基于图像的停车占用检测使用深度

学习-一个CPU友好的MATLAB教程

**摘要**

们使用卷积神经网络作为特征提取器，并使用一个二进制的SVM分类器来检测一个停车场的停车占用率。一个小数据集包含3000张停车位图像（包括占用和空）。随后，使用训练好的分类器对巴里街数据集进行停车占用率检测。我们还报告了通过使用不同的网络作为特征提取器所可实现的准确性，并报告了运行时。本

MATLAB教程将演示一个对CPU友好的管道，它可以用来损害其他硬件的不可用性，比如图形处理单元(GPU)。

关键词：实时停车占用率检测、闭路电视摄像头、深度学习、教程。

# 1介绍

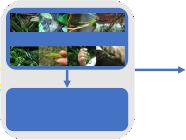
本教程是在本教程的范围内，我们从PKLot数据集中采样3000张图像（1500张被占用，1500张被空），从BarryStreet数据集采样100张图像（占用2800张）。与我们的论文相比，我们还比较了不同网络及其在MATLAB中的运行时间的准确性。完整的MATLABLive脚本作为附录作为附件。

# 2MATLAB和工具箱

这些教程本打算在MATLAB2020a版本上运行，尽管该代码可以在高于2018a版本的MATLAB版本中运行。可能需要额外的工具箱来运行实验，包括计算机视觉工具箱、统计学和机器学习工具箱、深度学习工具箱、信号处理工具箱和自动驾驶工具箱。为了顺利地运行实时脚本，请确保增加MATLAB的java堆内存，如在实时脚本开始时所示。

# 3教程：预先训练好的网络作为功能提取器-CPU友好

图1显示了本教程的管道，在那里我们将使用ResNet50(使用ImageNet预训练)提取3000个数据集的PKLot分割图像的图像特征。我们将使用这些图像特征训练一个二元SVM分类器（具有5倍交叉验证），并将评估训练的准确性。随后，我们用BarryStreet数据集测试训练过的模型，以评估其对看不见的数据集的泛化能力。



图像网

训练

使用ImageNet进

行预训练的CNN

被占用的

空

采样的PKLot



交叉验证

空的被占用

**二进制SVM 分类器**

图1：教程2的管道，其中我们使用一个预先训练过的CNN作为特征提取器来训练一个二进制SVM分类器。



图像网

**?**

测试

使用ImageNet进

行预训练的CNN

巴里街



分类

空

被占用的

### . 1 3使用预先训练过的ResNet50提取PKLot图像的特征，并训练一个SVM分类器

默认情况下，为了节省时间，我们将“加载训练矩阵”设置为true，它将训练矩阵和标签加载到工作区中，而不运行CNN。

其他3个

4负载（“培训矩阵”。席子

.

2

1加载训练矩阵=为真

“加载训练矩阵”被设置为false，将运行，在线执行特征提取。特性提取过程大约需要6分钟才能完成CPU， 使用GPU需要几秒钟。在第一步中，我们将预先训练好的CNN加载到工作区。为了使用预先训练好的网络作为特征提取器，我们提取了在分类层前的最后一层(Softmax)的全连接层的特征。因此，在本教程中，我们将使用被称为“fc1000”的全连接层的激活来提取特性。注意，这个特性将是一个1000维的向量。接下来，我们将以类似于在教程1中使用的方法设置分割的PKLot图像的路径。接下来，我们将初始化一个训练矩阵和地面真实标签，这将是输入到SVM分类器。

1%加载特征提取器，这是一个预先训练过的ResNet50CNN

2个负载(‘Resnet50特征提取器。席子

3

4%在分类前选择最后一层来提取特征

5层=‘fc1000’；

6

7%初始化训练矩阵

8培训 矩阵=零（长度（乘员数据）+长度（乘员数据），1001）；

9标签=单元格（长度（占用数据）+长度（占用数据），1）；

与教程1类似，我们将PKLot数据集的分割图像的大小调整为CNN的输入维数，然后提取图像特征（特定层的激活）并填充训练矩阵。我们执行两次步骤，一次是对空的分割图像，一次是对被占用的分割图像。

