An Edge Based Smart Parking Solution Using Camera Networks and Deep Learning

Harshitha Bura, Nathan Lin, Naveen Kumar, Sangram Malekar, Sushma Nagaraj, Kaikai Liu
Computer Engineering Department
San José State University (SJSU)
San José, CA, USA
Email: {harshitha.bura, nathan.lin, naveenkumar.bhuthakatanahalliramalingaiah, sangram.malekar, sushma.nagaraj, kaikai.liu}@sjsu.edu

***Abstract*—The smart parking industry continues to evolve as an increasing number of cities struggle with traffic congestion and inadequate parking availability. For urban dwellers, few things are more irritating than anxiously searching for a parking space. Research results show that as much as 30% of traffic is caused by drivers driving around looking for parking spaces in congested city areas. There has been considerable activity among researchers to develop smart technologies that can help drivers find a parking spot with greater ease, not only reducing traffic congestion but also the subsequent air pollution. Many existing solutions deploy sensors in every parking spot to address the automatic parking spot detection problems. However, the device and deployment costs are very high, especially for some large and old parking structures. A wide variety of other technological innovations are beginning to enable more adaptable systemsincluding license plate number detection, smart parking meter, and vision-based parking spot detection. In this paper, we propose to design a more adaptable and affordable smart parking system via distributed cameras, edge computing, data analytics, and advanced deep learning algorithms. Specifically, we deploy cameras with zoom-lens and motorized head to capture license plate numbers by tracking the vehicles when they enter or leave the parking lot; cameras with wide angle fish-eye lens will monitor the large parking lot via our custom designed deep neural network. We further optimize the algorithm and enable the real-time deep learning inference in an edge device. Through the intelligent algorithm, we can significantly reduce the cost of existing systems, while achieving a more adaptable solution. For example, our system can automatically detect when a car enters the parking space, the location of the parking spot, and precisely charge the parking fee and associate this with the license plate number.**

***Keywords*-Deep learning, edge devices, smart cities, smart parking.**

I. Introduction

Smart parking solutions are one of the basic and foremost requirements of smart cities [1]. According to [2], a motorist spends an average of 17 hours per year searching for a place to park his vehicle. This is the reason why the smart parking industry is on the rise and coming up with new solutions very often.

Many existing solutions use sensors at every parking spot to identify whether the spot is vacant or not. Installing sensors in every parking spot is a simple way to solve the automatic parking spot detection problem [3], [4], [5]. However, the device and deployment costs are very high, especially for some large and old parking structures. Another major problem is the limited sensor data, e.g., empty or not, without rich information like license plate number, vehicle or motorcycle identification. To get rich information, some solutions are deploying camera as the sensor in every parking spot. This will enable many new smart services, for example, users can use their license plate number to find their vehicle. However, this kind of system requires a robust network in the parking garage. Collecting all these camera data requires a huge amount of bandwidth, which poses significant challenges for the network infrastructure. Autonomous vehicles are another solution to solve parking problems [6]. But, the disadvantage [7] with them is that people will get off the car at the destination and leave the cars to find a parking spot on their own. In this case, if the car doesn't find a place to park it will end up circling the neighborhood, blocking others and wasting fuel. Robotic valet systems [8] can also be considered as a solution to this problem but they will involve very expensive and complex mechanical equipment. Some solutions using vision-based techniques to cover large sparking spaces in order to lower the sensor deployment cost [9], [10]. Deep learning based solution [9], which is based on the VGGNet family, has been proposed to solve the parking spot detection problem. However, the computation in the solution [9] requires the cloud for highly complex processing. Authors in [10] make use of miniAlexNet and reduced the layers to only have three convolutional layers each followed by a ReLu and max-pooling layer. However, covering a large area via computer vision means they lack features such as license place recognition which has been proved to be very useful in services like finding the vehicle online.

