使用深度学习技术提取图像验证码应用研究

宋　伟　黎银环　陈俊豪　雷绍缨
（江门职业技术学院信息工程学院，广东 江门 529000）

摘要：“没有网络安全，就没有国家安全”，随着网络安全被提到了国家战略发展的高度，护网行动，粤盾行动以及国内各 CTF 大赛等一系列由国家主导的攻防对抗活动和人才选拔赛也逐步开展，黑白对抗的世界由台后走到台前，国内各高校成培养网络安全人才多半从 web 方面的 owasp top 10 漏洞展开，而暴力破解技术正是 owasptop 10 漏洞中的重要的一项，随着验证码技术的发展，传统的暴力破解已无法暴力破解 web 登录，本文通过使用深度学习成手段来提取验证码，进而实现验证码的暴力破解。

关键词：网络安全；深度学习；验证码爆破

一、概述

（一）研究背景

2014 年 2 月，中央网络安全和信息化领导小组第一次会议，习总书记做出了没有网络安全就没有国家安全的重要讲话，随后在一系列的讲话和指示中，将网络安全的发展提到了空前的高度，网络安全人才成成时下最紧缺的人才，随着等级保护 2.0 的出台，护网行动的开展，强网杯，护网杯，网鼎杯等各大 CTF 竞赛更是推动了网络安全技术的发展，根据国际组织 owasp 公布的知名漏洞，目前威胁互联网的 top 10 类型漏洞中其中一项即成爆破破解，暴力破解包含了 web 表单的登录破解，而验证码技术在 web 登陆上有着较成广泛的应用，各种 web 站点涉及到登录方面的基本都在应用，它很好的保护了用户的账号不被别有用心的黑客给暴力破解。

（二）研究现状

验证码是保护用户账户不被暴力破解，传统的手段固然难以攻破，但道高一尺魔高一丈，随着深度学习技术的发展，黑客成了完成验证码的自动识别，采用的暴力破解技术也在逐步结合深度学习技术发展，基于 Python 语言的深度学习技术也被不当的用于实现自动识别验证码和自动进行登录测试，验证码防御技术在发展，深度学习技术也在发展，这是一个对抗的竞赛活动，需要做持续不断的研究和更新。

（三）研究意义

不知攻，焉知防，攻防对抗技术是敌亦是友，目前全球各国的黑客们都在应用深度学习技术逐步地完善验证码自动识别技术，我国网络安全发展也需要任重道远，一旦别人所拥有的暴力破解技术能突破我们的防线，那我们就只能向前发展，才能做到更好的防御，很多国家如美国，日本，澳大利亚等都对网络安全做出了重要的战略部署，我们自然不能成人后！

二、研究思路

针对 web 登录技术中存在的手动输入验证码防御技术进行自动识别测试，从而达到后续的自动化的渗透测试。

（一）所需工具

开发环境：python 3.7 、 pycharm2020

模块：Pillow、sklearn、numpy 及其他子模块

（二）实施流程

描述整个识别流程：

①清理验证码并生成训练集样本

②提取验证码特征

（三）生成数据集

本次所用数据集采用了一个基于 java 语言的验证码自动化生成脚本。验证码组合是纯数字 + 大写字母 + 小写字母的组合，即 [0–9]+[A–Z]+[a–z] 这种组合。文件名是验证码的正确数字标签，具体实例如下

图一：训练样本

使用到三个数据集：

①训练集（training set）：10000 张验证码

②测试集（test set）：100 张验证码

③验证集（validation set）：100 张验证码

三、实施过程

（一）清理验证码并生成训练集样本

1. 读取图片

首先要做的是读取该文件路径下的所有图片文件，并打开和获取返回结果 image_array，每一个元素类型成“<class 'PIL.JpegImagePlugin.JpegImageFile'>”。

```
def read_captcha(path):
 image_array = []
    image_label = []
    file_list = os.listdir(path)# 获取captcha元素文件
    for file in file_list:
        image = Image.open(path + '/' + file) # 逐张读取图片
        file_name = file.split(".")[0]  #获取文件名，制作图片标签
        image_array.append(image)
        image_label.append(file_name)
    return image_array, image_label
```

图二：读取代码源程序

2. 清理粗图像

清理粗图像包括以下步骤：

步骤 1: 生成的原始图像是 RGB 格式图像，维度成（25，78，3）。将其转换成灰度图像，维度变成（25，78）。

原始图像：

图三：原始图形

灰度图像：

图四“转换后的图形”

步骤 2: 对于要识别的验证码，里面有很多干扰元素的灰线条。本次通过设定灰度阈值（默认 100），对图像中大于阈值的像素，赋值成 255。发现对于这种类型的验证码，该方法很实用。

图五：原始图像

```
def image_transfer(image_arry):
    image_clean = []
    for i, image in enumerate(image_arry):
        image = image.convert('L') # 转换成灰度图像，即RGB通道从3变成1
        im2 = Image.new("L", image.size, 255)
        for y in range(image.size[1]): # 遍历像素，将灰度超过阈值的像素转变成255
            for x in range(image.size[0]):
                pix = image.getpixel((x, y))
                if int(pix) > threshold_grey:  # 灰度阈值
                    im2.putpixel((x, y), 255)
                else:
                    im2.putpixel((x, y), pix)
        image_clean.append(im2)
    return image_clean
```

