

序号(学号): 041740233

长 春 大 学

毕 业 设 计 (论 文)

基于深度学习的水果图片分类标注系统的设计 与开发

姓 名	<u>樊佳龙</u>
学 院	<u>计算机科学技术学院</u>
专 业	<u>计算机科学与技术系</u>
班 级	<u>计科 17402</u>
指导教师	<u>陈立(副教授)</u>

2021 年 5 月 22 日

基于深度学习的水果图片分类标注系统的设计与开发

[摘 要] 随着移动互联网信息技术的蓬勃发展,网络上的数字图像等信息数据呈现出了爆炸式的增长。图像等数据信息的获得和使用变得尤为方便。而与此同时,对海量图像的自动处理和分类则显得极其重要。因此利用计算机视觉领域的相关技术来对图像内容进行分类正是解决当前问题的有效手段。

随着图像数据的增加使图像识别的任务越来越艰巨,对比传统的图像识别技术,在对输入的图像内容进行特征提取时,主要依靠人工设计的提取器,且常常伴随着复杂的调参过程,所以就会对使用人员专业能力提出了要求。并且设计好的每个方法都只能针对于特定情形,所以存在着泛化能力差,适用性存在局限的天然劣势,它需要待识别的物体背景不易改变并且物体形状较为单一。因此基于传统的人工方法进行分类的方式已难以胜任。

对比现在利用的深度学习技术与人工提取特征技术可以明显的发现,深度学习技术可以更准确且快速的提取出图像数据中的隐含信息,并对图像有较高的特征提取能力,因此设计并开发出一种基于深度学习算法实现的图片识别标注系统具有重大实践意义。

而本文所采用的深度学习技术,特点是由数据驱动来对图片进行特征提取,是一个机器自学习的过程。并且针对图像识别原理与实现方式基本相同,泛化性强;参与训练的数据集样本数量足够多并且多样,可针对不同形态测试出极高的识别准确率与更好的鲁棒性。并且出于易用的角度,本课题对功能采用可视化操作,通过设计并开发出系统软件供用户进行操作,有效的提高图片识别与标注的效率与易用性。而且由于分类与标注的图片数据对象是水果,并且在标注模块可计算出标注的水果在窗口中的坐标,基于此亦可在农业方面的水果自动采摘方向得到应用,再次印证了深度学习技术泛化性强的特点。

本文首先完成实验环境安装、数据集样本制作和样本预处理,其次进行深度学习网络模型的搭建与训练,得到训练模型以后对测试数据集进行预测正确率,最后做出了一个水果识别可视化界面系统实现三个功能:一是对水果图片(24种)的分类。二是对水果(5种)图片进行标注框选并显示出水果种类与识别准确率,以及在窗口中的坐标。三是通过摄像头实时采集画面,并对水果实物(5种)进行标注,即实现对水果实物的实时标注。

[关键词] 水果识别;深度学习;卷积神经网络;特征提取;系统开发

Design and development of fruit image classification and labeling system based on deep learning

[Abstract] With the rapid development of mobile Internet information technology, digital images and other information data on the network show an explosive growth. Image and other data information to obtain and use become particularly convenient. At the same time, the automatic processing and classification of massive images is very important. Therefore, the use of computer vision technology to classify image content is an effective way to solve the current problem.

With the increase of image data, the task of image recognition becomes more and more difficult. Compared with the traditional image recognition technology, the feature extraction of input image content mainly depends on the manually designed extractor, and is often accompanied by a complex parameter adjustment process, so it will put forward requirements for the professional ability of users. And each method can only be designed for specific situations, so it has the natural disadvantages of poor generalization ability and limited applicability. It requires that the object background to be recognized is not easy to change and the object shape is relatively single. Therefore, the traditional manual classification method is not competent.

Compared with the deep learning technology and the manual feature extraction technology, it can be found that the deep learning technology can extract the hidden information in the image data more accurately and quickly, and has higher feature extraction ability for the image. Therefore, it is of great practical significance to design and develop an image recognition and annotation system based on the deep learning algorithm.

The deep learning technology used in this paper is characterized by data-driven feature extraction, which is a process of machine self-learning. The principle and implementation of image recognition are basically the same, and the generalization is strong; The number of training data set samples is enough and diverse, which can test high recognition accuracy and better robustness for different shapes. And from the perspective of ease of use, this topic uses visual operation to the function, through the design and development of system software for users to operate, effectively improve the efficiency and ease of use of image recognition and annotation. Moreover, the image data object of classification and annotation is fruit, and the coordinates of the labeled fruit in the window can be calculated in the annotation module. Based on this, it can also be applied in the automatic picking direction of

fruit in agriculture, which proves the strong generalization of deep learning technology.

This paper first completes the experimental environment installation, data set sample production and sample preprocessing, then builds and trains the deep learning network model, obtains the training model, and then predicts the accuracy of the test data set, and finally makes a fruit recognition visual interface system to realize three functions: first, the classification of Fruit Pictures (24 kinds). The second is to select the mark box of the fruit (5 kinds) image and display the fruit type and recognition accuracy, as well as the coordinates in the window. Third, real-time images are collected by camera, and real fruits (5 kinds) are labeled, that is to achieve real-time labeling of real fruits.

[key words] Fruit recognition; deep learning; CNN; Feature extraction; system development.

目 录

1 前 言.....	1
1.1 研究背景及意义.....	1
1.2 国内外研究现状.....	2
1.2.1 深度学习的国内外研究现状.....	2
1.2.2 水果识别的研究现状.....	2
2 实验环境搭建.....	3
2.1 基础实验环境.....	4
3 系统设计与界面开发.....	7
3.1 系统设计.....	7
3.1.1 系统视图.....	7
3.1.2 执行视图.....	7
3.2 界面开发.....	10
3.2.1 界面设计.....	10
3.2.2 编码实现.....	12
4 水果图片分类.....	20
4.1 水果数据集.....	20
4.2 网络结构与训练过程.....	22
4.2.1 网络结构.....	22
4.2.2 训练过程.....	23
4.3 实验测试与分析.....	24
4.3.1 识别准确率.....	24
4.3.2 系统功能开发与展示.....	24
5 水果图片标注.....	28
5.1 水果数据集.....	28
5.2 网络结构与训练过程.....	30
5.2.1 网络结构.....	30
5.2.2 训练过程.....	31
5.3 实验测试与分析.....	32
5.3.1 标注准确率.....	32
5.4 系统功能开发与展示.....	33
6 水果实时标注.....	36
6.1 功能实现.....	36
6.2 实验测试与分析.....	37

6.3 系统功能开发与展示.....	38
7 总结与展望.....	41
7.1 总结.....	41
7.2 展望.....	41
致 谢.....	42
参考文献.....	43

装

订

线

1 前 言

1.1 研究背景及意义

本文的研究背景及意义主要来自两个层面，即通过本文基于深度学习的水果图片分类标注系统的设计与开发，一方面系统中的水果实时标注模块所采用的是识别与检测方向上更为流行与准确率更高的深度学习方式，可以对摄像头实时展示的画面进行计算，检测画面中的水果并将检测到的水果在画面上进行框选标注，并返回画面中各个水果在画面中的坐标，因为系统中的此模块可以对农业上的水果自动采摘方向提供借鉴与参考，另一方面系统中的水果图片分类模块，以多种水果图片作为数据集，采用深度学习的机器自学习的模式，在功能实现与系统开发与测试中，验证其与传统图像识别所采用的人工特征提取相比具有更好的泛化能力与鲁棒性这一结论^[1]。

所以对于农业自动采摘领域：我国目前虽然已经成为了世界上水果生产的大国之一，但是，由于中国的水果种类及品种多样化，产品的商业化水平及机械化水平较为落后。我国的水果等产品明显适合于内销，而且价格相对较低；而在其他发达国家中，如美国，对水果的从种植到采摘等的过程，均采用机械化生产，因此出口量位居世界前列。这对我国的水果出口是不利的^[2]。我国柑橘等水果的产量占全世界非常大的份额，截止到 2019 年，我国的水果总种植面积比 2018 年上涨了 2.0%，其总产量也相对增长 7.5%左右^[3]。但是，水果的种植、施肥、打药及采摘等过程基本上还是依赖于农民的辛勤劳作及部分机械性，很难实现机械化的普及，果农们在水果的生长与产出过程中投入了大量的人力物力，可以说造成了一定人力资源上的浪费，而且降低了效率。其中水果的采摘工作便是其中耗费人力的重要一环，并且由于不同水果的特点，比如椰子处在树的顶端，人工采摘是件极其麻烦的事情，再比如板栗的结构致使采摘的劳动强度大大增加，所以如果能在水果采摘时，使用深度学习方式针对特定的水果数据进行训练，并应用到采摘机器上，与其他机械技术结合实现自动采摘，则可以节省很大的成本，所以对于农业方面有着重大的研究意义^[4]。

对于验证深度学习较传统图像识别方式的优越性上：随着图像数据的增加使图像识别的任务越来越艰巨，传统的人工方法已难以胜任。而随着近年来互联网的蓬勃发展，深度学习也乘上了这趟“快车”，得到了较大的发展，它对图像识别的效率及准确性等有了很大的提升，这也使得我们能够在大数据时代下对图像提取出丰富和有价值的隐含信息，并且随着计算机硬件设备(如 GPU，CPU，内存，硬盘等)性能的显著提升，深度学习对大数据集的训练耗时长的缺点也在慢慢减弱。通过深度学习的机器自学习模式与传统人工提取特征的方式相比可以发现，深度学习模式可以更准确且快速的提取出图像数据中的隐含信息，并且还可以针对性的对信息进行分类^[5]。因此，在当下这样的大数据时代，深度学习可以将图像识别技术推进到一个新的高度。因此有必要对深度学习这个近年来取得快速进步的技术验证其相对于传统图像识别的优越性。

1.2 国内外研究现状

1.2.1 深度学习的国内外研究现状

深度学习(deep learning)概念是由美国学者 Ference Marton 和 Roger Saljo 提出，并于1976年提出了关于学习层次的概念^[6]。此后，众多研究人员开始关注深度学习，Ramsden(1988)^[7]和Entwistle(1986,2001)^[8,9]、Collis、Biggs (1982)^[10]等学者均在全方位的角度对深度学习的理论进行了研究与应用，逐渐把理论变为实际的应用。近年来，很多大公司与个人都对深度学习进行了研究与探索，开启了21 世纪人工智能领域下深度学习的发展浪潮^[11]。

随着深度学习技术的快速发展，目前很多大公司都已经研发出了自己的深度学习框架。除了大量的学术界研究学者在进行深度学习的理论研究外，工业界的贡献也是非常巨大的，例如百度、阿里巴巴、微软等大公司深度学习理论转化为代码，已经应用到具体的领域中。

1.2.2 水果识别的研究现状

实质上，物体识别利用的就是深度学习中的图像识别技术。近年来，我国的水果产业发展迅猛，水果识别也成了目前科研机构及科研工作者的主要研究方向之一。

为了实现范围广泛、效率高效、精准度高的水果识别，国内外的研究人员对图像识别的不同算法进行了全方位的研究。吕等人^[12]对水果的颜色、纹理及形状进行了分析，并采用了 BP 神经网络对水果物体进行识别，然后对水果图像进行了分析，最终取得了95%以上的准确率。曾等人^[13]通过采用 BP 神经网络对多类水果进行分类与训练，取得了较高的识别准确率；并且发现BP 神经网络

对水果的特征提取明显要高于传统技术所采用的人工方法，准确识别率也要高于传统方法。孟等人^[14]通过 使用HSV 提取颜色特征及小波变换提取纹理特征，进而对水果图像特征进行识别和训练，比单一特征的识别准确率有一定的提升。由于光照及灰度分布对图像的影响，陈等人^[15]首先将图像进行预处理，然后再通过 BP 神经网络提取水果边缘、形状和颜色等浅层特征信息并进行训练后，对水果图像进行识别与分类，其准确识别率高达 98.6%。

通过研究上述文献可以发现，图像识别的关键之处在于对图像的预处理以及特征提取。目前为止，大部分的科研人员都是在某些特定的环境下获得图像，很大程度上减弱甚至消除了环境因素对图像识别的影响。但是，在实际的图像识别中，光照、水果与摄像的距离、树叶之间的相互遮挡等因素都会影响图像识别的准确率。

装
订
线

2 实验环境搭建

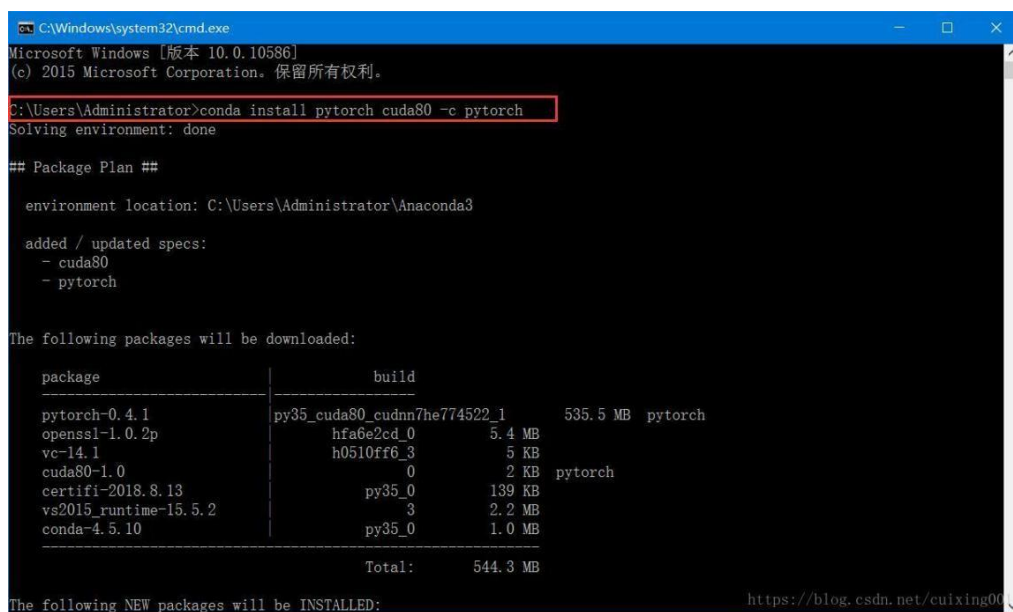
2.1 基础实验环境

本系统在 window10 操作平台进行开发。Anaconda 是一款可以在计算机上安装同一个软件的各种版本，非常方便而且速度快，所以使用它来安装环境，开发语言为 python3.7，具体详细电脑环境版本如表 2-1。

表2-1 软硬件环境表

操作系统	Windows 10
开发语言	Python3.7
GUI框架	PyQt5.9
功能框架	Pytorch1.6, YOLO v3
CPU	Intel(R) Core(TM) i7-8750HQ CPU @2.2GHz 2.91GHz
GPU	GEFORCE RTX 2070(Memory:8GB,CUDA Core:2304)
网络模型	ResNet101, DarkNet53

