

# Real-time Automated Detection and Recognition of Nigerian License Plates via Deep Learning Single Shot Detection and Optical Character Recognition

Kayode David Adedayo<sup>1</sup>, & Ayomide Oluwaseyi Agunloye<sup>1</sup>

<sup>1</sup>Department of Physics, Federal University of Technology, Akure, Ondo – State, Nigeria

Correspondence: Ayomide Oluwaseyi Agunloye, Department of Physics, Federal University of Technology, Akure, Ondo – State, Nigeria.

Received: July 8, 2021

Accepted: July 27, 2021

Online Published: August 24, 2021

doi:10.5539/cis.v14n4p11

URL: <https://doi.org/10.5539/cis.v14n4p11>

## Abstract

License plate detection and recognition are critical components of the development of a connected Intelligent transportation system but are underused in developing countries because of the associated costs. Existing license plate detection and recognition systems with high accuracy require the usage of Graphical Processing Units (GPU), which may be difficult to come by in developing nations. Single-stage detectors and commercial optical character recognition engines, on the other hand, are less computationally expensive and can achieve acceptable detection and recognition accuracy without the use of a GPU. In this work, a pre-trained SSD model and a tesseract tessdata-fast traineddata were fine-tuned on a dataset of more than 2,000 images of vehicles with license plates. These models were combined with a unique image preprocessing algorithm for character segmentation and tested using a general-purpose personal computer on a new collection of 200 automobiles with license plate photos. On this testing set, the plate detection system achieved a detection accuracy of 99.5 % at an IOU threshold of 0.45 while the OCR engine successfully recognized all characters on 150 license plates, one character incorrectly on 24 license plates and two or more incorrect characters on 26 license plates. The detection procedure took an average of 80 milliseconds, while the character segmentation and identification stages took an average of 95 milliseconds, resulting in an average processing time of 175 milliseconds per image, or 6 photos per second. The obtained results are suitable for real-time traffic applications.

**Keywords:** real-time, license plate recognition, single-shot detector, computer vision, graphical processing unit, optical character recognition

## 1. Introduction

The detection and recognition of license plates are critical parts of traffic monitoring and are essential to the development of a connected Intelligent transportation system. Automatic vehicle License Plate Detection and Recognition (ALPR) engines have been extensively explored and have attained state-of-the-art results with the recent use of Graphical Processing Units (GPU) in deep learning computation applications (Hendry & Chen, 2019). However, these GPU systems are expensive and may not be widely available in developing nations. On the other hand, the recent increase in ICT availability in developing countries has made basic ICT tools such as mobile phones and general-purpose computers available in these areas (Avgerou et al., 2016). Thus, deploying ALPR systems on easily accessed general-purpose PCs will facilitate ALPR usage in developing countries.

The process of ALPR majorly consists of three important stages that are the license plate detection stage, character segmentation, and character recognition stages (Agbemenu, Andrew et al., 2018). All stages of ALPR are important. Nonetheless, a failure in the plate detection stage causes the entire ALPR process to fail. Deep learning-based plate detection systems, on the other hand, have the lowest false detection rates (Wang et al., 2021). However, deep learning-based license plate detection and recognition systems that achieve exceptional accuracy typically use two-stage deep learning techniques such as R-CNN and FR-CNN for ALPR, which are too heavy to operate in real-time on simple non-GPU devices (Zebin et al., 2019). On the other hand, one-stage detectors such as YOLO and SSD, which are faster and can work in real-time, provide systems that are fast and can operate in real-time but produce multiple errors due to their poor performance in identifying small objects such as plate number characters (E. Dong et al., 2018). Character segmentation and character recognition using

deep learning techniques have also performed excellently in literature (Shivakumara et al., 2018) but are as well computationally expensive. Optical character recognition engines are intended for large-scale text recognition with decent accuracy, and they are optimized to run quickly on both GPU and non-GPU platforms.

This study aims to develop a system for detecting and recognizing Nigerian license plates in real-time using non-GPU general-purpose computers. The above objective will be achieved by integrating a computationally efficient fine-tuned single shot detector object detection pipeline and optical character recognition. The system will also be able to provide additional information about the class of the license plate and the issuer's information.

The rest of this paper is structured as follows. Section 2 gives a literature review of the various ALPR stages. Section 3 explains the proposed system and its implementation. Section 4 contains the results and discussion of validating the developed system on a test dataset. Section 5 concludes the paper.

