Design Automation for Embedded Systems
https://doi.org/10.1007/s10617-019-09230-5

An embedded automatic license plate recognition system using deep learning

Diogo M. F. Izidio[1,2] · Antonyus P. A. Ferreira[2] · Heitor R. Medeiros[1,2] · Edna N. da S. Barros[1]

Received: 24 May 2019 / Accepted: 4 November 2019

Abstract
A system to automatically recognize vehicle license plates is a growing need to improve safety and traffic control, specifically in major urban centers. However, the license plate recognition task is generally computationally intensive, where the entire input image frame is scanned, the found plates are segmented, and character recognition is then performed for each segmented character. This paper presents a methodology for engineering a system to detect and recognize Brazilian license plates using convolutional neural networks (CNN) that is suitable for embedded systems. The resulting system detects license plates in the captured image using Tiny YOLOv3 architecture and identifies its characters using a second convolutional network trained on synthetic images and fine-tuned with real license plate images. The proposed architecture has demonstrated to be robust to angle, lightning, and noise variations while requiring a single forward pass for each network, therefore allowing faster processing compared to other deep learning approaches. Our methodology was validated using real license plate images under different environmental conditions reached a detection rate of 99.37% and an overall recognition rate of 98.43% while showing an average time of 2.70 s to process 1024 × 768 images with a single license plate in a Raspberry Pi3 (ARM Cortex-A53 CPU). To improve the recognition accuracy, an ensemble of CNN models was tested instead of a single CNN model, which resulted in an increase in the average processing time to 4.88 s for each image while increasing the recognition rate to 99.53%. Finally, we discuss the impact of using an ensemble of CNNs considering the accuracy-performance trade-off when engineering embedded systems for license plate recognition.

Keywords Embedded systems · Automatic license plate recognition (ALPR) · Image processing · Deep learning · Neural networks

✉ Diogo M. F. Izidio
dmfi@cin.ufpe.br

1 Federal University of Pernambuco, Recife, Brazil

2 Center For Strategic Technologies of the Northeast, Recife, Brazil

Published online: 13 November 2019

1 Introduction

The problem of automatic license plate recognition (ALPR) consists of recognizing a vehicle's license plate number from an image. An ALPR system robust to different environmental conditions can be useful in numerous applications such as automatic toll collection, parking control, targeted advertising, traffic surveillance, and many others where vehicle identification is required [1].

ALPR systems are generally composed of four stages: (1) the car image is acquired using a camera; (2) the license plate is segmented from the whole picture based on some features such as boundaries or character presence; (3) the license plate characters are individually segmented; and (4) template matching or some classifier like neural networks [2] or Support Vector Machines (SVMs) [3] is used to recognize of each character [4].

However, the detection and recognition of license plates is a complex computer vision problem. Some variations in the environment and license plate conditions can cause challenges for an ALPR system.

– *Illumination* In some cases, illumination is low and not uniform, and different portions of the plate can have different background colors. Also, there may be present some other effects caused by lighting like reflexes on the plate, which hinders most systems performance.
– *Angle* In some applications, the camera cannot be placed directly in front of the vehicle. Therefore, the plate may be tilted in the image. This type of variation can cause significant difficulty in segmenting the plate and individual characters.
– *Noise* When an image is captured, the capturing device can introduce noise in the final image. The amount of noise is dependent on the device and its parameters such as ISO.
– *Distortion* When capturing dynamic scenes, such as vehicles moving, some images can be blurred or distorted due to movement.
– *Failures* In some cases, license plates can be obscured by dirt or have failures in the characters.

Also, the success of the final stage is heavily dependent on the ability of the previous steps to locate the license plate and segment each character correctly [5]. Therefore, the chance of misclassification increases as more intermediary steps are introduced. Although significant progress has been achieved in the last decades, most systems only work under controlled conditions and controlled environments [6]. Therefore, there is still much room for improvement since an ALPR system must be accurate within a wide range of environmental and plate conditions.

On the other hand, deep learning methods have drastically improved the state of the art in computer vision tasks, such as object detection and recognition. In particular, deep convolutional neural networks are now the dominant approach for almost all recognition and detection tasks [7].

The convolutional neural network (CNN) architecture is mainly composed of three types of layers: convolutional layers, pooling layers, and fully connected layers, which are utilized to transform the raw input into multiple levels of abstraction [7]. Convolutional layers incorporate units connected to local patches which are responsible for detecting features from the previous layer. Pooling layers are responsible for merging similar features into one, reducing its dimensionality by down-sampling. Fully connected layers utilize the features captured by the previous layers to perform classification, regression, or some other type of problem.

The incorporation of deep learning-based approaches could bring to embedded systems a whole new set of features. However, most of these architectures rely on the high processing capacity of general-purpose GPUs (Graphics Processing Units), which often does not fit the constraints of embedded systems. Also, the photos taken by modern traffic control cameras often have high resolutions which make standard ALPR algorithms computationally costly [8].

This work proposes a combination of deep learning techniques suitable for the development of an embedded system to automatically detect and recognize license plate numbers in a video scene. Our main goal is to present a design methodology based on convolutional neural networks for engineering ALPR systems that reaches high accuracy in complex environments compared to conventional methods while considering the constraints of an embedded system. We evaluate the effect of using a CNN-based real-time object detection system suited for mobile devices like Tiny YOLO v3 [9] and investigate the use of CNNs for LP recognition. We also propose an ensemble of models to provide better confidence in the results and evaluate the trade-off between accuracy and performance requirements. The resultant system was customized to achieve the best accuracy using an embedded microprocessor with similar performance to other CPU/GPU approaches, with short response time and energy requirements suitable to be powered by batteries.

This paper is organized as follows. Section 2 presents related work. Section 3 brings in details about the system architecture. Section 4 shows the system values for accuracy and performance and discuss how it compares to related work. Sections 5 and 6 describe the conclusions and proposes future work.

2 Related work

The problems of license plate detection and recognition have many proposed solutions in the literature.

One of them is proposed by Sarfraz et al. [10], in which license plates are extracted from their original images using an edge detection Sobel filter followed by a selection of the regions with the aspect ratio similar to a license plate. The resultant image is binarized, and the characters are segmented based on the vertical projection of each column of the image. This strategy segments the characters based on the valleys and peaks of pixels intensity histograms for each column. Following the segmentation, character recognition is implemented using the template matching technique, which consists in a pixel-wise comparison with predefined templates for possible characters. In the end, the achieved accuracy was 96.22%, 94.04% and 95.24% for plate extraction, character segmentation, and classification, respectively.

A similar technique is proposed by Kocer [2] that also brings a similar extraction and character segmentation strategy. However, Cevic uses two ensembled MLP neural networks for character classification to improve the system's accuracy. This way, Cevic obtains a recognition rate of 95.36% in a dataset containing 259 images.

In Bjorklund's work [11], two Convolutional Neural Networks (CNN) are used, one to solve the plate extraction problem and another to perform the plate recognition. A convolutional stage is followed by the two separated CNNs that carry out the segmentation and classification in parallel. In this approach, individual characters are individually segmented. Also, the query image is repeatedly re-scaled to detect plates of arbitrary size, therefore representing a significant backlog for larger images. After segmentation, the individual characters are then recognized by a second convolutional neural network. Both networks are trained using synthetic generated datasets, and the system achieved an accuracy of 93% on plate detection and 97% on character classification.