To balance the cost and service quality, we propose to monitor the parking lot using a distributed network of cameras, connected via a self-deployed WiFi mesh. One camera can cover a large area instead of covering individual parking spots. To enable smart services like finding the vehicle location via license plate number, we perform continuous vehicle tracking and feature extraction via the camera network. Specifically, some of the cameras with zoom-lens and motorized head can capture license plate numbers by tracking the vehicles when they enter or leave the parking lot. Other cameras with wide angle fish-eye lens will cover large parking spaces, and detect the parked parking lot via artificial intelligence. Deep learning algorithms will be used to identify occupied, vacant, and special parking spots, for example, disabled parking signs, carpool signs. Through the intelligent algorithm, we can significantly reduce the number of sensors and cameras of existing systems, while achieving a higher level of service quality. To further reduce the computational complexity of the deep learning algorithms, we propose a custom convolutional neural network that can be run on top of edge devices in real time. When compared with existing state-of-the-art approaches, our solution can achieve the same level of detection accuracy with a lower computational complexity. Along with the detected license plate numbers of vehicles, our system performs data fusion and object association from multiple cameras and enables applications such as keeping track of the duration of a vehicle's stay in a particular parking spot. The overall system will help motorists in navigating to a vacant spot with ease and will also help in charging them based on the exact amount of time the vehicle has been parked in the parking lot.

II. Problem Statement / Project Architecture

The motivation for this project comes from the fact that the quality of smart solutions available for parking management is not on par with the smart solutions available for other services required for smart cities such as waste management, resource management and utilities management. Transportation and parking being an important part of everyday life, there is a dire need to come up with smart and effective solutions. In this section we give a detailed description of our proposed system.

A. *Our proposed solution using a network of cameras*

The block diagram for the proposed system using camera networks is depicted in Fig. 1. The entire system consists of (i) A network of ground cameras which are placed close enough to the ground to capture license plate numbers of vehicles, (ii) A network of top-view cameras whose view when combined will cover the entire area of the parking lot, (iii) Edge devices i.e., an Nvidia Jetson Tx2 and Raspberry Pi, (iv) A cloud server and database which will be used to hold information such as the occupancy status of all the slots, license plate numbers of vehicles that are parked in the lot along with entry and exit time-stamps for each vehicle, (v) A web application which will display all the relevant and necessary information. A detailed description of each block of the system is given below.

1) Network of ground cameras: The ground cameras need to be appropriately focused as they are responsible for having a clear view of the license plate number of vehicles that enter and leave the parking lot. We choose the low cost Raspberry Pi (Rpi) camera and fit with a zoom lens

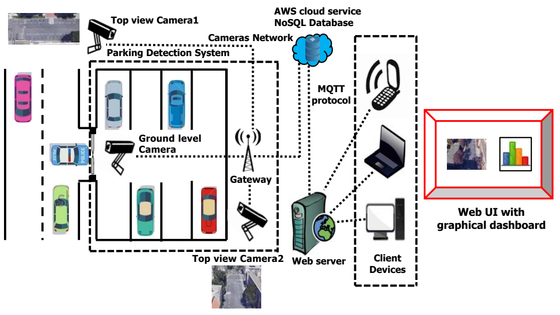

Figure 1. Block Diagram of our proposed solution using a network of cameras

to achieve the required focus. The Rpi runs algorithms for the license plate number recognition and vehicle tracking.

2) Network of top-view cameras: The top-view cameras have a bird's eye view of the entire parking lot. The view of all these cameras when combined will cover the whole parking lot. Rpi cameras have been selected for the top-view as well but they are fit with wide-angle lenses. The feed from these cameras is used to identify the occupancy status of each slot, performing feature extraction from the top-view and tracking vehicles until they are parked.

3) Edge Devices: The edge devices used in our system are Raspberry Pi and Nvidia Jetson Tx2. Both the ground cameras and top-view cameras are interfaced to an Rpi. The Rpi interfaced with the ground camera runs OpenALPR [11] to perform license plate number recognition. The Rpi also extracts features of vehicles that enter the parking lot using OpenCV and time-stamps their entry and exit times. All this information is sent to the server.

The top view cameras are also interfaced to an Rpi which is responsible for streaming its feed to Nvidia Jetson Tx2 via Wi-Fi mesh. The Jetson which receives the live feed time-stamps the entry of a vehicle and extracts its features. The time-stamp and features are compared to the entries in the database and if they match with any entry, the license plate number associated with that particular entry is used as the tracking ID for the vehicle. The vehicle is tracked until it comes to a halt in a parking slot. The newly occupied slot number is then associated with the vehicle's license plate number. This helps in making record of a vehicle's location in the parking lot. Tracking is done with help of a trained Tiny Yolo (based on Darknet framework) model run on top of the Nvidia Jetson Tx2. Tiny Yolo is very lite and is hence suitable for running on edge devices. Slot classification continuously runs on Tx2 with the help of a trained custom-designed neural network. The custom-designed neural network has only one convolutional layer and is very lite and is perfect for running on edge devices. It has an accuracy that is comparable to the standard AlexNet for small class numbers but is much faster.

