Deep Learning Based On-Street Parking Spot Detection for Smart Cities

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

***Abstract*— Parking is a big problem especially in populated cities. It is sometimes very difficult to find free on-street parking spots. Drivers have to do a blind search to find a free spot. Blind searching is not only time and fuel consuming, but also causes traffic congestion. Indoor parking garages have sensors or light systems to indicate free spots. However, indoor approach cannot be used for the on-street parking problem. To solve this, an image processing system is developed, and available parking spots nearby are detected using deep convolutional neural networks. With the help of road-side cameras, the system is first trained to detect free street parking spots. After the training phase, a frame taken from a roadside camera is analyzed whether there is a free spot or not. A mobile application is also developed which takes the request and triggers the corresponding road-side camera, and then notifies the driver about the available parking spots around the region.**

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

I. Introduction

Nowadays, mobile applications and Internet of Things (IoTs) have found their place in many aspects of our daily lives. Thanks to constant improvements in hardware, both in computation power and price, make these devices even more ubiquitous. Parking management with IoT is also gaining popularity and significance.

Finding a free on-street parking spot is an everyday chore for drivers in populated cities. The traditional method of circling around the parking lots or streets to find a spot (blind search) is inefficient, time-consuming, and frustrating. In a recent study [1], it is indicated that the 30% of traffic congestion in crowded cities is caused by drivers searching for parking spots which last about 7.8 minutes for each attempt. In addition, blind searching also increases fuel consumption and carbon dioxide level in the air.

In this paper, we propose a smart city parking management solution. In our solution, an image processing system is developed, and available parking spots nearby are detected using Convolutional Neural Networks (CNN). In our approach, streets are monitored by the road-side unit cameras and street images are collected for the learning phase of CNNs. As the proof-of-concept, a camera is placed on top of a building to monitor the sample street shown in Fig.1. As it can be seen in Fig.1, the chosen street has many features. It has both parking and non-parking spots as well as a street intersection. This street was chosen because of these features since it is a perfect example for the proof-of-concept. The sample street shown in Fig. 1 is an excellent case to prove the system to work in real time. The camera takes images of the street and, these images are used to train the system.

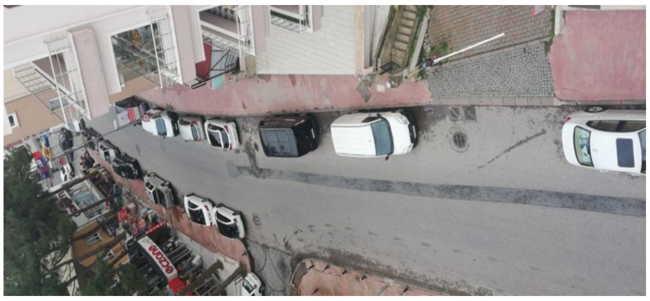
Fig. 1. Sample Street Image

Unlike other indoor and outdoor systems, our system does not need any markers such as parking spot lines between cars; our concern is only to detect parked cars or empty spots. With our approach, our system is easily adaptable for any street without spot lines.

An image dataset from our sample street is collected, our system is trained with these images. 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 sends a notification to the driver about available free parking spots and shows the optimum route to the spot. When a driver leaves the spot, he/she can use the system to locate his/her car on his/her return.

II. Background

Parking spot detection problem has different solutions such as with sensors [2, 3], video analytics [4], and laser line scanners [5]. Each approach has advantages and disadvantages. Some typical properties of these approaches are as follows:

- Most of them use sensors, lasers, or other hardware to detect if there is a car in the spot or not. This method requires sensors for every parking spot and not applicable for outdoor. However, in indoor parking like mall garages, it is much easier to set up and use, but more expensive for the outdoor parking case.
- Conventional video/image processing approaches have a problem with non-parking spots. They cannot identify non-parking spots from parking spots. For example, conventional approaches perceive garage entrances as a parking spot.
- Moreover, conventional approaches only detect a free spot after a car leaves it. If that spot is already is free, they cannot perceive that.

One of the most known on-street parking systems is funded by the U.S., whose name is SFPark [6]. This project has a total budget of 4.2 million dollars, and it adopts a wireless sensor network structure. Parking sensors and parking meters collect data via a wireless network. According to the authors, SFPark reduces the searching time 43% and, this affects the traffic positively. Traffic is reduced by 8%. Despite the success of the project, due to the high cost of sensors, SFPark did not spread across the country.

