An Automated License Plate Recognition System with Vehicle Detection and Classification based on Deep Learning

Reda Al-batat[1], Anastassia Angelopoulou[1*†], Jude Hemanth[2] and Epaminondas Kapetanios[3†]

[1]School of Computer Science and Engineering, University of Westminster, UK.
[2]Karunya Institute of Technology and Sciences, Karunya University, India.
[3]School of Physics, Engineering and Computer Science, University of Hertfordshire, UK.

*Corresponding author(s). E-mail(s): agelopa@westminster.ac.uk;
Contributing authors: reda.alb2@hotmail.com; judehemanth@karunya.edu; e.kapetanios@herts.ac.uk;
†These authors contributed equally to this work.

Abstract

An accurate and robust Automatic License Plate Recognition (ALPR) method is crucial in any Intelligent Transportation and Surveillance (ITS) system. Such systems are required to have 100% recognition rate, as one character mistake will lead to the wrong license plate (LP), making this a very challenging task. There have been many attempts in creating a robust ALPR system and great results have been achieved. However, most use prior knowledge such as specific country LP character layouts and fixed pre and post processing rules, which will for example, breakdown with foreign vehicles, personalised LPs, and add an overhead to the pipeline. In this paper, we propose an end-to-end generic ALPR system, that uses no prior knowledge on country LPs layout, with no pre or post processing steps, and achieve competitive results in 5 public datasets. The whole ALPR pipeline is considered, from the vehicle detection to LP detection and finally the LP recognition stage; including a vehicle classifier for heavy trucks and emergency vehicles. Each stage of the pipeline is based on the state-of-the-art YOLO detector with various data augmentation and data generation techniques to obtain an LP recognition accuracy that is on par with current proposed methods, and is country independent with no additional steps at inference.

1 Introduction

In any future of ITS systems, having a robust and reliable way of identifying vehicles no matter what the vehicle is or where it is from is always going to be important, especially if road rules are to be followed and prevent anyone breaking any laws, which ultimately is going to make the roads safer, and also ensure traffic flow is the most efficient and operated effectively. However, what is special about this task, is that it is extremely important to have a recognition rate that is 100% accurate as only one single character mistake in the LP will lead to the wrong identification of the vehicle.

In any ALPR system the whole pipeline should be considered, such as: 1) Vehicle detection (VD); 2) LP detection (LPD); 3) LP recognition (LPR). It all starts from an image data source, for example, from a CCTV camera overlooking a motorway. Detection of all the vehicles in the image is then performed since more information and surveillance data can be extracted from the vehicle, such as if it is a truck or emergency vehicle. Obtaining more information about the type of the vehicle is very important since emergency vehicles should not be fined for using a bus lane or for speeding up on a particular road. Preventing such vehicles from entering the system would make it more useful and streamlined in making a fully automated system. Furthermore, this can be expanded to obtain the vehicle make, model, colour, etc. to be more useful. The second stage is the LP detection, where the LP is located in the image. Image resolution is reduced to only the vehicle patch obtained from the previous stage, this leads to a reduced search space, increasing the efficiency. The final stage is the LP recognition, where for each LP detected, we obtain the characters, and from there, the full LP text is formed to identify the vehicle.

Furthermore, it is very important to consider all these three stages, because each stage affects the performance of the next. For example, starting of with the vehicle patches, gives a great advantage as we can assume 100% accuracy for the vehicle detection stage.

This also applies with the LP detection stage, if starting with just the LP patch, you will be skipping two stages, so you are able to assume that you have 100% detection accuracy on the two previous stages, this eliminates considerable amount of variability which otherwise in practise will be present and your final result will not reflect it. This ”skipping” of stages will not be reflected in the recognition accuracy, hence why it is critical to include all stages of the pipeline in an ALPR system.

Additionally, having an ALPR system that does not rely on the positions of letters and digits in an LP and one that is country independent is also important since, for example, of the possibility of vehicles entering the country from other countries which will have a different LP layout. Also, anyone is able to obtain a personalised LP, which may have a completely different layout from what is normally used in the country of reference. So any pre-defined rules that may have been setup to increase the recognition rate will most likely fail in those cases. Furthermore, if this is a known issue, it can be exploited and cause further problems, so using any specific prior knowledge is not the way forward. In addition, we have evaluated our ALPR system with as many datasets as possible, so we get a wide variety and a realistic representation of the real world with different lighting conditions, backgrounds, and orientations.

In this paper, we propose a fully automated ALPR pipeline that does not use any pre-defined rules, uses a wide range of datasets which have different character sequences and conditions, and increase the datasets by more than triple by using various data augmentation and data generation techniques coupled with the You Only Look Once (YOLO) detector at each stage. There are three main stages to the pipeline, the first stage is the vehicle detection stage, where all the vehicles in the image are detected. Following the vehicle detection stage, each vehicle patch is cropped and fed into an LP detector, which detects the LP of the vehicle, since each vehicle can only have one LP, the highest confidence detection is chosen if multiple detections occur. Additionally, for each vehicle patch after the vehicle detection stage, each vehicle patch goes through a ResNet50 classifier to classify the vehicles into three classes: truck, emergency vehicle, and other. The final stage is the LP character detection stage, where each character of the LP is detected and assembled to form the full LP text. The full implementation of the ALPR pipeline is openly available at: https://github.com/RedaAlb/alpr-pipeline.

The paper is structured as follows: Section 2 gives an overview of the related works. Section 3 outlines the methodological approach followed by section 4 with the experimental design and results. A conclusion provides a summary of the key contributions and results of this paper.

