Slice Captcha WriteUp

打开为swagger，其中包含两个接口

具体参数含义可查看swagger的Model获得

/captcha

获取验证码接口，返回以base64编码的图片，其中缺口图片为打乱的，需要进行复原

/validate

提交验证接口

思路

打乱顺序

多次获取可发现打乱顺序是固定的，图像大小也是固定的 `260x160` ，手工复原可发现每个分块大小为 `10x80`

可根据给出的 `src_bg` 和 `suffle_bg` 比对得到每个分块正确的位置

```python
from PIL import Image
# 获取完好的分块
block_width = 10
block_height = 80
src_img = Image.open("src_bg.png")
width, height = src_img.size
src_block_list = []
for y1 in range(0, height, block_height):
    for x1 in range(0, width, block_width):
        block = src_img.crop((x1, y1, x1 + block_width, y1 + block_height))
        block_list.append(block)
# 获取打乱的分块
src_img = Image.open("suffle_bg.png")
width, height = src_img.size
suffle_block_list = []
for y1 in range(0, height, block_height):
    for x1 in range(0, width, block_width):
        block = src_img.crop((x1, y1, x1 + block_width, y1 + block_height))
        block_list.append(block)
# 比对获取对应关系
for suffle_idx in range(len(suffle_block_list)):
    for src_idx in range(len(src_block_list)):
        if src_block_list[src_idx] == suffle_block_list[suffle_idx]:
            print(f"suffle_idx: {suffle_idx}, src_idx: {src_idx}")
            break
```

获得了分块对应关系

```
suffle_idx: 0, src_idx: 37
suffle_idx: 1, src_idx: 42
suffle_idx: 2, src_idx: 3
suffle_idx: 3, src_idx: 33
suffle_idx: 4, src_idx: 34
suffle_idx: 5, src_idx: 12
suffle_idx: 6, src_idx: 43
suffle_idx: 7, src_idx: 2
suffle_idx: 8, src_idx: 5
suffle_idx: 9, src_idx: 41
suffle_idx: 10, src_idx: 8
suffle_idx: 11, src_idx: 13
suffle_idx: 12, src_idx: 46
suffle_idx: 13, src_idx: 9
suffle_idx: 14, src_idx: 22
suffle_idx: 15, src_idx: 31
suffle_idx: 16, src_idx: 25
suffle_idx: 17, src_idx: 44
suffle_idx: 18, src_idx: 23
suffle_idx: 19, src_idx: 0
suffle_idx: 20, src_idx: 17
suffle_idx: 21, src_idx: 26
suffle_idx: 22, src_idx: 36
suffle_idx: 23, src_idx: 29
suffle_idx: 24, src_idx: 18
suffle_idx: 25, src_idx: 10
suffle_idx: 26, src_idx: 6
suffle_idx: 27, src_idx: 19
suffle_idx: 28, src_idx: 38
suffle_idx: 29, src_idx: 24
suffle_idx: 30, src_idx: 15
suffle_idx: 31, src_idx: 16
suffle_idx: 32, src_idx: 40
suffle_idx: 33, src_idx: 49
suffle_idx: 34, src_idx: 7
suffle_idx: 35, src_idx: 39
suffle_idx: 36, src_idx: 20
suffle_idx: 37, src_idx: 32
suffle_idx: 38, src_idx: 48
suffle_idx: 39, src_idx: 14
suffle_idx: 40, src_idx: 35
suffle_idx: 41, src_idx: 21
suffle_idx: 42, src_idx: 4
```

```
suffle_idx: 43, src_idx: 28
suffle_idx: 44, src_idx: 1
suffle_idx: 45, src_idx: 30
suffle_idx: 46, src_idx: 51
suffle_idx: 47, src_idx: 47
suffle_idx: 48, src_idx: 11
suffle_idx: 49, src_idx: 27
suffle_idx: 50, src_idx: 50
suffle_idx: 51, src_idx: 45
```

复原图像

根据上一步获得的对应关系即可复原图像

```
paste_data = [
    37, 42, 3, 33, 34, 12, 43, 2, 5, 41, 8, 13, 46,
    9, 22, 31, 25, 44, 23, 0, 17, 26, 36, 29, 18, 10,
    6, 19, 38, 24, 15, 16, 40, 49, 7, 39, 20, 32, 48,
    14, 35, 21, 4, 28, 1, 30, 51, 47, 11, 27, 50, 45
]
# 即 paste_data[0] 为37，则suffle的第0个分块应该在src的第37个位置
```

