On-Street Parking Spot Detection for Smart Cities

Sezer Gören[1], Dilan Fatma Öncevarlık[2], Kemal Doruk Yıldız[3] and Taha Zahit Hakyemez[4]
Department of Computer Engineering,
Yeditepe University,
Ataşehir, Istanbul, Turkey
[1]sgoren@cse.yeditepe.edu.tr
[2]dilanfatma.oncevarlik@std.yeditepe.edu.tr
[3]kemaldoruk.yildiz@std.yeditepe.edu.tr
[4]tahazahit.hakyemez@std.yeditepe.edu.tr

***Abstract*— Car parking in crowded cities is a big problem. Drivers have to do a blind search to find a free on-street parking spot. Blind searching not only causes traffic congestion but also fuel and time consumption. Indoor parking garages have sensors or light systems to point out free spots, unfortunately, an indoor approach is not applicable to the on-street parking problem due to its expensive nature. In our proposed solution, parking spots are monitored with roadside cameras. First, street images taken by roadside cameras are collected to form a dataset. Second, a Convolutional Neural Network (CNN) based on this dataset is built. Then the trained CNN analyzes a new street image to check if there is a free spot or not. In this paper, a mobile application is also developed and presented. Our mobile application takes the user's request and triggers the corresponding roadside cameras, and then notifies the driver about the available parking spots around the region and offers navigation to the spot upon the driver's confirmation.**

Keywords—Convolutional Neural Network, On-Street Parking, Parking Spot Detection, Image Processing

I. Introduction

On-street parking is to park a vehicle on the street, anywhere on or along the curb of streets, in contrast to off-street parking that is parking it in a parking garage or lot. Finding a free on-street parking spot is an everyday chore for drivers in crowded cities. Blind searching in the streets to find an empty parking spot is often a time-consuming and frustrating task. A recent study [1] reports that 30% of traffic congestion in crowded cities is caused by blind searching that lasts about 7.8 minutes for each attempt. Blind searching also increases fuel consumption and the carbon dioxide level in the air. Today, mobile applications and the Internet of Things (IoT) have found their place in many aspects of our daily lives. Thanks to IoT-based solutions, the strain of the parking problem can be alleviated.

In this paper, we propose a smart city parking management solution. An image processing system using Convolutional Neural Networks (CNN) is developed to detect available parking spots. Street images are taken with the roadside cameras to form a dataset to train CNN. After the training phase, upon a request made by a driver via our mobile application, the current image of the street is taken and analyzed for free on-street parking spots. Our mobile application notifies the driver about available parking spots. Unlike other indoor and outdoor systems, our system does not need any markers such as parking spot lines between cars. Our system can detect not only cars but also the empty spots and easily adaptable for any street without parking spot lines. With the help of our mobile application, the navigation to the free spot is possible.

II. Background and Related Work

The parking spot detection problem has different solutions based on fixed and mobile sensing. Fixed sensor-based methods require sensor installment and feasible for mostly indoor areas like mall garages, but not preferred in outdoor areas due to its expense. There are several projects [2] based on fixed and mobile sensing such as SFPark [3], Parknet [4], Parksense [5], Street Parking System [6], Smart Santander [7], Parking Spotter [8], Smart Parking [9], etc. Sonar, WiFi beacon, magnetic, ferromagnetic, infrared, and radar-based sensing technologies are commonly used. Detail comparison of these projects is given in [2].

Although some cities have installed sensor-based vacant parking spot detectors, the cost of this approach makes it unfeasible on a large scale. As an approach to implement a sustainable solution to the vacant parking spot detection problem in urban environments, the work [10] proposes fusing the information from small-scale sensor-based detectors with that obtained from exploiting the widely deployed video surveillance camera networks. However, vision-based detection methods in existing systems are not robust due to varying light intensities, occlusions, and their deployment costs can still be expensive.

