Using Edge Devices and Machine Learning for Controlled Access to a Smart Campus

Rajeev Ratan
BPP University
London, United Kingdom
mail@rajeevratan.com

Stephen Ghool
The University of the West Indies
St. Augustine, Trinidad
stephenghool@gmail.com

Reshawn Ramjattan
University of Pisa
Pisa, Italy
reshawn.ramjattan@phd.unipi.it

Shiva Ramoudith
The University of the West Indies
St. Augustine, Trinidad
shiva.ramoudith@gmail.com

Patrick Hosein
The University of the West Indies
St. Augustine, Trinidad
patrick.hosein@sta.uwi.edu

***Abstract*—University campuses can employ Fourth Industrial Revolution technologies to improve traffic flow and increase security. We provide one such solution that uses inexpensive edge devices (in our case, the Jetson Nano device from NVIDIA) to scan the license plates of cars entering a closed campus and to automatically determine whether or not entry should be allowed. Our solution eliminates the need for a guard to perform this check and speeds up the verification process, ultimately reducing the entry delay of incoming cars. We can also use the data captured from these devices to provide a wide range of services to the campus population such as, reducing traffic congestion, restricting access to parking lots and reducing car theft. These services will be inexpensive, thus requiring no additional fees to students and staff. We developed a custom license plate dataset for this study since our license plates are different to those in public datasets. In the proposed solution, the captured image is processed on the edge device but queries information from a centralized database to make a decision. A copy of this database is cached on the device but that copy is only used if communication to the central database is lost. We trialled the system under various conditions and present a subset of our results. Once sufficient data is collected we also plan to use AI tools to provide further enhancements.**

***Index Terms*—Smart Campus, Machine Learning, Jetson Nano, Deep Learning, Access Control**

I. Introduction

We propose a low cost portable device for capturing license plate numbers of cars entering a university campus, exiting the campus and entering/exiting various campus parking lots. This is done using a Jetson Nano Development Kit which captures the information and makes a decision using a central database. Access control rules in the database can be changed at any time (via the Internet) and can be dependent on factors such as time of day, day of the week, holidays, etc. This will be done in real time and will be used for access control (i.e., based on the response a barrier may or may not be opened). In some cases the driver is provided a notice on a LED screen and lasers are used to determine violation. In addition to access control there are several other uses for this data.

Campus Access Control: Presently campus staff must apply for a decal that is used to enter the campus. This approach of course has flaws and requires security personnel to continuously check for such decals on incoming cars. We instead propose that staff members use a web portal with authentication to indicate the plate number(s) of the car(s) they use to come to campus. When such plate numbers are detected the barrier is automatically opened otherwise a message on a LED display informs the driver that they are not allowed to enter. Allowances will be made for visitors and for temporary entry to drop off students. The time on campus of such entries will be timed (since exiting cars are monitored) and future access to violators will be blocked.

Parking Lot Control: Note that the same device can be used to allow cars to enter a parking lot. Furthermore, when the lot becomes full (this can be detected by monitoring exiting cars as well) drivers can be informed. The message can also include an alternative parking lot. Note that cars that enter illegally (e.g., by following a car on entry) can be detected and appropriate action taken. In fact parking lot occupancy rates can be computed and provided via an app.

Car Theft Control: One can provide various services based on the information collected. For example, a simple mobile app can be developed to do the following. When a car exits, a message is sent to a central server that checks the location of the car owner (if given permission) and prevents the exiting of the car if the person is not located in the car. For less intrusion, one can instead receive a text message when a car exits at an unusual (for that individual) time of day. There are many such combinations of events that may be used to determine a stolen car using various anomaly detection techniques.

Traffic and Speed Control: This automated process will increase car flow rates and also allows for the monitoring of traffic flow at different times of the day. This information can be used to better manage traffic during peak hours (e.g.,

978-1-6654-8771-9/22/$31.00 ©2022 IEEE
DOI: 10.1109/cicn.2022.90

allowing two traffic lanes during such periods). One can compute the time that should be taken to drive from one monitoring point to the next and if this is less than acceptable (based on speed limits) then appropriate action can be taken.

