山东大学软件学院 2023 年度项目实训	文档编号	0004	版本	Ver 1.8
	项目名称	工业品缺陷检测系统		
	项目来源	项目实训		

工业品缺陷检测系统
技术实现方案

（内部资料　请勿外传）

编　　写：	李岩霖，张新钧，王秀宇，方正，王修智	**日 期：**	2023/6/27
检　　查：	李岩霖	**日 期：**	2023/6/28
审　　核：	李岩霖	**日 期：**	2023/6/28
批　　准：	李岩霖	**日 期：**	2023/6/28

山东大学软件学院

文档变更记录

序号	变更（+/-）说明	作者	版本号	日期	批准
1	(+)增加引言部分	李岩霖	Ver 1.0	2023/6/27	√
2	(+)增加应用程序设计部分	李岩霖	Ver 1.1	2023/6/27	√
3	(+)增加微信小程序前端架构部分	张新钧	Ver 1.2	2023/6/27	√
4	(+)增加 Web 前端架构部分	方正	Ver 1.3	2023/6/28	√
5	(+)增加关键技术实现部分	李岩霖	Ver 1.4	2023/6/28	√
6	(+)修改关键技术实现部分	王修智	Ver 1.5	2023/6/28	√
7	(+)增加功能实现部分	李岩霖	Ver 1.6	2023/6/28	√
8	(+)增加系统架构中的文件结构	李岩霖	Ver 1.7	2023/6/28	√
9	(+)修改 Web 前端架构部分	王秀宇	Ver 1.8	2023/6/29	√

目　　录

1. 引言

1.1 编写目的和范围

制造业的全面智能化发展对工业产品的质量检测提出了新的要求。表面缺陷检测是工业产品质量检测的关键部分。工业品检测是在制造过程中使用各种技术和方法来验证产品质量的过程。它的背景可以追溯到工业革命时期，当时随着生产规模的扩大，制造商们开始面临更多的质量问题和挑战。在现代制造业中，工业品检测发挥着至关重要的作用，对于确保产品质量、提高客户满意度以及保护品牌声誉都起着重要作用。

本项目针对若干张组装后的工业品图片，通过 AI 模型，识别工业品是否有缺陷（缺少螺丝），基于训练好的深度学习模型，建立 web 服务。提供上传图片，同时识别图片中物品是否有缺陷的功能。

本需求分析说明书编写的目的是说明系统的设计考虑，包括程序描述、输入/输出、算法和流程逻辑等，为软件编程和系统维护提供便利和基础。

本项目计划书主要面向本项目开发项目组成员，让项目组成员充分了解到本系统开发项目的需求、功能模块、业务逻辑等，从而完整、有效地开发以及实现系统全部的功能。

本项目计划书的预期读者为：

(1)项目经理：项目经理可以根据该文档了解预期产品的功能，并据此进行系统设计、项目管理，其中包括对系统进行配置管理和数据库文件更新管理，确保迭代版本的兼容性。

(2)系统分析师：系统分析师对系统背景进行调查，对需求进行分析，给出详细的需求文档。

(3)文档研读分析师：文档研读分析师，通过阅读大量的有关工业品检测的论文，搜索有关工业品检测的常用方法，为算法工程师提供思路，辅助算法工程师设计算法，训练模型。

(4)前端开发工程师：前端开发工程师按照系统设计师的设计开发文档进行系统前端的实现，并编写用户使用手册。

(5)后端开发工程师：后端开发工程师按照系统设计师的设计开发文档进行系统的实现，并编写用户使用手册。

(6)算法工程师：算法工程师根据文档研读分析师提供的思路，进行算法的设计和模型

的训练，与文档研读分析师一起进行模型推理加速工作。

(7)测试工程师：测试工程师根据开发工程师的用户使用手册以及本说明编写测试用例，并对软件产品进行功能性测试和非功能性测试。

1.2 术语表

序号	术语或缩略语	说明性定义
1	PM	Project Manager，项目经理
2	SA	System Analyst，系统分析师
3	DRA	Document Research Analyst，文档研读分析师
4	FPD	Front-end Program Designer，前端开发工程师
5	BPD	Back-end Program Designer，后端开发工程师
6	AE	Algorithm Engineer，算法工程师
7	TE	Testing Engineer，测试工程师

1.3 参考资料

资料名称	作者	文件编号、版本
《2023 年-2020 级项目实训实施计划 - 学生版本》	戴鸿君	V3
《2023-暑期项目实训申报通知-Intel-2》	郑艳飞，张建宇	2023 版

2. 系统设计

2.1 应用程序设计

2.1.1 系统架构概述

系统采用前后端分离的架构，前端使用 Vue 框架，结合 JavaScript 中的 JQuery 库，CSS

和 Ajax 技术进行设计；后端使用 Flask 框架；模型训练基于 tensorflow 开源框架。

2.1.2 前端架构

Web 前端文件目录如下：

```
Vue-font
├─ node_modules
├─ public
├─ src
│      ├─ assets
│      │      ├─ global.css
│      │      └─ vue.svg
│      └─components
│      │      ├─ Header.vue
│      │      └─  HelloWorld.vue
│      ├─ Layout
│      │      └─ Layout.vue
│      ├─ router
│      │      └─  index.js
│      ├─ views
│      │      ├─ mul.vue
│      │      └─ single.vue
│      ├─App.vue
│      ├─ main.js
│      │
│      └─ style.css
├─ index.html
├─ package.json
├─package-lock.json
└─ vite.config.js
```

Vue-font 文件夹下是前端所有内容。

node_modules 在 Vue 项目中，node_modules 文件夹是用来存放所需的第三方库和依赖模块的目录。当你使用 npm（Node 包管理器）或者 yarn 安装依赖时，这些依赖项将会自动下载并保存在 node_modules 文件夹中。node_modules 是一个由 NPM 生成和维护的文件夹，并且它位于项目根目录下。该文件夹保存了每个依赖包的代码、资源文件和其他相关文件。当你在项目中引入某个依赖时，Vue 会从 node_modules 文件夹中查找相应的包，并加载它们的模块。尽管 node_modules 文件夹可能会包含大量文件，但你无需手动处理它们。构建工具（如 webpack）会自动处理这些依赖，并确保只打包所需的部分。这样可以减小最终生成的应用程序的大小。

public 是创建 vite 项目自带的标识，在 Vue 项目中，public 文件夹是存放静态资源的地方。当你将资源放置在 public 文件夹中时，它们会被直接复制到构建输出的根目录。这使得你可以轻松地引用这些资源，而无需进行特殊的配置。

src 里面是前端界面的各种内容。其中 assets 存放写好的静态方法 global.css 和 vue3 自带的图标。components 存放 HelloWorld.vue 为前端的主界面,Header.vue 为前端的头部框架。Layout 存放 Layout.vue 是前端具体架构布局。Router 中的 index.js 是前端用来配置开发环境和生产环境的配置参数，控制前端的界面跳转。views 中的 mul.vue 和 single.vue 是多图片和单图片两个功能各自的界面。main.js 设置入口文件并对前端使用的 elementplus 等进行引入。style.css 是对于一些组件、字体等用 css 的方法进行样式修改。

index.html

Vue 项目中的 index.html 是项目的主入口文件，它是一个 HTML 文件，作为整个 Vue 应用的起点。在这个文件中，你会找到以下重要内容：

<!DOCTYPE html>: 这是文档类型声明，告诉浏览器使用哪种 HTML 版本解析。

<html>: HTML 页面的根元素。

<head>: 包含了一些页面的元数据和引用的外部资源，例如 css 样式和 JavaScript 脚本等。

<title>: 指定网页的标题，在浏览器标签栏和搜索结果中展示。

<body>: 页面的主体部分，包含实际的内容。

: Vue 应用的根元素，Vue 实例将挂载到这个 DOM 元素上。

<script>: 引入各种 JavaScript 文件，包括 Vue 的核心库和其他自定义的脚本。

<style>: 定义页面的样式，可以是内联样式或者引入外部的 CSS 文件。

此外，Vue Cli 生成的默认 index.html 还会包含一些脚本和样式相关的标签。例如：

<link rel="icon">: 指定网站图标的链接。

<meta name="viewport">: 设置视口（viewport）属性，控制移动设备上显示的页面宽度和缩放比例。

<script src="..."></script>: 引入打包后的 JavaScript 文件。

在 index.html 中，你可以设置全局的样式和引入其他依赖的库，同时也是 Vue 应用的起始点。在 Vue 应用中，index.html 一般只有一个实例绑定的根元素，所有的组件都会在 Vue 组件中进行处理，并渲染到这个根元素中。

package.json：在该 VUE 项目中，package.json 是一个重要的配置文件，用于管理项目

依赖和脚本命令。它位于项目的根目录，并且包含以下主要内容：

name：项目的名称 "vue-font"。

version：项目的版本号 "0.0.0"。

"type": "module"。

scripts：定义了一些可执行的脚本命令，比如运行开发服务器、打包项目等。"dev": "vite", "build": "vite build", "preview": "vite preview"

另外，package.json 还包含其它属性，用于配置项目的依赖管理和构建工具。例如：

dependencies：项目的生产环境依赖，即项目在运行时所需的依赖。

devDependencies：项目的开发环境依赖，即项目在开发过程中所需的依赖，如测试工具、构建工具等。

package-lock.json：package-lock.json 是 Vue 项目中的一个文件，记录了当前安装的依赖包的确切版本号和依赖关系树。它是由 npm install 命令自动生成并在每次安装或更新依赖时更新。作用为锁定依赖版本、加快依赖安装速度、保证构建一致性；当其他开发者从仓库拉取代码时，只需运行 npm install 即可根据 package-lock.json 安装相应的依赖。此时，package-lock.json 将会被读取，并且确保所有开发者使用的依赖版本与之前的一致。

vite.config.js：vite.config.js 是用于配置 Vite 项目的配置文件。插件部分: 导入了'vite-plugin-vue'插件，并在 plugins 数组中进行了注册。服务器部分: 在 server 对象中添加了 proxy 属性，该属性用于配置代理服务器。其中，/api 路径是要被代理的请求路径，target 属性指定了目标路径为"http://127.0.0.1:5000"，changeOrigin 属性表示是否改变请求头中的 host 字段，rewrite 属性用于重写请求路径。

微信小程序前端架构如下：

```
WechatProject
├─ pages
│      ├─ home
│      │      ├─home.js
│      │      ├─home.json
│      │      ├─home.wxml
│      │      ├─home.wxss
│      ├─ images
│      │      ├─OIP-C.jpg
│      │      ├─R-C.jpg
│      ├─ index
│      │      ├─index.js
│      │      ├─index.json
```

```
│   │   ├─index.wxml
│   │   ├─index.wxss
├─ utils
    ├─ util.js
├─.eslintrc.js
├─app.json
├─app.js
├─app.wxss
├─project.config.json
├─project.private.config.json
├─sitemap.json
```

WechatProject 文件夹下是微信小程序前端所有内容。

Pages 文件夹包括前端所有页面内容，其下属文件夹是各个界面的设计。包括视图层（WXML 文件和 WXSS 文件）和逻辑层（JavaScript）。

Home 文件夹是小程序首页界面，home.wxml 和 home.wxss 是对首页界面的渲染文件，home.js 包含了此界面的逻辑设计，包括各种点击事件，响应事件，与后端的交互等。home.json 是对整个界面的全局设计。

Images 文件夹包括了整个小程序前端界面所使用到的各种图片背景等

Index 界面是小程序主要的运行界面，同样是分成四个文件，index.wxml 和 index.wxss 对界面进行渲染，index.js 完成了界面的逻辑设计，index.json 是对整个界面的全局设计

