**实验四 车牌检测**

1. 实验目的
2. 掌握基于卷积神经网络的目标检测基本原理。
3. 了解迁移学习思想，在数据样本较小的情况下学会利用预训练模型提升自身模型精度。
4. 进一步熟悉TensorFlow框架的使用，学会复用开源代码。
5. 实验要求
6. 基于Mask R-CNN的[开源实现](https://github.com/matterport/Mask_RCNN)，以及在MS COCO数据集上[预训练的模型](https://github.com/matterport/Mask_RCNN/releases/download/v2.0/mask_rcnn_coco.h5)（mask\_rcnn\_coco.h5），利用Python语言和TensorFlow框架（可结合Keras），结合提供的车牌目标检测数据集对Mask R-CNN模型（mask\_rcnn\_coco.h5）进行微调，实现对车牌的目标检测与提取。
7. 提取到的车牌部分，可采用传统数字图像处理方法对车牌字符进行分割（建议方法：可依据车牌部分二值化之后字符边界之间的像素关系；如有更高效准确的方法，更推荐使用），并利用实验二构造的卷积神经网络模型，实现对字符的识别。
8. 实验原理
9. Mask R-CNN的原理参考其[论文](https://arxiv.org/abs/1703.06870)。
10. 对Mask R-CNN的详细解析，模型各部分与开源代码一一对应关系，以及如何构建自己的数据集并利用预训练模型拓展到特定的任务，可参考附件博客文章“Splash of Color\_ Instance Segmentation with Mask R-CNN and TensorFlow”。
11. 实验所用工具及数据集

1、该实验思路参考附件博客文章“Splash of Color\_ Instance Segmentation with Mask R-CNN and TensorFlow”（该博客文章介绍了如何利用Mask R-CNN预训练模型来识别图片中的气球，代码位于Mask R-CNN的[开源实现](https://github.com/matterport/Mask_RCNN)中samples/balloon，本实验要实现从Mask R-CNN到车牌的识别，因此只需要替代气球识别数据集为车牌识别数据集并对该部分代码进行相应的修改，其余操作基本一致）。

2、采用的数据集来自GitHub相关开源项目提供的数据，训练集包含196张图片，验证集包含50张图片，使用了VGG Image Annotator（VIA）工具对图像中的车牌进行标注，标注结果保存在JSON文件中。数据集可在课程网站中进行下载，数据集的目录划分如下：

dataset

|— train

|— via\_project\_carplate\_train.json

|— via\_region\_data.json

|— \*.jpg

|— val

|— via\_project\_carplate\_val.json

|— via\_region\_data.json

|— \*.jpg

1. 实验步骤与方法

该实验复用了Mask R-CNN开源代码，因此用于车牌检测的网络模型结构与Mask R-CNN一致，只是利用了车牌的小型数据集对于MS COCO数据集上完成训练的模型进行了参数微调，以适应车牌检测任务。关于Mask R-CNN的网络结构，请自行查阅论文和相关资料。

1．数据读取

目标检测不同于简单的图像分类任务，需要对图像中车牌区域四个角点位置进行标注，同时该实验采用了Mask R-CNN，还需要获取车牌区域的掩码。

VIA将每个图片的车牌区域的角点信息保存为如下形式：

{"filename":"粤BB867A.jpg",

"size":230407,

"regions":[

{"shape\_attributes":{

"name":"polygon",

"all\_points\_x":[107,215,216,102],

"all\_points\_y":[324,322,354,354]

},

"region\_attributes":{}}

],

"file\_attributes":{}}

因此我们需要从json文件中获取车牌角点信息，并且将其转化为车牌掩码（掩码尺寸与图像尺寸一致，像素初始化为全0，依据车牌的四个角点把属于车牌区域的像素置为1），参考Mask R-CNN的[开源实现](https://github.com/matterport/Mask_RCNN)中samples/balloon/balloon.py，该部分代码定义了一个继承自utils.Dataset的类BalloonDataset，用于读取我们自己标注好的数据集，对于车牌检测任务，我们也需要重载这部分代码以读取车牌数据集（因为数据集格式一致，实际上也无需多做修改）。

class LPDDataset（utils.Dataset）：  
    def load\_carplate（self，dataset\_dir，subset）：  
        ...

def load\_mask（self，image\_id）：  
        ...

def image\_reference（self，image\_id）：  
        ...