II. Related Work and Contributions

Automatic license plate recognition (ALPR) has been investigated over the last decade as intelligent transportation and smart cities are on the rise. Coupled with the fact that GPUs support deep learning computations, this has made ALPR an interesting field [1]. ALPRs are commonly used for security, traffic control and law enforcement [2]. Air pollution due to traffic congestion in urban areas have contributed to 2200 premature annual deaths [3]. [4] investigated the environmental health impacts of exposure to vehicular emissions. They concluded that, although a decrease in emission was observed in 2020, there is an expected increase in 2030 due to increasing population.

[5] has provided a case study outlining the various techniques that can be used to develop a license plate recognition system. The techniques are categorized under different subsystems such as pre-processing, filtering, feature extraction, segmentation and recognition to translate images into meaningful data. However, for this project, we aim to combine the pre-processing, filtering, segmentation and recognition systems and utilize a single neural network to extract the license plate characters. Broadly speaking, ALPR comprises: image collection, object detection, segmentation and optical character recognition [1].

[6] utilized neural networks to develop a smart license plate recognition system. They used iterative thresholding to identify the license plate in the input image. A neural network was then used to extract the characters from the processed input image. The network consisted of 366 input neurons, one hidden layer with 50 neurons and an output layer with 46 neurons. The system was able to classify characters with 95% accuracy at 50% noise density.

[2] developed an automatic license plate recognition system using the YOLO (You Only Look Once) object detector. The system comprised of two main subsystems, vehicle recognition and license plate coupled character recognition. The system was tested on high-end GPUs that process 50 FPS (Frames per second) and achieved an end-to-end accuracy of 96.9%. [7] explored the use of the YOLO deep learning model to perform plate detection in bad environmental conditions such as weather and lighting. The model was modified with a performance of 54 FPS on high-end GPUs.

[8] utilized two CNNs and an RNN to develop a car license plate detection and recognition system. The CNNs were responsible for detecting the license plates in the image. They first used character-based license plate detection which employed a sliding window across the image to detect the presence of text. They then used a 4-layer network, to separate the license plates from general text. However, this system takes approximately 2 seconds to process a single image when using the Tesla K40c GPU. [9] used the Nvidia Jetson TX2 module with the YOLOV4 model to perform automatic license plate detection. They used low quality images to perform license plate recognition from still and moving vehicles.

[1] proposed sliding-window single class detection as opposed to using many class predictors. This does result in additional complexity but computation time was mitigated by the decision to reduce the number of layers in the deep learning model. The dataset used contained 2049 images on Taiwanese license plates with variation in location, weather, time and traffic density. Overall, the system achieves 98.22% accuracy in license plate detection and 78% accuracy in license plate recognition. No character segmentation nor noise reduction was applied.

Previous research into the study of ALPR systems used high-end GPUs for testing and deployment of their models. However, it is important that research is also being done on low-cost edge-based systems that use low-end GPUs to perform ALPR. We are aware that there are a range of existing services available such as OPENALPR [10] and Plate Recognizer [11] but these did not suit our specific needs.

Our contribution is a platform that serves our unique circumstances. Although systems such as OPENALPR and Plate Recognizer supports some of our requirements, neither satisfies all and hence our design.

1) We have multiple campuses and so we plan to install about 50 edge devices. These will all have WiFi or Cellular access and in some cases will be solar powered. Therefore the cost of devices, cost of installation and operational costs are important and this ruled out the other options.
2) Our license plates are different to those in Europe and the United States. PlateRecognizer does support Trinidadian license plates however, OPENALPR does not. Furthermore we can take advantage of the specific numbering scheme used in our country to improve accuracy.
3) We require a system that we can continuously update with new functions and hence full access to the code.

III. Dataset

A few datasets were considered (OpenAPLR, SSIG-SegPlate [12] and UFPR-ALPR [13]) but none contained license plates found locally and they did not consider rain and other conditions as experienced locally. Therefore we decided to collect our own data for the project. The CSI camera was mounted to the Jetson Nano to be able to capture images of the vehicles. The dataset was collected using a variety of cameras including the CSI camera, mobile phone cameras and CCTV cameras positioned facing roadways.