2 Related Work

This section is split into the main areas of a fully automated ALPR system, and discuss how previous and current methods address each stage of the pipeline.

2.1 ALPR pipeline

Considering the full ALPR pipeline may not only be very useful, but is very crucial. Additionally, disregarding stages might indirectly effect the final results of the overall ALPR system. Examples of this are [1, 2] where the first two stages were skipped, the VD and the LPD stages. And also [3, 4] where the VD stage was not considered but still performed the LPD and LPR stages. Now this might be just because of what was needed for their specific application, however, considering the full pipeline is needed for a complete automated ALPR system. [4, 5] also consider only one LP per image, so one vehicle per frame, this is not realistic in practise, as you are bound to have multiple vehicles in a single frame, which also will increase the processing time. However, this may be useful for parking spaces, where the conditions are very fixed and only one vehicle is present at the entrance gate, but not suitable for a real-world general scenario on a motorway.

There are some proposed methods that consider the whole pipeline, such as [6, 7], and achieve great results. However, [6] only considers one dataset with Brazilian LPs, where the images were only frontal views of the vehicles. [7] achieved very good results, and compared its method in 8 public datasets, and outperformed most previous methods. However, despite their very great results, they relied on exploiting specific country layouts. So they would detect the LP but also attach a country associated with that LP, and knowing the country of the LP, they use fixed rules of the LP layout, such as if it is known that the first two characters of the LP is only characters. Then if for the first two characters of the LP was predicted the digit ”1”, then it will be replaced and considered to be an ”I”. This works well, but it is impractical for LPs from all around the world with foreign and custom LPs in the mix, and will possibly require specific rules for each country in the world. This paper could be seen as a continuation from where [7] left off, with a change of making a general ALPR rather then focusing on country specific LPs.

[2] considered only the LPD stage, and focused on obtaining the angle of the LP bounding box (BB), they achieved great detection results, but simplified the problem by forcing their ALPR system to output only one BB per image (only one vehicle for every image), which is not practical and very limiting, especially for a general ALPR system.

2.2 Real-time

Having an ALPR system that performs in real-time is very important. This is because if a vehicle is travelling at, for example, 60 miles per hour, and you have a low FPS, the vehicle might only be in the frame once or twice, and depending on camera location and how many vehicles are in the frame and their speeds, vehicles might be completely missed. So having an ALPR system that performs in a relatively high FPS is important not only to ensure all vehicles are detected but also allows the system more attempts to detect and recognise the LP as the vehicle moves across the frame, where each position will present different lighting condition, camera angle, background, etc.

Examples of where the FPS was too low to use in practical settings are [3, 5]. Despite [5] using a dedicated GPU (GT-740M), performed very slow at 230ms ($\sim$4 FPS) to only detect LPs, which is way to slow for real-world applications with high speed moving vehicles. [3] achieved relatively good results, but on a very high end GPU (NVIDIA Tesla K40c), they had multiple steps, using high demanding methods such as sliding window, causing their system to operate at 2 seconds per frame, which is not practical.

There are some works that achieve great FPS, such as [2, 4, 6, 7]. [6] achieve 76 FPS using a high end GPU, but the recognition accuracy is very poor on the SSIG-SegPlate dataset, at 63%. [2] achieved a very good $\sim$5ms per frame in a relatively inexpensive GPU (FTX980) for the LPD stage, which is the only stage they considered and is not really comparable to the overall FPS of the above methods. [4] also used a relatively inexpensive GPU (NVIDIA 1080 Ti), and was able to process images in 0.0443 seconds for both the LPD and LPR stages but they only considered one LP per image. [7], achieves 73 FPS with a high end GPU (NVIDIA Titan

XP) for when 1 vehicle is present in the image, for 5 vehicles, the FPS drops to 29 FPS, which is still good, this is because they are using the YOLOv2 detector, which is known for its speed.

Based on the above, methods that gained a high FPS is for reasons that will not be present in an overall ALPR system, such as skipping stages, only considering one vehicle per image. This might work well for specific application domains but not towards a general ALPR system.

2.3 Datasets

Using more than one dataset is vital for a good overall ALPR system to obtain all kinds of different variations, such as lighting, backgrounds, size of vehicles, and camera angels. Having an ALPR system that performs in a wide range of datasets also means it will perform better in the real-world. There are some methods who use only one dataset, which may be that was what was available to them at the time, but it will not be very generalisable. Examples of this are [3, 5, 6]. Using a few datasets, makes the LPs biased towards those countries and all the processing steps will be biased, or will not provide enough substantial variations in the LP itself. [1, 2, 4] use a few datasets, but again, not quite enough, and most cases cover only one country. [7] uses 8 publicly available datasets, which contain significant variations between datasets, but [7] focuses on separating the ALPR system based on the country detected and have fixed rules at inference, which is not ideal for a fully automated ALPR as there are too many countries to cover.

3 Methodology

In this paper, we present a fully automated ALPR pipeline that does not use any pre-defined rules, together with the experimental findings associated with the YOLO detector. Each feature extraction model from our system is discussed in greater detail in the following sub-sections: 1) Datasets; 2) Pipeline, going through the whole overall stages of the pipeline; 3) Vehicle detection; 4) Vehicle type classification; 5) Licence plate detection; 6) License plate recognition.

This work is an improvement of what [7] started, but taking it towards a more generic automated ALPR system, and utilise newer and better version of YOLO. This is because [7] have already achieved great results in 5 public datasets, and have demonstrated their results very clearly and obtained better recognition rate than most previous methods. So here, we use the same datasets but with no post processing or fixed rules based on country specific layouts.