获取分块

```
from PIL import Image
src_img = Image.open(src_img)
width, height = src_img.size
# 获取已打乱图像中的各个图块
block_list = []
block_width = 10
block_height = 80
for y1 in range(0, height, block_height):
    for x1 in range(0, width, block_width):
        block = src_img.crop((x1, y1, x1 + block_width, y1 + block_height))
        block_list.append(block)
```

之后重排分块

```
restore_block = [i for i in range(len(paste_data))]
for n in range(len(paste_data)):
    idx = paste_data[n]
    restore_block[idx] = block_list[n]
    block = block_list[idx]
```

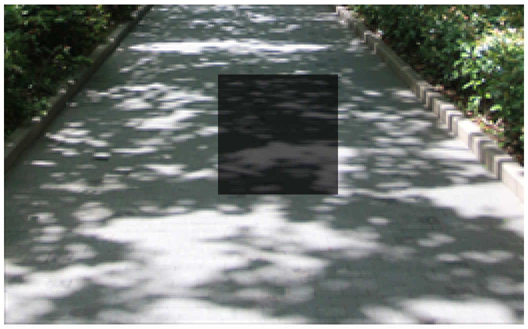
最后使用重排的分块复原图像

```
p_x = 0
p_y = 0
res_img = Image.new("RGB", (260, 160), (255, 255, 255))
for block in restore_block:
    res_img.paste(block, (p_x, p_y))
    p_x += block_width
    if p_x == 260:
        p_x = 0
        p_y += block_height
```

`suffle_slice_bg` 也使用相同思路进行复原，得到的复原图如下

原图

缺口图

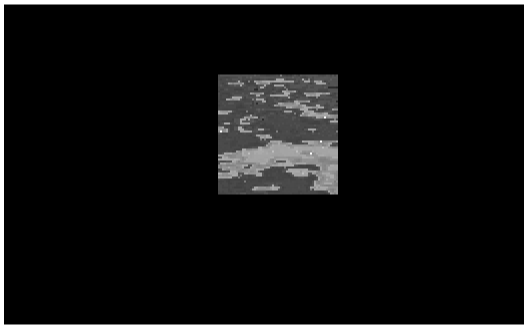


缺口比较

将复原的 `slice_bg` 与 `src_bg` 进行灰度处理后，单个像素只有0-255的灰度信息，对每个像素灰度进行XOR运算，即相同的像素最终为黑色(灰度为0)

```
def combine_images(img1_path, img2_path):
    # 打开图片1
    im1 = Image.open(img1_path)
    im1 = im1.convert('L') # 灰度处理
    pim1 = im1.load()
    # 打开图片2
    im2 = Image.open(img2_path)
    im2 = im2.convert('L') # 灰度处理
    pim2 = im2.load()
    width, height = im1.size
    column = []
    # 执行XOR运算
    for i in range(width):
        c = []
        for j in range(height):
            pim1[i, j] = pim1[i, j] ^ pim2[i, j]
            c.append(pim1[i, j])
        column.append(c)
    # 保存对比图
    im1.save("combine.png")
```

比对图



之后针对每一列与黑色列进行比对，不一样的列即为缺口位置

```
# 接着上步逐列像素点与黑色比对，找到不一样的纵列相对左边的像素偏移
null_column = [0] * height
for i in range(width):
    if column[i] != null_column:
        return i
```

获取和提交

使用requests进行图像获取和结果提交，判断code为0则退出

```
import base64
import requests
session = requests.Session()

def save_img(name, b64_data):
    header, b64_data = b64_data.split(",", 1)
    img_data = base64.b64decode(b64_data)
    with open(name, "wb") as f: f.write(img_data)

def captcha():
    url = f"{base_url}/captcha"
    r = session.get(url)
    data = r.json()
    save_img("src_bg.png", data["src_bg"])
    save_img("suffle_bg.png", data["suffle_bg"])
    save_img("slice.png", data["slice"])
    save_img("suffle_slice_bg.png", data["suffle_slice_bg"])

def validate(x_pos):
    url = f"{base_url}/validate"
    p = {
        "x_pos": x_pos
    }
    r = session.post(url, json=p)
    return r.json()
```

