Real-time image-based parking occupancy detection using deep learning - A CPU Friendly MATLAB tutorial

Debaditya Acharya* and Kourosh Khoshelham
acharyad@student.unimelb.edu.au and k.khoshelham@unimelb.edu.au
Department of Infrastructure Engineering, The University of Melbourne, Parkville, Victoria, Australia, 3010

Abstract

In this tutorial we use convolutional neural networks as feature extractor and use a binary SVM classifier for detecting the parking occupancy of a parking area. A small dataset is generated from PKLot dataset containing 3000 images of parking spaces (both occupied and empty). Subsequently, the trained classifier is used perform parking occupancy detection of Barry Street dataset. We also report the accuracy achievable by using different networks as feature extractor, and report the run-times. This MATLAB tutorial will demonstrate a CPU friendly pipeline that can be used to compromise the unavailability of additional hardware, like graphical processing units (GPU). The code is made public at https://github.com/debaditya-unimelb/real-time-car-parking-occupancy and we also provide the sampled PKLot dataset and Barry Street dataset along with the pre-trained models at https://melbourne.figshare.com/ndownloader/files/24726374

Keywords: Real-time parking occupancy detection; CCTV cameras; Deep learning; Tutorial.

1 Introduction

This tutorial is a supplementary material related to our work Acharya et al. (2018). Note that accuracies reported in the paper was based on the whole PKLot dataset and using ImageNet-VGG-f (Chatfield et al., 2014) network. For the scope of this tutorial we sampled 3000 images (1500 occupied and 1500 empty) from PKLot dataset and 100 images from Barry Street dataset (2800 occupancy). In contrast to our paper, we also compare the accuracies of different networks and their run-times in MATLAB. The complete MATLAB Live Script is attached as an annexure.

2 MATLAB and toolboxes

The tutorials are intended to run on MATLAB 2020a, although the code can run in MATLAB versions higher than 2018a. Additional toolboxes might be required to run the experiments that include computer vision toolbox, statistics and machine learning toolbox, deep learning toolbox, signal processing toolbox and automated driving toolbox. For running the live script smoothly, please ensure to increase the java heap memory of MATLAB, as demonstrated at the start of the live script.

3 Tutorial: Pre-trained network as feature extractor - CPU friendly

Figure 1 shows the pipeline of the tutorial where we will extract image features of 3000 segmented images of PKLot dataset using ResNet50 (pre-trained with ImageNet). We will train a binary SVM classifier (with 5-fold cross validation) using these image features and will evaluate the training accuracy. Subsequently, we test the trained model with Barry Street dataset to assess its generalisation ability to unseen dataset.

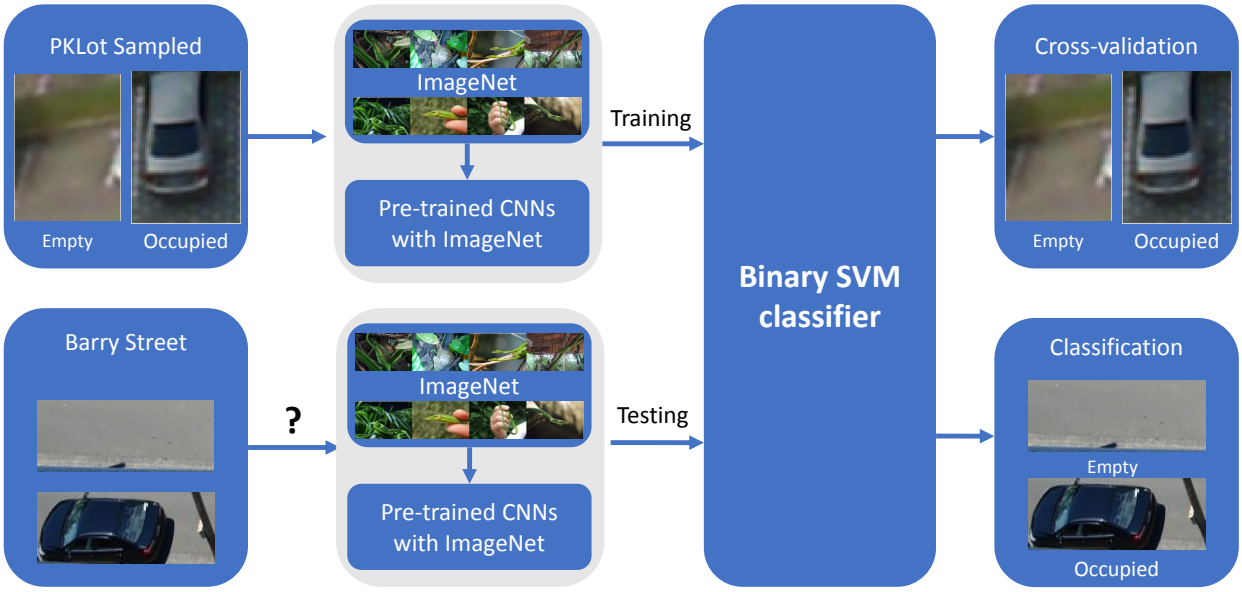


Figure 1: The pipeline of Tutorial 2, where we use a pre-trained CNN as a feature extractor to train a binary SVM classifier.