1%计算fc1000层的活化量

2个特征，占领=激活（网，占领我，层，“输出=”，“行”）；

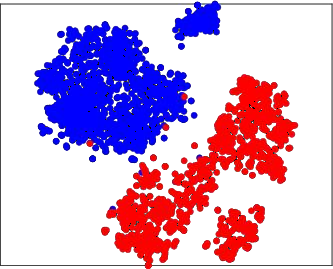
3

4%填充训练矩阵

5培训 矩阵（占领者图像指数，1：1000）=特征，占领者；

6培训 矩阵（居住图像指数，1001）=1；

在进行训练SVM之前，让我们用t-SNE算法将提取的特征可视化。从图2中我们可以看到这些类很好地聚类，很少有模糊的点，这表明提取的特征是类的良好表示。请注意，t-SNE图是用于数据分布的可视化，且距离是非线性的，因此，该图不包含任何轴。



占用的空

图2：预先训练过的CNN对提取的特征的可视化。这些类聚集得很好。

下一行代码训练一个立方SVM分类器，用预先训练好的CNN提取的特征进行5倍交叉验证。训练模型的交叉验证准确率为99：86%。

1%用提取的特征训练SVM分类器

2 [trainedClassifier, validationAccuracy, validationPredictions] = .. .

培训分类器（培训 矩阵

### 3.2Barry街数据集SVM分类的检验结果

在下一步中，我们将用训练过的SVM分类器来测试巴里街的图像。然而，在SVM分类器进行分类之前，需要使用预先训练好的CNN来提取Barry街图像的特征。默认情况下，为了减少运行时，我们提供包含预训练的CNN提取的特征的“测试特征”。这是通过将“在线测试”设置为假来实现的。

其他3个

4负载(“测试功能。席子

.

2

1测试在线=false；

“测试在线”被设置为true，将读取巴里街数据集的100张图像，并将裁剪每张图像的停车场。随后，使用预先训练好的CNN提取图像特征。这些

S

VM分类器将使用这些特征来对停车占用率进行分类。下面的代码使用训练过的SVM分类器来预测占用率。

1%SVM预测

2YPred（计数）=训练分类器。predictFcn (features(count,:));

请注意，SVM（具有2800个图像特征）在CPU上的训练时间大约是5秒，因此，与预先训练好的CNN相比，这个过程在CPU上是可行的。此外，SVM的测试时间非常快（2800个预测大约为0.3秒）。然而，对于100张BarryStreet图像的CPU，总的运行时间大约为7分钟（或每张图像为3.5秒），而使用GPU则只有几秒钟。

NN在{0；1}的范围内变化，SVM的分类得分是从该点到超平面的距离，可以在{-1；1}的范围内。最后，我们在图

##### 空

**输**

**出**

**等** 被占用的

**级**

**混淆矩阵**

|  |  |  |
| --- | --- | --- |
| **565** | **3** | 99.5% |
| 20.2% | 0.1% | 0.5% |
| **28** | **2204** | 98.7% |
| 1.0% | 78.7% | 1.3% |
| 95.3% | 99.9% | **98.9%** |
| 4.7% | 0.1% | **1.1%** |

##### 空 被占用的

**目标类别**

图3：分类的混淆矩阵，显示总体准确率为98.9%。这种错误分类很大程度上是由于将空停车位分类为被占用停车位（28个实例），而被占用停车位被分类为空停车位（3个实例）。

接下来，我们绘制混淆矩阵，并可视化错误的检测。图3显示了分类的混淆矩阵，我们可以看到总体准确率为98：9%。我们观察到，与（3）相比，空停车位（28）有更多的错误分类。图4显示了错误分类的图像补丁。我们看到，大多数错误的分类都是由于车辆的部分位置在停车位上的存在。我们还注意到，被错误分类的图像补丁中不包含任何车辆（比如最后一行的图像补丁）得分较低。请注意，与分数的情况不同

