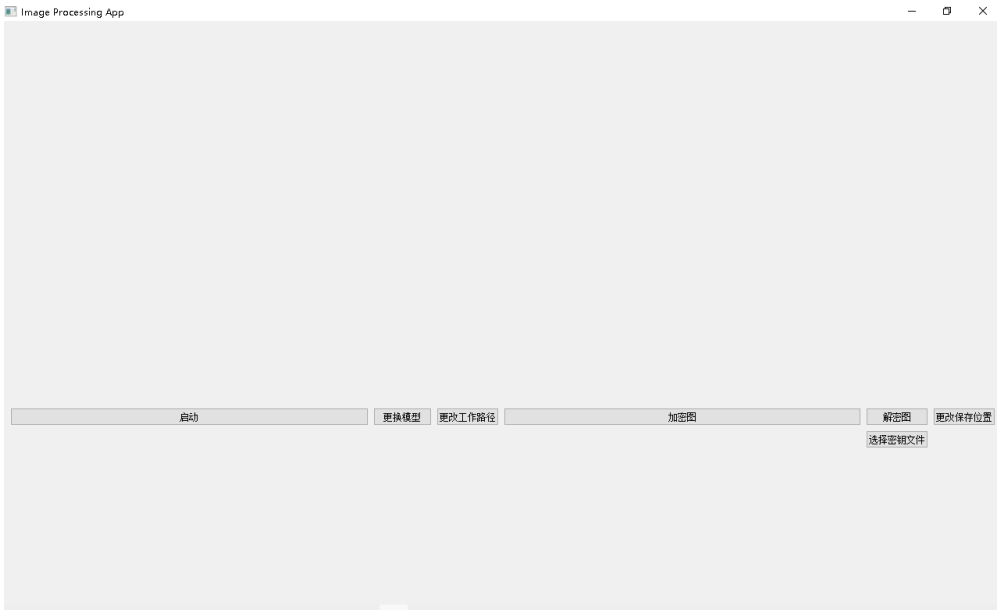


图 4.1 用户界面的测试运行图

4.2 按钮功能、用户修改运行配置的运行测试

确保函数的主要功能的正常运行，如下图 4.2，图 4.3 所示，程序执行对工作路径下的文件夹遍历处理病菌图片，自动统计各种类病菌的识别结果。

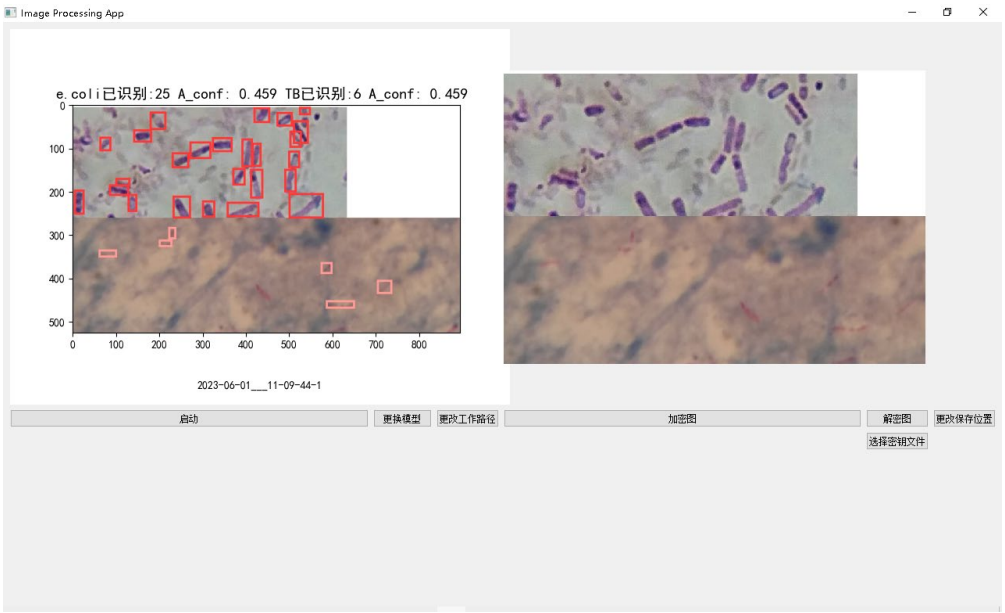


图 4.2 启动按钮的测试

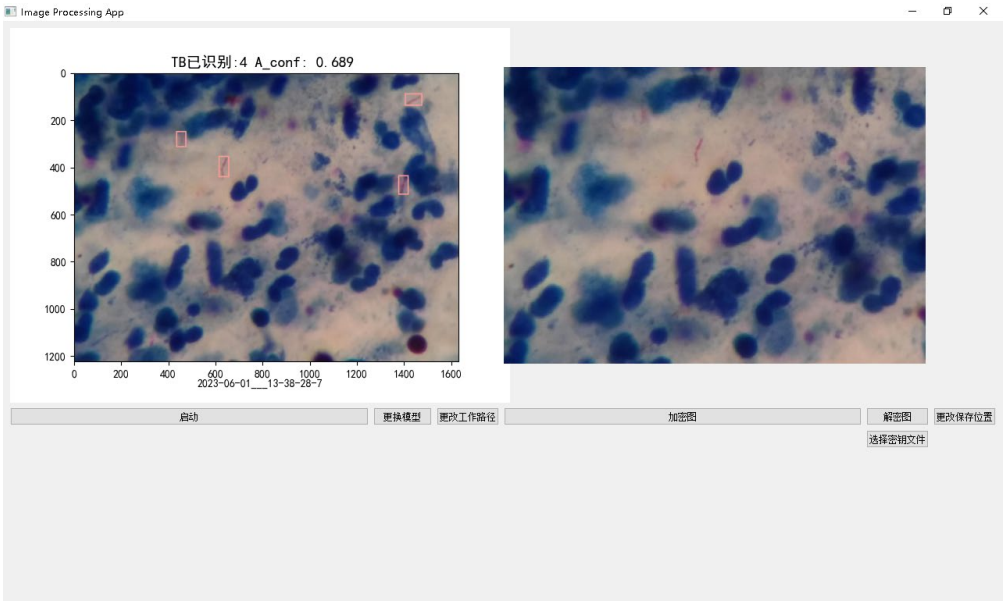


图 4.3 在新的工作目录下对所有图片遍历并自动图像处理

点击“更换模型”按钮，选择大肠杆菌的单菌种识别模型文件如图 4.4 所示，再点击“启动”按钮，可见识别结果如图 4.5 所示，对同一张包含两个种类的病菌图片处理，结果没有识别结核分枝杆菌，证明模型更换功能正常运行。