4) Cloud server and database: The data generated from the edge devices is sent to the cloud using MQTT protocol. All the edge devices si registered to AWS IoT as “things”. AWS IoT allows tracking of all the devices connected to it. The IoT certificates and policies should be added to the device. Once the device is connected to AWS, a rule should be defined which will help in communicating with the database. We are using dynamodb for storing the license plate numbers, parking slot information and features of vehicles. All the information is sent to the database in json format. Once all the data is stored in the database, a flask backend application will connect to the database and display the information on the website.

5) Web Application: The web application will display all the useful information such as the vacant and occupied slots and will be integrated with Google maps to help users in navigating to vacant spots. It will contain a feature which will allow users to find out the exact location of their vehicle by entering their license plate number along with a feature which will allow users to pay their parking fare online. The logical flow of the entire system explained till now is represented in Fig. 2.

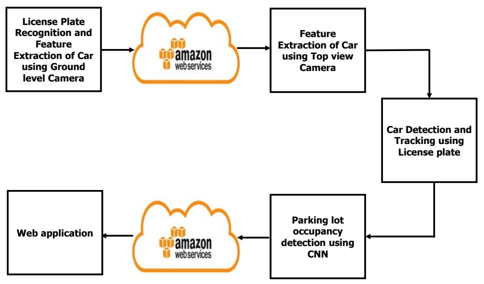
License Plate Recognition and Feature Extraction of Car using Ground level Camera
amazon webservices
Feature Extraction of Car using Top view Camera
Car Detection and Tracking using License plate
Parking lot occupancy detection using CNN
amazon webservices
Web application

Figure 2. Logical flow of our proposed solution

B. Our proposed solution using a network of LIDARs

Indoor parking garages have a very low ceiling and hence cameras installed on the top will not be able to cover a large area. Thus, camera networks will not work for indoor garages. To solve this problem, we utilize a low-cost LIDAR sensor to assist the detection. A traditional sensor based parking solution will require a sensor to be deployed under each parking slot. But our proposed LIDAR based solution will be able to cover 7 - 8 parking slots with only one LIDAR. The block diagram of this system is shown in Fig. 3.

The LIDAR is interfaced to a Raspberry Pi and a slot is classified as either occupied or empty based on the distance readings from the LIDAR. The LIDAR system will be combined with OpenALPR and a Raspberry Pi camera for detecting the license plate number of vehicles.

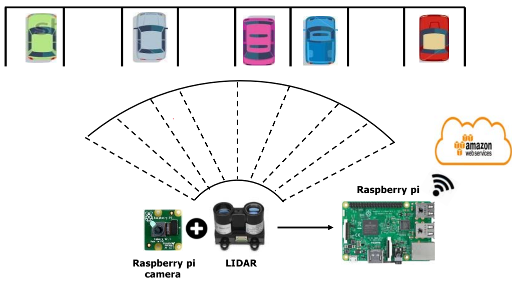

Figure 3. Block Diagram of our proposed solution using a network of LIDARs

III. Methods / System Design

A. Dataset

There is no AI (Artificial Intelligence) without IA (Information Architecture) [12]. Having a good all-round data set is very important for building good AI solutions. In our proposed solution we have performed object detection as well as object classification and hence two different types of datasets are required. They are discussed in the section below.

1) Dataset used for object classification: An open source dataset is available for parking lots and it is given in [10] and [13]. [13] called PKLot consisting of 12,417 images of parking lots and 695,899 images of parking spaces segmented and perspective transformed. [10] consists of a dataset which is an extension to [13] and it is called CNRPark. CNRPark adds about 12,000 images to the PKLot which were taken in different weather conditions, which have not been perspective transformed and some are occluded. However, Both of these datasets lack images of parking spots with special signs such as disabled signs and car pool signs. Our dataset collected in San Jose State University’s parking lot adds such images to those two datasets.