3.1 Extract features of PKLot images using pre-trained ResNet50 and train an SVM classifier

By default to save time, "LoadTrainingMatrix" us set to true, which loads the training matrix and the labels into the workspace without running the CNN.

```
LoadTrainingMatrix = true
.
else
load('trainingMatrix.mat');
```

"LoadTrainingMatrix" being set to false will run perform the feature extraction on-line. The feature extraction process takes around 6 minutes to complete with CPU or a couple of seconds with GPU. In the first step we load the pre-trained CNN to the workspace. To use a pre-trained network as a feature extractor, we extract the features of the fully-connected layer that is the last layer before the classification layer (Softmax). Therefore, in this tutorial, we will be using the activations of the fully-connected layer called "fc1000" to extract the features. Note, this feature will be a 1000-dimensional vector. Next, we will set the paths to the segmented PKLot images in a similar way we did in Tutorial 1. Next, we will initialise a training matrix and the ground truth labels that will be the input to the SVM classifier.

```
% load the feature extractor that is a pre-trained ResNet50 CNN
load('Resnet50FeatureExtractor.mat');

% select the last layer before classification to extract the features
layer = 'fc1000';

% initialise the training matrix
training_matrix= zeros(length(occupydata)+ length(emptydata), 1001);
labels = cell(length(occupydata)+ length(emptydata),1);
```

Similar to Tutorial 1, we resize the segmented images of the PKLot dataset to the input dimensions of the CNN, and we subsequently extract the image features (activations of the particular layer) and populate the training matrix. We perform the steps twice, once for empty segmented images and once for the occupied segmented images.

```
% Calculate the activations of fc1000 layer
featuresOccupy = activations(net,OccupyIm,layer,'OutputAs','rows');

% Populate the training matrix
training_matrix(occupyImageIndex,1:1000) = featuresOccupy;
training_matrix(occupyImageIndex,1001)=1;
```

Before proceeding to the training the SVM, let us visualise the extracted features with t-SNE algorithm. From Figure 2 we can see the classes are well clustered with a very few ambiguous points, and this shows that the extracted features are good representations of the classes. Note that t-SNE plot is meant for visualisation of data distribution, and the distances are non linear, and therefore, the plot does not contain any axes.

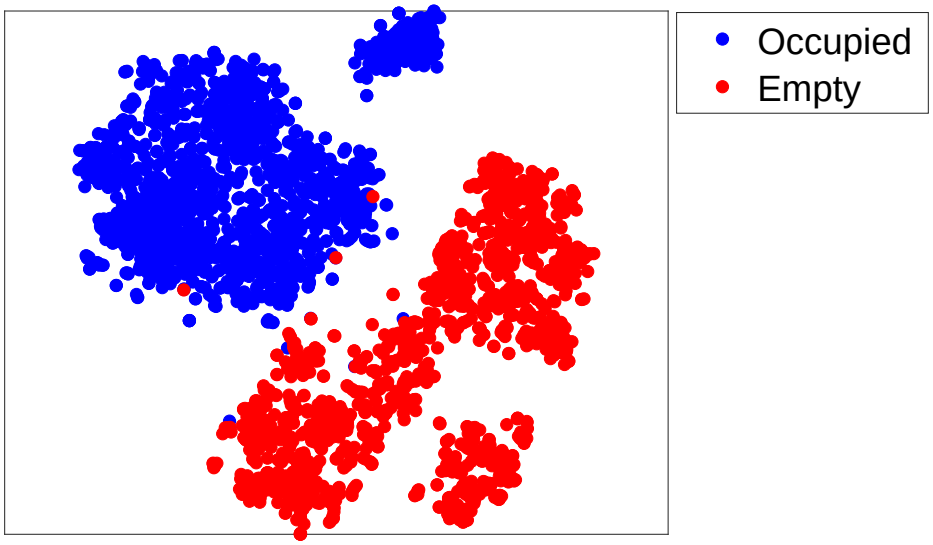


Figure 2: The visualisation of the extracted features by the pre-trained CNN. The classes are well clustered.

The next line of code trains a cubic SVM classifier with 5-fold cross-validation with the features extracted by the pre-trained CNN in a couple of seconds. The cross validation accuracy of the trained model is 99.86%.

```
% train the SVM classifier with extracted features
[trainedClassifier, validationAccuracy, validationPredictions] = ...
    trainClassifier(training_matrix);
```

3.2 Test classification results of SVM classifier with Barry street dataset

In the next step, we will be testing the Barry Street images with the trained SVM classifier. However, before the SVM classifier can make classification, the features of the Barry Street images are needed to be extracted using the pre-trained CNN. By default to reduce run-time, we provide the "TestFeatures" containing the features extracted by the pre-trained CNN. This is achieved by setting "TestOnline" to false.

```
TestOnline = false;
.
else
load('TestFeatures.mat');
```

"TestOnline" being set to true, will read the 100 images of Barry Street dataset and will crop parking slots of each image. Subsequently, the image features will be extracted using the pre-trained CNN. These

features will be used by the SVM classifier to classify the parking occupancy. The following code predicts the occupancy using the trained SVM classifier.

```
% SVM predictions
YPred(count) = trainedClassifier.predictFcn(features(count,:));
```

Note that training time for SVM (with 2800 image features) is approximately 5 seconds on CPU, and therefore, as compared to fine-tuning a pre-trained CNN this process is feasible with CPU. Also, the test time with SVM is very fast (approximately 0.3 seconds for 2800 predictions). However, the overall run-time is approximately 7 minutes with a CPU for 100 Barry Street images (or 3.5 seconds for each image) or a couple of seconds with a GPU.

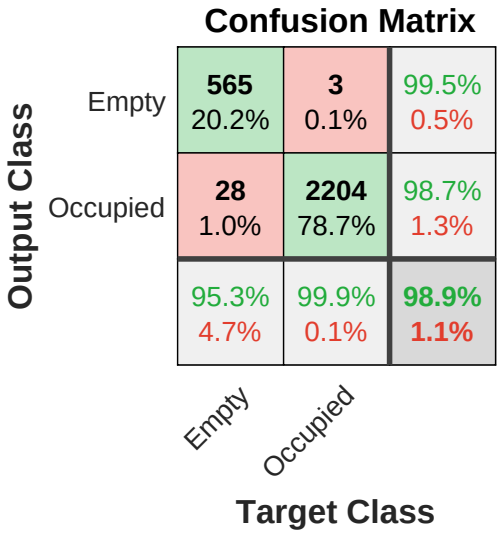


Figure 3: The confusion matrix of the classification, showing that the overall accuracy is 98.9%. The misclassifications are largely due to wrong classification of empty parking spaces as occupied ones (28 instances) and a very less instances of occupied ones classified as empty (3 instances).