If we compare the sensor versus vision-based approach, we can clearly say that vision-based model is much more cost efficient. As video/image processing-based approach, CNN can be a promising solution to park management problem. CNN is used to classify images, cluster them by similarity, and perform object recognition. CNN was firstly introduced in [7] to recognize handwritten ZIP code in 1989, and later extended to recognition and classification of various objects such as hand-written digits [8], house numbers [9], traffic signs [10], and more recently 1000-category ImageNet dataset [11]. One of the related works, Parking-Stall Vacancy Indicator System [12] uses CNN to label parking spots.

In Parking-Stall Vacancy Indicator System [12], authors also attempt to solve the outdoor parking problem. Our proposed system is a vision-based approach which uses CNN similar to [12]. However, the work in [12] focuses on detecting the parking stalls, not the empty spots. On the other hand, our system is capable of empty on-street spots and does not have a limitation of parking stalls like the work in [12]. Due to this feature, our proposed system can easily be adapted to any outdoor area.

III. Proposed System

In this work, an image processing model is created to find free parking spots in the chosen sample street by using deep learning techniques unlike other parking spot detection solutions. Moreover, one of the project's aim is to show that, with a small dataset, deep learning is applicable to this kind of problem. To achieve this, a deep learning model and mobile application are created.

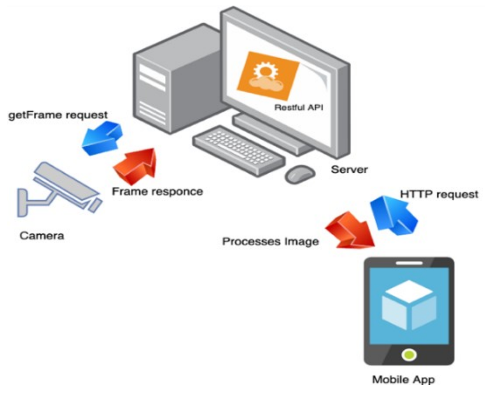

Fig. 2. Overview of Proposed System.

Before giving the details of the model and mobile application, we first present the overview and the flowchart of the proposed system in Fig. 2 and Fig. 3, respectively. When user clicks the "Free Parking Spot Detection" button, it triggers an HTTP Request from the server, server request from camera current picture of the area then camera takes picture and sends it to the server. Server processes this image and colors it, sends the processed image back to the mobile application.

The flow of image classification shown in Fig. 4 is as follows:

1. Defining the problem and collecting the data for this problem. Regarding outdoor parking spot detection, we collect images of the street with parked cars.
2. Labeling input images for different cases. It is a better approach to label images manually, after the labeling step, deep learning model can learn and start to make predictions by itself.
3. Training the system with the labeled training set, classifier can learn each class and their properties. This part is explained in more detail.
4. Last step, testing the model by asking the classifier to predict new images which has never beed used before. By this way, the success rate can be calculated.

To get the best and stabilized results, a classifier which is specified above for this task is developed. To create this new classifier, CNNs are chosen instead of picking parameters manually. Note that CNN can identify parameters by itself and no parameters can be missed.

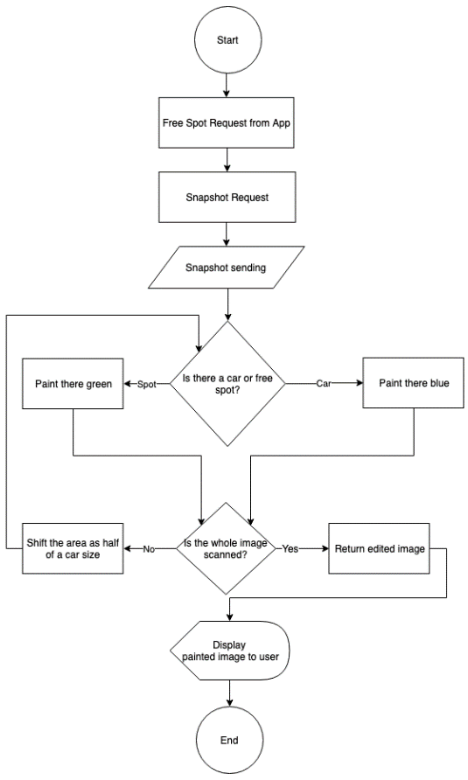
Start
Free Spot Request from App
Snapshot Request
Snapshot sending
Is there a car or free spot?
Paint there green
Spot
Car
Paint there blue
Shift the area as half of a car size
No
Is the whole image scanned?
Yes
Return edited image
Display painted image to user
End

Fig. 3. Flowchart of Proposed System.