CNN is used to classify [11-15] images, cluster them by similarity, and perform object recognition. CNN can be a promising solution to park management problems as a part of the vision-based detection approach. Parking-Stall Vacancy Indicator System [16] and UAV Assisted Parking [17] solutions use CNN to label parking spots in outdoor parking lots. The work in [18] uses CNN to capture license plate numbers by tracking the vehicles when they enter or leave the parking lot. However, the works in [16-18] target off-street parking i.e. parking lots. In this paper, we propose an on-street parking solution based on CNN. Our system is capable of empty on-street spots and does not have a limitation of parking stalls like the works in [16-18].

III. The Proposed System

In this work, an image processing model is developed to find free parking spots in the chosen sample street by using deep learning techniques. One of the project's aims is to show that, deep learning is applicable to this kind of problem with a small dataset. To achieve this, a deep learning model, server, and mobile applications are created. Our early results were presented in [19]. We first present the street and proposed system illustration and the sequence diagram in Fig. 1 and Fig. 2, respectively. When the user clicks the "Free Parking Spot Detection" button, it triggers an HTTP Request through the mobile application. Then, the server instructs the camera to

Fig. 1 Street Illustration (top) and Proposed System (bottom).

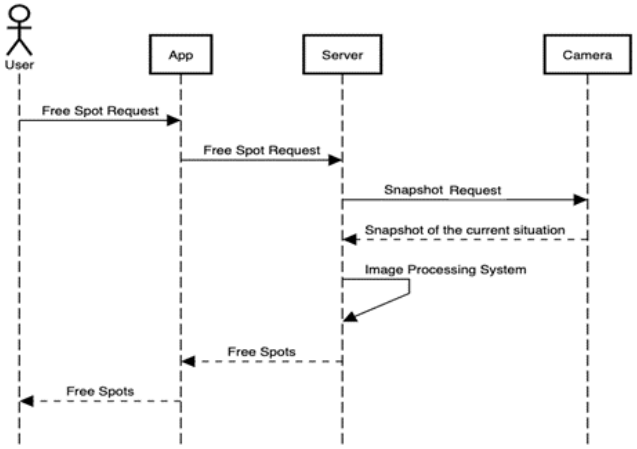

Fig. 2. Sequence Diagram of Proposed System.

take a picture of the area and send it back to the server. The server processes this image and colors it, then sends the processed image back to the mobile application.

In this work, we developed a classifier based on CNN. The flow of image classification is as follows: Several images of the sample street are collected. These images are labeled for different cases. After the labeling step, CNN is trained with these images. By training the CNN with the labeled set, the classifier can learn each class and its properties. The generated model is then tested by asking the classifier to predict new images that have never been used before. In this way, the success rate can be calculated.

A. Data Gathering

In this work, images are categorized either as "has a car in it" or "has not a car in it". It is important to have 2 categories because unlike other approaches, our approach is not an object detection approach, but an image classification approach. Our approach determines whether there is a car or not in the selected area. To create a dataset, images of the selected street is captured manually. There are a couple of important points which should be paid attention to. First, images should be captured from a higher level to create a bird's eye view with an angle. This is an important point because the plan is to integrate this model to already existed roadside units. Second, some of the images should be collected from the place where this system is going to be implemented.

B. Data Labeling

After collecting data from the sample street, the second step is to label the data. Data must be labeled because supervised deep learning algorithms classify images according to the labels. In this work, there are 2 labels so that all images should be labeled with one of them. Images are cropped into little images that only "contain a car in a spot" or "contain no car in a spot" as shown in Fig. 3.

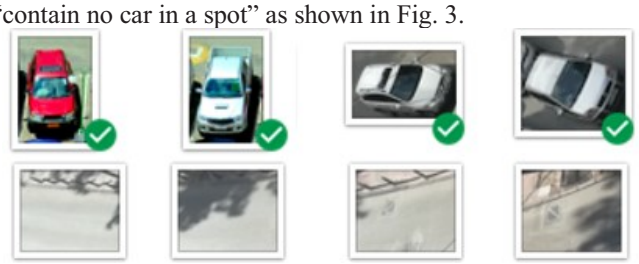

Fig. 3. Labelling Process.