3.1 Datasets

It is very important to use datasets that are publicly available and have been used by most previous methods to get fair and good comparisons. There are 8 publicly available datasets that are used by most researchers, some use a few, some use several of them. In this paper, 5 of them are used as they are openly available. The datasets used are Caltech Cars [8], English LP [9], OpenALPR EU [10], AOLP [11], and UFPR ALPR [12]. Table 1, shows key details of the datasets.

Table 1 Datasets used.

Dataset	**# Samples**	**Country**
Caltech Cars	124	America
English LP	509	EU
OpenALPR EU	108	EU
AOLP	2049	Taiwan
UFPR ALPR	4500	Brazil
Total	**7290**	

3.2 Pipeline

All stages of the pipeline is made up of a YOLO [13] detector, more specifically YOLOv4 [14] tiny. The reason why YOLO is chosen is because it is currently the state-of-the-art detector when it comes to speed without too much compromise in accuracy. It is also used by many proposed methods in this domain because of its speed and desire to obtain real-time performance, such as [2, 4, 6, 7]. However, all previous methods are using old versions of YOLO, as YOLOv4 was recently published in 2020. In figure 1, the full pipeline of our ALPR is illustrated.

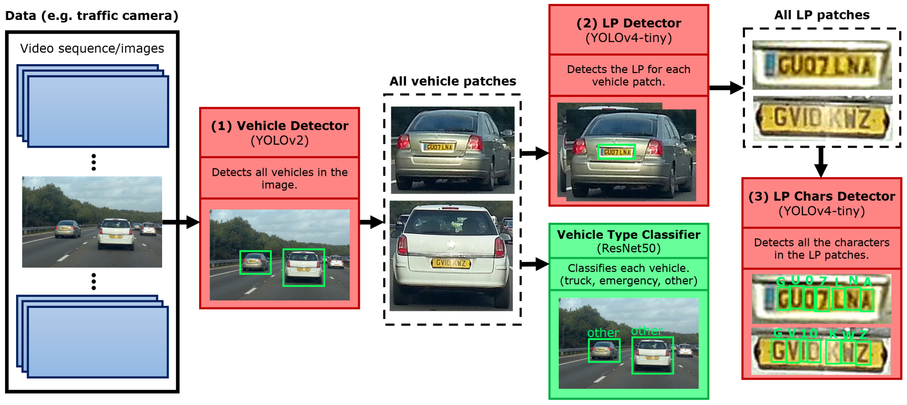
Data (e.g. traffic camera)
Video sequence/images
(1) Vehicle Detector (YOLOv2)
Detects all vehicles in the image.
All vehicle patches
(2) LP Detector (YOLOv4-tiny)
Detects the LP for each vehicle patch.
Vehicle Type Classifier (ResNet50)
Classifies each vehicle. (truck, emergency, other)
other
other
All LP patches
(3) LP Chars Detector (YOLOv4-tiny)
Detects all the characters in the LP patches.

Fig. 1 Full ALPR pipeline, including a vehicle type classifier.

We start of with the full image from a surveillance camera. The image first goes through the vehicle detector (1), where all vehicles are detected. All vehicle patches of the detected vehicles gets cropped, and each of those patches then go through two models. First, it goes through the vehicle classifier, to determine what type of vehicle it is, for example, an emergency vehicle or a truck. Secondly, each patch also continuous through to the LP detector (2), where the LP is detected. So now we would have an LP patch for each of the vehicles detected. All those LP patches then go through the final model, the LP character detector (3), which is a detector that detects all characters in the LP patch. From there, the full LP characters are constructed. So at the end, for every vehicle in the image, we will end up with the type of the vehicle (which can be extended to multiple classes, even to the vehicle make, model, year, etc. if needed) and the LP of the vehicle.

3.3 Vehicle Detection (VD)

For the vehicle detection, the same exact method as [7] was used, as a very high accuracy with excellent FPS was already achieved, there was not much improvement to do of 99.92% recall when an intersection over union (IoU) of 0.25 was used. In addition, this method has a very high accuracy with 117 FPS. However, an addition was made to ignore all vehicle patches that were less than 40 pixels either in width or height as it would be too small for even a human to read the LP when enlarged. YOLOv2 is used for this stage, the model architecture is shown in table 2.

Table 2 Vehicle detection model architecture.

Layer	Filters	Size/Strd	Input	Output
0 conv	32	3x3/1	608x416x3	608x416x32
1 max		2x2/2	608x416x32	304x208x32
2 conv	64	3x3/1	304x208x32	304x208x64
3 max		2x2/2	304x208x64	152x104x64
4 conv	128	3x3/1	152x104x64	152x104x128
5 conv	64	1x1/1	152x104x128	152x104x64
6 conv	128	3x3/1	152x104x64	152x104x128
7 max		2x2/2	152x104x128	76x52x128
8 conv	256	3x3/1	76x52x128	76x52x256
9 conv	128	1x1/1	76x52x256	76x52x128
10 conv	256	3x3/1	76x52x128	76x52x256
11 max		2x2/2	76x52x256	38x26x256
12 conv	512	3x3/1	38x26x256	38x26x512
13 conv	256	1x1/1	38x26x512	38x26x256
14 conv	512	3x3/1	38x26x256	38x26x512
15 conv	256	1x1/1	38x26x512	38x26x256
16 conv	512	3x3/1	38x26x256	38x26x512
17 max		2x2/2	38x26x512	19x13x512
18 conv	1024	3x3/1	19x13x512	19x13x1024
19 conv	512	1x1/1	19x13x1024	19x13x512
20 conv	1024	3x3/1	19x13x512	19x13x1024
21 conv	512	1x1/1	19x13x1024	19x13x512
22 conv	1024	3x3/1	19x13x512	19x13x1024
23 conv	1024	3x3/1	19x13x1024	19x13x1024
24 conv	1024	3x3/1	19x13x1024	19x13x1024
25 route	16			38x26x512
26 reorg		/2	38x26x512	19x13x2048
27 route	26 24			19x13x3072
28 conv	1024	3x3/1	19x13x3072	19x13x1024
29 conv	35	1x1/1	19x13x1024	19x13x35