A. Data Gathering

In this work, images categorized as "has a car in it" and "has not a car in it". It is important to have 2 categories because unlike other approaches, this approach is not object detection. It is an image classification, which understands is there a car or not in the selected area. To create a dataset, images of the selected street is captured manually. There are couple of important factors which should not be disregarded.

First one is, images should be captured from a higher point to create bird's eye view with angle. This is an important point because, the plan is to integrate this model to already existed road-side units. When images are captured, this point has been cared. The second one is, images should be collected from a place where this model can be applied, because one of the purposes of this project's aim is to make this system work with a small dataset. If a dataset is collected from the place which is intended to implement the system, the latency is shorter.

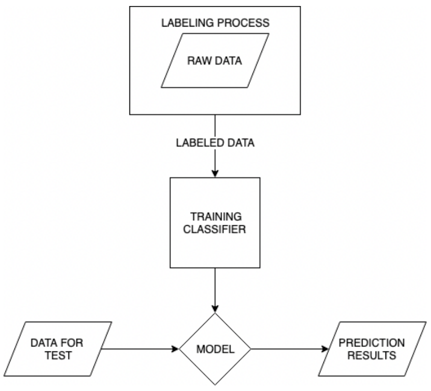

Fig. 4 Flowchart of Image Classification

B. Data Labeling

After collecting data from 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 cropped one by one into little images which only contain car or road in it. Images with cars in it go in to "car" folder shown in Fig. 5 and images with no car in it goes into "road" folder shown in Fig. 6. Note that we do the labeling process manually.

Fig. 5. Car Labeled Dataset

Ekran Resmi 2019-0...14.00.46
Ekran Resmi 2019-0...14.00.53
Ekran Resmi 2019-0...14.00.57
Ekran Resmi 2019-0...14.01.03
Ekran Resmi 2019-0...14.01.20
Ekran Resmi 2019-0...14.03.09

Fig. 6. Road Labeled Dataset

C. CNN Design

During model design, 3D RGB arrays are used to represent images shown in Fig. 7. For feature extraction, convolution is used. In the convolution step, there is a kernel as known as feature detector which traverses all around the image and creates new images.

$$f * g \equiv \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau \tag{1}$$

Kernel uses (1) to create new images. In Keras API [13], there is a spatial convolution layer which is called Conv2D. By using Conv2D, with 3 by 3 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.

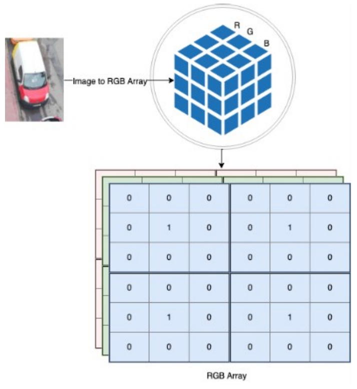

Fig. 7. Image to RGB Array

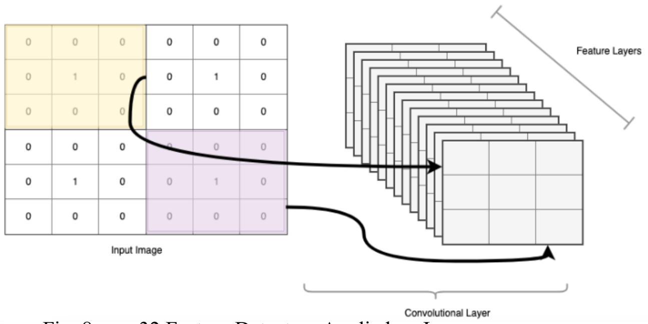

Fig. 8. 32 Feature Detectors Applied on Image

An object's image can be taken at different angles and distances. However, all images of the same object have the same features, like having two headlights, big front window, mirrors and 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 shape from other inputs, the model can still recognize it. We used MaxPooling2D [13] layer. Max pooling is a sample-based discretization process which down-samples an input representation (image, hidden-layer output matrix, etc.) shown in Fig. 9, reduces its dimensionality and allows features contained in the sub-regions binned.

As it can be seen in the Fig. 9., Max Pooling down-samples the Feature Map which is created at Convolutional Layer, finds the maximum number in the pool, and creates a newly created Pooled Feature Map. In this work, Max Pooling pool is chosen as 2 by 2 because the dataset which this model works on is a small one. While max pooling features stand out, some features could be lost. Due to this, small pool size is used.