C. CNN Design

During model design, 3D RGB arrays are used to represent images. For feature extraction, a kernel, also known as feature detector, traverses the image and generates new images. The kernel uses convolution to create new images shown in Fig. 4. In Keras API [20], there is a spatial convolution layer which is called Conv2D. By using Conv2D, with 3X3 kernel and 32 feature detectors for each image, 32 different feature maps are created. Then, the first convolutional layer is completed and 32 feature maps for every image are created.

An object's image can be taken at different angles and distances. However, all similar objects have similar features, for example, all the cars have the same features such as two headlights, a big front window, mirrors, etc. Pooling layers helps to find those features and make them stand out. With the help of pooling, even if the input has a different structure from other inputs, the model can still recognize it. We used MaxPooling2D [20] layer. MaxPooling is a sample-based discretization process that down-samples an input representation shown in Fig. 5, reduces its dimensionality and allows features contained in the sub-regions binned. As can be seen in Fig. 5, MaxPooling down-samples the Feature Map, which is created at Convolutional Layer, finds the maximum

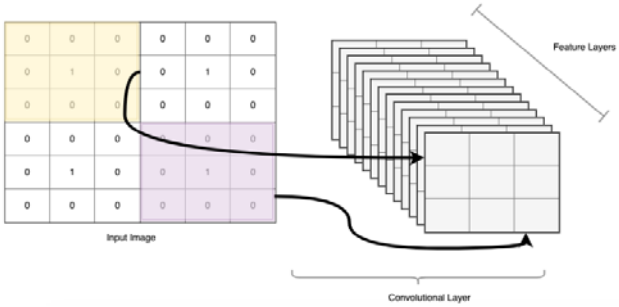

Fig. 4. Feature Detectors (32) Applied on Input Image [19].

number in the pool, and creates a newly created Pooled Feature Map. In this work, the MaxPooling pool is chosen as2X2 because the dataset which this model works on is a small one. While MaxPooling features stand out, some features could be faded away.

Next, we flatten the pooled feature maps shown in Fig. 6. Flattening [20] is a very simple method to convert 2D or 3D arrays into a 1D array because in full connection layer the image will be the input to Artificial Neural Network (ANN). ANN can only take a 1D array as an input.

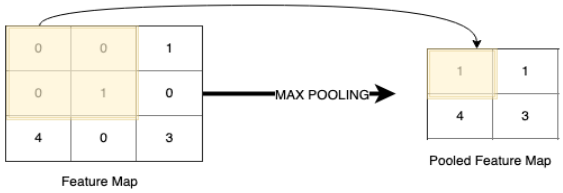

Fig. 5. MaxPooling [19].

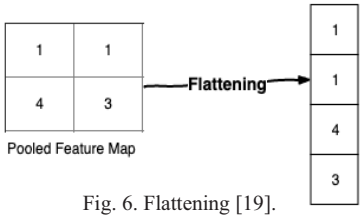

Fig. 6. Flattening [19].

Full Connection Layer is the last part of the CNN. After convolving image, the input is fed into an ANN. In ANN, there is an input layer that takes the flattened image as an input. There is an activation function which decides whether to let input to go to the next neuron or not. Every neuron takes the information from the previous neuron by multiplying it with a number that is close to zero. Information flows from neuron to neuron throughout the whole neural network. When it reaches the output layer and both loss and accuracy are calculated. After that, the neural network back propagates this loss and accuracy to the previous layers. Layers change the weights of information in each epoch and new accuracy and loss are calculated. ANN repeats this for the given number of epochs. To summarize, when a new image comes, the system first convolves it, applies max pooling, then flattens it, and lastly classifies the image.

D. Mobile & Server-Side Applications

A mobile application takes a parking spot request from the user and sends the request to the server. When the server receives this request, it triggers a web-service from the server-side. It takes a snapshot of that moment with the selected area's camera. This snapshot is processed by CNN which is also on the server-side to detect free spots and cars. CNN scans the area for cars. If there is a car in that area, the area is colored in blue, else it is colored in green.