In Rizvi et al. [12], two different convolutional networks are utilized for detecting Italian plates and characters simultaneously. Both networks are able to concurrently perform the task of detection and localization of each character by using shared connections between the convolutional layers. For recognition, a massive flow of operations which include image scaling, rotation and merging of results is presented for unrestricted environments like a desktop with a powerful GPU, which is reported to have an accuracy of 98% while showing a response time of 1.45 s in a Jetson TX1. While the response time seems reasonable for an embedded system, the Jetson TX1 board contains a GPU acceleration based on an NVIDIA Maxwell architecture with 256 CUDA cores, and the system is expected to run significantly slower on less powerful CPUs. A simplified flow for mobile applications is also proposed. However, the simplified approach requires specific light conditions and user intervention to apply the appropriate zoom for correct classification.

Li [6] proposes a system with a single Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) units in the context of sequence labeling to recognize the vehicle plates with no individual character segmentation. The system is also composed of four and nine layers CNNs to detect and classify the characters. An accuracy of 97.75% on the detection phase and 95.28% on character recognition is obtained on the AOLP dataset [13].

Netaji et al. [8] approaches the problem of license plate recognition by using an ensemble of three multilayer perceptrons (MLPs) based on the mixture of experts architecture for each segmented character. The detection of the license plate is performed through edge detection, and the segmentation of each character is obtained through a vertical projection histogram. The overall accuracy for localization is reported as 95.39% and total accuracy as 92.45%.

However, most related works share a common assumption that unlimited computational resources are available while trying to achieve maximum accuracy. Different from related works, our research aims to present a design methodology for a plate detection and recognition system utilizing CNNs suitable for mobile applications which require a single forward pass for each phase. The main objective is to propose a methodology that allows the implementation of the resulting system in embedded platforms while keeping accuracy levels comparable to state-of-the-art methods. The final system should not rely on methods that require high processing power or large memory requirements and be easily adaptable to suit any kind of license plate.

Consequently, recognition should not require any recurrent units such as RNN or LSTM, which require higher processing power and memory requirements [14] or any character segmentation, which requires multiple evaluations of the recognition network. We have also avoided CPU intensive processes such as pyramid images with multiple evaluations to design our methodology.

As a result, our final system should achieve less compute time compared to other works while utilizing a less powerful CPU and higher resolution images. Also, we aim to reach a power efficiency suitable to be powered by batteries for a reasonable amount of time and an inferior solution cost compared to other CPU/GPU based approaches.

3 ALPR system design methodology

The proposed system consists of two major phases. Firstly, the detection phase is performed receiving as input the entire image frame. The detection operation returns the detected plates as bounding boxes coordinates. After that, the step for recognition feature takes the seg-

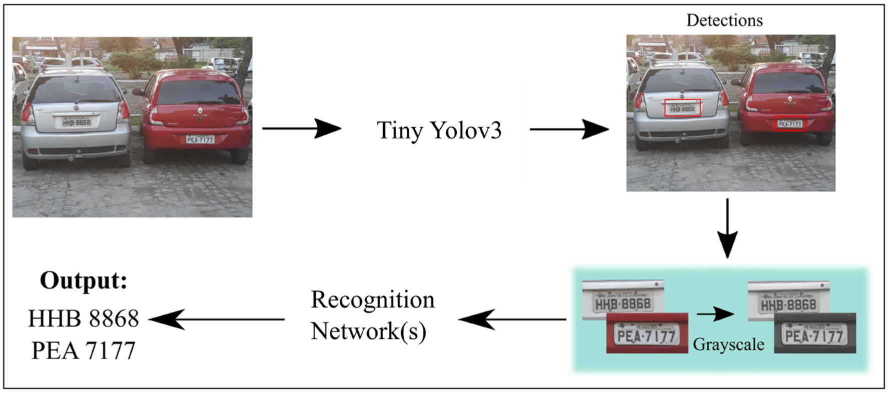

Fig. 1 Strategy for license plate recognition using deep learning

mented plates and performs character recognition of the whole sequence. This process can be visualized in Fig. 1.

The YOLO algorithm optimizes the processing using an image grid, which requires a single forward pass, instead of a sliding window strategy.

So, in this work, we combined the YOLO module [15] with the recognition phase. This strategy has the advantage of optimizing the computation while it only passes the small segmented plates to the recognition phase. Also, since the recognition is performed for the whole license plate, without the need for running the classification model for each character, the recognition process also requires a single forward pass, therefore reducing the processing power needed for each processed license plate.

3.1 Image generation

A large number of labeled images would be necessary to train the recognition network from scratch. Some deep convolutional networks are trained with millions of samples [16]. Therefore, to avoid needing millions of license plate images, the chosen solution consisted of the generation of synthetic images of license plates images with an 8-bit grayscale format as proposed by Earl [17]. Using synthetic images to train neural networks has shown to be a valuable low-cost approach in [11] while circumventing the time-consuming task of creating a hand-labeled dataset.

The use of synthetic license plates images greatly decreases the development cost of our system while still allowing CNN to achieve high accuracy. The benefits from using synthetic images are due to the similarities between the generated images and the real environment since it is generally possible to generate an image of a license plate while achieving high fidelity due to the simplicity of their designs. Also, using synthetic images allows multiple transformations to be performed in the generated license plate that could not be present in real pictures. Besides the excellent capability of generalization of the CNN reported by [11], it is also possible to further refine the accuracy results by utilizing a small number of real images, as is described in Sect. 3.4.

The process of generating the synthetic images used by this work follow well-defined steps. Firstly, in a blank canvas, random values for the intensity of the background and characters were chosen to compose the license plate layout. The colors were chosen, such as

Table 1 Fonts utilized in generating synthetic image for training

Fonts
Charles write bold
Heavy equipment
GL-Nummernshild-Eng
GL-Nummernshild-Mtl
License plate
Alte DIN 1451 Mittelschrift
Mandatory
UK number plate
Licenz plate

that the pixel value for the background should be at least 30% higher than the pixel value for the characters. This feature allows enough contrast so that the characters are utterly visible.

Secondly, the license plate characters are randomly picked following the Brazilian format (three letters followed by four numbers) and drawn on the background using a group of fonts often used for license plates. In Brazil, the standard font used in license plates is the *Mandatory* font. However, previous versions of LPs which are still in circulation still contain the *DIN Mittelschrift* font, which was included. Some other fonts, like the font utilized in UK license plates, were also used, resulting in the list of fonts which can be visualized in Table 1. We have used a uniform distribution to select which font is being used for a single image. As a result, at the beginning of the generation process, each font has an equal probability of being chosen, and we expect that our training batch has an approximately an equal number of examples of each font. The reason for the use of different fonts besides the Brazilian standard is because most fonts used in license plates around the world contain similarities, therefore, utilizing different fonts allows the network to learn high-level features associated with each letter without affecting the learning process. During our preliminary tests, it was observed that the addition of a more diverse set of fonts assisted the recognition network to recognize real images correctly.

After the layout is finished, affine transformations are made on the license plate layout on a 128×64 canvas, which is the output image resolution. The transformations used include rotation in three dimensions, scale, and translation within a range of commonly found values in license plates captured images. Finally, the image is composited with a random image from the SUN397 Scene benchmark [18], which contains 130,519 images of different scene categories. Even though in most testing cases the background is filled mostly by the vehicle color, the use of complex scenery in the generation process avoids overfitting by requiring the CNNs to not be dependent on the background information for the recognition process and rely solely on the information inside the license plates. To further improve the generalization capabilities of the CNN in the presence of small disturbances, we have also included at the end of the generation process some Gaussian noise with an amplitude of 0.09. The whole approach for image generation can be viewed in Fig. 2.