Next, we flaten the pooled feature maps. Flattening [13] is a very simple method to convert 2D or 3D arrays into 1D array,

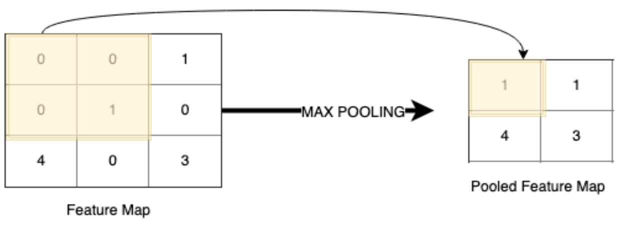

Fig. 9. Max Pooling.

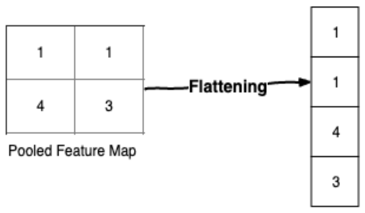

Fig. 10. Flattening.

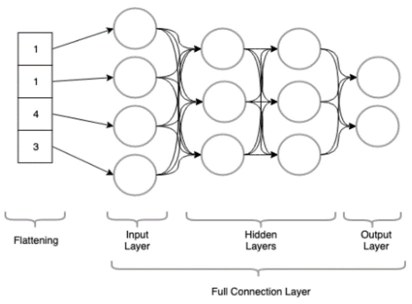

Fig. 11. Artificial Neural Network

because in full connection layer the image will be the input to ANN. ANN can only take 1D array as an input.

Full Connection Layer is the last part of the CNN. After convolving image, to start to the learning process, input simply put in an ANN. In ANN, first there is an input layer, it takes the flattened image as an input. There is an activation function which lets or not an input to go to the next neuron. Every neuron takes the information from the previous neuron by multiplying it with a number which is close to zero. Information flows from neuron to neuron for the whole neural network. It gets to the output layer and calculates accuracy and loss. After that, neural network back propagates this loss and accuracy to the previous layers and layers change the weights of information, in each epoch and calculate new accuracy and loss. ANN repeats this for the number of epochs which is defined before. 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 is developed which takes parking spot request from the user, sends the request to the server. When the server receives this request, it triggers a web-service from server-side. It takes a snapshot of that moment with the selected area's camera. This snapshot is used by CNN which is also in 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 painted as blue, else it is painted as green.

As mentioned above, in the setup of the system, parkable areas must be specified. Doing this specification, street perspective must be considered. An example street perspective is shown in Fig. 12.

To calculate the change rate in street perspective, Eq. (2) is used.

$$Change\_Rate = \left|\frac{Y}{\Delta x}\right|, \qquad \Delta x = x_1 - x_2 \tag{2}$$

To apply this formula to the image, first a rectangle is drawn as half of the size of the first car shown in the bottom of Fig. 12. While sliding this rectangle from bottom to top in y-axis for every pixel, $x_{next} = x_{prev} - Change\_Rate$ formula is used and rectangle area is rescaled. Note that x_1 is used as the first x_{prev}. By 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 blue layer of the rectangle area is changed to 255. If no car is detected in that area, then green layer of the rectangle is changed to 255. When labeling the regions are completed, web-service returns the colored image having regions with cars are painted in blue, empty spots are painted in green.

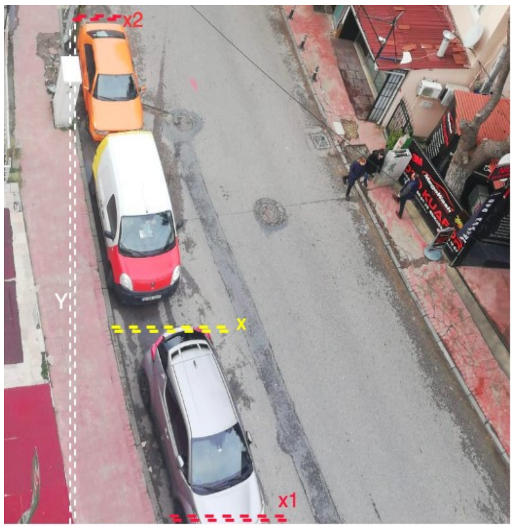

Fig. 12. Representation of Street Perspective.

When web-service returns with the colored image, mobile application is ready to inform the user. Mobile application is designed by using React Native [14]. 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.

In the server-side, Python's Flask-RESTful [15] extension is used for building REST API. Flask is a micro web framework written in Python. This API takes the requests from the user, do the processes which are explained above, and returns the results to the user.