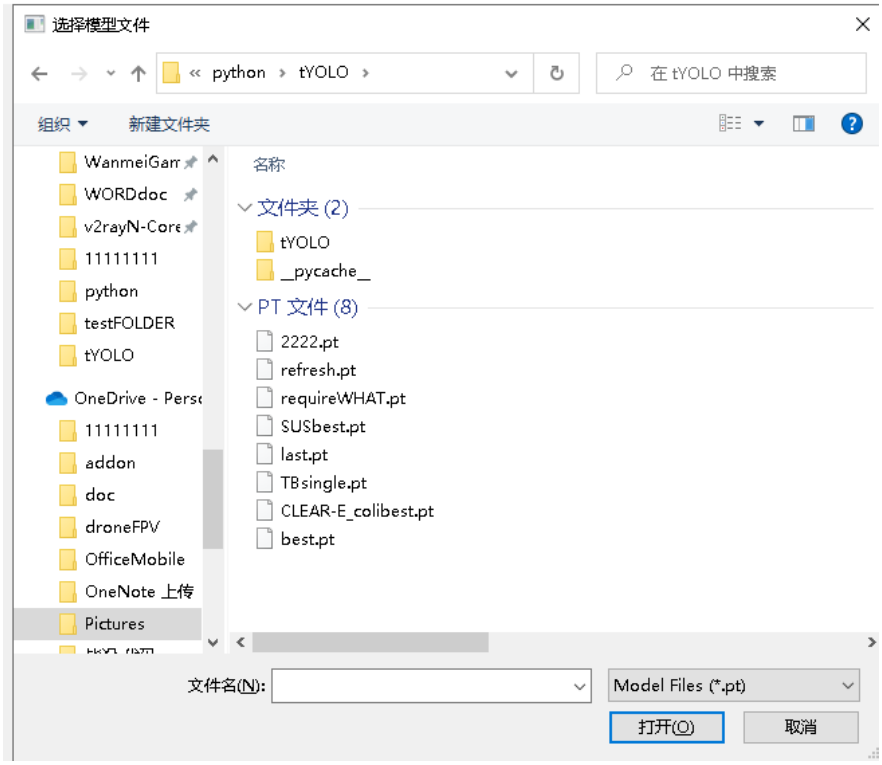


图 4.4 替换病菌识别模型文件

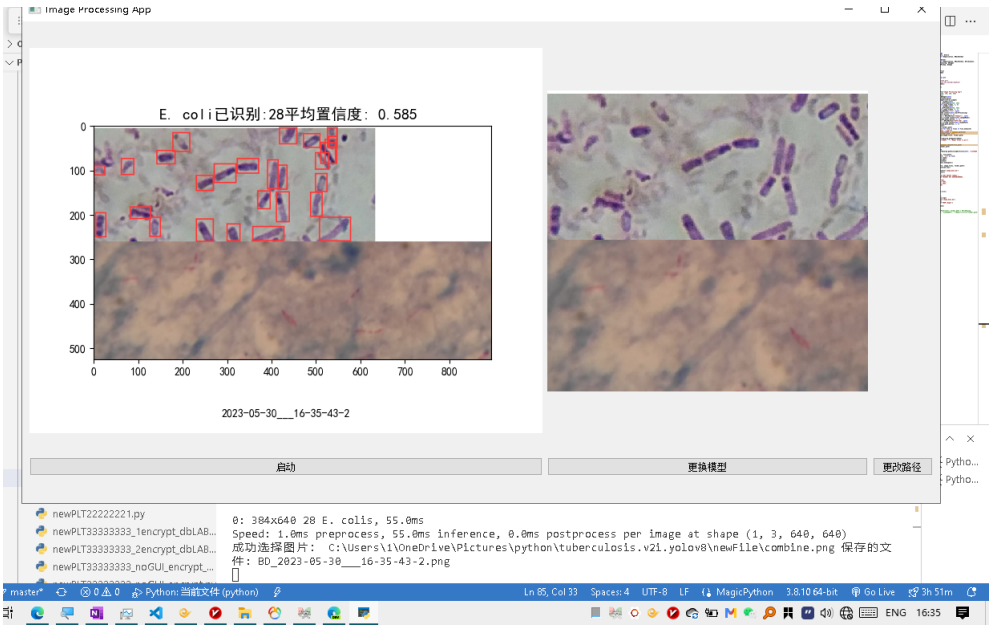


图 4.5 替换单菌种识别模型的程序只能识别一种病菌

现在的文件保存在当前文件夹下的“YOLO//tYOLO”如图 5.6 所示。

图 4.6 默认的图片保存位置

在点击“更改保存位置”按钮后，再点击”更改工作路径“，可发现处理后的文件存放在刚刚改变保存位置的路径里，如图 5.7 所示。

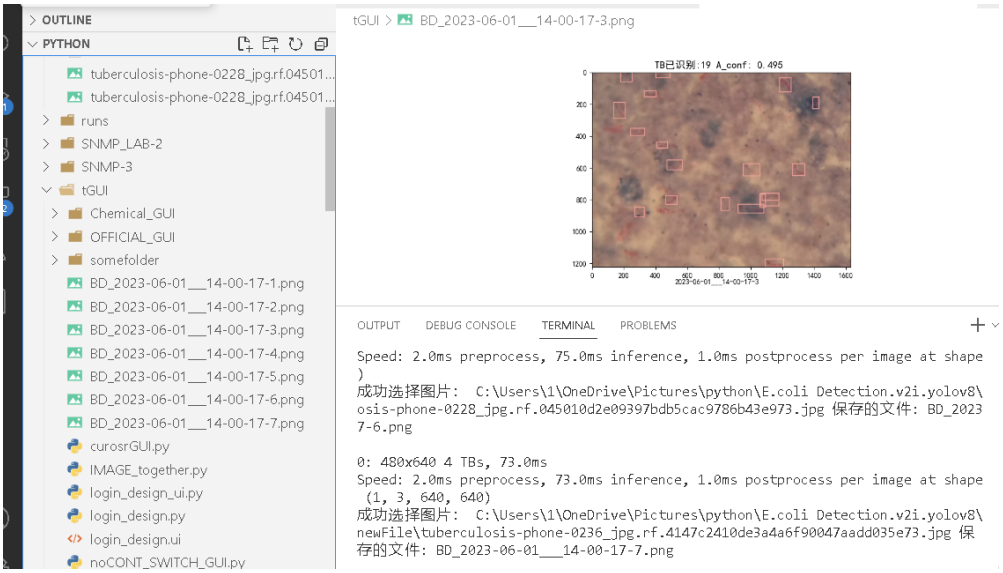


图 4.7 更改了保存路径后的图片存放位置

4.3 加密解密功能的测试

测试本课题设计附带的 Fernet 模块的加密解密功能，如图 4.8 所示，每次代码运行后，会根据 Fernet.generate_key()函数生成一段随机密钥并放入"key_"加上当前日期时间的文本文档案里。用加密模块对图片的二进制数据加密，打乱后的文本数据就会导致图片无法正常打开。