## 2. Literature Review

ALPR is a critical component of ITS security and traffic control systems that have received extensive attention from researchers. Images of stationary or moving vehicles with valid license plate instances are the primary input to ALPR systems. ALPR consists of three major steps, which are implemented in the literature using various processing methods. The first steps of ALPR are license plate detection and localization. Edge detection and feature-based detection algorithms are two extensively utilized license plate detection and localization techniques in literature. Edge detection (Agbemenu, Andrew et al., 2018; Dalarmelina et al., 2020; Ibiyemi et al., 2020; Oluchi et al., 2019) yields high inferencing speed but also produces several incorrect candidates since the system frequently detects undesirable edges (Hendry & Chen, 2019). Feature-based detection techniques make use of license plate-specific features such as color and texture (Deb & Jo, 2008; Wu et al., 2013), or a mix of features. Deep learning plate detectors that use several license plate features to localize them mostly employ machine learning models and techniques such as support vector machines (SVM) (Oluchi et al., 2019) and convolution neural networks (CNN) (M. Dong et al., 2017; Shivakumara et al., 2018). The capacity of deep learning-based detectors to use several features of a license plate in localizing it enhances plate detection accuracy and generalization at the expense of greater processing cost.

Character segmentation in ALPR entails the process of denoising the detected plate image and separating characters into individual units that can be properly recognized. Commonly used character segmentation techniques are; character erosion and dilation with edge and contour detection (Attah et al., 2016; Dalarmelina et al., 2020), character aspect ratio sampling (Agbemenu, Andrew et al., 2018; Ibiyemi et al., 2020), and the use of multiple sliding concentric window character region of Interest (ROI) pooling (Anagnostopoulos et al., 2006). CNN-based ALPR systems (Hendry & Chen, 2019; Shivakumara et al., 2018) often replace the character segmentation stage with a localization process carried out by the deep learning model. In this process, individual plate characters are localized feature-wise and then automatically segmented before they are passed up for recognition. Character segmentation is important for proper character recognition. If characters are not well segmented, the accuracy of the character recognition stage will be affected.

Character recognition (CR) is frequently the final stage in most ALPR systems. In the work of (Ibiyemi et al., 2020), CR was achieved using template matching. This method of CS has little capacity for generalization and is computationally expensive. In the studies of (M. Dong et al., 2017; Hendry & Chen, 2019), CR was achieved using multiple lightweight CNN classifier models. These models were trained to detect each character of the 35 license plate characters (A-Z and 0-9). They achieved great accuracy at the expense of speed. CR performed using OCR engines such as tesseract (RNN based LSTM engine maintained by Google) (Agbemenu, Andrew et al., 2018; Dalarmelina et al., 2020) are very fast but typically achieves full license plate recognition accuracies between 60 and 70%. These recognition accuracy values can be attributed to the character segmentation methods employed in the studies which do not completely conform to the black texts on with background requirements of the OCR engine (tesseract-ocr, 2021b).

In this study, we combined a light SSD mobilenetV2 CNN license plate object detector with a fast OCR engine to obtain higher plate detection and recognition accuracy at a real-time inferencing speed. We have also focused heavily on the character segmentation stage to supply our OCR engine with text blobs that meet the engine's requirements and so improve character recognition accuracy. The license plate of emphasis in this study is the Nigerian license plate system, which has been fully discussed in the work of (Ibiyemi et al., 2020).

## 3. Materials and Methods

In this section, we introduced the proposed real-time ALPR system and its constituents.

### 3.1 Proposed System

The proposed system is designed to acquire stationary images of vehicles from any input source. The images used in this study were acquired using an 8-megapixel smartphone camera with each picture having an average size of 3 MB under various lighting conditions. 700 images of vehicles with Nigerian license plates (both old and new) were captured and digitally augmented by rotation and skewing to provide a dataset of 2100 images to be used in training the plate detection and character recognition algorithms. 200 images were also captured and preserved for validating the ALPR system. To correctly detect and recognize a license plate, the system proceeds through the algorithm below;

1. *Obtain a high-quality image*
2. *Detect license plate via the object detection pipeline*
3. *If a license plate is detected;*
4. *Separate the RGB channels of the license plate image and compute their averages*
5. *Classify the license plate as private, commercial or government*
6. *Grayscale the image*
7. *Perform canny edge detection*
8. *If numbers of edges greater than the threshold;*
9. *Perform adaptive thresholding*
10. *End if*
11. *Else perform Otsu thresholding*
12. *End else*
13. *Compute plate aspect ratio and interpolate a character region*
14. *Crop out the character region strip and pass to the OCR engine*
15. *Perform OCR and get raw text*
16. *Compare the raw text to the required format based on its type*
17. *If a valid pattern is found;*
18. *Extract the local government or agency code provide the designated state or agency*
19. *Store the plate number character to a database*
20. *End if*
21. *End if*

The software implementation of the above algorithm is achieved using the python programming language.