The screenshots of the mobile application are shown in Fig. 13. The left-hand side image in Fig. 13 is the user interface for the parking spot request. The right-hand side image in Fig. 13 is the result of the free parking spot detection. As it can be seen in Fig. 13, the developed mobile application is simple and flat. User's only concern is to click the button which is labeled as "Find me a spot", then a request is sent to the server.

In server-side, there is a function which prepares rectangle image with the size of half of a car and sends this rectangle image to CNN. CNN's only concern is to detect a car or free spot in that rectangle image. After detection, CNN sends the result to the server and server-side shifts the rectangle and again sends the new rectangle image to CNN. This process continues until the whole image is scanned. When all rectangle images are processed, the original image is painted according to the detections. The painted image is then sent back to the mobile application. In Fig. 14, example outputs of the system are presented.

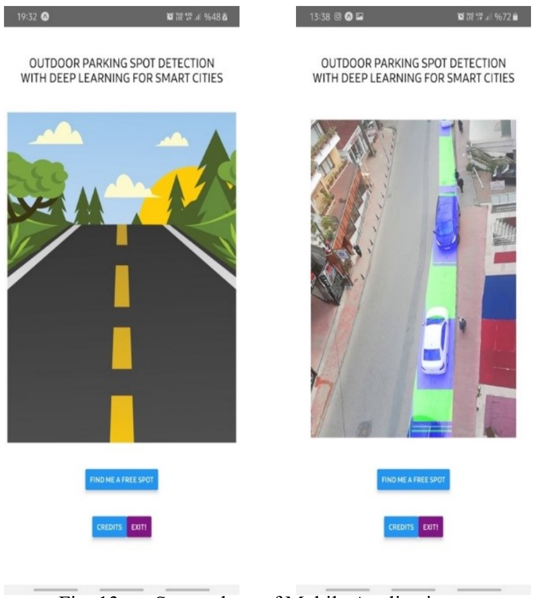
19:32
13:38
OUTDOOR PARKING SPOT DETECTION WITH DEEP LEARNING FOR SMART CITIES
OUTDOOR PARKING SPOT DETECTION WITH DEEP LEARNING FOR SMART CITIES
FIND ME A FREE SPOT
CREDITS
EXIT!
FIND ME A FREE SPOT
CREDITS
EXIT!

Fig. 13. Screenshots of Mobile Application.

IV. Tests and Results

While optimizing CNN, several different neural networks with different activation functions are used to find out the best result. In convolutional block, the neural network has 2D Convolution and MaxPooling. 2D Convolutionals has 32 or 64 different kernels with 3 by 3 sizes. MaxPooling layers have 2 by 2 or 3 by 3 feature extractors. 3 different activation functions (*elu, selu, relu*) are combined with convolutional block. In Table I, test results are shown.

By analyzing test results, *relu* activation function gave the best results and two convolutional blocks with 32 kernels with 3 by 3 sizes and MaxPooling with 2 by 2 feature extractors maximize the test results. When one convolutional block is used, enough features could not detect but when three convolutional blocks, 64 kernels or 3 by 3 feature extractor used, it caused overfitting. Tests are done with one or more hidden layers. But the system started to overfit when hidden layer number is increased. 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.

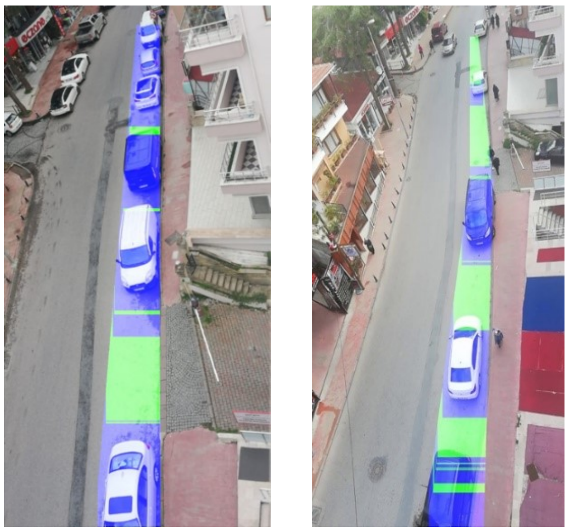

Fig. 14. Example Outputs.

TABLE I. Optimization Of CNN.

