# **在线图像篡改检测**

### 1. 任务要求

（1）了解Django框架。

（2）基于Django实现在线图像篡改检测。

### 2. 运行步骤

（1）环境配置

程序运行环境说明如表1所示。

表1 程序运行环境

|  |  |
| --- | --- |
| 项目 | 产品/版本 |
| 虚拟机 | VMware® Workstation 15 Pro |
| Ubuntu | Ubuntu 22.04.1 LTS |
| Anaconda | Anaconda3-2022.10-Linux-x86\_64 |
| Python | 3.10 |
| Django | 4.1 |
| Jquery | 3.5.1 |
| SQLite | 3.39.3 |
| Tensorflow | 2.10 |

（2）模型下载

由于模型较大，并未放入虚拟机中（采用了共享文件夹的方式）。下载模型后请将imdNet/upimg/utils.py中的model\_path改为模型所在文件夹路径。

（3）运行程序

在终端中进入与manage.py文件同级目录，输入命令如下：

conda activate imd

python manage.py runserver

程序输出如图1，表示服务器现在正在运行。

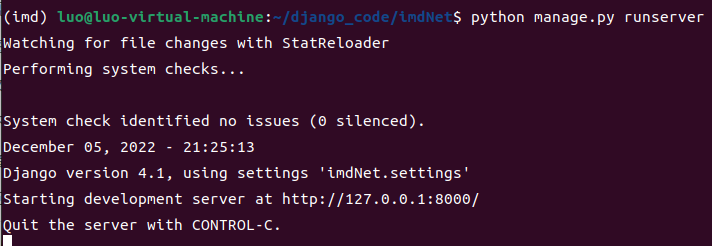


图1 启动程序

通过浏览器访问 <http://127.0.0.1:8000/>，进入应用页面。

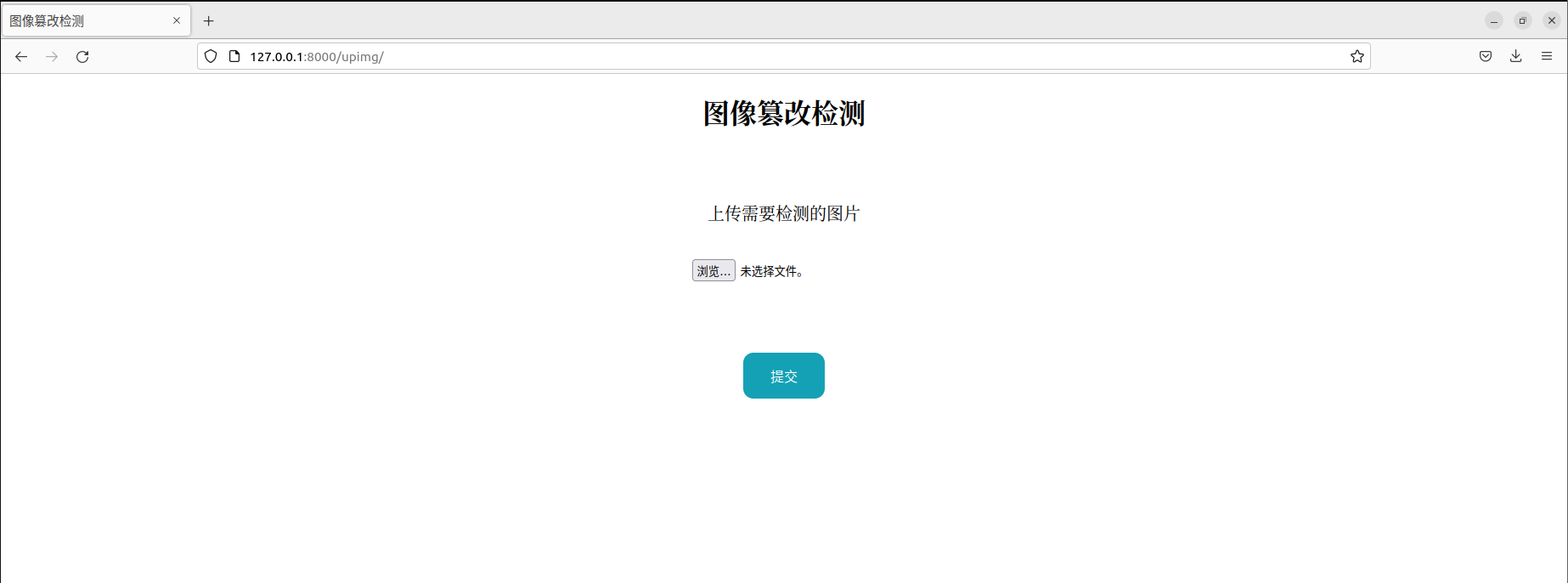
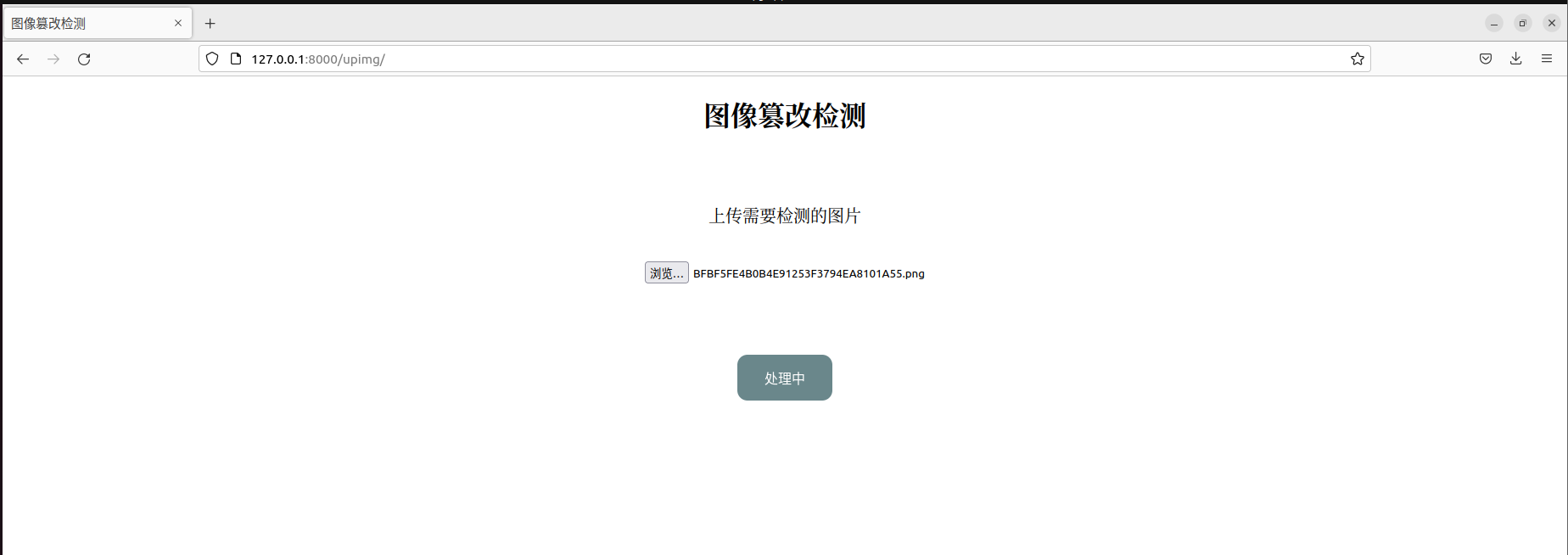
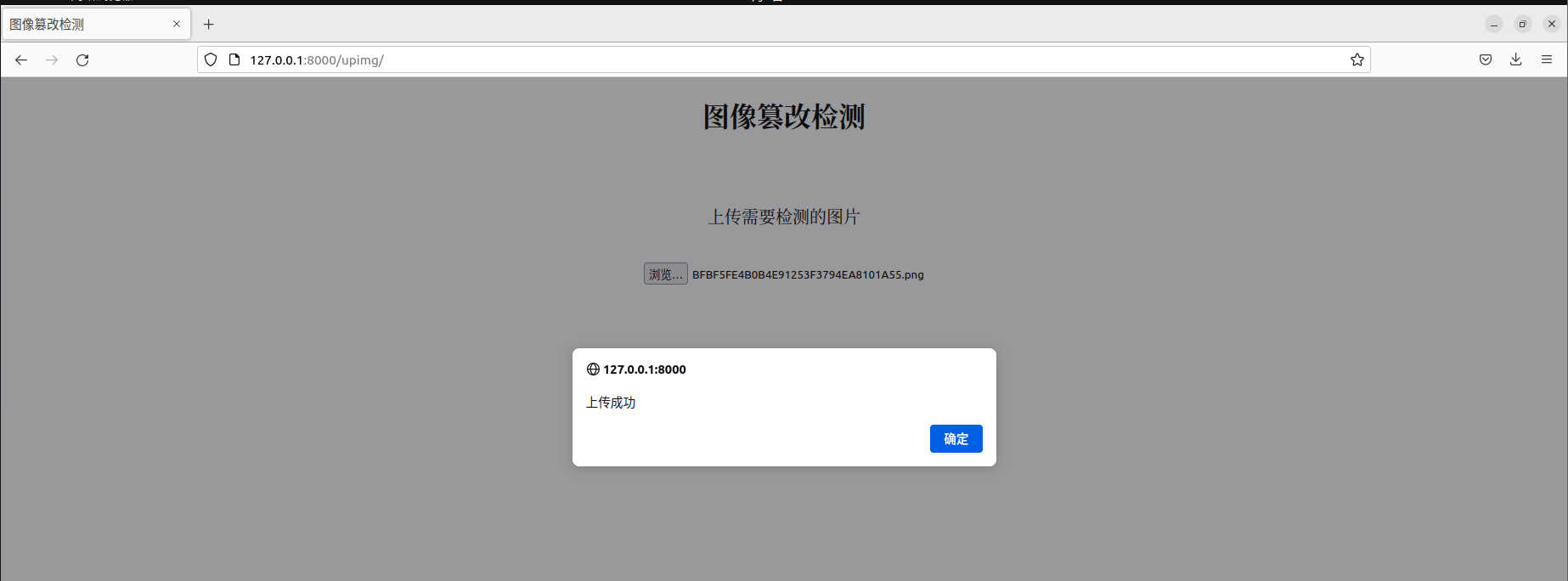