图六：转换图像源程序

3. 图像细清理

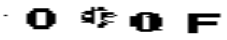
通过清理粗图像的办法还无法去除所有噪声点。需要引入细粒度的清理办法，

主要有三个步骤：

步骤 1：找出所有图像中的孤立点

步骤 2：计算黑色点近邻 9 宫格中黑色点个数，若小于等于 2 个，那么认成该点成噪声点；

步骤 3：清除噪声点。经过细清理后，虽然可以看到还存在一个噪声点，但效果其实很不错了。



图七：清除噪点后图像

4. 剥离单字符图像

去除孤立点后，我们还是没法一下子就识别出这四个字符，需要对经过处理后的图片进行分离。

分离方式主要有以下步骤：

步骤 1：找出图片中需要分离的开始和结束位置。对 width&height 进行遍历，每当出现一个黑色点，记为该字符起始位置；当新的一列出现全白色点，那么记为结束位置。[（8，9），（14，22），（29，38），（42，50），（57，66）]

步骤 2：经过清理后，还可能存在噪声点。在找到所有切割开始结束位置后，计算并选出（结束值 – 开始值）最大的切割位置。

[（14，22），（29，38），（42，50），（57，66）]

切割后视图如图：

图八：切割前图形

```
def image_split(image):
    """
    :param image:单张图像
    :return:单张图像被切割后的图像列表
    """
    inletter = False    #找出每个字母开始位置
    foundletter = False #找出每个字母结束位置
    start = 0
    end = 0
    letters = []    #储存坐标
    for x in range(image.size[0]):
        for y in range(image.size[1]):
            pix = image.getpixel((x, y))
            if pix != True:
                inletter = True
        if foundletter == False and inletter == True:
            foundletter = True
            start = x
        if foundletter == True and inletter == False:
            foundletter = False
            end = x
            letters.append((start, end))
        inletter = False
    # 切割出来的图像有可能是噪声点
```

图九：切割代码源程序

筛选可能切割出来的噪声点，只保留开始结束位置差值最大的位置信息

```
subtract_array = []    # 储存 结束-开始值
    for each in letters:
        subtract_array.append(each[1]-each[0])
    reSet = sorted(subtract_array, key=lambda x:x, reverse=True)[0:image_character_num
    letter_chioce = []    # 储存 最终选择点坐标
    for each in letters:
        if int(each[1] - each[0]) in reSet:
            letter_chioce.append(each)
    image_split_array = []    #储存切割后的图像
    for letter in letter_chioce:
        im_split = image.crop((letter[0], 0, letter[1], image.size[1]))
        im_split = im_split.resize((image_width, image_height)) # 转换格式
        image_split_array.append(im_split)
    return image_split_array[0:int(image_character_num)]
```

图十：去噪声点源程序

5. 保存到新的训练集

将按上述方法切分后的单个数字、字母，保存到新建的文件夹里，专门用来作成模型的训练集。

（二）特征提取

针对切割出后的每一个单字符进行特征提取，如 6。此处构建特征的方法较为简单，统计出每个字符图像每一行像素值成黑色的总和，加上每一列像素值成黑色的总和。因为我们切割后的图像大小成 8*25（width*height），故特征个数成 34=8+25。当然其实可以把单字符图像按像素值展开成一个 208=8*25 的向量，以此作为特征向量，也是可以的。示例结果如图所示：

feature vector：[7，11，13，4，4，13，11，7，0，0，0，0，0，4，6，4，6，6，6，6，6，6，6，4，6，4，0，0，0，0，0，0，0，0]

可能存在的问题，如果使用新类型 / 不同像素大小的验证码来做处理和特征提取，程序是否报错？此处需要在切割的步骤后面加上像素大小转换：

im_split = im_split.resize（（image_width，image_height））# 格式转换，im_split 切割后图像，image_width 目标像素宽度，image_height 像素高度

四、研究总结

深度学习的验证码提取，本次只是做了一个简单例子的演示。是针对某种特定类型的验证码，而时下的验证码有多种多样的存在，若换成其他类型的验证码做测试，还不能保证识别的准确率。

传统深度学习目前存在的不足：需要人工做数据清理和特征提取。目前已有思路可以解决这种烦琐的数据清理，以及人工提取验证码特征的缺点，目前此技术仍在发展中，还需要进一步实施研究，这是一项长期发展的过程，需要开展持续不断的研究。

参考文献：

[1] 温明莉，赵轩，蔡梦倩 . 基于深度学习的端到端验证码 识别 [J]. 无线互联科技，2017（14）：85-86.

[2] 王泽建 . 基于模板整体匹配的验证码识别算法研究与实现 [D]. 厦门大学，2012

[3] 汪洋，许映秋，彭艳兵 . 基于 KEN 技术的校内网验证码识别 [J]. 计算机与现代化，2017（2）：93-97.

[4] 刘欢，邵蔚元，郭跃飞 . 卷积神经网络在验证码识别上的应用与研究 [J]. 计算机工程与应用，2016，52（18）：1-7.

[5] 包乾，李文超，张庆东 Android 平台下的验证码识别研究 [J]. 科技通报，2017，33（9）：73-75，219.

[6] 杨雄 . 基于 python 语言和支持向量机的字符验证码识别 [J]. 数字技术与应用，2017（4）：72-74.

（支持项目：2022 年广东省科技创新战略专项资金（大学生科技创新培育）项目：pdjh2022b1045 基于深度学习的验证码破解与反破解技术研究与实现）