IET Intelligent Transport Systems

Research Article

IET Journals
The Institution of Engineering and Technology

EDF-LPR: a new encoder–decoder framework for license plate recognition

ISSN 1751-956X
Received on 28th April 2019
Revised 20th April 2020
Accepted on 4th May 2020
E-First on 30th June 2020
doi: 10.1049/iet-its.2019.0253
www.ietdl.org

Fei Gao[1] ✉, *Yichao Cai*[1], *Yisu Ge*[1], *Shufang Lu*[1]

[1]*Department of Computer Science, Zhejiang University of Technology, Hangzhou, People's Republic of China*

✉ *E-mail: feig@zjut.edu.cn*

Abstract: Although automatic license plate recognition (ALPR) has been studied for decades, the final recognition result can be accurate only if the license plate is detected and the standard format is unambiguous. However, since an image may contain license plates with different formats and scales, license plate detection and standard format classification may fail. In this study, a new ALPR codec framework named EDF-LPR is presented. As for the encoder, at the first stage, candidate license plate characters are detected and recognised directly without considering the format of license plate, and candidate regions of characters are extracted by density-based spatial clustering of applications with noise-like algorithm; at the second stage, poor regions are processed by tilt correction and scale normalisation to obtain more accurate candidate characters. As for the decoder, a sequence learning model is trained to convert each unordered coded sequence into a sequence composed of marks that indicate a way to construct the final result string. Experiments are designed to evaluate the performance of EDF-LPR on both detection rate and recognition rate. The experimental results on public datasets show that the detection rate and recognition rate are 99.51 and 95.3%, respectively, at about 40 fps.

1 Introduction

Automatic license plate recognition (ALPR) is a classical but critical problem. With the development of deep learning, a convolutional neural network (CNN) [1] has been widely applied to ALPR and performs well. However, according to recent surveys and reviews [2–5], the challenges from unrestricted environments and variations of license plates are still puzzling. Moreover, Zhao *et al.* [6] recently evaluated five Chinese commercial ALPR systems on their dataset and found that the average accuracy decreased from claimed 99 to 75–92%. It was found that the decreasing accuracy was affected by the performance of both license plate detection and standard format classification.

License plate detection is the first step of almost all ALPR methods [7]. Traditional ALPR algorithms followed the so-called three-stage pipeline, which includes license plate detection, character segmentation, and character recognition. Since license plate detection affects the final recognition accuracy, most of the current state-of-the-art deep learning-based ALPR methods modified the pipeline. Naturally, there are two interesting questions: does license plate detection have to be the first step? How to improve detection performance?

Can synchronised character detection and recognition be the first step? As is known to all, a license plate is composed of characters. In the past, it was hard to detect and recognise characters directly since they are so small. At present, deep neural network (DNN) has managed to detect small objects. Since it is difficult to detect all license plate characters successfully, the license plate can just be roughly located when at least one character is detected. However, taking synchronised character detection and recognition as the first step of ALPR is still worth trying.

License plate standard format classification is another important concern, on which the relationship between license plate image and its processing algorithm is established. Commercial ALPR systems usually set up a series of heuristic rules to handle different kinds of license plates. However, when new standard formats are published, both heuristic rules and recognition algorithms need to be updated. Obviously, it is inflexible. Recent end-to-end ALPR methods showed that standard classification is not a necessary step. The latest encoder–decoder-based DNN models [8–11] are flexible, but can only handle certain fixed types of license plates. Besides, the codec mechanism autonomously learned by the black box is not reliable enough because it is difficult to explain.

From a systematic point of view, the interpretability of the model is very important. According to [12], analogies and associations are used to generate predictions by a human. With reference to human behaviours, the license plate of the new standard format can be recognised easily when the characters are identifiable. There is a reasonable bottom-up two-stage license plate recognition pipeline, i.e. recognising characters at the first stage and sorting characters into strings at the second stage. However, computers cannot detect and recognise all characters accurately. In particular, false positives and false negatives are unavoidable.

Inspired by the progress made in related work, in this study, a new bottom-up encoder–decoder framework, called EDF-LPR, is proposed to complete the above-mentioned pipeline. Referring to popular design ideas presented in [13, 14], the key components of the proposed framework can be easily extended or replaced. The encoder is responsible for synchronised detection and recognition of license plate characters, and the decoder determines how to filter and sort the characters into a sequence according to the position and confidence level of each candidate character. In the encoder, a recognition strategy is designed to ensure the quality of encoding. Specifically, regardless of the license plate format, the encoder is completed in two stages. At the first stage, all candidate characters in the image are detected and recognised directly by a multi-scale character detection model, while candidate regions of characters can be extracted by density-based spatial clustering of applications with noise (DBSCAN)-like algorithms. After tilt correction and scale normalisation, poor regions are focused on a single-scale character detection model to obtain more accurate candidates at the second stage. Therefore, each license plate in the image is encoded into a variable-length sequence that consists of 'synthetic element'. Each 'synthetic element' is a tuple of the bounding box, confidence level, and character label. In the decoder, a sequence-to-sequence [15] model is trained to predict a rule to filter the false positives and sort the true positives. It is noteworthy that character labels are removed and bounding boxes are normalised before sending to the sequence-to-sequence model. The predicted rule is composed of semantic marks and has the same length as the input sequence.

There are totally three kinds of semantic markers: drop-out, valid-index, and end-of-sequence.

The proposed framework is proved to be robust, flexible, accurate, and efficient. Both license plate detection rate and recognition rate are important indexes for accuracy verification, running time reflects efficiency, and actual performance on license plates with different standard formats and scales shows robustness and flexibility. To be specific, robustness and flexibility are results of qualitative analysis, which can be drawn from quantitative analysis of accuracy and efficiency. Thus, on the one hand, three public datasets, i.e. NTUA [1], application-oriented license plate (AOLP) [16], and PKU-Vehicle [17] are tested for evaluating the detection rate and running time. On the other hand, two public datasets, i.e. AOLP and OpenITS [6] are checked for measuring the recognition rate and running time. Besides, to verify the stability of the proposed framework, popular object detection models, e.g. you only look once v3 (YOLOv3) [18], Faster region-CNN (R-CNN) [19], and single shot detector (SSD) [20] are chosen to be the specific DNNs in the experiments.

The main contributions can be summarised as follows:

(i) A new encoder–decoder-based framework named EDF-LPR is proposed to unify ALPR in different standard formats and scales, which converts the issue of ALPR to a sequence processing task with a distinctive codec mechanism. EDF-LPR is very generic for detecting and recognising license plates with different formats since no specific DNN model needs to be specified.
(ii) The encoder is well designed to accept a wide variety of inputs by following the idea of recognition. The encoder can not only improve the detection rate but also save the recognition time by processing the bottom character information directly.
(iii) The decoder manages to automatically filter and sort candidate characters from encoded sequences of different types of license plates. Sequence-to-sequence learning is applied to ALPR in this form for the first time.

It should be noted that the proposed encoder–decoder framework is quite different from those pure DNN-based models. In some previous encoder–decoder frameworks, it is common to encode the image into fixed dimension feature vectors, and then decode feature vectors into the final results. Although these frameworks seem simple and great, it is not enough for ALPR. Since the processing of small-scale license plate needs a DNN model with a large input size, which means an increase of computation at the same time. Moreover, a small-scale license plate stresses the importance of recognition. Though the proposed framework seems a little complex or may be thought shallow, the encoder is no longer mysterious or hard to explain, and the decoder is flexible. Besides, better performance can be achieved when more excellent character detection DNN is applied.

The rest of this paper is organised as follows: a brief review of related work is discussed in Section 2; in Section 3, the proposed framework including the principle and implementation method is presented in detail; Section 4 shows a series of experimental results on four public datasets; finally, the conclusion is drawn in Section 5.

2 Related work

Over the last two decades, impressive progress has been made. Both the three-stage methods and end-to-end methods are summarised up in the following sub-sections.

2.1 Three-stage ALPR

License plate detection is a key stage for ALPR, and many works are focused on it. Existed methods can be classified into three categories: image processing based, pattern recognition based, and machine learning based. Firstly, there are many image processing-based methods. As the colour and edge of the license plate are different from that of the neighbour, multi-wavelet transform, colour space converting, and morphological operations were explored in [21–23], respectively. There is no doubt that these methods are susceptible to imaging conditions. Secondly, pattern recognition plays an important role in license plate detection. Dun *et al.* [24] and Ashtari *et al.* [25] detected license plates by mask templates. Since characters on the license plate are representative, Lim and Tay [26] and Hsu *et al.* [16] made use of maximally stable extremal regions (MSER) [27] to extract candidate regions of license plate. Some manual features such as scale-invariant feature transform and local binary pattern were used to extract license plates in the image [28, 29]. However, MSER-based methods are time-consuming, and manual features are not robust. Thirdly, machine learning-based methods represent the most advanced level. Exemplar-support vector machine (SVM) and Faster R-CNN were employed by Rafique *et al.* [30]. Molina-Moreno *et al.* [31] applied deformable part-based model on license plate detection. Although machine learning performs well, a common problem is lacking training data. Besides, prior knowledge was also used to improve the detection performance, such as background modelling in [32] and temporal redundancy in [33].