### 3.2 License Plate Detection

In this stage, a high-quality image is downsampled and sent into a deep learning object detection pipeline. The pipeline returns box coordinates, which reflect the recognized license plate's bounding boxes. The pipeline extrapolates the supplied box coordinates to their original places on the high-quality input image. The license plate ROI is then retrieved from the specified bounding box coordinates and passed to the next stage of the ALPR. A MobileNetV2-based (Sandler et al., 2018) single shot detector pipeline from the TensorFlow Object Detection team was the model of choice in this study. This model was chosen for its flexibility, fast inferencing speed without the use of GPU, and the capacity for model optimization which reduces the size of its weights and biases and even further increases the inferencing speed. It has an input feature size of 320 by 320 by 3 and is maintained by (*Tf2\_detection\_zoo.Md*, 2021). Input images are first resized from their respective input image sizes to 320 by 320 pixels via the OpenCV resize function and then passed to the pipeline for plate detection. The accuracy of the plate detection pipeline on the validation set is calculated using Equation 1.

$$accuracy = \frac{\text{Total number of license plate detected}}{\text{Total number of license plates in dataset}} \quad (1)$$

### 3.3 Plate Number Segmentation

In this stage, an extracted plate ROI is morphologically processed using the OpenCV open-source image

processing library (Opencv-python, 2021). First, the extracted plate is separated into its constituent channels and the mean of each channel is computed. This is to enable the system to determine the type of plate number being processed from the color information (Attah et al., 2016). Next, the input image is grayscale and then Otsu or adaptively threshold based on the average number of edges present in the input image. Next, the image is further cropped to remove other irrelevant information at its top and bottom by comparing the plate's length and width to that of a standard Nigerian license plate. Finally, the extracted license plate strip is passed to the OCR engine for character recognition. Figure 1 to 5 shows the various morphological preprocessing operations in the segmentation step.



Figure 1. A sample high-quality Input image to be preprocessed by the pipeline



Figure 2. Extracted license plate after plate detection



Figure 3. Grayscale image of the extracted license plate

The image shows the text 'AGL 621FR' in a bold, black, monospace font. The characters are slightly irregular and have a hand-drawn appearance, typical of license plate characters. The background is white.

Figure 4. Plate strip after Otsu thresholding and character segmentation

The image shows the text 'KUJ 328TK' in a bold, black, monospace font. The characters are slightly irregular and have a hand-drawn appearance, typical of license plate characters. The background is white.

Figure 5. Plate strip after adaptive thresholding and character segmentation

### 3.4 Optical Character Recognition

The plate strip from the character segmentation stage is passed to the OCR engine for character detection and recognition in this stage. A simple plate postprocessing approach is applied to the raw recognized raw text. First, whitespaces and hyphens are removed from the raw text by this technique. Following that, the algorithm pattern matches the raw text based on the observed license plate class. This is because government plate numbers frequently contain mixed character agency or state distinguishing codes, whereas private and commercial license plates use exclusively alphabetic local government codes. Once a valid pattern is found in the raw text, the algorithm extracts this matched text and uses its LGA or agency code to look up its designated state or issuer information. Finally, the algorithm outputs the issuer and plate number of the provided license plate strip.

The tesseract engine is a general-purpose Long short-term memory OCR engine. It provides an avenue for using custom trained weights and activations bundled into a traineddata file type. Our custom traineddata was a finetuned base tesseract eng-fast.traineddata file (tesseract-ocr, 2021b). 700 plate images were extracted manually from our training dataset and were preprocessed into license plate strips samples of which are shown in Figure 6. These strips were then converted to the '.tiff' image format and then passed to the tesseract trainer 'box-to-text' script which localizes the various characters in the license plate image. Wrongly localized characters were manually corrected by adjusting the character information provided in the associated automatically generated '.txt' file. All images and their ground truth files are placed into a training file used to fine-tune the base trained data.