In order to capture the diversity seen in real-world conditions, collected images encompassed a wide variety of lighting conditions, angles, and vehicle types. Furthermore, images were also obtained (using a Python Script) by scraping footage

from a local, public, authority-approved traffic cam live stream website. Images of vehicles that were clearly visible were manually extracted from the footage and labelled. The following figures (1 and 2) show the wide assortment of images captured for the dataset. The dataset contained 4897 images of various cars and with 33 text characters for OCR. The character frequencies are shown in Figure 3.

IV. Model Description

The system used the incremental software development life cycle model. It was segmented into three main subsystems, image pre-processing, license plate detection, and OCR recognition. This life cycle was used as it is easy to test, debug, maintain and manage risks [14]. Deep Learning was used within the latter two subsystems to accomplish the respective tasks. Three important components were used to generate the output, the CSI camera, the Jetson Nano and a cloud server.

In the first step the YOLOv5s object detection model was trained to detect a singular class namely license plates. The lightweight YOLOv5s model (7.2M parameters) was used as the object detection model for detecting license plates from images. Transfer learning was applied using a YOLOv5 model that was pre-trained on the COCO dataset [15] and then trained on the collected license plate dataset. Transfer learning was used as it is more efficient than manual development since it usually requires a smaller dataset and less training time [16]. YOLOv5 was selected as the object detection model due to the availability of lightweight models (YOLOv5s) and a high MAPE score of 37.4 on the COCO val2017 dataset.

YOLO is a single shot (one stage) object detector model. It uses a convolutional neural network object detection system that has been optimized for real-time processing. YOLO works by dividing the input images into an $N \times N$ grid (typically 7 x 7) where each grid cell predicts only a single object. For each grid cell, it predicts a number of boundary boxes.

The YOLO architecture consists of 27 CNN layers, with 24 convolutional layers, two fully connected layers, and a final detection layer. The model outputs a vector containing the predicted bounding boxes and the associated confidence score. The confidence score indicates whether each boundary box contains an object and the accuracy of the bounding box. YOLO predicts a final (7, 7, 30) tensor which stores the relevant object detection information (box locations, classes and confidence levels). The final layer in the YOLO model is used to retain boundary boxes with a high confidence score.

YOLO uses three loss calculation functions in training, (1) classification loss, (2) localization loss and (3) confidence loss. It uses the sum-squared error to calculate error loss between the predictions and the real objects. Classification loss is the squared error of the class probability. Localization loss measures the errors in the boundary box locations and sizes. Confidence loss is the error in the measurement of the likelihood of objects in the box. The final loss of YOLO is the sum of the three loss functions.

In the second step, image pre-processing, simple and computationally inexpensive steps are applied to the detected license plate (cropped out of the original image) to enhance the image by applying Gaussian blur to reduce noise and an auto contrast function to ensure license plate text was more distinct.

The third and final step is the OCR (Optical Character Recognition) module. OCR algorithms typically involve the detection of text and recognition of each character of text. Most approaches for OCR on license plates rely on character segmentation methods. These methods either involve using image processing techniques (edge detection, contour detection etc.) to localize individual characters, or an object detector trained to detect smaller-scale objects such as individual characters of text [17]. These methods can work well in controlled environments where characters are visibly separated and distinct. However, in the real world, due to skewed angle viewpoints, poor lighting conditions, and camera resolution limitations, characters often bleed into each other resulting in poor performance by traditional character segmentation methods (e.g., see 4)

The OCR model was developed using both Convolutional Neural Networks (CNN) and Recurrent Neural Network (RNN) layers on a hybrid structure which was influenced by [18] and [19]. [19] indicated that the CNN layers perform the feature extraction and the RNN adds sequence modelling which leads to a segmentation free OCR system. Connectionist Temporal Classification (CTC) loss was used in training this hybrid network. The network consists of 3 2D convolutional layers, 3 LSTM layers and an output layer.