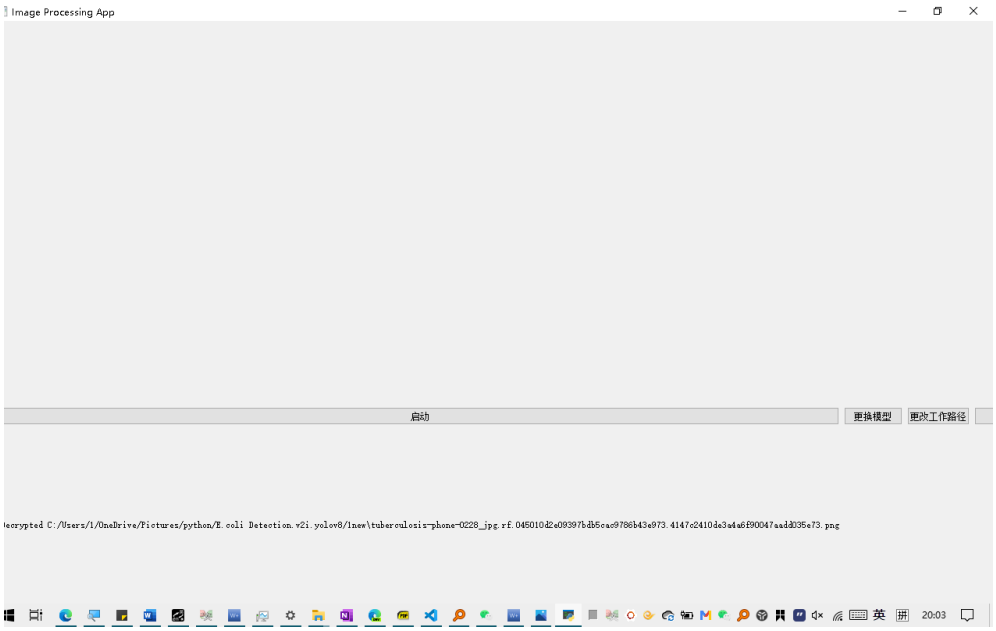


图 4.8 每次程序启动时都会先生成随机密钥文件

加密结果如图 4.9 所示，png 的图片在加密后无法打开。

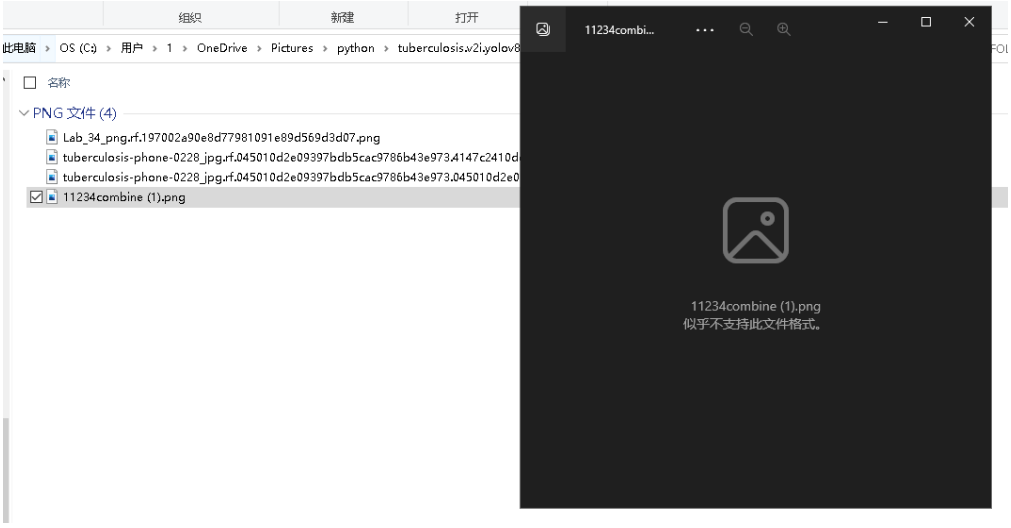


图 4.9 加密后的文件无法打开

通过点击”选择密钥文件“，选择用于解密的密钥文本，运行后解密如图 4.10 所示，

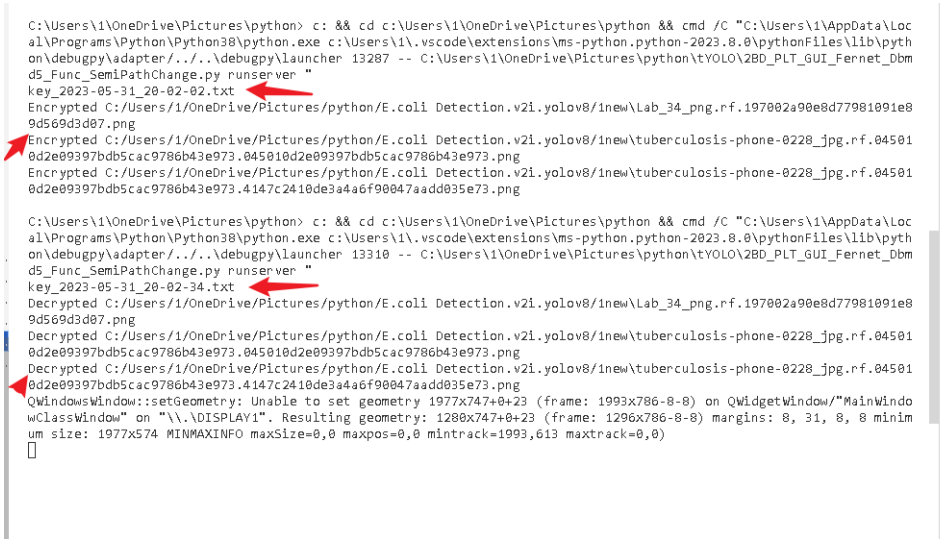


