BABEŞ BOLYAI UNIVERSITY, CLUJ NAPOCA, ROMÂNIA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

Video-based parking spot detection using machine learning

– MIRPR report –

Team members
Olga Ţurcan, Computer Science (in German), 732
Viorel Gurdiş, Computer Science (in German), 731
Raul-Robert Zavaczki, Computer Science (in German), 732

2019-2020

Abstract

Living in a big city, it's a common problem to find available parking spaces for your car. Having a real time parking space detection system based on images provided by CCTVs could considerably improve the parking experience at a relatively low cost.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Problem definition

Statistically the average time spent searching for a parking spot represents **7.8 minutes**. That is a waste of time and can also cause traffic congestion. There are several types of parking spaces monitoring systems, including counter-based and sensor-based, which try to solve the mentioned problem. Counter-based systems work by counting cars at the parking lot entrance, but their disadvantage is that they don't provide concrete informations about available spots, leaving the task of finding the place to the driver. A more precise solution could be sensor-based systems which provide availability information about each spot specifically. However, this solution implies higher costs (c.a **$40** per unit). An alternative approach would be using video provided by already installed surveillance cameras for real-time detection by implementing a computer vision system. This method relies on the detection of vehicles in delimited space areas and classification of a spot as available or occupied. The processed information will be delivered by a server in form of a web page and updated in real-time, which would make the solution accessible for everyone. That would definetely save time of the driver and considerably reduce traffic problems. Challenges that we should expect will be related to the scalability of the application. We should, for example, take into consideration that the application should work in different weather conditions. We also focus on the accessibility of the processed information in real-time, for which we will try to compare different AI algorithms and will choose the one with the best balance between accurate results and small processing time.

Chapter 2

Related work

One approach to video-based real-time parking space detection application was described in the academic paper of [3]. To minimize weather and lighting conditions influences and to maximize the accuracy of the results, the approach evaluates several combinations of feature extractors and machine learning algorithms. They used a self-built dataset containing ca. 10,000 samples, from which they extracted features like Color Histograms [RGB, HSV, YUV] (to distinguish between asphalt color and the cars, to solve problems with brightness), Gradient Histograms, Difference-of-Gaussian Histograms and Haar-like (to extract edge information). Three classifiers were trained and compared to each other based on features mentioned above: k-Nearest Neighbor, Linear Discriminant Analysis, Support Vector Machine. The final solution relies on HSV color histogram and Difference-of-Gaussian features and a SVM classifier, which reached an accuracy of 99.8%.

An alternate approach for vacant parking space detection is described in [1], which uses features extracted by a pre-trained CNN to train an SVM classifier for the detection of parking occupancy in a CCTV image sequence. The CNN extracts features from the publicly available PKLot dataset which consists of more than 12,000 images collected from 3 parking sites on different weather conditions. Consequently, the extracted features from images of the PKLot dataset were used to train and test a binary SVM classifier. The evaluation of the classification accuracy is done by cross validation on the PKLot dataset and the transfer learning ability on the Barry Street dataset, which was created for the purpose of that research and includes a sequence of images captured by a camera overlooking a street with marked parking places.

The binary classification using the deep features achieved consistently reliable results with an average accuracy of 99.7% across different weather conditions for the PKLot dataset. As transfer learning was a more challenging task because the classifier was required to recognize unfamiliar images, the classification of Barry Street images achieved the overall accuracy of 96.65%. It is worth mentioning

that the processing time for each image region representing a parking space is 0.067 seconds on a simple desktop computer which means it takes approximately 2 seconds to process all the parking spaces in an image and hence the solution is suitable for real-time applications without any dedicated hardware.

A more rudimentary approach for parking spots detection was used in [4], but they provided valuable information on the different approaches used to determine whether a parking spot is free or not. Also the FMPH dataset was introduced, consisting of 1,093 pictures depicting 25,139 parking spots during a 30-day timespan, taken in various weather conditions. For detecting the parking spots in an image, they provided an API that allowed an administrator to manually draw masks of each parking spot. Assuming that the camera rotation doesn't change, they mapped the masks for each subsequent image, allowing them to generate an 80x80 pixel image for each parking spot in an image, allowing them to normalize the inputs.