3.4 Vehicle Type Classification (VTC)

For the vehicle type classification, as a start, a classifier was made for trucks and emergency vehicles, which could easily be expanded to other classes. There is also a third class ”other”, which refers to all other types of vehicles, for example, cars and motorcycles. Images were gathered from random places from the internet, including images from Open Images V6 [15]. 449 samples of emergency vehicles, 374 samples of trucks, and 831 samples of ”other” were used. The classifier used is a ResNet50 [16] with data augmentations of rotation, width and height shifts, brightness, shear, zoom, and horizontal flip. Transfer learning was used where the weights used is the ResNet50 model trained on the COCO dataset [17], all the ResNet50 layers are frozen, and average pooling was added to the end, followed by a fully connected layer of 32, followed by the softmax output layer, which lead to 65,667 trainable parameters, with 23,587,712 frozen weights which were trained on the COCO dataset.

3.5 Licence Plate Detection (LPD)

The LPD stage is a YOLOv4 tiny model. The training images are the cropped vehicle patches from the full images. To increase the dataset and improve accuracy, the dataset is doubled using the negative images of all samples, this doubling of the dataset increased the accuracy significantly. The detection is constrained to detections with an IoU of greater than 0.65, and if multiple were detected, the highest confidence score was chosen, as a vehicle can only have one LP. The full model architecture can be seen in table 3. By using the latest YOLOv4 detector, we will see in section 4 that this detector outperforms previous methods in the same dataset.

3.6 Licence Plate Recognition (LPR)

The first two stages are much easier, they do not bring any major challenges, with enough data, it is fairly straight forward to get high enough accuracy. The real challenge is the final stage of obtaining each character of the LP to ultimately identify the vehicle.

In this stage, again, the YOLOv4 tiny detector was used. Table 4 shows the full LPR YOLO network architecture. There was no major modifications done to the architecture apart from. 1) Changing the network input to 352x128, which was chosen because the average aspect ratio (w/h) of all LP patches across all datasets is 2.86. 2) The number of filters of each convolutional layer before each YOLO layer was $filters = ((classes + 5) \times anchors)$, where the number of classes is 36 (0-9 + A-Z) and the number of anchors is 3, resulting in 123 filters. 3) Disabled the flip augmentation as this will result in flipped characters which will not be useful for the model to learn. Only LP patches that are larger than 20 and 10 pixels in width and height respectively was considered to be an LP.

It is important to note that all digits and characters were considered as their own class, so the classes were 0-9, and A-Z, making a total of 36 classes. Unlike other methods such as [1, 6, 7], where for example, the digit ”0” was assumed to be the

Table 3 LP detection model architecture.

Layer	Filters	Size/Strd	Input	Output
0 conv	32	3x3/2	416x416x3	208x208x32
1 conv	64	3x3/2	208x208x32	104x104x64
2 conv	64	3x3/1	104x104x64	104x104x64
3 route	2		1/2	104x104x32
4 conv	32	3x3/1	104x104x32	104x104x32
5 conv	32	3x3/1	104x104x32	104x104x32
6 route	5 4			104x104x64
7 conv	64	1x1/1	104x104x64	104x104x64
8 route	2 7			104x104x128
9 max		2x2/2	104x104x128	52x52x128
10 conv	128	3x3/1	52x52x128	52x52x128
11 route	10		1/2	52x52x64
12 conv	64	3x3/1	52x52x64	52x52x64
13 conv	64	3x3/1	52x52x64	52x52x64
14 route	13 12			52x52x128
15 conv	128	1x1/1	52x52x128	52x52x128
16 route	10 15			52x52x256
17 max		2x2/2	52x52x256	26x26x256
18 conv	256	3x3/1	26x26x256	26x26x256
19 route	18		1/2	26x26x128
20 conv	128	3x3/1	26x26x128	26x26x128
21 conv	128	3x3/1	26x26x128	26x26x128
22 route	21 20			26x26x256
23 conv	256	1x1/1	26x26x256	26x26x256
24 route	18 23			26x26x512
25 max		2x2/2	26x26x512	13x13x512
26 conv	512	3x3/1	13x13x512	13x13x512
27 conv	256	1x1/1	13x13x512	13x13x256
28 conv	512	3x3/1	13x13x256	13x13x512
29 conv	18	1x1/1	13x13x512	13x13x18
30 yolo				
31 route	27			13x13x256
32 conv	128	1x1/1	13x13x256	13x13x128
33 up	2x		13x13x128	26x26x128
34 route	33 23			26x26x384
35 conv	256	3x3/1	26x26x384	26x26x256
36 conv	18	1x1/1	26x26x256	26x26x18
37 yolo				

same class as the letter ”O”. Then using this coupled with fixed rules of the LP character sequence, they would choose whether it is a zero or ”O” after the predictions have happened based on how the LP characters are sequenced for that country in the dataset. For example, if the first three characters of an LP in a certain country is said to always be letters, so any digit zero predicted for the first three characters will be classified as the letter ”O”. However, as discussed earlier, having such fixed rules is not ideal, and will breakdown in certain cases, such as foreign LPs or custom LPs, it is not generalised. So here, all digits and letters are considered in the alphabet as their own specific class. This allowed for no fixed post processing at all during inference and allows for a general ALPR system.