Character segmentation is the following stage after a single license plate is detected. At this stage, there are mainly three kinds of methods to extract foreground characters with representative colour and edge. First of all, image processing is widely used. Joint horizontal and vertical projection is a common way to segment characters in the binary image. Besides, super-pixel technology was applied to character segmentation in [34]. In [35], blue–green–red colour space was converted to LAB to enhance the contrast between the foreground and background. Hou *et al.* [36] applied a stroke width transform to segment characters. Secondly, pattern recognition appears to be effective as well. Characters were segmented by a template in the method proposed by Povolotskiy *et al.* [37]. However, it would fail when a wrong template was chosen. Extremal regions based on edges were chosen as the candidates in [16, 38]. Thirdly, those methods based on machine learning regarded multiple characters as one class and used object detection models to complete segmentation. Besides, different algorithms were combined. A two-stage segmentation method was proposed in [39], which combined template matching and histogram projection to finish character segmentation.

Character recognition is the third stage in the three-stage ALPR. The extracted characters are usually recognised one by one through pattern recognition and machine learning. A hybrid K-nearest neighbour–SVM model was trained in [40]. Taking advantage of the robustness of CNN-based features, Yang *et al.* proposed a deep architecture named CNN-KELM for Chinese license plate classification [41]. Although this task is well understood, the actual effect is not always satisfactory, because license plates have different formats, scales, and angles.

In general, although deep learning technology can enhance the performance of each stage independently, there is an obvious redundancy between different stages, and the cumulative error is still large.

2.2 End-to-end ALPR

End-to-end methods are proposed to make up for the shortcomings of three-stage solutions. In fact, license plate detection, character segmentation, and character recognition are not mutually independent. Thus, segmentation-free methods have been implemented in the latest literature. These methods can be divided into two categories: top-down approach and bottom-up approach.

Top-down approach focuses more on the license plate itself. Owing to the limitation of structure, DNN with eight branches of fully connected layers that are built in [8] can recognise license plates with no more than eight characters. In addition, Kim *et al.* [42] proposed a multi-task DNN to recognise the license plate. Intel proposed an efficient neural network called LPRNet [43], which could handle the ALPR problem in <3 ms. In these methods, recognition can be completed in an end-to-end way, but the license plate must be detected in advance.

On the contrary, the bottom-up approach focuses more on the characters. Most recently, ALPR has been handled as a sequence processing problem. Guo *et al.* [44] proposed a hybrid CNN–hidden Markov model (HMM) model. Bulan *et al.* [45] also

applied the Viterbi algorithm to HMM. Bidirectional recurrent neural network (BRNN) and connectionist temporal classification (CTC) were jointly used to recognise an extracted license plate in [9–11]. Wang *et al.* [9] combined spatial transformer network, CNN, BRNN, and CTC to recognise rotated single-row license plate with arbitrary length. Li *et al.* [10] proposed a complete end-to-end neural network to detect and recognise simultaneously. Although their method had been evaluated on four datasets and performed well, there were still some problems, e.g. incapable of dealing with double-row license plates and poor real-time performance.

In summary, end-to-end ALPR methods break through the bottleneck of convolutional three-stage ALPR algorithms. According to the latest comparative study of ALPR techniques [7], although accuracy and efficiency have been much better, robustness and flexibility still need to be improved. Specifically, most of the above-mentioned methods can recognise license plates in an end-to-end manner only if the license plate is detected in advance. Moreover, it is hard for most of them to deal with a variety type of license plate.

2.3 Most relevant work

According to [6], license plate detection and standard format classification are bottlenecks of ALPR. Nowadays, object detection neural networks are committed to improving detection performance. For example, real-time object detection framework YOLOv2 [46] was used to detect vehicles, license plate, and characters in [47–51]. Recently, YOLOv3 [18] achieved better performance on small target detection. Even though license plates are extracted, the success of standard format classification is another premise. Concrete heuristic rules have to be made for license plates with different formats. For example, character recognition network (CR-NET) was applied for character segmentation and recognition, and different heuristic rules were used to filter the final results in [47, 48]. The reason why commercial ALPR systems can deal with different types of license plates is that they integrate various license plate recognition algorithms suitable for different standard formats.

From the perspective of practical application, a pure algorithm is difficult to meet all the requirements of ALPR in robustness, flexibility, accuracy, and efficiency. Unrestricted environments and variations of license plates are challenging. Since ALPR is gradually treated as a sequence processing problem, experiences from natural scene text recognition are valuable. As encoder–decoder architecture is becoming more and more popular, an image can be converted into a description statement in [52]. ALPR is a special case of image caption, where only information on a license plate is valuable. Sequence-to-sequence learning [13] has succeeded in automatic language translation, speech recognition, and other sequence processing problems. It is reasonable and feasible that spatial relationships among license plate characters can be learned.

After witnessing the powerful codec architectures based on deep neural networks, the performance of pure ALPR algorithm is expected to make a breakthrough. With the advancement of object detection, a character-based workflow that was considered impossible in the past has become a reality. Modified SSD and mask R-CNN were used for character recognition in [53, 54], respectively. Since they paid little attention to the license plate standard format, it is hard to achieve multinational license plate recognition based on recognised characters. Moreover, if the license plate is detected successfully, false positives and negatives remain to be solved; otherwise, these methods are useless. Therefore, in this study, EDF-LPR adopts a recognition strategy to guarantee character recognition result of license plates with different scales and employs sequence-to-sequence learning to handle different formats in a uniform way. Especially, sequence-to-sequence learning is used to replace the original inflexible artificial heuristic rules in the decoder of EDF-LPR. Besides, Similar to [13, 14], the core components of EDF-LPR are extensible and replaceable.

3 Methodology

In this section, more information about the proposed bottom-up encoder–decoder framework will be introduced. As mentioned above, the framework is an implementation of a character level two-stage ALPR pipeline. The whole framework takes practice on the thought of divide-and-conquer and the problem of ALPR is taken apart and broken into encoder and decoder. Although not completed in end-to-end, it is well designed. The overview of the proposed framework is shown in Fig. 1.

3.1 Principle of the encoder

The encoder is responsible for synchronised detection and recognition of license plate characters. A recognition strategy is carried out to ensure that rich candidate regions are extracted from multi-scale inputs. The encoder is completed in two stages regardless of the license plate format. The encoding result of each license plate in the image is a variable-length sequence consisting of 'synthetic element'. More details are as follows.

3.1.1 First encoding stage: The task of this stage is to detect and recognise all candidate characters in the image and provide candidate regions of license plates.

A multi-scale character detection DNN model is employed to accomplish the basic task. To train such a DNN model, each character is equipped with a bounding box and an independent label instead of a uniform label. Any character, A, is manually labelled as a tuple (c, x, y, w, h), where c is the category that A belongs to, (x, y) means the centre of bounding box of A, (w, h) represents the size of bounding box of A. The DNN model takes an image I as input, and output a target sequence of characters $S = \{s_1, s_2, \ldots, s_n\}$, where s_i is a synthetic element consisting of (c, x, y, w, h, t), t means the confidence level of s_i, and n represents the number of characters. The training task can be described as maximising the probability of the target sequence by the following formulation:

$$w^* = \arg\max_{w} \sum_{I,S} \log p(S|I; w) \tag{1}$$

where w is the parameter of the model and w^* is the optimally trained weights.

Through the DNN model, the input image has been encoded into sequence S, which contains all candidate characters. These characters may come from different license plates, or they may not belong to any license plate at all. Thus, candidate regions of license plates should be determined according to the original encoded sequence S.

The DBSCAN-like algorithm is applied to extract the candidate regions. It is achieved by expanding the bounding box of each candidate character twice. More concretely, in the first expansion, the centre point Q_i of each candidate character s_i is served as an anchor point, and the bounding box of s_i is expanded to three times as it used to be. The second expansion is then applied to the minimum enclosing rectangles generated by the first expansion. Thus, the original sequence S is divided into several sub-sequences D, where D is a collection of collections of characters on each candidate region, D_i is the ith sequence in D and $i < d$, d is the count of candidate license plates.

After a license plate is encoded into a sequence, there are three common situations (for illustration purpose, it is assumed that there is only one license plate in the image):

- The ideal case as shown in Fig. 2*a*. The encoding performs very well when the input is a roughly extracted license plate and the characters are identifiable.
- The false-positive case as shown in Fig. 2*b*. When the scale becomes smaller, the characters are more confusing, and wrong recognition results appear.
- The false-negative case as shown in Fig. 2*c*. When the scale becomes much smaller, the performance of character detection

and recognition is worse. Characters are easy to be missed by the DNN model.

Apparently, the encoding process can be stopped in the ideal case. However, the false-positive case will cause trouble to the decoder, and the false-negative case means a direct failure. In fact, in addition to scale, there are many other factors that affect the quality of encoding, such as imaging angle of view, illumination change, and so on. Therefore, the above-mentioned situations should be further verified and supplemented at the second encoding stage.