Therefore, 1000 images were generated for evaluating the training process as validation data and training images were generated during training itself by a background process on the CPU.

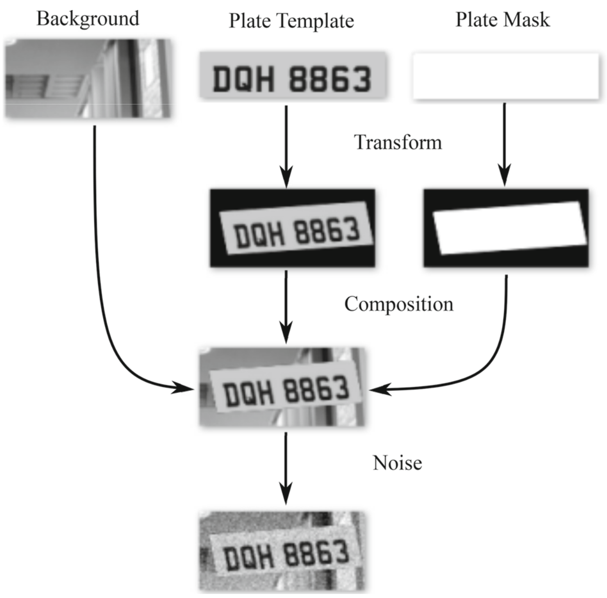

Fig. 2 Approach of generating synthetic images based on Brazilian format

3.2 License plate detection

To perform license plate detection on larger images, we used a miniature version of the state-of-the-art object detection network YOLOv3 [9], named Tiny YOLOv3, which is suited for mobile applications. Tiny YOLOv3 architecture can be viewed in Table 2.

The detection process starts by resizing the image to a 416 × 416 resolution and dividing the input image into two output grids of 13 × 13 and 26 × 26, which are responsible for performing object detection at two different scales while utilizing predefined anchor boxes [15] for each element of the grid. The use of two different grids allows the detection system to detect license plates in widely different scales without the need for repeated resizing and smoothing of images. We have used the default configuration and values for Tiny YOLOv3 [9] anchor boxes, where the 13 × 13 grid utilizes the three (3) largest anchor boxes, and the three (3) smallest anchor boxes are utilized by the 26 × 26 grid.

The training was performed using our custom dataset of 240 images obtained from the smartphones *Galaxy J5 Pro* and *Asus Zenfone 3*, which have 13 megapixels (f/1.9) and 16 megapixels (f/2.0), respectively, using a *Geforce GTX 1070 8GB*. The images were resized to 1024 × 768 and manually labeled with the smallest bounding boxes that enclosed the license plates for this dataset using the open-source software *BBox-Label-Tool*. The anchor boxes were obtained through the K-Means algorithm, and the convolutional layers were initialized with pre-trained weights for faster convergence.

After training, an additional 150 pictures were taken for testing following the same characteristics from the previous dataset, where the detection rate for our trained network was obtained.

For avoiding cases where the license plate is not entirely inside the box boundaries, the width and the height of the bounding box predicted from the output of the Tiny YOLOv3 are increased by 10%. After the first dilation, the height of the bounding box is adjusted to

Table 2 Architecture of Tiny YOLOv3 network

#	Layer	Filters	Size	Input	Output
0	conv	16	$3 \times 3/1$	$416 \times 416 \times 3$	$416 \times 416 \times 16$
1	max		$2 \times 2/2$	$416 \times 416 \times 16$	$208 \times 208 \times 16$
2	conv	32	$3 \times 3/1$	$208 \times 208 \times 16$	$208 \times 208 \times 32$
3	max		$2 \times 2/2$	$208 \times 208 \times 32$	$104 \times 104 \times 32$
4	conv	64	$3 \times 3/1$	$104 \times 104 \times 32$	$104 \times 104 \times 64$
5	max		$2 \times 2/2$	$104 \times 104 \times 64$	$52 \times 52 \times 64$
6	conv	128	$3 \times 3/1$	$52 \times 52 \times 64$	$52 \times 52 \times 128$
7	max		$2 \times 2/2$	$52 \times 52 \times 128$	$26 \times 26 \times 128$
8	conv	256	$3 \times 3/1$	$26 \times 26 \times 128$	$26 \times 26 \times 256$
9	max		$2 \times 2/2$	$26 \times 26 \times 256$	$13 \times 13 \times 256$
10	conv	512	$3 \times 3/1$	$13 \times 13 \times 256$	$13 \times 13 \times 512$
11	max		$2 \times 2/1$	$13 \times 13 \times 512$	$13 \times 13 \times 512$
12	conv	1024	$3 \times 3/1$	$13 \times 13 \times 512$	$13 \times 13 \times 1024$
13	conv	256	$1 \times 1/1$	$13 \times 13 \times 1024$	$13 \times 13 \times 256$
14	conv	512	$3 \times 3/1$	$13 \times 13 \times 256$	$13 \times 13 \times 512$
15	conv	18	$1 \times 1/1$	$13 \times 13 \times 512$	$13 \times 13 \times 18$
16	yolo				
17	route 13				
18	conv	128	$1 \times 1/1$	$13 \times 13 \times 256$	$13 \times 13 \times 128$
19	upsample 2×			$13 \times 13 \times 128$	$26 \times 26 \times 128$
20	route 19,8				
21	conv	256	$3 \times 3/1$	$26 \times 26 \times 384$	$26 \times 26 \times 256$
22	conv	18	$1 \times 1/1$	$26 \times 26 \times 256$	$26 \times 26 \times 18$
23	yolo				

keep a 2:1 aspect ratio for width and height, respectively. The resulting window inside the bounding box is converted to gray-scale, and then cropped and resized to the recognition network input size, which is 128×64 pixels.

3.3 License plate recognition

The recognition CNN receives as input the windows previously cropped and resized by the detection algorithm and outputs character predictions for each detected license plate.

The chosen architecture for the convolutional neural network of the recognition phase is shown in Fig. 3, which is based on the CNN proposed by Stark [19]. The proposed network architecture is composed of three convolutional layers, with 48, 64 and 128 filters, respectively. In all convolutional layers, 5×5 filters and padding of 2 were used. A max-pooling operation follows each convolutional layer. Lastly, there are two fully connected layers of size 2048 and 252, the last one being responsible for the output.

The output layer has 36×7 neurons, where it represents 36 possibilities (26 letters and ten digits) for each of the seven positions on the license plate.

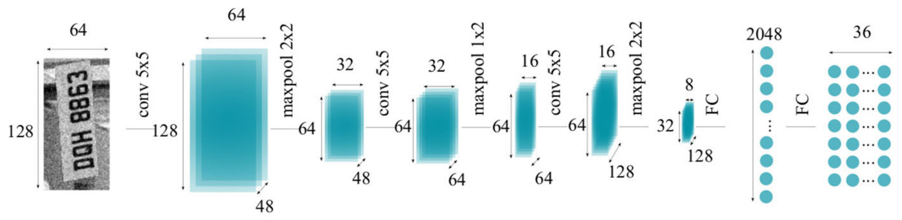

Fig. 3 Architecture for recognition network

Table 3 Training parameters used for Adam optimizer

Parameter	Value
Learning rate (α)	0.0001
1st order decay (β_1)	0.9
2nd order decay (β_2)	0.999
Epsilon (ϵ)	10^{-8}
Batch size	50

The first 36 output neurons are related to the first position. The first neuron represents the probability of the first position being character ‘A’, the second neuron represents the probability of the first position being ‘B’ and so on. Generally, the neuron which gives the probability of position i being character x is given by $n = 36i + \theta(x)$, where $\theta(x)$ is given by Equation 1.

$$\theta(x) = \begin{cases} 0 \ldots 25, & \text{if } x = \text{‘}A\text{’} \ldots \text{‘}Z\text{’} \\ 26 \ldots 36, & \text{if } x = \text{‘0’} \ldots \text{‘9’} \end{cases} \tag{1}$$

Each group of 36 neurons belongs to an individual *softmax* layer since each character is mutually exclusive. In total, there are seven *softmax* layers on the output, which represents the seven characters on the license plate.