Utils 文件夹主要包含 util.js 文件，用来存放全局的一些.js 文件，公共用到的一些事件处理代码文件可以放到该文件夹下，用于全局调用。

App.js 文件是小程序入口文件，用于定义全局数据和函数的调用，可以指定微信小程序的生命周期函数。

App.json 文件对小程序进行配置，小程序的全局配置，小程序的所有页面路径、界面表现、网络超时时间、底部 tab 等，可以在这个文件中配置小程序是由哪些页面组成，配置小程序的窗口及背景色，配置导航条样式，配置默认标题。

App.wxss 文件是全局的样式文件

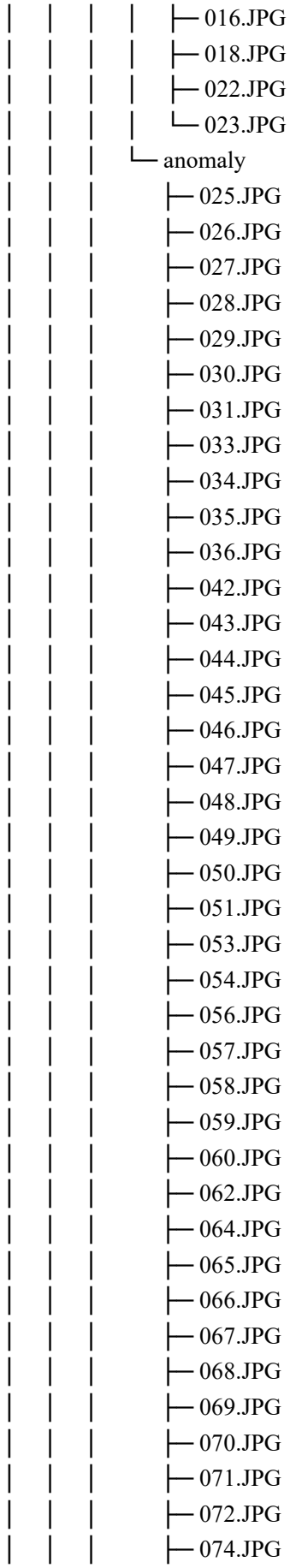
Project.config.json 文件用来保存开发工具配置项

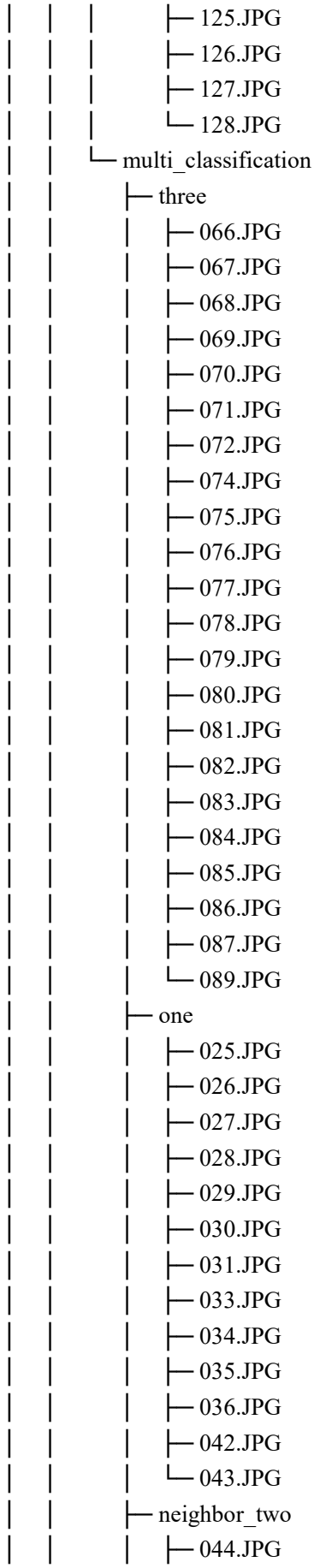
Sitemap.json 文件是网络地图，可以对小程序进行 seo 优化，让搜索排名靠前。

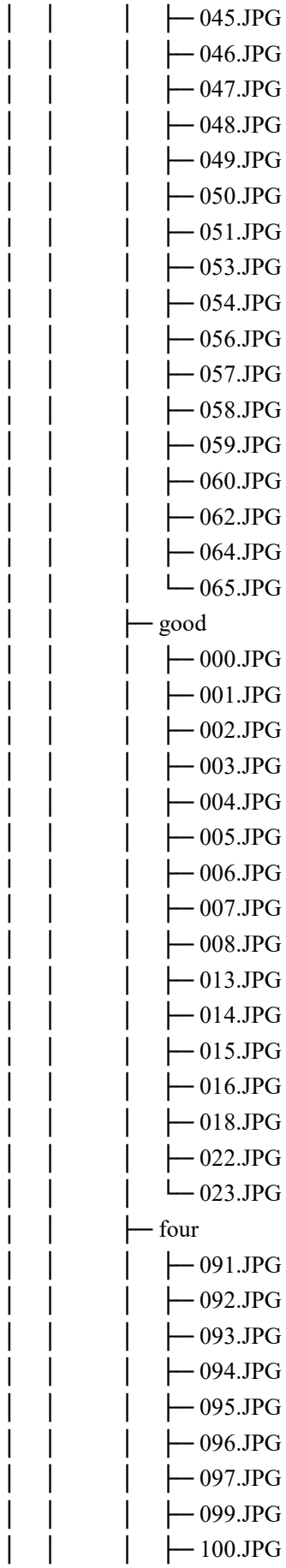
2.1.3 后端架构

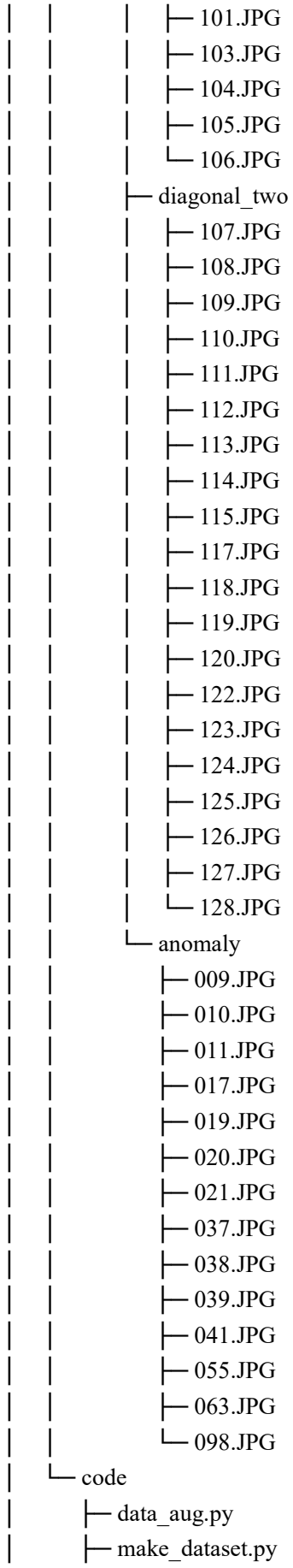
系统后端文件目录如下：

```
defect_product_backEnd
├── app.py
├── results.txt
├── requirements.txt
├── uploads
│   ├── 000.JPG
│   ├── 001.JPG
│   ├── 002.JPG
│   ├── 003.JPG
│   ├── 004.JPG
│   ├── 006.JPG
│   ├── 009.JPG
│   ├── 013.JPG
│   ├── 014.JPG
│   ├── 019.JPG
│   ├── 021.JPG
│   ├── 027.JPG
│   ├── lyc.png
│   ├── product_test.zip
│   └── product_test
│       ├── 000.JPG
│       └── 017.JPG
├── train_process
│   ├── model
│   │   ├── inceptionResNet152_ft.h5
│   │   ├── ResNet101_ft.h5
│   │   ├── ResNet152_ft.h5
│   │   ├── ResNet50_ft.h5
│   │   └── VGG19_ft.h5
│   ├── defect_product_dataset
│   │   ├── second_classification
│   │   │   ├── good
│   │   │   │   ├── 000.JPG
│   │   │   │   ├── 001.JPG
│   │   │   │   ├── 002.JPG
│   │   │   │   ├── 003.JPG
│   │   │   │   ├── 004.JPG
│   │   │   │   ├── 005.JPG
│   │   │   │   ├── 006.JPG
│   │   │   │   ├── 007.JPG
│   │   │   │   ├── 008.JPG
│   │   │   │   ├── 013.JPG
│   │   │   │   ├── 014.JPG
│   │   │   │   ├── 015.JPG
```

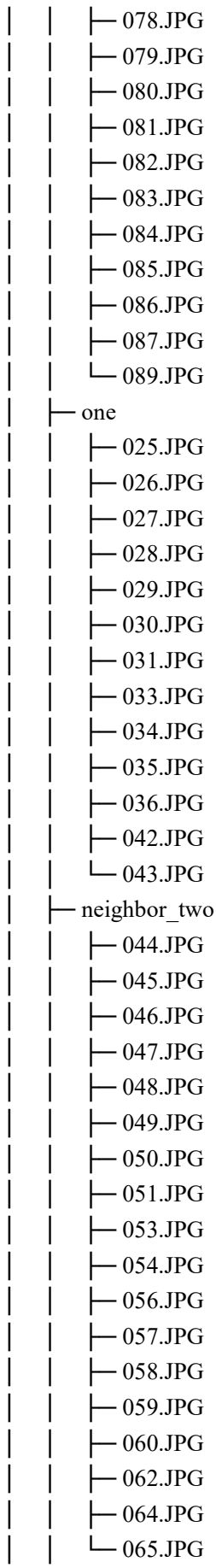

016.JPG
018.JPG
022.JPG
023.JPG
anomaly
025.JPG
026.JPG
027.JPG
028.JPG
029.JPG
030.JPG
031.JPG
033.JPG
034.JPG
035.JPG
036.JPG
042.JPG
043.JPG
044.JPG
045.JPG
046.JPG
047.JPG
048.JPG
049.JPG
050.JPG
051.JPG
053.JPG
054.JPG
056.JPG
057.JPG
058.JPG
059.JPG
060.JPG
062.JPG
064.JPG
065.JPG
066.JPG
067.JPG
068.JPG
069.JPG
070.JPG
071.JPG
072.JPG
074.JPG

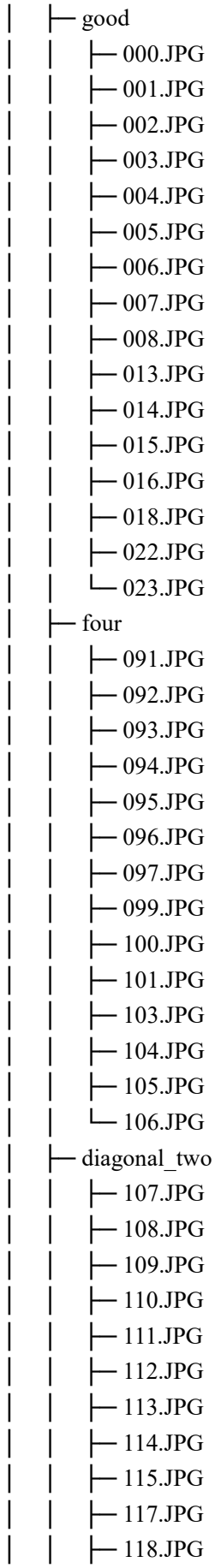

125.JPG
126.JPG
127.JPG
128.JPG
multi_classification
three
066.JPG
067.JPG
068.JPG
069.JPG
070.JPG
071.JPG
072.JPG
074.JPG
075.JPG
076.JPG
077.JPG
078.JPG
079.JPG
080.JPG
081.JPG
082.JPG
083.JPG
084.JPG
085.JPG
086.JPG
087.JPG
089.JPG
one
025.JPG
026.JPG
027.JPG
028.JPG
029.JPG
030.JPG
031.JPG
033.JPG
034.JPG
035.JPG
036.JPG
042.JPG
043.JPG
neighbor_two
044.JPG