Next we plot the confusion matrix and visualise the wrong detections. Figure 3 shows the confusion matrix of the classification, and we can see the overall accuracy is 98.9%. We observe that there are more wrong classifications of empty parking slots (28) as compared to occupied ones (3). Figure 4 shows the wrongly classified image patches. We see most of the wrong classifications are due to the presence of a vehicle partially in the parking slot. We also notice lower scores for the image patches that are wrongly classified and does not contain any vehicles in them (such as ones in the last row). Note that unlike the case of the scores of the CNN that vary in the range of $\{0, 1\}$, the classification scores of SVM is the distance from the point to the hyperplane and can be in the range of $\{-\infty, \infty\}$. Finally, we visualise the occupancy of the parking slots in Figure 5.

4 Time and accuracy benchmarking using different CNN architectures

In this section we compare different CNN architectures in terms of their runtime and accuracies achievable to help the audience to in choosing the right architecture for their needs. For the experiments, an i7 5600U CPU @2.6 GHz was used and the GPU was NVIDIA Tesla P100 with 12 GB of memory. Note we used only one core of the CPU for the experiments. We compared ResNet50 with AlexNet, GoogleNet, MobileNet v2, SqueezeNet and VGG-16.

Table 1 shows the accuracy achieved by CNN and SVM classifier, their run-time and train-time with CPU and the layers used to extract the image features. We observe that with the exception of VGG-16 network,

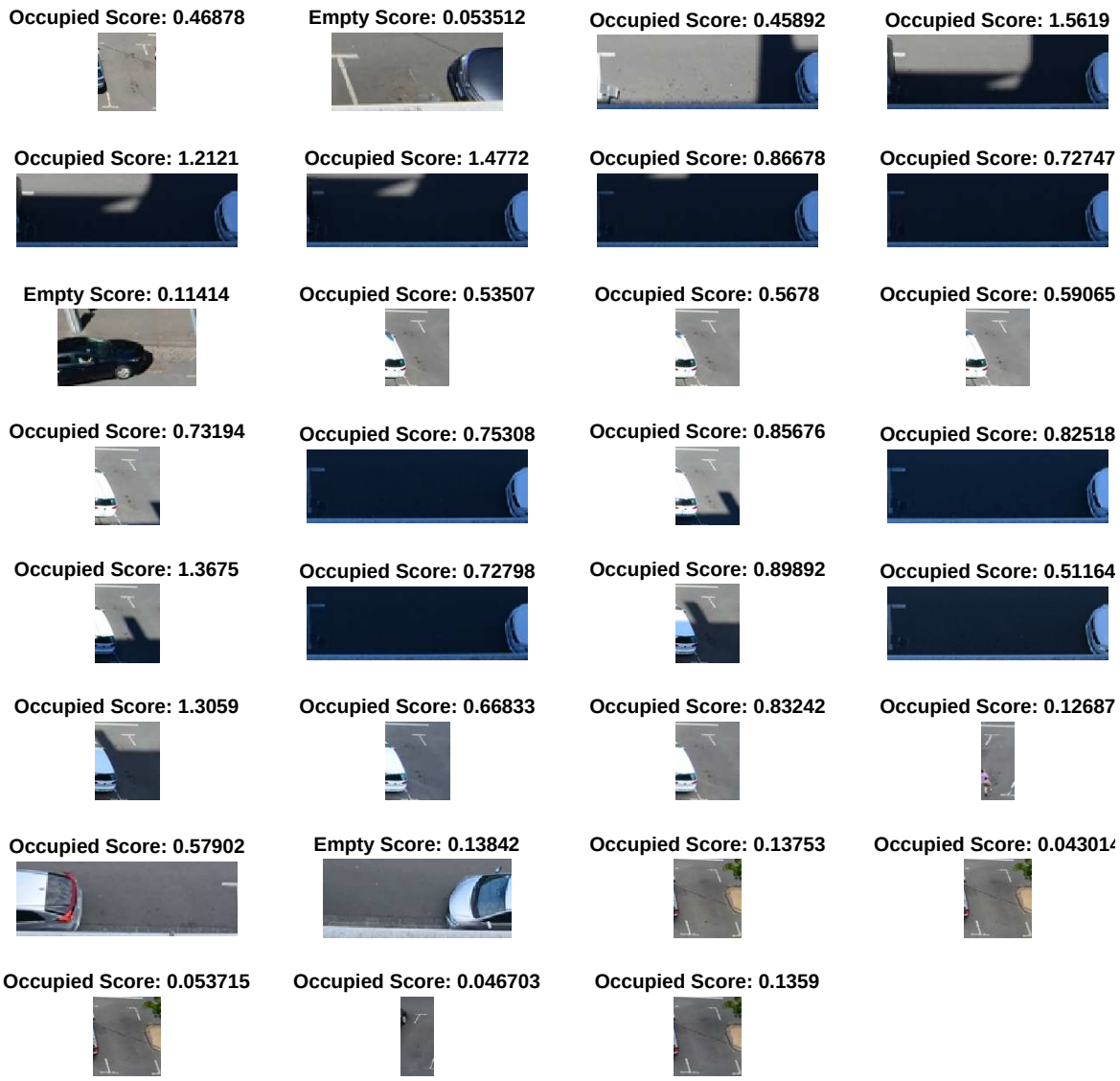


Figure 4: Some of the wrongly classified image patches with their respective classification scores.

all the networks are accurate in occupancy prediction. We see an increase in the run-times as compared to running the pre-trained networks alone. This computational overhead is largely due to the slow function that is used for calculating the "activations" of the pre-trained CNN. In terms of training-time, we were able to extract features and train the SVMs with CPU under 10 minutes, with the exception of VGG-16 that took around half an hour.