The decision to keep the 36 characters for each output, allows the method to be easily adapted to different license plate with minimal modifications. As a result, our method remains versatile and can be used for engineering an ALPR system for distinct character arrangements with minimal intervention.

The training of the recognition network was conducted using a *Geforce GTX 1070 8GB* for around 10^6 batches. We trained the convolutional neural network six (6) separate times, keeping the model that performed best in the test set composed of 1000 synthetic images.

The Adam optimizer algorithm was used, which was found to be well-suited for non-convex optimization problems [20]. The training parameters can be observed in Table 3.

The plot for the training loss and the accuracy in the test set can be visualized in Fig. 4. After this training phase, the network performed 98.19% of accuracy in the test set of synthetic images.

3.4 Transfer learning

After using synthetic images, a dataset of real images was created using the same smartphones utilized in the creation of the detection dataset. Images were evenly split between the two capturing devices and later resized to 1024×768 resolution.

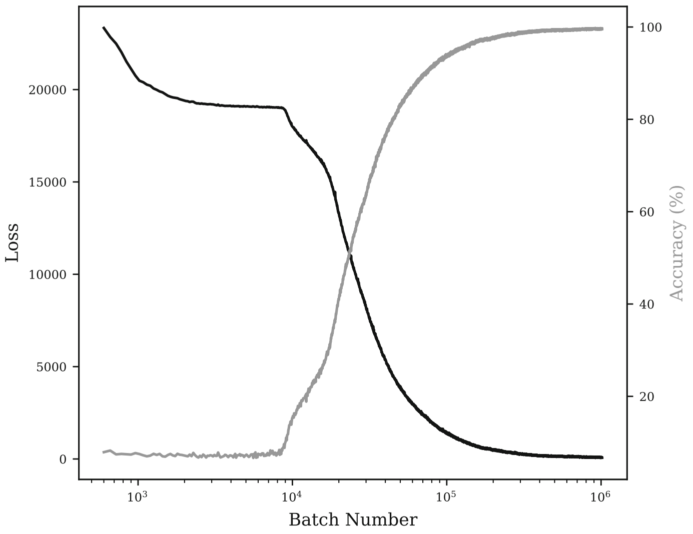

Fig. 4 Test loss and accuracy during training for our best network. Accuracy represents percentage of characters that are correctly recognized from the 1000 images test set

The images were taken in different weather conditions and different times of the day, between the period of April 2018 and May 2019, on the Geosciences and Technology Center (CTG) parking lot, from Federal University of Pernambuco. In total, 385 images were taken, where each image contains at least one license plate, photographed from different angles and distances. In general, we have aimed to equally distribute the angles from where the pictures were taken together with the camera height and kept the distances roughly between one (1) and thirty (30) meters.

Fifteen additional images were added to the dataset taken from online classified advertising websites which contained a license plate image to ensure variation in the capturing device, which resulted in a total of 400 images.

From this dataset, 800 license plate images were manually segmented and labeled. The increase in image numbers is possible due to the same license plate yielding different segmented images, with different scales or positions. Some examples of images used are present in Fig. 5. The resulting images were split into training (497 images), validation (121 images), and testing (182 images) sets, approximately following a 60%–15%–25% split.

By utilizing the training dataset, which was generated by the process described above, some fine-tuning was then performed using real license plate images to increase the accuracy with real data. This tuning allows the network to adjust to real license plate nuances that cannot be reproduced by synthetic images. This fine-tuning is a transfer learning technique which allows better predictions for license plate characters.

The learning rate was then lowered to $\eta = 0.00001$, and the network was trained once more for 10^3 batches, which resulted in the final network used for classification.

The complete training method for our approach is presented in Fig. 6. Also, the complete summary of all the datasets utilized in our work can be visualized in Table 4.

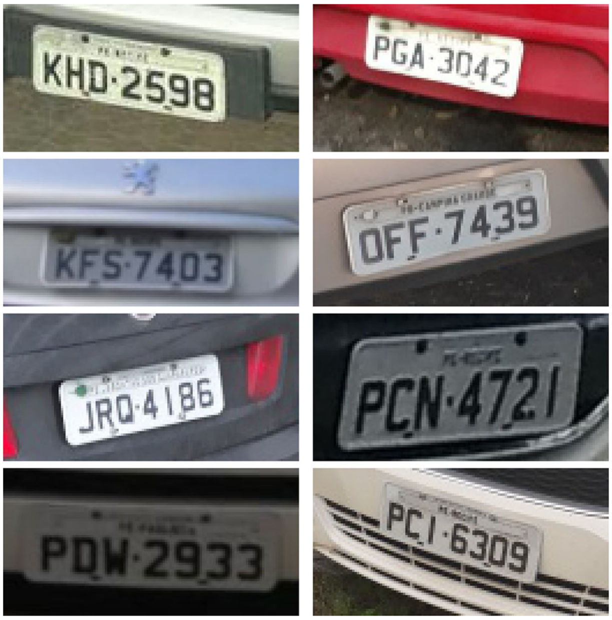

Fig. 5 Examples of manually segmented license plate images used in the feedback phase

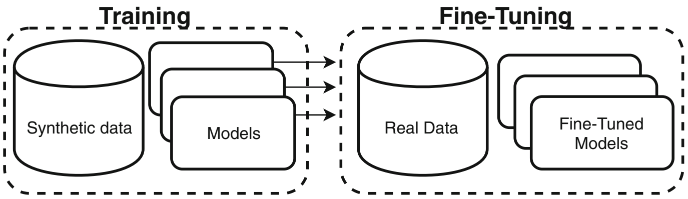

Fig. 6 Training pipeline utilized in our work. In the first step, the CNN models are trained with synthetic data, and fine-tuned with real data

Table 4 Number of images utilized for each step in our work

Task	Resolution	Train	Validation	Test	Total
Detection	1024 × 768	240	–	150	390
Recognition	128 × 64	497	121	182	800
Overall					1190

3.5 Ensembling

For each model that was trained using the method described above, we evaluated its accuracy for our validation recognition dataset. It was noticed that some specific characters caused the most number of errors. The characters that were responsible for the most significant amount of error varied for each training, which indicates that there is a stochastic variation inherent to the learning process.