045.JPG
046.JPG
047.JPG
048.JPG
049.JPG
050.JPG
051.JPG
053.JPG
054.JPG
056.JPG
057.JPG
058.JPG
059.JPG
060.JPG
062.JPG
064.JPG
065.JPG
good
000.JPG
001.JPG
002.JPG
003.JPG
004.JPG
005.JPG
006.JPG
007.JPG
008.JPG
013.JPG
014.JPG
015.JPG
016.JPG
018.JPG
022.JPG
023.JPG
four
091.JPG
092.JPG
093.JPG
094.JPG
095.JPG
096.JPG
097.JPG
099.JPG
100.JPG


101.JPG
103.JPG
104.JPG
105.JPG
106.JPG
diagonal_two
107.JPG
108.JPG
109.JPG
110.JPG
111.JPG
112.JPG
113.JPG
114.JPG
115.JPG
117.JPG
118.JPG
119.JPG
120.JPG
122.JPG
123.JPG
124.JPG
125.JPG
126.JPG
127.JPG
128.JPG
anomaly
009.JPG
010.JPG
011.JPG
017.JPG
019.JPG
020.JPG
021.JPG
037.JPG
038.JPG
039.JPG
041.JPG
055.JPG
063.JPG
098.JPG
code
data_aug.py
make_dataset.py

```
│       ├── predict.py
│       ├── result.txt
│       ├── test.py
│       ├── train_inceptionResnet152.py
│       ├── train_resnet101.py
│       ├── train_resnet152.py
│       ├── train_resnet50.py
│       └── train_VGG19.py
├── selenium_test
│   ├── geckodriver.log
│   ├── speed_test.py
│   ├── test_result.txt
│   ├── web_test.py
│   └── test_data
│       ├── 000.JPG
│       └── product_test.zip
├── quantization
│   ├── compressor.py
│   ├── dataset.py
│   ├── make_dataset.py
│   └── __pycache__
│       ├── dataset.cpython-38.pyc
│       └── make_dataset.cpython-38.pyc
├── model_information
│   └── flops.py
├── model
│   ├── inceptionResNet152_ft.h5
│   ├── ResNet101_ft.h5
│   ├── ResNet152_ft.h5
│   ├── ResNet50_ft.h5
│   └── VGG19_ft.h5
├── dataset
│   ├── three
│   │   ├── 066.JPG
│   │   ├── 067.JPG
│   │   ├── 068.JPG
│   │   ├── 069.JPG
│   │   ├── 070.JPG
│   │   ├── 071.JPG
│   │   ├── 072.JPG
│   │   ├── 074.JPG
│   │   ├── 075.JPG
│   │   ├── 076.JPG
│   │   ├── 077.JPG
```


078.JPG
079.JPG
080.JPG
081.JPG
082.JPG
083.JPG
084.JPG
085.JPG
086.JPG
087.JPG
089.JPG
one
025.JPG
026.JPG
027.JPG
028.JPG
029.JPG
030.JPG
031.JPG
033.JPG
034.JPG
035.JPG
036.JPG
042.JPG
043.JPG
neighbor_two
044.JPG
045.JPG
046.JPG
047.JPG
048.JPG
049.JPG
050.JPG
051.JPG
053.JPG
054.JPG
056.JPG
057.JPG
058.JPG
059.JPG
060.JPG
062.JPG
064.JPG
065.JPG


good
000.JPG
001.JPG
002.JPG
003.JPG
004.JPG
005.JPG
006.JPG
007.JPG
008.JPG
013.JPG
014.JPG
015.JPG
016.JPG
018.JPG
022.JPG
023.JPG
four
091.JPG
092.JPG
093.JPG
094.JPG
095.JPG
096.JPG
097.JPG
099.JPG
100.JPG
101.JPG
103.JPG
104.JPG
105.JPG
106.JPG
diagonal_two
107.JPG
108.JPG
109.JPG
110.JPG
111.JPG
112.JPG
113.JPG
114.JPG
115.JPG
117.JPG
118.JPG

```
│   │   ├── 119.JPG
│   │   ├── 120.JPG
│   │   ├── 122.JPG
│   │   ├── 123.JPG
│   │   ├── 124.JPG
│   │   ├── 125.JPG
│   │   ├── 126.JPG
│   │   ├── 127.JPG
│   │   └── 128.JPG
│   └── anomaly
│       ├── 009.JPG
│       ├── 010.JPG
│       ├── 011.JPG
│       ├── 017.JPG
│       ├── 019.JPG
│       ├── 020.JPG
│       ├── 021.JPG
│       ├── 037.JPG
│       ├── 038.JPG
│       ├── 039.JPG
│       ├── 041.JPG
│       ├── 055.JPG
│       ├── 063.JPG
│       └── 098.JPG
└── .idea
    ├── .gitignore
    ├── defect_product_backEnd.iml
    ├── misc.xml
    ├── modules.xml
    ├── workspace.xml
    └── inspectionProfiles
        ├── profiles_settings.xml
        └── Project_Default.xml
```

defect_product_backEnd 文件夹下是后端所有内容，包括 flask 后端所有内容和模型训练的所有相关内容。

app.py 是后端 flask 服务入口。

results.txt 是后端接收 Web 端接收批量文件上传后，将检测结果写入的文件，作为附件发送至用户邮箱。

uploads 下存放用户通过前台上传的待检测的文件，其中，Web 端和微信小程序端上传的单张图片直接存在 uploads 文件下，Web 端批量上传的 zip 压缩包先存在 uploads 下，然后以原名解压在 uploads 下。

model 下存放基于迁移学习的 Tensorflow 模型文件，VGG19_ft.py 是基于预训练的 VGG19 模型进行 fine-turning 后的模型；ResNet50_ft.py 是基于预训练的 ResNet50 模型进行 fine-turning 后的模型；ResNet101_ft.py 是基于预训练的 ResNet101 模型进行 fine-turning 后的模型；ResNet152_ft.py 是基于预训练的 ResNet152 模型进行 fine-turning 后的模型；inceptionResNet152_ft.py 是基于预训练的 inceptionResNet152 模型进行 fine-turning 后的模型。

dataset 下存放的是原始数据集，dataset 下有 7 个文件夹，表示 7 类，其中，three 表示零件缺三个螺丝；neighbor_two 表示零件缺临边两个螺丝；four 表示零件缺四个螺丝；diagonal_two 表示零件缺对角两个螺丝；anomaly 表示零件图片缺角；good 表示零件合格。

.idea 是运行时产生的一些文件。

model_information 下存放着评估模型参数量和 FLOPs 的代码，用以评估模型。

quantization 下存放模型量化相关的代码，compressor.py 是模型量化的主要代码；dataset.py 是读取原始数据制作输入模型的 dataloader 的代码；make_dataset.py 是读取原始数据制作数据集的代码。

selenium_test 下存放的是 selenium 自动化测试的有关文件。geckodriver.log 是自动化测试记录，speed_test.py 是使用 cProfile 测试项目性能的代码，web_test.py 是 Web 端的 selenium 自动化测试的代码，test_data 文件夹下存放测试需要的数据。

train_process 下存放的是有关模型训练的相关文件。下面的 model 文件夹是训练过程中模型的保存位置，dafect_product_dataset 文件下存放 second_classification 和 multi_classification 两个文件夹，分别表示二分类的原始数据和多分类的原始数据。second_classfication 下有 good 和 anomaly 两个文件夹，分别表示是合格零件和不合格零件。multi_classification 文件夹下有 7 个文件夹，表示 7 类，其中，three 表示零件缺三个螺丝；neighbor_two 表示零件缺临边两个螺丝；four 表示零件缺四个螺丝；diagonal_two 表示零件缺对角两个螺丝；anomaly 表示零件图片缺角；good 表示零件合格。code 文件夹下存放的是有关模型训练的相关代码，data_aug.py 是数据增强代码，make_dataset.py 是读取原始数据制作数据集的代码，predict.py 是利用模型检测单张图片的代码，result.txt 是模型检测图片的结果，test.py 是模型检测批量图片的代码，train_inceptionResnet152.py 是训练 inceptionResnet152 模型的代码，train_resnet101.py 是训练 resnet101 模型的代码，train_resnet152.py 是训练 resnet152 模型的代码，train_resnet50.py 是训练 resnet50 模型的代码，train_VGG19.py 是训练 VGG19 模型的代码。

3. 系统实现

3.1 关键技术实现

3.1.1 Vue 框架的使用

Vue 是一种流行的 JavaScript 前端框架，用于构建用户界面。它是一个开源项目，由尤雨溪（Evan You）创建并维护。

Vue 的设计目标是提供一种简单、灵活和高效的方式来构建交互式的 Web 界面。它采用了组件化的开发模式，允许开发者将界面拆分为独立的可重用组件。每个组件都有自己的逻辑和状态，并可以通过组合这些组件来构建复杂的应用程序。

Vue 使用了虚拟 DOM (Virtual DOM) 技术来优化性能。当数据发生变化时，Vue 会计算出与当前 DOM 状态的差异，并只更新需要改变的部分，而不是重新渲染整个页面。这使得 Vue 在处理大规模数据和频繁更新的情况下表现出色。

Vue 还提供了响应式数据绑定机制，即数据的变化可以自动反映在界面上，使得开发者无需手动操作 DOM。此外，Vue 还支持指令系统，通过指令可以对 DOM 进行操作和控制。

Vue 的学习曲线相对较低，文档详细且易于理解，使得初学者也能够迅速上手。同时，Vue 具有庞大的社区支持和丰富的插件生态系统，为开发者提供了丰富的资源和工具。

总之，Vue 是一款功能强大、易用且高效的 JavaScript 前端框架，适合用于构建现代化的 Web 应用程序。

3.1.2 用户输入验证

在系统中，有用户输入的地方，如邮箱。为了保证用户输入的合法性质，系统设置了输入验证。代码如下：

```
rules: {
  email: [
    { required: true, message: '请输入邮箱', trigger: 'blur' },
    { type: 'email', message: '邮箱格式不正确', trigger: ['blur', 'change'] }
  ]
}
```

3.1.3 Tensorflow2.0 和 keras

TensorFlow 2 和 Keras 是两个与深度学习相关的 Python 库。

TensorFlow 2 是 Google 开发的开源机器学习框架，它提供了一种灵活且高效的方式来构建和训练各种机器学习模型。相对于早期版本的 TensorFlow，TensorFlow 2 更加易用且用户友好。TensorFlow 2 采用了 Eager Execution 模式，使得代码编写更加直观，并且提供了大量高级 API，使开发者能够快速构建深度学习模型。此外，TensorFlow 2 还增加了许多新功能，如动态图计算、模型序列化和可部署性的改进等。它支持多种硬件设备（如 CPU、GPU 和 TPU）以及分布式计算，可以处理从小规模实验到大规模生产环境的需求。

Keras 是一个高级神经网络 API，最初是作为独立库开发的，后来被整合到 TensorFlow 中。它专注于简化深度学习模型的构建过程，提供了一组简洁而直观的接口，使得使用者可以轻松地定义、训练和评估神经网络模型。Keras 具有可扩展性强、模块化设计和易用性等优点，适用于各种任务，包括图像分类、文本处理和序列生成等。在 TensorFlow 2 中，Keras 被作为其官方的高级 API，成为了构建深度学习模型的主要工具。

总结而言，TensorFlow 2 是一个功能强大的机器学习框架，而 Keras 则提供了简洁易用的接口来构建深度学习模型。由于它们的紧密结合，用户在使用 TensorFlow 2 时可以自由选择使用原生 TensorFlow API 或者更简单的 Keras API 来构建自己的模型。