While creating our dataset with annotated individual slots from each frame of the video, an annotation mask has been defined using a tool called LabelMe [14] developed by MIT. After an entire frame has been segmented into individual slots, the image of each slot undergoes perspective transformation. With the help of perspective transformation it can be ensured that the angle in which the image has been taken will not effect the performance of the model. If the model is trained with images before they are perspective transformed, the model might fail while inferencing images in angles different than that of the trained images. A performance comparison of all the trained models can be found in section V. A comparison of images before and after they are perspective transformed can be found in Fig. 4. Out of the created dataset 75% of images were used for training the network model whereas 25% of the images were used for validation.

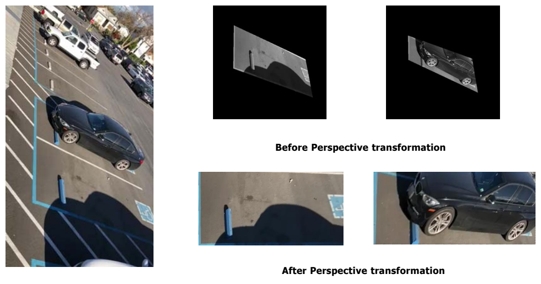
Before Perspective transformation
After Perspective transformation

Figure 4. View of parking slots before and after perspective transformation

2) Dataset for object detection and tracking: The dataset used for object tracking project was provided by Nvidia in the AI city challenge 2017 [15]. It consisted of raw images, without any annotations. So, the first step is to label the objects in the images and prepare the dataset. We utilize the annotation tool developed and provided by Nvidia. After completing the annotation, 150,000 labeled images extracted from eighty hours of traffic video was available. These labeled images were then divided into training and validation datasets. Next, the labels were converted into formats compatible with the models that have been selected. The YOLO model requires the images to be in Darknet format. Finally after converting the images into Darknet format, 59,482 training images and 19,272 validation images of size 1920 x 1080 pixels were available for training.

B. Efficient Neural Network Model Design for Parking Lot Occupancy Detection

Many existing vision-based vehicle and parking spot detection services use modern machine learning techniques, such as deep learning. However, the state-of-the-art results by deep learning come at the price of an intensive use of computing resources. For example, the leading deep learning frameworks (e.g., TensorFlow) are expected to run on top of high-end servers in datacenters. On the other end, there is a proliferation of personal devices with possibly free CPU cycles; this can enable services to run in users' homes, embedding machine learning operations.

Keeping in mind that our processing has to be performed on the edge and should not require a GPU with high computational capability, we have designed our custom network to be very lite. Our model is based on the AlexNet neural network. AlexNet has 5 convolution layers, 7 ReLu layers, 3 max pooling layers, 3 fully connected layers [16] and though it is very accurate, it is slow for real time computation. AlexNet was originally created for the ImageNet dataset which consisted of 22,000 categories [16]. But since we require only two classes (occupied and empty) for identifying vacant spots in a parking lot, such a deep convolution neural network is not necessary. A neural network with a single convolution layer will be sufficient for this application. This will also help in reducing the time taken for the real time inferencing. The network model created for this application has 1 input layer, 1 convolution layer, 1 ReLu, 1 max pooling and 3 fully connected layer. The construction of the custom network model can be observed in Fig. 5.

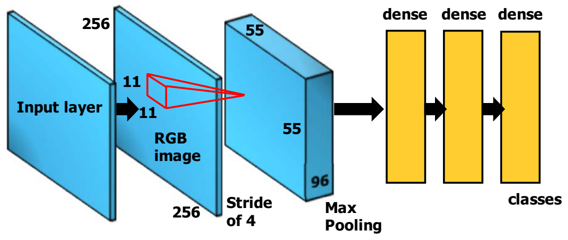

Figure 5. Our Custom Designed Network Model

Our custom-designed model consists of one convolution layer which performs the convolution operation over the 256x256 RGB input images.The weights in the convolution layers are trained and updated in each learning iteration using a back-propagation algorithm. It has a stride of 4 with receptive field or kernel of 11x11, giving the output a dimension of 55x55 as calculated using equation (1)

$$W_{out}(i) = 1 + (\frac{(W_{in}(i) - R + 2P)}{S}) \tag{1}$$

Where, R - Receptive field or Kernel, P - Padding, S - Stride. The convolutional layer is followed by the ReLu layer with sigmoid activation function to introduce non linearity in the model as shown in (2)

$$f'(x) = \frac{d}{dx} \ln(1 + e^x) = \frac{1}{1 + e^x} \tag{2}$$

ReLu layer is followed by a Max Pool layer which performs down-sampling. The reduction of size leads to loss of information. However, such a loss is beneficial for the network for two reasons: 1) The decrease in size leads to less computational overhead for the upcoming layers of the network and it works against over-fitting; 2) At the end we have a fully connected layer which connects the output of previous layer and gives a 2 class output.