As can be seen in Fig. 7, two different networks provide a histogram of errors with significant differences for each character. Although most errors are present for the same characters

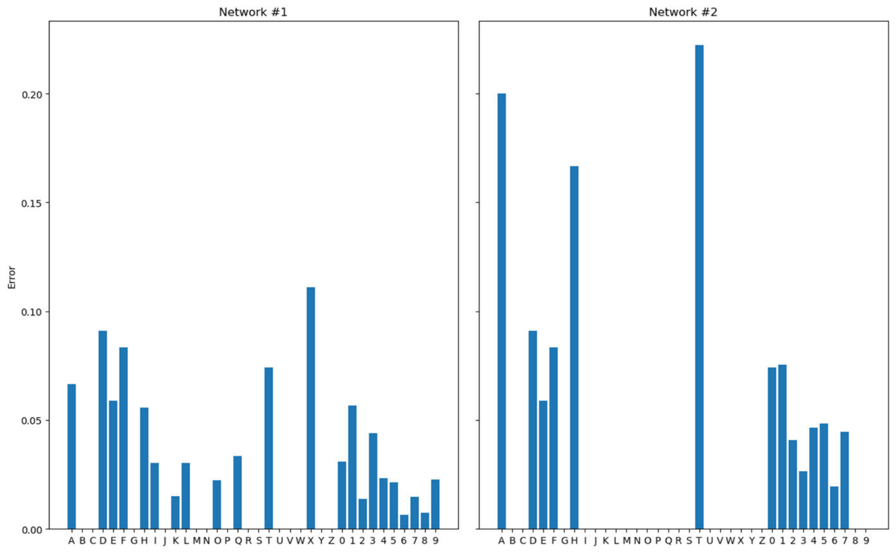

Fig. 7 Histogram of errors for each letter in two different networks

in both networks, there some stochastic variation between each training. For example, Network #2 provides an error for the letter T that is almost double than Network #1. Also, the subset of characters from I through S causes many spikes in error for Network #1 but not for Network #2. These different results are due to the stochastic nature of the training process for neural networks. The variability present in the generation of the dataset utilized for training in each network leads to different results.

Since the errors for each character are somewhat unrelated, models trained with this method are diverse enough to benefit from the effect of ensembles. As noted by [21], an ensemble is expected to correctly classify the input often performing better than any single classifier if the errors made by accurate classifiers are somewhat uncorrelated.

Due to this fact, we propose an ensemble of models where each separately trained model is responsible for recognizing the input image, and then a combination of its outputs is performed to provide better confidence in the result.

However, since diversity is a significant component of ensemble accuracy [22], we propose a method to evaluate diversity specifically to the ALPR problem by comparing the confusion matrix for each classifier using Euclidean distance, which is given by Eq. 2.

$$d(\mathbf{A}, \mathbf{B}) = \sum_{i=1}^{n} \sum_{j=1}^{n} |a_{ij} - b_{ij}| \tag{2}$$

A total of six (6) different networks were trained separately. For analyzing each network individually, data from the recognition validation set of real LPs were used to determine which letters caused the most errors and confusion matrices were built. The distance for each confusion matrix was calculated according to Eq. 2 and the results were displayed in a square matrix which can be visualized in Fig. 8.

Given the calculated distances and the fact that all models individually achieved similar accuracy, the selected networks were chosen given the biggest values for distance. We have,

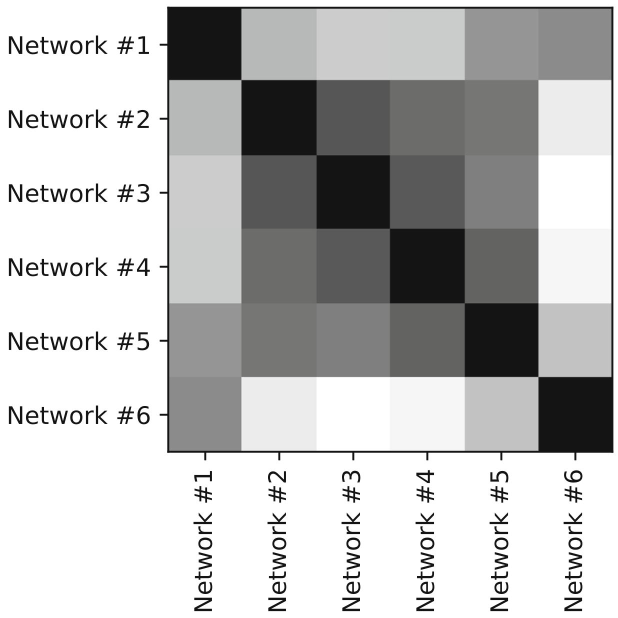

Fig. 8 Distance for the confusion matrix of each trained network. Brighter colors represent more distant confusion matrices

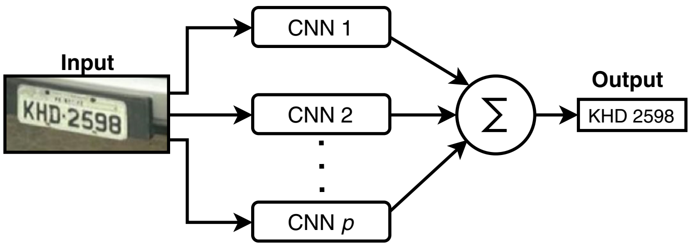

Fig. 9 Depiction of the utilized ensemble configuration, where each model receives the LP segmented image, and its outputs are combined through summation

Table 5 Accuracy results for CNN before and after feedback for a single network

Model	Accuracy (%)
No feedback	95.94
Feedback	98.43

in fact, verified that the chosen combination of models resulted in the best accuracy gain from the all possible combinations.

We evaluated two different approaches on how to combine different models. The first method consisted of a common ensemble technique which is presented in Fig. 9, where each CNN prediction is summed to compute the final output, which represents the probabilities for each letter. The second method consisted of an aggregation of each CNN output using the majority voting rule.

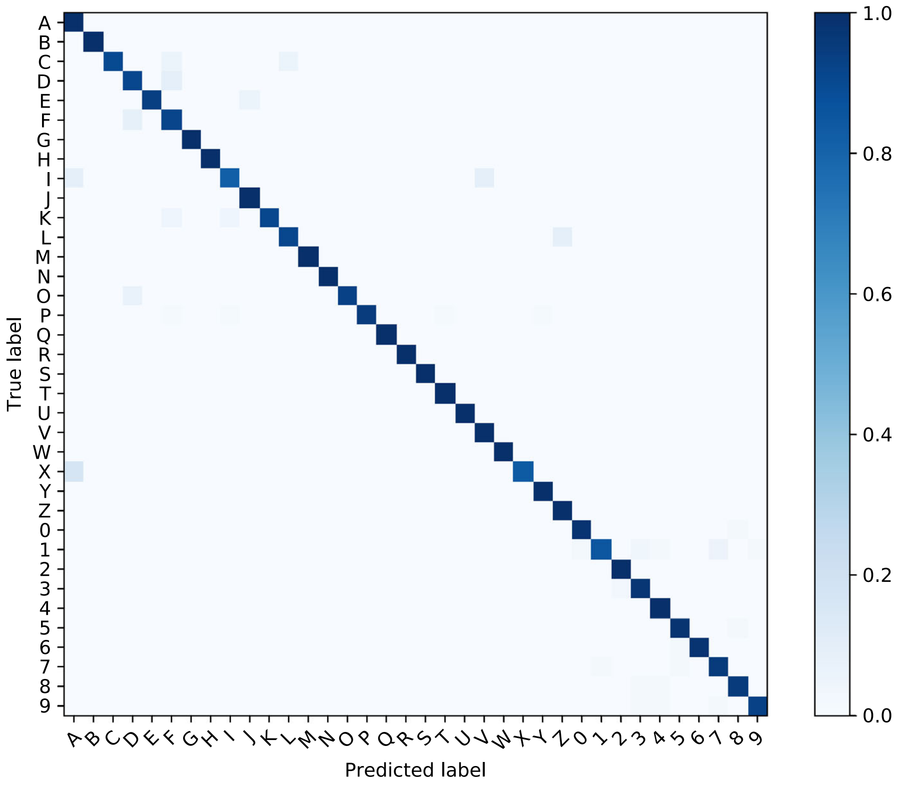

Fig. 10 Confusion matrix for the characters recognition for our best network before the feedback

When more models are allocated for the recognition task, mode processing power, and therefore more energy usage is required. Therefore, there is a trade-off between recognition rate (accuracy) and power and memory requirements (performance), as is commonly present in embedded systems.