TensorFlow 在工业界受欢迎的原因有以下几点：

广泛的支持和社区: TensorFlow 是由 Google 开发并维护的，得到了大型公司和机构的广泛支持。它拥有庞大而活跃的开发者社区，提供了丰富的文档、教程、示例代码和预训练模型等资源，使得使用者可以轻松获取帮助和分享经验。

灵活性和可扩展性: TensorFlow 提供了强大的灵活性和可扩展性，适用于各种规模的任务和需求。它支持多种硬件设备（如 CPU、GPU 和 TPU），以及分布式计算，能够处理大规模数据和复杂模型的训练和推理。此外，TensorFlow 支持多种编程语言接口，包括 Python、C++ 和 Java，使得开发者可以根据自己的偏好选择合适的编程语言。

生态系统和部署能力: TensorFlow 提供了丰富的工具和库，构建了一个完整的深度学习生态系统。例如，TensorFlow Hub 提供了大量的预训练模型和模型组件，方便开发者快速搭建模型；TensorBoard 可以可视化模型训练过程和性能指标；TensorFlow Serving 和

TensorFlow Lite 等工具支持模型的部署和移植到生产环境。这些工具和库提供了综合而完善的解决方案，满足了工业界对于深度学习模型开发和部署的需求。

产业应用和成功案例: TensorFlow 在工业界有许多成功的应用案例，被广泛应用于各个领域，如图像识别、自然语言处理、语音识别、推荐系统等。许多大型公司和组织都在使用 TensorFlow 进行创新研究和产品开发，使得 TensorFlow 成为业界的首选之一。

尽管 TensorFlow 在工业界拥有广泛的应用和支持，但也要根据具体需求选择合适的工具。近年来，PyTorch 也获得了很大的关注和增长，成为另一个备受青睐的深度学习框架，特别在学术界和研究领域有较高的使用率。

针对我们这一项目，我们最后决定选用 Tensorflow 中的 keras 来搭建训练模型。

3.1.4 keras.applications 中的预训练模型

keras 库中的 keras.applications 模块提供了预训练模型，具有以下优点：

高性能：预训练模型经过大规模的训练，在广泛的数据集上表现出色。这些模型已经学会了从图像中提取有用的特征，可以用于各种计算机视觉任务，如图像分类、目标检测和图像生成等。

节省时间和资源：使用预训练模型可以避免从头开始训练深度神经网络所需的大量时间和计算资源。这些模型已经在强大的硬件设备上进行了训练，并且经过了仔细的调优，因此可以立即应用于下游任务中。

迁移学习：预训练模型提供了迁移学习的便利。你可以使用这些模型作为基础，并通过微调或添加自定义层来适应你的具体任务。通过在预训练模型的基础上进行微调，你可以在较少的训练样本上获得更好的性能。

广泛的应用领域：Keras 库中的预训练模型适用于各种计算机视觉任务和领域。无论是图像分类、目标检测、图像分割还是图像生成，你都可以从这些模型中选择适合新任务的模型。

社区支持和文档丰富：由于这些预训练模型的受欢迎程度，可以轻松地找到大量的教程、示例代码和文档资源。Keras 社区非常活跃，可以与其他开发人员交流经验并获得支持。

综上所述，Keras 库中的 keras.applications 模块中的预训练模型具有高性能、节省时间和资源、适用于迁移学习、广泛的应用领域以及丰富的社区支持等优点，使其成为计算机视觉任务中强有力的工具。

基于此，本项目选择使用基于 keras.applications 的预训练模型进行迁移学习。

3.2 功能实现

3.2.1 Web 端功能实现

1. 单张图片上传功能

(1)单图片上传：用户通过点击图片上传按钮，调用本地文件上传界面，用户选择待检测的图片，上传成功后，即时反馈检测结果，推理用时和置信度。

部分重要代码如下：

```
<script >
import { ref } from 'vue'
import axios from "axios";
import { ElMessage } from 'element-plus'
import {result} from "lodash-es";
export default {
  data() {
    return {
      server_url: "http://10.27.202.195:5000",
      activeName: "first",
      active: 0,
      centerDialogVisible: true,
      url_1: "",
      url_2: "",
      textarea: "",
      srcList: [],
      srcList1: [],
      feature_list: [],
      feature_list_1: [],
      feat_list: [],
      url: "",
```

```
        visible: false,
        wait_return: "等待上传",
        wait_upload: "等待上传",
        loading: false,
        table: false,
        isNav: false,
        showbutton: true,
        percentage: 0,
        fullscreenLoading: false,
        opacitys: {
            opacity: 0,
        },
        dialogTableVisible: false,
        dialogFormVisible:false,
        form:{},
        imageUrl:null
    };
},
created() {
    console.log("")
},
methods: {
    result,
    getObjectURL(file) {
        var url = null;
        if (window.createObjcectURL != undefined) {
            url = window.createOjcectURL(file);
        } else if (window.URL != undefined) {
            url = window.URL.createObjectURL(file);
        } else if (window.webkitURL != undefined) {
```

```
        url = window.webkitURL.createObjectURL(file);
    }
    return url;
},
update(e) {
    this.percentage = 0;
    this.dialogTableVisible = true;
    this.url_1 = "";
    this.url_2 = "";
    this.srcList = [];
    this.srcList1 = [];
    this.wait_return = "";
    this.wait_upload = "";
    this.feature_list = [];
    this.feat_list = [];
    this.fullscreenLoading = true;
    this.loading = true;
    this.showbutton = false;
    let file = e.target.files[0];
    this.url_1 = this.$options.methods.getObjectURL(file);
    //let param = new FormData(); //创建 form 对象
    //param.append("file", file, file.name); //通过 append 向 form 对象添加数据
    let param =file;
    var timer = setInterval(() => {
        this.myFunc();
    }, 30);
    let config = {
        headers: { "Content-Type": "multipart/form-data" },
    }; //添加请求头
    axios
```

```
            .post(this.server_url + "/upload", param, config)
            .then((response) => {
              this.percentage = 100;
              clearInterval(timer);
              this.url_1 = response.data.image_url;
              this.srcList.push(this.url_1);
              this.url_2 = response.data.draw_url;
              this.srcList1.push(this.url_2);
              this.fullscreenLoading = false;
              this.loading = false;
              this.feat_list = Object.keys(response.data.image_info);
              for (var i = 0; i < this.feat_list.length; i++) {
                response.data.image_info[this.feat_list[i]][2] = this.feat_list[i];
                this.feature_list.push(response.data.image_info[this.feat_list[i]]);
              }
              this.feature_list.push(response.data.image_info);
              this.feature_list_1 = this.feature_list[0];
              this.dialogTableVisible = false;
              this.percentage = 0;
              this.notice1();
            });
    },
    update1(e) {
      let param = new FormData(); //创建 form 对象
      param.append("address",form.email); //通过 append 向 form 对象添加数据
      let config = {
        headers: { "Content-Type": "multipart/form-data" },
      }; //添加请求头
      axios
            .post(this.server_url + "/email", param, config)
```

```
            .then(response => {
              const responseData = response.data
              console.log(responseData)
            })
        ;
      },
      FilesUploadSuccess(response) {
        console.log(response)
        // 在这里获取后端返回的 JSON 数据
        // 在这里对 responseData 进行处理
        let rs1={
          class:response.result,
          time:response.time,
          con:response.con
        }
        console.log(rs1)
        this.feature_list.push(rs1)
      },
      handleFileChange(file) {
        const reader = new FileReader();
        reader.onload = (e) => {
          this.imageUrl = e.target.result; // 使用 FileReader 读取文件并生成图片链接
        };
        reader.readAsDataURL(file.raw); // 读取文件内容
      },
    },
}
</script>
<template>
  <div class="container">
```

```
<el-card class="card" >
   <div class="upload" style="text-align: center">
   <el-upload
      action="api/upload"
      :on-success="FilesUploadSuccess"
      @change="handleFileChange"
      accept="image/*">
   <el-button type="primary" >图片上传</el-button>
</el-upload>
   </div>
</el-card>
<el-card class="card" style="height: 250px">
   <div class="image">
   <img :src="imageUrl" alt="Selected Image" v-if="imageUrl" class="card-image"/>
</div>
</el-card>
<el-card class="card">
   <div class="table">
   <el-tabs v-model="activeName">
      <el-table
            :data="feature_list"
            border
            style="width: 750px;"
            v-loading="loading"
            element-loading-text="数据正在处理中，请耐心等待"
            element-loading-spinner="el-icon-loading"
            lazy>
         <el-table-column label="目标类别" width="250px" height="50px" prop="class" >
         </el-table-column>
         <el-table-column label="推理时间" width="250px" height="50px" prop="time" >
```

```
            </el-table-column>
            <el-table-column label="置信度" width="250px" height="50px" prop="con" >
            </el-table-column>
          </el-table>
        </el-tabs>
        </div>
    </el-card>
    </div>
</template>
<style scoped>
.container {
    display: flex;
    flex-direction: column;    /*  竖直方向排列  */
}
.card {
    margin-bottom: 20px;    /*  设置卡片之间的间距  */
}
.table {
    display: flex;
    justify-content: center; /*  水平方向居中  */
    align-items: center; /*  垂直方向居中  */
}
.image {
    display: flex;
    justify-content: center; /*  水平方向居中  */
    align-items: center; /*  垂直方向居中  */
}
.card-image{
    height: 15%;
    width: 15%;
```

```
    object-fit: contain;
}
</style>
```