图2 浏览器访问页面

上传图片。





组图3 上传图片

获得检测结果。

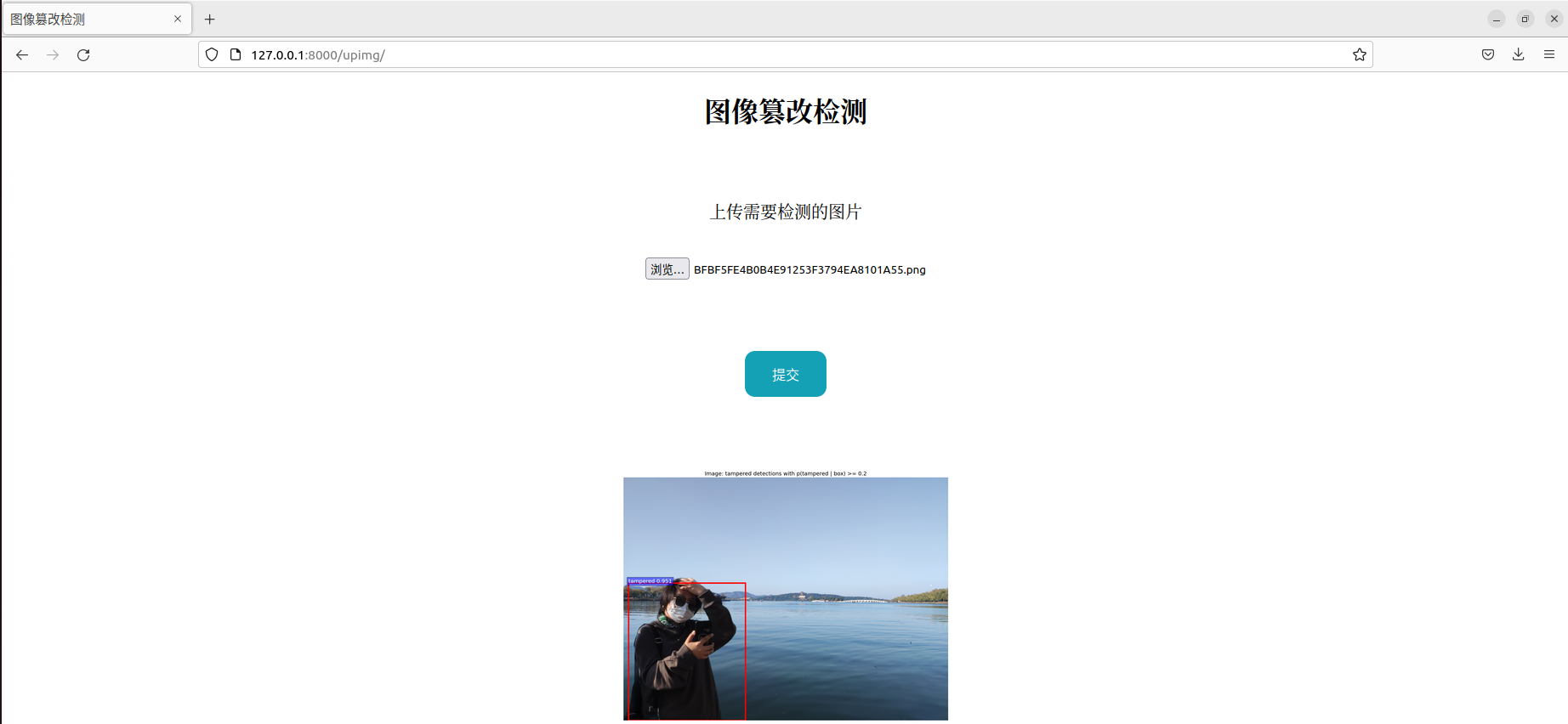


图4 获得检测结果

如果上传不支持的文件格式，则出现弹窗。

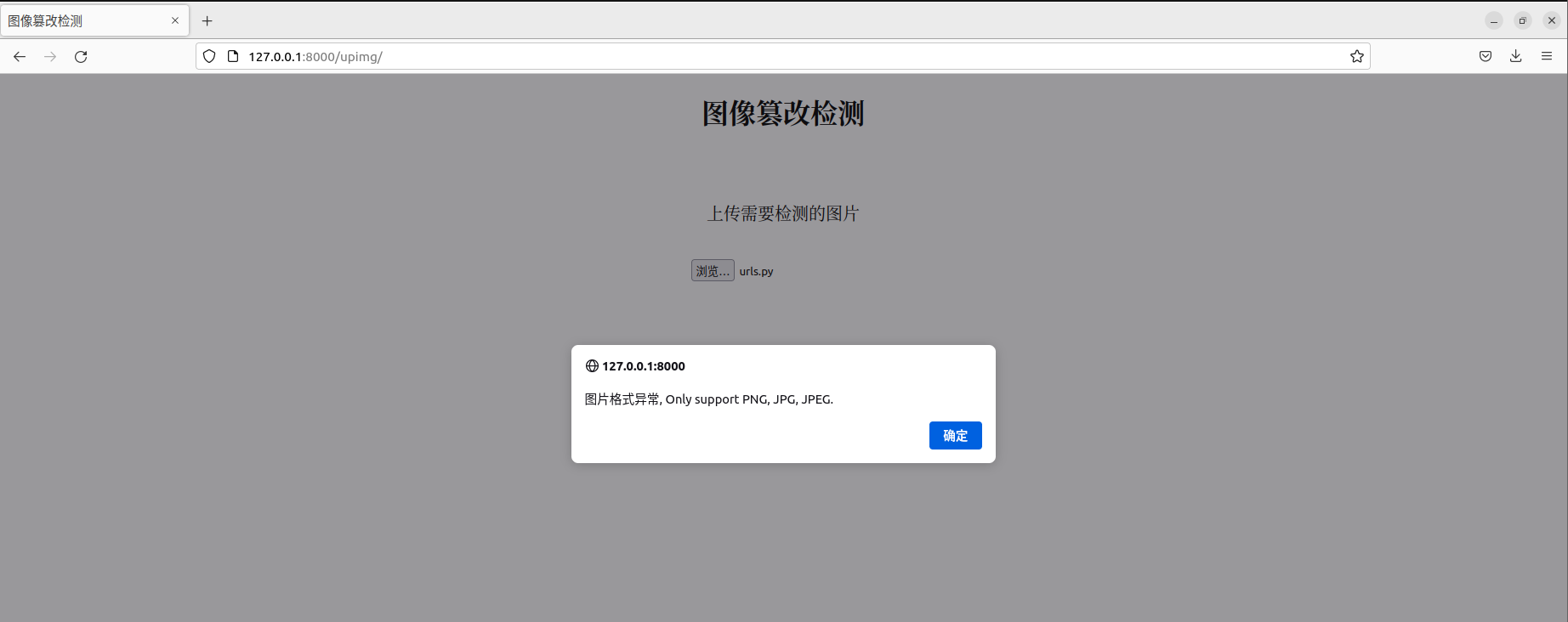


图5 文件格式不支持提示

### 3. 相关原理

#### 3.1 Django

Python下有许多款不同的Web框架，Django是具有代表性的一个。它是一个由Python写成的开源Web应用框架，许多成功的网站和APP都基于Django。