The most notably performer was an MLP(15,15) classifier, which got an accuracy of 88.2% on the mentioned FMPH dataset. Among the other tested approaches were kNN algorithms with k=1 and k=3, getting an accuracy of 82.2% and 82.3% respectively. Another approach discussed in the paper is using Logistic Regression to classify whether a parking spot is free or not, but it proved to be the most underperforming algorithm used, scoring only 74,7% accuracy.

Chapter 3

Proposed approach

Our research focuses on detecting the occupancy of parking lots from the images obtained by CCTVs. The algorithm we've implemented consists of a Convolutional Neural Network which was trained on 8000 images from the opensource [2] dataset which consists of 150x150 pixels patches (fig. 3.3) grouped into 2 categories: busy and free. Our Keras model with Tensorflow backend has the following architecture (fig. 3.1): One convolutional layer having 3x3 image kernels, followed by a max-pooling layer with 2x2 pool size, a flatten layer, which reshapes the tensor for the upcoming dense layer with ReLU as activation. To complete our CNN, we need to give it the ability to actually make predictions. We'll do that by using the standard final layer for a multiclass classification problem: the Softmax layer, a fully-connected (dense) layer that uses the Softmax function as its activation. Before begining the training, we added some configurations during the compilation step: As **optimizer**, we decided for the Adam gradient-based optimizer and for the **loss function** we used the sparse-categorical cross-entropy loss. The model was able to classify parking slots into occupied and free with 99% accruacy.

A different approach is using an already trained CNN and extending it to meet our classifying needs. For this approach, we chose VGG-16 CNN-Architecture with 'imagenet' weights, and extended it by adding a Flatten layer, a Dense(256) layer with ReLU as its activation function, followed by a Dropout layer for feature-recognition, and finally a Dense(2) layer to classify into two categories: busy or free. (fig. 3.2) Additionally, for this model we used a Stochastic gradient descent (SGD) optimizer and the sparse-categorical crossentropy loss function. With this approach, we obtained a 86.1% accuracy, with only 4 epochs of training.