Acknowledgements

This research is supported by a Research Engagement Grant from the Melbourne School of Engineering and a Melbourne Research Scholarship. This research was undertaken using the LIEF HPC-GPGPU Facility hosted at the University of Melbourne. This Facility was established with the assistance of LIEF Grant LE170100200.

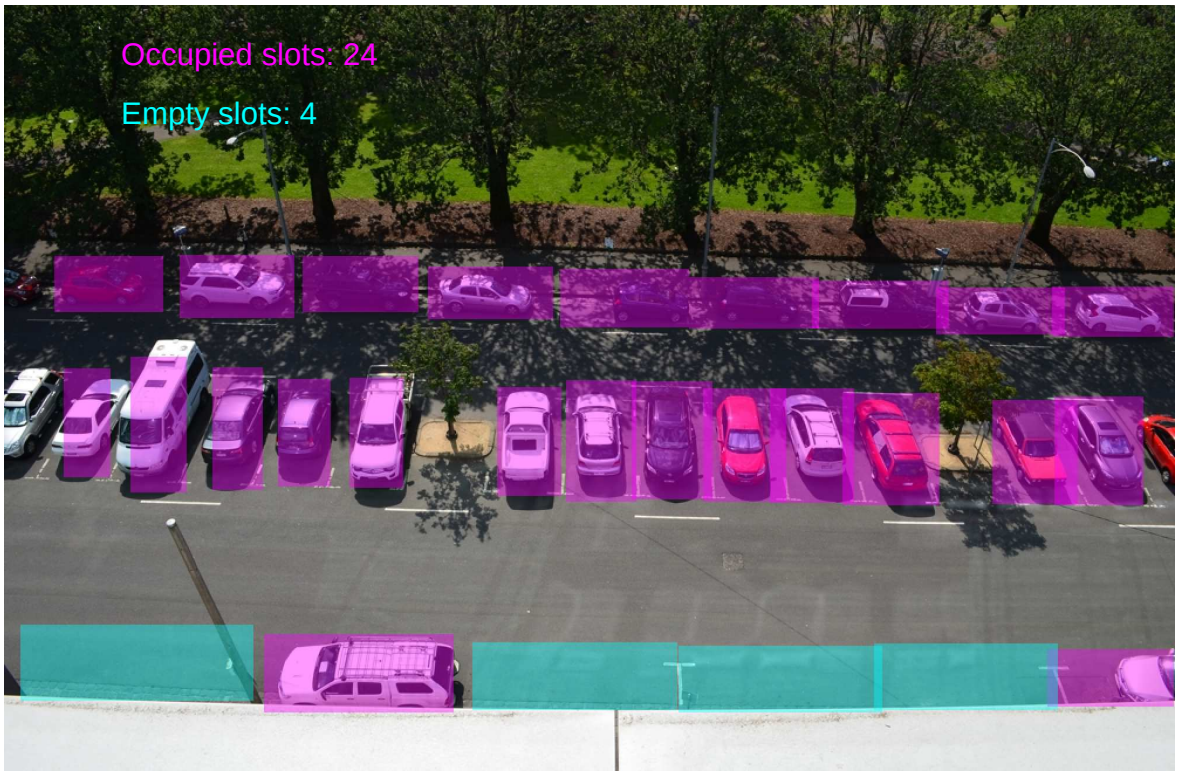


Figure 5: The visualisation of the parking occupancy, where magenta colour denotes the occupied and cyan colour denotes vacant spaces.

Table 1: The comparison of the achievable accuracy and the run-times for different CNN architectures by using CNN as the feature extractor and SVM for classification. The fine-tuning data was PKLot segmented images and test data was Barry street data.

Network	Accuracy	Run-time on CPU (msec)	Train time on CPU (min)	Feature layer
AlexNet	98.9%	110.3	3:18	'fc8'
GoogleNet	98.9%	119.0	5:04	'loss3-classifier'
MobileNet v2	97.3%	118.7	4:18	'Logits'
ResNet50	98.9%	150.3	6:12	'fc1000'
SqueezeNet	98.1%	81.8	2:00	'pool10'
VGG-16	93.4%	581.1	26:52	'fc8'