2. 多图片批量检测功能

(1)文件上传：用户通过点击文件上传按钮，调用本地文件上传界面，用户选择待检测图片组成的 zip 包。

(2)填写邮箱：用户上传完压缩包后，界面显示填写邮箱界面，用户输入邮箱，后端将检测结果发送至用户的邮箱。

部分重要代码如下：

```
<script >
import { ref } from 'vue'
import axios from "axios";
import { ElMessage, ElMessageBox } from 'element-plus'
export default {
    data() {
        return {
            server_url: "http://127.0.0.1:5000",
            activeName: "first",
            active: 0,
            centerDialogVisible: true,
            url_1: "",
            url_2: "",
            textarea: "",
            srcList: [],
            srcList1: [],
            feature_list: [],
            feature_list_1: [],
            feat_list: [],
            url: "",
            visible: false,
            wait_return: "等待上传",
            wait_upload: "等待上传",
            loading: false,
            table: false,
            isNav: false,
            showbutton: true,
            percentage: 0,
            fullscreenLoading: false,
            opacitys: {
                opacity: 0,
```

```
        },
        dialogTableVisible: false,
        dialogFormVisible:false,
        form:{},
        is : 0,
        formdata: new FormData(),
        is1: 0,
        rules: {
          email: [
            { required: true, message: '请输入邮箱', trigger: 'blur' },
            { type: 'email', message: '邮箱格式不正确', trigger: ['blur', 'change'] }
          ]
        }
      };
    },
    methods: {
      getObjectURL(file) {
        var url = null;
        if (window.createObjcectURL != undefined) {
          url = window.createOjcectURL(file);
        } else if (window.URL != undefined) {
          url = window.URL.createObjectURL(file);
        } else if (window.webkitURL != undefined) {
          url = window.webkitURL.createObjectURL(file);
        }
        return url;
      },
       update(e) {
         this.is=1;
        this.percentage = 0;
        this.url_1 = "";
        this.url_2 = "";
        this.srcList = [];
        this.srcList1 = [];
        this.wait_return = "";
        this.wait_upload = "";
        this.feature_list = [];
        this.feat_list = [];
        this.fullscreenLoading = true;
        this.loading = true;
        this.showbutton = false;
        let file = e.target.files[0];
        this.url_1 = this.$options.methods.getObjectURL(file);
        //let param = new FormData(); //创建 form 对象
```

```
    //param.append("file", file, file.name); //通过 append 向 form 对象添加数据
    this.formdata.append('files',file);
     this.is1=1;
    //let param =files;
    // var timer = setInterval(() => {
    //     this.myFunc();
    // }, 30);
    let config = {
      headers: { "Content-Type": "multipart/form-data" },
    }; //添加请求头
    axios
        .post(this.server_url + "/upload_zip", this.formdata,)
        .then(response=>{
          if (response.status===200){
            this.is1=1
          }
        });
},
update1(e) {
    this.$refs.form.validate(valid => {
      if (valid) {
        // 表单验证通过，执行提交操作
        let param = new FormData(); //创建 form 对象
        param.append("address",this.form.email); //通过 append 向 form 对象添加数据
        let config = {
          headers: { "Content-Type": "multipart/form-data" },
        }; //添加请求头
        axios
            .post(this.server_url + "/email", param, config)
            .then(response=>{
              ElMessageBox.alert("结果已发送至您的邮箱")
              this.dialogFormVisible=false
            });
        console.log('提交表单');
      } else {
        // 表单验证不通过
        console.log('表单验证失败');
      }
    });
},
httpRequest(){
    if (this.is1===0){
      ElMessageBox.alert("请先选择文件");
      stop();
```

```
        }
        if (this.is1===1){
          this.dialogFormVisible=true;
        }
      },
      uploadsucess(){
          this.is=1
        this.is1=1
      },
      uploaderror(response){
        console.log(response)
      }
    },
}
</script>
<template>
  <el-dialog v-model="dialogFormVisible" title="请输入邮箱:" width="30%" >
    <el-form :model="form" :rules="rules">
      <el-form-item label="邮箱地址:" :label-width="formLabelWidth" prop="email">
        <el-input v-model="form.email" autocomplete="off" />
      </el-form-item>
    </el-form>
    <el-button @click="update1">确定</el-button>
  </el-dialog>
  <div class="container">
    <el-card class="card">
      <div class="card-footer" style="width: 100%;">  <el-upload
          action="api/upload_zip"
          :on-success="uploadsucess"
          :on-error="uploaderror"
          accept=".zip">
        <div style="text-align: center;width: 200px" >
          <el-button  type="primary"  >文件上传</el-button>
        </div>
      </el-upload>
      </div>
    </el-card>
    <el-card class="card">
      <div class="card-footer">
        <el-button @click="httpRequest()" v-if="this.is===1">输入邮箱</el-button>
      </div>
    </el-card>
  </div>
</template>
```

```
<style scoped>
.container {
  display: flex;
  flex-direction: column;  /* 竖直方向排列 */
  width: 100%;
  justify-content: center;
}
.card {
  margin-bottom: 20px;  /* 设置卡片之间的间距 */
  height: 150px;
}
.card .card-footer {
  display: flex;
  justify-content: center;
  width: 100%;
}
</style>
```

3.2.2 微信小程序端功能实现

1. 首页

(1)展示界面：用户无需授权即可进入

部分重要代码如下：

```
<view>
<image class="img" src="../images/R-C.jpg"></image>
<view class="t1">工业品组装缺陷检测中心</view>
<button class="btn1" bindtap="index">开始检测</button>
</view>
/**home.wxss**/
.btn1{
  position: relative;
  top:700rpx;
  background-color: rgb(236, 108, 187);
  color: rgb(255, 255, 255);
  font-family: 楷体;
  font-size: 40rpx;
  margin: 50rpx;

  border-radius: 40rpx;
}
.btn1:after{
  color: blue;
}
```

```
.t1{
    position: relative;
    top: 200rpx;
    text-align: center;
    font-size: 60rpx;
    line-height:2;
    opacity: 0.8;
    color:white;
    font-family: 隶书;
}
.img{
    position: absolute;
    top:0rpx;
    left:0rpx;
    width:100%;
    height:100%;
}
// pages/home/home.js
Page({
  data: {
  },
  onLoad: function (options) {
    //查看是否授权
    wx.getSetting({
      success: function(res) {
        if (res.authSetting['scope.userInfo']) {
          console.log("用户授权了");
        } else {
          //用户没有授权
          console.log("用户没有授权");
        }
      }
    });
},
  index:function(){
    wx.getUserProfile({
      desc: '用于微信账号与平台账号绑定',
      success: (res)=>{
        wx.navigateTo({
          url: '/pages/index/index',
        })
        console.log("获取到的用户信息成功: ",JSON.stringify(res));
        this.setData({
          userInfo: res,
```

```
        userInfoStr: JSON.stringify(res)
      })
    },
    fail: (res)=>{
      console.log("获取用户个人信息失败: ",res);
      //用户按了拒绝按钮
          wx.showModal({
            title: '警告',
            content: '您点击了拒绝授权，将无法进入小程序，请授权之后再进入!!!',
            showCancel: false,
            confirmText: '返回授权',
            success: function(res) {
              if (res.confirm) {
                console.log('用户点击了“返回授权”');
              }
            }
      });
    }
})
})
```

2. 检测服务界面

(1)检测服务界面：用户可以选择拍照或者本地图片上传来使用检测服务。

部分重要代码如下：

```
/**index.wxss**/
.userinfo {
  display: flex;
  flex-direction: column;
  align-items: center;
  color: #aaa;
}
.userinfo-avatar {
  overflow: hidden;
  width: 128rpx;
  height: 128rpx;
  margin: 20rpx;
  border-radius: 50%;
}

.usermotto {
  margin-top: 200px;
}
.demo{
  display:flex;
```

```
  align-items:center;
  height: 100px;
  justify-content:center;
  font-size: 13px;
}
.img{
  position: absolute;
  top:0rpx;
  left:0rpx;
  width:100%;
  height:100%;
}
.demo1{
  display:flex;
  align-items:center;
  height: 36px;
  justify-content:center;
  font-size: 18px;
  font-size: 50rpx;
  font-family: 楷体;
}
/* 上传图片 */
.load-name {
  height: 80rpx;
  line-height: 80rpx;
  font-size: 30rpx;
}
.load-box {
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
}
.img-item, .img-add {
  position: relative;
  width: 100rpx;
  height: 100rpx;
  margin: 20rpx;
}
.b1{

    width: 715rpx;
    height: 140rpx;
    margin: 20rpx;
    align-items: center;
```

```
    justify-content:center;
    display:flex;
    font-family: 楷体;
    font-size: 50rpx;
}
.img-add {
  position: relative;
  height: 250rpx;
  width: 250rpx;
  left: 31%;
  background-color: rgb(186, 217, 245);
  border: 1px solid rgb(255, 255, 255);
}
.img-add:after{
  width: 10rpx;
  height: 150rpx;
  content: " ";
  position: absolute;
  top: 50%;
  left: 50%;
  -webkit-transform: translate(-50%, -50%);
  -ms-transform: translate(-50%, -50%);
  transform: translate(-50%, -50%);
  background-color: rgb(228, 247, 250);
}
.detect-btn{
    float: center;
    width: 300px;
    height: 40px;
    margin-top: 10px;
    margin-left: 3px;
    margin-bottom: 0rpx;
    background-color: #9cd9f5;

}
.img-add:before{
  position: absolute;
  top: 50%;
  right: 20%;
  width: 150rpx;
  height: 10rpx;
  content: " ";
  display: inline-block;
  background-color: rgb(228, 247, 250);
```

```
}
.img-item {
  margin-right: 20rpx;
}
.img-item image {
  width: 100%;
  height: 100%;
  border-radius: 10rpx;
}
.icon {
  position: absolute;
  top: 0;
  right: 0;
}
<!--index.wxml-->
<!-- <image class="img"  src="../images/OIP-C.jpg" /> -->
<view>
<view class="b1">
<text decode="{{true}}" >请上传待检测图片</text>
</view>
<view class='load-img'>
   <view class='load-box'>
     <view class='img-item' wx:for="{{fileList}}" wx:key="index" >
       <image src="{{item.path}}" data-src="{{item}}" mode="aspectFill" data-list="{{fileList}}"
 bindtap=""></image>
       <icon class='icon' type="clear" size="50" color='#EF5631' catchtap='_onDelTab' data-idx="{
{index}}" wx:if="{{!prevent}}"/>
     </view>
     <image class='img-add' bindtap='chooseimage_cloud' wx:if="{{!prevent}}"></image>
   </view>
 </view>
<!-- <button bindtap="chooseimage_cloud" style="width :80%">添加图片</button>   -->
<view class="demo1">
<text decode="{{true}}">图片预览</text>
</view>
<image src="{{tempFilePaths }}" mode="aspectFit" style="width: 100%; height: 380rpx"/>
<view class="demo1">
<text decode="{{true}}">分类结果:{{listResult}}</text>
</view>
<view class="demo1">
<text decode="{{true}}">预测时间:{{listTime}}</text>
</view>
<view class="demo1">
<text decode="{{true}}">置信度:{{listCon}}</text>
```

```
</view>
</view>
<!-- <button class="detect-btn" data-index="{{index}}" bindtap="detect">开始检测</button> -->
<!-- <button bindtap="postInfo">POST 方法测试</button> -->
<!-- <view class="demo">
<text decode="{{true}}"> 检 测 结 果 :{{result}}    预 处 理 时 间:{{pre_time}}ms   推理时间:{{detect_time}}ms</text>
</view> -->
<!-- <view class="demo1">
<text decode="{{true}}">标注后的图片:</text>
</view> -->
<!--
<image src="https://t54897w513.goho.co/image" mode="aspectFit" style="width: 100%; height: 450rpx"/>  -->
<!-- <image src="{{cloudPath}}" mode="aspectFit" style="width: 100%; height: 450rpx"/> -->
// const { VertexBuffer } = require("XrFrame/kanata/lib/index");
var app = getApp()
Page({
 data: {  //下面的数据是设置全局变量
   tempFilePaths: '',
   result:'',
   pre_time:'',
   detect_time:'',
   cloudPath:'',
   listResult:'',

 },
 onLoad:function(options){
    this.chooseimage_cloud
 },
 chooseimage_cloud: function () {
  var fileid_id = ''
  var _this = this;
  wx.chooseMedia({
    count: 1, //指定最多上传图片的大小
    mediaType: ['image'],
    sourceType: ['album', 'camera'], // 可以指定来源是相册还是相机，默认二者都有
    sizeType: ['original'],
    camera: 'back',
    success: function (res) {
      const tempFilePaths = res.tempFiles
      console.log('==================tempFiles = ',tempFilePaths)
      const filePath = String(res.tempFiles[0].tempFilePath)
      const cloudPath = 'WX.image'
```

```
_this.setData({
  //这里设置在前端展示的内容,第一个是是否有缺陷,第二个是处理速度
  tempFilePaths:res.tempFiles[0].tempFilePath
})
wx.showLoading({
  title: '正在上传请稍后',
  mask:true
})
wx.uploadFile({
  filePath: _this.data.tempFilePaths,
  name: 'file',
  // 服务器地址等后端部署到服务器上之后需要更改
  url: 'http:///192.168.43.202:5000/upload',//服务器地址
  header:{'Content-Type':'application/json'},
  success: (res2) => {   //成功之后执行的方法

     console.log(res2.data)
     var obj=JSON.parse(res2.data);
     var result=obj.result;
     var time=obj.time;
     var con=obj.con;
     _this.setData({
       listResult:result,
       listTime:time,
       listCon:con
       })
     wx.hideLoading({ //隐藏加载框
     }).then(res2 => {
       wx.showToast({ //提示框
         title: '上传成功',
         duration:2000,
         success:function(){}
       }) })
},
  fail: res => {
  wx.hideLoading({ //隐藏加载框
  }).then(res => {
    wx.showModal({
      title: '请求失败',
      content: "请检查服务器连接"
    })
  })
}
  })
```

```
    }
   })
 },
})
```