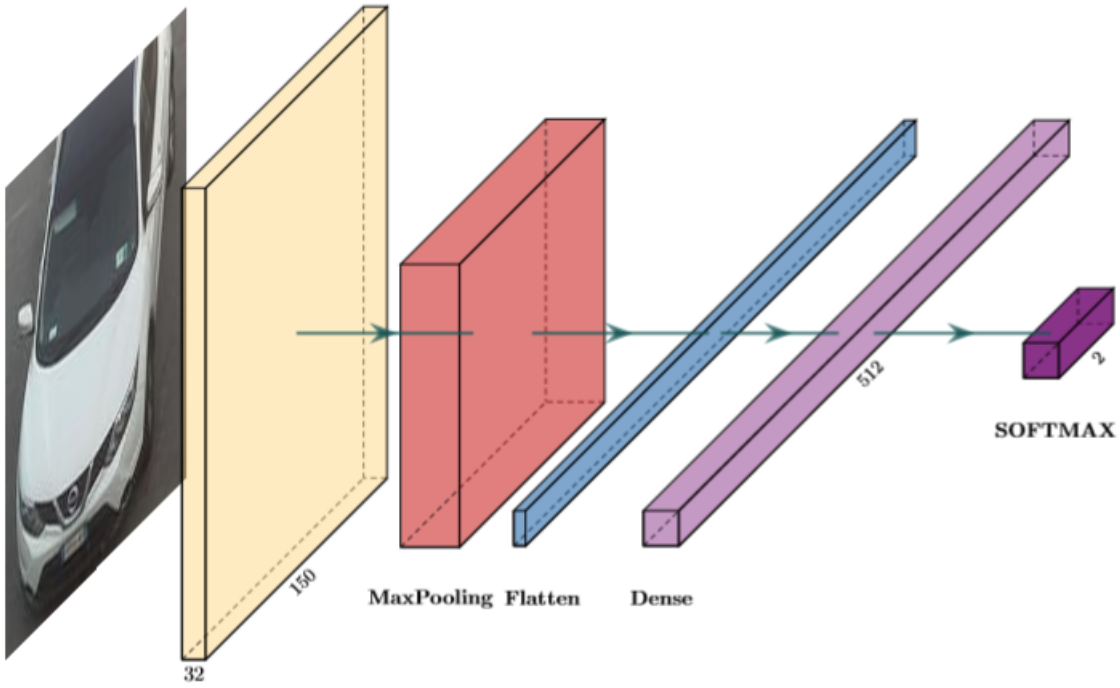

Figure 3.1: Architecture of the custom CNN

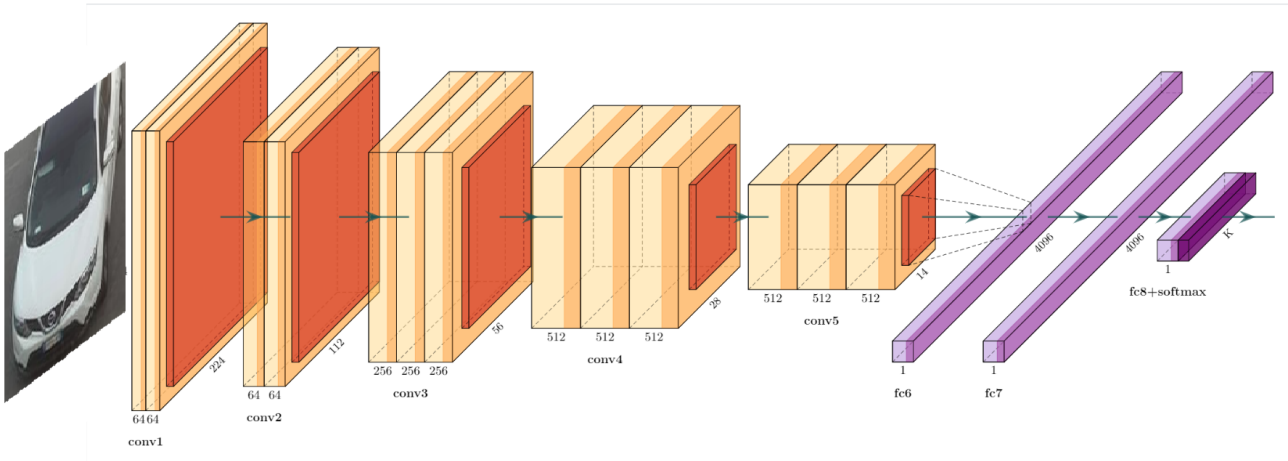

Figure 3.2: Architecture of the VGG-16

Figure 3.3: Example of 150x150 px patches, free and occupied

Chapter 4

Application

4.1 Useful Tools

The following tools were used for implementing above mentioned approaches.

- Tensorflow
- Keras
- Opencv
- Matplotlib

4.2 Methodology

For testing the models, we used a set of 20 full images from the [2] dataset. Each image contained 54 parking slots which were initially marked manually and their coordinates stored into a database. By uploading the image of the entire parking area, our algorithm is able to iterate through all labeled regions of interest (ROIs) stored in the database, retrieve the patch corresponding to its coordinates from the entire image, scale each obtained ROI patch from the original image to the coresponding resolution (150x150px) and then apply the Convolutional Neural Network model to predict if it represents an empty or an occupied slot. While using the first approach to test the classification on this 20 test images, the average percentage of correctly classified parking spots from an input image was 96%, as seen in the figure 4.1. With the second approach we managed to get a 79% accuracy.