Our designed model has been trained using a custom dataset as well as the CNRpark and PKLot datasets [10] in DIGITS (Deep Learning GPU Training System by Nvidia).

C. YOLO model based on darknet framework

YOLO [17] is an object detection model that uses regression. Every input image to the model is split into grids and each cell in a grid predicts some bounding boxes and gives a confidence value which indicates how sure the model is that the box contains an object. After this step the model knows where the object is in that image but it does not know what that object is. For knowing the class, each cell predicts a class probability using the pre-trained weights. Finally the box and class predictions are combined to identify the object.

D. License plate number recognition and vehicle tracking

OpenALPR is used for Optical Character Recognition. It uses Tesseract OCR library. For the application at hand, OpenALPR bindings have been used along with python to recognize the characters in an image. OpenALPR can recognize license plate at any position within the image and Raspberry Pi is completely responsible for running OpenALPR along with its python bindings.

The vehicles have to be tracked until they reach and settle in a spot in order to identify the slot number each vehicle has occupied. By doing this the location of a particular vehicle can be shared with its owner in case he/she is unable to locate the vehicle. This information will be secured and information pertaining to each vehicle will be shared only after entering a pass code which is created beforehand. This feature is yet to be implemented.

E. The LIDAR based system

As menioned in Section II, for indoor parking lots, the Raspberry Pi interfaced with a camera and a LIDAR has been proposed. The LIDAR will be rotated in equal steps using a stepper motor and the distance reading will help in identifying if the slot is occupied or not. If the slot is occupied, the camera will start capturing the license plate number to make a note of that particular car's position in the parking lot. This information will then be sent to the web application. Fig. 6 shows the hardware setup containing the Raspberry Pi, LIDAR, stepper motor and camera. The slot number will be identified by the the angular position of the stepper motor.

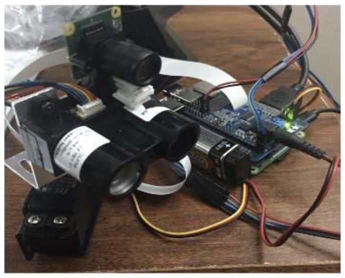

Figure 6. Hardware setup for indoor parking lots

IV. Results

A. Results of our custom model used for object classification

The trained models have been evaluated by using them to inference videos which have been captured from different parking lots in different angles and conditions. The efficiency of a model can be measured in terms of its accuracy and the time taken to classify a single image. The accuracy of a model is given by the formula

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn} \tag{3}$$

Where, tp - true positives, tn - true negatives, fp - false positives and fn - false negatives. The three models trained (standard AlexNet, AlexNet reduced to two convolution layers and AlexNet reduced to one convolution layer) have different accuracy and different inference time. The frames extracted from a video and the output after inferencing the entire frame are shown in the Fig. 7 and Fig. 8. Each figure shows various parking lots with different view angles.

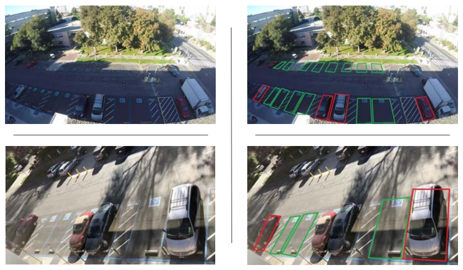

Figure 7. Input Image To The Model and The Output Image Indicating the Vacant and Occupied Parking Slots at SJSU 20Min Parking lot .

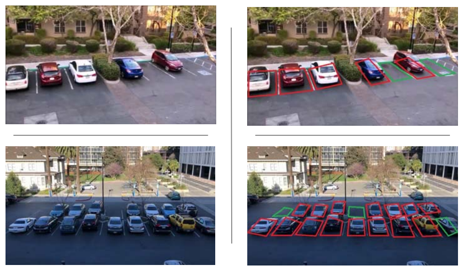

Figure 8. Slot Occupancy detection results at Avalon on the Alameda(above) and SJSU south garage(below) Parking lots

The total number of images for each class in the dataset (CNRPark + PKLot + own data set) and total number of images are given in Table I.

