
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/utlis.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。

```
(ind) Luo@luo-virtual-machine:~/django_code/indNet$ python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
December 05, 2022 - 21:25:13
Django version 4.1, using settings 'indNet.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

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。

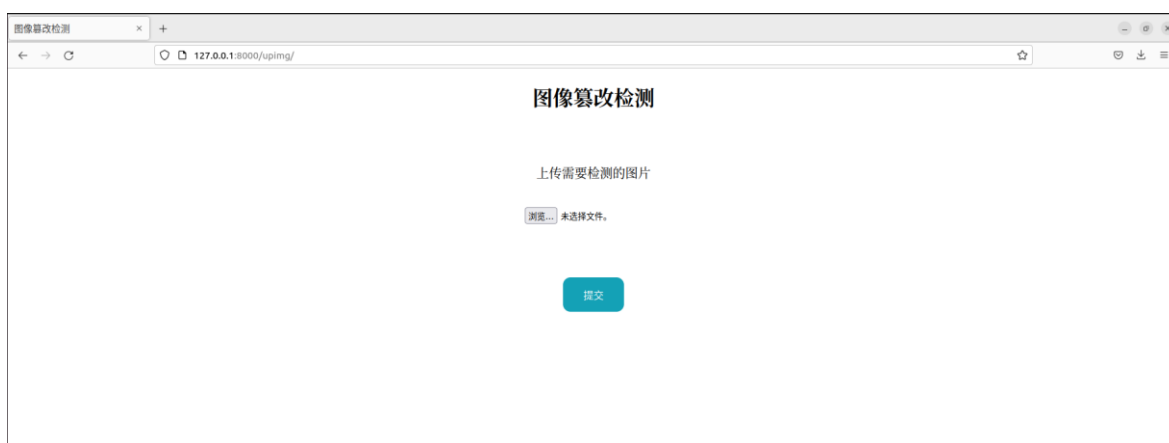
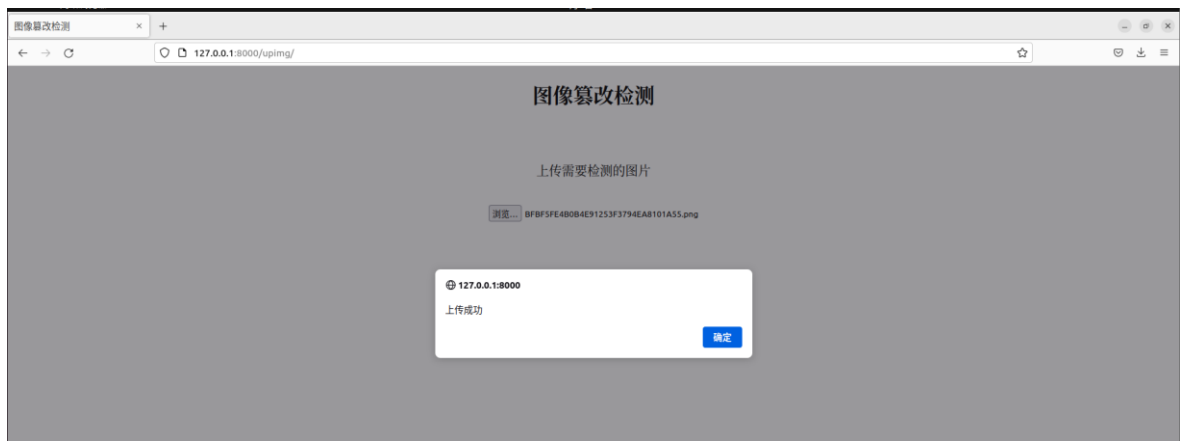


图 2 浏览器访问页面

上传图片。





组图 3 上传图片

获得检测结果。

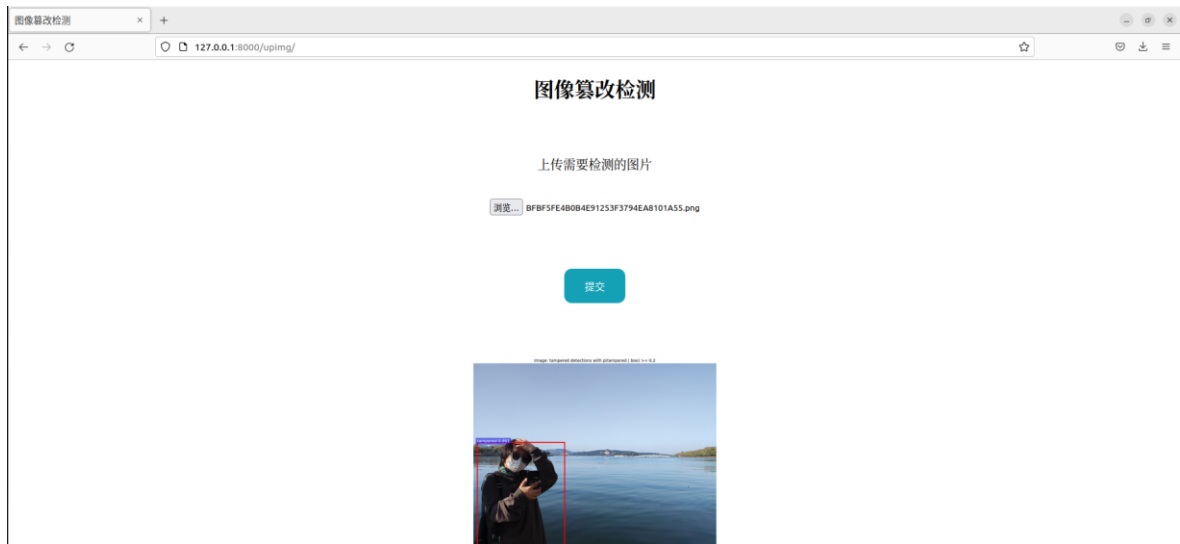


图 4 获得检测结果

如果上传不支持的文件格式，则出现弹窗。

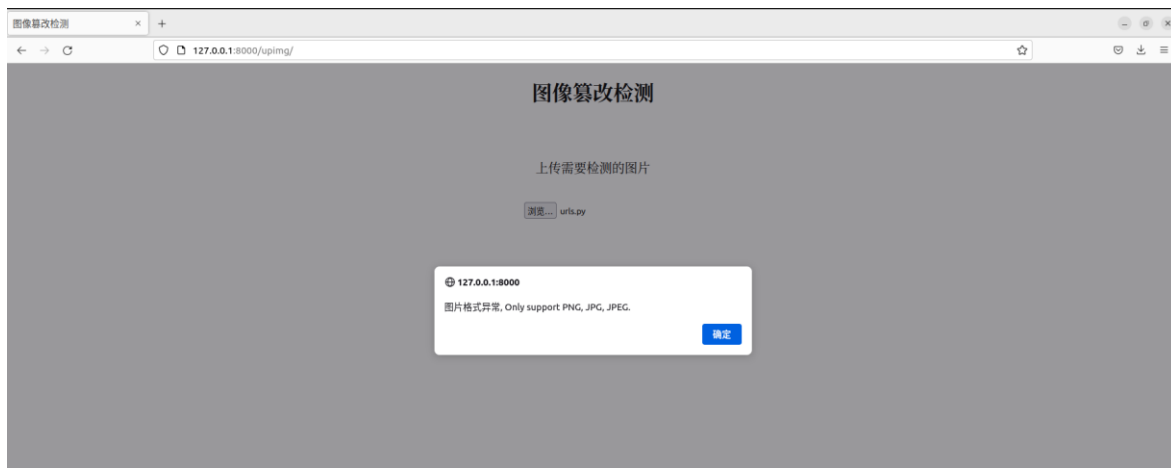


图 5 文件格式不支持提示

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。

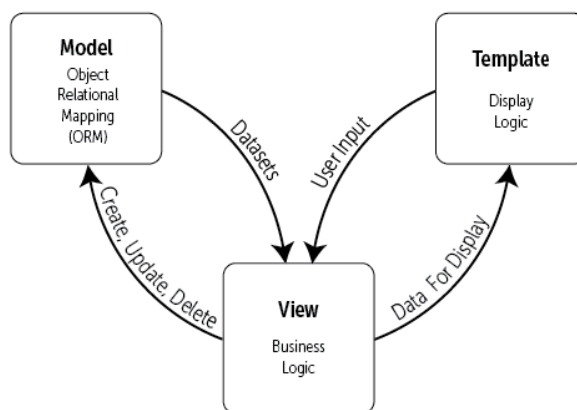


图 6 MVT 模型

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。

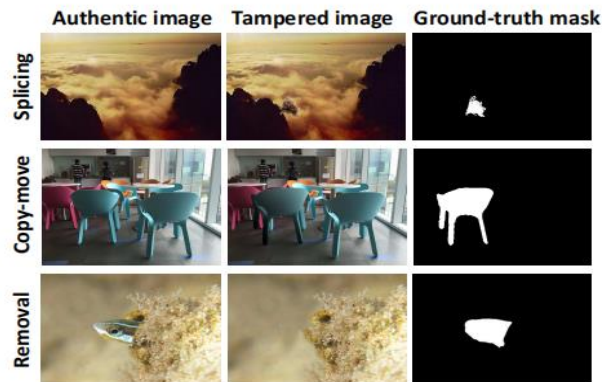


图 7 图像篡改示例

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 IpadOS16 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，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。

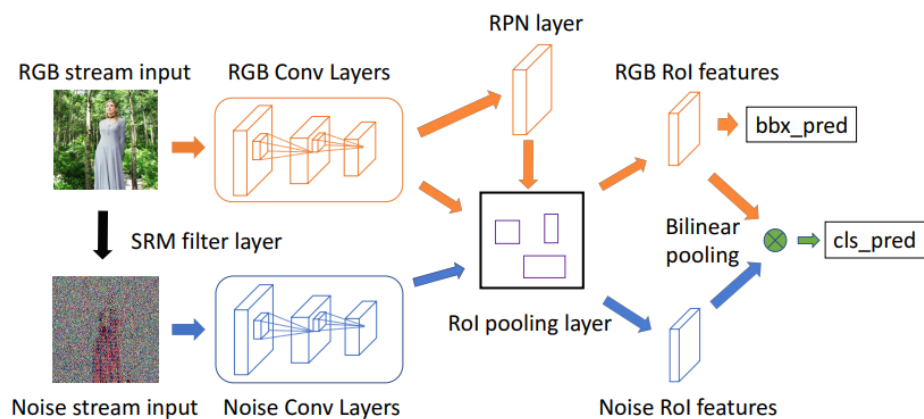


图 9 双流图像篡改检测框架 RGB-N