Django本身基于MVC模型，即Model（模型）+ View（视图）+ Template（模板）设计模式。其中，Model负责业务对象与数据库的映射，视图负责与用户的交互，Template负责把页面展示给用户，View负责业务逻辑，并在适当时候调用Model和Template。除了以上三层之外，还需要一个URL分发器，它的作用是将URL的页面请求分发给不同的View处理。MVT响应模式如图6所示。MVT具有低耦合、开发快捷、部署方便、可重用性高、维护成本低等优势。

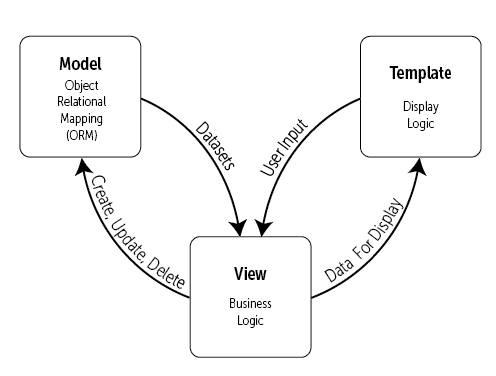


图6 MVT模型

Django框架如下：

项目名称

├─项目名称

│ ├─asgi.py

│ ├─wsgi.py

│ ├─settings.py

│ └─urls.py

├─manage.py

├─应用1

└─…

* manage.py一个可用各种方式管理Django项目的命令行工具。
* settings.py包含所有的网站设置。这是可以注册所有创建的应用的地方，也是静态文件，数据库配置的地方，等等。
* urls.py定义了网站url到view的映射。
* asgi.py一个ASGI兼容的Web服务器的入口
* wsgi.py一个WSGI兼容的Web服务器的入口。

#### 3.2 图像篡改检测

在篡改技术中，拼接（splicing）、复制-移动（copy-move）和移除（removal）是最常见的操作。图像拼接从真实图像中复制区域并将其粘贴到其他图像中。复制-移动操作选取某一图像中的区域，复制粘贴给自身。移除操作从真实图像中消除区域并进行绘制。图7是这三种操作的简单示例，其中每行对应一种篡改方法；第一列为真实图像，第二列为篡改后的图像，第三列为篡改区域的掩码。

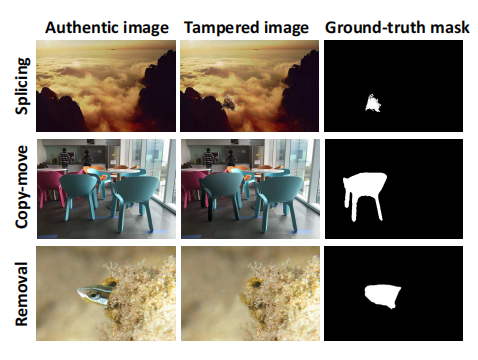


图7 图像篡改示例

随着多媒体采集设备的快速发展和众多操作简单的图像编辑软件的出现，普通人也能很轻易地对图像进行加工和修改。图8所示为利用IpasOS16自动提取图像主体，从而实现图像拼接的示例。



图8 利用IpadOS16实现图像拼接

随着计算机和互联网的飞速发展，数字图像的存储和传递也变得简单，多媒体已经成为信息承载与共享的重要途径。日常生活中人们对图像进行修改，往往是出于美化、娱乐的目的；但是在有些情况下，被恶意篡改的图像经过传播，会影响人们对客观事物的判断，有时甚至会对社会和国家造成不良的影响。

Zhou等人提出了一种[双流图像篡改检测框架RGB-N](https://openaccess.thecvf.com/content_cvpr_2018/papers/Zhou_Learning_Rich_Features_CVPR_2018_paper.pdf)，如图9所示，它不仅建模了视觉篡改，而且还捕获了局部噪声特征的不一致性。其中视觉信息为RGB格式的图像信息，噪声指一个像素的值和该像素的估计值（使用周围像素产生的插值）之间的残差。