C

5中可视化了停车场的占用情况。

# 4使用不同的CNN架构的时间和准确性基准测试

在本节中，我们将比较不同的CNN架构的运行时时间和准确性，以帮助观众根据他们的需要选择正确的架构。在实验中，使用了i7 5600UCPU@2：6GHz，GPU为NVIDIATeslaP100,12GB内存。注意，我们只使用了CPU的一个核心。我们将ResNet50与AlexNet、谷歌网、移动网v2、挤压网和VGG-16进行了比较。

表1显示了CNN和SVM分类器所获得的精度，它们使用CPU的运行时和训练时，以及用于提取图像特征的层。我们观察到，除了VGG-16网络外，

**占用分数：0.46878**



**占用分数：1.2121**



**空分：0.11414**



**占用分数：0.73194**



**占用分数：1.3675**



**占用分数：1.3059**



**占用分数：0.57902**



**占用分数：0.053715**

**空分：0.053512**



**占用分数：1.4772**



**占用分数：0.53507**



**占用分数：0.75308**



**占用分数：0.72798**



**占用分数：0.66833**



**空分：0.13842**



**占用分数：0.046703**

**占用分数：0.45892**



**占用分数：0.86678**



**占用分数：0.5678**



**占用分数：0.85676**



**占用分数：0.89892**



**占用分数：0.83242**



**占用分数：0.13753**



**占用分数：0.1359**

**占用分数：1.5619**



**占用分数：0.72747**



**占用分数：0.59065**



**占用分数：0.82518**



**占用分数：0.51164**



**占用分数：0.12687**



**占用分数：0.04301**



##### 

图4：一些错误分类的图像补丁及其各自的分类分数。

所有网络的占用率预测都是准确的。与单独运行预训练的网络相比，我们看到运行时间有所增加。这种计算开销很大程度上是由于用于计算预先训练过的CNN的“激活”的缓慢函数。在训练时间方面，我们能够提取特征并使用CPU训练支持向量机，除了VGG-16花了大约半个小时。

# 致谢

该研究由墨尔本工程学院的研究参与基金和墨尔本研究奖学金支持。这项研究是利用在墨尔本大学主办的LIEFHPC

-GPGPU设施进行的。该设施是在LIEF拨款LE170100200的协助下建立的。



占用的插槽：24

空插槽：4

图5：停车场占用率的可视化，其中洋红色表示被占用的空间，青色表示空置的空间。

表1：使用CNN作为特征提取器和SVM进行分类的不同CNN架构的可实现精度和运行时的比较。ine调整数据为PKLot 分割图像，测试数据为Barry街数据。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 网络 | 精度 | CPU运行时(msec) | CPU上的训练时间（分钟 | 特征层 |
|  |  |  | ） |  |
| 亚历克斯网 | 98:9% | 110.3 | 3:18 | 'fc8' |
| 谷歌网 | 98:9% | 119.0 | 5:04 | 'loss3- classiier' |
| MobileNet v2 | 97:3% | 118.7 | 4:18 | “日志” |
| ResNet50 | 98:9% | 150.3 | 6:12 | 'fc1000' |
| 方框网 | 98:1% | 81 8 | 2:00 | 'pool10' |
| VGG-16 | 93:4% | 581.1 | 26:52 | 'fc8' |
| **参考文献** |  |  |  |  |  |

阿查里亚，D.，严，W.，和科谢勒姆，K。基于深度学习的实时图像的停车占用检测。发表在第五届研究年会会议记录@Locate，第2087卷，第33-40页，2018年4月。

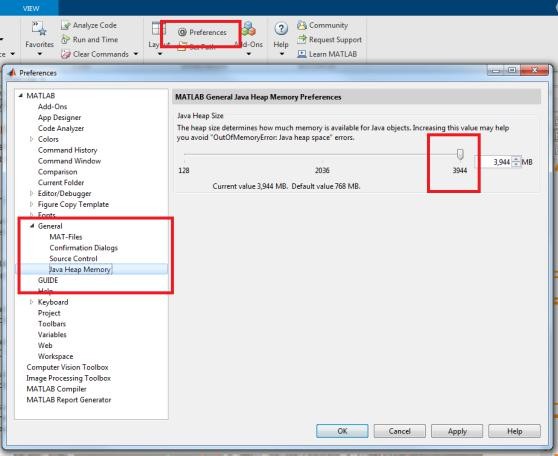
夏蒂尔德，西蒙扬，吠达尔迪， A., 和齐泽曼， A.在细节上的魔鬼的回归：深入研究卷积网。arXiv预印本arXiv：1405。3531, 2014.