V. Experimental Results

The system consisted of two subsystems, license plate detection and OCR recognition. As previously described, primary data collection was used to create a dataset to be able to use supervised learning for the deep learning models. Neural networks, such as VGG16, are susceptible to image quality distortions [20]. In this system, the distance between the incoming vehicle and the camera may vary the quality of the captured image. Therefore, the location of the system needs to be optimized for maximum performance. The plate detection model's performance was observed for various horizontal distances, Figure 5 and vertical distances, Figure 6. A decreasing trend was observed for increases in the horizontal distance whilst an increasing trend was observed for increases in vertical distance with a drop off after 5ft. There are two possible reasons for these trends, firstly, changing the distances changes the image quality and secondly, the training data was skewed, that is, it captured vehicles at a particular distance (approximately where the prediction accuracy is maximized). The sudden decline in Figure 6 was due to the loss of license plate line of sight.

Variations in lighting conditions due to weather or lack of artificial lighting could affect the quality of the captured image [21]. As such, the system's performance was observed

Fig. 1. Examples of Angled viewpoint, Dim Lighting and Low Resolution Images

Fig. 2. Examples of Poor Lighting, Non-Standard Location and Rain Obstruction Images

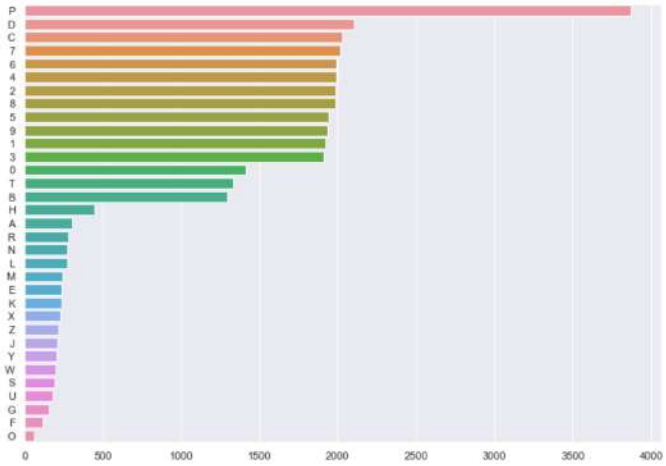

Fig. 3. OCR Character Class Count

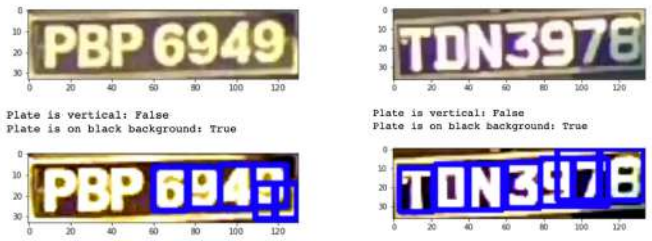

Fig. 4. Scenarios where classical character segmentation techniques fail

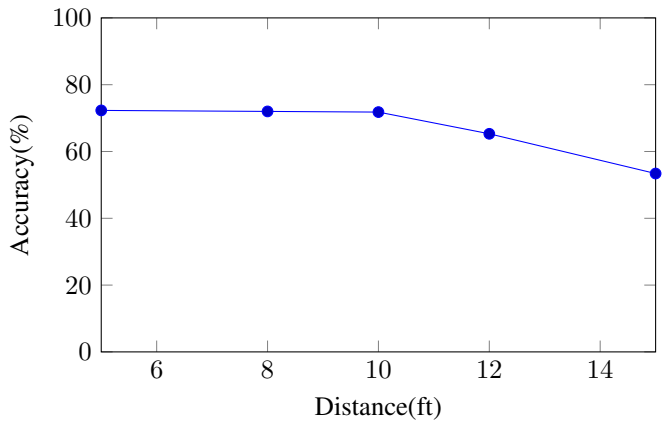

Fig. 5. Plate Detection Accuracy Vs Horizontal Distance

in different lighting conditions, Figure 7. Increasing the light intensity positively affected the system's performance as one would expect. In the case of light intensity 1 (no light) the accuracy drops to zero.