完整EXP

```python
import requests
import base64
from PIL import Image
import logging

logging.basicConfig(
    level = logging.INFO,
    format = '[%(asctime)s] - %(message)s',
    datefmt  = '%Y-%m-%d %H:%M:%S')
logging.getLogger('schedule').propagate = False
logging.getLogger('requests').propagate = False
logger = logging.getLogger(__name__)

base_url = "http://47.98.117.93:43919"

session = requests.Session()

def save_img(name, b64_data):
    header, b64_data = b64_data.split(",", 1)
    img_data = base64.b64decode(b64_data)
    with open(name, "wb") as f: f.write(img_data)

def captcha():
    url = f"{base_url}/captcha"
    r = session.get(url)
    data = r.json()
    save_img("src_bg.png", data["src_bg"])
    save_img("suffle_bg.png", data["suffle_bg"])
    save_img("slice.png", data["slice"])
    save_img("suffle_slice_bg.png", data["suffle_slice_bg"])

def validate(x_pos):
    url = f"{base_url}/validate"
    p = {
        "x_pos": x_pos
    }
    r = session.post(url, json=p)
    return r.json()

def restore_slices(src_img, save_name):
    # 打开打乱后的图像
    src_img = Image.open(src_img)
    width, height = src_img.size
    res_img = Image.new("RGB", (260, height), (255, 255, 255))
    p_x = 0
    p_y = 0

    # 按照打乱时的顺序进行复原
    paste_data = [
        37, 42, 3, 33, 34, 12, 43, 2, 5, 41, 8, 13, 46,
        9, 22, 31, 25, 44, 23, 0, 17, 26, 36, 29, 18, 10,
        6, 19, 38, 24, 15, 16, 40, 49, 7, 39, 20, 32, 48,
        14, 35, 21, 4, 28, 1, 30, 51, 47, 11, 27, 50, 45
    ]

    # 重新构建图像
    block_list = []
    block_width = 10
    block_height = 80
    restore_block = [i for i in range(len(paste_data))]
    # 分块获取已打乱图像中的各个图块
    for y1 in range(0, height, block_height):
        for x1 in range(0, width, block_width):
            block = src_img.crop((x1, y1, x1 + block_width, y1 + block_height))
            block_list.append(block)

    # 恢复源顺序数组
    for n in range(len(paste_data)):
        idx = paste_data[n]
        restore_block[idx] = block_list[n]
        block = block_list[idx]
    # 将图像归为
    for block in restore_block:
        res_img.paste(block, (p_x, p_y))
        p_x += block_width
        if p_x == 260:
            p_x = 0
            p_y += block_height
    # 保存复原后的图像
    res_img.save(save_name)

def combine_images(img1_path, img2_path):
    im1 = Image.open(img1_path) #打开图片
    im1 = im1.convert('L')
    pim1 = im1.load()
    im2 = Image.open(img2_path)
    im2 = im2.convert('L')
    pim2 = im2.load()
    width, height = im1.size
    column = []
    for i in range(width):
        c = []
        for j in range(height):
            pim1[i,j] = pim1[i,j] ^ pim2[i,j]
            c.append(pim1[i,j])
        column.append(c)

    im1.save("combine.png")
    null_column = [0] * height
    for i in range(width):
        if column[i] != null_column:
            return i

def get_slice_pos(slice_name):
    slice_img = Image.open(slice_name)
    pim1 = slice_img.load()
    width, height = slice_img.size
    null_column = [(0, 0, 0, 0)] * height
    for i in range(width):
        c = []
        for j in range(height):
            r1,g1,b1, a1 = pim1[i,j]
            c.append((r1,g1,b1, a1))
        if c != null_column: return i

get_flag = False
while not get_flag:
    logger.info("获取新验证码")
    captcha()
    restore_slices("suffle_slice_bg.png", "slice_bg.png")
    padding = combine_images("src_bg.png", "slice_bg.png")
    logger.info(f"缺口偏移: {padding}")
    result = validate(padding)
    print(result)
    if result["code"] == 0: get_flag = True
```