3.1.2 Second encoding stage: The task of this stage is to identify encoding quality and improve the poor quality.

According to the recognition strategy, whether or not a single license plate is well encoded is important prior to that almost half of the computing time can be saved. Encoded sequence can be judged as excellent or bad by encoding quality identification. If the encoded sequence has good quality, it corresponds to a rich candidate region; otherwise, it corresponds to a poor candidate region. Specific policies are made based on the length and confidence level of the encoded sequence to complete the identification. For each sub-sequence D_i, heuristic rules are set according to (2) and (3). A sequence is a well-encoded one only if all heuristic rules are satisfied

$$N_i \geq N_{\min} \tag{2}$$

$$\frac{1}{N_i}\sum_{j=1}^{N_i} t_{ij} \geq T_{\min} \tag{3}$$

where N_i represents the amount of characters in sequence D_i, $N_{\min}$ means the minimal sequence length, t_{ij} is the jth character in D_i, $T_{\min}$ is the minimal sequence confidence level.

Following the re-cognition strategy, more attentions are paid on poor candidate regions extracted at the first stage. Since scale, tilt angle, and illumination of license plate are the main factors that lead to current problems, it is reasonable to improve the encoding performance by dealing with these factors. Since candidate regions of license plates have been extracted, scale can be considered as unified. At the same time, the influence of illumination also decreases with the increase of scale. Since the encoding process is controllable, an efficient tilt correction algorithm is proposed. More concretely, it is a method based on a weighted voting strategy. As shown in Fig. 3*a*, the row with most candidate characters is located at first, and then the skew correction matrix for affine transformation by the first and last characters of that row is calculated. It is worth noting that if the tilt angle of a license plate can be calculated easily, so if the tilt angle is small, tilt correction can be skipped.

After poor candidate regions are enhanced by tilt correction, missing characters can be found back by being encoded again, which is the so-called recognition strategy. As shown in Figs. 3*b* and *c*, because license plate characters in candidate regions are no longer the small ones, an undesirable case is converted to an ideal case. Except for the multi-scale character detection DNN model mentioned at the previous stage, there are wider choices of single-scale character detection DNN model at this stage, i.e. no specific DNN model needs to be specified, i.e. one of the reasons why EDF-LPR is generic. The same as the first stage, at this stage, the new encoding result of each poor region is a sequence consists of synthetic elements.

In general, the encoded sequences with bad quality will be filtered out, and well-decoded sequences will be sent to the decoder to finish the final recognition. These well-decoded sequences are represented as $S^* = \{S_1, S_2, \ldots, S_j\}$, where S_j is the jth sequence in S.

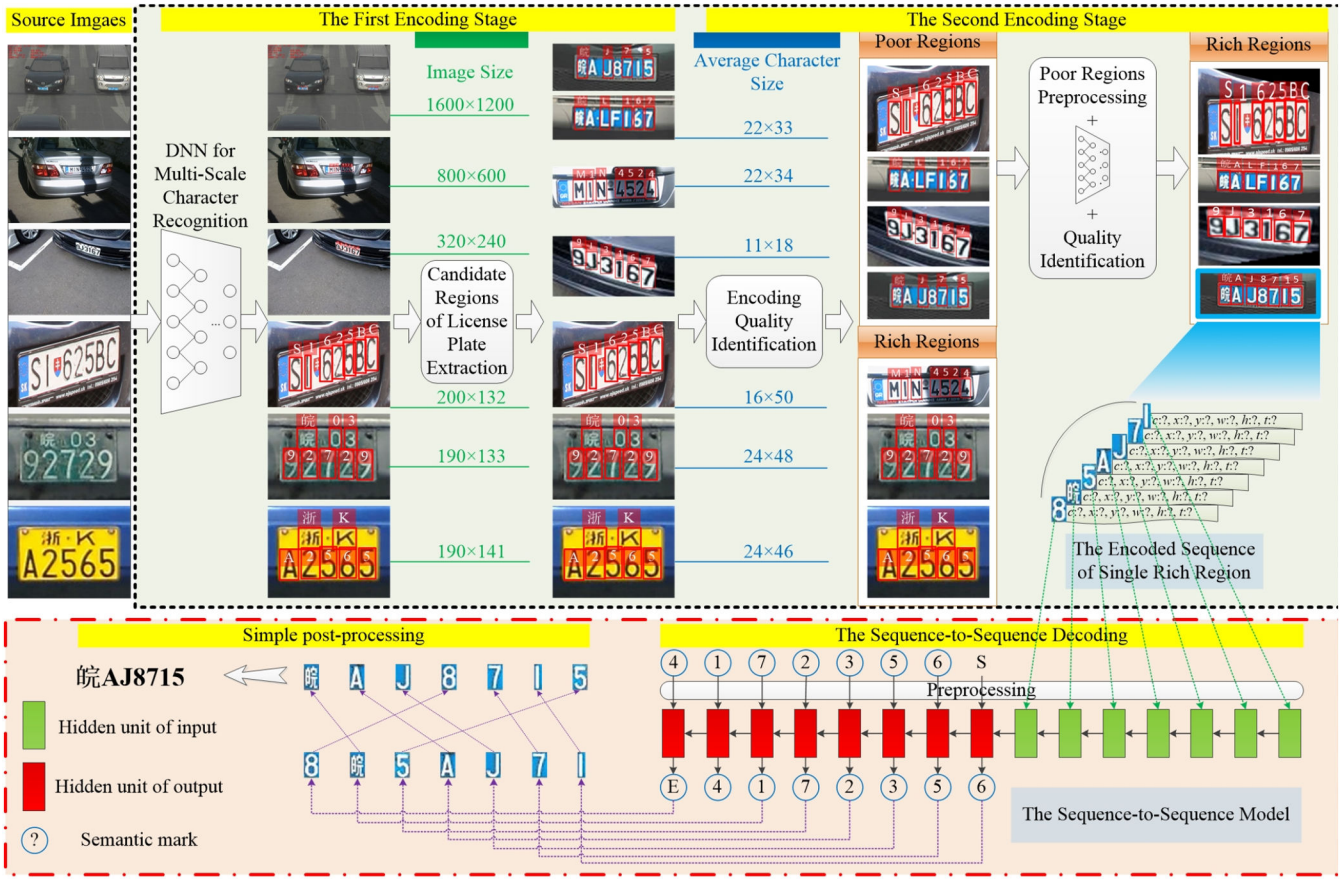

Fig. 1 *Overview of the proposed encoder-decoder framework, which converts the issue of ALPR to a sequence processing task. As shown in the Ivory tint area, the encoder is responsible for synchronised detection and recognition of license plate characters. As shown in the flesh colour area, the decoder determines how to filter and sort the characters into a sequence according to the position and confidence level of each candidate character*

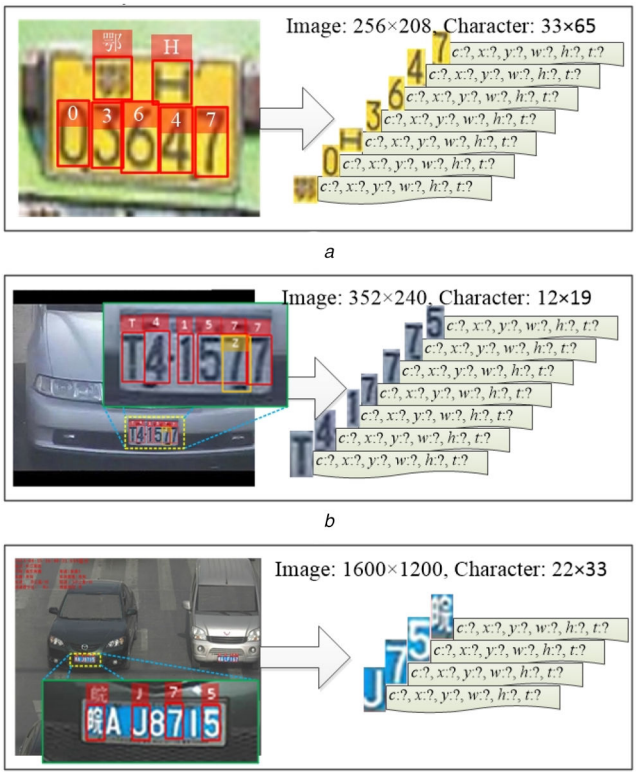

Fig. 2 *Three situations of the first encoding stage*
(a) Ideal case, ***(b)*** False-positive case, ***(c)*** False-negative case

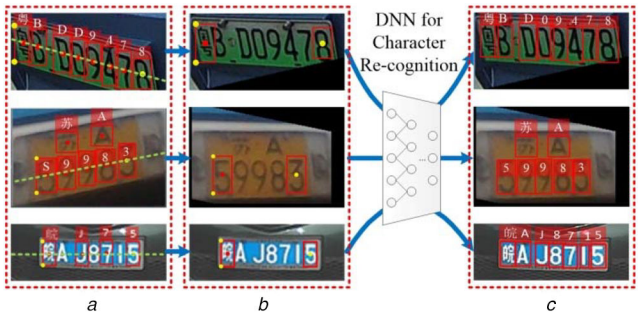

Fig. 3 *Illustration of the second encoding stage*
(a) Candidate poor regions, ***(b)*** Regions after tilt correction, ***(c)*** Encoding results after recognition