References

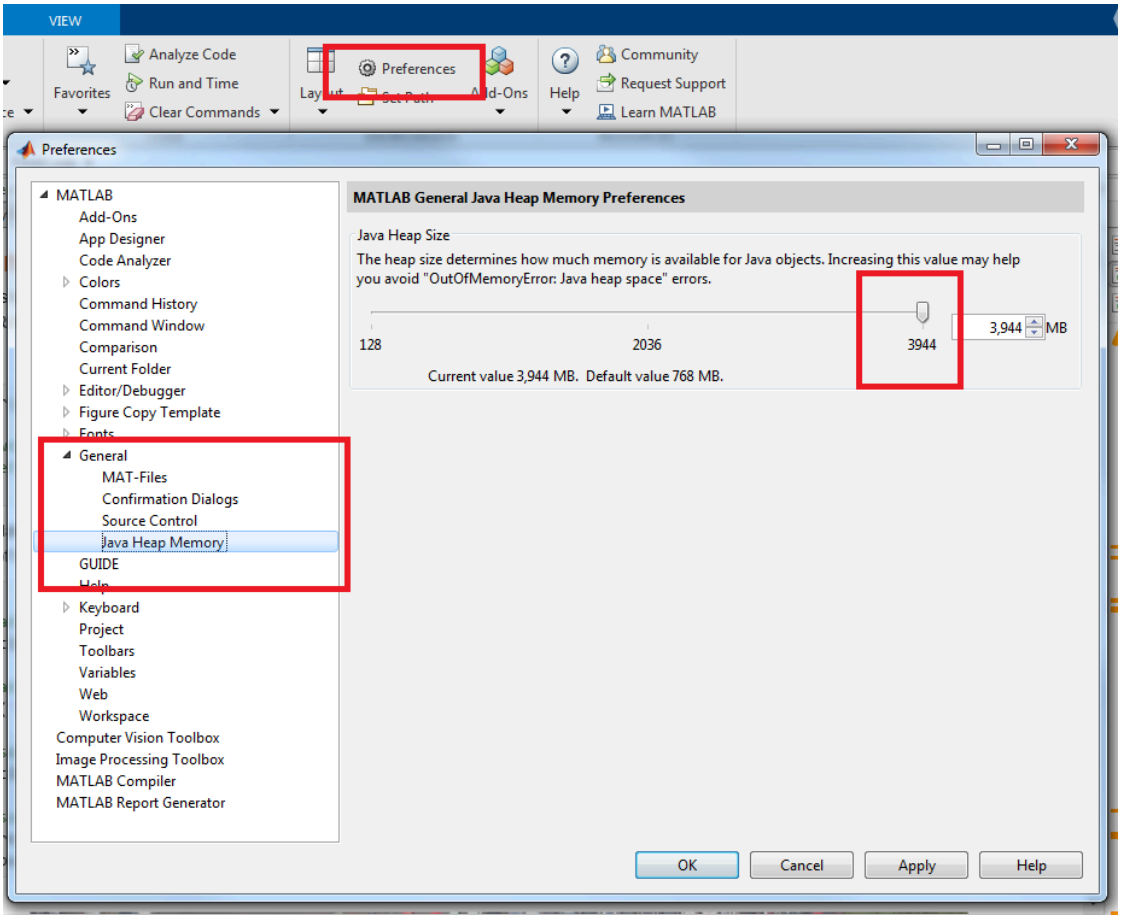
Acharya, D., Yan, W., and Khoshelham, K. Real-time image-based parking occupancy detection using deep learning. In *Proceedings of the 5th Annual Conference of Research@Locate*, volume 2087, pages 33–40, April 2018.

Chatfield, K., Simonyan, K., Vedaldi, A., and Zisserman, A. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.

Annexure

Setting up data and visualisation

Before starting the tutorial make sure you increase the java heap memory to the maximum avaialble space, otherwise MATLAB will crash many times. Refer to the below image for setting up. It can be found under Home button -> Preferences.




```
clear;clc; close all;% clear workspace and command window

if ~exist('MATLABCodeCNNSVM.zip','file')
    disp('Downloading file (218 MB)...');
    URL = 'https://melbourne.figshare.com/ndownloader/files/24726374';
    websave('MATLABCodeCNNSVM.zip',URL);
else
    disp('Zip file exists')
end
```

```
Zip file exists
```

```
unzip('MATLABCodeCNNSVM.zip')
disp('Unzipped files')
```

```
Unzipped files
```

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

Visualise PKLot dataset

PKLot dataset contains 12000+ CCTV images and 695,000+ segmented parking slot images.

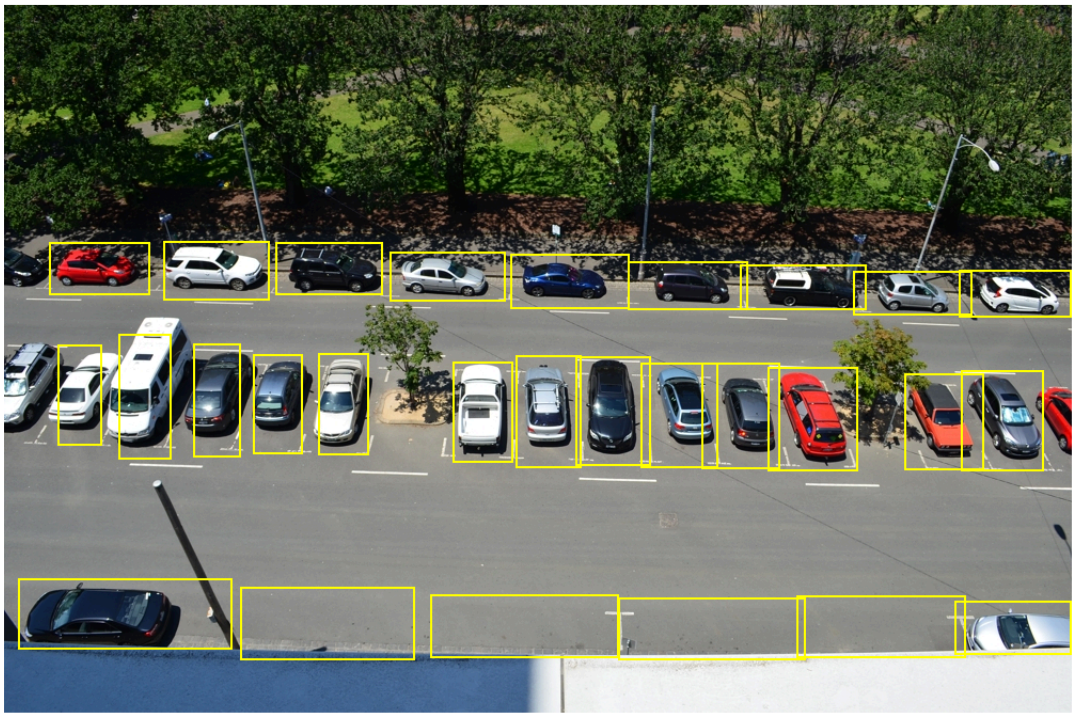
The occupancy was manually generated. For the scope of the tutorial, we sampled 3000 parking slot images.

```
% Visualise the segmented images of PKLot dataset
% (We sampled 1500 occupied and 1500 Empty spaces)
PKLotEmpty = imresize(imread(...
    'FinalCodeSVM\PKLotSegmentedSampled\Empty\2012-09-11_15_45_57#004.jpg'), [70 50]);
PKLotOccupied = imresize(imread(...
    'FinalCodeSVM\PKLotSegmentedSampled\Occupied\2012-09-11_15_36_32#100.jpg'), [70 50]);
figure
subplot(1,2,1)
imshow(PKLotEmpty);
title ('1500 Empty slots')
subplot(1,2,2)
imshow(PKLotOccupied);
title ('1500 Occupied slots.')
```