车牌数据读取示例（代码可参考Mask R-CNN的[开源实现](https://github.com/matterport/Mask_RCNN)中samples/balloon/inspect\_balloon\_data.ipynb）如图1所示：

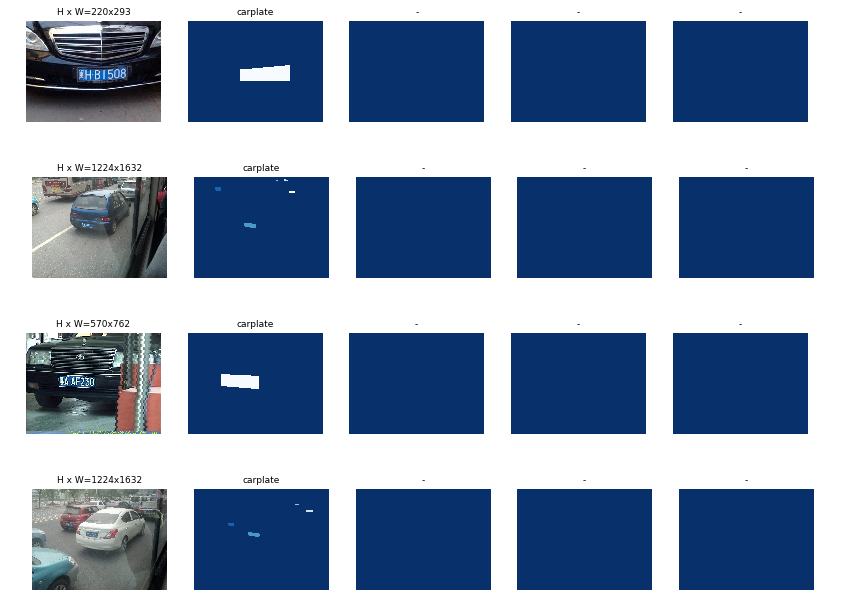


图1 数据读取

2．模型构建

复用Mask R-CNN源码之后，只需要引入源码中的model.py，使用其中的MaskRCNN()即可构建其模型结构，参考Mask R-CNN的[开源实现](https://github.com/matterport/Mask_RCNN)中samples/balloon/balloon.py中的main函数部分。

Mask R-CNN源码中config.py定义了模型训练过程中一些参数值以及可选操作，在Mask R-CNN的[开源实现](https://github.com/matterport/Mask_RCNN)samples/balloon/balloon.py中定义了一个继承自config.py的类，如下。在实现车牌检测时，也可定义相关的类修改与我们任务相关的一些参数。

class LPDConfig(Config):

# Give the configuration a recognizable name

NAME = "LPD"

# Adjust down if you use a smaller GPU.

IMAGES\_PER\_GPU = 2

# Number of classes (including background)

NUM\_CLASSES = 1 + 1 # Background + carplate

# Number of training steps per epoch

STEPS\_PER\_EPOCH = 100

# Skip detections with < 90% confidence

DETECTION\_MIN\_CONFIDENCE = 0.9

3．模型训练

不需要从头训练一个Mask R-CNN，只需要用车牌数据集对训练好的Mask R-CNN模型的部分层进行微调即可。因此需要先下载预训练模型，并将其参数导入到定义好的Mask R-CNN模型中，再导入模型参数时，不需要导入最后面的几层的参数，毕竟车牌识别任务只需要识别车牌，与MS COCO上的目标检测略有区别。

# Load weights

print("Loading weights ", weights\_path)

if args.weights.lower() == "coco":

# Exclude the last layers because they require a matching

# number of classes

model.load\_weights(weights\_path, by\_name=True,

exclude=["mrcnn\_class\_logits","mrcnn\_bbox\_fc",

"mrcnn\_bbox", "mrcnn\_mask"])

对模型的训练参考Mask R-CNN的[开源实现](https://github.com/matterport/Mask_RCNN)samples/balloon/balloon.py中的train()函数。

4．Mask R-CNN模型测试及后续操作

模型训练完成之后，还需要实现对车牌区域的字符分割。模型的测试参考Mask R-CNN的[开源实现](https://github.com/matterport/Mask_RCNN)samples/balloon/ inspect\_balloon\_model.ipynb。测试示例如图2、3所示，图2为测试图片的标注数据显示，图3为模型预测检测到的车牌区域：



图2 标注数据



图3 模型预测

到这一步，我们已经获取到图像中的车牌区域，接下来需要提取车牌区域的图像，并将其按照字符划分。

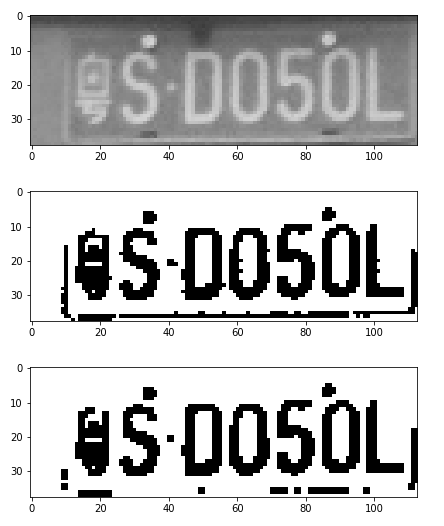


图4 车牌

可以将车牌区域照片进行二值化，然后依据字符之间的像素空白关系进行划分。也可以自行查找相关资料，使用更有效的方法。