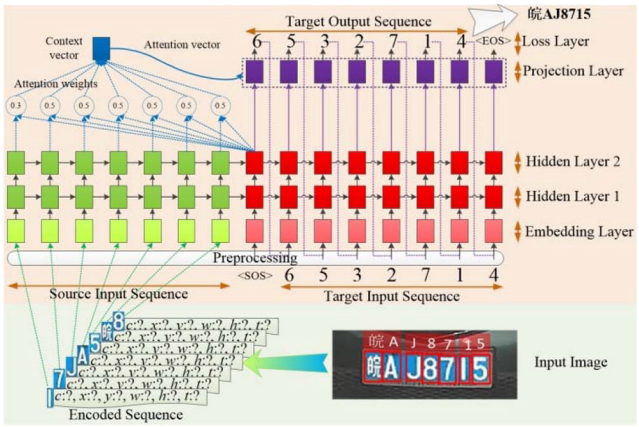

Fig. 4 *Overview of sequence-to-sequence model to translate source input sequence with flexible length into the final string*

3.2 Principle of the decoder

The decoder determines how to filter and sort the characters into result string by making full use of the encoded sequence. Since the length of the encoded sequence is not fixed, a sequence-to-sequence model is trained to decode a rule from the encoded sequence to filter the false positives and sort the true positives. The predicted rule is represented as a sequence composed of semantic marks, which has the same length as the input sequence. Semantic markers are divided into three categories: drop-out, valid-index, and end-of-sequence.

The input and output of the decoding task should be clarified above all. Given an input sequence $(x_1,\ldots, x_T)$, the goal is to get an output sequence $(y_1,\ldots, y_T)$ with the same length T to the input. The ith input x_i is exactly a synthetic element consists of (c, x, y, w, h, t) in the encoded sequence. In addition, there is a special synthetic element (0, 0, 0, 0, 0, 0) to define the end of sequence ('〈EOS〉'). The jth output y_j is set as a semantic mark corresponding to the input character x_j. Since the license plate usually contains no more than nine characters, at worst each character may be accompanied by a false positive one in the encoding result. Therefore, the vocabulary of semantic marks is represented as V, which is composed of 20 numbers from 0 to 19. Especially, a drop-out mark is labelled as 19; an end-of-sequence mark, '〈EOS〉', is labelled as 0; a valid-index mark is labelled as a number in the range of [1, 17]. Besides, there is also a preset '〈SOS〉' that indicates the starting of the sequence. Since '〈SOS〉' is just a symbol and does not appear in the output sequence, it is set to 0.

As shown in Fig. 4, the sequence-to-sequence model is a neural network with a preprocessing step, an attention module, and five layers, i.e. one embedding layer, two hidden layers, one projection layer, and one loss layer.

In the preprocessing step, it is necessary to understand that the input and output of the sequence model are a little different from those of the decoding task. Sequence learning is originally used in natural language processing and the relationship between words can be learned, e.g. noun, verb, and adjective. It is quite different when it is applied to ALPR, because the combination of license plate character labels is arbitrary, and the vocabulary of output is composed of semantic marks rather than words. Therefore, on the one hand, the input element x_i is converted into $(x/W, y/H, w/W, h/H, t)$, where H and W are the approximate height and width of the license plate, respectively. On the other hand, the output element y_i is normalised into one-hot coding.

In the embedding layer, source input sequence and target input sequence are processed to reinforce data relationships. The embedding layer converts element of both input and output to a vector with a fixed dimension.

In the hidden layer, long short-term memory (LSTM) is employed to learn the temporal dependence of synthetic elements, i.e. spatial relationship of characters. Compared with natural language processing tasks, the sequence of this task is shorter and the dictionary is smaller. Therefore, each hidden layer is set to contain 128 LSTM units. The goal of the LSTMs here is to estimate the conditional probability $p(y_1,\ldots, y_{T'} \mid x_1,\ldots, x_T)$ where T' is the length of output sequence. Although T' is equal to T during training, T' may be different from T while testing. This probability is calculated in three steps. First, a fixed-dimensional vector of the input sequence is obtained by the first part of the LSTM module (as shown in the green units of Fig. 4). Second, the attention module is used to obtain a weighted features vector to enhance temporal correlation, and the enhanced vector is represented as o. Third, the probability of the output sequence is calculated by the second part of the LSTM module (as shown in the red units of Fig. 4). The calculation of probability p can be expressed as

$$p(y_1,\ldots, y_{T'} \mid x_1,\ldots, x_T) = \prod_{i=1}^{T'} p(y_i \mid o, y_1,\ldots, y_{i-1}) \tag{4}$$

where $p(y_1,\ldots, y_{T'} \mid x_1,\ldots, x_T)$ represents the probability of converting sequence X to sequence Y, each probability distribution $p(y_i \mid o, y_1,\ldots, y_{i-1})$ is calculated by the projection layer.

In the projection layer, a feature vector is mixed at first, and then softmax is applied over all the semantic marks in the vocabulary V. In the loss layer, the loss value is calculated

according to the target input sequence and target output sequence. The loss function is a weighted sparse-softmax-cross-entropy. The training process follows (5):

$$\arg\max_{\theta} \frac{1}{|\Psi|} \sum_{(Y,X)\in\Psi} \log p(Y|X;\theta) \tag{5}$$

where ψ represents the training data, θ is the parameter of the sequence model.

After training, the most likely result is produced by the greedy algorithm, namely, each semantic mark is unique. Using the trained sequence-to-sequence model, the decoding will stop when an '〈EOS〉' mark is decoded. Under normal circumstances, the length of the decoded sequence is equal to the length of the input. In this case, a simple post-processing step is needed to get the final result string, which includes filtering candidate characters with drop-out mark and sorting candidate characters with valid-index mark. However, there may be two failed cases. One case is that the lengths of input and output are different; another is that the valid-index marks are not consecutive.

3.3 Training detail of the framework

Both the encoder and decoder contain specific training process. Apparently, the performance of trained models is crucial. In this section, a traffic intersection scene is taken as an example to introduce more training details. As shown in Fig. 5, the camera in this traffic scene can change its field of view to capture license plates with different formats and scales.

Fig. 5 *Some examples of images with license plates to be recognised*

3.3.1 Training details of the encoder: In the encoder, two DNN models are needed. One is for natural multi-scale character detection and recognition and the other is for single-scale recognition.

First of all, training data should be prepared. According to the requirements of image annotation mentioned in Section 3.1.1, each license plate character is equipped with a bounding box and an independent label. For example, the license plate in the top right corner of Fig. 5 is labelled as follows: '皖' is labelled as number 14, 'K' is labelled as number 50, '2' is labelled as number 32, '6' is labelled as number 36, '0' is labelled as number 30, '3' is labelled as number 33. However, training data is different for the two DNN models. The multi-scale DNN model in the first encoding stage is trained with images framed by the yellow rectangle as shown in Fig. 5. The single-scale DNN model in the second encoding stage is trained with roughly extracted license plates framed by the green rectangle as shown in Fig. 5.

Secondly, the two required DNN models for synchronised character detection and recognition are built and trained with NVIDIA GeForce GTX 1080 GPU. Since YOLOv3 performs well on detecting small objects by drawing lessons from residual network (ResNet) and feature pyramid network, a network prototype of YOLOv3 is modified as the specific multi-scale DNN model. As shown in Fig. 6, the input size is changed to 352 × 288. Six preset anchors are recomputed by K-means based on the training set. Training batch size is set to 64, while subdivisions size is set to 16. Other hyperparameters are set as follows: momentum is 0.9, decay is 0.005, total iteration is 212,020, and the learning rate is burned in 0.001 and shrinks ten times when iteration value reaches 136,000 and 175,000. Tiny-YOLOv3 is used to train the specific single-scale DNN model. Although the training data is different, the training process of Tiny-YOLOv3 is similar to YOLOv3.

Finally, the trained models are embedded into the encoder of the proposed framework, which work along with other modules to provide encoded sequence with high quality.

3.3.2 Training details of the decoder: In the decoder, a sequence-to-sequence model is trained to generate semantic marks corresponding to the encoded sequence.