The execution times for both the license plate recognition and the OCR subsystems were observed for changes in vertical distance. Increasing the vertical distance led to a slight decrease in the total execution time (corresponding to increased accuracy performance) with the plate detector consistently requiring more time than the OCR, Figure 8. Therefore run-time is not significantly affected by camera placement.

VI. Conclusions and Future Work

Our custom ALPR system has the advantage of being portable and cost-effective allowing deployment at multiple locations. We intend to perform further optimizations with the

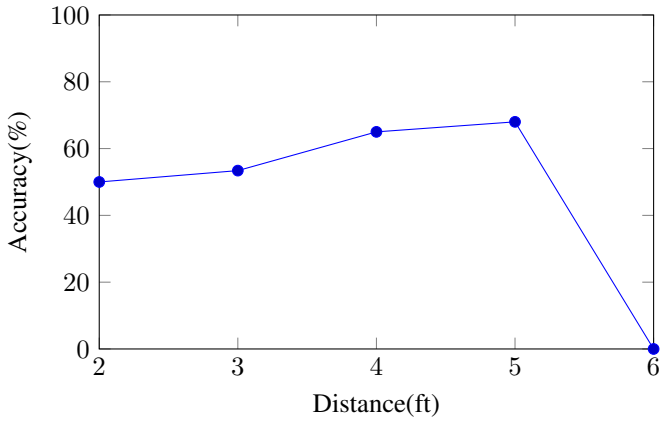

Fig. 6. Plate Detection Accuracy Vs Vertical Distance

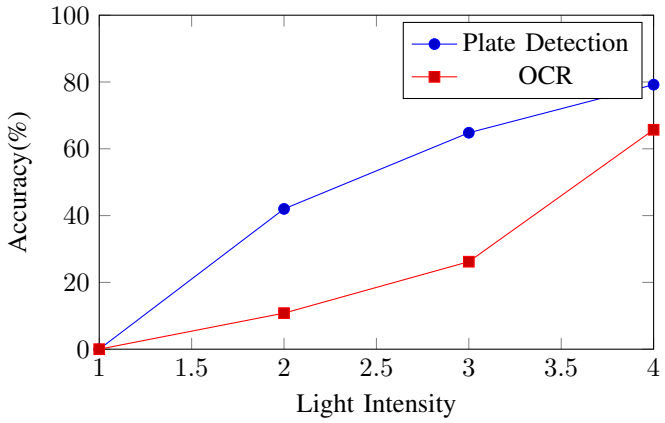

Fig. 7. Accuracy Vs Light Intensity

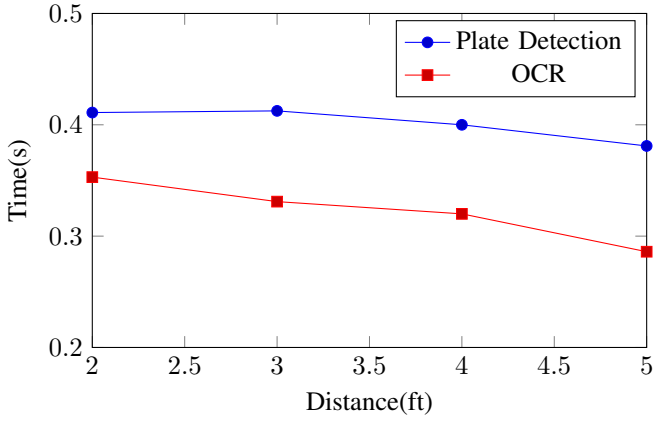

Fig. 8. Time Taken for Execution Vs Vertical Distance

Jetson Nano and employ active cooling. If the Jetson Nano is not suitable, we would consider more powerful edge computing devices such as the Jetson Xavier NX. However, the Xavier NX costs 50% more which will limit the number of deployments. We also plan to install 5 of these devices so that data can be collected for further improvements. In addition, algorithms for anomaly detection will be developed.