为了进一步加快深度学习的开发进度，本文使用的关于卷积神经网络是在python3.7 的基础上利用pytorch1.6 来搭建使用的。搭建好深度学习环境和 anaconda 后， 如下图 2-1 所示，进行开发环境的搭建。



```

C:\Windows\system32\cmd.exe
Microsoft Windows [版本 10.0.10586]
(c) 2015 Microsoft Corporation. 保留所有权利。

C:\Users\Administrator>conda install pytorch cuda80 -c pytorch
Solving environment: done

## Package Plan ##

environment location: C:\Users\Administrator\Anaconda3

added / updated specs:
- cuda80
- pytorch

The following packages will be downloaded:

package                                     build                                535.5 MB  pytorch
pytorch-0.4.1                             py35_cuda80_cudnn7he774522_1
openssl-1.0.2p                             hfa6e2cd_0                          5.4 MB
vc-14.1                                    h0510ff6_3                           5 KB
cuda80-1.0                                 0                                    2 KB
certifi-2018.8.13                          py35_0                              139 KB
vs2015_runtime-15.5.2                      3                                    2.2 MB
conda-4.5.10                               py35_0                              1.0 MB

Total:                                     544.3 MB

The following NEW packages will be INSTALLED:

```

图2.1 Pytorch 库的安装使用

安装好后上述 Pytorch 库后,还需要安装 CUDA 的 cudatoolkit, 它是一个

提供用户在 NVIDIA 的 GPU 上进行并行计算的平台,相对于传统的使用 CPU 运算,可以极大加速深度学习运算以及模型训练的效率;而 cudatoolkit 是针对此平台的工具包。选择自己 GPU 对应的版本进行安装,安装完成后在 PowerShell 命令窗口中输入 `nvcc -V` 命令,当出现制造商信息及版权信息时,说明安装成功,如下图 2-2 所示。

```
PS C:\Users\Dell> nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2018 NVIDIA Corporation
Built on Tue_Jun_12_23:08:12_Central_Daylight_Time_2018
Cuda compilation tools, release 9.2, V9.2.148
```

图 2-2 cudatoolkit 包的安装

之后还需要安装 cudnn,这是搭配 CUDA 运算神经网络的工具包,下载完成后将数据包在 CUDA 文件夹中指定位置进行替换,替换后进入里面的 extras 文件夹,进如 demo_suite,在命令行下运行 bandwidthTest.exe 与 deviceQuery.exe 这两个程序,若两个出现如下图 2-3,2-4 所示,显示 Result=PASS 的结果,说明安装成功。

最后还需要安装 torchvision,此工具包集成了一些数据集、深度学习模型、数据集转换等,安装完成后在命令行输入 python,回车,进入 python 环境,运行以下程序代码:

```
import torch
import torchvision
torch.cuda.is_available()
```

当运行结果返回 True 时说明安装成功。

为进行系统的 GUI 界面开发,还需要安装 PyQt 框架,在前面已经安装完成,Anaconda 基础上,直接在 Anaconda 的 PowerShell 窗口中,输入命令 `pip install pyqt5` 即可完成安装。正如本节所说,目前搭建的环境只是能够保证项目正常开发的基础环境,在后续章节中开发各个功能模块时,还会针对不同模块的功能实现,引入相关的库或者工具,所以在后续章节还会对实验环境进行补充。

```
Windows PowerShell
PS C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.2\extras\demo_suite> .\bandwidthTest.exe
[CUDA Bandwidth Test] - Starting...
Running on...

Device 0: GeForce GTX 1050
Quick Mode

Host to Device Bandwidth, 1 Device(s)
PINNED Memory Transfers
Transfer Size (Bytes)      Bandwidth(MB/s)
33554432                   6341.3

Device to Host Bandwidth, 1 Device(s)
PINNED Memory Transfers
Transfer Size (Bytes)      Bandwidth(MB/s)
33554432                   6445.2

Device to Device Bandwidth, 1 Device(s)
PINNED Memory Transfers
Transfer Size (Bytes)      Bandwidth(MB/s)
33554432                   94810.0

Result = PASS
```

图 2-3 cudnn 包的安装效果图 1

```
PS C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.2\extras\demo_suite> .\deviceQuery.exe
C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.2\extras\demo_suite\deviceQuery.exe Starting...

CUDA Device Query (Runtime API) version (CUDA static linking)

Detected 1 CUDA Capable device(s)

Device 0: "GeForce GTX 1050"
  CUDA Driver Version / Runtime Version      9.2 / 9.2
  CUDA Capability Major/Minor version number: 6.1
  Total amount of global memory:              4096 MBytes (4294967296 bytes)
  ( 5) Multiprocessors, (128) CUDA Cores/MP: 640 CUDA Cores
  GPU Max Clock rate:                        1493 MHz (1.49 GHz)
  Memory Clock rate:                         3504 Mhz
  Memory Bus Width:                          128-bit
  L2 Cache Size:                             524288 bytes
  Maximum Texture Dimension Size (x,y,z)      1D=(131072), 2D=(131072, 65536), 3D=(16384, 16384, 16384)
  Maximum Layered 1D Texture Size, (num) layers 1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers 2D=(32768, 32768), 2048 layers
  Total amount of constant memory:            65536 bytes
  Total amount of shared memory per block:    49152 bytes
  Total number of registers available per block: 65536
  Warp size:                                  32
  Maximum number of threads per multiprocessor: 2048
  Maximum number of threads per block:        1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size (x,y,z):  (2147483647, 65535, 65535)
  Maximum memory pitch:                       2147483647 bytes
  Texture alignment:                           512 bytes
  Concurrent copy and kernel execution:        Yes with 2 copy engine(s)
  Run time limit on kernels:                   Yes
  Integrated GPU sharing Host Memory:          No
  Support host page-locked memory mapping:     Yes
  Alignment requirement for Surfaces:          Yes
  Device has ECC support:                      Disabled
  CUDA Device Driver Mode (TCC or WDDM):       WDDM (Windows Display Driver Model)
  Device supports Unified Addressing (UVA):    Yes
  Device supports Compute Preemption:          No
  Supports Cooperative Kernel Launch:          No
  Supports MultiDevice Co-op Kernel Launch:    No
  Device PCI Domain ID / Bus ID / location ID: 0 / 1 / 0
  Compute Mode:
    < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 9.2, CUDA Runtime Version = 9.2, NumDevs = 1, Device0 = GeForce GTX 1050
Result = PASS
```

图 2-4 cudnn 包的安装效果图 2

3 系统设计与界面开发

3.1 系统设计

3.1.1 系统视图

采用 PyQt5 框架开发，系统设计包含四个功能模块，依次为水果图片分类，水果图片标注，水果实时标注，使用说明。

其中水果图片分类模块的界面默认为主界面，功能上包括单图模式与多图模式；

水果图片标注模块上，包括单图模式与多图模式；

水果实时标注模块通过调用机器上的摄像头设备，读取摄像头实时图像并对图像中的待识别物体进行标注；

使用说明模块用来将系统的使用说明以文本叙述的方式，将其显示到界面上。系统视图如下图 3-1 所示。

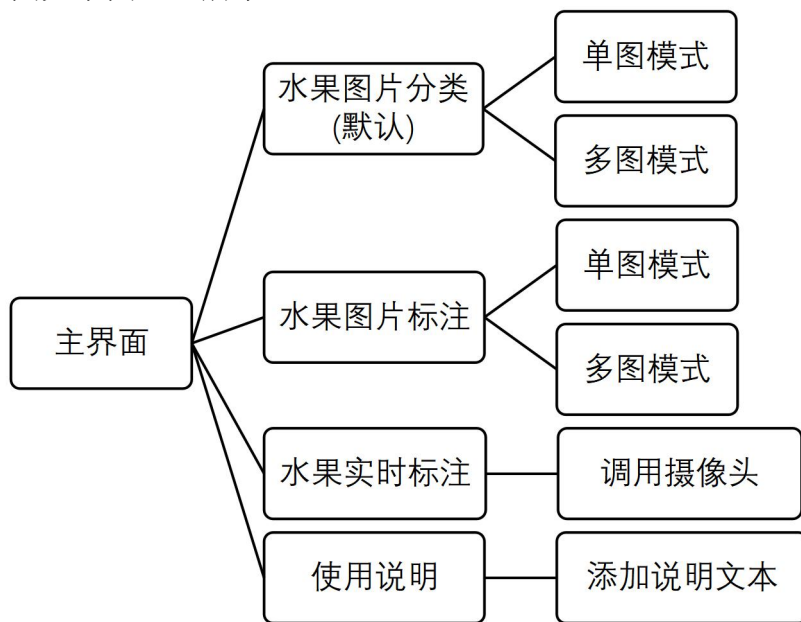


图 3-1 系统视图

3.1.2 执行视图

在上节介绍的四个功能模块后，本节介绍各个模块的执行流程，如下：

水果图片分类模块：分为单图模式与多图模式，单图模式下点击选择图片按钮选择图片，之后点击运行按钮，在已选择图片情况下，系统会执行已编写好的图像识别函数对图片进行识别，然后返回结果，并将结果在界面上进行展示。多图模式下点击选择文件夹按钮选择待检测图片文件夹，点击保存结果文件夹选择将识别结果保存到那个文件夹下，在两个文件夹已选择的情况下，点击运行按钮系统开始进行批量识别，并返回结果，将结果在页面上进行展示，

如下图 3-2 所示。

水果图片标注模块：分为单图模式与多图模式，单图模式下点击选择图片按钮选择图片，之后点击运行按钮，在已选择图片情况下，系统会执行已编写好的图像检测函数对图片进行检测，然后返回结果，并将结果在界面上进行展示。多图模式下点击选择文件夹按钮选择待检测图片文件夹，点击保存结果文件夹选择将识别结果保存到那个文件夹下，在两个文件夹已选择的情况下，点击运行按钮系统开始进行批量检测，并返回结果，将结果在页面上进行展示，如下图 3-3 所示。

水果实时标注模块：在该模块下，点击开启检测按钮，系统会自动检测机器是否拥有摄像头设备，当拥有摄像头设备后，会开启摄像头，并采集摄像头的实时画面去调用已编写好的目标检测函数去进行检测，并将检测的结果标注到摄像头画面上。如下图 3-4 所示。

使用说明模块：在该模块下，编辑好系统使用说明的文本，并在系统中使用指定标签及语句将说明文本显示到界面上。如下图 3-5 所示。

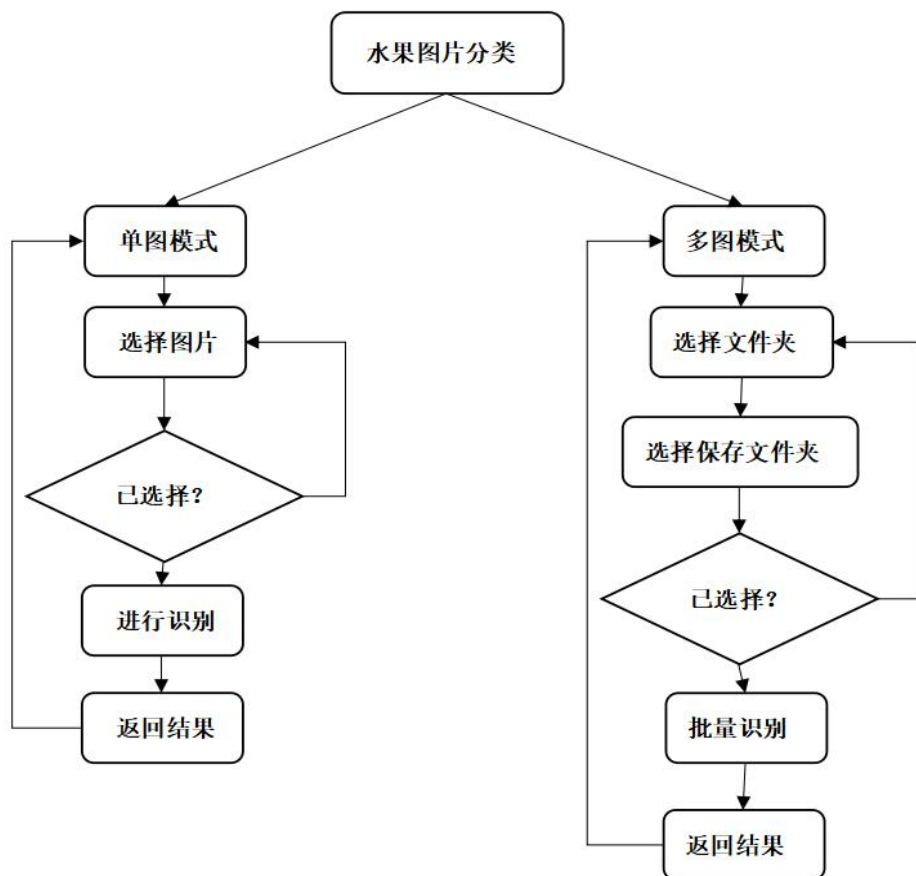


图 3-2 水果图片分类模块执行视图

装
订
线

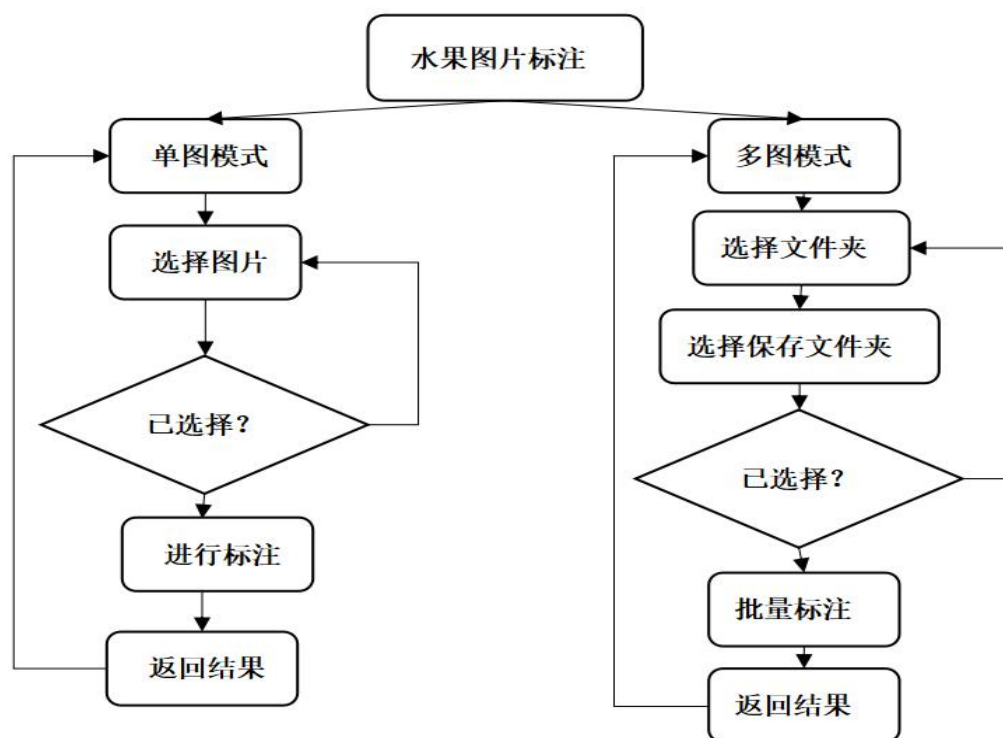


图 3-3 水果图片标注模块执行视图

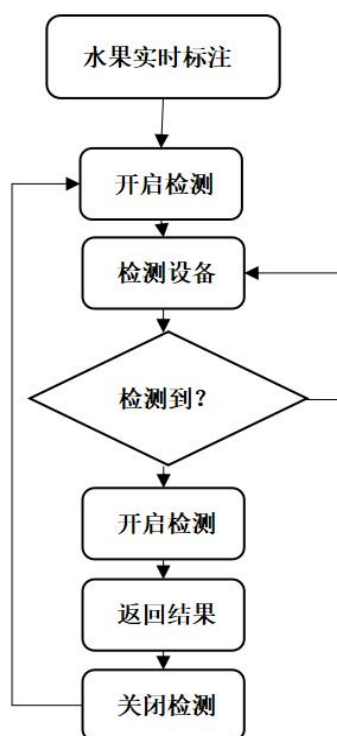


图 3-4 水果图片实时标注模块执行视图

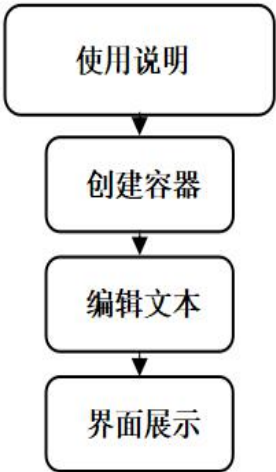


图 3-5 使用说明模块执行视图

3.2 界面开发

3.2.1 界面设计

在上节介绍完系统的功能及执行流程后，本节任务是依照系统的功能要求及执行流程，结合自己的开发经验，设计并开发出既易用，又能兼具美观的系统前台界面，方便后续章节实现重要功能后，可以方便的将功能代码封装成函数，并在前台界面对应的后端函数中进行修改，继而实现从功能到系统，从前端到后端的开发。