附录

设置数据和可视化

#### 在开始本教程之前，请确保您将java堆内存增加到最大的可用空间，否则MATLAB将会多次崩溃。有关设置情况

，请见下图。它可以在Home按钮->偏好下找到。



清除，关闭全部；%清除工作区和命令窗口

如果存在~(‘MATLABCodeCNNSVM。zip，“文件”）

disp(‘下载文件(218MB)。..');

URL = 'https://melbourne .figshare .com/ndownloader/files/24726374'; websave('MATLABCodeCNNSVM.zip ',URL);

其他的

disp(“Zip文件存在”)

最后部分

Zip文件存在

unzip(“MATLABCodeCNNSVM。zip') disp（“解压缩文件”）

解压缩文件

addpath ( 'FinalCodeSVM/') addpath('FinalCodeSVM/SupportingFunctions/')

**可视化PKLot数据集**

#### PKLot数据集包含12000张+闭路电视图像和695,000张+分段停车位图像。占用率是手动生成的。在本教程的范围内，我们采样了3000个停车位图像。

%可视化PKLot数据集的分割图像

%（我们采样了1500个被占用空间和1500个空空间） PKLotEmpty = imresize (imread(. . .

* F i n a l C o d e S V M \ P K L o t S e g m e n t e d S a m p l e d \ E m p t y \ 2 0 1 2 - 0 9 - 1 1 \_ 1 5 \_ 4 5 \_ 5 7 # 0 0 4 . jp g ' ) , [ 7 0 5 0 ] ) ; PKLotOccupied = imresize (imread( .. .
* F i n a l C o d e S V M \ P K L o t S e g m e n t e d S a m p l e d \ O c c u p i e d \ 2 0 1 2 - 0 9 - 1 1 \_ 1 5 \_ 3 6 \_ 3 2 # 1 0 0 . j p g ' ) , [ 7 0 5 0 ] ) ;

图

subplot (1,2,1) imshow (PKLotEmpty);

标题（“1500个空插槽”）

subplot (1,2,2)

imshow (PKLotOccupied);

标题(“1500个被占用的插槽”。')



**可视化巴里街数据**

可视化巴里街（100张图片，28个停车场）。这是我们的测试数据，我们希望自动识别停车位，并随后执行占用率检测。地面真相和停车位注释是为了评估的目的。



%加载一个图像。

BarryStreetImage = imread('FinalCodeSVM\BarryStreetData\DSC\_0455.JPG');

%装载停车位标记和占用率load('GroundTruthBarryStreet.席子

%Visuise巴里街

图

=插入形状(条纹图像，“矩形”，停车槽，. . .

“线宽”，2）；

imshow (BarryStreetImageAnn);

这就是数据集可视化的结论。

教程-从预先训练过的网络中提取特性，以训练二进制SVM(CPU 友好)

**训练可以用CPU完成，总训练时间不到7分钟**

#### 下面的代码片段将加载一个预先训练过的Resnet50CNN，并从PKLot数据集中提取图像特征，这些数据集随后将用于训练一个具有5倍交叉验证的三次SVM分类器。训练后的模型将用巴里街数据集进行测试，以进行评估

。

清除，clc；关闭所有；

**使用预先训练好的Resnet50提取PKLot图像的特征，并训练一个SVM分类器**

使用CNN提取这些特征。在CPU上运行可能需要6分钟或在GPU上运行几秒钟。为了方便起见，我们提供了默认加载的训练矩阵。下面的一行应该更改为false进行在线处理。

加载培训矩阵=true；

如果(加载培训矩阵==false)

##### %加载一个预先训练过的ResNet50CNN的特征提取器 负荷(‘Resnet50功能提取器。席子

%在分类前选择最后一层以提取特征层=‘fc1000’；

%获取CNN的输入图像大小，以调整裁剪图像的大小 输入大小=net。层（1）。输入大小；

%设置分割的空和占用停车位的路径

occupydata = dir (fullfile(pwd,. . . 'FinalCodeSVM\PKLotSegmentedSampled\Occupied\', '\* .jpg'));

occupydata = {occupydata .name}'; emptydata = dir (fullfile(pwd,. . .