图 4.10 正确输出了解密信息

此时再次打开被加密的图片，正常打开如图 4.11 所示

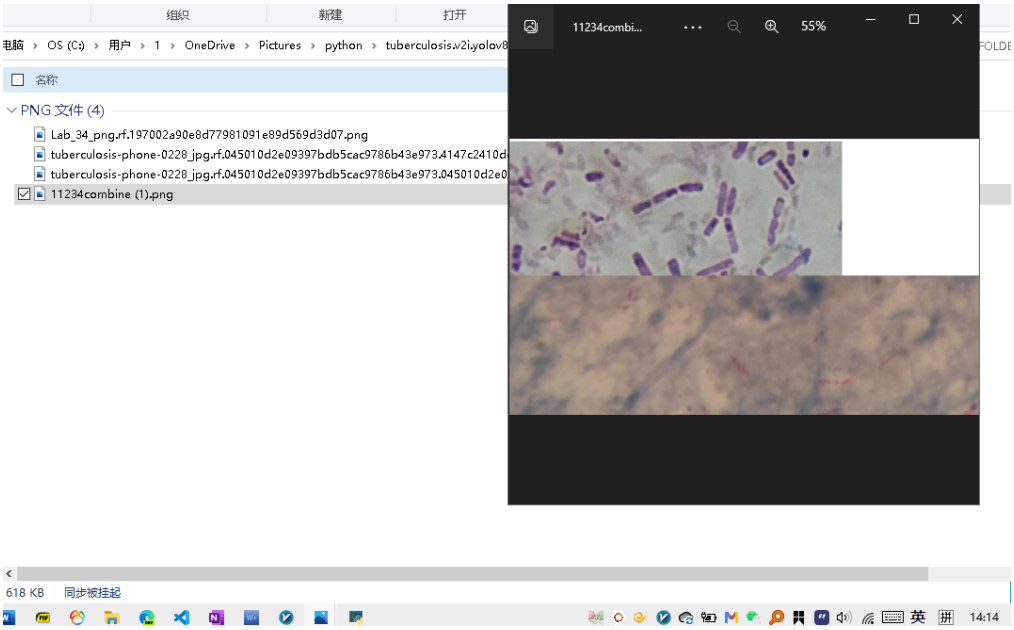


图 4.11 解密后的文件可以正常打开

4.4 测试中的难题与对策

（1）用于显示处理前和处理后的两个图片框没有随着代码遍历而逐步出现多个图片，就只无响应一会后，界面停留在最后一个图片组，并且文件夹里处理超过 20 个文件程序也会报错，卡死。