系统界面设计采用经典的左右结构，即左侧包含一个功能模块的按钮“列表”，只能同时点击一个功能按钮。而右侧显示的界面为对应功能的操作界面，默认第一个按钮为已点击状态，当点击不同的功能按钮后，切换到对应功能在右侧所对应的操作界面，这样就完成了系统界面的整体架构，如下图 3-6 所示，之后就是在此架构基础上，针对不同的功能模块，在其对应的右侧界面进行界面开发即可，对应功能模块的界面设计图如下图 3-7，3-8，3-9，3-10 所示：

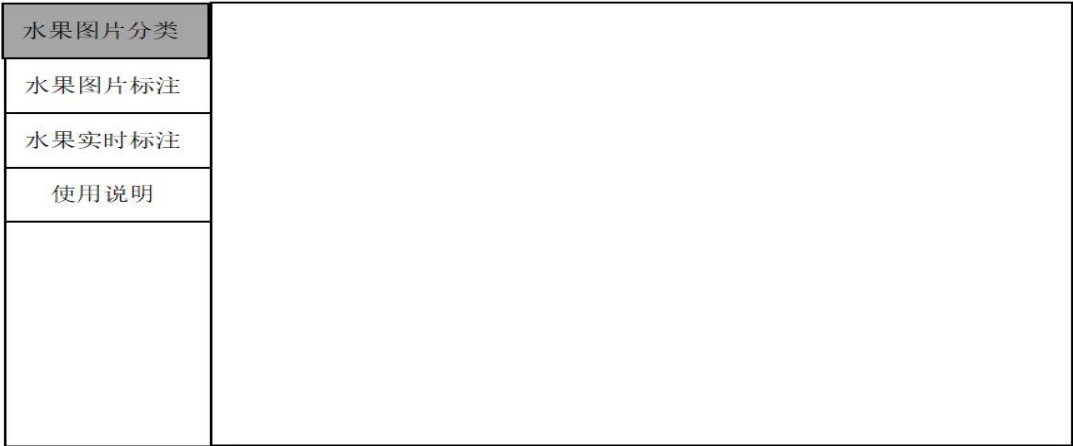


图 3-6 系统界面的整体架构图

装
订
线

水果图片分类	<div>单图图片缩略图</div>	选择图片	
水果图片标注		运行	
水果实时标注		识别结果：	
使用说明		识别准确率别准	
	文件夹路径	选择文件夹：	
	文件夹路径	保存结果文件夹	
	识别准确率下限(为进度条)	运行	
	当前文件名	进度	平均准确率

图 3-7 水果图片分类界面的设计图

水果图片分类	<div>单图图片缩略图</div>	选择图片
水果图片标注		运行
水果实时标注		
使用说明		
	识别结果展示容器	识别结果：
		识别准确率别准
	文件夹路径	选择文件夹：
	文件夹路径	保存结果文件夹
	当前文件	进度

图 3-8 水果图片标注界面设计图

装
订
线

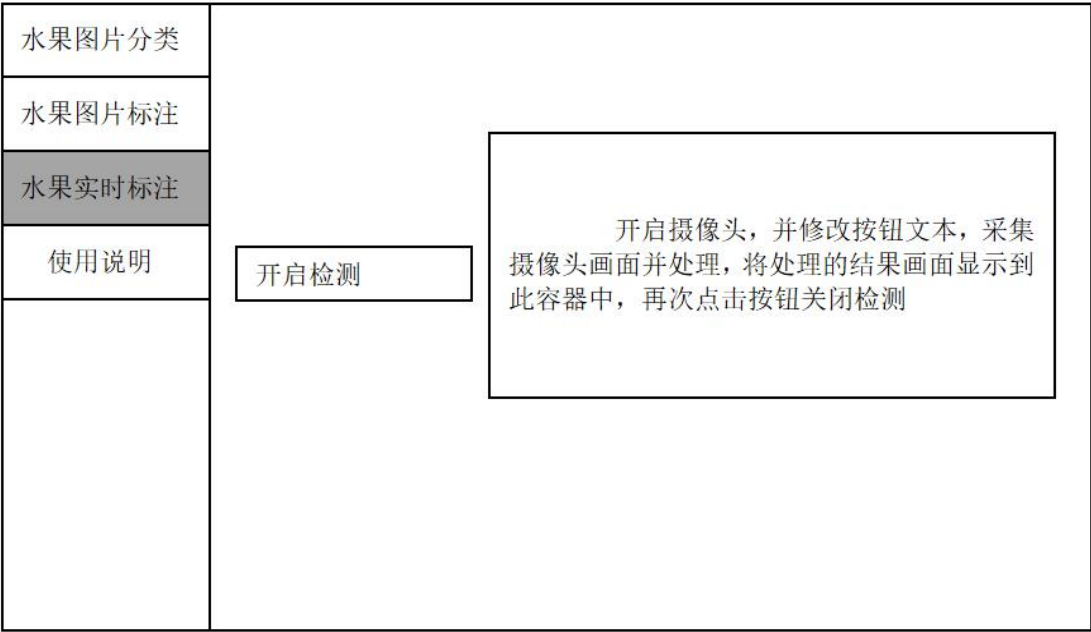
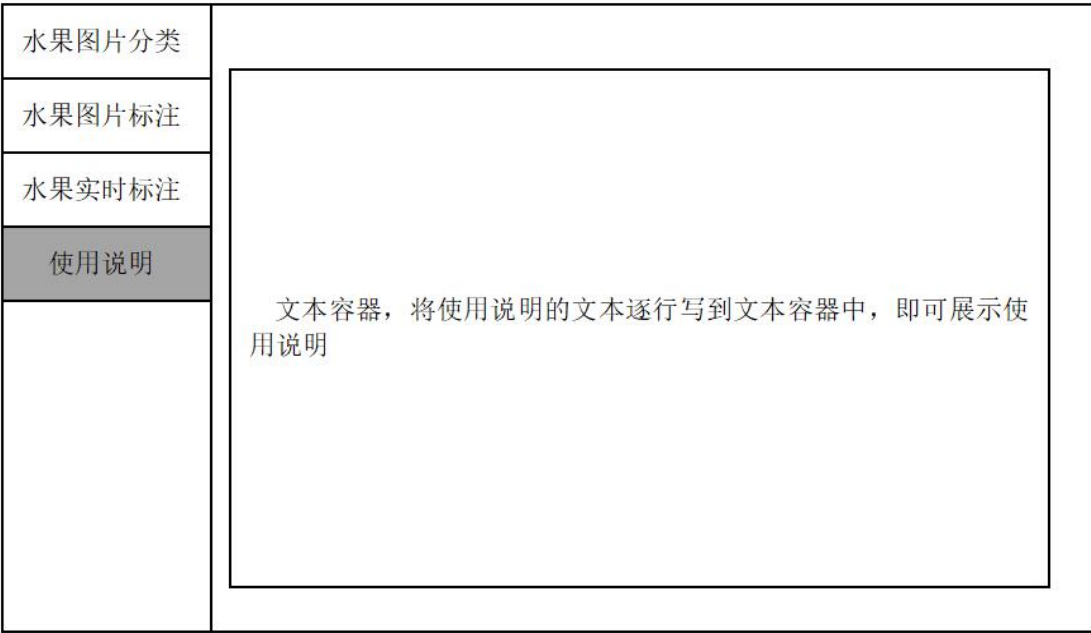


图 3-9 水果实时标注模块界面设计图



图片 3-10 使用说明模块界面设计图

3.2.2 编码实现

在上一节确定并画出了系统各个功能模块的界面原型图后，在本节就可以使用 PyQt5 框架将系统界面编写出来了。对于系统界面的整体架构，先在项目中创建主窗口程序，主窗口使用 QListWidget 组件显示，在主窗口中可以使用 PyQt 框架中的 QStackedWidget 堆栈窗口组件，依靠这个组件可以在窗口中创建堆栈窗口，组件中每个功能按钮在点击后都会链接到对应的 stack 页面，然后将四个功能模块的 stack 页面链接到对应模块的子界面程序中，这样可以保证主窗口程序的简洁，并且代码逻辑清晰，起到了解耦合的作用，关键代码如下

下：

```
self.list = QListWidget()
self.list.setMaximumWidth(100)
# 设置列表内容（stack 的索引）
self.list.insertItem(0, '水果图片分类')
self.clearFocus()
self.list.insertItem(1, '水果图片标注')
selectItem = self.list.item(0)
selectItem.setSelected(True) # 默认选中第一项
self.list.insertItem(2, '水果实时标注')
self.list.insertItem(3, '使用说明')
# 创建四个 stack 页面
self.stack1 = SanUI1()
self.stack2 = SanUI2()
self.stack3 = SanUI3()
self.stack4 = SanUI4()
# 将三个 stack 页面加入 stackWidget
self.stackWidget = QStackedWidget()
self.stackWidget.addWidget(self.stack1)
self.stackWidget.addWidget(self.stack2)
self.stackWidget.addWidget(self.stack3)
self.stackWidget.addWidget(self.stack4)
```

接下来开始编写功能模块界面，即堆栈窗口所链接的右侧界面。首先进入水果图片分类模块所创建的 SanUI1() 类中，对于界面中的各个组件，使用 PyQt 中的 QGridLayout 即栅格布局来进行摆放，创建好相应的组件，并将组件加入到栅格布局中，加入时需传递四个参数，分别为组件从第几行第几列开始摆放，组件占用几行几列。

重点是对于界面中各个按钮的槽函数链接问题，即需要对各个按钮在被点击时链接到指定槽函数中（下文直接称呼为槽函数），这样在后续章节实现了功能后，就可以将实现功能的代码改写到指定槽函数中，就实现了系统的功能，所以这些按钮对应的槽函数必须先创建好留着备用。

将单图模式下的选择图片按钮链接到创建的 def openimage() 槽函数中，将运行按钮链接到创建的 def run() 槽函数中，将多图模式下选择文件夹按钮链接到创建的 def openimagedir() 槽函数中，将保存结果文件夹按钮链接到创建的 def saveimgdir() 槽函数中，对于识别准确率下限的滑动条组件，当其数值改

变时，将其链接到创建的 `def ratechange()` 槽函数中，在此函数中实时修改滑动条的数值即可实现滑动条的组件效果，将运行按钮链接到创建的 `def multirun()` 槽函数中。上述流程的关键代码如下：

```

        self.button_search_image.clicked.connect(lambda: self.openimage())
        self.button_run.clicked.connect(lambda: self.run())
        self.button_search_dir.clicked.connect(lambda:
self.openimagedir())
        self.button_save_dir.clicked.connect(lambda: self.saveimgdir())
        self.rate_slider.valueChanged.connect(lambda: self.ratechange())
        self.button_multirun.clicked.connect(lambda: self.multirun())
        self.setLayout(self.layout)
def openimage(self):
    print('self.label_image: {}'.format(self.label_image))
def run(self):
    print('filename: {}'.format(filename))
def openimagedir(self):
    print('test')
def saveimgdir(self):
    print('test')
def multirun(self):
    print(11)
def ratechange(self):
    self.ratetext_display.setText(str(self.rate_slider.value()) + ' %')
    global currentRate
    currentRate = str(self.rate_slider.value())

```

编写好水果图片分类的界面后，可以对一些组件添加样式，PyQt 中的样式表成为 QSS，与 CSS 很相似，所以修改样式比较简单，美化样式后，可以运行系统，得到的实际界面效果如下图 3-11 所示。

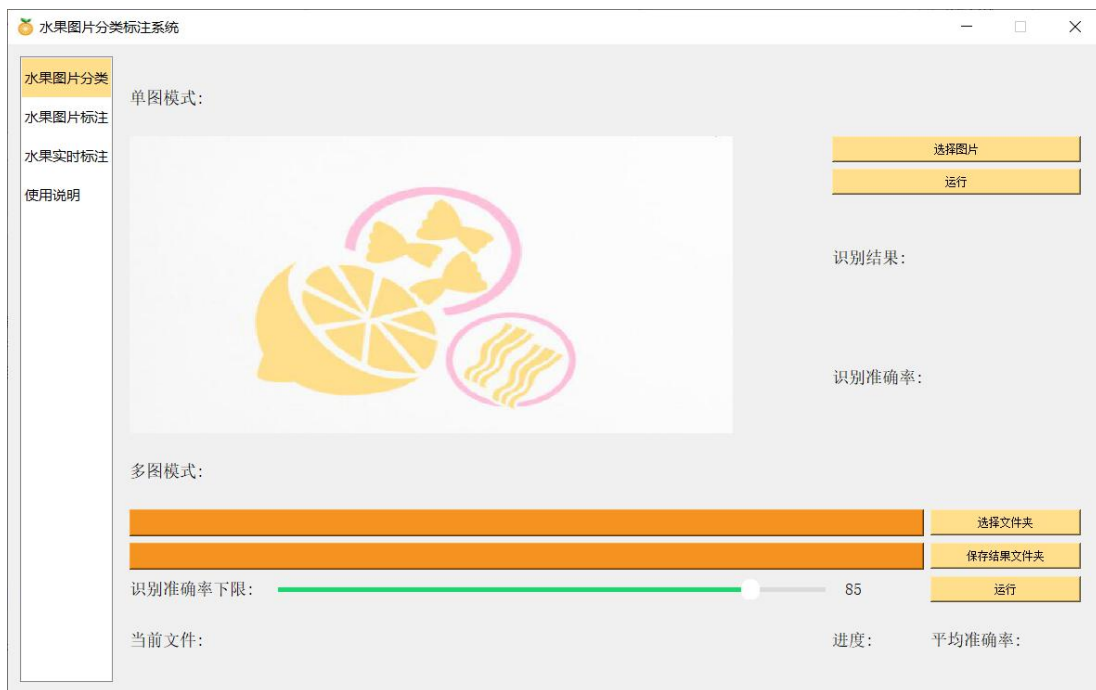


图 3-11 水果图片分类模块实际界面图

接下来编写水果图片标注模块的右侧界面：

首先进入水果图片标注模块所创建的 SanUI2() 类中，对于界面中的各个组件，使用 PyQt 中的 QGridLayout 即栅格布局来进行摆放，创建好相应的组件，并将组件加入到栅格布局中。

对于界面中各个按钮的槽函数链接问题，即需要对各个按钮在被点击时链接到指定槽函数中(下文直接称呼为槽函数)，这样在后续章节实现了功能后，就可以将实现功能的代码改写到指定槽函数中，就实现了系统的功能，所以这些按钮对应的槽函数必须先创建好留着备用。