	ANN with 0 Hidden Layer, 128 Neuron in Input Layer x 10 epoch		
	elu	**selu**	**relu**
1x(Conv2D(32,(3x3)), MaxPooling(2x2))	76%	56%	86%
2x(Conv2D(32,(3x3)), MaxPooling(2x2))	86%	62%	**92%**
3x(Conv2D(32,(3x3)), MaxPooling(2x2))	80%	54%	82%
1x(Conv2D(64,(3x3)), MaxPooling(2x2))	76%	42%	82%
2x(Conv2D(64,(3x3)), MaxPooling(2x2))	84%	48%	90%
3x(Conv2D(64,(3x3)), MaxPooling(2x2))	72%	44%	76%
1x(Conv2D(32,(3x3)), MaxPooling(3x3))	74%	48%	74%
2x(Conv2D(32,(3x3)), MaxPooling(3x3))	70%	54%	78%
3x(Conv2D(32,(3x3)), MaxPooling(3x3))	66%	52%	70%
1x(Conv2D(64,(3x3)), MaxPooling(3x3))	68%	34%	72%
2x(Conv2D(64,(3x3)), MaxPooling(3x3))	70%	38%	76%
3x(Conv2D(64,(3x3)), MaxPooling(3x3))	60%	36%	68%

V. Conclusion and Future Work

In this paper, we showed that deep learning can be efficiently applied to on-street parking management problem. In this work, as a proof-of-concept, a single camera and one street are used to evaluate the system.

As future works, the system will be extended with more cameras to monitor more streets shown as in Fig. 15. RaspberryPis will be considered for edge computing. The server will be placed in the Cloud to make it accessible from anywhere. In addition, the mobile application will have an improved user interface such that user can choose the location or system selects the closest location and give the results according to closest parameters. The future system will have navigation to the free spot, so that the user can easily find the free spots from his/her location. Fig. 16 presents mockups of future application.

In our current implementation, parkable regions are defined manually and done in advance. In addition to the above improvements, we will also automate the selecting parkable regions in the street. Another learning algorithm which can detect the road and the parkable road-sides automatically will be developed to decrease human intervention.

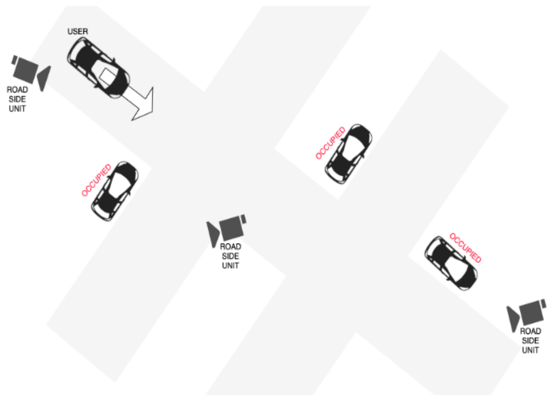

Fig. 15. Future System Illustration.

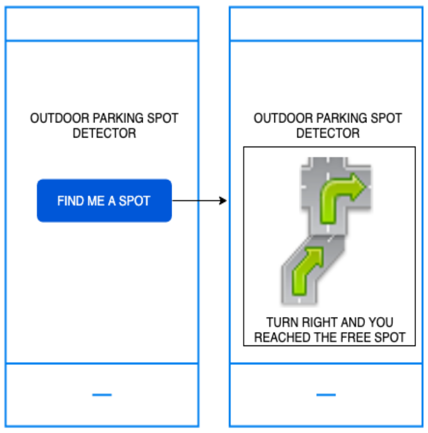
OUTDOOR PARKING SPOT DETECTOR
FIND ME A SPOT
OUTDOOR PARKING SPOT DETECTOR
TURN RIGHT AND YOU REACHED THE FREE SPOT

Fig. 16. Mockups of Future Work in Mobile Application.

REFERENCES

[1] Richard Arnott and Eren 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] U. Fastenrath, “Parking space detection,” [Online]. Available: https://patents.google.com/patent/US6266609B1/en. [Accessed 02/15/2019].

[4] 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.

[5] J. Zhou, L. E. Navarro-Serment, M. Hebert, “Detection of parking spots using 2D range data,” *15th International IEEE Conference on Intelligent Transportation Systems*, pp. 1280 – 1287, 2012.

[6] Daniel 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.

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

[12] 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.

[13] Keras, “Keras: The Python Deep Learning library,” [online] Available: https://keras.io/

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

[15] K. Burke, K. Conroy, R. Horn, F. Stratton, G. Bine, “Flask RESTful,” [online] Available: https://flask-restful.readthedocs.io/en/latest/, 2018.