解决办法：上网查询后发现是需要在 process_images（）函数的最后加上 QApplication.processEvents()和 plt.close()以把图片加载到 pixmap 内处理，并通过后者释放内存，避免卡死。

（2）用 plt.show 的图像色调偏蓝

解决办法：总之就是使用 OpenCV 库中的`cv2.cvtColor()`函数，转换 res_plotted 图像的 BGR 颜色空间为 RGB。

（3）plt 的图像总是会在用户界面外自动跳出一个独立的图像框，影响界面其他功能的使用

解决办法：网上的案例是通过设置 fig = plt.figure()再将保存图片是设置用 fig 的方法来保存图片，即可阻止 plt 图片窗口像广告一样弹出

（4）更改工作参数的按钮类都没有生效

解决办法：通过复习 Python 的基础知识可知，相关功能函数里的变量是局部的，要改变的路径变量前加上 global 才能影响全局。

（5）encrypt_allpng_button.clicked.connect(self.encrypt_allpng(key))报错

解决办法：使用 lambda 函数是为了在加密解密按钮点击中传递额外的参数 key 给 self.encrypt_allpng()方法。通过网上资料得知，作为回调函数连接到信号时，此函数无法接受额外的参数。所以需要改成 encrypt_allpng_button.clicked.connect(lambda: self.encrypt_allpng(key))

（6）keyfname = QFileDialog.getOpenFileName(self, "选择密钥文件")报错

解决办法：经过查阅资料后发现，下划线_在某些情况下被用作占位符变量，用于表示不需要使用的返回值。在 Python 中，一般习惯使用下划线作为临时变量名，表示某个值将被忽略或不需要使用。因此，在处理该问题时，可以将下划线_添加到 keyname 之后，以接收多余的参数，并将第一个参数保留到 keyfname 中。

4.5 本章小结

经过测试本课题设计代码的各个功能模块，并在此过程中对本课题设计系统的实现逻辑进行不断优化，确保了所有界面的基本功能能够正常运行。这样的设计不仅适用于生产环境，还为将来进一步优化本课题奠定了坚实基础。

总结与展望

工作总结：本课题主要针对某地医疗检测机构工作流程的需求，对人工病菌图像的识别与计数这一传统流程实现一定程度的的自动化，可以对识别结果进行一定的数据保护，并期望通过本课题的设计，减少检测过程的时间，提高机构效率，降低生产成本。为此本课题设计并实现了一个基于 YOLOv8 的病菌图像识别系统，选择大肠杆菌和结核分枝杆菌作为识别对象。该系统具的目标检测功能能够对输入的工作目录下的所有 PNG 格式的病菌图像进行识别，每张平均处理时间不超过 0.2 秒，并能够在用户界面上展示识别结果的同时系统还能将图像和结果存储于 SQLite 数据库中，并利用 Fernet 技术对数据进行保护。

本课题设计的开篇部分阐述了研究的背景、目的和意义，以及国内外研究现状。随后进行了需求分析和设计内容的介绍。然后，详细介绍了涉及到的技术要点，包括 Python 语言、PyQt5 库、SQLite、Fernet 加密技术和 YOLOv8 目标检测的应用，并重点阐述了各项功能的程序设计和实现，包括病菌图像识别模型的训练、用户界面的设计、图像识别处理程序的逻辑设计、用户配置更改的设计以及数据安全模块的设计。最后，进行了系统测试，结果显示程序的各功能正常运行，验证了系统的可用性。

本课题的经济分析与社会影响

根据本课题设计内容，有如下经济性分析:

（1）节省人力成本：本课题设计的系统可以自动分析与识别病菌图像，相比于传统实验室方法，减少了人力资源和时间的消耗。

（2）设备和设施成本的节约：本课题的系统可能需要投资一些设备和设施，如计算机硬件、图像采集设备和存储设备等。但是，这些设备和设施的成本是一次性的，并且可以在长期使用中为实验室和医疗机构带来更多的效益。

（3）提高诊断准确性与效率：本课题的系统可以提高病菌诊断的准确性和效率。一定程度地减少误诊和漏诊，降低相应的治疗和药物的消耗的同时，减少平均等待时间，提高患者的满意度和就诊效率。

（4）系统设计的可持续性和长期效益：本自动分析与识别病菌图像的处理系统具备可持续性和潜在的长期效益。系统一旦建立并运行，它能够持续地提供高品质的病菌检测和分析服务，并在医疗机构提供的众多样本中不断更新，提升识别精准度，为将来的医疗机构和相关产业的实验室带来一定的经济效益。。

社会影响

（1）早诊早治：本系统设计能够自动分析与识别病菌图像，实现病菌的早发现和早分析，从而有利于早期疾病的治疗和诊断。这对于控制疾病的扩散和传播具有重要意义，公众的健康可以得到更好的保护。

（2）提升医疗服务质量：自动化的病菌图像分析系统能够提供准确和快速的识别病菌，降低了人为因素的误差和干扰。这将提升医疗服务的可靠性和质量，增加患者的满意度和信任。一定程度上减少医闹的发生。

（3）促进科学研究进展：本系统设计为科学研究提供了强大的平台和工具。在获得高质量的图像数据集的情况下，它能够加快对不同类型和变种的病菌的研究，使得医学领域的人才能够进一步深入了解病原体的特性和传播途径，向未来的疾病预防和控制提供重要的科学支撑。