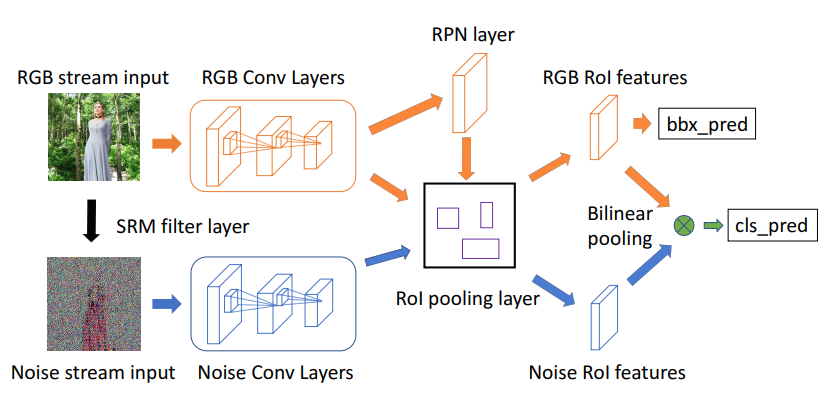


图9 双流图像篡改检测框架RGB-N

### 4. 核心程序

本程序使用Python在Ubuntu 22.04环境下开发，使用Anaconda创建虚拟环境。程序的处理流程如图10所示。

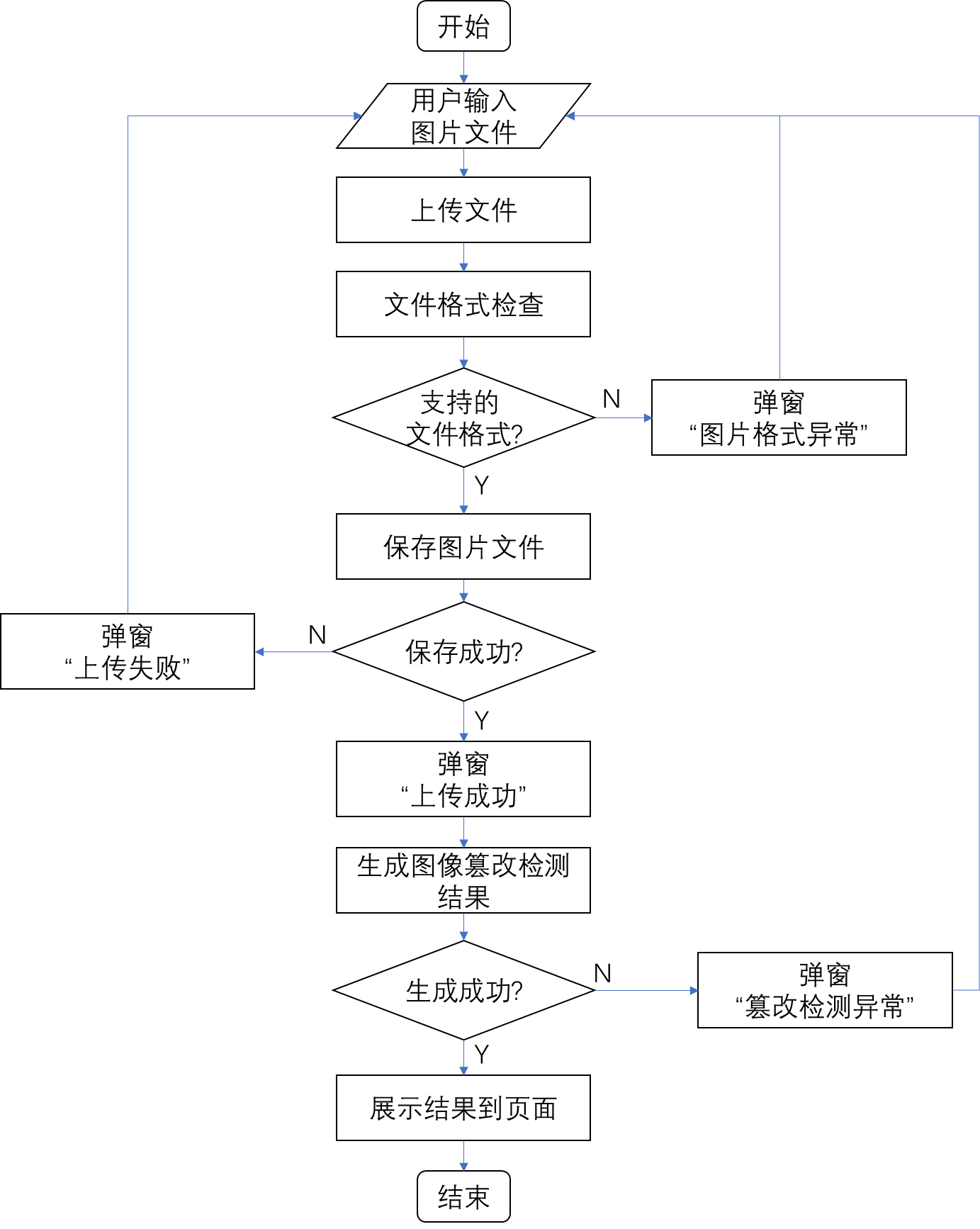


图10 在线图像篡改检测程序流程

程序源码重要文件列表如下：

**imdNet**

├─**imdNet**

│ ├─settings.py

│ └─urls.py

├─manage.py

├─**media** 存放媒体资源

│ ├─**upload** 存放用户上传的图片.