3.2.3 后端功能及算法实现

1. 服务启动程序：开启后端，提供检测服务。

部分重要代码如下：

```
import zipfile
import cv2
from flask import *
import tensorflow as tf
from flask_mail import Mail, Message
from pasta.base.annotate import expression
import numpy as np
import os
from flask_cors import CORS, cross_origin
# classes_name_list=['anomaly', 'good', 'one', 'neighbor_two', 'diagonal_two', 'three', 'four']
classes_name_list=['图片缺角（建议更换图片角度）', '合格', '不合格（缺一角螺丝）', '不合格（缺临边两螺丝）', '不合格（缺对角两螺丝）', '不合格（缺三角螺丝）', '不合格（缺四角螺丝）']
UPLOAD_PATH = r'./uploads'
model = tf.keras.models.load_model(r'C:\Users\LiYanLin\Desktop\defect_product_backEnd\model\VGG19_ft.h5')
# 1. 初始化 flask app
app = Flask(__name__)
app.config['MAIL_DEBUG'] = False                # 开启 debug，便于调试看信息
app.config['MAIL_SUPPRESS_SEND'] = False        # 发送邮件，为 True 则不发送
app.config['MAIL_SERVER'] = 'smtp.qq.com'       # 邮箱服务器
app.config['MAIL_PORT'] = 465                   # 端口
app.config['MAIL_USE_SSL'] = True               # 重要，qq 邮箱需要使用 SSL
app.config['MAIL_USE_TLS'] = False              # 不需要使用 TLS
app.config['MAIL_USERNAME'] = '516680918@qq.com'   # 填邮箱
app.config['MAIL_PASSWORD'] = '          '      # 填授权码（此处为了保密而删去授权码）
app.config['MAIL_DEFAULT_SENDER'] = '516680918@qq.com'   # 填邮箱，默认发送者
mail = Mail(app)
# 2. 解决跨域
@app.after_request
def after_request(response):
    response.headers['Access-Control-Allow-Origin'] = '*'
    response.headers['Access-Control-Allow-Credentials'] = 'true'
```

```
    response.headers['Access-Control-Allow-Methods'] = 'POST'
    response.headers['Access-Control-Allow-Headers'] = 'Content-Type, X-Requested-With'
    return response
# 3. 主页面
@app.route('/', methods=['GET', 'POST'])
def home():
    return redirect(url_for('static', filename='./index.html'))
# 4. 预测函数
def predict(path):
    import time
    img = cv2.imread(path)
    img = cv2.resize(img, (224, 224))
    img_tensor = tf.convert_to_tensor(img, dtype=tf.float16)
    img_tensor /= 255.0
    test_img = np.expand_dims(img_tensor, 0)
    begin_time = time.time()
    out = model.predict(test_img)
    end_time = time.time()
    t = end_time - begin_time
    time_result = str(round(t, 6)) + 's'
    con = str(round(out[0][out.argmax()], 5))
    if round(out[0][out.argmax()], 5) > 0.975:
        return classes_name_list[out.argmax()], time_result, con
    else:
        return '该图片中无零件', time_result, con
# 5. 上传待检测图片
@app.route('/upload', methods=['POST'])
def upload():
    ## 接收前端发送的待检测图片
    file = request.files['file']
    ## 获取图片名
    file_name = file.filename
    ## 图片保存路径
    file_path = UPLOAD_PATH + '/' + file_name
    ## 图片保存
    file.save(file_path)
    ## 调用预测
    result, time, con = predict(file_path)
    ## 返回分类结果和时间
    return jsonify({
        'result': result,
        'time': time,
        'con': con
    })
```

```
# 6. 接受邮箱并发送结果
@app.route('/email', methods=['POST'])
def email():
    address = request.form.get('address')
    msg = Message(
        subject="Predict Results",
        recipients=[address]
    )
    msg.body = "Please check the test results of the industrial products you submitted."
    with app.open_resource("results.txt") as fp:
        msg.attach("results.txt", "file/txt", fp.read())
    try:
        mail.send(msg)
        return "发送成功"
    except expression(BaseException) as e:
        print(e)
        return "发送失败"
# 7. 接受前端传来的 zip 压缩包
@app.route('/upload_zip', methods=['POST'])
def upload_zip():
    file = request.files['file']
    file_name = file.filename
    file_path = UPLOAD_PATH + '/' + file_name
    file.save(file_path)
    zip_files = zipfile.ZipFile(file_path)
    try:
        zip_files.extractall(UPLOAD_PATH)
        names = file_name.split('.')
        name = names[0]
        with open('results.txt', 'w', encoding='utf-8') as f:
            for root, dirs, files in os.walk(UPLOAD_PATH + '/' + name):
                for file in files:
                    path = os.path.join(root, file)
                    path = path.replace('\\', '/')
                    result, time, con = predict(path)
                    f.writelines(file + ' ' + result + ' ' + time + ' ' + con + '\n')


        return "上传并解压成功"
    except expression(BaseException) as e:
        print(e)
        return "解压失败"
if __name__ == '__main__':
    CORS(app, supports_credentials=True)
```

```
    app.run(host='0.0.0.0', port=5000)
```