REFERENCES

[1] R.-C. Chen *et al.*, "Automatic license plate recognition via sliding-window darknet-yolo deep learning," *Image and Vision Computing*, vol. 87, pp. 47–56, 2019.

[2] R. Laroca, L. A. Zanlorensi, G. R. Gonçalves, E. Todt, W. R. Schwartz, and D. Menotti, "An efficient and layout-independent automatic license plate recognition system based on the yolo detector," *IET Intelligent Transport Systems*, vol. 15, no. 4, pp. 483–503, 2021.

[3] J. I. Levy, J. J. Buonocore, and K. Von Stackelberg, "Evaluation of the public health impacts of traffic congestion: a health risk assessment," *Environmental health*, vol. 9, no. 1, pp. 1–12, 2010.

[4] ——, "Evaluation of the public health impacts of traffic congestion: a health risk assessment," *Environmental health*, vol. 9, no. 1, pp. 1–12, 2010.

[5] N. K. Ibrahim, E. Kasmuri, N. A. Jalil, M. A. Norasikin, S. Salam, and M. R. M. Nawawi, "License plate recognition (lpr): a review with experiments for malaysia case study," *arXiv preprint arXiv:1401.5559*, 2014.

[6] V. Koval, V. Turchenko, V. Kochan, A. Sachenko, and G. Markowsky, "Smart license plate recognition system based on image processing using neural network," in *Second IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 2003. Proceedings.* IEEE, 2003, pp. 123–127.

[7] G.-S. Hsu, A. Ambikapathi, S.-L. Chung, and C.-P. Su, "Robust license plate detection in the wild," in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS).* IEEE, 2017, pp. 1–6.

[8] H. Li and C. Shen, "Reading car license plates using deep convolutional neural networks and lstms," *arXiv preprint arXiv:1601.05610*, 2016.

[9] J.-Y. Sung and S.-B. Yu, "Real-time automatic license plate recognition system using yolov4," in *2020 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia).* IEEE, 2020, pp. 1–3.

[10] "OpenALPR - Automatic License Plate Recognition — openalpr.com," www.openalpr.com, [Accessed 12-Sep-2022].

[11] "Automatic license plate recognition - high accuracy alpr," Jan 2022. [Online]. Available: https://platerecognizer.com/

[12] G. R. Gonçalves, S. P. G. da Silva, D. Menotti, and W. R. Schwartz, "Benchmark for license plate character segmentation," *Journal of Electronic Imaging*, vol. 25, no. 5, p. 053034, 2016.

[13] R. Laroca, E. Severo, L. A. Zanlorensi, L. S. Oliveira, G. R. Gonçalves, W. R. Schwartz, and D. Menotti, "A robust real-time automatic license plate recognition based on the yolo detector," in *2018 international joint conference on neural networks (ijcnn).* IEEE, 2018, pp. 1–10.

[14] I. Sommerville, *Engineering software products*. Pearson London, 2020.

[15] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[16] C. C. Aggarwal *et al.*, "Neural networks and deep learning," *Springer*, vol. 10, pp. 978–3, 2018.

[17] Hendry and R.-C. Chen, "Automatic license plate recognition via sliding-window darknet-yolo deep learning," *Image and Vision Computing*, vol. 87, pp. 47–56, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0262885619300575

[18] S. Y. Yerima, M. K. Alzaylaee, and A. Shajan, "Deep learning techniques for android botnet detection," *Electronics*, vol. 10, no. 4, p. 519, 2021.

[19] S. Rawls, H. Cao, S. Kumar, and P. Natarajan, "Combining convolutional neural networks and lstms for segmentation-free ocr," in *2017 14th IAPR international conference on document analysis and recognition (ICDAR)*, vol. 1. IEEE, 2017, pp. 155–160.

[20] S. Dodge and L. Karam, "Understanding how image quality affects deep neural networks," 2016. [Online]. Available: https://arxiv.org/abs/1604.04004

[21] C. Cuhadar, G. P. S. Lau, and H. N. Tsao, "A computer vision sensor for efficient object detection under varying lighting conditions," *Advanced Intelligent Systems*, vol. 3, no. 9, p. 2100055, 2021.