From early baseline experiments carried out, it was found out that there was only a few characters that were performing poorly, specifically, characters with an average

Table 4 LP recognition model architecture.

Layer	Filters	Size/Strd	Input	Output
0 conv	32	3x3/2	352x128x3	176x64x32
1 conv	64	3x3/2	176x64x32	88x32x64
2 conv	64	3x3/1	88x32x64	88x32x64
3 route	2		1/2	88x32x32
4 conv	32	3x3/1	88x32x32	88x32x32
5 conv	32	3x3/1	88x32x32	88x32x32
6 route	5 4			88x32x64
7 conv	64	1x1/1	88x32x64	88x32x64
8 route	2 7			88x32x128
9 max	0	2x2/2	88x32x128	44x16x128
10 conv	128	3x3/1	44x16x128	44x16x128
11 route	10		1/2	44x16x64
12 conv	64	3x3/1	44x16x64	44x16x64
13 conv	64	3x3/1	44x16x64	44x16x64
14 route	13 12			44x16x128
15 conv	128	1x1/1	44x16x128	44x16x128
16 route	10 15			44x16x256
17 max		2x2/2	44x16x256	22x8x256
18 conv	256	3x3/1	22x8x256	22x8x256
19 route	18		1/2	22x8x128
20 conv	128	3x3/1	22x8x128	22x8x128
21 conv	128	3x3/1	22x8x128	22x8x128
22 route	21 20			22x8x256
23 conv	256	1x1/1	22x8x256	22x8x256
24 route	18 23			22x8x512
25 max		2x2/2	22x8x512	11x4x512
26 conv	512	3x3/1	11x4x512	11x4x512
27 conv	256	1x1/1	11x4x512	11x4x256
28 conv	512	3x3/1	11x4x256	11x4x512
29 conv	123	1x1/1	11x4x512	11x4x123
30 yolo				
31 route	27			11x4x256
32 conv	128	1x1/1	11x4x256	11x4x128
33 up		2x	11x4x128	22x8x128
34 route	33 23			22x8x384
35 conv	256	3x3/1	22x8x384	22x8x256
36 conv	123	1x1/1	22x8x256	22x8x123
37 yolo				

precision (AP) below 0.95. The low performing characters (LPC) were all letters, and they were [”G”, ”K”, ”M”, ”O”, ”Q”, ”S”]. To increase the AP of the LPC, the number of samples that include any LPC were increased by using data permutations and data generation.

3.6.1 Data permutations

In this method, every LP that contained any of the LPC, was duplicated by replacing other numbers or other letters that were not the LPC, with LPC characters. This is illustrated in figure 2. This is done by using the annotated BB of each character and replacing it with the corresponding LPC patch. The digit one, was not replaced as it made the patches of the other letters resize into a narrow vertical patch that would distort the character, which caused the model to get confused and perform poorly.

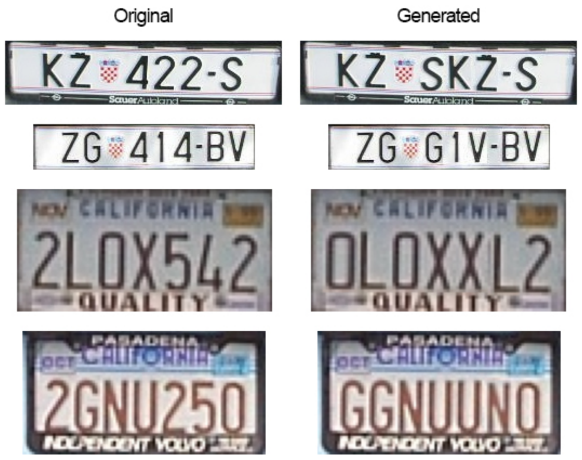

Fig. 2 Data generation using permutations

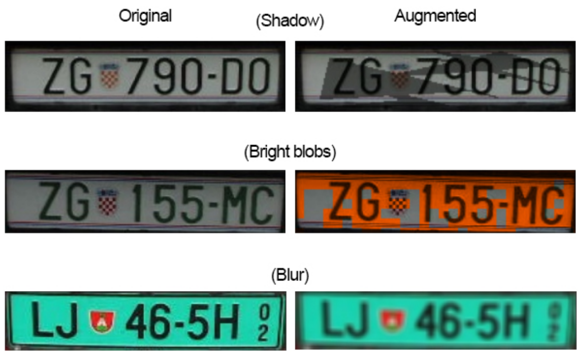

Fig. 3 Data generation techniques

3.6.2 Data generation

With Data generation, all LP samples that contained the LPC were doubled by using three different augmentation techniques to imitate certain natural changes that might happen to the LP under certain circumstances. This is illustrated in figure 3. The first is generating artificial shadow and placing it randomly over the LP patch. This will change the overall look of some of the characters, and this situation can also be met in practise. The second is adding a colour that is similar to the sun but more importantly is adding a variation to the LP that will force the model to learn to ignore, making it more generalisable for real-world scenarios. The third method is adding random blur, which is to replicate bad camera angles, speeding vehicles and such. So for each LP patch sample that included the LPC, each of these three techniques had an equal chance of being applied.