2. 数据增强：对原数据集进行数据增强。

部分重要代码如下：

```
import os
import random
import imgaug.augmenters as iaa
import imageio.v2 as imageio
def get_dataset():
    print("------------开始制作数据集------------")
    # 之前的扩充方法是，每张图像只使用了一种扩充方法，但实际上完全可以先旋转各种角度然后再加噪声
    # 测试集里最好不加噪声，只是旋转
    ## 原图系列
    #### 原图水平翻转 180
    seq1 = iaa.Sequential([
        iaa.Fliplr(1),
    ])
    #### 原图垂直翻转 180
    seq2 = iaa.Sequential([
        iaa.Flipud(1),
    ])
    #### 原图旋转 90
    seq3 = iaa.Sequential([
        iaa.Affine(rotate=90)
    ])
    #### 原图旋转 270
    seq4 = iaa.Sequential([
        iaa.Affine(rotate=270)
    ])
    #### 原图-180 到 180 之间旋转随机角度
```

```
    seq5 = iaa.Sequential([
        iaa.Affine(rotate=(-180, 180))
    ])
    #### 原图高斯噪声
    seq6 = iaa.Sequential([
        iaa.AdditiveGaussianNoise(scale=(0, 0.05 * 255))
    ])
    #### 原图散粒噪声
    seq7 = iaa.Sequential([
        iaa.imgcorruptlike.ShotNoise(severity=2)
    ])
    #### 原图斑点噪声
    seq8 = iaa.Sequential([
        iaa.imgcorruptlike.SpeckleNoise(severity=2)
    ])
    #### 原图像变暗
    seq9 = iaa.Sequential([
        iaa.Multiply(mul=(0.75), per_channel=False, name=None,    deterministic="deprecated",
random_state=None)
    ])
    #### 原图像变亮
    seq10 = iaa.Sequential([
        iaa.Multiply(mul=(1.25), per_channel=False, name=None,    deterministic="deprecated",
random_state=None)
    ])
    #### 原图锐化
    seq11 = iaa.Sequential([
        iaa.Sharpen(alpha=(0, 1.0), lightness=(0.75, 1.5))
    ])
    ## 水平翻转 180 系列
```

```
    #### 水平 180 + 高斯噪声
    seq12 = iaa.Sequential([
        iaa.Fliplr(1),
        iaa.AdditiveGaussianNoise(scale=(0, 0.05 * 255))
    ])
    #### 水平 180 + 散粒噪声
    seq13 = iaa.Sequential([
        iaa.Fliplr(1),
        iaa.imgcorruptlike.ShotNoise(severity=2)
    ])
    #### 水平 180 + 斑点噪声
    seq14 = iaa.Sequential([
        iaa.Fliplr(1),
        iaa.imgcorruptlike.SpeckleNoise(severity=2)
    ])
    #### 水平 180 + 图像变暗
    seq15 = iaa.Sequential([
        iaa.Fliplr(1),
        iaa.Multiply(mul=(0.75), per_channel=False, name=None,  deterministic="deprecated",
random_state=None)
    ])
    #### 水平 180 + 图像变亮
    seq16 = iaa.Sequential([
        iaa.Fliplr(1),
        iaa.Multiply(mul=(1.25), per_channel=False, name=None,  deterministic="deprecated",
random_state=None)
    ])
    #### 水平 180 + 图像锐化
    seq17 = iaa.Sequential([
        iaa.Fliplr(1),
```

```
        iaa.Sharpen(alpha=(0, 1.0), lightness=(0.75, 1.5))
    ])
    ## 垂直翻转 180 系列
    #### 垂直 180 + 高斯噪声
    seq18 = iaa.Sequential([
        iaa.Flipud(1),
        iaa.AdditiveGaussianNoise(scale=(0, 0.05 * 255))
    ])
    #### 垂直 180 + 散粒噪声
    seq19 = iaa.Sequential([
        iaa.Flipud(1),
        iaa.imgcorruptlike.ShotNoise(severity=2)
    ])
    #### 垂直 180 + 斑点噪声
    seq20 = iaa.Sequential([
        iaa.Flipud(1),
        iaa.imgcorruptlike.SpeckleNoise(severity=2)
    ])
    #### 垂直 180 + 图像变暗
    seq21 = iaa.Sequential([
        iaa.Flipud(1),
        iaa.Multiply(mul=(0.75), per_channel=False, name=None,    deterministic="deprecated",
random_state=None)
    ])
    #### 垂直 180 + 图像变亮
    seq22 = iaa.Sequential([
        iaa.Flipud(1),
        iaa.Multiply(mul=(1.25), per_channel=False, name=None,    deterministic="deprecated",
random_state=None)
    ])
```

```
#### 垂直 180 + 图像锐化
seq23 = iaa.Sequential([
    iaa.Flipud(1),
    iaa.Sharpen(alpha=(0, 1.0), lightness=(0.75, 1.5))
])
## 旋转 90 系列
#### 90 + 高斯噪声
seq24 = iaa.Sequential([
    iaa.Affine(rotate=90),
    iaa.AdditiveGaussianNoise(scale=(0, 0.05 * 255))
])
#### 90 + 散粒噪声
seq36 = iaa.Sequential([
    iaa.Affine(rotate=90),
    iaa.imgcorruptlike.ShotNoise(severity=2)
])
#### 90 + 斑点噪声
seq25 = iaa.Sequential([
    iaa.Affine(rotate=90),
    iaa.imgcorruptlike.SpeckleNoise(severity=2)
])
#### 90 + 图像变暗
seq26 = iaa.Sequential([
    iaa.Affine(rotate=90),
    iaa.Multiply(mul=(0.5), per_channel=False, name=None,    deterministic="deprecated",
random_state=None)
])
#### 90 + 图像变亮
seq27 = iaa.Sequential([
    iaa.Affine(rotate=90),
```

```
        iaa.Multiply(mul=(1.5), per_channel=False, name=None,    deterministic="deprecated",
random_state=None)
    ])
    #### 90 + 图像锐化
    seq28 = iaa.Sequential([
        iaa.Affine(rotate=90),
        iaa.Sharpen(alpha=(0, 1.0), lightness=(0.75, 1.5))
    ])
    ## 旋转 270 系列
    #### 270 + 高斯噪声
    seq29 = iaa.Sequential([
        iaa.Affine(rotate=270),
        iaa.AdditiveGaussianNoise(scale=(0, 0.05 * 255))
    ])
    #### 270 + 散粒噪声
    seq30 = iaa.Sequential([
        iaa.Affine(rotate=270),
        iaa.imgcorruptlike.ShotNoise(severity=2)
    ])
    #### 270 + 斑点噪声
    seq31 = iaa.Sequential([
        iaa.Affine(rotate=270),
        iaa.imgcorruptlike.SpeckleNoise(severity=2)
    ])
    #### 270 + 图像变暗
    seq32 = iaa.Sequential([
        iaa.Affine(rotate=270),
        iaa.Multiply(mul=(0.75), per_channel=False, name=None,    deterministic="deprecated",
random_state=None)
    ])
```

```
    #### 270 + 图像变亮
    seq33 = iaa.Sequential([
        iaa.Affine(rotate=270),
        iaa.Multiply(mul=(1.25), per_channel=False, name=None,    deterministic="deprecated",
random_state=None)
    ])
    #### 270 + 图像锐化
    seq34 = iaa.Sequential([
        iaa.Affine(rotate=270),
        iaa.Sharpen(alpha=(0, 1.0), lightness=(0.75, 1.5))
    ])
    ## 旋转 180 到 180 随机角度系列
    #### 随机 + 高斯噪声
    seq35 = iaa.Sequential([
        iaa.Affine(rotate=(180, 180)),
        iaa.AdditiveGaussianNoise(scale=(0, 0.05 * 255))
    ])
    #### 随机 + 散粒噪声
    seq36 = iaa.Sequential([
        iaa.Affine(rotate=(180, 180)),
        iaa.imgcorruptlike.ShotNoise(severity=2)
    ])
    #### 随机 + 斑点噪声
    seq37 = iaa.Sequential([
        iaa.Affine(rotate=(180, 180)),
        iaa.imgcorruptlike.SpeckleNoise(severity=2)
    ])
    #### 随机 + 图像变暗
    seq38 = iaa.Sequential([
        iaa.Affine(rotate=(180, 180)),
```

```
        iaa.Multiply(mul=(0.75), per_channel=False, name=None,    deterministic="deprecated",
random_state=None)
    ])
    #### 随机 + 图像变亮
    seq39 = iaa.Sequential([
        iaa.Affine(rotate=(180, 180)),
        iaa.Multiply(mul=(1.25), per_channel=False, name=None,    deterministic="deprecated",
random_state=None)
    ])
    #### 随机 + 图像锐化
    seq40 = iaa.Sequential([
        iaa.Affine(rotate=(180, 180)),
        iaa.Sharpen(alpha=(0, 1.0), lightness=(0.75, 1.5))
    ])
    ## dataset 路径
    dir = '/home/sduu2/userspace/lyl/product_detect/defect_product_dataset/multi_classification/'
    ## aug 之后的数据集路径
    aug_train_dir = '/home/sduu2/userspace/lyl/product_detect/dataset_aug/train/'
    aug_test_dir = '/home/sduu2/userspace/lyl/product_detect/dataset_aug/test/'
    ##  在 train 和 test 下面创建每个类别的文件夹
    os.makedirs(aug_train_dir + 'anomaly')
    os.makedirs(aug_test_dir + 'anomaly')
    os.makedirs(aug_train_dir + 'good')
    os.makedirs(aug_test_dir + 'good')
    os.makedirs(aug_train_dir + 'one')
    os.makedirs(aug_test_dir + 'one')
    os.makedirs(aug_train_dir + 'neighbor_two')
    os.makedirs(aug_test_dir + 'neighbor_two')
    os.makedirs(aug_train_dir + 'diagonal_two')
    os.makedirs(aug_test_dir + 'diagonal_two')
```

```
os.makedirs(aug_train_dir + 'three')
os.makedirs(aug_test_dir + 'three')
os.makedirs(aug_train_dir + 'four')
os.makedirs(aug_test_dir + 'four')
## 遍历每个类别文件夹
folderNames = os.listdir(dir)
for folderName in folderNames:
    print(folderName)
    folderPath = dir + folderName + '/'
    files = os.listdir(folderPath)
    files.sort()
    #### 该类别下的数据数量
    size = len(files)
    #### 确定测试集数据数量，剩余即为训练集
    test_size= size//10
    test_size *= 2
    #### 确定测试集的 index
    test_index = random.sample(range(0, size), test_size)
    #### 根据 index，划分训练和测试
    for i in range(size):
        fileName = files[i]
        filePath = folderPath + fileName
        img = imageio.imread(filePath)
        names = str.split(fileName, '.')
        name = names[0]
        if i in test_index:
            ## 原图
            imageio.imwrite(aug_test_dir + folderName + '/' + name + "_0.JPG", img)
            [img_seq1,]=seq1(images=[img])
            imageio.imwrite(aug_test_dir + folderName + '/' + name + "_1.JPG",
```

```
img_seq1)
                [img_seq2,]=seq2(images=[img])
                imageio.imwrite(aug_test_dir     + folderName + '/' + name + "_2.JPG",
img_seq2)
                [img_seq3,]=seq3(images=[img])
                imageio.imwrite(aug_test_dir     + folderName + '/' + name + "_3.JPG",
img_seq3)
                [img_seq4,]=seq4(images=[img])
                imageio.imwrite(aug_test_dir     + folderName + '/' + name + "_4.JPG",
img_seq4)
                [img_seq5,]=seq5(images=[img])
                imageio.imwrite(aug_test_dir     + folderName + '/' + name + "_5.JPG",
img_seq5)
                # [img_seq6,]=seq6(images=[img])
                #  imageio.imwrite(aug_test_dir     + folderName + '/' + name + "_6.JPG",
img_seq6)
                [img_seq7,]=seq7(images=[img])
                imageio.imwrite(aug_test_dir     + folderName + '/' + name + "_7.JPG",
img_seq7)
                [img_seq8,]=seq8(images=[img])
                imageio.imwrite(aug_test_dir     + folderName + '/' + name + "_8.JPG",
img_seq8)
                [img_seq9,]=seq9(images=[img])
                imageio.imwrite(aug_test_dir     + folderName + '/' + name + "_9.JPG",
img_seq9)
                [img_seq10,]=seq10(images=[img])
                imageio.imwrite(aug_test_dir     + folderName + '/' + name + "_10.JPG",
img_seq10)

                # [img_seq11,]=seq11(images=[img])
```

```
                # imageio.imwrite(aug_test_dir    + folderName + '/' + name + "_11.JPG",
img_seq11)
                # [img_seq12,]=seq12(images=[img])
                # imageio.imwrite(aug_test_dir    + folderName + '/' + name + "_12.JPG",
img_seq12)
                # [img_seq13,]=seq13(images=[img])
                # imageio.imwrite(aug_test_dir    + folderName + '/' + name + "_13.JPG",
img_seq13)
                # [img_seq14,]=seq14(images=[img])
                # imageio.imwrite(aug_test_dir    + folderName + '/' + name + "_14.JPG",
img_seq14)
                # [img_seq15,]=seq15(images=[img])
                # imageio.imwrite(aug_test_dir    + folderName + '/' + name + "_15.JPG",
img_seq15)
                # [img_seq16,]=seq16(images=[img])
                # imageio.imwrite(aug_test_dir    + folderName + '/' + name + "_16.JPG",
img_seq16)
                # [img_seq17,]=seq17(images=[img])
                # imageio.imwrite(aug_test_dir    + folderName + '/' + name + "_17.JPG",
img_seq17)
                # [img_seq18,]=seq18(images=[img])
                # imageio.imwrite(aug_test_dir    + folderName + '/' + name + "_18.JPG",
img_seq18)
                # [img_seq19,]=seq19(images=[img])
                # imageio.imwrite(aug_test_dir    + folderName + '/' + name + "_19.JPG",
img_seq19)
                # [img_seq20,]=seq20(images=[img])
                # imageio.imwrite(aug_test_dir    + folderName + '/' + name + "_20.JPG",
img_seq20)
                # [img_seq21,]=seq21(images=[img])
```

```
                # imageio.imwrite(aug_test_dir    + folderName + '/' + name + "_21.JPG",
img_seq21)
                # [img_seq22,]=seq22(images=[img])
                # imageio.imwrite(aug_test_dir    + folderName + '/' + name + "_22.JPG",
img_seq22)
                # [img_seq23,]=seq23(images=[img])
                # imageio.imwrite(aug_test_dir    + folderName + '/' + name + "_23.JPG",
img_seq23)
                # [img_seq24,]=seq24(images=[img])
                # imageio.imwrite(aug_test_dir    + folderName + '/' + name + "_24.JPG",
img_seq24)
                # [img_seq25,]=seq25(images=[img])
                # imageio.imwrite(aug_test_dir    + folderName + '/' + name + "_25.JPG",
img_seq25)
                # [img_seq26,]=seq26(images=[img])
                # imageio.imwrite(aug_test_dir    + folderName + '/' + name + "_26.JPG",
img_seq26)
                # [img_seq27,]=seq27(images=[img])
                # imageio.imwrite(aug_test_dir    + folderName + '/' + name + "_27.JPG",
img_seq27)
                # [img_seq28,]=seq28(images=[img])
                # imageio.imwrite(aug_test_dir    + folderName + '/' + name + "_28.JPG",
img_seq28)
                # [img_seq29,]=seq29(images=[img])
                # imageio.imwrite(aug_test_dir    + folderName + '/' + name + "_29.JPG",
img_seq29)
                # [img_seq30,]=seq30(images=[img])
                # imageio.imwrite(aug_test_dir    + folderName + '/' + name + "_30.JPG",
img_seq30)
                # [img_seq31,]=seq31(images=[img])
```

```
                # imageio.imwrite(aug_test_dir    + folderName + '/' + name + "_31.JPG",
img_seq31)
                # [img_seq32,]=seq32(images=[img])
                # imageio.imwrite(aug_test_dir    + folderName + '/' + name + "_32.JPG",
img_seq32)
                # [img_seq33,]=seq33(images=[img])
                # imageio.imwrite(aug_test_dir    + folderName + '/' + name + "_33.JPG",
img_seq33)
                # [img_seq34,]=seq34(images=[img])
                # imageio.imwrite(aug_test_dir    + folderName + '/' + name + "_34.JPG",
img_seq34)
                # [img_seq35,]=seq35(images=[img])
                # imageio.imwrite(aug_test_dir    + folderName + '/' + name + "_35.JPG",
img_seq35)
                # [img_seq36,]=seq36(images=[img])
                # imageio.imwrite(aug_test_dir    + folderName + '/' + name + "_36.JPG",
img_seq36)
                # [img_seq37,]=seq37(images=[img])
                # imageio.imwrite(aug_test_dir    + folderName + '/' + name + "_37.JPG",
img_seq37)
                # [img_seq38,]=seq38(images=[img])
                # imageio.imwrite(aug_test_dir    + folderName + '/' + name + "_38.JPG",
img_seq38)
                # [img_seq39,]=seq39(images=[img])
                # imageio.imwrite(aug_test_dir    + folderName + '/' + name + "_39.JPG",
img_seq39)
                # [img_seq40,]=seq40(images=[img])
                # imageio.imwrite(aug_test_dir    + folderName + '/' + name + "_40.JPG",
img_seq40)
            else:
```

```
                ## 扩充训练集
                imageio.imwrite(aug_train_dir    + folderName + '/' + name + "_0.JPG", img)
                [img_seq1,]=seq1(images=[img])
                imageio.imwrite(aug_train_dir     + folderName + '/' + name + "_1.JPG",
img_seq1)
                [img_seq2,]=seq2(images=[img])
                imageio.imwrite(aug_train_dir     + folderName + '/' + name + "_2.JPG",
img_seq2)
                [img_seq3,]=seq3(images=[img])
                imageio.imwrite(aug_train_dir     + folderName + '/' + name + "_3.JPG",
img_seq3)
                [img_seq4,]=seq4(images=[img])
                imageio.imwrite(aug_train_dir     + folderName + '/' + name + "_4.JPG",
img_seq4)
                [img_seq5,]=seq5(images=[img])
                imageio.imwrite(aug_train_dir     + folderName + '/' + name + "_5.JPG",
img_seq5)
                # [img_seq6,]=seq6(images=[img])
                #  imageio.imwrite(aug_train_dir     + folderName + '/' + name + "_6.JPG",
img_seq6)
                [img_seq7,]=seq7(images=[img])
                imageio.imwrite(aug_train_dir     + folderName + '/' + name + "_7.JPG",
img_seq7)
                [img_seq8,]=seq8(images=[img])
                imageio.imwrite(aug_train_dir     + folderName + '/' + name + "_8.JPG",
img_seq8)
                [img_seq9,]=seq9(images=[img])
                imageio.imwrite(aug_train_dir     + folderName + '/' + name + "_9.JPG",
img_seq9)
                [img_seq10,]=seq10(images=[img])
```

```
                imageio.imwrite(aug_train_dir    + folderName + '/' + name + "_10.JPG",
img_seq10)

                # [img_seq11,]=seq11(images=[img])
                # imageio.imwrite(aug_train_dir    + folderName + '/' + name + "_11.JPG",
img_seq11)
                # [img_seq12,]=seq12(images=[img])
                # imageio.imwrite(aug_train_dir    + folderName + '/' + name + "_12.JPG",
img_seq12)
                # [img_seq13,]=seq13(images=[img])
                # imageio.imwrite(aug_train_dir    + folderName + '/' + name + "_13.JPG",
img_seq13)
                # [img_seq14,]=seq14(images=[img])
                # imageio.imwrite(aug_train_dir    + folderName + '/' + name + "_14.JPG",
img_seq14)
                # [img_seq15,]=seq15(images=[img])
                # imageio.imwrite(aug_train_dir    + folderName + '/' + name + "_15.JPG",
img_seq15)
                # [img_seq16,]=seq16(images=[img])
                # imageio.imwrite(aug_train_dir    + folderName + '/' + name + "_16.JPG",
img_seq16)
                # [img_seq17,]=seq17(images=[img])
                # imageio.imwrite(aug_train_dir    + folderName + '/' + name + "_17.JPG",
img_seq17)
                # [img_seq18,]=seq18(images=[img])
                # imageio.imwrite(aug_train_dir    + folderName + '/' + name + "_18.JPG",
img_seq18)
                # [img_seq19,]=seq19(images=[img])
                # imageio.imwrite(aug_train_dir    + folderName + '/' + name + "_19.JPG",
img_seq19)
```

```
                # [img_seq20,]=seq20(images=[img])
                # imageio.imwrite(aug_train_dir    + folderName + '/' + name + "_20.JPG",
img_seq20)
                # [img_seq21,]=seq21(images=[img])
                # imageio.imwrite(aug_train_dir    + folderName + '/' + name + "_21.JPG",
img_seq21)
                # [img_seq22,]=seq22(images=[img])
                # imageio.imwrite(aug_train_dir    + folderName + '/' + name + "_22.JPG",
img_seq22)
                # [img_seq23,]=seq23(images=[img])
                # imageio.imwrite(aug_train_dir    + folderName + '/' + name + "_23.JPG",
img_seq23)
                # [img_seq24,]=seq24(images=[img])
                # imageio.imwrite(aug_train_dir    + folderName + '/' + name + "_24.JPG",
img_seq24)
                # [img_seq25,]=seq25(images=[img])
                # imageio.imwrite(aug_train_dir    + folderName + '/' + name + "_25.JPG",
img_seq25)
                # [img_seq26,]=seq26(images=[img])
                # imageio.imwrite(aug_train_dir    + folderName + '/' + name + "_26.JPG",
img_seq26)
                # [img_seq27,]=seq27(images=[img])
                # imageio.imwrite(aug_train_dir    + folderName + '/' + name + "_27.JPG",
img_seq27)
                # [img_seq28,]=seq28(images=[img])
                # imageio.imwrite(aug_train_dir    + folderName + '/' + name + "_28.JPG",
img_seq28)
                # [img_seq29,]=seq29(images=[img])
                # imageio.imwrite(aug_train_dir    + folderName + '/' + name + "_29.JPG",
img_seq29)
```

```
                # [img_seq30,]=seq30(images=[img])
                # imageio.imwrite(aug_train_dir    + folderName + '/' + name + "_30.JPG",
img_seq30)
                # [img_seq31,]=seq31(images=[img])
                # imageio.imwrite(aug_train_dir    + folderName + '/' + name + "_31.JPG",
img_seq31)
                # [img_seq32,]=seq32(images=[img])
                # imageio.imwrite(aug_train_dir    + folderName + '/' + name + "_32.JPG",
img_seq32)
                # [img_seq33,]=seq33(images=[img])
                # imageio.imwrite(aug_train_dir    + folderName + '/' + name + "_33.JPG",
img_seq33)
                # [img_seq34,]=seq34(images=[img])
                # imageio.imwrite(aug_train_dir    + folderName + '/' + name + "_34.JPG",
img_seq34)
                # [img_seq35,]=seq35(images=[img])
                # imageio.imwrite(aug_train_dir    + folderName + '/' + name + "_35.JPG",
img_seq35)
                # [img_seq36,]=seq36(images=[img])
                # imageio.imwrite(aug_train_dir    + folderName + '/' + name + "_36.JPG",
img_seq36)
                # [img_seq37,]=seq37(images=[img])
                # imageio.imwrite(aug_train_dir    + folderName + '/' + name + "_37.JPG",
img_seq37)
                # [img_seq38,]=seq38(images=[img])
                # imageio.imwrite(aug_train_dir    + folderName + '/' + name + "_38.JPG",
img_seq38)
                # [img_seq39,]=seq39(images=[img])
                # imageio.imwrite(aug_train_dir    + folderName + '/' + name + "_39.JPG",
img_seq39)
```

```
                    # [img_seq40,]=seq40(images=[img])
                    # imageio.imwrite(aug_train_dir    + folderName + '/' + name + "_40.JPG",
img_seq40)
    print("------------数据集制作完成------------")
if __name__ == '__main__':
    get_dataset()
```