Specific training data is prepared by tagging manual annotation to the encoded sequences. Each license plate corresponds to a sequence pair consisting of a source sequence and a target sequence. For example, a roughly extracted Chinese license plate with resolution of 100 × 30 is encoded into a sequence {{*c*: 33-'3',

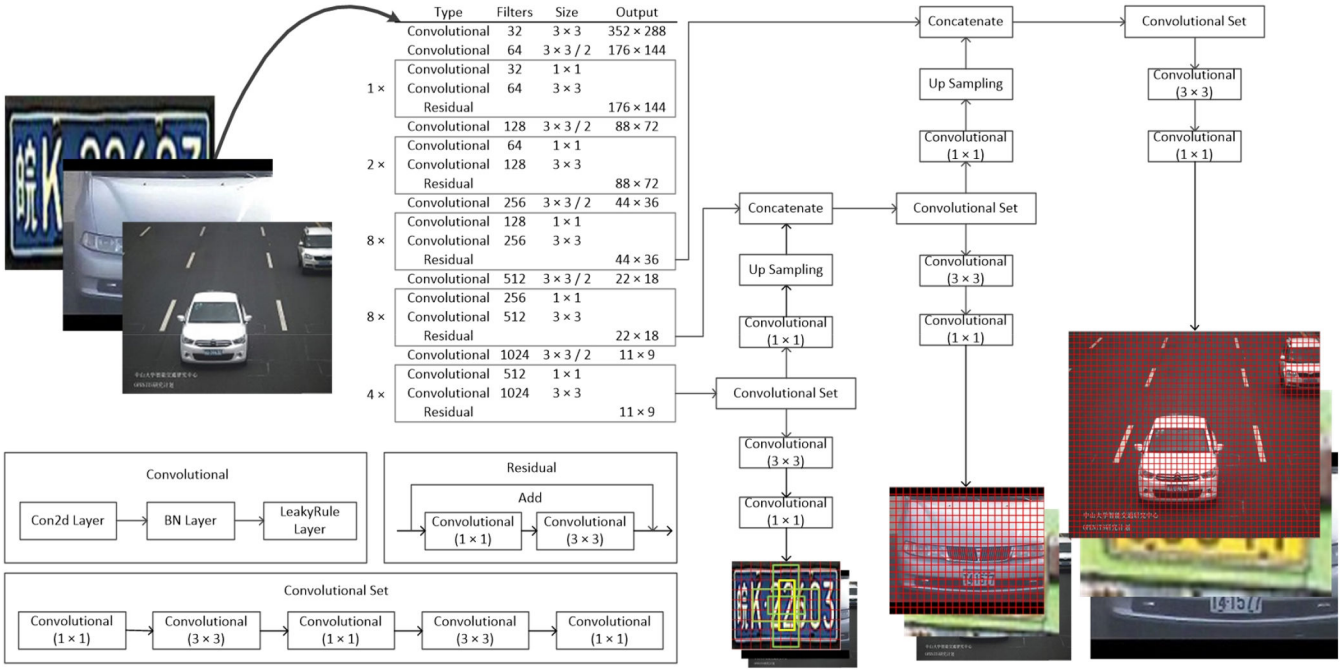

Fig. 6 *DNN for synchronised detection and recognition at the first encoding stage*

Table 1 Detection performance on the AOLP

Method	Usage pattern	Detection rate, %			Time, ms	
		AC	LE	RP	CPU	GPU
Hsu *et al.* [16]	N	96	95	94	90–320	—
Li *et al.* [11]	S	98.38	97.62	95.58	—	1000–2000
Li *et al.* [10]	S	99.12	99.08	98.20	—	400
Xie *et al.* [49]	N	**99.51**	**99.43**	99.46	**7.7**	—
ours (Y* + TY)	N	**99.51**	99.39	**100**	—	20 + 5 = **25**
ours (Y* + Y)	N	**99.51**	99.39	**100**	—	20 + 20 = 40
ours (Y* + TY)	S	**99.51**	99.39	**100**	—	20 + 5 = **25**
ours (Y* + Y)	S	**99.51**	99.39	**100**	—	20 + 20 = 40
ours (Y* + F)	S	**99.51**	99.39	**100**	—	20 + 125 = 146
ours (F* + Y)	S	97.93	98.32	98.63	—	140 + 20 = 161
ours (S* + Y)	S	96.90	88.89	98.04	—	35 + 20 = 56

*P*S: in **Method** column, Y* = YOLOv3 (352 × 288) for multi-scale input, Y = YOLOv3 (352 × 288) for single-scale input, TY = Tiny-YOLOv3 (352 × 288) for single-scale input, F* = Faster R-CNN_ResNet101 (800 × 600) for multi-scale input, F = Faster R-CNN_ResNet101 (352 × 288) for roughly extracted license plate, S* = SSD_MobileNetv2 (300 × 300) for multi-scale input; in usage pattern column, N = Normal usage pattern, S = Special usage pattern.

x: 88, *y*: 14, *w*: 12, *h*: 18, *t*: 0.99}, {*c*: 36-'6', *x*: 63, *y*: 15, *w*: 12, *h*: 19, *t*: 0.96}, {*c*: 50-'K', *x*: 22, *y*: 14, *w*: 11, *h*: 18, *t*: 0.99}, {*c*: 30-'0', *x*: 76, *y*: 14, *w*: 11, *h*: 18, *t*: 0.99}, {*c*: 14-'皖', *x*: 10, *y*: 15, *w*: 13, *h*: 18, *t*: 0.98}, {*c*: 32-'2', *x*: 40, *y*: 14, *w*: 13, *h*: 19, *t*: 0.99}, {*c*: 44-'D', *x*: 77, *y*: 13, *w*: 111, *h*: 19, *t*: 0.57}, {*c*: 32-'2', *x*:52, *y*: 13, *w*: 11, *h*: 18, *t*: 0.99}}. Since the ground-truth is '皖 K22603', the target semantic sequence is labelled as {7, 5, 2, 6, 1, 3, 19, 4}. According to Section 3.2, '19' in the target semantic sequence means drop-out, namely, the false positive result 'D' is dropped out. To get more training pairs, same shuffle strategies, e.g. swapping the first and second elements can be used to re-order the source sequence and target sequence.

Then, a sequence-to-sequence model is built and trained based on the training data. As described in Section 3.2, the model is a neural network with an attention module and five layers. Dropout is used in each LSTM cell, and the dropout rate is set to 0.3. The learning rate is 0.01, clip-gradients-to the-norm is 1.0, the batch size is 128, and max-epoch is 100.

4 Experiments

License plates with different formats and scales can be handled by EDF-LPR because of the distinctive codec mechanism. In the experimental phase, four public datasets, i.e. NTUA, PKU-Vehicle, AOLP, and OpenITS are used to evaluate the actual performance of the proposed framework. License plate detection rate and recognition rate are two important indicators for accuracy verification. Efficiency can be reflected by operation time. Comprehensive performance of license plates with different standard formats and scales shows robustness and flexibility. Although the robustness and flexibility are mainly assessed through qualitative analysis, they also can be derived from quantitative analysis of accuracy and efficiency. In order to facilitate the readers to view and compare the experimental results, some values are displayed in bold in all experimental tables.

4.1 Datasets and preparation

The first dataset is the AOLP dataset, which contains 2049 images of three kinds from Taiwan, China. In this dataset, all images are divided into three groups: access control (AC), law enforcement (LE), and road patrol (RP). Nowadays, quite a few methods have built their experiments on this dataset. So, comparable results can be very intuitive.

The second dataset is PKU-Vehicle, which classifies 3977 vehicle images into five groups (i.e. G1–G5) according to different surroundings. Since only labels for license plate detection are provided, the dataset is used to evaluate the performance on license plate extraction.

The third dataset is from the National Technical University of Athens (NTUA). It is a large image dataset composed of Greek license plates, containing several criteria such as type and colour of plates, illumination conditions, various angles of vision, and indoor or outdoor images. The dataset is really a challenge to verify the robustness of the method for ALPR.

The last dataset, OpenITS, is recently proposed by Zhao *et al.* [6] and consists of Chinese license plates with a wide variety of types. The images are taken at natural urban traffic intersections, and every image contains only one license plate. In addition, each image is attached with a ground-truth of license plate string.

The preparation is to retrain the framework for each dataset before testing by referring to the training details mentioned in Section 3.3. Specifically, both the two DNN models are needed to evaluate the performance of license plate detection and all the three models are necessary for testing the performance of license plate recognition.

4.2 Performance of detection

As mention in Section 3.1, when the encoded sequence is identified as a well-encoded sequence, it is considered that the license plate is indirectly detected. Therefore, this section shows the actual effect of license plate detection based on the two-stage encoder. Since the definition of successful detection varies in different methods, in this study, if R_c in formula (6) is >80% and R_b in formula (7) is <3, it is thought to be a successful license plate detection

$$R_c = \frac{\text{Area}(r_{gt} \cap r_p)}{\text{Area}(r_{gt})} \tag{6}$$

$$R_b = \frac{\text{Area}(r_p)}{\text{Area}(r_{gt})} \tag{7}$$

where, r_{gt} represents the ground-truth, r_p is the predicted result, Area(*) is a function that counts the area of the input rectangle, R_c is the cover rate, the operator '∩' calculates the union of two rectangles, and R_b is the rough detection rate.