（4）环境保护与食品安全：本系统设计可以应用于食品安全和环境监测领域。通过对病菌的快速分析和检测，可以及时发现和控制病原体在食品和环境中的存在，保障公众的安全与健康。

工作展望:本课题设计的不足之处有：

（1）由于无法获得医疗机构的充分合作与接触，病菌图像识别模型的训练数据集容量较小。因此，本课题只能从网上寻找图像数据集，但在图像质量稂莠不齐的训练样本下，这可能对模型的泛化能力和准确度造成影响。

（2）目前用户界面的美观性和交互性有待提高。当前界面的布局存在混乱，用户需要先最大化界面才能正确显示控件的排布。

（3）加密模块的安全性和稳定性需要进行进一步测试，以确保数据的保密性和稳定性。

（4）目前代码启动后无法自动扫描工作路径下的文件夹并处理新加入的图片文件，因此识别过程无法实现完全的自动化。

改进之处有：

（1）扩充病菌图像数据的种类和数量，使得模型的识别性能更加精确，高效。

（2）改进用户界面的设计布局，以提升用户体验。

（3）加密算法和机制方面寻找更先进的模块，以降低学习成本的同时确保数据的安全性和完整性。

系统的开发是需要与用户不断互动沟通中，实现功能优化。目前本课题设计仍有诸多不足之处，但期待将来能设计出高效的病菌识别系统，为攻克疾病尽一份力。

参考文献

[1] 张新峰, 沈兰荪.模式识别及其在图像处理中的应用[J].测控技术, 2004, (05) : 28 - 32.DOI : 10.19708 / j.ckjs .2004.05.010

[2] 黄煜. 基于广义结构元的动物病菌图像分形和识别方法研究[D].重庆交通大学,2013.

[3] 刘友江.细菌显微图像自动识别技术[J].计算机光盘软件与应用,2015,18(02):86-88.

[4] 邓佳丽,龚海刚,刘明.基于目标检测的医学影像分割算法[J].电子科技大学学报,2023,52(02):254-262.

[5] Hung, Jane, and Anne Carpenter. "Applying faster R-CNN for object detection on malaria images." Proceedings of the IEEE conference on computer vision and pattern recognition workshops. 2017.

[6] Yoshihara, Kyohei, and Kouichi Hirata. "Object Detection as Campylobacter Bacteria and Phagocytotic Activity of Leukocytes in Gram Stained Smears Images." ICPRAM. 2022.

[7] Ma, Luyao, et al. "Accelerating the Detection of Bacteria in Food Using Artificial Intelligence and Optical Imaging." Applied and Environmental Microbiology 89.1 (2023): e01828-22.

[8] 黄峰,冯勇.利用图像分割思想的二维混沌映射及图像加密算法[J].光学精密工程,2007,(07):1096-1103.

[9] 张菁.基于遗传蛙跳神经网络的马铃薯病斑图像分割研究[D].甘肃农业大学,2018.

[10] 王健.可视化病虫害信息采集与处理系统的开发与研究[D].北京工业大学,2017.

[11] 陈峰. 基于 YOLOv5 的细菌目标检测系统的设计与实现[D].华东师范大学,2022.DOI:10.27149/d.cnki.ghdsu.2022.003475.

致 谢

完成这篇论文，我首先就要衷心感谢我的导师[redacted]教授的指导和帮助。他对我的研究工作给予了非常多的支持和指导，为我提供了研究方向和解决课题的思路，还在我遇到困难和挫折时提供了有效及时的信息和支持。如果没有这样耐心而学识丰富的导师，我恐怕就要延迟毕业了。

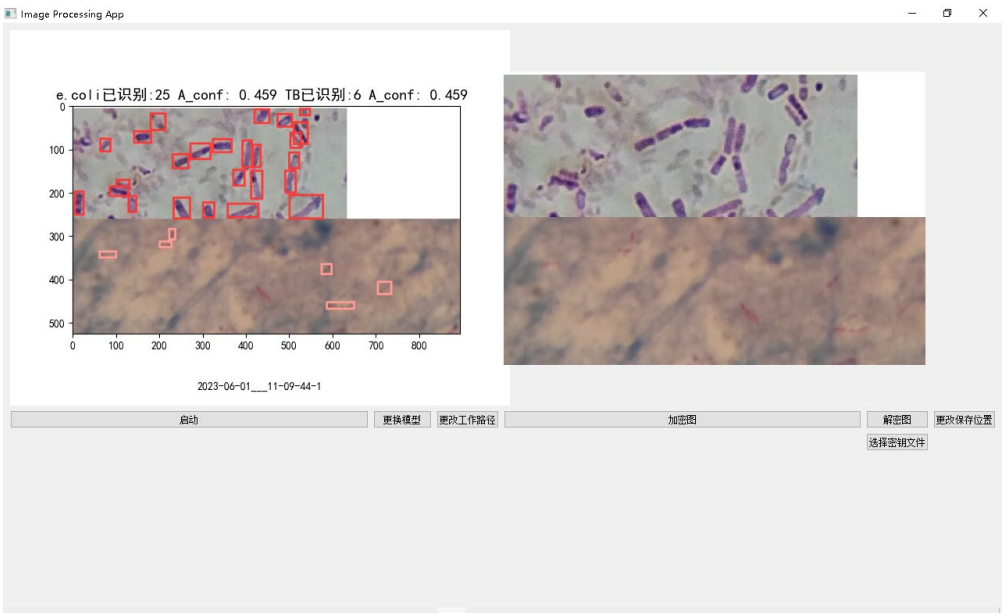
大学四年的时间过得很快，回想四年前刚入学的那一刻，似乎就在眨眼间。四年来，我所在的学府一直提供着优越的学习环境和平台，这是一份宝贵的财富。在这里，我接受了系统的专业知识培训和学术教育，为我的课题设计奠定了坚实的基础。我要衷心感谢母校的教师们，他们的教诲与指导使我受益匪浅。同时，我还要对学校领导表示感激，他们对我给予了无私的支持和关怀，为我提供了良好的学习条件和丰富的研究资源，这对我的学习生活起到了宝贵的支撑和保障。他们的付出与奉献我将永远铭记在心。此外，我也要感谢我的同学们，我们相互扶持、共同进步。我们一起度过了人生中的一个四年，互相分享知识和经验，成长。他们的陪伴和支持使我在人生道路上更加坚定。