Using these two methods increased the number of samples by 2381, and all those samples included the LPC, which are in addition to the doubling of samples of using the negative image of each sample, making a total of 16,961 samples. Figure 4 shows

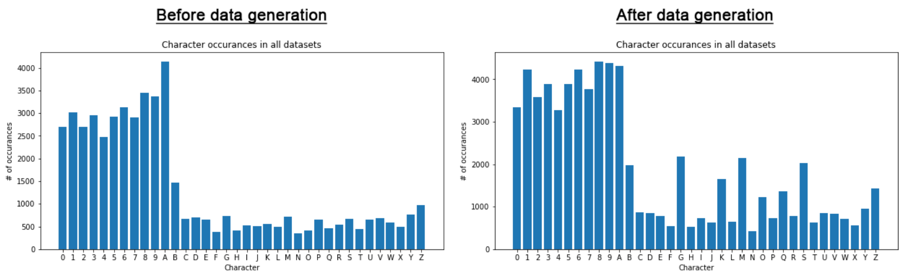

Fig. 4 The total number of character occurrences for all datasets before and after data generation.

two histograms of all the character counts before and after the data generation, we can clearly see that each LPC has a significant increase in the number of total characters, giving the model more samples to train on, hence improving the accuracy for those characters. This turned out to be very useful as we will see in section 4.

4 Results

Since some papers only partially considered the ALPR pipeline, each stage will be evaluated and compared separately in this section. Only the UFPR ALPR dataset had fixed training, validation, and testing sets. For the rest of the datasets, they were split using 0.7, 0.2, and 0.1 ratios for training, validation and testing sets respectively. To ensure there was no bias in selecting the sets, each experiment was carried out using five different random splits. All results are obtained using the test sets average results across the five splits.

4.1 Vehicle detection results

As we can see from table 5, the VD stage is not an issue, a very impressive accuracy can be achieved across all datasets, not open to much improvements. What should be noted, is that the slight precision loss is only due to false positive vehicles detected that did not have an LP visible (apart from AOLP). So all vehicles that should have been detected were detected, just with some extra vehicles detected in the background, which did not have an LP visible. A confidence and IoU threshold of 0.5 and 0.25 was used respectively.

Table 5 Vehicle detection results (%).

Dataset	Precision	Recall	Avg IoU
Caltech cars	100.00	100.00	96.84
English LP	99.88	100.00	94.98
AOLP	98.96	99.52	94.27
Open ALPR EU	100.00	100.00	95.30
UFPR ALPR	99.50	100.00	90.35
Average	**99.71**	**99.90**	**94.35**

4.2 Vehicle type classification results

Trucks and emergency vehicles are just used as example classes, this stage can be expanded to many more classes and data, even identifying for example vehicle make, model, year, colour, etc.

In table 6, the results are shown. All samples used are made public and can be accessed through the repository. The results are all from the average test sets across the five different splits. As we can see, a very high accuracy of 98.22% is achieved. This stage can easily be expanded by just adding more annotations to the samples.

Table 6 Vehicle type classification test set results.

Accuracy (%)	**Loss**
98.22	0.1130

In figure 5, we can see some samples of the two main classes.

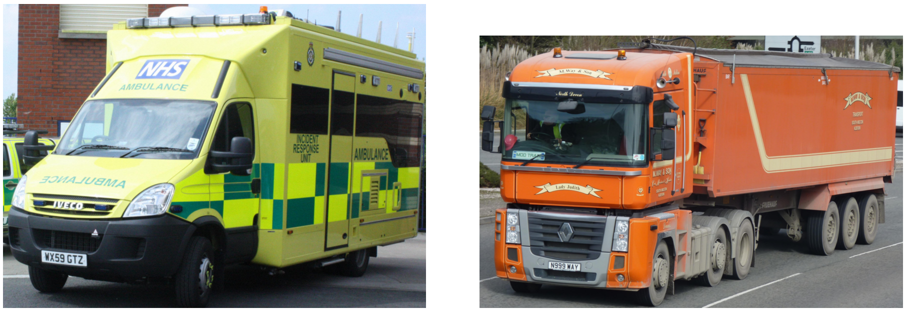

Fig. 5 Sample images of an emergency vehicle and a truck

4.3 LP detection results

In table 7, we can see that the average recall for the LP detection stage across all datasets is above 99%, which is certainly an acceptable performance. A confidence threshold of 0.75 and IoU threshold of 0.5 was used.

Table 7 LP detection results (%).

Dataset	**Precision**	**Recall**	**Avg IoU**
Caltech cars	100.00	99.19	86.72
English LP	99.61	99.21	83.70
AOLP	99.43	99.67	86.26
Open ALPR EU	100.00	99.07	85.54
UFPR ALPR	96.78	98.67	83.52
Average	**99.16**	**99.36**	**85.15**

As we can see from the results, the first two stages of the ALPR is not a problem, a very high accuracy can be achieved consistently throughout multiple datasets. However, there could be improvements on the average IoU.

4.4 LP recognition results

In this section, we will focus on the results specifically of the last stage, the LPR stage, so skipping the first two stages to isolate this stage. In the next section, we will see the results of the full ALPR pipeline. So here, all LP patches of all datasets are considered, this is to evaluate only the LP recognition stage.

Table 8 LP recognition results without the first two stages (%).

Dataset	**Precision**	**Recall**	**Avg IoU**
Caltech cars	100.00	98.98	90.42
English LP	99.91	99.87	93.16
AOLP	99.94	99.87	89.38
Open ALPR EU	100.00	98.66	91.30
UFPR ALPR	98.57	91.08	85.57
Average	**99.68**	**97.69**	**89.97**

Table 8, shows the results, a confidence threshold of 0.75 and IoU threshold of 0.5 was used. We can see that we get pretty impressive results across all datasets, apart from the UFPR ALPR dataset, however, the reason why this is will be discussed in the next section.

