利用Tensorflow机器学习框架进行验证码识别

1. 项目简介

> 验证码（CAPTCHA）识别在网站中非常常见，是一种区分用户是计算机还是人的公共全自动程序，可以用于防止程序爬虫等功能。 验证码的存在对于我们利用程序访问某些网站造成了困难，由于验证码中人为添加的干扰元素，使得机器对于验证码的识别非常有难度。再加上不同字体的影响，验证码识别程序的通用性非常差，无法完成通用的验证码识别。
>
> 我们可以对于特定网站生成相似的验证码图像，利用机器学习Tensorflow框架实现OCR识别模型创建针对性的模型对于网站的验证码进行预测。 整个过程基本是全自动化的，人力成本低，有较强的可行性。

2. 系统环境

Python解释器、系统版本信息

```
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32
```

第三方库

库名称	版本	下载地址
Tensorflow	2.6.0	下载Tensorflow==2.6.0（通过清华大学镜像下载）
numpy	1.19.5	下载numpy==1.19.5（通过清华大学镜像下载）
Pillow	8.4.0	下载Pillow==8.4.0（通过清华大学镜像下载）
matplotlib	3.4.3	下载matplotlib==3.4.3（通过清华大学镜像下载）
requests	2.26.0	下载requests==2.26.0（通过清华大学镜像下载）

**注意安装时要保证 `keras` 版本为 `2.6.0` 否则可能导致程序无法运行，可安装 `Tensoflow` 后使用 `pip install keras==2.6.0` 命令。*

3. 项目文件总体框架

```
│  create_captcha.py-----------创建训练图片
│  create_model.py-------------创建模型
│  dir.txt---------------------目录框架
│  get_captcha.py--------------爬虫下载验证码（北理统一身份认证）
│  predict.py------------------预测验证码
│  README.md-------------------说明文件
│  train_model.py--------------训练模型
│
├─captcha_fonts----------------验证码字体文件集合
│      arial.ttf
│      arialbi.ttf
│      ariblk.ttf
```

```
│       ARLRDBD.TTF
│       bahnschrift.ttf
│       cambriab.ttf
│       consolaz.ttf
│       courbd.ttf
│       micross.ttf
│       msyh.ttc
│
├─model------------------------模型数据（不可用于预测，可载入模型继续训练）
│       checkpoint
│       model_weights.data-00000-of-00001
│       model_weights.index
│
└─model_to_predict-------------完整已训练的模型（可用于预测）
    │   keras_metadata.pb
    │   saved_model.pb
    │
    ├─assets
    └─variables
            variables.data-00000-of-00001
            variables.index
```

4. 项目流程、技术细节介绍

创建训练所需验证码图像

> 验证码的生成有以下几个方面的考虑：验证码字体样式、图片字体大小，水平和竖直排列、字体颜色、背景颜色、噪点（个数、颜色）、干扰线（条数、颜色）、图片扭曲、字母边界增强等。
>
> 利用Python图像库 `PIL` 以及随机库 `random` ，可以基本满足对于验证码图片的生成。图片大小设置为 `100*40` 像素，包含4个待识别待识别的大小写字母、数字；其中字体样式选择与目标识别验证码相似的多款字体；水平竖直排列在一定范围内随机；字体颜色、背景颜色采用随机的RGB数值；噪点数和干扰线数设置为10个、2条，颜色随机；对于最后生成的图片进行随机的扭曲以及滤镜处理。

将样本、标签进行预处理和归一化处理

> 对于深度学习网络而言，输入的数据需要转化成实数组成的向量。对于图片，统一大小后进行灰度处理，然后将每个像素点的RGB值映射到`[0,1]`区间的实数内；对于标签，将`0~9,a~z`作为字符集将验证码字母映射为数字。
>
> 说明：这里对于图片暂时不进行包括分割、去除噪音等繁琐的预处理过程，直接输入。考虑到一般网站的验证码不考虑大小写且较大的字符集对于模型的拟合有更大挑战，这里仅考虑小写字符和数字作为输出（输入包含大写字母）。

构建深度学习神经网络模型

> 这里利用 `TensorFlow` 框架以及相应api构建了一个简单的OCR识别模型，包括了CNN(输入层、Conv2D、数据最大池化技术)、RNN循环、优化器、CTC损失函数、输出层。这里对于模型的具体构建不进行讨论，仅对网络的基本参数例如 `batch_size` （批尺寸）、 `epoch` （训练次数）、`early_stopping_patience` (提前停止拟合的最大容忍次数)进行调整。