将单图模式下的选择图片按钮链接到创建的 def openimage() 槽函数中，将运行按钮链接到创建的 def run() 槽函数中，将保存图片按钮链接到创建的 def singleimgsave() 槽函数中，将保存结果数据按钮链接到创建的 def singledatasave() 槽函数中，将多图模式下选择文件夹按钮链接到创建的 def openimagedir() 槽函数中，将保存结果文件夹按钮链接到创建的 def saveimgdir() 槽函数中，将运行按钮链接到创建的 def multirun() 槽函数中。上述流程关键代码如下：

```
self.button_search_image.clicked.connect(lambda: self.openimage())
self.button_run.clicked.connect(lambda: self.run())
self.button_saving.clicked.connect(lambda: self.singleimgsave())
self.button_savdata.clicked.connect(lambda: self.singledatasave())
## print('传回来的 jpg: {}'.format(jpg))
```

```

        self.button_search_dir.clicked.connect(lambda:
self.openimagedir())
        self.button_save_dir.clicked.connect(lambda: self.saveimgdir())
        self.button_multirun.clicked.connect(lambda: self.multirun())
        self.setLayout(self.layout)
def openimage(self):
    print('self.label_image: {}'.format(self.label_image))
def run(self):
    print('filename: {}'.format(filename))
def singleimgsave(self):
    print('labelimg: {}'.format(labelimg))
def singledatasave(self):
    print(666)
def openimagedir(self):
    global dirname
    dirname = QtWidgets.QFileDialog.getExistingDirectory(None, "选取
文件夹", "C:/") # 起始路径
    self.button_dir_display.setText(dirname)
def saveimgdir(self):
    global savedirname
    savedirname = QtWidgets.QFileDialog.getExistingDirectory(None, "
选取文件夹", "D:/") # 默认打开的起始路径
    self.button_savedir_display.setText(savedirname)
def multirun(self):
    print(666)

```

编写好水果图片标注的界面后，可以对一些组件添加样式，美化样式后，可以运行系统，得到的实际界面效果如下图 3-12 所示。

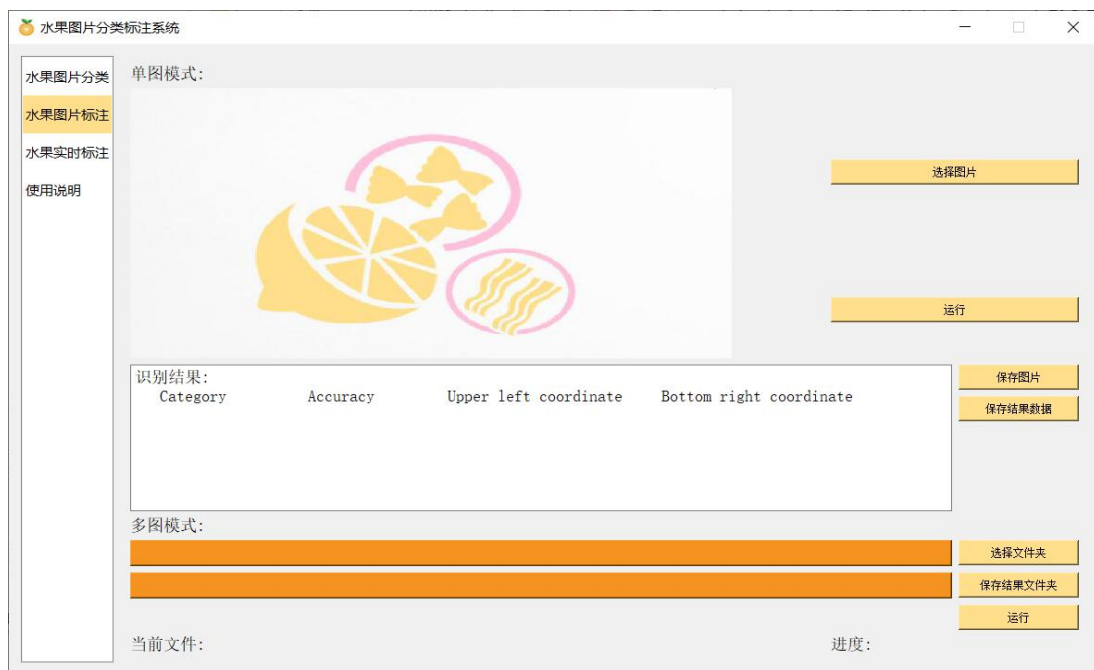


图 3-12 水果图片标注模块实际界面图

接下来编写水果实时标注模块的右侧界面：

此模块界面简单，首先进入水果实时标注模块所创建的 `SanUI3()` 类中，对于界面中只有一个 `QPushButton` 开启检测按钮，和一个 `QLabel` 标签组件用来在后续章节中实现采集摄像头画面并标注，将界面实时显示到这个标签中。使用 `PyQt` 中的 `QGridLayout` 即栅格布局来进行摆放，创建好相应的组件，并将组件加入到栅格布局中。

此模块只涉及开启检测这一个按钮，被点击时将其链接到创建的 `def run()` 槽函数中，在 `run` 函数中需要使用 `if else` 语句判断按钮状态，当按钮文本为“开启检测”时，说明还未开启检测，则导入 `Python` 下 `OpenCV` 库 `cv2`，并使用此库中的 `VideoCapture` 函数来载入摄像头设备，函数需要传递一个数字参数，表示载入设备的第几个摄像头设备（如笔记本中自带一个前置摄像头，此设备默认为 0，所以当要载入外接的摄像头设备时，则需要传入 1），然后在主函数中使用 `QTimer` 定时器组件创建一个定时器，并开启定时器，间隔设为 100 毫秒。在 `run` 函数中当定时器计时完毕后执行采集摄像头画面并标注的函数 `capPicture()` 函数，这样就实现了每隔 100 毫秒获取一次摄像头画面标注的功能。然后将更改开启检测按钮的文本为“关闭检测”，此时再点击此按钮时，会进入 `else` 语句代码块，操作为关闭摄像头并修改按钮文本为“开启检测”。关键代码如下：

```
self.btn_cameractrl.clicked.connect(lambda: self.run())
self.timer = QTimer()
self.timer.start()
```

```

self.timer.setInterval(100)
self.setLayout(self.layout)
def run(self):
    if self.btn_cameractrl.text() == '开启检测':
        self.cap = cv2.VideoCapture(1, cv2.CAP_DSHOW)
        self.timer.timeout.connect(self.capPicture)
        self.label.setStyleSheet('border:none')
        self.btn_cameractrl.setText('关闭检测')
    else:
        self.btn_cameractrl.setText('开启检测')
        self.cap.release()
def capPicture(self):
    if (self.cap.isOpened()):
        ret, img = self.cap.read()
        imgHeight, imgWidth, perImage = img.shape
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        frame = Image.fromarray(np.uint8(img))

```

编写好水果实时标注的界面后，可以对一些组件添加样式，美化样式后，可以运行系统，得到的实际界面效果如下图 3-13 所示。

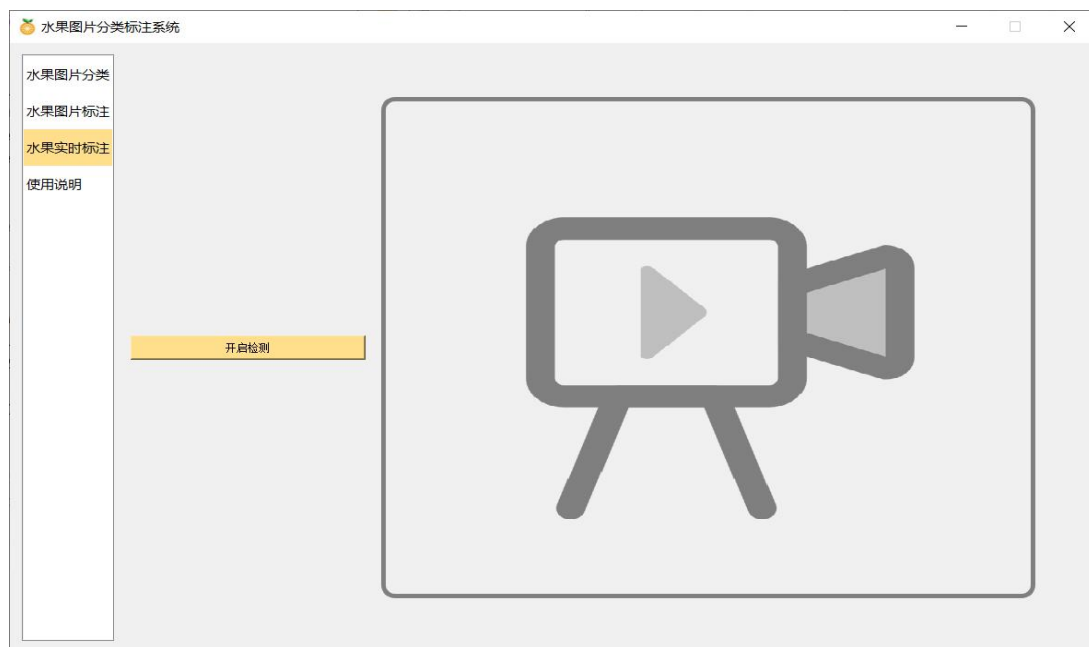


图 3-13 水果实时标注模块实际界面图

接下来编写使用说明的右侧界面：

这个是最简单的一个功能模块,首先进入使用模块所创建的 SanUI4()类中,只需创建多个 QLabel 标签组件,将使用说明的文本按行追加到标签组件中即可实现界面使用说明文本的展示(代码比较简单,不在展示代码)。使用 PyQt 中的 QGridLayout 即栅格布局来进行摆放,创建好相应的组件,并将组件加入到栅格布局中。可以运行系统,得到的实际界面效果如下图 3-14 所示。



图 3-14 使用说明模块实际界面图

4 水果图片分类

4.1 水果数据集

为了使系统功能在实验测试环境及实际生产环境的识别准确率都能取得更好的结果，实现所采用的水果数据集一部分来自知名开源水果数据集 Fruits 360 中的部分数据集图片（24 种水果），由于 Fruits 360 数据集在制作过程中是将水果放置在低速马达的轴上，通过旋转来水果的不同角度来记录 20 秒的视频短片，并将视频按指定帧数分割成图片制成，所以其包括了不同水果拍摄的不同视角（俯视、正视和左视），保证了实验环境下的识别准确率，如下图 4-1 所示。同时另一部分数据集通过使用 Python 爬虫的方式爬取了网络上对应的 24 种水果图片对数据集进行补充，保证了系统在现实环境下的识别准确率，提升了系统识别的兼容性。

水果一共有 24 个种类，如下：哈密瓜，山竹，杏，杨梅，柚子，柠檬，桃子，桑葚，梨子，樱桃，橙子，火龙果，猕猴桃，石榴，芒果，苹果，草莓，荔枝，菠萝，葡萄，蓝莓，西瓜，香蕉，龙眼。

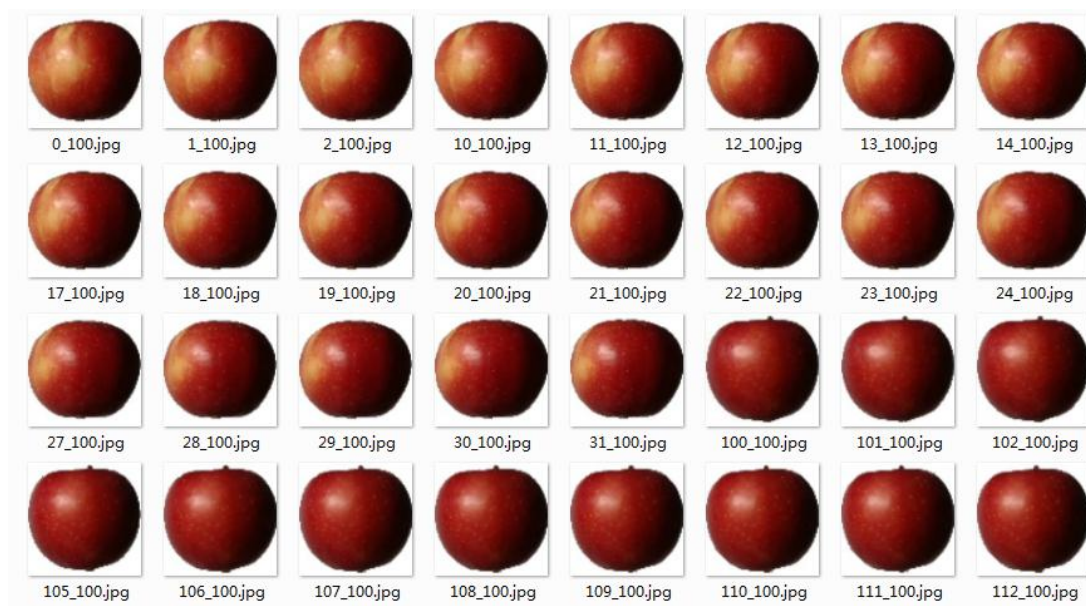


图4-1不同种类水果的数据集中均包含水果的多个角度图片

接下来将获取到的水果图片数据集的按照9:1的比例分成训练集(train)和验证集(val)，其中训练集是训练模型时用到的数据集，验证集中存放少量数据集图片，由于其没有参与到模型的训练，即学习后的模型还没有“认识”它，因此可用来验证模型所展现的能力，图片数据集的存放形式如下图 4-2所示。