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。

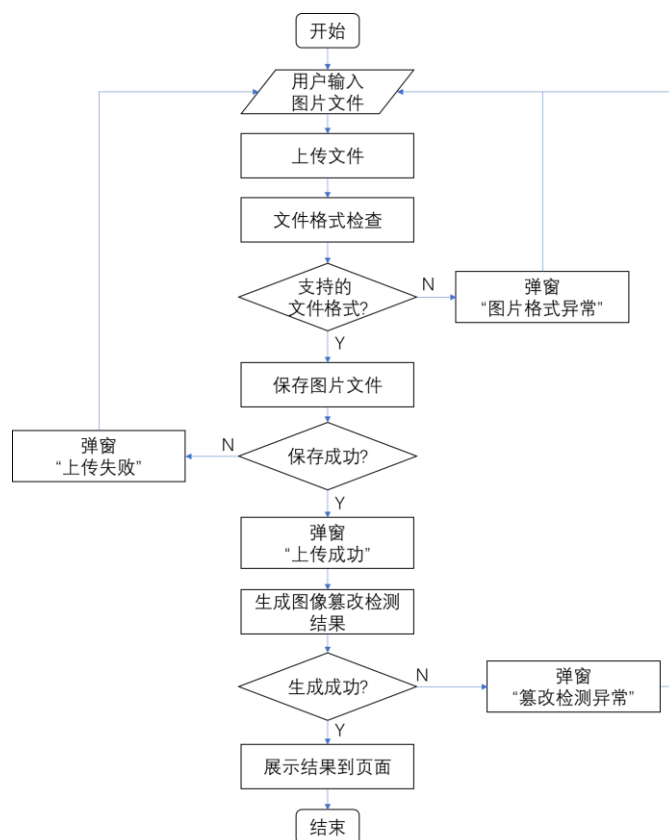


图 10 在线图像篡改检测程序流程

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。

```
1  # /upimg/views.py
2  from django.shortcuts import render
3  from .models import Image
4  from django.shortcuts import HttpResponseRedirect
5
6  def to_img_load(request):
7      '''
8      展示页面
9      '''
10     return render(request, 'img_upload.html')
11
```



```

12 from django.http import JsonResponse
13 from io import BytesIO
14 from django.core.files.uploadedfile import InMemoryUploadedFile
15 from django.conf import settings
16 import os
17 from .utils import *
18 from .imd import imd
19 from PIL import Image as PILImage
20
21 def image_upload(request):
22     '''
23     处理上传图片，存入数据库并返回相应信息
24     '''
25     ## 获取生成的图像，检查文件格式，重命名
26     try:
27         img_s = request.FILES['img'] # 获取文件对象
28         img_s.name = get_new_random_file_name(img_s.name) # 检查文件格式，重命名
29         # 图片格式异常
30     except ImgTypeError as err:
31         return JsonResponse({"data":2, "info":err.errorinfo},
32 json_dumps_params={'ensure_ascii':False}, safe=False)
33     # 上传失败
34     except Exception as err:
35         print(err)
36         return JsonResponse({"data":0},
37 json_dumps_params={'ensure_ascii':False}, safe=False)
38
39     ## 保存上传的图像，生成篡改检测图像，保存并返回
40     try:
41         # 保存数据
42         image = Image(name = img_s.name, img = img_s)
43         image.save()
44         #img_rec = Image.objects.get(name = img_s.name)
45
46         # 生成响应图片
47         imd(imgs=[img_s.name],
48             load_path = os.path.join(settings.MEDIA_ROOT, image.img_path),
49             save_path = os.path.join(settings.MEDIA_ROOT, image.img_new_path),
50             model_path = model_path
51         )
52         imd_new_file = os.path.join(image.img_new_path, img_s.name)
53
54         if os.path.exists(os.path.join(settings.MEDIA_ROOT, imd_new_file)):
55             image.img_new = imd_new_file

```

```

56         image.save()
57         # 图像篡改检测异常
58         else:
59             return JsonResponse({"data":3},
60 json_dumps_params={'ensure_ascii':False}, safe=False)
61
62         return JsonResponse({"data":1, "info":str(image.img_new)},
63 json_dumps_params={'ensure_ascii':False}, safe=False)
64         # 上传失败
65         except Exception as err:
66             print(err)
67             return JsonResponse({"data":0},
68 json_dumps_params={'ensure_ascii':False}, safe=False)

```

图像检测 ER 图如图 11 所示。

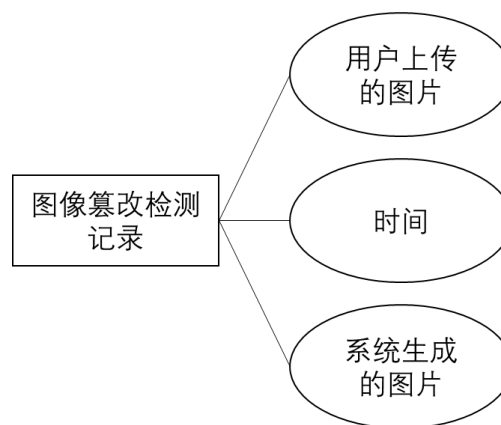


图 11 图像篡改检测 ER 图

在数据库中记录如图 12 所示。

```

sqlite> select * from uping_image;
IMDSMI3902804FB5R1670396250.png|upload/IMDSMI3902804FB5R1670396250.png|2022-12-07 06:57:30.935977|download/IMDSMI3902804FB5R1670396250_N7RwEjA.png
IMDKMP7C3JHBUQI4N1670396472.png|upload/IMDKMP7C3JHBUQI4N1670396472.png|2022-12-07 07:01:12.290958|download/IMDKMP7C3JHBUQI4N1670396472_oRsgjmc.png
IMDLRZK9UDSJBW6WQ1670396810.png|upload/IMDLRZK9UDSJBW6WQ1670396810.png|2022-12-07 07:06:50.692152|download/IMDLRZK9UDSJBW6WQ1670396810_Eh3BbpG.png
IMDEH5TQUF4W9SND1670396983.png|upload/IMDEH5TQUF4W9SND1670396983.png|2022-12-07 07:09:43.292679|download/IMDEH5TQUF4W9SND1670396983_Vu2FVE0.png
IMD6CWGKDBH90PIX51670397169.jpg|upload/IMD6CWGKDBH90PIX51670397169.jpg|2022-12-07 07:12:49.367488|download/IMD6CWGKDBH90PIX51670397169_1wCJlnh.jpg
IMDPF1SRNB54MVEHA1670397191.jpg|upload/IMDPF1SRNB54MVEHA1670397191.jpg|2022-12-07 07:13:11.157617|download/IMDPF1SRNB54MVEHA1670397191_45RV0bS.jpg