As mentioned above, in the setup of the system, free spots must be specified, and the street perspective must be taken into account. An example of a street perspective is shown in Fig. 7.

The change rate in the street perspective is given in Eq. (1). First, a rectangle is drawn as half of the size of the first car

$$Slope = \left|\frac{\Delta y}{\Delta x}\right|,\ \Delta x = x_1 - x_2 \qquad (1)$$

shown in the bottom of Fig. 7. While sliding this rectangle from bottom to top in y-axis for every pixel, $x_{next} = x_{prev} -$ $Slope$ formula is used and the rectangle area is rescaled. Note that x_1 is used as the first x_{prev}. In this way, a rectangle can traverse the street perspective view. Every image block in the rectangle area is sent to CNN for detection. If the rectangle area which is sent to CNN contains a car in it, the rectangle area is painted as blue. If no car is detected in that area, then the rectangle area is painted as green. When labeling the regions is completed, web-service returns the colored image having regions with cars are colored in blue, empty spots are colored in green.

When web-service returns with the colored image, the mobile application is ready to inform the user. The mobile application is designed by using React Native [21]. React Native is a JavaScript framework for writing real, natively rendering mobile applications for iOS and Android. It is based on React, Facebook's JavaScript library for building user interfaces, but instead of targeting the browser, it targets mobile platforms. In this work, React Native is chosen because of its flexible, simple design. Moreover, when an application is developed in React Native, it can be published for IOS and Android at the same time.

Fig. 7. Representation of Street Perspective.

In the server-side, Python's Flask-RESTful [22] extension is used for building REST API. Flask is a micro web framework written in Python. This API takes the requests from the user, does the processes which are explained above, and returns the results to the user.

Our mobile application has "Find Me a Free Spot" and "Navigate Me" buttons. When the "Find Me a Free Spot" button is pressed, the user's location information is sent to the server, the server-side detects the closest available parking locations. The location information is sent from the server to the mobile application. The mobile application saves this information. when the "Navigate Me" button is pressed, the application creates a Google Maps link using the longitude and latitude values of the location of the free spot and the location of the user's mobile device. If Google Maps application is installed on the device, it opens and automatically draws the path from the current location to the destination location. If not, it does this via the default browser.

The screenshots of the mobile application are shown in Fig. 8. The top-left-hand-side image in Fig. 8 is the user interface for the parking spot request. The top-right-hand-side image shows the user interface options. the result of the free parking spot detection. User-interface has two options "Find Me a Free Spot" and "Navigate Me". The bottom-left-hand-side

Fig. 8. Screenshots of Mobile Application.

image is an example of the server's response after the "Find Me a Free Spot" request. The bottom-right-hand-side image is an example snapshot of the application for navigation to the free spot after the "Navigate Me" request.

IV. Tests and Results

While optimizing CNN, several different neural networks with different activation functions are tried. 2D Convolutional has 32 or 64 different kernels with 3X3 sizes. MaxPooling layers have 2X2 or 3X3 feature extractors. 3 different activation functions (*elu, selu, relu*) are combined with a convolutional block. The *relu* activation function gave the best results and two convolutional blocks with 32 kernels with 3X3 sizes and MaxPooling with 2X2 feature extractors maximize the test results. We tested different areas of the same street at different angles and measured response times. The average of total response time was measured as around 15s on a Windows-running PC with 16GB RAM, Intel i7.7 2.8GHz processor.

V. Conclusion and Future Work

In this paper, we demonstrated that deep learning can be efficiently applied to the on-street parking management problem. The current application has a navigation feature so that the user can easily reach the free spots from his/her location. As future works, the system will be extended with more cameras to monitor more streets. The detection and classification of soft shoulders, curbs, and guardrails [23] will be included in the current system.

References

[1] R. Arnott and E. Inci. "An integrated model of downtown parking and traffic congestion," *Journal of Urban Economics*, vol. 60, no. 3, pp. 418–442, 2006.