3. 制作数据集：读取图片，制作数据集。

部分重要代码如下：

```
import os
import cv2
import numpy as np
import tensorflow as tf
def get_dataset():
    print("------------开始制作数据集------------")
    ## dataset 路径
    train_dir = 'train_process/dataset_aug/train'
    test_dir = 'train_process/dataset_aug/test'
    ##  按照类别读取数据，添加标签
    train_data = []
    train_label = []
    test_data = []
    test_label = []
    ##  制作训练集
    folderNames = os.listdir(train_dir)
    for folderName in folderNames:
        print(folderName)
        if folderName == 'anomaly':
            label = 0
        elif folderName == 'good':
            label = 1
```

```
        elif folderName == 'one':
            label = 2
        elif folderName == 'neighbor_two':
            label = 3
        elif folderName == 'diagonal_two':
            label = 4
        elif folderName == 'three':
            label = 5
        else:
            label = 6
        folderPath = train_dir + folderName + '/'
        files = os.listdir(folderPath)
        for file in files:
            img = cv2.imread(folderPath + file)
            img = cv2.resize(img, (224, 224))
            img_tensor = tf.convert_to_tensor(img, dtype=tf.float16)
            img_tensor /= 255.0
            train_data.append(img_tensor)
            train_label.append(label)
## 制作测试集
folderNames = os.listdir(test_dir)
for folderName in folderNames:
        print(folderName)
        if folderName == 'anomaly':
            label = 0
        elif folderName == 'good':
            label = 1
        elif folderName == 'one':
            label = 2
        elif folderName == 'neighbor_two':
```

```
            label = 3
        elif folderName == 'diagonal_two':
            label = 4
        elif folderName == 'three':
            label = 5
        else:
            label = 6
        folderPath = test_dir + folderName + '/'
        files = os.listdir(folderPath)
        for file in files:
            img = cv2.imread(folderPath + file)
            img = cv2.resize(img, (224, 224))
            img_tensor = tf.convert_to_tensor(img, dtype=tf.float16)
            img_tensor /= 255.0
            test_data.append(img_tensor)
            test_label.append(label)
    train_data = tf.convert_to_tensor(train_data, dtype=tf.float16)
    test_data = tf.convert_to_tensor(test_data, dtype=tf.float16)
    print("train data length      : "+str(len(train_data)))
    print("train label length    : "+str(len(train_label)))
    print("test data length       : "+str(len(test_data)))
    print("test label length      : "+str(len(test_label)))
    print("------------数据集制作完成------------")
    return np.array(train_data), np.array(train_label), np.array(test_data), np.array(test_label)
```

3．训练：模型训练。

部分重要代码如下：

```
import tensorflow as tf
import numpy as np
import cv2
import imgaug.augmenters as iaa
```

```
from make_dataset import get_dataset
from keras.utils.np_utils import to_categorical
# 加载预训练模型
inputs = tf.keras.layers.Input(shape=[224, 224, 3])
base_model = tf.keras.applications.vgg19.VGG19(include_top=False, weights='imagenet', input_tensor=inputs)
# base_model = tf.keras.applications.inception_resnet_v2.InceptionResNetV2(layers = tf.keras.layers, include_top=False, weights = 'imagenet', input_tensor=inputs)
## 冻结参数
for l in base_model.layers:
    l.trainable = False # 第一步锁定前层，不进行训练
# 添加输出层 (batch_size, 1000)->(batch_size, 7)
x = tf.keras.layers.GlobalAveragePooling2D(name='average_pool')(base_model.output)
fla = tf.keras.layers.Flatten()(base_model.output)
fc1 = tf.keras.layers.Dense(1024, activation='relu')(fla)
drop1 = tf.keras.layers.Dropout(rate=0.2)(fc1)
fc2 = tf.keras.layers.Dense(512, activation='relu')(drop1)
drop2 = tf.keras.layers.Dropout(rate=0.2)(fc2)
predictions = tf.keras.layers.Dense(7, activation='softmax')(drop2)
model = tf.keras.models.Model(inputs=inputs, outputs=predictions)



## 不使用预训练
# inputs = tf.keras.layers.Input(shape=[512, 512, 3])
# base_model = tf.keras.applications.vgg19.VGG19(include_top=False, input_tensor=inputs)
# x = tf.keras.layers.GlobalAveragePooling2D(name='average_pool')(base_model.output)
# fla = tf.keras.layers.Flatten()(base_model.output)
# fc1 = tf.keras.layers.Dense(1024, activation='relu')(fla)
# drop1 = tf.keras.layers.Dropout(rate=0.2)(fc1)
```

```
# fc2 = tf.keras.layers.Dense(512, activation='relu')(drop1)
# drop2 = tf.keras.layers.Dropout(rate=0.2)(fc2)
# predictions = tf.keras.layers.Dense(7, activation='softmax')(drop2)
# model = tf.keras.models.Model(inputs=inputs, outputs=predictions)
## 加载训练集和测试集
train_data, train_label, test_data, test_label = get_dataset()
train_label = to_categorical(train_label, num_classes=7)
test_label = to_categorical(test_label, num_classes=7)
## 定义超参数
learning_rate = 0.001
batch_size = 64
epochs = 1000
## 定义优化器和损失函数
optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate)
loss = tf.keras.losses.categorical_crossentropy
## 编译模型
checkpoint = tf.keras.callbacks.ModelCheckpoint(filepath='train_process/model/VGG19_ft.h5',
monitor='val_categorical_accuracy', verbose=1, save_best_only=True, mode='max')
callback_list = [checkpoint]
model.compile(optimizer=optimizer, loss=loss, metrics=['categorical_accuracy'])
print("----------开始训练----------")
model.fit(train_data, train_label, epochs = epochs, batch_size = batch_size,
validation_data=(test_data, test_label), callbacks=callback_list)
loss, accuracy = model.evaluate(train_data, train_label, verbose=0)
print(f" Train Loss: {loss:.4f} Train Accuracy: {accuracy:.4f}")
loss, accuracy = model.evaluate(test_data, test_label, verbose=0)
print(f"Test Loss: {loss:.4f} Test Accuracy: {accuracy:.4f}")
```