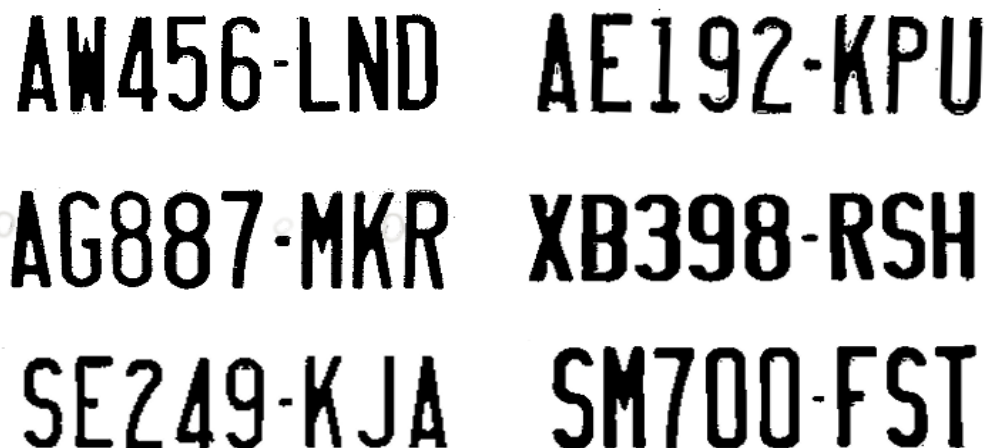
The image displays six license plate strips arranged in a 3x2 grid. The top row contains 'AW456-LND' and 'AE192-KPU'. The middle row contains 'AG887-MKR' and 'XB398-RSH'. The bottom row contains 'SE249-KJA' and 'SM700-FST'. All characters are in a bold, black, monospace font.

Figure 6. Sample old license plate strips for OCR engine training



#### 4. Results and Discussion

The plate detection pipeline was trained for 50,000 steps using the TensorFlow framework on the Google Collaboratory platform after which it obtained a final training loss of 0.095. The normalized training loss graph for the plate detection pipeline is shown in Figure 7. To improve its performance, the trained model was optimized to the TensorFlow Lite version. Instead of the standard command-line package, the OCR engine was implemented utilizing a python API package which increased the OCR engine's character recognition speed.

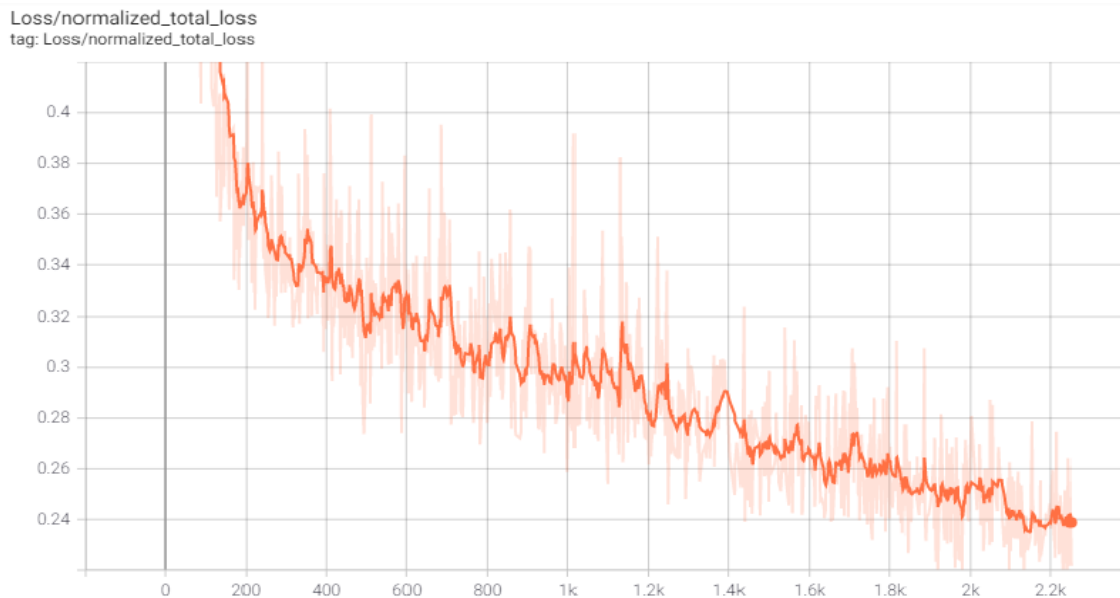


Figure 7. Plate detector model normalized training loss graph after 2,200 steps









All model implementations and algorithms were compiled into a single python script and validated in a single step with 200 vehicle and license plate images in the validation set on a general-purpose Personal computer without GPU. The object detection pipeline detected all the 199 license plates (99.5 % accuracy) in the validation dataset at an Intersect of Union (IOU) threshold of 0.4 and achieved an average processing time of 80 milliseconds. Figure 8 shows a snapshot of 6 different detections by the pipeline.