The detection performance on AOLP is evaluated at first. Based on the fact that a regular license plate in AOLP has six characters, N_{min} is set to 6, and T_{min} is set to 0.9. Since AOLP has already been divided into training parts (100 images from AC, 100 images from LE, and 100 images from RP, totally 300 images) and testing part (the rest images of AC, LE, and RP, totally 1749 images), a normal usage pattern to AOLP is just to follow the division. However, Li *et al.* proposed a special usage pattern to AOLP in [10, 11], which use images from different sub-datasets for training and test, respectively. For example, images from LE and RP subsets are used to train the network, and the AC subset is used to evaluate the performance. For each training set, a multi-scale DNN model and a single-scale DNN need to be re-trained. As shown in Table 1, both two usage patterns are referred for comparison. It is found that the detection performance mainly depends on the multi-scale DNN model trained in the first encoding stage. It is noted that ours (*S** + *Y*) gets a low detection rate in LE because the traffic

Fig. 7 *Examples of license plate detection results from AOLP, which are indicated by red rectangles*

Table 2 Detection performance on the PKU-Vehicle

Method	Detection rate, %					Time, ms
	G1	G2	G3	G4	G5	
Zhou *et al.* [28]	95.43	97.85	94.21	81.23	82.37	475
Yuan *et al.* [17]	98.76	98.42	97.72	96.23	97.32	42
Li *et al.* [10]	99.88	99.86	99.60	**100**	**99.31**	279
ours	**100**	**100**	**100**	96.5	87.82/**99.31**	**≥20**

Fig. 8 *Examples of license plate detection results from PKU-Vehicle, which are indicated by red rectangles*

Table 3 Detection performance on the NTUA

Method	Detection rate, %	Time, ms
Anagnostopoulos *et al.* [1]	89.1	117
Zhai *et al.* [55]	98	143
Le *et al.* [56]	97.37	—
Yepez *et al.* [23]	98.54	**20**
ours (Incomplete)	**99.47**	25

scene is complex and feature extracting neural network is shallow. On the whole, YOLOv3 behaves better than Faster R-CNN and SSD. It is consistent with the fact that YOLOv3 is good at dealing with small targets. In addition to satisfactory detection results, the detection efficiency is also excellent. It takes the framework about 20 ms per image to finish the first encoding stage, and 5 ms per license plate to finish the second encoding stage. Although Xie *et al.* finished the detection in 7.7 ms, further recognition would cost more time. As shown in Fig. 7, the failed case does not contain enough context of the license plate. This discovery is explored below.

Secondly, the detection performance on PKU-Vehicle is assessed. Here, $N_{\min}$ is set to 7 and $T_{\min}$ is set to 0.9. Since PKU-Vehicle does not define a usage pattern, training set preparation is a problem. Luckily, our personal dataset is similar to PKU-Vehicle, so models are trained based on our own dataset, which is composed of 6759 images. Specifically, a YOLOv3 model is trained for the first encoding stage and a Tiny-YOLOv3 model is trained for the second encoding stage. Since others also use their personal data, the comparison is fair. As shown in Table 2, the results are satisfactory. As shown in Fig. 8, the illumination may be bad. Since our personal training set does not contain similar images, the detection rate of G4 is a little lower. Besides, G5 also shows a controversial issue caused by the training data. It is noticed that the context of the license plate region is also considered during the training phase. That is to say, the trained model is sensitive to vehicle information, and when features of vehicles are weak, license plate detection may fail. It helps filter false positives, but it also removes some true positives. Thus, it is hard to define the detection rate of G5 because some license plates do not have enough contexts. Therefore, two detection rates are exhibited in column G5. Although still insist on formulas (6) and (7), one is calculated on the basis of all license plates, and the other is calculated on the basis of majority of license plates. The latter regards the license plate as a valid one, if and only if the window of the vehicle to which the license plate belongs can be observed. Li *et al.* [10] achieved a pretty good performance because 332,000 images with rich information are used for training. However, they have not released the dataset. It means that there is a big margin for improvement because training data plays an important role in deep learning.

Thirdly, the detection performance on NTUA is tested. Both AOLP and PKU-Vehicle are composed of Chinese license plates. NTUA is a European dataset. However, NTUA is a raw dataset that has no manual annotations. Owing to the great difference between the Chinese license plate and European license plate in font and colour, our personal dataset cannot be used as a training set. Therefore, manual annotations of license plate characters are added to NTUA. One-seventh of images are selected for training and validation, and others are chosen for testing. Similarly, a YOLOv3 model is trained for the first encoding stage and a Tiny-YOLOv3 model is trained for the second encoding stage. Based on the fact that a single license plate in this dataset has at least six characters, $N_{\min}$ is set to 6 and $T_{\min}$ is set to 0.9. As shown in Table 3, the detection rate reaches 99.47%. Since not all data are used for testing, special tag 'Incomplete' is attached in the table. Visual results are shown in Fig. 9.

4.3 Performance of recognition

The proposed EDF-LPR is ready for ALPR, so the performance of recognition is tested on both AOLP and OpenITS. The recognition rate is the most convincing indicator in the task of ALPR. Since the sequence-to-sequence model is trained to replace heuristic rules, a heuristic decoder is necessary to help evaluate the performance of sequence-to-sequence decoder.

First of all, the recognition performance on AOLP is evaluated. Although the detection performance on AOLP has been proved to be excellent, it only proves the detection quality of encoded sequences. The recognition quality needs to be further evaluated by

Fig. 9 *Examples of license plate detection results from NTUA, which are indicated by red rectangles*

Table 4 Recognition performance on the AOLP

Method	Usage pattern	Recognition rate, %			Time, ms	
		AC	LE	RP	CPU	GPU
Hsu *et al.* [16]	N	88.5	86.6	85.7	90–320	—
Li *et al.* baseline [11]	S	93.53	89.83	86.58	—	—
Li *et al.* [11]	S	94.85	94.19	88.38	—	1000–2000
Li *et al.* [10]	S	**95.59**	**96.43**	83.80	—	400
ours (Y* + TY + HR)	N	90.50	86.22	82.19	—	20 + 5 + 1 = **26**
ours (Y* + TY + Seq2Seq)	N	90.50	86.22	82.19	—	20 + 5 + 3 = 28
ours (Y* + Y + HR)	N	**94.13**	**91.42**	**94.91**	—	20 + 20 + 1 = 41
ours (Y* + Y + Seq2Seq)	N	93.61	91.12	94.72	—	20 + 20 + 3 = 43
ours (Y* + TY + HR)	S	94.13	86.32	89.03	—	20 + 5 + 1 = **26**
ours (Y* + TY + Seq2Seq)	S	93.61	86.22	88.38	—	20 + 5 + 3 = 28
ours (Y* + Y + HR)	S	**95.30**	**94.56**	**94.44**	—	20 + 20 + 1 = 41
ours (Y* + Y + Seq2Seq)	S	94.30	92.83	92.14	—	20 + 20 + 3 = 43

PS: in method column, Y* = YOLOv3 (352 × 288) for multi-scale input, Y = YOLOv3 (352 × 288) for single-scale input, TY = Tiny-YOLOv3 (352 × 288) for single-scale input, Seq2Seq = sequence-to-sequence, HR = heuristic rules; in usage pattern column, N = normal usage pattern, S = special usage pattern.

experiments. The performance of the sequence learning based decoder is just as important to be tested. Therefore, a heuristic rule-based decoder is built for comparison. All heuristic rules are checked to be right for each encoded sequence. As shown in Table 4, the sequence-to-sequence model reaches almost the same level as the heuristic rules. As expected, the sequence-to-sequence model can learn to filter and sort the encoded sequence automatically. Moreover, speed is also satisfactory. The recognition rate of AC shows the stability of EDF-LPR. The recognition rate of LE shows how important the feature extraction capability of DNN should be. Since images in LE are captured at complex traffic scene, deeper neural networks are stronger. The recognition rate of RP verifies the superiority of the proposed controllable encoder. All the license plates can be effectively recognised even if parts of them are tilted, which is attributed to the consideration of tilt correction in EDF-LPR. Besides, the whole processing is fast at about 26–43 ms.

OpenITS contains nearly all kinds of Chinese license plates. There was no pure ALPR algorithm that can deal with this dataset except the commercial license plate recognition system. Thus, it is comprehensive to compare the proposed framework with five Chinese commercial ALPR systems mentioned in [6]. However, the dataset released online is not the same as the one introduced in [6]. So, there are no other comparison methods in the following two experiments.

One experiment is designed to verify the compatibility of EDF-LPR with license plates in different standard formats. Our personal data is used to train the model when evaluating the detection performance of EDF-LPR on PKU-Vehicle. However, our personal data contains only a few formats, which is much less than OpenITS. Since OpenITS has been divided into three parts, two of which are natural images, a new usage pattern is applied in this study. The two parts (one contains 1403 images, the other contains 3204 images) are used for training and testing respectively. Since single license plate in OpenITS has at least six characters, $N_{\min}$ is set to 6 and $T_{\min}$ is set to 0.9. As shown in Table 5, independent YOLOv3-based encoder and sequence learning-based decoder are implemented for each of the three training sets separately. It shows that EDF-LPR is able to recognise multiple types of license plates flexibly. The count numbers in the table address the imbalance of samples. The influence of imbalance is more serious in MD than SD. As a matter of fact, the recognition performance of license plates in single-row style is strengthened, while the effect of license plates in the double-row style is weakened. Besides, some recognition results are presented in Fig. 10, which shows that the overall performance of ALPR is good, but insufficient training samples lead to some mistakes.