4.5 Full ALPR pipeline results

Here we see the results of the full ALPR pipeline. A correct LP recognition is only considered if all stages in the pipeline was successful and all LP characters are correctly detected. So if for example, a vehicle or an LP is not detected and does not go to the next stage(s), it is considered as a wrong sample.

As we can see from table 9, without any post processing to the LP patches or any fixed rules based on prior knowledge of the LPs after predictions, highly accurate results are achieved across all datasets. Each crop at each stage is directly fed into the next stage the same way it was detected. Note that for the Caltech cars, English LP, AOLP, and Open ALPR EU, we have relatively small number of FN, and in some cases where the test set is so small, this makes a huge impact on the final recall when it really is just one vehicle or LP that is not detected correctly.

The UFPR ALPR is clearly the more challenging dataset, however, it has to be noted that the UFPR ALPR dataset is made up images from a video. For example, you would have 30 images from the same video with slight differences as the vehicle is moving. The UFPR ALPR test set is made up of 60 videos with each having a total of 30 frames. In practise, when you have a video stream like that, you will have many different opportunities (frames) to correctly detect the full LP, as the car moves across the frame, and you can for example consider a 100% confidence recognition if the same LP for the same vehicle has been detected three times in a row (3 frames in

Table 9 Full ALPR pipeline results. Please note these results are chosen from one of the 5 test sets just to include the exact TP and FN to show how significant only 1 wrong sample in the relatively small test sets can be in some datasets. All other test sets had very similar results.

Dataset	**Stage**	**TP**	**FN**	**Recall**
Caltech cars	VD	14	0	100
	LPD	13	1	92.86
	LPR	13	1	92.86
English LP	VD	52	0	100
	LPD	50	2	96.15
	LPR	50	2	96.15
AOLP	VD	218	1	99.54
	LPD	216	3	98.63
	LPR	214	5	97.72
Open ALPR	VD	12	0	100
	LPD	12	0	100
	LPR	12	0	100
UFPR ALPR	VD	1800	0	100
	LPD	1769	31	98.28
	LPR	1117	683	62.06
Average LPR				**89.56**
UFPR ALPR as vid	LPR	44	16	73.33

Table 10 ALPR comparison to other methods across all datasets used. *The first two stages of the pipeline was skipped. **When considering the UFPR ALPR dataset as a video.

Method / Dataset	[18]	[1]	[19]	[20]	[21]	OpenALPR	[7]	Proposed
Caltech cars	-	-	-	-	95.7 ± 2.7	99.1 ± 1.2	98.7 ± 1.2	97.1
English LP	97.0	-	-	-	92.5 ± 3.7	78.6 ± 3.6	95.7 ± 2.3	95.5
AOLP	-	99.8*	-	-	87.1 ± 0.8	-	99.2 ± 0.4	98.0
Open ALPR EU	-	-	93.5	85.2	93.5	91.7	97.8 ± 0.5	98.7
UFPR ALPR	-	-	-	-	62.3	82.2	90.0 ± 0.7	62.1 (73.3**)
Average	-	-	-	-	87.8 ± 2.4	90.7 ± 2.3	96.9 ± 1.0	90.3

a row). Meaning the rest of the frames where the vehicle is in frame is not significant as you only need for example 3 consecutive frames for the LP to be successfully recognised. If we treat the UFPR ALPR dataset in this way, as in practical settings, from the bottom of table 9 (UFPR ALPR as vid), we can see the recognition result is significantly higher as we do not need to detect the LP of the vehicle on every frame if we have already detected it correctly 3 times in 3 consecutive frames.

We can see that the full ALPR pipeline, all stages are performing very well, and although the last stage is definitely the least performing, it is still producing very good results without assuming any prior knowledge on the LP across all datasets and considering all characters of the alphabet as their own class.

4.6 Comparison

In this section, we compare the results of proposed systems alongside the results for this work, across 5 datasets. Table 10 summarises all the results.

Table 11 All characters average precision (AP) across all datasets in the test sets combined.

C	**AP(%)**	**TP**	**FP**	**C**	**AP(%)**	**TP**	**FP**
0	98.26	724	22	**I**	98.90	94	0
1	99.65	844	4	**J**	100.00	145	0
2	100.00	497	0	**K**	83.86	115	0
3	99.51	645	0	**L**	100.00	146	0
4	99.93	811	16	**M**	86.50	165	23
5	100.00	758	0	**N**	100.00	48	0
6	99.94	922	6	**O**	38.96	35	28
7	99.17	669	15	**P**	99.99	282	0
8	98.72	1037	15	**Q**	70.84	43	1
9	99.66	1103	8	**R**	99.98	148	2
A	98.82	1494	2	**S**	96.47	228	0
B	97.98	363	0	**T**	100.00	112	0
C	97.64	133	3	**U**	100.00	128	0
D	99.53	99	13	**V**	100.00	171	0
E	97.59	136	6	**W**	95.70	229	1
F	99.06	40	1	**X**	98.69	107	0
G	95.68	147	0	**Y**	99.95	274	0
H	99.96	109	0	**Z**	100.00	280	0

As we can see, without any processing at inference or relying on any fixed rules in a streamlined pipeline, similar and comparable results are achieved while still considering all characters as their own class, 0-9 and A-Z. Table 11 shows the results for each character.