Figure 8. Snapshots of 6 different detections after object detection

On 150 images in the collection, the OCR engine accurately recognized all plate number information, including the designated state or agency code. On 24 license plate photos, the engine incorrectly recognized one character, while the engine incorrectly recognized two or more characters on 26 license plate images. This equates to a license plate recognition accuracy of 75% and a single character recognition accuracy of 92%. The OCR engine had a processing time of 90 milliseconds on average. In addition, the system properly categorized all license plates in the dataset. The system takes an average of 175 milliseconds to process a license plate. This is a reasonable value for real-time detection in low-speed traffic. Table 1 provides information on the result of running the developed system on 8 different plate strips from the validation set.

Table 1. OCR report of 8 different detected plates and their plates strips

S/N	Detected plate	Plate Strip	Ground truth	OCR result	Final Report
1.		KUJ 328TK	KUJ-328TK	KUJ328TK	ABUJA KUJ-328TK
2.		AKR 123LZ	AKR-123LZ	AKR123EZ	ONDO AKR-123EZ
3.		NND 827AE	NND-827AE	NN0827AE	NN0-827AE
4.		AKR 07AP	AKR-07AP	07APAKR	ONDO 07AP-AKR
5.		AAA 752FF	AAA-752FF	AAAJ32FF	LAGOS AAA-J32FF
6.		AGL 621FR	AGL-621FR	AGL621FR	LAGOS AGL-621FR
7.		WWR 919TD	WWR-919TD	WWR919TD	DELTA WWR-919TD
8.		WWW 234MW	WWW-234MW	WWW234MW	ONDO WWW-234MW

## 5. Conclusion

Using a lightweight deep learning SSD object detection pipeline, an improved character segmentation algorithm, and the tesseract OCR engine, this study has successfully developed a system that detected and recognized the plate numbers of Nigerian license plates in real-time and without the use of GPU on general-purpose PCs.

The model used in this study detected both old and new Nigerian license plates of various sizes in the validation dataset and thus, demonstrates the robustness of SSD detection pipelines in fast and accurate inferencing of object instances that are not too small compared to the image size, as well as their generalization potential. The system was able to categorized license plates as private, commercial, or government, and it also provides the plate's issuance information. This advanced function can be used at traffic gates for precise access control.

While the retrained OCR engine was able to differentiate identical character pairs such as 'A' as '4', 'G' and '6', and '2' and 'Z', it had difficulty distinguishing '0' and 'D' and 'B' and '8' on fading or scratched license plates. Using an end-to-end multiclass SSD detection pipeline for character identification may improve accuracy because CNN classifiers are resistant to character rotation, breakage, or skews. To sustain the achieved rapid inferencing, a general-purpose PC with a better CPU will be required.