> 保存模型由于技术原因只能保存数据以及丢失了训练所需要的函数的模型（不能再次训练但是可以利用其进行预测）。

利用样本和标签对于模型进行拟合

> 使用 `model.fit` 函数对于预处理好的图像和标签进行拟合。

利用简单的爬虫下载相应网站的少量验证码，手动标签用于测试

> 使用 `requests` 库下载北京理工大学统一身份认证验证码图片（仅适用于针对性训练版本）。

5. 关键代码说明（具体实现参考源代码）

创建训练用的验证码图片

```
img = ValidCodeImg(width=random.randint(100, 100), height=random.randint(40, 40),#
设置验证码宽和高为100像素
                   code_count=4, font_size=24, # 验证码字符个数和字体大小
                   point_count=10, line_count=2, # 验证码干扰点和线数目
                   is_transform=random.choice([True]),# 是否添加扭曲效果
                   is_filter=random.choice([True]),# 是否添加滤镜效果
                   background_random=random.choice([True]),# 背景颜色是否随机
                   color_random=random.choice([True]),# 字体颜色是否随机
                   font_dir=random.choice(["ARLRDBD.TTF", "cambriab.ttf",
"courbd.ttf", # 验证码使用的字体
                                           "bahnschrift.ttf","arial.ttf",
"ariblk.ttf",
                                           "micross.ttf",
"arialbi.ttf","consolaz.ttf"]),
                   img_format='png', is_show=False) # 选择验证码图片格式以及是否展示
生成的图片
data, valid_str = img.getValidCodeImg() # 创建验证码图片以及对应字符串
```

输入数据预处理

```
def encode_single_sample(img_path, label): # 处理单张验证码图片
    img = tf.io.read_file(img_path) # 读取图像
    img = tf.io.decode_png(img, channels=1) # 解码并转换为灰度图片
    img = tf.image.convert_image_dtype(img, tf.float32) # 将图片数据转化为[0,1]区间
内的float32变量
    img = tf.image.resize(img, [img_height, img_width]) # 调整图片至预设大小
    img = tf.transpose(img, perm=[1, 0, 2]) # 转置图像使图像的宽对应于时间维度
    label = char_to_num(tf.strings.unicode_split(label, input_encoding="UTF-8")) #
将验证码对于字符映射为数字
    return {"image": img, "label": label}# 返回处理后的图片数据、标签数据元组
```

建立模型

```
def build_model():
    input_img = layers.Input( # 创建输入层
        shape=(img_width, img_height, 1), name="image", dtype="float32"
    )
    labels = layers.Input(name="label", shape=(None,), dtype="float32")

    x = layers.Conv2D(  # 二维卷积层1
        32,
        (3, 3),
        activation="relu",
        kernel_initializer="he_normal",
        padding="same",
        name="Conv1",
    )(input_img)
    x = layers.MaxPooling2D((2, 2), name="pool1")(x)

    x = layers.Conv2D( # 二维卷积层2
        64,
        (3, 3),
        activation="relu",
        kernel_initializer="he_normal",
        padding="same",
        name="Conv2",
    )(x)
    x = layers.MaxPooling2D((2, 2), name="pool2")(x)

    # 池化技术
    new_shape = ((img_width // 4), (img_height // 4) * 64)
    x = layers.Reshape(target_shape=new_shape, name="reshape")(x)
    x = layers.Dense(64, activation="relu", name="dense1")(x)
    x = layers.Dropout(0.2)(x)

    # 循环神经网络
    x = layers.Bidirectional(layers.LSTM(128, return_sequences=True,
dropout=0.25))(x)
    x = layers.Bidirectional(layers.LSTM(64, return_sequences=True, dropout=0.25))
(x)

    x = layers.Dense( # 输出层
        len(char_to_num.get_vocabulary()) + 1, activation="softmax", name="dense2"
    )(x)

    output = CTCLayer(name="ctc_loss")(labels, x) # 添加损失函数

    model = keras.models.Model( # 建立模型
        inputs=[input_img, labels], outputs=output, name="ocr_model_v1"
    )
    opt = keras.optimizers.Adam() # 创建优化器
    model.compile(optimizer=opt) # 编译模型并返回
    return model
```

利用模型进行预测

```
def decode_batch_predictions(pred):
    input_len = np.ones(pred.shape[0]) * pred.shape[1]
    results = keras.backend.ctc_decode(pred, input_length=input_len, greedy=True)
[0][0][ #利用贪心搜索获取最佳路径
        :, :max_length
    ]
    output_text = []
    for res in results: # 遍历输出结果获取预测文本
        res = tf.strings.reduce_join(num_to_char(res)).numpy().decode("utf-8")
        output_text.append(res)
    return output_text # 返回预测文本
```

利用"北理统一身份认证"验证码测试模型在未训练的数据集上的准确度

```
import requests

url = "http://login.bit.edu.cn/authserver/getCaptcha.htl?" # 验证码生成目标url
headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/94.0.4606.81 Safari/537.36'
}
for i in range(16):
    img = requests.get(url) # 获取验证码
    with open(str(i) + ".png", "wb") as f:
        f.write(img.content) # 将验证码写入文件保存
```

6. 运行效果展示

7. 完成度自我评价以及总结