Dataset	Empty	Occupied
TraindB	49632	1266
valdB	12,400	949

Table I The total number of images, TraindB and ValdB in the dataset.

Table II, III and IV represent the confusion matrices for the three network models considering empty to be positive and occupied to be negative.

	Empty	Occupied
Empty	12400(tp)	0(fn)
Occupied	5(fp)	944(tn)

Table II The confusion matrix for standard AlaxNet

	Empty	Occupied
Empty	12400(tp)	0(fn)
Occupied	40(fp)	909(tn)

Table III The confusion matrix for AlaxNet with two convolutional layers

	Empty	Occupied
Empty	12400(tp)	0(fn)
Occupied	36(fp)	913(tn)

Table IV The confusion matrix for the custom designed network model

A comparison of the accuracy and time taken for inference for the standard AlexNet, mAlexNet and the custom designed network model are given in Table V and Table VI.

Network model	Accuracy in percentage
Standard AlexNet	99.8584
AlexNet with two convolution layers	99.70
Custom designed network model(One convolution layer)	99.51

Table V Comparing accuracy of standard AlexNet, mAlexNet and the custom designed network model on 25% of CNRPark, PKLot and our own data set combined

The miniAlexNet network model proposed in [10] in the best case has a maximum accuracy of 98.125%. Hence our proposed network model has a higher accuracy.

Network model	Time taken for inference
Standard AlexNet	19.98ms
AlexNet with Two convolution layers	13.54ms
Custom designed network model(One convolution layer)	7.11ms

Table VI Comparing inference times of standard AlexNet, mAlexNet and the custom designed network model on 25% of CNRPark, PKLot and our own data set combined.

The miniAlexNet proposed in [10] takes 15 seconds to classify 50 slots. Our custom designed model takes $50 \times 7.11ms = 355.5ms$ to classify 50 slots.

B. Results of our YOLO model used for object detection

Our trained YOLO model has a mean average precision (mAP) of 0.27. A graph containing Average precision and F1-score for the Nvidia dataset is shown in Fig. 9. Also, Fig. 10 shows the Bounding boxes drawn as a result of object detection using Yolo model.

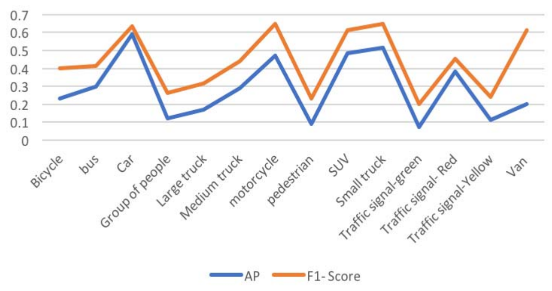

Figure 9. Average Precision and F1-score for the YOLO model on the Nvidia dataset.

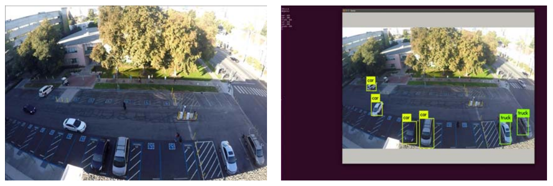

Figure 10. Object detection using YOLO model on SJSU Parking lot.

To test the car parking occupancy using LIDAR, we place the LIDAR in fixed distances from the slots. The expected distance that would be measured by the LIDAR for each slot was calculated. By observing the average distance between two vehicles parked in the lot, the angle the LIDAR must be rotated in order to face the next slot has been calculated. The performance of this system is evaluated by calculating the number of cars that were correctly classified (i.e. the distance measured by the LIDAR is equal to or less than the expected distance when a slot is occupied).

Fig. 11 shows a graph of the expected distance value from the LIDAR when each slot is occupied vs the actual reading obtained from the LIDAR. Fig. 12 shows the accuracy of the LIDAR system.