[2] R. L. P. B. Cristian Roman, "Detecting On-Street Parking Spaces in Smart Cities: Performance Evaluation of Fixed and Mobile Sensing Systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 7, pp. 2234 – 2245, 2018.

[3] D. G. Chatmana, M. Manville, "Theory versus implementation in congestion-priced parking: An evaluation of SFpark, 2011–2012," *Research in Transportation Economics*, vol.44, pp. 52-60, 2014.

[4] S. Mathur et al., "Parknet: drive-by sensing of road-side parking statistics," *in Proc. 8th Int. Conf. Mobile Syst., Appl., Services*, 2010, pp. 123–136.

[5] S. Nawaz, C. Efstratiou, and C. Mascolo, "Parksense: A smart phone based sensing system for on-street parking," *in Proc. 19th Annu. Int. Conf. Mobile Comput. Netw*., 2013, pp. 75–86.

[6] Z. Zhang, X. Li, H. Yuan, and F. Yu, "A street parking system using wireless sensor networks," *Int. J. Distrib. Sensor Netw*., vol. 9, no. 6, p. 107975, 2013.

[7] J. Lanza et al., "Smart city services over a future Internet platform based on Internet of Things and cloud: The smart parking case," *Energies*, vol. 9, no. 9, p. 719, 2016.

[8] Ford. (2015). *Parking Spotter*. Accessed: Apr. 28, 2017. [Online]. Available:https://media.ford.com/content/fordmedia/fna/us/en/news/2015/01/06/mobility-experiment-parking-spotter-atlanta.html

[9] S. Parking. (2015). *On-Street Parking*. Accessed: Apr. 28, 2017. [Online]. Available: http://www.smartparking.com/on-street.

[10] X. Sevillano, E. Màrmol, V. Fernandez-Arguedas, "Towards smart traffic management systems: Vacant on-street parking spot detection based on video analytics," *17th International Conference on Information Fusion (FUSION)*, pp. 1-8, 2014.

[11] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, 1(4):541–551, 1989.

[12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[13] P. Sermanet, S. Chintala, and Y. LeCun, "Convolutional neural networks applied to house numbers digit classification," *IEEE 21st International Conference on Pattern Recognition (ICPR)*, pp. 3288–3291, 2012.

[14] P. Sermanet and Y. LeCun, "Traffic sign recognition with multi-scale convolutional networks," *International Joint Conference Neural Networks (IJCNN)*, pp. 2809–2813, 2011.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, pp.1097–1105, 2012.

[16] M. S. E. S. M. J. Sepehr Valipour, "Parking-Stall Vacancy Indicator System, Based on Deep Convolutional Neural Networks," *IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pp. 655 – 660, 2016.

[17] X. Li, M.C. Chuah, S. Bhattacharya, "UAV Assisted Smart Parking Solution," *International Conference on Unmanned Aircraft Systems (ICUAS),* 2017.

[18] H. Bura, N. Lin, N. Kumar, S. Malekar, S. Nagaraj, K. Liu, "An Edge Based Smart Parking Solution Using Camera Networks and Deep Learning," *IEEE International Conference on Cognitive Computing,* 2018.

[19] D.F. Oncevarlik, K.D. Yildiz, S. Gören, "Deep Learning Based On-Street Parking Spot Detection for Smart Cities," *Proceedings of 4th International Conference on Computer Science and Engineering (UBMK'19)*, 2019.

[20] Keras, "Keras: The Python Deep Learning library," [online] Available: https://keras.io/

[21] B. Eisenman, "Learning React Native." O'Reilly | Safari, O'Reilly Media, Inc., www.oreilly.com/library/view/learning-react-native/9781491929049/ch01.html, 2019.

[22] K. Burke, K. Conroy, R. Horn, F. Stratton, G. BINE, "Flask RESTful," [online] Available: https://flask-restful.readthedocs.io/en/latest/, 2018.

[23] A. Seibert, M. Hähnel, A.Tewes, R. Rojas, "Camera based detection and classification of soft shoulders, curbs and guardrails," *IEEE Intelligent Vehicles Symposium*, 2013.