Visualise Barry street data

Visualise Barry street (100 images with 28 parking lots). This is our test data, and we want to identify the parking slots automatically, and subsequently perform occupancy detection. Ground truth and parking slot annotations are for evaluation purpose.

```
% Load one image.
BarryStreetImage = imread('FinalCodeSVM\BarryStreetData\DSC_0455.JPG');

% load the parking slot markings and occupancy
load('GroundTruthBarryStreet.mat');

% Visualise Barry Street
figure
BarryStreetImageAnn = insertShape(BarryStreetImage,'rectangle',ParkingSlots,...
    'LineWidth', 2);
imshow(BarryStreetImageAnn);
```



This concludes the dataset visualisation.

Tutorial - Extract features from a pre-trained network to train a binary SVM (CPU Friendly)

Training can be done with CPU and total training time is under 7 minutes

The following snippet will load a pre-trained Resnet50 CNN and extract image features from PKLot dataset which will be subsequently be used to train a cubic SVM classifier with 5-fold cross validation. The trained model will be tested with Barry street dataset for evaluation.

```
clear;clc;close all;
```

Extract features of PKLot images using pre-trained Resnet50 and train an SVM classifier

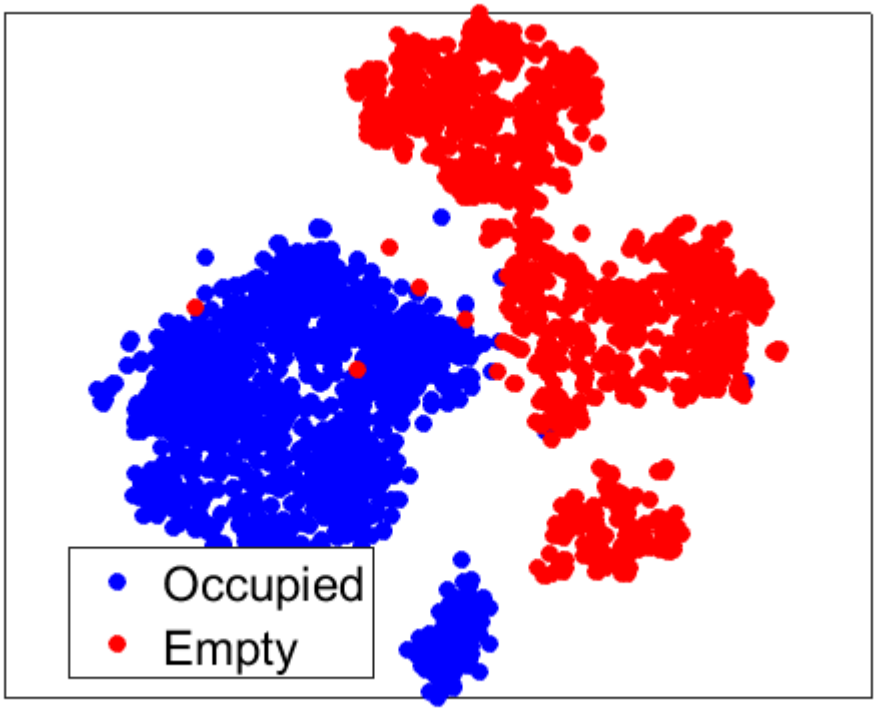
Extract the features using the CNN. This might take 6 minutes to run on CPU or a couple of seconds on GPU. For convinience we provide the training matrix that is loaded by default. The following line should be changed to false do the process online.

```
LoadTrainingMatrix = true;

if (LoadTrainingMatrix == false)

    % load the feature extractor that is a pre-trained ResNet50 CNN
    load('Resnet50FeatureExtractor.mat');

    % select the last layer before classification to extract the features
    layer = 'fc1000';

    % fetch the input image size of the CNN for resizing the cropped images
    inputSize = net.Layers(1).InputSize;

    % setting up the paths of the segmented empty and occupied parking spots
    occupydata = dir(fullfile(pwd,...
        'FinalCodeSVM\PKLotSegmentedSampled\Occupied\', '*.jpg'));
    occupydata = {occupydata.name}';
    emptydata = dir(fullfile(pwd,...
        'FinalCodeSVM\PKLotSegmentedSampled\Empty\', '*.jpg'));
    emptydata = {emptydata.name}';

    % initialise the training matrix that will serve as an input to the SVM
    % classifier containing the ground truth labels
    training_matrix= zeros(length(occupydata)+ length(emptydata), 1001);
    labels = cell(length(occupydata)+ length(emptydata),1);

    % extract features of 1500 occupied parking slots
    tic
    for occupyImageIndex =1:length (occupydata)
        OccupyIm = imread (fullfile ( ...
            pwd,'FinalCodeSVM\PKLotSegmentedSampled\Occupied\',...
            occupydata{occupyImageIndex}));
        OccupyIm = imresize(OccupyIm, inputSize(1:2));
        featuresOccupy = activations(net,OccupyIm,layer,'OutputAs','rows');
        training_matrix(occupyImageIndex,1:1000) = featuresOccupy;
        training_matrix(occupyImageIndex,1001)=1;
        labels{occupyImageIndex} = 'Occupied';
    end
    toc
```

```
    %Extract features of 1500 empty parking slots
    for emptyImageIndex =1:length (emptydata)
        EmptyIm = imread (fullfile ( pwd,'FinalCodeSVM\PKLotSegmentedSampled\Empty\',...
            emptydata{emptyImageIndex}));
        EmptyIm = imresize(EmptyIm, inputSize(1:2));
        featuresEmpty = activations(net,EmptyIm,layer,'OutputAs','rows');
        training_matrix(occupyImageIndex+emptyImageIndex,1:1000) = featuresEmpty;
        training_matrix(occupyImageIndex+emptyImageIndex,1001)=0;
        labels{occupyImageIndex+emptyImageIndex}='Empty';
    end
else
    load('trainingMatrix.mat');
    disp('Training matrix loaded')
end
```

```
Training matrix loaded
```

```
% visualise the features extracted by the CNN with t-SNE algorithm
Y = tsne(training_matrix(:,1:1000));
figure
h= gscatter(Y(:,1),Y(:,2), labels, '', 'o');
set(h(1), 'Color', 'b', 'MarkerFaceColor', 'b');
set(h(2), 'Color', 'r', 'MarkerFaceColor', 'r');
set(gca,'FontSize',20)
axis equal;
set(gca,'xtick',[], 'ytick', [])
```



```
% train the cubic SVM classifier with 5-fold cross validation, should take
% 6 seconds to train the classifier

tic
% using 5 fold cross validation
[trainedClassifier, validationAccuracy, validationPredictions] =...
    trainClassifier(training_matrix);
toc
```

```
Elapsed time is 3.944439 seconds.
```

```
disp('Training finished for the SVM classifier')
```

```
Training finished for the SVM classifier
```

```
% display the cross-validation accuracy of the trained classifier for the training data
fprintf('The validation accuracy of the classifier %f %%.\n',validationAccuracy*100);
```

```
The validation accuracy of the classifier 99.800000 %.
```