Another experiment is conducted to further evaluate the encoding performance based on different DNN models. The detection performance of the encoder has been proved in Section 4.2, and the recognition performance of the encoder is tested in this experiment. To ensure that the final results are only affected by the encoder, standard format is fixed to 'small vehicle', and the decoder based on heuristic rules is used uniformly. Experimental results are shown in Table 6. Faster R-CNN model with large size of input tensor can be a strong encoder in detection, but it is time-consuming. Faster R-CNN model appears to be weaker than YOLOv3 in recognition, because Faster R-CNN may assign more than one bounding box for each ground-truth object. On the contrary, YOLO only assigns one bounding box prior to each ground-truth object. If more than one bounding box prior is assigned for each ground-truth object, it incurs loss for coordinate and class prediction. Changing the input size of Faster R-CNN seems to be a valid solution because the intersection over union (IoU) between a bounding box prior and its ground-truth affects the calculation of loss a lot.

5 Conclusion

In this study, a new bottom-up encoder–decoder framework named EDF-LPR is proposed to unify ALPR in different standard formats and scales. This issue is solved by converting the ALPR to a sequence processing task with a distinctive controllable codec mechanism. The idea is ingenious and its implementation is brand-new and skilful. Engineering ideas can be used to extend the framework.

State-of-the-art performance is achieved via EDF-LPR, which is proved to be robust, flexible, accurate, and efficient through a series of experiments. The well-designed encoder shows the value of indirect license plate detection. The sequence learning-based decoder proves that heuristic rules can be learned by the machine

Table 5 Recognition performance on the OpenITS

Type of license plates	Rows style	Number	Accuracy of ALPR, %					
			Ours (PD)		Ours (SD)		Ours (MD)	
			Seq2Seq	HR	Seq2Seq	HR	Seq2Seq	HR
small vehicles	1	**2507**	**97.6**	**97.8**	96.5	96.9	96.6	96.6
large vehicles (front)	1	98	**99.0**	**100**	94.9	95.9	95.9	96.9
large vehicles (back)	2	31	—	—	**77.4**	**80.7**	67.7	77.4
trailer	2	36	—	—	**55.6**	**61.1**	50.0	55.6
test-drive vehicle	1	86	91.9	91.9	93.0	93.0	**97.7**	**98.8**
police service vehicle	1	84	86.9	86.9	**95.2**	**96.4**	**95.2**	95.2
military vehicle	1, 2	70	82.9	84.3	80.0	80.0	**88.6**	**90.0**
armed police force vehicle	1, 2	84	—	—	89.3	90.5	**90.5**	**91.7**
cross-border vehicle	1	158	—	—	72.8	72.8	**73.4**	**73.4**
embassy vehicle	1	11	—	—	**100**	**100**	**100**	**100**
tractor	2	4	—	—	**50.0**	**50.0**	25.0	**50.0**
low-speed vehicle	2	27	—	—	**81.5**	81.5	**81.5**	**85.2**
ordinary motorcycle	2	8	—	—	**75.0**	**75.0**	50.0	**75.0**

PS: in rows style column, 1 represents single-row standard formats, 2 represents double-row standard formats. PD means the model is trained with personal data, SD means the model is trained with specific data of OpenITS, MD means the model is trained with mixture data of PD and SD. Seq2Seq = sequence-to-sequence, HR = heuristic rules.

Fig. 10 *Examples of license plate recognition results from OpenITS. In each sub-figure, detected license plates are listed on the right side of the traffic image, and results are tagged below. In addition, rainbow colours are used to present the sorted characters. Seq2Seq means the result of sequence learning and HR means the result of heuristic rules*

(a) Correct recognition example of single-row license plate, ***(b)*** Correct recognition example of double-row license plate, ***(c)*** Wrong decoding example, ***(d)*** Wrong encoding example

Table 6 Recognition performance of different encoders on the category of small vehicle in the OpenITS

The multi-scale DNN model	The single-scale DNN model	Detection rate, %	Recognition accuracy, %	Speed, ms
YOLOv3 (352 × 288)	Tiny-YOLOv3 (352 × 288)	100	96.9	26
YOLOv3 (352 × 288)	YOLOv3 (352 × 288)	100	98.4	42
YOLOv3 (352 × 288)	Faster R-CNN_ResNet101 (352 × 288)	100	78.3	146
YOLOv3 (352 × 288)	Faster_RCNN_ResNet101 (800 × 600)	100	92.9	161
Faster_RCNN_ResNet101 (800 × 600)	YOLOv3 (352 × 288)	98.27	98.1	161
Faster_RCNN_ResNet101 (800 × 600)	Faster R-CNN_ResNet101 (352 × 288)	98.27	78.4	277
Faster_RCNN_ResNet101 (800 × 600)	Faster R-CNN_ResNet101 (800 × 600)	98.27	97.5	301

automatically. It is believed that EDF-LPR will be more powerful when a stronger object detection DNN is chosen. EDF-LPR is very generic for detecting and recognising license plates with different formats since no specific DNN model needs to be specified.

However, there remain some issues to be handled further. Sometimes, a few target characters may be missed even though the recognition strategy is facilitated. Besides, similar characters may be confusing during recognition. Those questions are part of our future research.

6 Acknowledgments

This work was supported by the National Natural Science Foundation of China under grant no. 61976193, the Zhejiang Provincial Science and Technology Planning Key Project of China under grant no. 2018C01064, and the Zhejiang Provincial Natural Science Foundation of China under grant no. LY19F020027.

7 References

[1] Hinton, G.E., Salakhutdinov, R.R.: 'Reducing the dimensionality of data with neural networks', *Science*, 2006, **313**, (5786), pp. 504–507

[2] Anagnostopoulos, C.E., Anagnostopoulos, I.E., Psoroulas, I.D., *et al.*: 'License plate recognition from still images and video sequences: a survey', *IEEE Trans. Intell. Transp. Syst.*, 2008, **9**, (3), pp. 377–391

[3] Buch, N., Velastin, S.A., Orwell, J.: 'A review of computer vision techniques for the analysis of urban traffic', *IEEE Trans. Intell. Transp. Syst.*, 2011, **12**, (3), pp. 920–939

[4] Du, S., Ibrahim, M., Shehata, M., *et al.*: 'Automatic license plate recognition (ALPR): a state-of-the-art review', *IEEE Circuits Syst. Video Technol.*, 2013, **23**, (2), pp. 311–325

[5] Tian, B., Morris, B.T., Tang, M., *et al.*: 'Hierarchical and networked vehicle surveillance in ITS: a survey', *IEEE Trans. Intell. Transp. Syst.*, 2017, **18**, (1), pp. 25–48

[6] Zhao, Y., Yu, Z., Li, X.: 'Evaluation methodology for license plate recognition systems and experimental results', *IET Intell. Transp. Syst.*, 2018, **12**, (5), pp. 375–385

[7] Sadique, M.F., Haque, S.M.R: 'A comparative study of license plate detection and recognition techniques'. Proc. 22nd Int. Conf. on Computer Information Technology, Dhaka, Bangladesh, 2019, pp. 1–6

[8] Spanhel, J., Sochor, J., Juranek, R., *et al.*: 'Holistic recognition of low-quality license plates by CNN using track annotated data'. Proc. 14th IEEE Int. Conf. on Advanced Video and Signal Based Surveillance, Lecce, Italy, 29 August–1 September 2017, pp. 1–6

[9] Wang, J., Huang, H., Qian, X., *et al.*: 'Sequence recognition of Chinese license plates', *Neurocomputing*, 2018, **317**, pp. 149–158

[10] Li, H., Wang, P., Shen, C.: 'Towards end-to-end car license plates detection and recognition with deep neural networks', *IEEE Trans. Intell. Transp. Syst.*, 2019, **20**, (3), pp. 1126–1136

[11] Li, H., Wang, P., You, M., *et al.*: 'Reading car license plates using deep neural networks', *Image Vis. Comput.*, 2018, **72**, pp. 14–23

[12] Bar, M.: 'The proactive brain: using analogies and associations to generate predictions', *Trends Cogn. Sci.*, 2007, **11**, (7), pp. 280–289

[13] Huang, X., He, P., Rangarajan, A., *et al.*: 'Intelligent intersection: two-stream convolutional networks for real-time near-accident detection in traffic video', *ACM Trans. Spatial Algorithms Syst.*, 2020, **6**, (2), pp. 1–28

[14] Ke, R., Li, Z., Kim, S., *et al.*: 'Real-time bidirectional traffic flow parameter estimation from aerial videos', *IEEE Trans. Intell. Transp. Syst.*, 2016, **18**, (4), pp. 890–901

[15] Sutskever, I., Vinyals, O., Le, Q.V.: 'Sequence to sequence learning with neural networks'. Conf. on Neural Information Processing Systems, Montreal, Canada, 2014, pp. 3104–3112

[16] Hsu, G., Chen, J., Chung, Y.: 'Application-oriented license plate recognition', *IEEE Trans. Veh. Technol.*, 2013, **62**, (2), pp. 552–561

[17] Yuan, Y., Zou, W., Zhao, Y., *et al.*: 'A robust and efficient approach to license plate detection', *IEEE Trans. Image Process.*, 2017, **26**, (3), pp. 1102–1114