再次感谢[redacted]教授的悉心指导和耐心帮助，我才能顺利完成这篇论文。再次感谢母校、校领导、同学和家人的互相帮助和支持，使得我可以没有后顾之忧地全力进行毕业设计的研究，并在学习中成长，开阔视野。我在将来的日子里，一定会铭记您们的帮助和付出，将以尽力继续追求学术进步，不停学习，回报您们的关爱和期望。

再次衷心感谢您们！

附 录

附录 A 用户界面的运行状态



附录 A1 界面的运行状态图

附录 B 程序主要代码

```
from datetime import datetime
from PyQt5.QtCore import Qt
from PyQt5.QtWidgets import QApplication, QMainWindow
import sys
import os
from cryptography.fernet import Fernet
from PyQt5.QtWidgets import QFileDialog
from PyQt5.QtCore import QTimer
from PyQt5.QtWidgets import QApplication, QMainWindow, QGridLayout, QLabel, QPushButton, QFileDialog, QWidget
from PyQt5.QtGui import QPixmap
from PIL import Image
import cv2
from ultralytics import YOLO
import hashlib
import datetime
import matplotlib.pyplot as plt
import sqlite3
key = Fernet.generate_key()
nowftime = datetime.datetime.now()
```

```
dt_string = nowftime.strftime("%Y-%m-%d_%H-%M-%S")
keyfname = "key_" + dt_string + ".txt"
with open(keyfname, "wb") as f:
    f.write(key)
print(f"{keyfname}")
SAVE_picPATH = r'tYOLO//tYOLO'
model = YOLO(r"tYOLO//refresh.pt")
folder_path = r"tuberculosis.v2i.yolov8//testFOLDER"
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.initUI()
    def initUI(self):
        self.setWindowTitle("Image Processing App")
        self.label = QLabel(self)
        self.setGeometry(50,50, 800, 600)
        grid = QGridLayout()
        grid.addWidget(self.label, 4, 0)
        central_widget = QWidget(self)
        central_widget.setLayout(grid)
        self.setCentralWidget(central_widget)
        self.image_frame1 = QLabel(self)
        self.image_frame1.setFixedSize(640, 480)
        grid.addWidget(self.image_frame1, 0, 0)
        self.image_frame2 = QLabel(self)
        self.image_frame2.setFixedSize(540, 380)
        grid.addWidget(self.image_frame2, 0, 3)
        start_button = QPushButton("启动", self)
        start_button.clicked.connect(self.startProcessing)
        grid.addWidget(start_button, 1, 0)
        change_model_button = QPushButton("更换模型", self)
        change_model_button.clicked.connect(self.changeModel)
        grid.addWidget(change_model_button, 1, 1)
        change_path_button = QPushButton("更改工作路径", self)
        change_path_button.clicked.connect(self.changePath)
        grid.addWidget(change_path_button, 1, 2)
        encrypt_allpng_button = QPushButton("加密图", self)
        encrypt_allpng_button.clicked.connect(lambda: self.encrypt_allpng(key
))
        grid.addWidget(encrypt_allpng_button, 1, 3)
        decrypt_allpng_button = QPushButton("解密图", self)
        decrypt_allpng_button.clicked.connect(lambda: self.decrypt_allpng(key
))
        grid.addWidget(decrypt_allpng_button, 1, 4)
```

```
        keyfnameb_button = QPushButton("选择密钥文件", self)
        keyfnameb_button.clicked.connect(self.keyfnameb_button)
        grid.addWidget(keyfnameb_button, 2, 4)
        change_save_path_button = QPushButton("更改保存位置", self)
        change_save_path_button.clicked.connect(self.change_save_path_button)

        grid.addWidget(change_save_path_button, 1, 5)
    def decrypt_allpng(self,key):
        f = Fernet(key)
        folder_path = QFileDialog.getExistingDirectory()
        for file in os.listdir(folder_path):
            file_path = os.path.join(folder_path, file)
            if file_path.endswith(('.png')):
                with open(file_path, 'rb') as image_file:
                    encrypted_data = image_file.read()
                    decrypted_data = f.decrypt(encrypted_data)
                with open(file_path, 'wb') as image_file:
                    image_file.write(decrypted_data)
            print(f'Decrypted {file_path}')
            self.label.setText(f'Decrypted {file_path}')
    def encrypt_allpng(self,key):
        f = Fernet(key)
        folder_path = QFileDialog.getExistingDirectory()
        for file in os.listdir(folder_path):
            file_path = os.path.join(folder_path, file)
            if file_path.endswith(('.png')):
                with open(file_path, 'rb') as image_file:
                    image_data = image_file.read()
                encrypted_data = f.encrypt(image_data)
                with open(file_path, 'wb') as image_file:
                    image_file.write(encrypted_data)
                print(f'Encrypted {file_path}')
                self.label.setText(f'Encrypted {file_path}')
    def keyfnameb_button(self):
        keyfname, _ = QFileDialog.getOpenFileName(self, "选择密钥文件")
        if keyfname:
            global key
            with open(keyfname, "rb") as f:
                key = f.read()
        pass
    def startProcessing(self):
        files = os.listdir(folder_path)
        image_files = [file for file in files if file.endswith(
            (".jpg", ".jpeg", ".png"))]
```

```
105.         self.process_images(image_files, folder_path)
106.     def changeModel(self):
107.         file_path, _ = QFileDialog.getOpenFileName(