OpenALPR is evaluated by its accuracy, i.e the number of licence plate number predictions which were correct in the top 10 predictions and the top 1 prediction [18]. Table VII consists of the benchmarks for OpenALPR [18]. Easy set indicates a collection of license plates with a non-obstructive

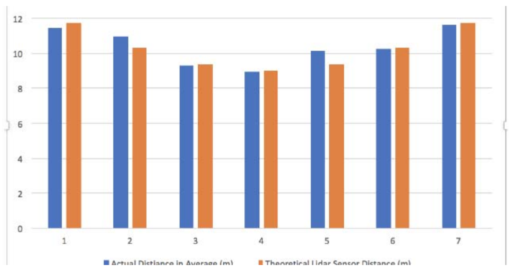

Figure 11. A graph showing the expected LIDAR reading (blue bars) and the actual LIDAR readings (orange bars) when each slot is occupied.

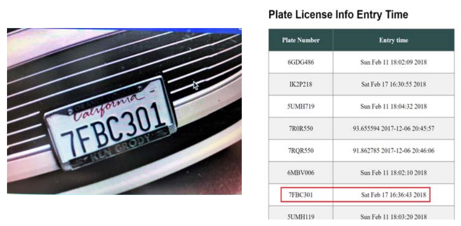
Plate License Info Entry Time

Plate Number	Entry time
6GDG486	Sun Feb 11 18:02:09 2018
IK2P218	Sat Feb 17 16:30:55 2018
5UMH719	Sun Feb 11 18:04:32 2018
7R0R550	93.655594 2017-12-06 20:45:57
7RQR550	91.862785 2017-12-06 20:46:06
6MBV006	Sun Feb 11 18:02:10 2018
7FBC301	Sat Feb 17 16:36:43 2018
5UMH119	Sun Feb 11 18:03:20 2018

Figure 13. License plate number recognised and pushed to the web server

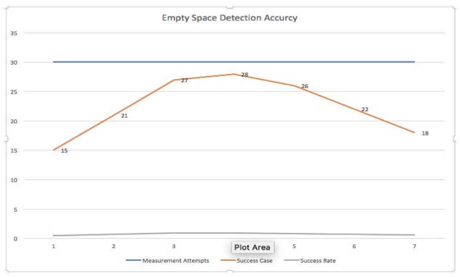

Figure 12. A graph showing the accuracy trend for each spot. blue - number of attempts, orange - correct classifications

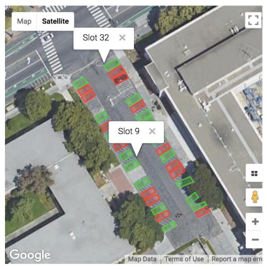

Figure 14. Vacant and occupied slots updated in the web application integrated with Google maps

view and for which the camera angle is good. The hard set consists of number plates in unexpected conditions, resolutions and for which the camera view is not optimal.

Benchmark	State Recognition	Top 10 (percent)	Top 1 (percent)
Easy Set	0	71.19	49.15
Hard Set	0	32.56	23.26

Table VII OpenALPR Benchmarks

Fig. 13 shows the license plate number of a car being recognized with the help of OpenALPR and the result being pushed to the web server with time stamp.

The web application shows the occupancy information for each slot being updated in Google maps and this information is also displayed in a separate table. This can be seen in Fig. 14.

The Jetson Tx2 processes live videos that are being transmitted to it from a raspberry pi. As of now there is a transmission delay of about 10 seconds and since this project is still in progress, efforts are being put into reducing this delay.

V. Related Works

The state-of-the-art solutions available for smart parking lots have been studied before deciding on our proposed system. The products currently available in the market are based on sensors and the working of these systems is described in [9], [3] and [5].

Data is the most important aspect for deep learning and artificial intelligence. Research has been performed towards finding out the requirements for obtaining or generating a good data set. According to [13], to make the data set as generic as possible, data must be collected in every conceivable condition and situation. Though the data set provided in [10] and [13] is generic, it can still be improved. Some projects for developing a similar application have been implemented using standard techniques of computer vision instead of deep learning and artificial intelligence. These techniques have been discussed in [19] and gives an idea on the drawbacks of this technique. Next, various network models have been studied to decide on which model should be used for this application. Most of the network models as discussed in [16] and [20] are too large and do not perform well for real time object classification. Hence, it has been decided to create a model by modifying the existing models such that it will perform well for real time classification. In [10] a model has been developed for a similar application and gives a brief description on what has been done by them to modify a model and reduce its layers without effecting the accuracy.