'FinalCodeSVM\PKLotSegmentedSampled\Empty\', '\* .jpg'));

##### emptydata = {emptydata.name}';

%初始化将作为SVM的输入的训练矩阵

包含地面分类器标签

training\_matrix=零（长度（乘员数据）+长度（乘员数据），1001）；

##### 标签=单元格（长度（占用数据）+长度（占用数据），1)；

%提取了1500个被占用的停车场的特征

tic

对于乘员图像指数=1：长度（乘员数据） OccupyIm = imread (fullfile (. . .

##### pwd, 'FinalCodeSVM\PKLotSegmentedSampled\Occupied\',.. .

occupydata{occupyImageIndex}));

OccupyIm = imresize (OccupyIm, inputSize(1:2)); 特征占领=激活（净，占领，层，“输出=”，“行”）； training\_matrix（居住者图像指数，1：1000）=特征，占领； training\_matrix（居住者成像指数，1001）=1；

标签：=“被占用”； 最后部分

##### toc

%3x斗Je)斗 }oe斗uJos o} tS66 omd斗( deJ入Sn8 sto斗s

}oJ omd斗(Ime8oInpox =t:ton8斗| )omd斗(pe斗e(

3md斗(Im = SmJoep )}utt}Sto) dwp' '」SnetDopoSVM\d〉1o斗So8mon斗opSemdtop\3md斗(\'' omd斗(pe斗e}omd斗(Ime8oInpox杨((!...

3md斗(Im = SmJosSzo)3md斗(Im' Sndu斗SSzo)t:C((!

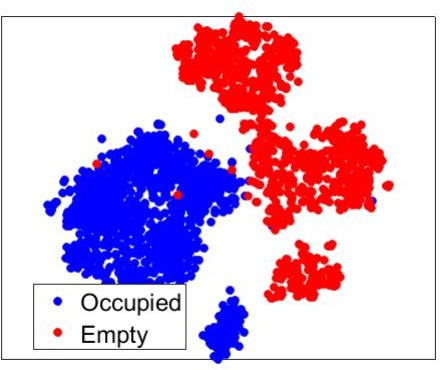
}oe斗uJos3md斗( = e)斗Sve斗Sons)no斗'3md斗(Im'te (oJ' 'Ou斗du斗As'' 'Jows' (!

斗JeSnSn8\_me斗JSx)o))ud (Ime8oInpox+omd斗(Ime8oInpox't:t666( = }oe斗uJos3md斗(!斗JeSnSn8\_me 斗 JSx)o))ud (Ime8oInpox+omd 斗 (Ime8oInpox't66t(=6!teqots}o))ud (Ime8oInpox+omd斗(Ime8oInpox杨= '3md斗(' !

onp 奥特索

toep)'斗JeSnSn8Me斗JSx.me斗'(! pSsd)'TJeSnSn8 me斗JSx toepop'(

onp

TJeSnSn8 me斗JSx toepop

% vSsuetSso 斗|o }oe斗uJos ox斗Je)斗op q ( 斗|o DNN wS斗| 斗 -SN3 et8oJS斗|m Y = 斗sno)斗JeSnSn8\_me斗JSx):'t:t666((!

}S8uJo

|= 8s)e斗斗oJ)Y):'t('Y):'C(' teqots' ''' 'o'(!

so斗)|)t(' 'DotoJ'' 'q'' 'MeJ入oJ」e)oDotoJ'' 'q'(!

so斗)|)C(' 'DotoJ'' 'J'' 'MeJ入oJ」e)oDotoJ'' 'J' (! so斗)8)e' '」on斗SSzo''C6(

exSs obuet!

so斗)8)e' 'x斗S)入' '][' '(斗S)入'' ][(



%训练三次SVM分类器与5倍交叉验证，应该采取

%6秒来训练分类器

tic

%使用5倍交叉验证

[trainedClassifier, validationAccuracy, validationPredictions] =.. .