装
订
线

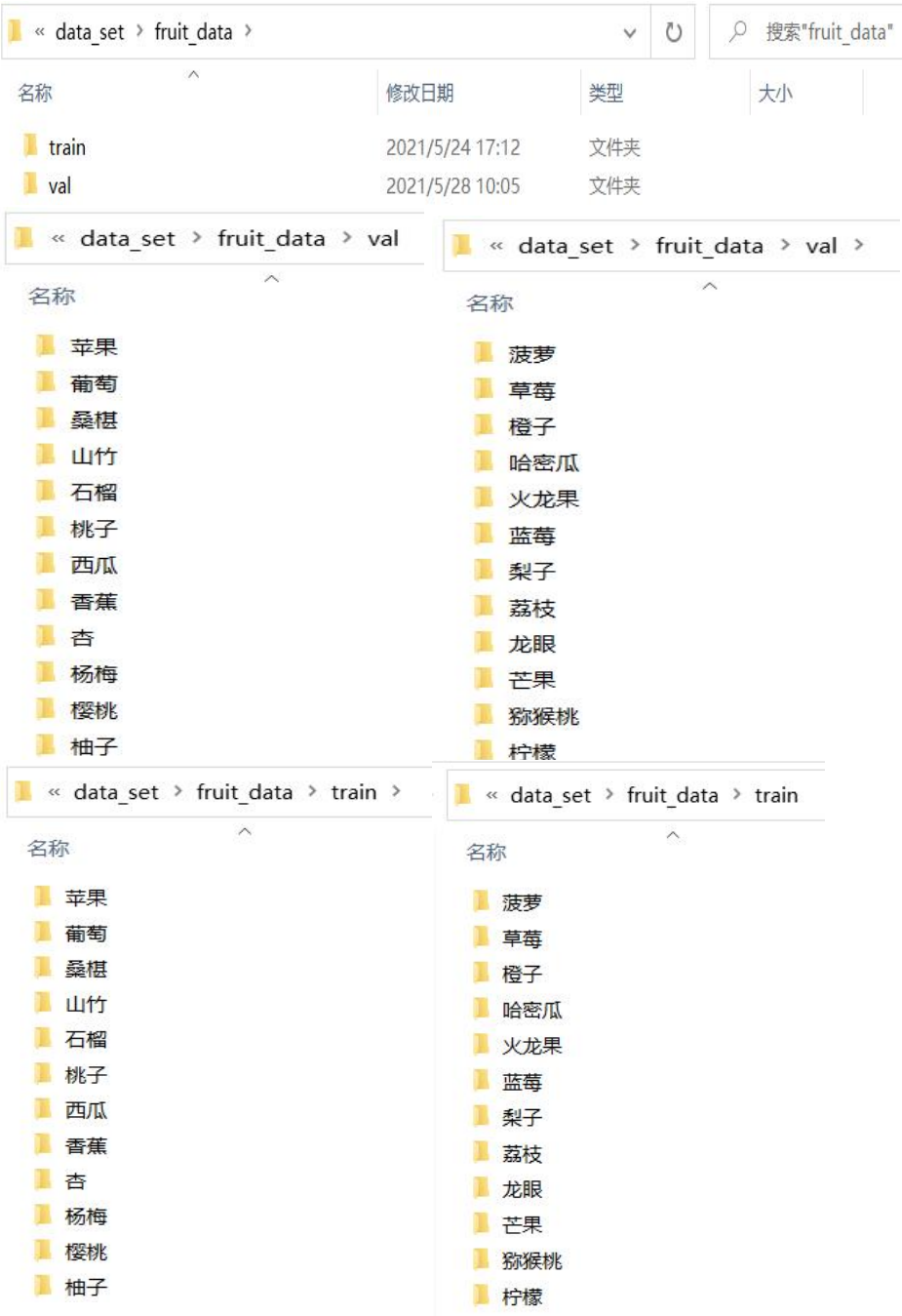


图 4-2 图片数据集整体存放形式与训练集验证集存放形式

4.2 网络结构与训练过程

4.2.1 网络结构

为了能够使用深度学习根据自己的数据集训练模型，构建一个好的网络模型十分重要。本文在水果图片分类模块中所使用的网络模型为含有 101 层网络的 ResNet 模型，即 ResNet101。所以有必要介绍下 ResNet 的由来并对其结构进行简单解析。

ResNet 网络于 2015 年提出，在此之前，人们发现对于当时的浅层网络模型，适当增加网络的深度是提高网络模型对图像的识别准确率的一种有效手段。所以人们自然想到通过堆叠网络的层数来增加网络的深度，继而提高网络模型的识别准确率。但随着网络深度的增加，却出现了非过拟合的准确率下降的情况。

原因是对于当时的普通网络(Plain network)如 VGGNet 等网络而言，简单的依靠卷积层与池化下采样去堆叠网络层数后，会使得网络算法难以在训练上进行优化，并且训练误差会越来越大(如梯度消失和梯度爆炸)，并会出现网络退化的现象，如下图 4-3 所示。

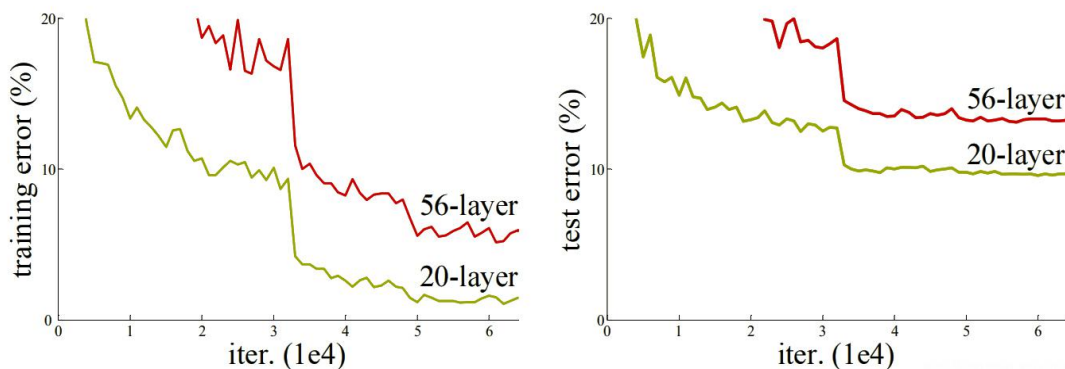


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer “plain” networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

图 4-3 对于两种层次的普通网络使用 CIFAR-10 数据集进行测验效果图

其中梯度消失，梯度爆炸的问题可以对数据进行标准化处理还有权重初始化，以及 BN 即 Batch Normalization(本质还是一种标准化处理)。但当时还是无法处理网络的退化问题。

此时随着含有残差结构的 ResNet 网络的出现，使得网络的“退化”问题最终得以解决。面对普通网络(Plain network)在加深网络深度后所出现的问题，发明者何凯明博士提出并应用了残差结构，在残差结构中，会将输入与输出之间建立一个直接相连的连接，在其论文中被称为 Shortcut Connection，此时于后面增加的网络层，只需要在原来输入层的基础上继续学习新的特征即可，这

样就避免了网络层数增加而产生的退化问题。

由于使用了残差结构，使得 ResNet 网络创造出了多种层数的网络模型，几种常见的层数的 ResNet 网络结构图如下图 4-4 所示，其中包括本文所使用的 ResNet101 网络。

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

图 4-4 ResNet 的几种常见层数的网络结构

其中对于 101 层的 ResNet 网络，参考其网络结构图，分析出其对出入图像内容的大致操作流程如下：

首先在 conv1 中对图像内容进行 7*7 的卷积操作，然后在 conv2_x 中，先经过 3*3 的最大池化下采样操作提取特征，之后就会进入到 4 个残差结构的“集合”（从 conv2_x 到 conv5_x），所以对于 101 层的网络，要经过 33（3+4+23=33）个残差结构，由前文已知，每个残差结构又包含三层，并且这些层数的网络最后都会经过 fc 层即全连接层用来分类，所以一共有 101（1+99+1=101）层。

4.2.2 训练过程

在上一节了解了 ResNet 网络的结构及操作的大致流程后，拥有了理论基础，就可以调用 torchvision.models 中的 ResNet101 模型，并根据自己的数据集的特点进行适当的修改，用此网络来训练自己的数据集了。

设计好网络结构后就可以对网络进行训练了，模型参数在一次训练后更新参数，这时需要一个学习率决定了参数更新的幅度。而在编写好的训练程序 train.py 文件中，通过使用 Pytorch 深度学习框架中提供的相关组件，我们可以方便的生成训练模型期间的学习情况的图像，如下图 4-5 所示，loss 曲线的横坐标代表训练的次数，纵坐标反映了训练过程中 loss 的变化，分析图像内容可以得到，随着训练迭代 Epoch 次数的增加，loss 的大小首先剧烈的下降，然后在某一个数值范围内上下震荡，然后最终的模型选择就是在 loss 曲线平稳的位置选择。由此可见对于普通网络而言，使用 ResNet 这样的深层网络，可以得到更好的收敛效果。

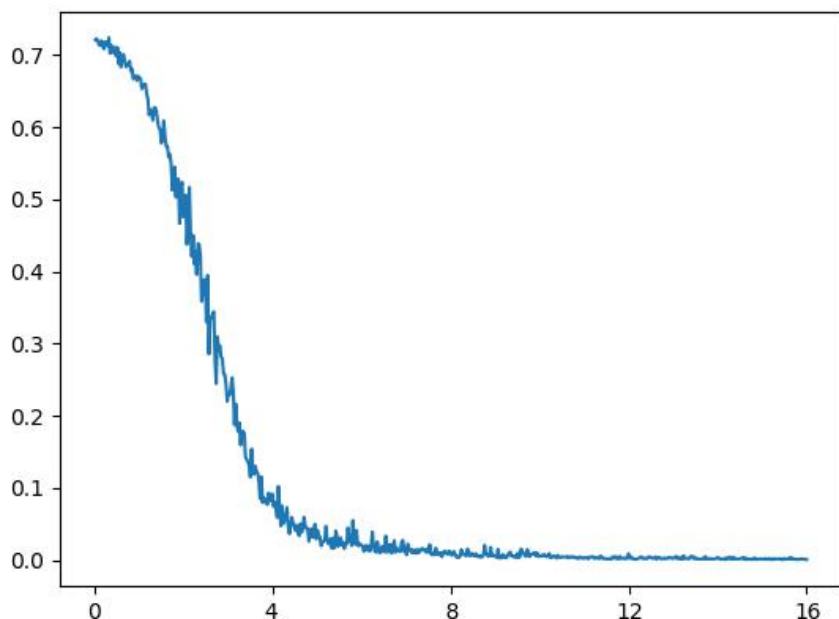


图 4-5 训练时的 loss 曲线图

4.3 实验测试与分析

4.3.1 识别准确率

经过上述的计算，本文计算出了这 24 类水果的精确率识别效果图如下图 4-6 所示，由图 4-6可知，大部分的目标识别率为 100%，只有少数个别的目标识别会有少许的错误，因此所有数据集的测试正确率为 97%，得到了十分可观的测试效果。

```
Apple Pink Lady测试数据集的正确率为1.0
Avocado测试数据集的正确率为1.0
Banana Lady Finger测试数据集的正确率为1.0
Carambula测试数据集的正确率为0.9457831325301205
Cherry Wax Black测试数据集的正确率为1.0
Grape White 2测试数据集的正确率为1.0
Pepino测试数据集的正确率为1.0
Pitahaya Red测试数据集的正确率为1.0
Raspberry测试数据集的正确率为1.0
Strawberry测试数据集的正确率为0.774390243902439
所有测试数据集的正确率为0.9713395638629283
```

图 4-6 验证训练后的模型的识别准确率(部分)

4.3.2 系统功能开发与展示

在验证了模型对目标识别的能力并符合要求后，将目标识别的功能代码封

装成函数，就可以将识别功能链接到项目的系统界面中来提升易用性了，大致流程如下：

- （1） 在主窗口中QStackedWidget堆栈窗口控件中,对于水果图片分类这一层，链接到新创建的子程序的 SanUI1 类中。
- （2） 对于单图模式，在选择图片按钮所链接的槽函数 openimage 中，使用 QFileDialog 类下的 getOpenFileName() 函数获得用户选择的图片的路径，声明全局变量去接收此路径。

然后点击运行按钮开始识别，在对应的槽函数 run() 中先进行条件判断(如路径是否为空)后，调用之前封装好的目标识别函数，并将得到的返回结果(一个是识别到的种类，一个是识别的准确率)赋给对应的 QLabel 标签，使得界面能够显示出识别的结果与准确率。如下图 4-7 所示。

对于多图模式，相对于单图模式，多图模式下需要对选中目录下的所有图片进行循环识别，所以当数据量过大时，循环执行的时间过长，在 PyQt 框架中就会出现窗口“假死”的情况下，而且当耗时的操作长期占用 UI 主线程时，导致 UI 界面无法实时刷新，造成的结果就是当程序运行时，无法通过鼠标拖动程序窗口使其移动，并且界面上无法实时显示正在操作的图片文件名和进度数。所以出于以上考虑，决定对多图模式这个耗时长功能，在他的运行按钮所链接的槽函数 multirun() 中先将运行按钮禁用，然后创建新的线程类，即新开辟一个线程，让功能函数在这个新线程中执行，这样就避免了耗时操作与主线程抢占资源所造成较差的使用体验。在线程类中的 run() 函数中创建循环来遍历目录中文件列表，并对遍历到的文件调用目标识别函数得到返回结果并调整数据(如系统中识别准确率下限功能，当返回的结果数据中的准确率低于这个下限时，说明准确率未达到预期，将识别结果的水果种类设为待定)与显示数据(如界面中最后要显示出目录下所有图片识别完成后的平均识别准确率)，如图。其中线程类中需要创建两个信号量，一个在循环中进行发射，执行对应函数来实时更新界面中显示的当前执行标注的图片名称与进度数字，另一个信号在循环结束后发射，执行对应函数来更新按钮状态，将按钮变成可点击。如下图 4-8，4-9 所示。

装
订
线



图 4-7 单图模式点击运行按钮得到识别结果与识别准确率



图 4-8 多图模式点击运行后按钮变灰，不可点击，并显示实时进度

装
订
线



图 4-9 多图模式运行结束后按钮变为正常，可再次执行

5 水果图片标注

5.1 水果数据集

对于前文的水果图片分类功能，由于原理是将一整幅图像的内容作为输入，然后经过计算并生成一个概率最高的单一标签，所以在制作数据集时只需要保证对应水果种类的图片数据量足够即可，但对于水果图片标注功能而言，输入的图像内容可能会包含多个种类水果，而且为了能够通过合适的网络模型让机器学习到识别物体种类及位置的能力，在数据集获取到之后，还需要对数据集图片进行“制作”，即手动标注出图片中包含的水果种类及各个水果在图片中的位置。所以不难发现，对于人工智能而言，“人工”与“智能”同样重要。

显然在需要保证识别准确率的情况下，手动标注数据集图片是一个重复性较强的操作，所以对于水果图片标注功能，本文仅以五种水果的图片标注作为演示，五种水果为：苹果，香蕉，草莓，橙子，猕猴桃。

为了能够提高标注的效率，一个趁手的工具显得尤为重要，开源的 labelImg 深度学习网络训练数据集标注工具是一个不错的选择，在 Anaconda 的命令窗口下输入 `pip install labelimg` 命令即可完成工具的安裝，安裝完成后在命令窗口下输入 `labelimg` 命令即可启动工具，工具界面如下图 5-1 所示。

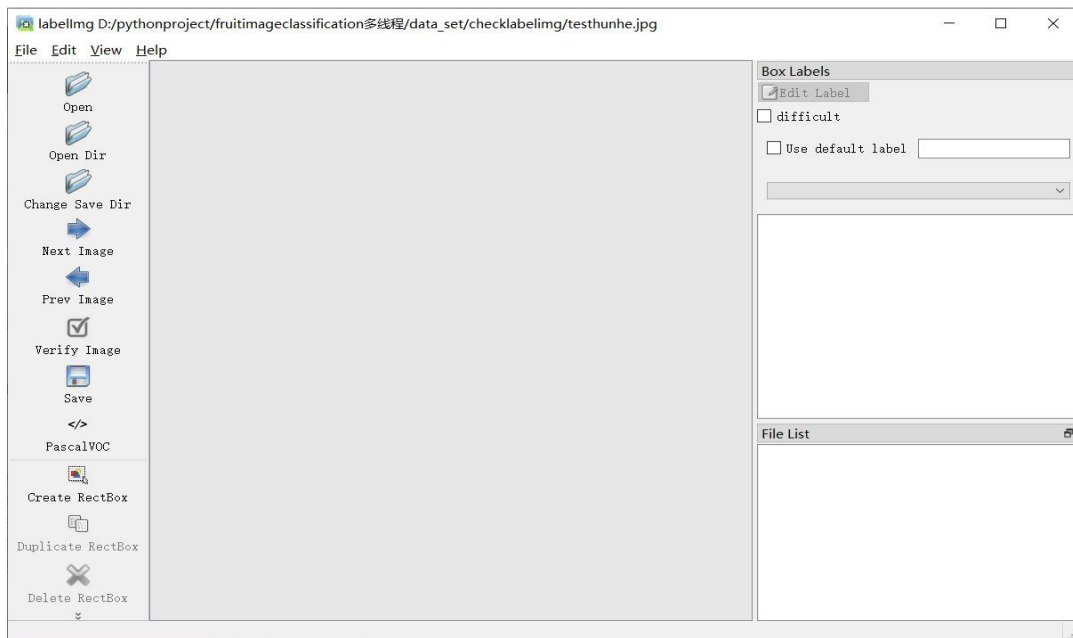


图 5-1 labelImg 工具界面

将待标注的数据集图片存放到同一个文件夹下，之后点击 Open Dir 按钮选择刚才存放数据集图片的文件夹，然后再点击 Change Save Dir 按钮选择存放标注结果的文件夹，即可对目录中的所有数据集图片进行依次标注，并将图片中手动标注的物体信息(物体种类，位置坐标)保存到生成的 xml 文件中，并移动