Test the classification accuracy of the classifier with Barry street dataset

In this section we take images of Barry street dataset, crop out all the parking slots of each image and extracts features using the CNN. This might take 8 minutes on CPU or a couple of seconds on GPU. For convinience we provide the testing matrix that is loaded by default. To run online please change the next code line to true.

```
TestOnline = false;

load('GroundTruthBarryStreet.mat') % load the ground truth to evaluate accuracy

% setup the path to Barry street dataset
imageName = dir(fullfile(pwd,'FinalCodeSVM\BarryStreetData\', '*.JPG'));
imageName = {imageName.name}';

% load the feature extractor that is a pre-trained ResNet50 CNN
load('Resnet50FeatureExtractor.mat');

% set the feature extraction layer
layer = 'fc1000';

% fetch the input image size of the CNN for resizing the cropped images
inputSize = net.Layers(1).InputSize;

if (TestOnline == true)
    tic
    count =1;
    for n=1:length(imageName)
        BarryStreetImage = imread (fullfile (...
            pwd, 'FinalCodeSVM\BarryStreetData\', imageName{n}));
        for m=1:length(Occupancy(:,1))
            % crop out individual parking slots
            cropImage{count} = imcrop(BarryStreetImage, ParkingSlots(m,:));
```

```
            % Process ground truth to cell
            if Occupancy (m,n)==1
                AnnotationTable{count} = 'Occupied';
            else
                AnnotationTable{count} = 'Empty';
            end
            % resize the crops to input size of the CNN
            imdsIm = imresize(cropImage{count}, inputSize(1:2));

             % calculate activations
            features(count,:) = activations(net,imdsIm,layer,'OutputAs','rows');

            % SVM predictions
            [YPred(count), probs(count,:)] =...
                trainedClassifier.predictFcn(features(count,:));

            % Process predictions to cells
            if YPred(count) == 1
                YpredClass {count} = 'Occupied' ;
            else
                YpredClass {count} = 'Empty' ;
            end
            count=count+1;
        end
    end
    toc
else
    load('TestFeatures.mat');
    disp('Test Features loaded');
    tic
    % perform prediction with SVM classifier
    [YPred, probs] = trainedClassifier.predictFcn(features); % SVM predictions
    YPred = YPred'; % transpose matrix to serve input to plotconfusion function

    count =1;
    for n=1:length(imageName)
        for m=1:length(Occupancy(:,1))

            % Process predictions to cells
            if YPred(count) == 1
                YpredClass {count} = 'Occupied' ;
            else
                YpredClass {count} = 'Empty' ;
            end
            count=count+1;
        end
    end
    toc
end
```