## References

- Agbemenu, A. S., Yankey, J., & Addo, E. O. (2018). An Automatic Number Plate Recognition System using OpenCV and Tesseract OCR Engine. *International Journal of Computer Applications*, 180(43), 1-5. <https://doi.org/10.5120/ijca2018917150>
- Anagnostopoulos, C. N. E., Anagnostopoulos, I. E., Loumos, V., & Kayafas, E. (2006). A License Plate-Recognition Algorithm for Intelligent Transportation System Applications. *IEEE Transactions on Intelligent Transportation Systems*, 7(3), 377-392. <https://doi.org/10.1109/TITS.2006.880641>
- Attah, A. B., ADEDIGBA, A. P., & Aibinu, A. M. (2016). Development of Nigerian vehicles license plate recognition and classification system. *Journal of Science, Technology, Mathematics and Education*, 12(March), 125-133.
- Avgerou, C., Hayes, N., & La Rovere, R. L. (2016). Growth in ICT uptake in developing countries: New users, new uses, new challenges. *Journal of Information Technology*, 31(4), 329-333. <https://doi.org/10.1057/s41265-016-0022-6>
- Dalarmelina, V., Teixeira, M. A., & Meneguette, R. I. (2020). *A Real-Time Automatic Plate Recognition System Sensor Networks for ITS. 1*.
- Deb, K., & Jo, K. H. (2008). HSI color-based vehicle license plate detection. *2008 International Conference on Control, Automation, and Systems, ICCAS 2008*, 687-691. <https://doi.org/10.1109/ICCAS.2008.4694589>
- Dong, E., Zhu, Y., Ji, Y., & Du, S. (2018). An Improved Convolution Neural Network for Object Detection Using YOLOv2. *2018 IEEE International Conference on Mechatronics and Automation (ICMA)*, 1184-1188. <https://doi.org/10.1109/ICMA.2018.8484733>
- Dong, M., He, D., Luo, C., Liu, D., & Zeng, W. (2017). A CNN-based approach for automatic license plate recognition in the wild. *British Machine Vision Conference 2017, BMVC 2017*, 1-12. <https://doi.org/10.5244/c.31.175>
- Hendry, & Chen, R. C. (2019). Automatic License Plate Recognition via sliding-window darknet-YOLO deep learning. *Image and Vision Computing*, 87, 47-56. <https://doi.org/10.1016/j.imavis.2019.04.007>
- Ibiyemi, T. S., Owotogbe, J. S., & Adu, B. A. (2020). A Comparative Study of Vehicle Number Plate Recognition Systems. *African Journal of Management Information System*, 2(1), 10-23.
- Oluchi, I. B., Adedokun, E. A., Mua'zu, M. B., Salefu, O. N., & Oyibo, P. O. (2019). Development of a Nigeria Vehicle License Plate Detection System. *APPLICATIONS OF MODELLING AND SIMULATION*, 3(November), 3-11.
- Opencv-python. (2021). *Morphological Transformations*. OpenCV-Python Tutorials 1 Documentation. Retrieved from [https://opencv-python-tutroals.readthedocs.io/en/latest/py\\_tutorials/py\\_imgproc/py\\_morphological\\_ops/py\\_morphological\\_ops.html](https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html)
- Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 4510-4520. <https://doi.org/10.1109/CVPR.2018.00474>
- Shivakumara, P., Tang, D., Asadzadehkaljahi, M., Lu, T., Pal, U., & Anisi, M. H. (2018). CNN-RNN based method for license plate recognition. *CAAI Transactions on Intelligence Technology*, 3(3), 169-175. <https://doi.org/10.1049/trit.2018.1015>
- tesseract-ocr. (2021a). *tessdata\_fast: Fast integer versions of trained LSTM models*. Github. Retrieved from [https://github.com/tesseract-ocr/tessdata\\_fast](https://github.com/tesseract-ocr/tessdata_fast)
- tesseract-ocr. (2021b). *Tesseract Open Source OCR Engine (main repository)*. Github - Tesseract-Ocr. Retrieved from <https://github.com/tesseract-ocr/tesseract>
- tf2\_detection\_zoo.md. (2021). *GitHub - TensorFlow/Models*. Retrieved from [https://github.com/tensorflow/models/blob/master/research/object\\_detection/g3doc/tf2\\_detection\\_zoo.md](https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md)
- Wang, Y., Bian, Z.-P., Zhou, Y., & Chau, L.-P. (2021). Rethinking and Designing a High-Performing Automatic License Plate Recognition Approach. *IEEE Transactions on Intelligent Transportation Systems*, 1-13. <https://doi.org/10.1109/TITS.2021.3087158>
- Wu, Y., Liu, S., & Wang, X. (2013). License plate location method based on texture and color. *Proceedings of the*



*IEEE International Conference on Software Engineering and Service Sciences, ICSESS, 1*, 361-364.  
<https://doi.org/10.1109/ICSESS.2013.6615324>

Zebin, T., Scully, P. J., Peek, N., Casson, A. J., & Ozanyan, K. B. (2019). Design and Implementation of a Convolutional Neural Network on an Edge Computing Smartphone for Human Activity Recognition. *IEEE Access*, 7, 133509-133520. <https://doi.org/10.1109/ACCESS.2019.2941836>

### **Copyrights**

Copyright for this article is retained by the author(s), with first publication rights granted to the journal.

This is an open-access article distributed under the terms and conditions of the Creative Commons Attribution license (<http://creativecommons.org/licenses/by/4.0/>).