列车分类器(training\_matrix)；

toc

运行时间为3.944439秒。

disp(“对SVM分类器的训练已经完成”)

对SVM分类器的训练已经完成

%显示训练后的分类器对训练数据的交叉验证精度

fprintf（“分类器的验证精度%f%%”。\n”，验证精度\*100）；

该分类器的验证准确率为99.800000%。

## 用巴里街数据集测试该分类器的分类精度

#### 在本节中，我们拍摄巴里街数据集的图像，裁剪出每个图像的所有停车位，并使用CNN提取特征。这可能在CPU上需要8分钟或在GPU上需要几秒钟。为了方便起见，我们提供了默认加载的测试矩阵。要在线运行，请将下一个代码行更改为true。

测试在线=false；

load('GroundTruthBarryStreet.加载地面真实值以评估准确性

%设置到巴里街数据集的路径

imageName = dir (fullfile(pwd, 'FinalCodeSVM\BarryStreetData\', '\* .JPG'));

imageName = {imageName.name}';

%加载一个预先训练过的ResNet50CNN的特征提取器负荷(‘Resnet50功能提取器。席子

%设置特征提取层层=‘fc1000’；

%获取CNN的输入图像大小，以调整裁剪图像的大小输入大小=net。层（1）。输入大小；

如果(测试在线==true)

tic

计数=1；

对于n=1：长度（图像名）

BarryStreetImage = imread (fullfile (. . .

pwd, 'FinalCodeSVM\BarryStreetData\', imageName{n})); 对于m=1：长度（占用（：，1））

%裁剪出个人停车场

cropImage{count} = imcrop (BarryStreetImage, ParkingSlots(m,:));

##### %处理地面真相到单元格

如果占用（m，n）==1

=“已占用”； 其他的

注释表{count}=‘Empty’；

最后部分

##### %调整作物的大小到CNN的输入大小

imdsIm=不调整大小（裁剪图像{计数}，输入大小（1：2））；

%计算激活

特性（计数，：）=激活(网络，imdsIm，图层，“OutputAs”，“行”)；

%SVM预测

[YPred（计数），procs（计数，：）]=。..

trainedClassifier.predictFcn (features(count,:));

%处理对单元格的预测

如果YPred（计数）==1

=“被占用”； 其他的

=“空”；

##### 最后部分

count=count+1;

最后部分

最后部分toc

其他的

加载(“测试功能。席子

显示器（“测试特性加载”）； tic

%使用SVM分类器进行预测

[YPred, probs] = trainedClassifier .预测tFcn（特征）；%SVM预测

##### YPred=YPred‘；%转置矩阵，服务于输入到绘图混淆函数

计数=1；

对于n=1：长度（图像名）

##### 对于m=1：长度（占用（：，1））

%处理对单元格的预测

如果YPred（计数）==1

=“被占用”； 其他的

=“空”；

##### 最后部分

count=count+1;