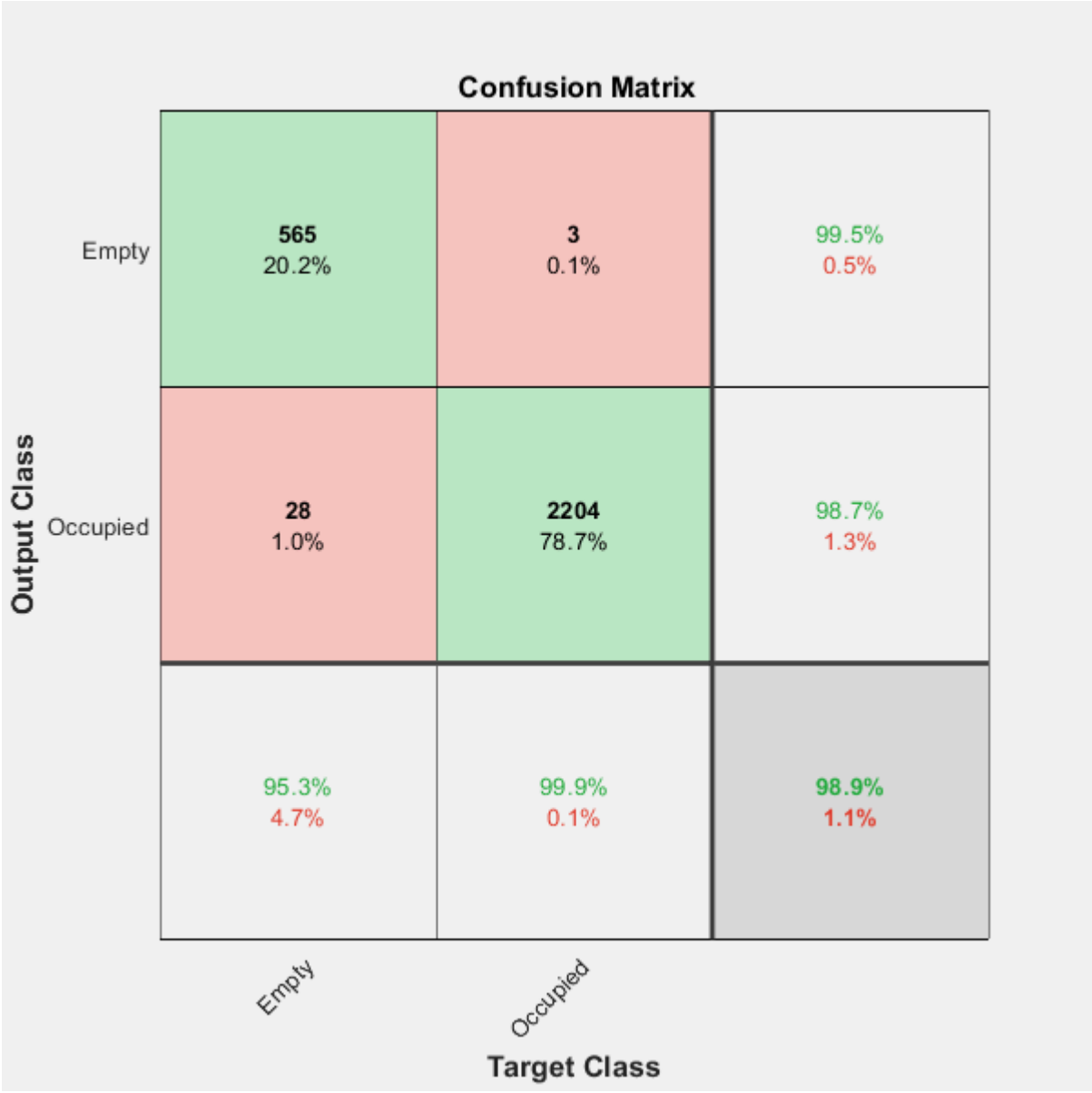
```
Test Features loaded
Elapsed time is 0.250680 seconds.
```

```
testAccuracy = mean(Occupancy(:) == YPred');
```

```
fprintf('The test accuracy of the classifier %f %%.\n',testAccuracy*100);
```

```
The test accuracy of the classifier 98.892857 %.
```

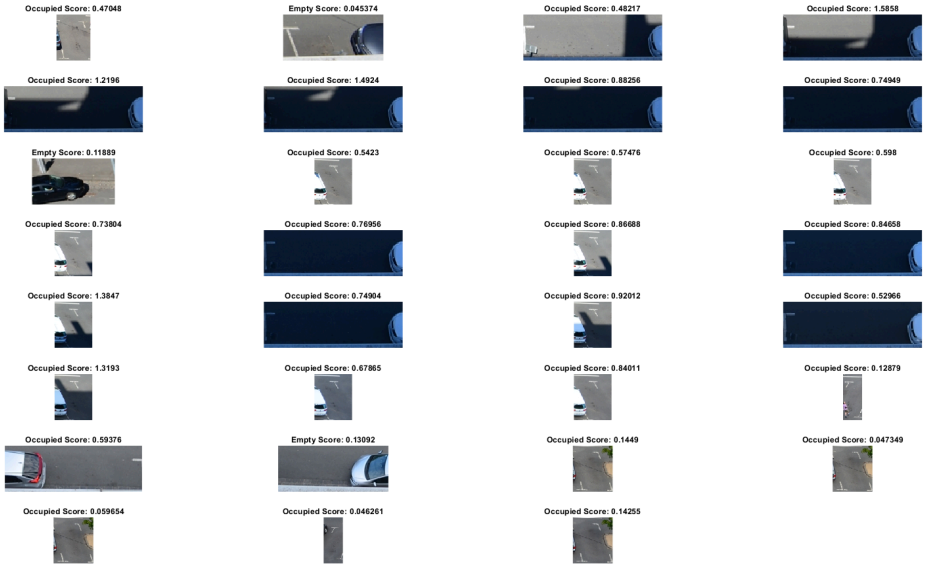
```
figure
plotconfusion (categorical(AnnotationTable'), categorical(YpredClass'))
```



Visualise some wrongly classified slots

```
WrongSlots = find(categorical(YpredClass) ~= AnnotationTable);
figure
set(gcf, 'Units', 'Normalized', 'OuterPosition', [0, 0.04, 1, 0.96]);
for n=1:length(WrongSlots)
    subplot(8,4,n);
    imshow(cropImage{WrongSlots(n)});
```

```
    str = [char(YpredClass(WrongSlots(n)))...
        ' Score: ' num2str(max(probs(WrongSlots(n),:)))];
    title(str);
end
```



Visualise the occupancy

```
figure
AnnotatedImage = imread("FinalCodeSVM\BarryStreetData\DSC_0040.JPG");
emptySlots = 0;
occupiedSlots =0;
for n=1:28
    if YpredClass(n) == "Empty"
       AnnotatedImage =  insertShape(AnnotatedImage,'FilledRectangle',...
    ParkingSlots(n,:),'LineWidth', 1, 'Color', "cyan", 'Opacity', 0.4);
    emptySlots = emptySlots + 1;
    else
       AnnotatedImage =  insertShape(AnnotatedImage,'FilledRectangle',...
    ParkingSlots(n,:),'LineWidth', 1, 'color', "magenta", 'Opacity', 0.4);
    occupiedSlots = occupiedSlots + 1;
    end
end
imshow(AnnotatedImage)
text (100, 50,['Occupied slots: ' num2str(occupiedSlots)],'Color', 'magenta','FontSize',20);
text (100, 100,['Empty slots: ' num2str(emptySlots)],'Color', 'cyan','FontSize',20);
```

This concludes the tutorial.