到刚才指定的文件夹中。大大提高了制作数据集的效率，如下图 5-2，5-3 所示

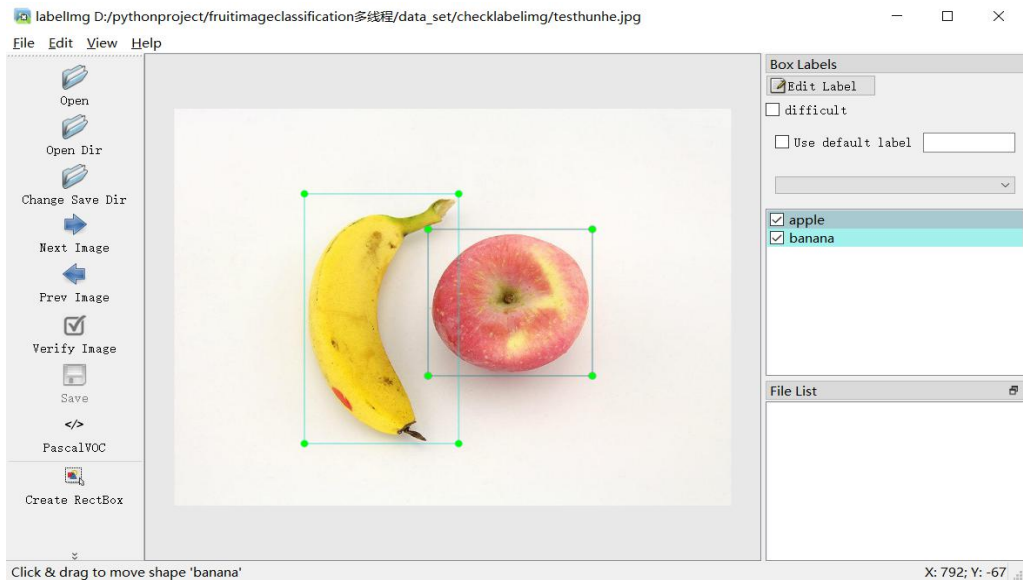


图 5-2 使用 labelImg 工具标注物体效果

```
testhunhe.xml - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
<annotation>
  <folder>checklabelimg</folder>
  <filename>testhunhe.jpg</filename>
  <path>D:\pythonproject\fruitimageclassification多线程\data_set\checklabelimg\testhunhe.jpg</path>
  <source>
    <database>Unknown</database>
  </source>
  <size>
    <width>1000</width>
    <height>750</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>apple</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>455</xmin>
      <ymin>227</ymin>
      <xmax>750</xmax>
      <ymax>504</ymax>
    </bndbox>
  </object>
  <object>
    <name>banana</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>233</xmin>
      <ymin>160</ymin>
      <xmax>510</xmax>
      <ymax>632</ymax>
    </bndbox>
  </object>
</annotation>
```

图 5-3 保存标注结果后生成的文件中的物体信息

5.2 网络结构与训练过程

5.2.1 网络结构

由于水果图片标注功能涉及到了深度学习中的目标检测模块，而对于上一章水果图片分类功能所使用的网络模型而言，主要基于图像的内容对图像进行标记，生成单一的标签，所以当图像的内容包含多个待检测，分类的物体时，传统用于分类的网络模型很难再出色的完成任务，此时对于水果图片标注的功能，需要“另起炉灶”，即使用专门的目标检测的网络模型，并在此基础上进行开发。

本文所使用的目标检测算法为目标检测领域知名的 YOLO v3 目标检测算法，算法中所使用的网络为 DarkNet53 网络，关于 YOLO v3 算法与 DarkNet53 网络的结构图如下图 5-4 所示

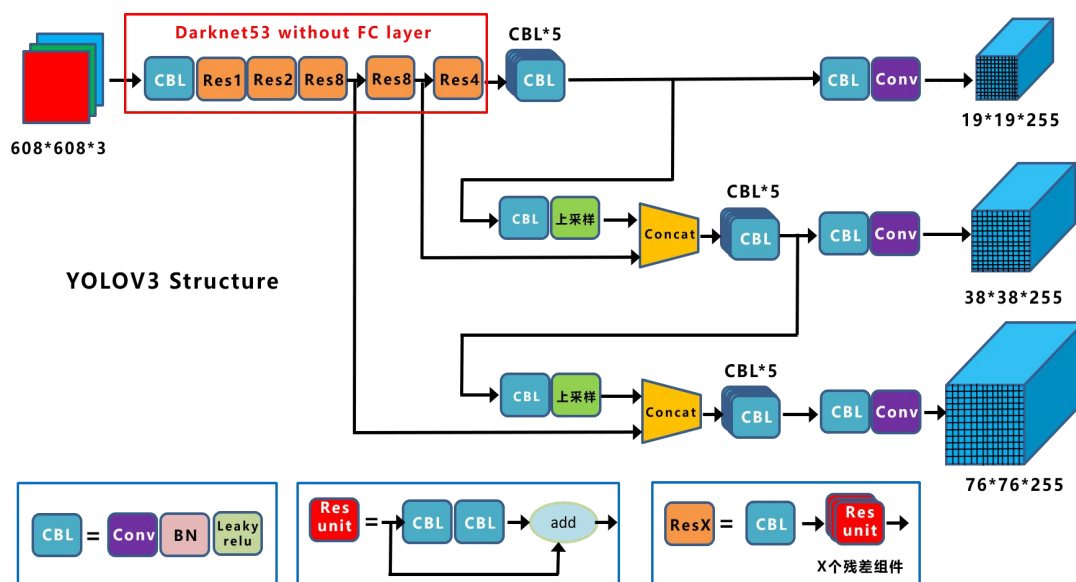


图 5-4 DarkNet53 网络的结构图

上图底部的三个蓝色框的内容为 Yolo v3 中的三个基本组件，如下：

(1) CBL：是 DarkNet53 网络结构的一个小组件。

(2) Res unit：借鉴了 Resnet 网络中所使用的残差结构，采用此结构可以在避免退化问题基础上构建的更深的网络层数。

(3) ResX：由一个 CBL 组件和多个残差组件组成，每个模块前面的 CBL 组件都会对输入的图像内容进行下采样操作。其中 DarkNet53 网络结构及主要解析如下图 5-5 所示。

装
订
线

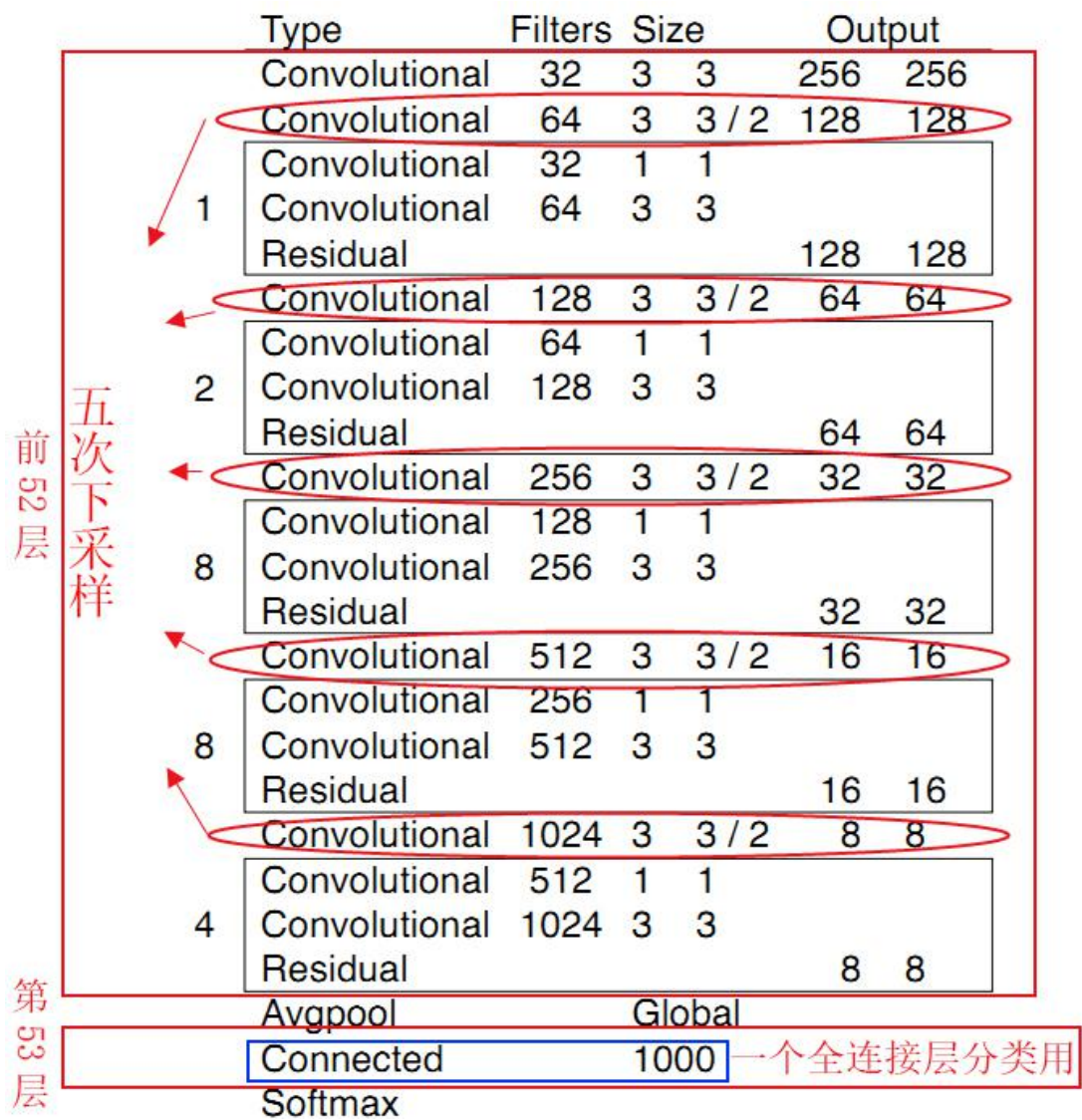


图 5-5 DarkNet53 网络结构及主要解析

5.2.2 训练过程

在了解 YOLO v3 算法的思想与 DarkNet53 网络模型的结构与运行流程后，并且用于训练的水果数据集已经制作完毕，接下来就可以着手用代码将功能实现了。

然后将完全处理好的数据集放入到源码指定位置，并针对个人的情形修改源码中的配置(如需要标注的物体种类数，每训练多少个 Epoch 后保存训练后的模型等)，之后就可以打开源码中的 train.py 文件，根据自己机器的配置及数据量修改单次训练所选取的样本数 Batch Size 以及循环训练的迭代次数 Epoch 等参数后，就可以运行 train.py 程序对自己的数据集进行训练了。由于水果图片标注功能的数据集需要手动标注，最后制作的数据集较小，所以采用了程序默认的配置，其中每迭代训练 100 轮后保存一次训练结果模型，一共迭代训练了 3200 次。训练过程中程序会载入自带的预训练模型权重，这样可以大大提高

训练的效率，如下图 5-6 所示。从图中可以印证了这一结论，可以看到，通过使用预训练模型权重进行训练，仅前几轮训练中，loss 值就已经到了 5%左右，极大地提高了训练效率。

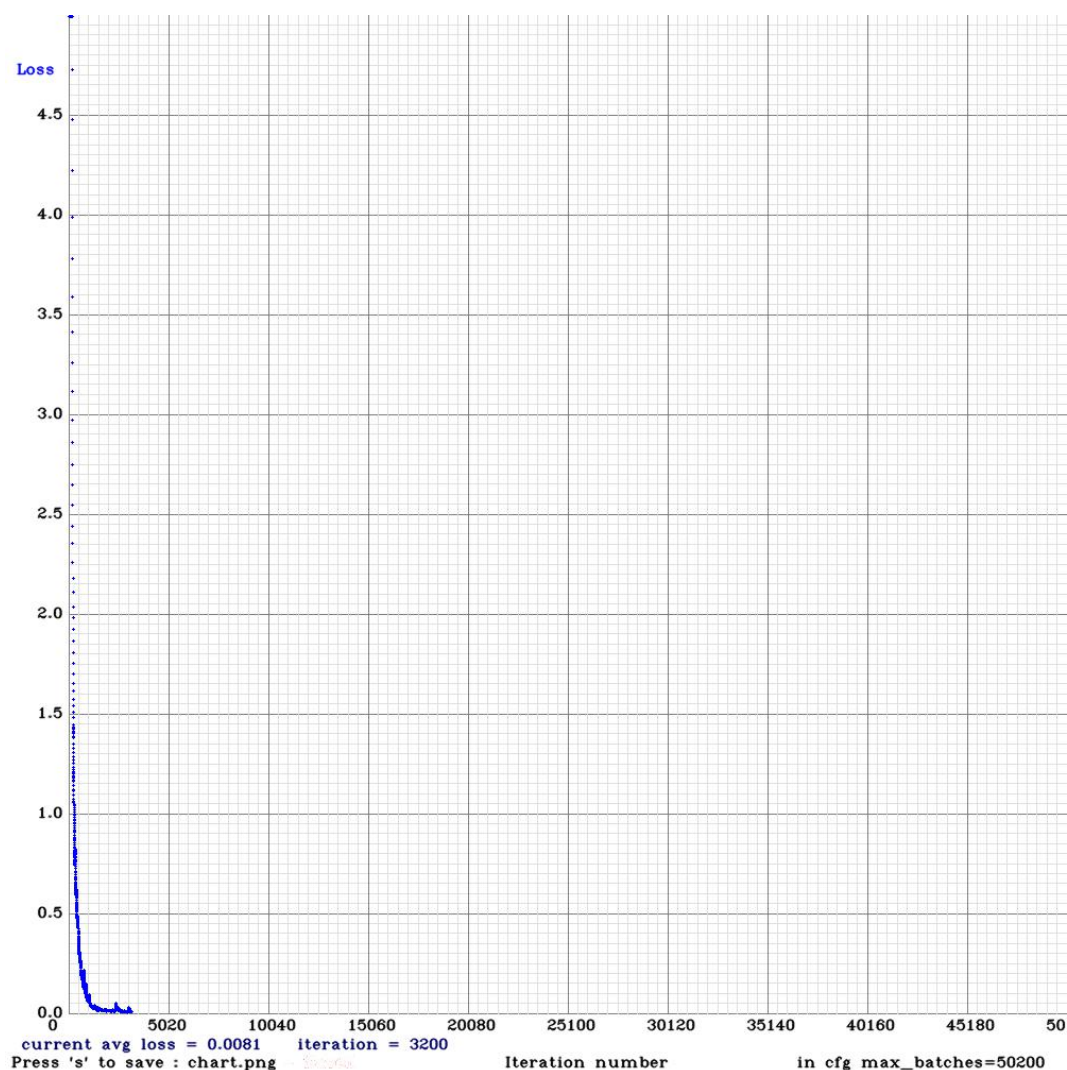


图 5-6 水果标注模型的训练情况

5.3 实验测试与分析

5.3.1 标注准确率

在上一步针对自己的数据集训练好模型后，就可以运行源码中的 predict.py 程序来检测训练的模型的效果了，编辑器的控制台中出入待检测的图片路径并回车，程序会对图片按照前文目标检测的流程进行运算，然后将结果输出到控制台中，并自动打开系统默认的图像浏览器来展示效果，如下图 5-7 所示，可以看到目标检测的效果还是十分强大的。