│ └─**download** 存放篡改检测生成的图片.

├─**static** 存放静态资源，如jquery.

│ └─css 存放css文件

├─**upimg** 核心app.

│ ├─models.py 设置数据表.

│ ├─urls.py 设置url与view的映射.

│ ├─views.py 编写业务逻辑.

│ ├─utils.py 为views提供一些辅助函数.

│ ├─imd.py 运行图像检测模型，生成检测结果.

│ ├─**\_lib** 图像检测模型辅助代码.

│ ├─**templates** 存放html文件.

│ └─**migration** 执行数据迁移生成的文件夹.

└─db.sqlite 数据库文件.

核心程序及注释如下。

**# /upimg/views.py**

from django.shortcuts import render

from .models import Image

from django.shortcuts import HttpResponse

def to\_img\_load(request):

'''

展示页面

'''

return render(request, 'img\_upload.html')

from django.http import JsonResponse

from io import BytesIO

from django.core.files.uploadedfile import InMemoryUploadedFile

from django.conf import settings

import os

from .utils import \*

from .imd import imd

from PIL import Image as PILImage

def image\_upload(request):

'''

处理上传图片，存入数据库并返回相应信息

'''

## 获取生成的图像，检查文件格式，重命名

try:

img\_s = request.FILES['img'] # 获取文件对象

img\_s.name = get\_new\_random\_file\_name(img\_s.name) # 检查文件格式，重命名

# 图片格式异常

except ImgTypeError as err:

return JsonResponse({"data":2, "info":err.errorinfo}, json\_dumps\_params={'ensure\_ascii':False}, safe=False)

# 上传失败

except Exception as err:

print(err)

return JsonResponse({"data":0}, json\_dumps\_params={'ensure\_ascii':False}, safe=False)

## 保存上传的图像，生成篡改检测图像，保存并返回

try:

# 保存数据

image = Image(name = img\_s.name, img = img\_s)

image.save()

#img\_rec = Image.objects.get(name = img\_s.name)

# 生成响应图片

imd(imgs=[img\_s.name],

load\_path = os.path.join(settings.MEDIA\_ROOT, image.img\_path),

save\_path = os.path.join(settings.MEDIA\_ROOT, image.img\_new\_path),

model\_path = model\_path

)

imd\_new\_file = os.path.join(image.img\_new\_path, img\_s.name)

if os.path.exists(os.path.join(settings.MEDIA\_ROOT, imd\_new\_file)):

image.img\_new = imd\_new\_file

image.save()

# 图像篡改检测异常

else:

return JsonResponse({"data":3}, json\_dumps\_params={'ensure\_ascii':False}, safe=False)

return JsonResponse({"data":1, "info":str(image.img\_new)}, json\_dumps\_params={'ensure\_ascii':False}, safe=False)

# 上传失败

except Exception as err:

print(err)

return JsonResponse({"data":0}, json\_dumps\_params={'ensure\_ascii':False}, safe=False)