As a result, the engineer can, based on the application requirements, decide how much accuracy loss is acceptable in order to gain performance. In applications that allow more extensive periods for processing a single image, an ensemble of three (3) or more networks may be adequate. However, for faster processing, a single network represent the best cost-efficient solution.

In our tests, we evaluated the use of two and three models for the ensemble. However, the final user may set the number of models utilized in the ensemble using the same methodology according to its expectancy of the response time of the system. For tasks like access control, where the recognition task must be performed more rapidly, it is reasonable to allocate fewer models to recognize the license plate, at the cost of more errors. In applications where the time is not as limited, or that the recognition rate affects the most, like speeding tickets, more models can be allocated, therefore improving the final accuracy of the system.

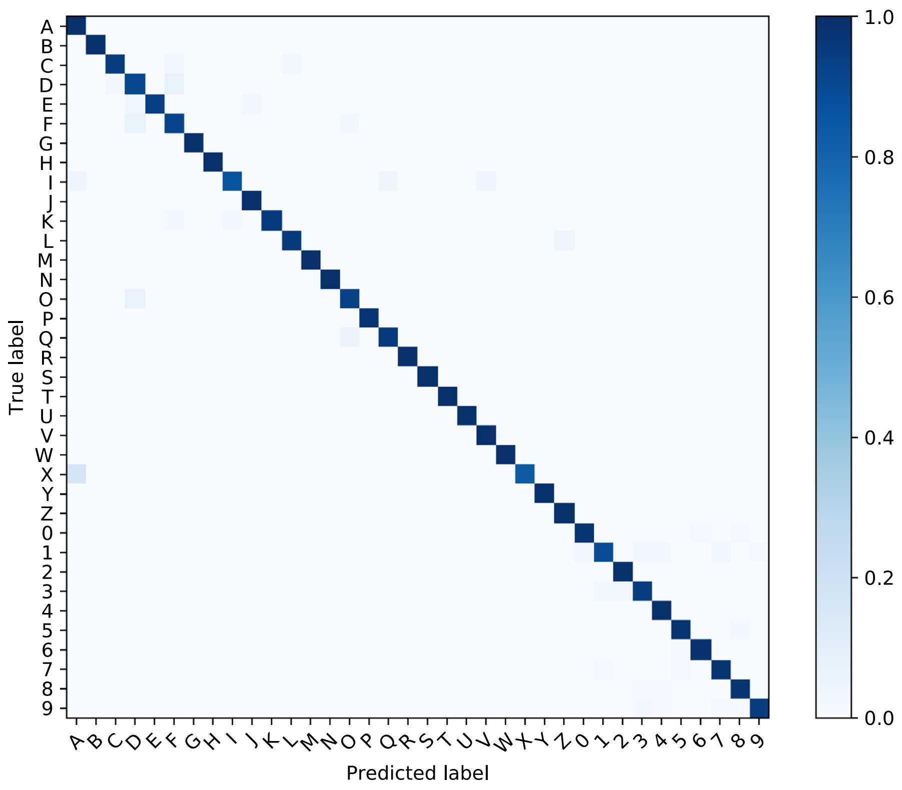

Fig. 11 Confusion matrix for the characters recognition for our best network after the feedback

4 Results and discussion

4.1 Accuracy

To evaluate the ability of each CNN to classify a LP correctly, we calculated accuracy for each network as the ratio between correctly recognized digits and the total number of digits in the utilized dataset.

The first test was performed on the testing set of the manually segmented images, which included the weights before and after the feedback with real license plate images for our best performing network. The difference in accuracy for this dataset between the network before and after fine-tuning can be visualized in Table 5.

Therefore, even though the original network showed good generalization to real license plate images, the accuracy gain of fine-tuning with real license plate images was about 2.49 percentage point. As such, the weights for the network after the feedback were used for the final system. The accuracy achieved for each character before and after the feedback can be visualized in the confusion matrices in Figs. 10 and 11, respectively.

In the confusion matrix representation, darker cells have higher percentages for that particular cell. The ideal confusion matrix is a diagonal matrix with dark cells only in the main diagonal.

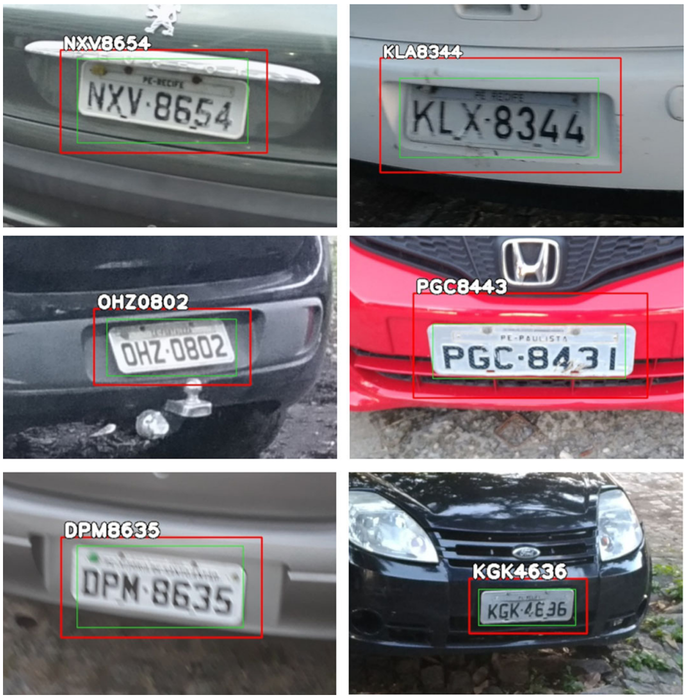

Fig. 12 Examples of correct predictions (on the left) and wrong predictions (on the right)

As indicated by Figs. 10 and 11, the system accuracy for all characters represents a great fit for the problem. One also can see the improvement obtained after the feedback strategy over the previous result. An example of a common mistake made by CNN includes misclassifying the number 1 as the number 7 and the letter F as the letter E, which is expected given the similarities in those characters.

After testing with the manually segmented images, the recognition network was tested receiving the output for Tiny YOLOv3 network on larger images. These tests were realized on the secondary dataset of 150 images, where each image contained one or more license plates. The detection accuracy for this dataset was 99.37%.

Some examples of correct and wrong predictions are present in Fig. 12, where the green box represents the output from the Tiny YOLOv3 network and the red box represents the window used for the input of the recognition CNN.

The final ensemble method utilized for our evaluations was the summation method, since it provided better results than the majority-voting classifier (99.53% vs. 99.06% for an ensemble of three networks), while requiring roughly the same amount of processing. Consequently, each CNN outputs a probability for the 7 digits in the LP and the digits that present the largest sum are selected as the predicted digit. The confusion matrix for the ensemble composed of three models can be visualized in Fig. 13.

Therefore, it can be seen that the use of an ensemble of three (3) independent models causes a 1.10 percentage point increase in recognition accuracy. While this does not seem like much, the recognition accuracy for the whole license plate, which contains seven (7) characters, is given by $0.9843^7 \approx 89.51\%$ for a single network and $0.9953^7 \approx 96.75\%$ for an ensemble composed of three (3) CNNs.