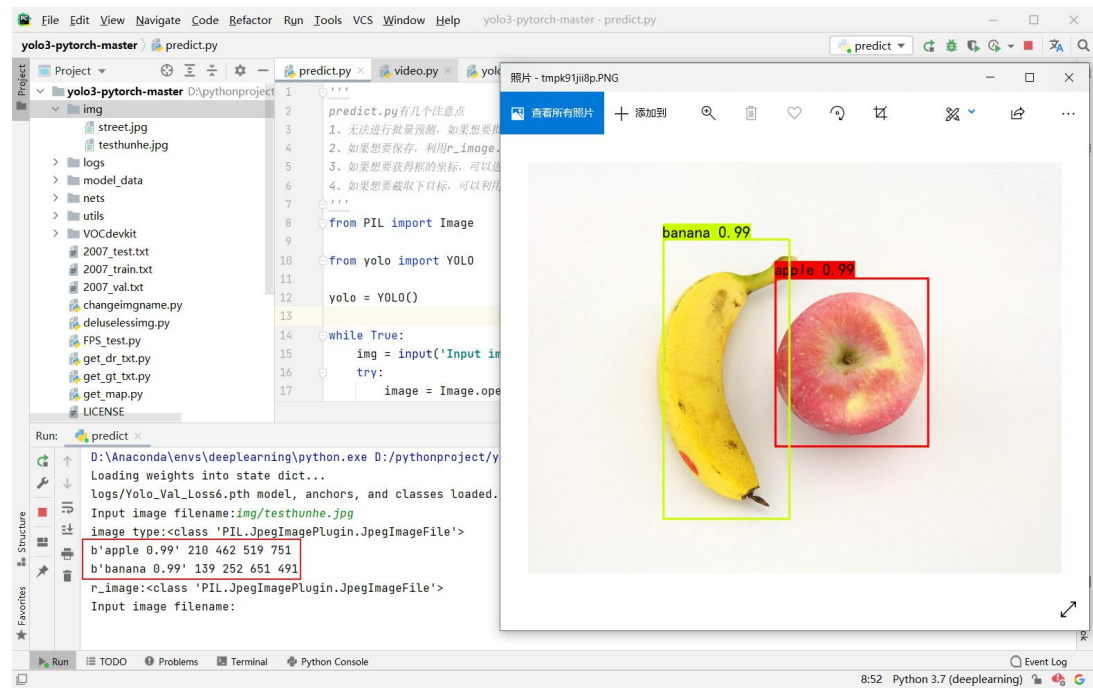


图 5-7 水果标注功能的实现效果

5.4 系统功能开发与展示

在验证了模型对目标检测的能力并符合要求后，将目标检测的功能代码封装成函数，接下来就可以将功能链接到项目的系统界面中来提升易用性了，大致流程如下：

- （1）在主窗口中 QStackedWidget 堆栈窗口控件中，对于水果图片标注这一层，链接到新创建的子程序的 SanUI2 类中。
- （2）对于单图模式，在选择图片按钮所链接的槽函数 openimage 中，使用 QFileDialog 类下的 getOpenFileName() 函数获得用户选择的图片的路径，声明全局变量去接收此路径。
- （3）然后点击运行按钮开始识别，在对应的槽函数 run() 中先进行判断(如路径是否为空)，再调用之前封装好的目标标注函数，并将得到返回的结果数据进行清洗与组装表头内容，最后将得到结果图片替换原来展示的图片，并将结果表输出到 QTextEdit 多行文本框控件中。如下图 5-8 所示。

对于多图模式，参考水果图片分类中多图模式的系统功能设计，在点击运行按钮后，需要在运行按钮对应的槽函数 multirun() 中先将运行按钮禁用，然后创建新的线程类，在线程类中的 run() 函数中创建循环来遍历目录中文件列表，并对遍历到的文件调用目标标注函数得到返回结果并整理数据与保存数据。其中线程类中需要创建两个信号量，一个在循环中进行发射，执行对应函数来实时更新界面中显示的当前执行标注的图片名称与进度数字，另一个信号在循环

结束后发射,执行对应函数来更按钮状态,将按钮变成可点击。如下图 5-9,5-10, 5-11, 5-12 所示。



图 5-8 水果标注功能单图模式系统效果图



图 5-9 水果标注功能多图模式运行时效果图

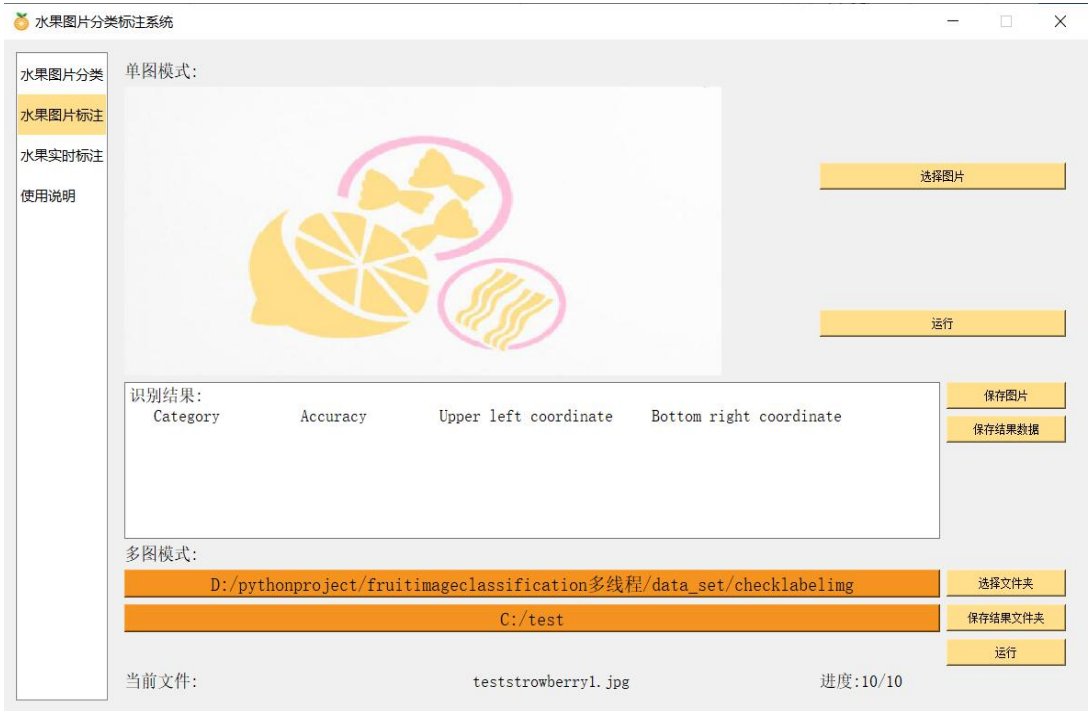


图 5-10 水果标注功能多图模式运行结束后效果图

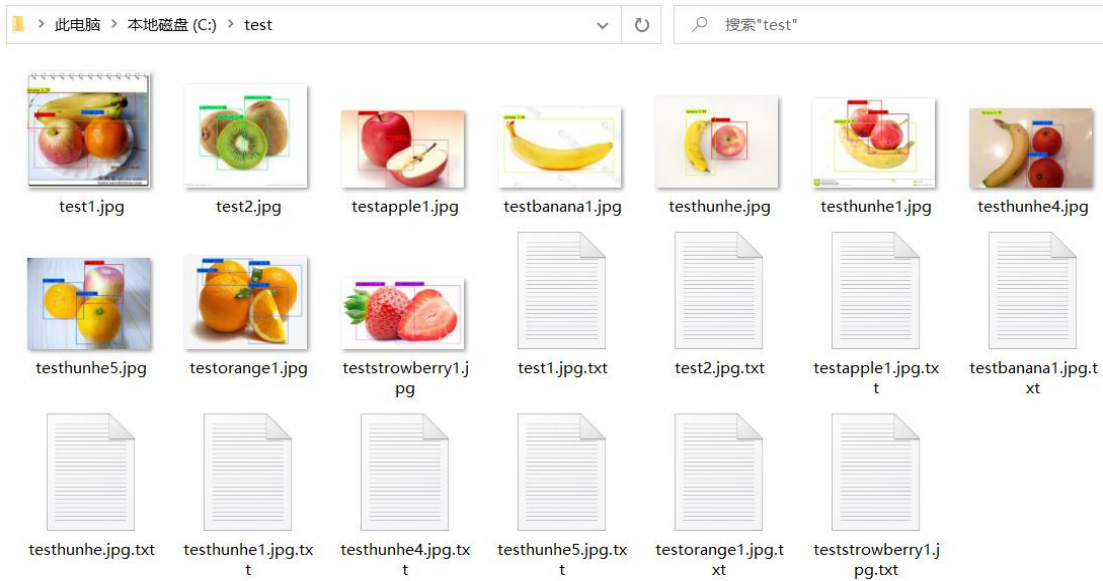


图 5-11 水果标注功能多图模式运行后生成标注结果文件

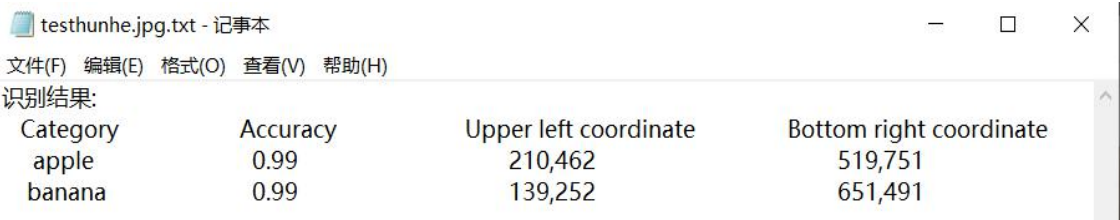


图 5-12 标注结果中的数据文件内容

6 水果实时标注

6.1 功能实现

由于实时标注需要对摄像头进行操作，并需要捕捉图像并对图像进行标注。而在图像处理方面，OpenCV 库无疑是最佳的图像处理工具，OpenCV 库是开源的跨平台计算机视觉库，有着轻量，高级的特点，并且 OpenCV 针对 Python 语言专门封装好的库 cv2，在 Anaconda 的命令行窗口中，进入到系统开发的环境，输入 `conda install opencv-python` 命令即可在开发环境下安装 cv2 库。安装好后创建个功能测试程序的 python 文件，在文件中引入 cv2 库，再引入 Python 中的图像处理库 PIL，通过 cv2 库中的 VideoCapture 函数创建调用摄像头并开启窗口显示画面的这个对象，用变量 capture 接收返回的对象，之后创建 fps 变量，用来检测调用摄像头采集图像并处理的效率，默认设初值为 0.0。之后创建 while 循环，先在外引入 Python 中的 time 库，在循环中使用 time 库中的 time 函数获取当前时间，之后使用 capture 对象的 read 方法读取摄像头某一帧画面，然后将此帧画面的格式进行转变，然后再将这一帧画面使用 PIL 库中的 Image.fromarray 函数将这一帧画面转成 Image 图片格式。之后将这张图片送到之前已封装好的目标检测函数中去进行检测，检测后需要再讲画面的格式进行转化，才能在窗口中看到正常的彩色画面。并且在这一次循环结尾再次获取此时的时间，通过计算循环前后时间差值并赋给 fps 变量，然后再使用 cv2 库中的 putText 函数即可将当前的 FPS 值显示到窗口。整个功能实现的代码如下：

```
import time
import cv2
import numpy as np
from PIL import Image
from yolo import YOLO
yolo = YOLO()
capture = cv2.VideoCapture(1)
fps = 0.0
while (True):
    t1 = time.time()
    # 读取某一帧
    ref, frame = capture.read()
    # 格式转变, BGRtoRGB
    frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
```

```
# 转变成 Image
frame = Image.fromarray(np.uint8(frame))
# 进行检测
frame = np.array(yolo.detect_imagevideo(frame))
# RGBtoBGR 满足 opencv 显示格式
frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
fps = (fps + (1. / (time.time() - t1))) / 2
print("fps= %.2f" % (fps))
frame = cv2.putText(frame, "fps= %.2f" % (fps), (0, 40),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
cv2.imshow("video", frame)
```

6.2 实验测试与分析

运行这个功能测试程序，程序会自动弹出测试窗口，窗口中实时的显示摄像头画面，并且当画面中存在待检测的五种水果时，程序会自动在画面中将水果进行标注，标注出水果种类与识别的准确率，并且此功能程序在运行时的帧率可以稳定的保持在 30 帧左右，而且画面切换还是十分流畅的，如下图 6-1 所示。

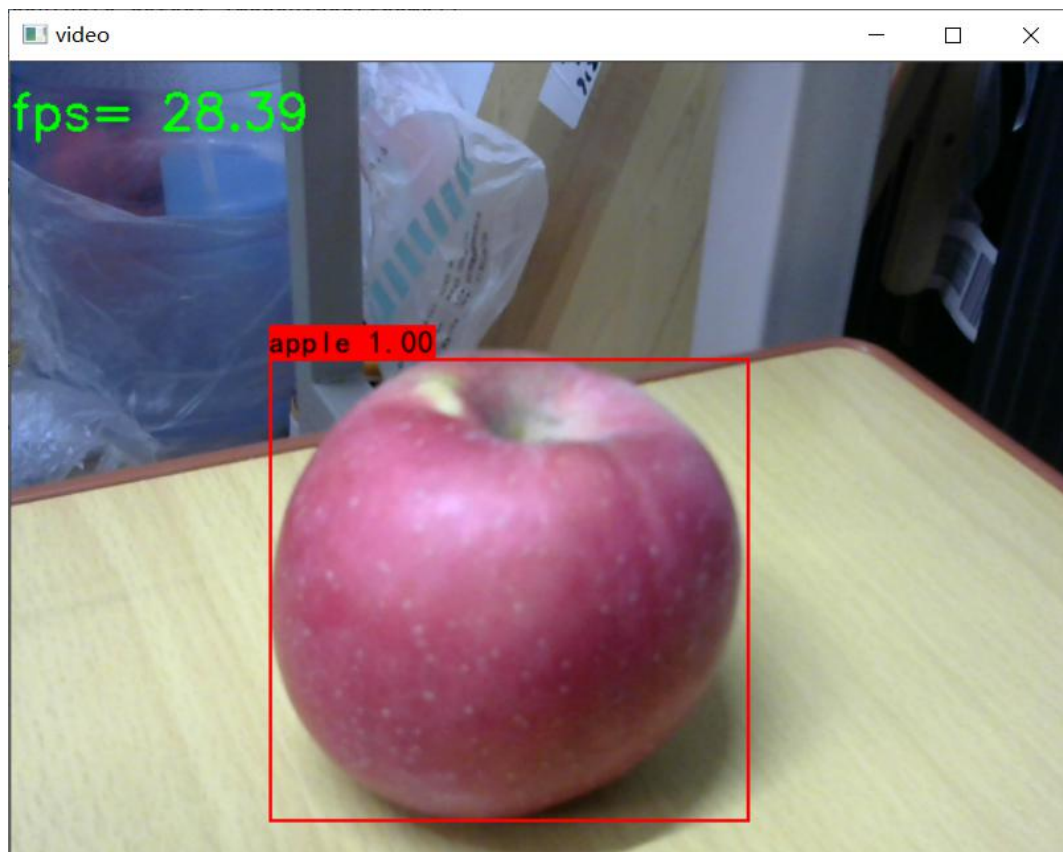


图 6-1 实时标注功能测试效果

6.3 系统功能开发与展示

在上一节运行了水果实时标注功能的测试程序，并对功能进行了测试，取得了比较令人满意的效果。所以本节我们要做的就是将功能实现的代码改写到之前预留的函数中，使这个功能能够在系统中方便的使用。