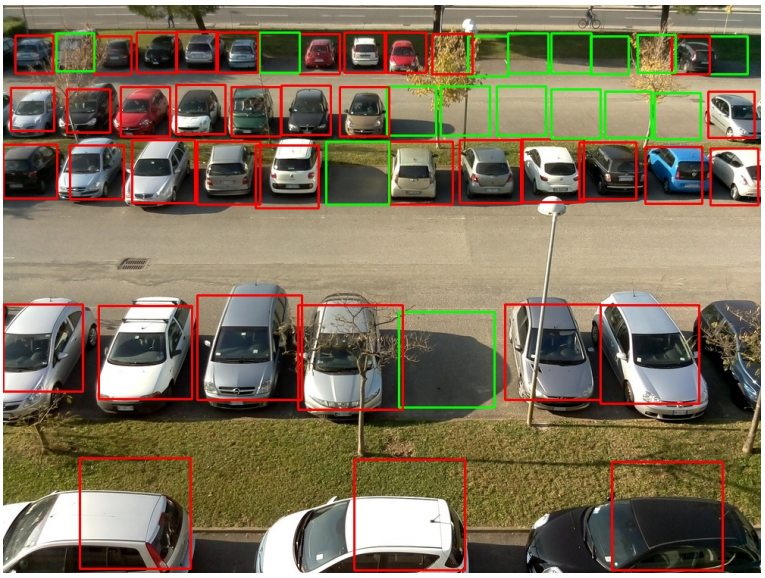

Figure 4.1: Obtained image after labeling the ROIs

4.3 Experimental Design

While our algorithm is capable of classifying parking spots in a single input image, the ultimate goal of our application is to obtain information about the availability of parking spots from a video. For the demo, we used a recorded sequence from a live stream video of a parking lot in Hicksville, New York, USA. The location of each parking spot was manually marked and saved as coordinates into a database. At a fixed interval of 8 seconds the current frame is extracted from the video and passed to the algorithm which processes each spot and updates the database with the predicted occupancy of each slot. The latest state of the data is then fetched by the client and displayed in form of a grid of red and green boxes, for occupied and free spots, respectively. The position of the boxes in the grid corresponds to those of the spots, so the user can easily identify the location of the empty parking spot. In addition, by clicking on a cell of the table, the exact spot is highlighted in the video.

In a real life scenario, the Youtube video could be replaced by a live stream from a parking lot surveillance camera so the app could provide real time information about the parking lot occupancy.

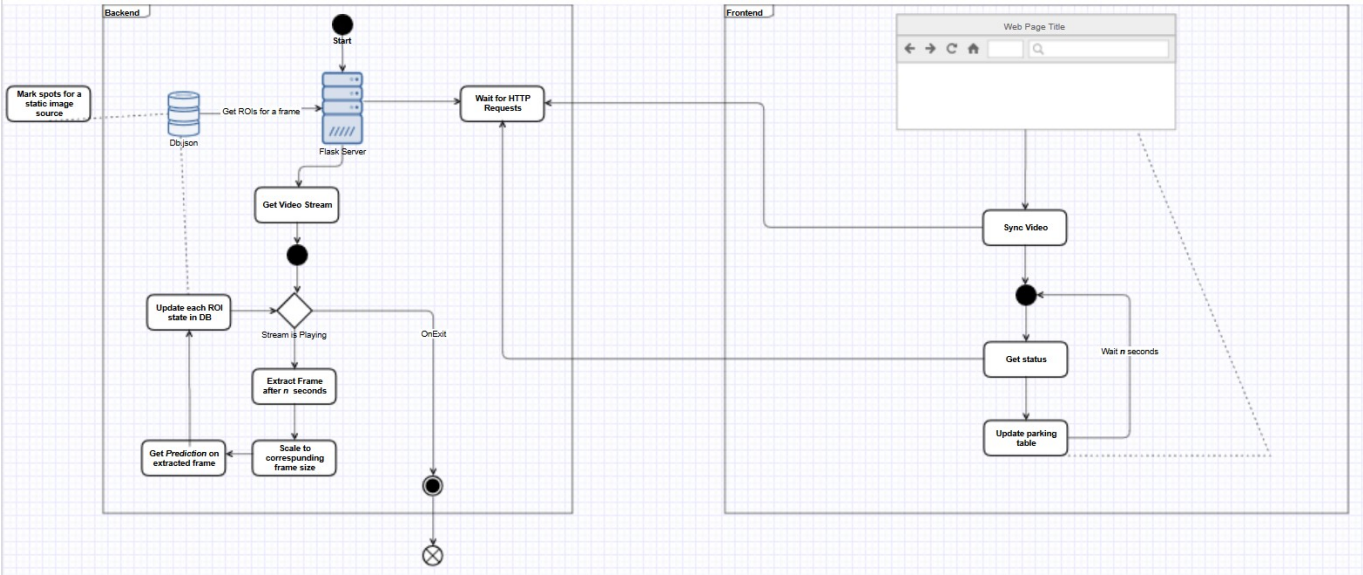

Figure 4.2: App flow diagram

4.4 Results

To measure the performance of our algorithms, we used 20 randomly selected frames from the video. Our CNN model with the custom architecture was able to classify parking spots with an overall average accuracy of 87%. Occupied slots were detected with a precision of 100% an a recall of 75%, while free slots achived 75% precision and 100% recall (results could be viewed in **Table** 4.1) This means that when a spot is classified as occupied, the classification is almost always done right - there were no free spots that the algorithm considered occupied. As a result we could say that if a free spot is present in the parking lot, it will be most probably detected. The classification of occupied slots has a lower precision, being sometimes classified as free. Some slots tend to be wrongly classified more often than others: those that are in the far back of the parking slot, due to the lower resolution of the slot cropped out of the full image, and also those that have a car with a color similar to the color of the ground.