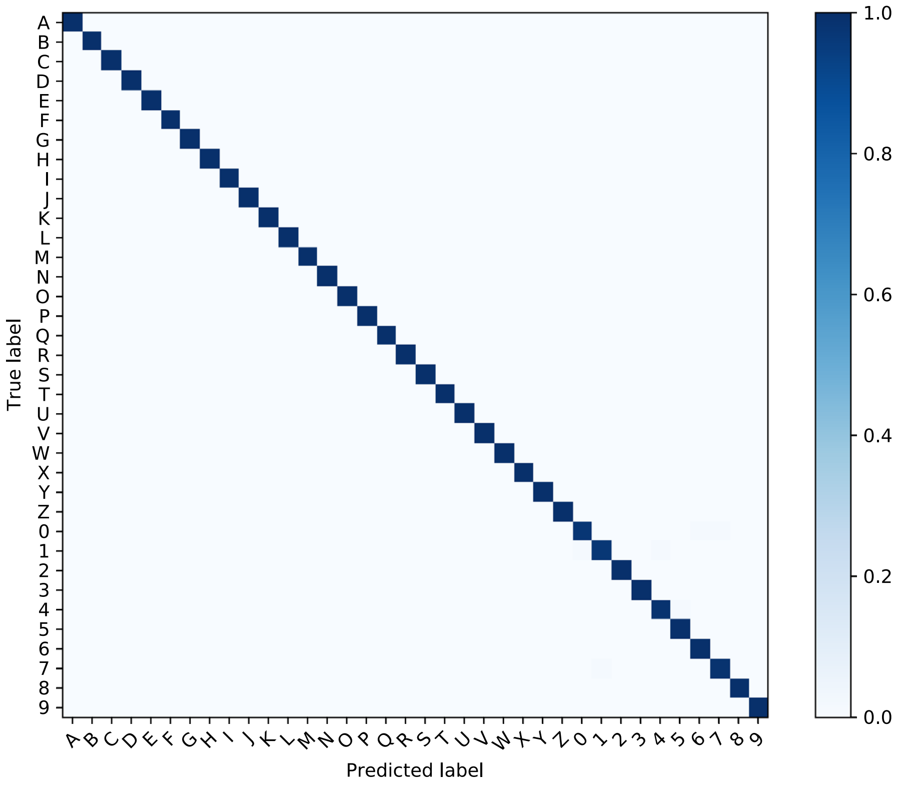

Fig. 13 Confusion matrix for the characters recognition for our ensemble composed by three different models

Overall, the CNN showed excellent performance on recognizing license plate from different angles and illumination conditions, achieving state-of-the-art results, as depicted in Table 6. Also, noise and distortion did not significantly affect the accuracy of the predictions. However, failures in license plates (like scratches and erased characters) were the leading cause of wrong classifications. Also, since each detection is processed individually by the recognition module, more than one license plate in the image would result in an increase in recognition time by the number of detections.

4.2 Performance and energy/memory requirements

The proposed system was implemented in a Raspberry Pi3, which used the Pi NoIR V2 camera module for capturing the images of the vehicles (see Fig. 14). The NoIR camera works like a regular camera, with some color distortion which can easily be corrected with the introduction of an infrared (IR) filter. The advantage of using NoIR camera is that it can be used in low light settings if accompanied by infrared lighting. However, the use of a regular RGB camera should not affect our methodology since our models don't rely on color information for recognition.

Table 6 Comparison of the accuracy of the proposed method with other works in the literature

Work	Dataset	LP detection (%)	Char. Segm. (%)	Char. Recog. (%)	Total Acc. (%)
Sarfras [10]	610 images	96.22	94.04	95.24	86.17
Kocer [2]	259 images	98.45	98.82	98.17	95.36
Li [6]	1176 images	97.30	–	95.70	93.10
Nejati [8]	5032 images	95.39	–	–	92.45
Single	1190 images	99.37	–	98.43	97.80
3-Ensemble	1190 images	99.37	–	99.53	98.90

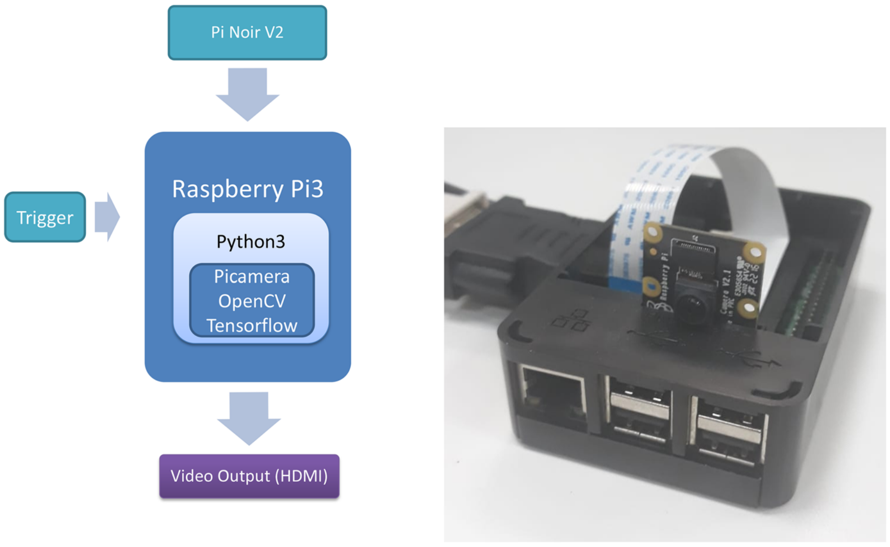

Fig. 14 Block diagram of embedded system for automatic license plate recognition using Raspberry Pi3 and final hardware assembly

Python3 programming language was used in conjunction with the following modules: (1) OpenCV library for image manipulation, (2) Picamera Python module for camera interfacing and (3) Tensorflow module as the Deep Learning framework used in both neural networks.

In terms of throughput, the system has shown an average response time of 2.70 s to process a 1024×768 image containing one license plate for a single network. The mean time taken for all approaches in each step and their respective accuracy values for character recognition are depicted in Table 7.

We have also measured the current consumed from the power supply for each component of our system. First, we observed the consumed current during an idle state and while processing an image and obtained the values in Table 8. As can be seen, the typical current drained by our system is 625 mA even when processing uninterruptedly, which under 5 V results in power consumption of 3.12 W. This power consumption allows the system to be powered by batteries for a reasonable amount of time. Even though the peak current increases the power consumption to 5.8 W, during our testing, this peak current generally lasts under half a second, therefore influencing the general power consumption on a small scale.

Table 7 Performance results on the Raspberry Pi3 for 1024 × 768 images

Method	Detection (s)	Recognition (s)	Total (s)	Recog. Acc. (%)
Single recognition network	1.61	1.09	2.70	98.43
Ensemble of 2 Recog. networks	1.61	2.18	3.79	99.14
Ensemble of 3 Recog. networks	1.61	3.27	4.88	99.53

Table 8 Energy requirements for our method

Component	Minimum	Typical	Peak
Idle	260 mA	260 mA	260 mA
Processing	20 mA	140 mA	650 mA
Camera	200 mA	225 mA	250 mA
Total	480 mA	625 mA	800 mA
	(2.4 W)	(3.12 W)	(5.8 W)

Table 9 Comparison of the implementation details of the proposed method with other works in the literature

Work	Total Acc. (%)	Platform	Resolution	Proc. time (s)
Sarfras [10]	86.17	Pentium III 700 Mhz	640 × 480	–
Kocer [2]	95.36	–	–	–
Li [6]	93.10	NVIDIA Tesla K40c	360 × 240	5
Nejati [8]	92.45	Intel Core i3	1280 × 960	–
Rizvi [12]	98.00	NVIDIA Jetson TX1	640 × 480	1.45
Single	97.80	Raspberry Pi3	1024 × 768	2.70
3-Ensemble	98.43	Raspberry Pi3	1024 × 768	4.88

We have also recorded memory requirements during our system implementation. Our records indicate that a RAM of $732MB$ is optimal for a single network because it is the peak value utilized during prediction. For an ensemble of three networks, this represents some limitation since the Raspberry Pi3 only has 1 GB of RAM has to utilize SWAP space or load the weight from storage for each network before prediction, which can take up to seven (7) seconds on top of the processing time, therefore significantly affecting the response time of the system. Newer versions of the Raspberry Pi4, however, include models with up to 4 GB of RAM, which enables them to perform the recognition significantly faster, without memory limitations for three models.