108.             self, "选择模型文件", "", "Model Files (*.pt)")
109.         if file_path:
110.             global model
111.             model_path = os.path.abspath(file_path)
112.             model = YOLO(model_path)
113.     def change_save_path_button(self):
114.         global SAVE_picPATH
115.         SAVE_picPATH = QFileDialog.getExistingDirectory()
116.     def changePath(self):
117.         global folder_path
118.         folder_path = QFileDialog.getExistingDirectory(self, "选择将要处理的文
    件夹", "")
119.     def calculate_MD5(self, file_path):
120.         with open(file_path, 'rb') as file:
121.             content = file.read()
122.         MD5_hash = hashlib.MD5()
123.         MD5_hash.update(content)
124.         MD5_value = MD5_hash.hexdigest()
125.         return MD5_value
126.     def process_images(self, image_files, folder_path):
127.         now = datetime.datetime.now()
128.         coun = 1
129.         conn = sqlite3.connect('image_data.db')
130.         cursor = conn.cursor()
131.         cursor.execute('''''
132.             CREATE TABLE IF NOT EXISTS images (
133.                 id INTEGER PRIMARY KEY AUTOINCREMENT,
134.                 MD5 TEXT,
135.                 filepath TEXT,
136.                 created_at TEXT,
137.                 updated_at TEXT,
138.                 labels TEXT
139.             )
140.         ''')
141.         conn.commit()
142.         for file in image_files:
143.             filename = os.path.join(folder_path, file)
144.             filename = os.path.normpath(filename)
145.             cursor.execute(
146.                 'SELECT MD5 FROM images WHERE filepath = ?', (filename,))
147.             result = cursor.fetchone()
```

```
148.             if result is None:
149.                 im1 = Image.open(filename)
150.                 results = model.predict(source=im1)
151.                 res_plotted = results[0].plot(
152.                     line_width=5, conf=True, boxes=True, labels=False, pil=Tr
    ue)
153.                 boxes = results[0].boxes
154.                 names = model.names
155.                 sums = {}
156.                 counts = {}
157.                 for box in boxes:
158.                     cls = box.cls.item()
159.                     conf = box.conf.item()
160.                     if cls not in sums:
161.                         sums[cls] = 0
162.                         counts[cls] = 0
163.                     sums[cls] += conf
164.                     counts[cls] += 1
165.                 for cls in sums:
166.                     average = sums[cls] / counts[cls]
167.                 img = cv2.cvtColor(res_plotted, cv2.COLOR_BGR2RGB)
168.                 fig = plt.figure()
169.                 plt.imshow(img)
170.                 label_text = ''
171.                 for namee, countt in counts.items():
172.                     label_text += f"{names[namee]}已识
    别:{countt} A_conf: {average:.3f} "
173.                 plt.rcParams['font.sans-serif'] = ['SimHei']
174.                 title_text = f"{label_text}"
175.                 date_time = now.strftime("%Y-%m-%d___%H-%M-%S-
    ") + str(coun)
176.                 plt.title(title_text, fontsize=14, fontweight='bold')
177.                 plt.figtext(0.5, 0.05, date_time, color='black', fontsize=10,

178.                             fontweight='bold', ha='center', va='center')
179.                 save_fn = "BD_" + date_time + ".png"
180.                 fig.savefig(os.path.join(SAVE_picPATH, save_fn))
181.                 print("成功选择图片: ", filename, "保存的文件:", save_fn)
182.                 coun += 1
183.                 MD5 = self.calculate_MD5(filename)
184.                 cursor.execute('''''
185.                     INSERT INTO images (MD5, filepath, created_at, updated_at
    , labels)
186.                     VALUES (?, ?, ?, ?, ?)
```

```
                ''', (MD5, filename, date_time, date_time, label_text))
                conn.commit()
                pixmap=QPixmap(filename)
                pixmap = pixmap.scaled(640, 480, Qt.KeepAspectRatio)
                self.image_frame2.setPixmap(pixmap)
                self.image_frame2.update()
                pixmap=QPixmap(os.path.join(SAVE_picPATH, save_fn))
                pixmap = pixmap.scaled(640, 480, Qt.KeepAspectRatio)
                self.image_frame1.setPixmap(pixmap)
                self.image_frame1.update()
                QTimer.singleShot(2000, lambda: None)
                QApplication.processEvents()
                plt.close()
            else:
                print("未选择图片")
        cursor.close()
        conn.close()
        print("\n")
if __name__ == "__main__":
    app = QApplication(sys.argv)
    conn = sqlite3.connect('image_data.db')
    cursor = conn.cursor()
    cursor.execute('DELETE FROM images')
    conn.commit()
    mainWindow = MainWindow()
    mainWindow.show()
    sys.exit(app.exec_())
```