之后进入之前为水果实时标注所创建的 SanUI3() 类中，进入到开启检测这个按钮的槽函数 def run() 中，在前文中介绍过在 run 函数中当定时器计时完毕后执行采集摄像头画面并标注的函数 capPicture() 函数，这样就实现了每隔 100 毫秒获取一次摄像头画面标注的功能。然后将更改开启检测按钮的文本为“关闭检测”，此时再点击此按钮时，会进入 else 语句代码块，操作为关闭摄像头并修改按钮文本为“开启检测”。

所以现在需要实现 capPicture() 函数。在 capPicture() 函数中，前面的代码与功能

实现代码基本一致，只不过在返回结果图像时，由于图像的格式原因，无法直接将他放到 QLabel 标签内进行展示，需要使用 QImage 组件将图像转换成 PyQt 中可用的图像，然后再对 QLabel 标签使用 setPixmap 函数将标签内容设置为图像本身，即可实现将返回的结果图像展示到界面上，由于此函数是使用定时器每隔 100 毫秒进行调用，所以用户看到的效果就是调用摄像头实时采集画面，当画面中出现 5 种水果时，会在画面上将其框出，并在矩形框上显示框选出的水果种类与识别准确率，关键代码如下：

```
def capPicture(self):
    if (self.cap.isOpened()):
        ret, img = self.cap.read()
        height, width, bytesPerComponent = img.shape
        bytesPerLine = bytesPerComponent * width
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        frame = Image.fromarray(np.uint8(img))
        img = np.array(yolo.detect_imagevideo(frame))
        self.image = QImage(img.data, width, height,
                             bytesPerLine, QImage.Format_RGB888)
        self.label.setPixmap(QPixmap.fromImage(self.image).
                             scaled(self.label.width(), self.label.height()))
```

在系统功能开发完成后，就可以运行系统进行系统测试并查看效果了，水果实时标注效果如下图 6-2，6-3，6-4，6-5 所示。

可以看到，测试中为了保证实验的严谨性，特意调转摄像头位置，捕获水果的不同角度，并对获取的不同角度的图像进行检测与框选，可以发现，对于不同角度的图像，其检测后的结果均取得了极高的识别准确率，再次体现了在目标检测领域中，YOLO 算法的强大。

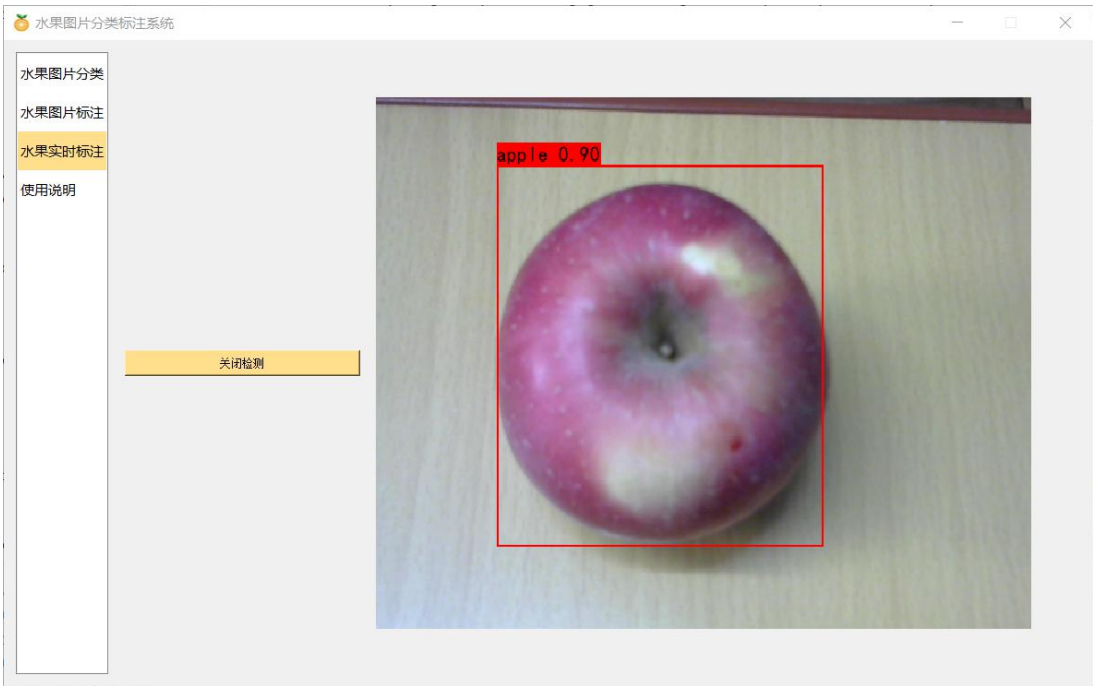


图 6-2 实时标注功能系统测试效果(俯视图 1)



图 6-3 实时标注功能系统测试效果(俯视图 2)

装
订
线

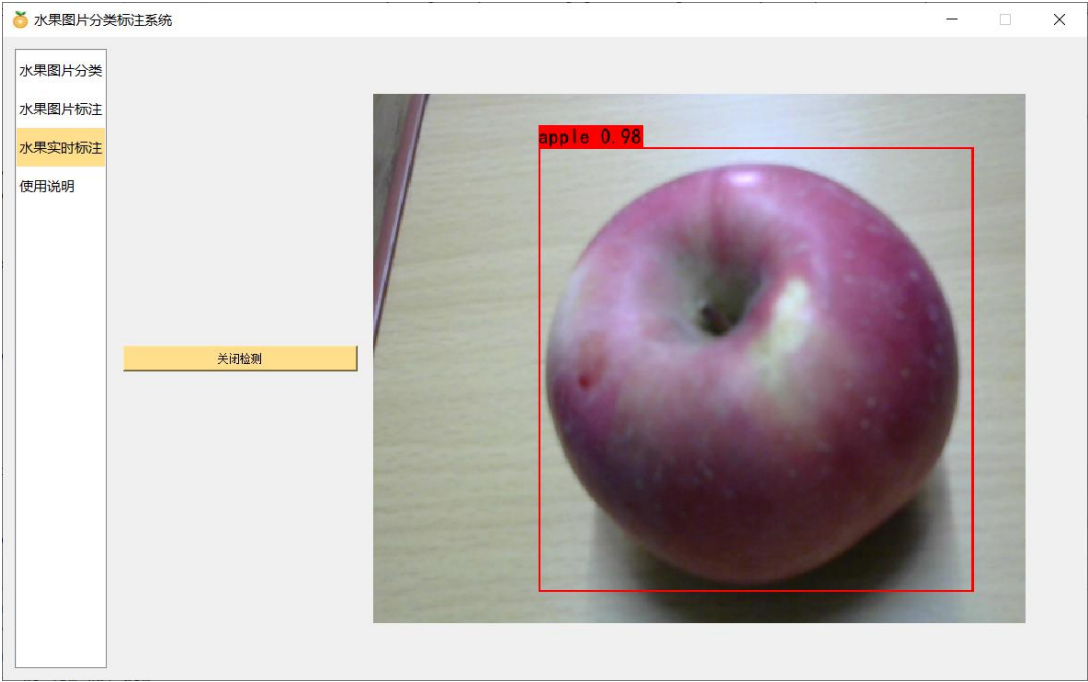


图 6-3 实时标注功能系统测试效果(侧视图 1)

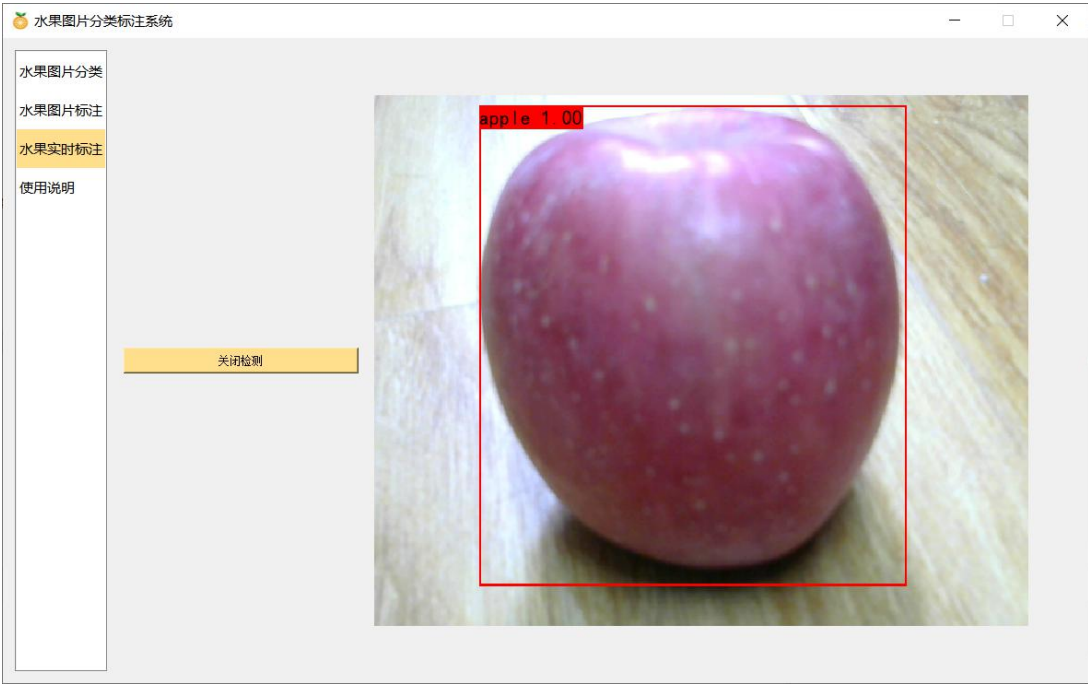


图 6-5 实时标注功能系统测试效果(侧视图 2)

7 总结与展望

7.1 总结

随着计算机视觉领域的不断发展,传统图像识别技术很难再满足日常生活中的应用需求,所以我们需要一种更有效的方式来对水果的种类进行精确的识别与分类,深度学习作为人工智能领域的重要组成部分,并且在蓬勃发展中引起了广泛的研究,在图像识别领域取得了重大的进展,基于深度学习的水果识别系统的实现具有重大意义。

通过实验使用深度学习技术对水果种类进行识别与标注,验证了相对于传统的识别方式这种方法具有优越的识别性能与泛化能力,可广泛应用于图形识别领域中。

7.2 展望

本文对深度学习的方法进行了一些分析与研究,并设计了基于深度学习的水果图片分类标注系统,并将卷积神经网络技术应用在了水果识别领域,通过实验取得了令人满意的效果,实现了课题的基本要求,但是由于时间与条件的限制,还有一些不足之处需要进一步完善。

深度学习需要大量的数据,由于选择水果样本的不足,利用深度学习方法进行水果图像的分类时,样本的数量会直接影响识别的精度,下一步可以扩充水果的数据集中图片的数量。

在实际的应用场景中,获取以及收集到图像由于光照、遮挡等因素的影响,而之前模型训练时此种情况的数据集图片数量过少,所以会导致在这种情况下的识别准确率会有所降低。在未来的实验中可以尝试更加深的网络,或者在这种特殊情况下导致相应的数据集体量不足时,可以对模型采用迁移学习的训练方式进行弥补。

致 谢

首先要感谢我的指导老师陈立，本论文是在老师的指导和帮助下修改完成的。在此，我要对导师的细心指导与帮助表示由衷的感谢。在这段时间里，我从导师的身上不仅学到了许多专业知识，更深刻的感受到导师在工作上的兢兢业业，诲人上的孜孜不倦，以及平易近人生活态度。所以导师严谨的治学态度和忘我的工作精神值得我去学习，今后不论是参加工作还是继续深造，我都要以导师为榜样，保持着对工作积极乐观的心态以及对学术上的严谨与敬畏。而且也非常希望在日后能通过这样坚持不懈的努力，为计算机行业做出自己的一份贡献。

同时还要感谢四年来其他传授我知识的老师们，毕业设计和论文是检验大学四年专业能力的一场“练兵”，没有一个强健的“体”与“魄”是通过不了这场检阅的，在此之前必须要拥有扎实的编程能力与计算机学科的理论基础。而这些显然不能一蹴而就的获得，要通过老师们四年来不断地在知识上对我的灌溉，才会让我在编写毕业设计程序与论文时能够厚积薄发，文思泉涌。所以也要感谢老师们对培养我专业能力与素养上的辛勤付出，同时也领悟到了要终身学习，尤其是计算机行业的迅速发展，技术迭代的迅速，更要在日后保持不断的学习，才能在专业上得到源源不断的活水，才能保证即便年龄增加，依旧能够保持在计算机行业学习与探索的活力。

最后还要感谢那些对我学习上支持和鼓励的人。再次感谢所有关心帮助过我的同学和老师，祝愿同学们学业有成，也祝愿老师们在学术上的造诣更上一层楼。

参考文献

- [1]梁礼明, 盛校棋, 郭凯, 邓广宏. 基于改进的 U-Net 眼底视网膜血管分割.江西理工大学, 2020(037)004.
- [2]王恒欢. 基于深度学习的图像识别算法研究[D]. 北京邮电大学, 2015.
- [3]张磊. 中国水果出口影响因素及竞争力的研究[D]. 江南大学, 2013.
- [4]陈源,张长江. 水果自动识别的 BP 神经网络方法[J]. 微型机与应用(22):40- 43,48.
- [5]蔡健荣, 赵杰文. 自然场景下成熟水果的计算机视觉识别. 农业机械学报, 2005, 36(2): 62-64.
- [6]陈兆勇.基于深度学习的小学数学图形与几何教学探析. 福州市湖滨小学, (2020)03-0077-02
- [7]Ramsden P , Martin E , Bowden J . School environment and sixth form pupils approaches to learning[J]. British Journal of Educational Psychology, 1989, 59(2):129- 142.
- [8]Newble D I , Entwistle N J . Learning styles and approaches: implications for medical education[J]. Medical Education, 1986, 20(3):162-175.
- [9]Entwistle, Noel. Styles of learning and approaches to studying in higher education[J]. Kybernetes, 2001, 30(5/6):593-603.
- [10]Biggs J B , Collis K F . The Psychological Structure of Creative Writing[J]. Australian Journal of Education, 1982, 26(1):59-70.
- [11]Hinton, G. E . Reducing the Dimensionality of Data with Neural Networks[J].
- [12]Faria F A, dos Santos J A, Rocha A, et al. Automatic Classifier Fusion for Produce Recognition[J]. Patterns and Images, Brazil, 2012: 20—25.
- [13]兰豫.创新与发展:教育技术标准与整合研究. 中国教育技术协会,2004
- [14]年年会综述[J]. 电化教育研究, 2005(01):3-8.
- [15]何玲, 黎加厚. 促进学生深度学习[J]. 现代教学, 2005(05):31-32.