图像检测ER图如图11所示。

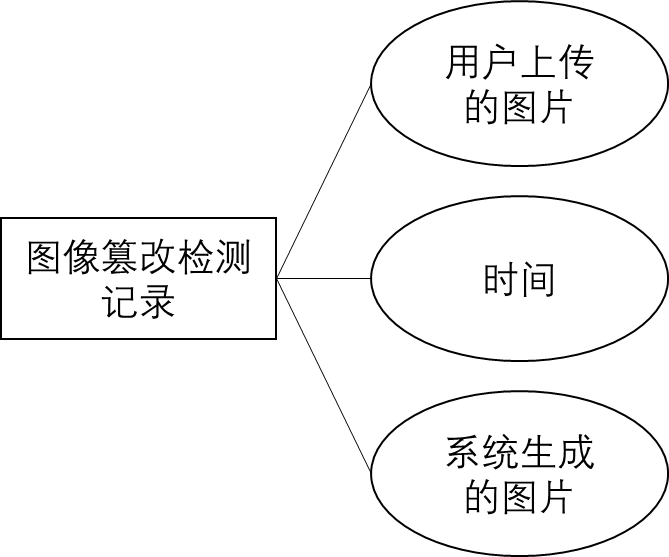


图11 图像篡改检测ER图

在数据库中记录如图12所示。

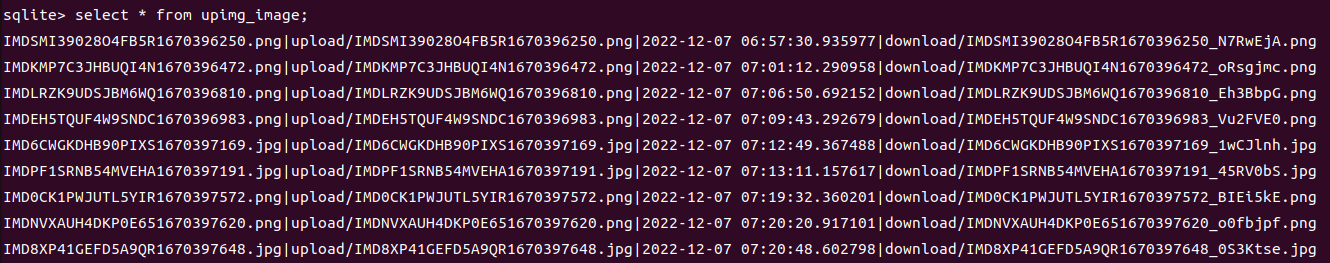


图12 数据库记录

### 附录

（1）数据迁移

数据库使用Ubuntu自带的SQLite，若imdNet文件夹下不含db.sqlite3文件，请执行以下命令：

python manage.py makemigrations

python manage.py migrate

执行结果如图13所示。

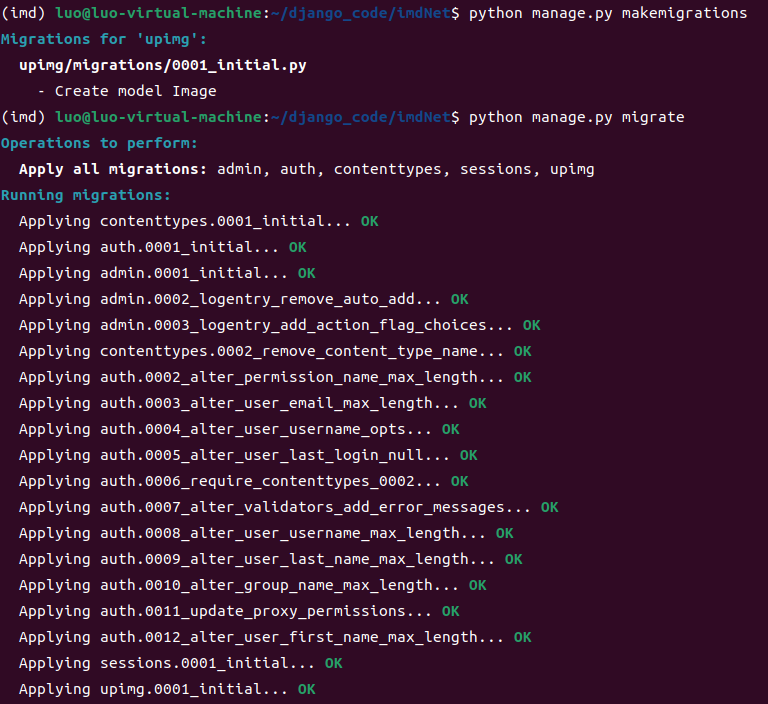


图13 数据迁移执行结果

（2）代码链接

图像篡改检测模型链接: [https://pan.baidu.com/s/1Mp5a56H4CP9Et5yDu0GDnQ?  
pwd=qrzj](https://pan.baidu.com/s/1Mp5a56H4CP9Et5yDu0GDnQ?pwd=qrzj) 提取码: qrzj 。

代码链接：[C-ljy/Online\_Image\_Manipulation\_Detection: 在线图像篡改检测 (github.com)](https://github.com/C-ljy/Online_Image_Manipulation_Detection)

虚拟机打包链接: <https://pan.baidu.com/s/1wXlRFeUwxec0Fa-EAJtOlw?pwd=j33f> 提取码: j33f

在ImageForTest文件夹中提供了一些可做测试的图片。链接: https://pan.baidu.com/s/1kh\_x7uP0RJ5ftpLzEZw84g?pwd=v7k1 提取码: v7k1