Also, since the CNNs work in parallel, this limitation in memory for ensembles in Raspberry Pi3 can be easily improved by using one device to process each CNN output. For low-cost platforms, this option represents a great solution to keep the response time equal to a single recognition network while still keeping a relatively small cost for the system. However, this results in another trade-off between Memory and Energy requirements, since powering n devices would require n times the amount of energy required to run a single network.

It can be seen by Table 9 that our method has achieved performance results similar to other works. However, we have user higher resolution images, and an ARM processor with limited processing power compared to other approaches, while reaching similar or less compute time. Since most works don't report energy consumption or memory requirements for their methods, a direct comparison for these values was not possible. However, subsequent works may use our values as a reference to guide future research.

5 Conclusions

The combination of Tiny YOLOv3 object detection and deep learning for character recognition is valuable for the problem of automatic license plate recognition in embedded platforms. Overall, the system has shown to be robust to variations in angle, lighting, and achieved state-of-the-art results. The proposed approach is segmentation-free, which reduces errors caused by lousy segmentation of the characters and requires a single evaluation for each license plate. Also, further feedbacks are expected to increase the overall accuracy of the system, which can continuously be improved using the same methodology used in Sect. 3.4.

Our results are significant while still considering the restrictions of an embedded system. We also have, to the best of our efforts, created a dataset that reflects real-life conditions, as most research claims to do, so that accuracy values across methods are somewhat comparable. However, the validity of our results relies on the assumption that the difficulty between datasets is similar, which cannot be empirically tested for private datasets.

The Raspberry Pi3 has shown to be capable of performing the calculations for the proposed algorithm in a reasonable amount of time. It also serves as a capturing device, which indicates that it can be used as a low-cost full ALPR system.

The needed time to process a single image frame depends on the number of license plates present in the image and represents the most significant constraint for the system. Despite having shown accuracy comparable to other methods in the literature, it cannot be applied to real-time video, which would require around 50 ms for each prediction to reach at least 20 frames per second. However, the response time is appropriate for applications that the pictures are taken sparsely and do not need an immediate response like toll collection, parking control, traffic enforcement, and many others.

6 Future work

For future work, the implementation of this system on platforms that contain CUDA enabled GPUs, like the Nvidia JETSON TK1, may be investigated. All Raspberry Pi3 calculations were performed by a CPU, which represents a significant backlog for the system. On a GPU, this system is expected to run at 20 frames per second, therefore allowing real-time monitoring of vehicles.

Regarding accuracy, even though our results are considered reliable, different configurations of ensembles could also be explored for further refinement. The use of ensembles of diverse models that have uncorrelated errors can provide increased accuracy. However, to the best of our knowledge, there no clear theoretical link between the different approaches of combining classifiers and final accuracy, which warrants further investigation. We have confirmed that the chosen combination has resulted in the best accuracy from all the combinations available for our trained networks. This result endorses our approach.

Finally, the proposed system only works for private vehicle license plates. One disadvantage of our proposed method is that it only works in LP with a specific format that was used for training. The inclusion of motorcycle and special license plates, which have a different format, can also be explored in further work. Also, due to the lack of availability for testing, vehicle registration plates of the Mercosur system were not included in this study. However, since most of the recognition phase is trained with a synthetically generated dataset, with a few changes in parameters, our methodology can be easily adapted to suit various types of license plate.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Yung N, Au K, Lai A (1999) Recognition of vehicle registration mark on moving vehicles in an outdoor environment. In: IEEE conference on intelligent transportation systems, proceedings, ITSC. IEEE
2. Kocer HE, Cevik KK (2011) Artificial neural networks based vehicle license plate recognition. Procedia Comput Sci 3:1033–1037
3. Chen Z-X, Liu C-Y, Chang F-L, Wang G-Y (2009) Automatic license-plate location and recognition based on feature salience. IEEE Trans Veh Technol 58(7):3781
4. Du S, Ibrahim M, Shehata M, Badawy W (2013) Automatic license plate recognition (ALPR): a state-of-the-art review. IEEE Trans Circuits Syst Video Technol 23(2):311–325
5. Patel C, Shah D, Patel A (2013) Automatic number plate recognition system (ANPR): a survey. Int J Comput Appl 69(9):21–33
6. Li H, Shen C (2016) Reading car license plates using deep convolutional neural networks and lstms, arXiv preprint arXiv:1601.05610
7. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436
8. Nejati M, Majidi A, Jalalat M (2015) License plate recognition based on edge histogram analysis and classifier ensemble. In: 2015 Signal processing and intelligent systems conference (SPIS) IEEE, 2015, pp 48–52
9. Redmon J, Farhadi A (2018) Yolov3: an incremental improvement. arXiv preprint arXiv:1804.02767
10. Sarfraz M, Ahmed MJ, Ghazi SA (2003) Saudi Arabian license plate recognition system. In: 2003 International conference on geometric modeling and graphics, 2003. Proceedings. IEEE, 2003, pp 36–41
11. Björklund T, Fiandrotti A, Annarumma M, Francini G, Magli E (2017) Automatic license plate recognition with convolutional neural networks trained on synthetic data. In: 2017 IEEE 19th international workshop on multimedia signal processing (MMSP). IEEE, 2017, pp 1–6
12. Rizvi S, Patti D, Björklund T, Cabodi G, Francini G (2017) Deep classifiers-based license plate detection, localization and recognition on gpu-powered mobile platform. Fut Internet 9(4):66
13. Hsu G-S, Chen J-C, Chung Y-Z (2012) Application-oriented license plate recognition. IEEE Trans Veh Technol 62(2):552–561
14. Rezk NM, Purnaprajna M, Nordström T, Ul-Abdin Z (2019) Recurrent neural networks: an embedded computing perspective, arXiv preprint arXiv:1908.07062
15. Redmon J, Farhadi A (2016) YOLO9000: better, faster, stronger, *ArXiv e-prints*
16. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105
17. Earl M (2016) Number plate recognition with tensorflow. https://github.com/matthewearl/deep-anpr
18. Xiao J, Hays J, Ehinger KA, Oliva A, Torralba A (2010) Sun database: large-scale scene recognition from abbey to zoo. In: 2010 IEEE conference on computer vision and pattern recognition (CVPR). IEEE, 2010, pp 3485–3492
19. Stark F, Hazırbas C, Triebel R, Cremers D (2015) Captcha recognition with active deep learning. In: GCPR workshop on new challenges in neural computation
20. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980
21. Dietterich TG (2000) Ensemble methods in machine learning. In: International workshop on multiple classifier systems. Springer, pp 1–15
22. Brown G, Kuncheva LI (2010) “Good” and “bad” diversity in majority vote ensembles. In: International workshop on multiple classifier systems. Springer, Berlin, pp 124–133