[18] Redmon, J., Farhadi, A.: 'YOLOv3: an incremental improvement'. Available at http://arxiv.org/abs/1804.02767, accessed April 2018

[19] Ren, S., He, K., Girshick, R., *et al.*: 'Faster R-CNN: towards real-time object detection with region proposal networks', *IEEE Trans. Pattern Anal. Mach. Intell.*, 2017, **39**, (6), pp. 1137–1149

[20] Liu, W., Dragomir, A., Dumitru, E., *et al.*: 'SSD: single shot MultiBox detector'. Proc. 14th European Conf. on Computer Vision, Amsterdam, The Netherlands, 8–16 October 2016, pp. 21–37

[21] Saini, M.K., Saini, S.: 'Multiwavelet transform based license plate detection', *J. Vis. Commun. Image Represent.*, 2017, **44**, (4), pp. 128–138

[22] Asif, M.R., Chun, Q., Hussain, S., *et al.*: 'Multinational vehicle license plate detection in complex backgrounds', *J. Vis. Commun. Image Represent.*, 2017, **46**, (7), pp. 176–186

[23] Yepez, J., Ko, S.: 'Improved license plate localisation algorithm based on morphological operations', *IET Intell. Transp. Syst.*, 2018, **12**, (6), pp. 542–549

[24] Dun, J., Zhang, S., Ye, X., *et al.*: 'Chinese license plate localization in multi-lane with complex background based on concomitant colors', *IEEE Trans. Intell. Transp. Syst.*, 2015, **7**, (3), pp. 51–61

[25] Ashtari, A.H., Nordin, M.J., Fathy, M.: 'An Iranian license plate recognition system based on color features', *IEEE Trans. Intell. Transp. Syst.*, 2014, **15**, (4), pp. 1690–1705

[26] Lim, H.W., Tay, Y.H.: 'Detection of license plate characters in natural scene with MSER and SIFT unigram classifier'. Proc. IEEE Conf. on Sustainable Utilization Development Engineering Technology, Petaling Jaya, Malaysia, 2010, pp. 95–98

[27] Matas, J., Chum, O., Urban, M., *et al.*: 'Robust wide baseline stereo from maximally stable extremal regions', *Image Vis. Comput.*, 2004, **22**, (10), pp. 761–767

[28] Zhou, W., Li, H., Lu, Y., *et al.*: 'Principal visual word discovery for automatic license plate detection', *IEEE Trans. Image Process.*, 2012, **21**, (9), pp. 4269–4279

[29] Al-shemarry, M.S., Li, Y., Abdulla, S.: 'Ensemble of adaboost cascades of 3L-LBPs classifiers for license plates detection with low quality images', *Expert Syst. Appl.*, 2018, **92**, (2), pp. 216–235

[30] Rafique, M.A., Pedrycz, W., Jeon, M.: 'Vehicle license plate detection using region-based convolutional neural networks', *Soft Comput.*, 2018, **22**, (19), pp. 6429–6440

[31] Molina-Moreno, M., González-Díaz, I., Díaz-de-María, F.: 'Efficient scale-adaptive license plate detection system', *IEEE Trans. Intell. Transp. Syst.*, 2019, **20**, (6), pp. 2109–2121

[32] Safaei, A., Tang, H.L., Sanei, S.: 'Real-time search-free multiple license plate recognition via likelihood estimation of saliency', *Comput. Electr. Eng.*, 2016, **56**, (11), pp. 15–29

[33] Menotti, D., Schwartz, W.R., Federal, U., *et al.*: 'License plate recognition based on temporal redundancy'. Proc. 19th IEEE Int. Conf. on Intelligent Transportation Systems, Rio de Janeiro, Brazil, 2016, pp. 2577–2582

[34] Kim, D., Song, T., Lee, Y., *et al.*: 'Effective character segmentation for license plate recognition under illumination changing environment'. Proc. IEEE Int. Conf. on Consumer Electronics, Las Vegas, NV, USA, 2016, pp. 532–533

[35] Khan, M.A., Sharif, M., Javed, M.Y., *et al.*: 'License number plate recognition system using entropy-based features selection approach with SVM', *IET Image Process.*, 2018, **12**, (2), pp. 200–209

[36] Hou, Y., Qin, X., Zhou, X., *et al.*: 'License plate character segmentation based on stroke width transform'. Proc. 8th Int. Congress Image Signal Processing, Shenyang, China, 2015, pp. 954–958

[37] Povolotskiy, M.A., Kuznetsova, E.G., Khanipov, T.M.: 'Russian license plate segmentation based on dynamic time warping'. Proc. 31st European Conf. on Modelling Simulation, Budapest, Hungary, 2017, pp. 285–291

[38] Gou, C., Wang, K., Yao, Y., *et al.*: 'Vehicle license plate recognition based on extremal regions and restricted Boltzmann machines', *IEEE Trans. Intell. Transp. Syst.*, 2016, **17**, (4), pp. 1096–1107

[39] Tian, J., Wang, R., Wang, G., *et al.*: 'A two-stage character segmentation method for Chinese license', *Comput. Electr. Eng.*, 2015, **46**, (8), pp. 539–553

[40] Tabrizi, S.S., Cavus, N.: 'A hybrid KNN-SVM model for Iranian license plate recognition'. Proc. 12th Int. Conf. on Application Fuzzy Systems Soft Computing, Vienna, Austria, 2016, pp. 588–594

[41] Yang, Y., Li, D., Duan, Z.: 'Chinese vehicle license plate recognition using kernel-based extreme learning machine with deep convolutional features', *IET Intell. Transp. Syst.*, 2018, **12**, (3), pp. 213–219

[42] Kim, H.H., Park, J.K., Oh, J.H., *et al.*: 'Multi-task convolutional neural network system for license plate recognition', *Int. J. Control Autom. Syst.*, 2017, **15**, (6), pp. 2942–2949

[43] Zherzdev, S., Gruzdev, A.: 'License plate recognition via deep neural networks'. Available at https://arxiv.org/abs/1806.10447v1, accessed June 2018

[44] Guo, Q., Wang, F., Lei, J., *et al.*: 'Convolutional feature learning and hybrid CNN-HMM for scene number recognition', *Neurocomputing*, 2016, **184**, pp. 78–90

[45] Bulan, O., Kozitsky, V., Ramesh, P., *et al.*: 'Segmentation- and annotation-free license plate recognition with deep localization and failure identification', *IEEE Trans. Intell. Transp. Syst.*, 2017, **18**, (9), pp. 2351–2363

[46] Redmon, J., Farhadi, A.: 'YOLO9000: better, faster, stronger'. Proc. IEEE Conf. on Computer Vision Pattern Recognition, Honolulu, HI, USA, 2017, pp. 6517–6525

[47] Laroca, R., Severo, E., Zanlorensi, L.A., *et al.*: 'A robust real-time automatic license plate recognition based on the YOLO detector'. Proc. Int. Joint Conf. on Neural Networks, Rio de Janeiro, Brazil, 2018, pp. 1–10

[48] Silva, S.M., Jung, C.R.: 'Real-time Brazilian license plate detection and recognition using deep convolutional neural networks'. Proc. 30th Conf. on Graphics Patterns Images, Niteroi, Brazil, 2017, pp. 55–62

[49] Xie, L., Ahmad, T., Jin, L., *et al.*: 'A new CNN-based method for multi-directional car license plate detection', *IEEE Trans. Intell. Transp.*, 2018, **19**, (2), pp. 507–517

[50] Hsu, G., Ambikapathi, A., Chung, S., *et al.*: 'Robust license plate detection in the wild'. Proc. 14th IEEE Int. Conf. on Advanced Video Signal Surveillance, Lecce, Italy, 2017, pp. 1–6

[51] Lin, C., Lin, Y., Liu, W.: 'An efficient license plate recognition system using convolution neural networks'. Proc. IEEE Int. Conf. on Applied System Invention, Chiba, Japan, 2018, pp. 224–227

[52] Vinyals, O., Toshev, A., Bengio, S., *et al.*: 'A neural image caption generator'. Proc. IEEE Conf. Computer Vision Pattern Recognition, Boston, MA, USA, 2015, pp. 3156–3164

[53] Lin, C., Li, Y.: 'A license plate recognition system for severe tilt angles using mask R-CNN'. Proc. Int. Conf. on Advanced Mechatronic Systems, Kusatsu, Shiga, Japan, 2019, pp. 229–234

[54] Castro-Zunti, R.D., Yépez, J., Ko, S.: 'License plate segmentation and recognition system using deep learning and OpenVINO', *IET Intell. Transp. Syst.*, 2020, **14**, (2), pp. 119–126

[55] Zhai, X., Bensaali, F., Ramalingam, S.: 'Improved number plate localisation algorithm and its efficient field programmable gate arrays implementation', *IET Circuits Devices Syst.*, 2011, **7**, (2), pp. 93–103

[56] Le, T., Tran, V., Hamamoto, K.: 'A robust and flexible license plate detection method'. Proc. Int. Conf. on Advanced Technologies Communications, Hanoi, Vietnam, 2014, pp. 326–331