最后部分

最后部分toc

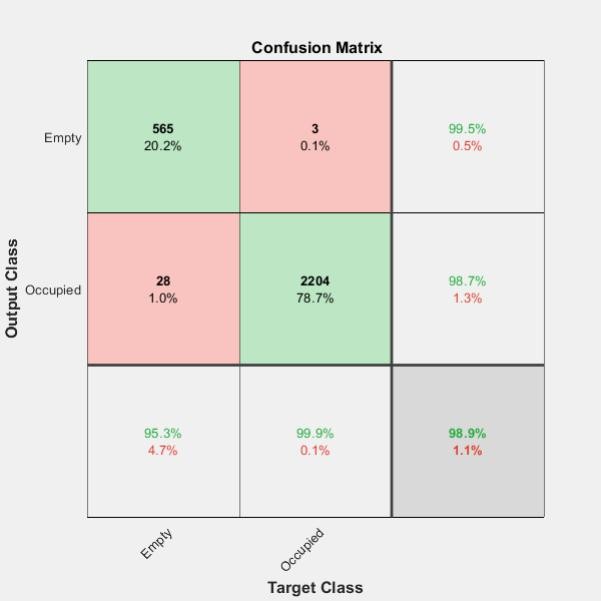
##### 最后部分

测试功能加载

运行时间为0.250680 秒。

测试精度=平均值（占用率（：）==YPred‘）；

打印错误（“分类器的测试精度%f%%”。\n”，测试精度\*100）；

该分类器的测试准确率为98.892857%。

图

情节混淆（分类（注释表）、分类（分类‘））

**可视化一些错误分类的插槽**

错误插槽=发现（分类（错误类）~=注释表）；

图

设置（gcf、“单位”、“标准化”、“外向”、[0、0.04、1、0.96]）；

对于n=1：长度(WlongSlot)

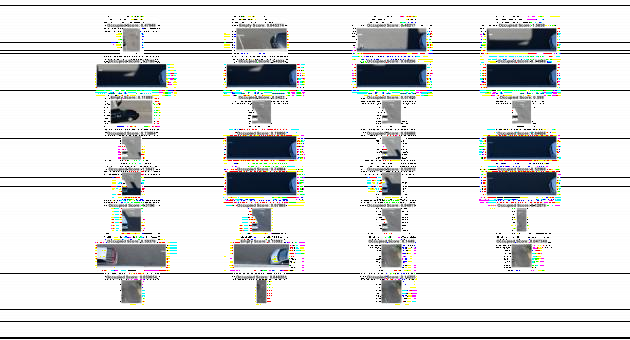
subplot (8,4,n);

imshow (cropImage{WrongSlots (n)});

str=[字（字类（错误槽(n)）...

分数：num2str（最大值（错误(n)，：）））；

标题(str)； 最后部分



## 可视化入住率

图

AnnotatedImage = imread("FinalCodeSVM\BarryStreetData\DSC\_0040.JPG");

emptySlots = 0;

occupiedSlots =0;

对于n= 1 ： 28

如果YpredClass(n)==“空”

注释图像=插入形状(注释图像，“文件矩形”，存储槽(n，：)，“线宽度”，1，“ 颜色”，“青色”，“不透明度”，0.4)；空槽=空槽+1；...

其他的

AnnotatedImage = insertShape (AnnotatedImage, 'FilledRectangle',... “停车槽”（n、：）、“线宽”、1、“颜色”、“品红”、“不透明度”、0.4）； occupiedSlots = occupiedSlots + 1;

最后部分

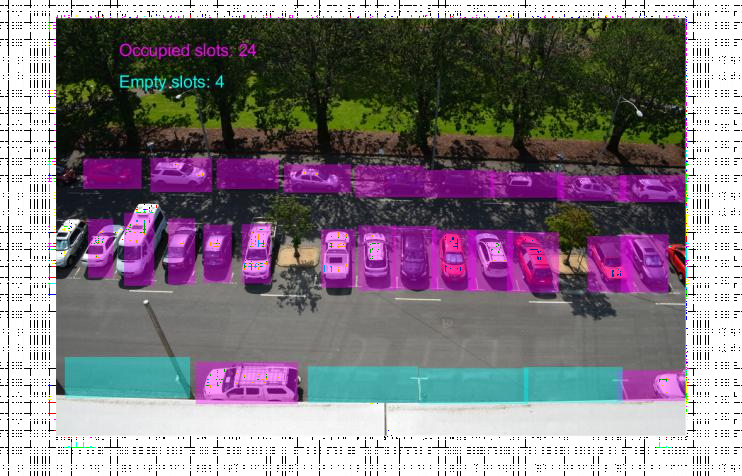
最后部分

imshow (AnnotatedImage)

文本（100、50、[“占用槽：“num2str（占用槽）]、“颜色”、“品红”、“FontSize”

文本（100、100，[“空槽：”num2str（空槽）]，“颜色”、“青色”、“FontSize”，20）；

,20);



#### 本教程结束。