IMD0CK1PWJUTLSYIR1670397572.png|upload/IMD0CK1PWJUTLSYIR1670397572.png|2022-12-07 07:19:32.360201|download/IMD0CK1PWJUTLSYIR1670397572_0IEt5kE.png
IMDNVXAUH4DKP0E651670397620.png|upload/IMDNVXAUH4DKP0E651670397620.png|2022-12-07 07:20:20.917101|download/IMDNVXAUH4DKP0E651670397620_o0fbjpf.png
IMD8XP41GEFD5A9QR1670397648.jpg|upload/IMD8XP41GEFD5A9QR1670397648.jpg|2022-12-07 07:20:48.602798|download/IMD8XP41GEFD5A9QR1670397648_0S3Ktse.jpg

```

图 12 数据库记录

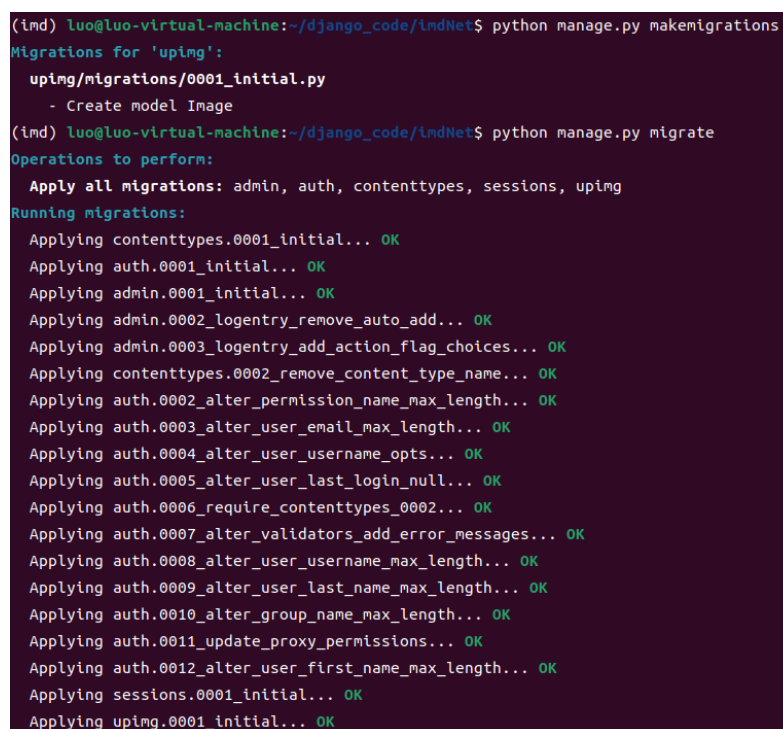
附录

(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。



```
(imd) luo@luo-virtual-machine:~/django_code/imdNet$ python manage.py makemigrations
Migrations for 'uping':
  uping/migrations/0001_initial.py
    - Create model Image
(imd) luo@luo-virtual-machine:~/django_code/imdNet$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, uping
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
  Applying uping.0001_initial... OK
```

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> 提取码: qrzj 。

代码链接: C-ljy/Online_Image_Manipulation_Detection: 在线图像篡改检测 (github.com)

虚拟机打包链接: <https://pan.baidu.com/s/1wXIRFeUwxec0Fa-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