While using the same approach to test the performance of the VGG-16 model, we obtained similar results, although surprisingly, the metrics were lower than for our custom model, achieving 82% accuracy. The tendency to classify free spots correctly better than occupied ones remained the same, the precision reaching 93% for busy and 72% for free spots, respectively.

Based on a number of factors, we decided to further use our custom model for the app. Firstly, the experimental results showed a slightly better performance in terms of accuracy, precision and recall for the custom model than for the VGG-16 model. Secondly, we wanted to experiment with training the model so it could achive better results, and doing this with the VGG-16 architecture would have

required significantly more computational power and execution time. This were the main factors that convinced us to further fine tune the custom model.

To improve the accuracy of the model, we first decided to increase the number of train epochs. We added Keras callbacks to monitor the accuracy, save the model after the completion of each epoch in case the accuracy improved and stop the training in case the accuracy did not improve in the last n epochs. This approach created a model that was trained in 10 epochs and achived a validation accuracy of 99%. As the classification result is dependent not only by the model but also by the input it receives, we decided to add a preproccessing step to the extracted frames. For this, we used the opencv implementation of Contrast Limited Adaptive Histogram Equalization for enhancing the contrast of the image.

Measuring the performance metrics of the algorithm after adding the improvements on the same 20 frames as before showed a significant increase in accuracy, precision and recall, all of them reaching 97-99% as seen in **Table** 4.1

Model	Accuracy	Busy		Free	
		Precision	Recall	Precision	Recall
Custom CNN model	0.86	1	0.77	0.74	1
VGG-16 model	0.79	0.87	0.75	0.70	0.85
Custom CNN model with improvements	0.98	0.99	0.98	0.97	0.99

Table 4.1: Experimental results

Chapter 5

Conclusion and future work

Our application tries to solve the problem of automatical detection of free parking spots, using image recognition from a live video stream. In the experimental phase, we tried to apply two algorithms for solving this classification problem. As our app is focused more on speed efficiency, rather than accuracy, we decided to keep the first algorithm, which is a custom CNN architecture implemented using Tensorflow Keras. The significantly smaller training time of the model compared to training the well-known VGG16 has allowed us to research the power of the Tensorflow framework, customizing different parameters and observing related improvements to the performance metrics. Of course, the app is just a proof of concept, meaning that the classification accuracy is directly linked to the video quality. Our algorithm would decrease in it's accuracy in another conditions than that on which it was trained, e.g. different camera angle, night light and weather, that affect the image quality. A solution to this problem would be to extend the dataset to include patches from all the described conditions. This would require more computational power to process the larger dataset in the training phase. Even then we would try to keep the computational time of the prediction low, so that the app would be scalable enough and show information that is up to date. The architecture of the model could also be improved by adding some new layers to the model or finetuning the parameters. If we would take into consideration all the above mentioned improvements, the app would definetely provide accurate classification regardless of weather, light and other conditions. This means that the provided information would be reliable enough for the app to be used in the real world.

Bibliography

[1] Debaditya Acharya, Weilin Yan, and Kourosh Khoshelham. Real-time image-based parking occupancy detectionusing deep learning. 2018.

[2] Giuseppe Amato, Fabio Carraraa, Fabrizio Falchia, Claudio Gennaroa, and Claudio Vairo. Cnrpark, a dataset for visual occupancy detection of parking lots. http://cnrpark.it/, 2015.

[3] Marc Tschentscher and Marcel Neuhausen. Video-based parking space detection. 2012.

[4] Roman Števaňák, Adrián Matejov, Marek Šuppa, and Ondrej Jariabka. Pkspace: An open-source solution for parking space occupancy detection. 2017.