VI. Conclusions and Future Work

We proposed a well rounded solution for solving parking problems. Our solution overcomes the disadvantages of currently existing non-vision based solutions and also adds more useful features to vision based techniques such as license plate detection and vehicle tracking. A custom network model has been developed for identifying vacant slots in parking lots. From the obtained results, we can observe that there is a huge reduction in the inference time for the custom network model (7.11ms) when compared to the standard AlexNet (19.98ms) and mAlexNet (13.54ms). Our reduction in inference time has been achieved with little reduction in accuracy. The 0.034 percent reduction in accuracy compared to the standard AlexNet is acceptable. Our model has a higher accuracy and much faster than the model presented in [10]. Our future work includes increasing the speed and accuracy of the object detection model and establishing a link between the ground cameras and top-view cameras to achieve object tracking.

Acknowledgment

The work presented in this paper is funded by Cisco Systems and National Science Foundation under Grant No. CNS 1637371.

References

[1] S. Kumar, "Smart city solutions   smart parking lots - internet of things — iot india," 2018. [Online]. Available: https://electronicsofthings.com/expert-opinion/smart-city-solutions-smart-parking-lots/

[2] 2018. [Online]. Available: https://www.usatoday.com/story/money/2017/07/12/parking-pain-causes-financial-and-personal-strain/467637001/

[3] R. Grodi, D. B. Rawat, and F. Rios-Gutierrez, "Smart parking: Parking occupancy monitoring and visualization system for smart cities," in *SoutheastCon 2016*, March 2016, pp. 1–5.

[4] J. K. Suhr and H. G. Jung, "Sensor fusion-based vacant parking slot detection and tracking," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 1, pp. 21–36, Feb 2014.

[5] H. Chaudhary, P. Bansal, and B. Valarmathi, "Advanced car parking system using arduino," in *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Jan 2017, pp. 1–5.

[6] K. W. Min and J. D. Choi, "Design and implementation of autonomous vehicle valet parking system," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, Oct 2013, pp. 2082–2087.

[7] 2018. [Online]. Available: https://spectrum.ieee.org/transportation/self-driving/the-big-problem-with-selfdriving-cars-is-people

[8] A. K. Nayak, H. C. Akash, and G. Prakash, "Robotic valet parking system," in *2013 Texas Instruments India Educators' Conference*, April 2013, pp. 311–315.

[9] S. Valipour, M. Siam, E. Stroulia, and M. Jagersand, "Parking-stall vacancy indicator system, based on deep convolutional neural networks," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, Dec 2016, pp. 655–660.

[10] G. Amato, F. Carrara, F. Falchi, C. Gennaro, C. Meghini, and C. Vairo, "Deep learning for decentralized parking lot occupancy detection," *Expert Systems with Applications*, vol. 72, no. Supplement C, pp. 327 – 334, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S095741741630598X

[11] S. Du, M. Ibrahim, M. Shehata, and W. Badawy, "Automatic license plate recognition (alpr): A state-of-the-art review," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 2, pp. 311–325, Feb 2013.

[12] S. Earley, "There is no ai without ia," *IT Professional*, vol. 18, no. 3, pp. 58–64, May 2016.

[13] P. R. de Almeida, L. S. Oliveira, A. S. Britto, E. J. Silva, and A. L. Koerich, "Pklot   a robust dataset for parking lot classification," *Expert Systems with Applications*, vol. 42, no. 11, pp. 4937 – 4949, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417415001086

[14] "LabelMe an open annotation tool," http://labelme.csail.mit.edu/Release3.0/, accessed:2018-02-19.

[15] 2018. [Online]. Available: http://smart-city-sjsu.net/AICityChallenge/

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'12. USA: Curran Associates Inc., 2012, pp. 1097–1105. [Online]. Available: http://dl.acm.org/citation.cfm?id=2999134.2999257

[17] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015. [Online]. Available: http://arxiv.org/abs/1506.02640

[18] "OpenALPR Benchmarks," http://www.openalpr.com/benchmarks.html, accessed:2018-02-20.

[19] K. Hammoudi, M. Melkemi, H. Benhabiles, F. Dornaika, S. Hamrioui, and J. Rodrigues, "Analyzing and managing the slot occupancy of car parking by exploiting vision-based urban surveillance networks," in *2017 International Conference on Selected Topics in Mobile and Wireless Networking (MoWNeT)*, May 2017, pp. 1–6.

[20] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014. [Online]. Available: http://arxiv.org/abs/1409.4842