We can clearly see that there are only a few characters that are performing poorly (AP less than 95), which are the [”K”, ”M”, ”O”, ”Q”] characters, which for those letters, it is very understandable why they would be very hard and challenging for a model to distinguish between. However, with more data, and more data generation techniques, the accuracy of these characters can be increased and a full generic ALPR system can be achieved.

4.7 Performance Evaluation

Table 12 shows the time it takes to perform each stage in seconds for when there are 1, 2, and 3 vehicles in the frame where the LP is visible. It also shows the total FPS it takes to process the whole frame for all stages. It has to be noted that, these performance results are not a fair comparison to other methods as a very low-end GPU (NVIDIA GTX 1060) was used in these experiments, whereas most other methods used a very high-end expensive GPU. However, the fact that we achieve this FPS even when using a low end GPU shows promising results in terms of performance. We believe, if the same high-end GPUs was used, this ALPR pipeline can easily achieve real-time performance.

5 Conclusions

We have presented a method to perform the ALPR task in a fully automated and stream-lined pipeline which includes all the stages, including a vehicle classifier,

Table 12 The FPS for the full ALPR pipeline, as well as the processing time in seconds for each stage of the pipeline for when there are 1, 2, and 3 vehicles in the frame.

# Vehicles / **Stage**	**1**	**2**	**3**
Vehicle detection	0.0349	0.0389	0.0449
LP detection	0.0080	0.0150	0.0239
LP recognition	0.0120	0.0239	0.0239
Total FPS	18	13	11

without any prior knowledge of the LP or post or fixed rules. Multiple data generation techniques was developed to increase data samples, and competitive results were achieved when compared to other latest methods tackling the ALPR problem, showing promising results in terms of accuracy and performance for an end-to-end ALPR system in 5 different publicly available datasets. We have also made this methodology and results all open-source for anyone to use and/or contribute to. Only a few characters are left that are poorly detected, with slightly more data, a very robust and general ALPR system is possible. This method is a step towards a complete and fully automated ALPR system that works with any type of LP regardless of where it is from or its layout, which will be essential in a complete future ITS system.

Acknowledgements

This work has been supported by the QR Global Challenges Research Fund (GCRF), SRO03, UK.

Conflicts of Interest

The authors have no conflicts of interest to declare that are relevant to the content of this article.

References

[1] Zhuang, J., Hou, S., Wang, Z., Zha, Z.-J.: Towards human-level license plate recognition. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 306–321 (2018)

[2] Xie, L., Ahmad, T., Jin, L., Liu, Y., Zhang, S.: A new cnn-based method for multi-directional car license plate detection. IEEE Transactions on Intelligent Transportation Systems **19**(2), 507–517 (2018)

[3] Li, H., Wang, P., You, M., Shen, C.: Reading car license plates using deep neural networks. Image and Vision Computing **72**, 14–23 (2018)

[4] Kessentini, Y., Besbes, M.D., Ammar, S., Chabbouh, A.: A two-stage deep neural network for multi-norm license plate detection and recognition. Expert systems with applications **136**, 159–170 (2019)

[5] Kurpiel, F.D., Minetto, R., Nassu, B.T.: Convolutional neural networks for license plate detection in images. In: 2017 IEEE International Conference on Image Processing (ICIP), pp. 3395–3399 (2017). IEEE

[6] Montazzolli, S., Jung, C.: Real-time brazilian license plate detection and recognition using deep convolutional neural networks. In: 2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), pp. 55–62 (2017). IEEE

[7] Laroca, R., Zanlorensi, L.A., Gonçalves, G.R., Todt, E., Schwartz, W.R., Menotti, D.: An efficient and layout-independent automatic license plate recognition system based on the yolo detector. arXiv preprint arXiv:1909.01754 (2019)

[8] Vision, C.: Caltech Cars. http://www.vision.caltech.edu/archive.html (2021)

[9] Srebrić, V.: English LP. http://www.zemris.fer.hr/projects/LicensePlates/english/results.shtml (2021)

[10] Hill, M.: Open ALPR. https://github.com/openalpr/benchmarks/tree/master/endtoend/eu (2021)

[11] Hsu, G.-S., Chen, J.-C., Chung, Y.-Z.: Application-oriented license plate recognition. IEEE transactions on vehicular technology **62**(2), 552–561 (2012)

[12] Laroca, R., Severo, E., Zanlorensi, L.A., Oliveira, L.S., Gonçalves, G.R., Schwartz, W.R., Menotti, D.: A robust real-time automatic license plate recognition based on the yolo detector. In: 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1–10 (2018). IEEE

[13] Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779–788 (2016)

[14] Bochkovskiy, A., Wang, C.-Y., Liao, H.-Y.M.: Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934 (2020)

[15] Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Kolesnikov, A., Duerig, T., Ferrari, V.: The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. IJCV (2020)

[16] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)

[17] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár,

P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European Conference on Computer Vision, pp. 740–755 (2014). Springer

[18] Panahi, R., Gholampour, I.: Accurate detection and recognition of dirty vehicle plate numbers for high-speed applications. IEEE Transactions on intelligent transportation systems **18**(4), 767–779 (2016)

[19] Silva, S.M., Jung, C.R.: License plate detection and recognition in unconstrained scenarios. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 580–596 (2018)

[20] Silva, S.M., Jung, C.R.: Real-time license plate detection and recognition using deep convolutional neural networks. Journal of Visual Communication and Image Representation **71**, 102773 (2020)

[21] Masood, S.Z., Shu, G., Dehghan, A., Ortiz, E.G.: License plate detection and recognition using deeply learned convolutional neural networks. arXiv preprint arXiv:1703.07330 (2017)