Pattern Recognition 93 (2019) 134–146

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Robust license plate recognition using neural networks trained on synthetic images

Tomas Björklund [a], Attilio Fiandrotti [b,c], Mauro Annarumma [b], Gianluca Francini [b], Enrico Magli [a,*]

[a] Politecnico di Torino, Corso Duca degli Abruzzi, 24, Torino, 10129, Italy
[b] Telecom Italia, Via Pier Carlo Boggio 59, Torino, 10138, Italy
[c] Télécom Paristech, Université Paris-Saclay, Paris, 75013, France

ARTICLE INFO

Article history:
Received 23 August 2018
Revised 19 March 2019
Accepted 9 April 2019
Available online 10 April 2019

Keywords:
License plate recognition (LPR)
Convolutional neural network (CNN)
Synthetic training

ABSTRACT

In this work, we describe a License Plate Recognition (LPR) system designed around convolutional neural networks (CNNs) trained on synthetic images to avoid collecting and annotating the thousands of images required to train a CNN. First, we propose a framework for generating synthetic license plate images, accounting for the key variables required to model the wide range of conditions affecting the aspect of real plates. Then, we describe a modular LPR system designed around two CNNs for plate and character detection enjoying common training procedures and train the CNNs and experiment on three different datasets of real plate images collected from different countries. Our synthetically trained system outperforms multiple competing systems trained on real images, showing that synthetic images are effective at training a CNNs for LPR if the training images have sufficient variance of the key variables controlling the plate aspect.

1. Introduction

License Plate Recognition (LPR) [1] is expected to be a key computer vision problem in tomorrow smart cities, where pervasive networks of cameras deployed at road intersections identify the vehicles routing through urban environments. The possibility of leveraging pre-existing networks of visible light cameras for LPR is very appealing due to the cost of installing ad-hoc infrastructures such as IR illuminators and cameras. Unfortunately, LPR in natural visible light is a daunting task due to the often non-optimal camera placement which results in a widely varying plate perspective and scale. Similarly, the need to rely on ambient illumination (daylight or street illumination) makes LPR very challenging due to large variations in plate scale, perspective, illumination and limited resolution. In addition, massive-scale vehicle tracking requires conveying a large number of video flows to a central server [2], which calls for installing the required bandwidth and dimensioning the datacenter to handle peak-time requests, making it desirable that the LPR tasks can be performed at least in part locally on the cameras.

Convolutional Neural Networks (CNNs) have emerged as the cornerstone to efficiently solve a number of computer vision problems [3] such as object detection [4,5] or character recognition [6–8] after winning the 2012 ImageNET Challenge [9,10] with a large margin compared to more traditional image processing techniques [11–15]. CNNs are feed-forward, multi-layer neural networks, structured into a feature extraction stage followed by an inference stage [16]. The feature extraction stage includes a number of convolutional layers, each encompassing a number of learnable filters. Each filter activates upon the detection of one specific feature in the input. The output of the feature extraction stage is processed by one or more fully-connected layers, the actual number of layers and learnable parameters in each layer depending on the specific application. Finally, the last layer of the network provides the desired network output such as the class an object belongs to (classification problems) or the object position in the image (regression problems).

While CNNs have shown groundbreaking performance in a number of image processing tasks [6,10,17–19], designing a practical LPR system around CNNs poses some significant design challenges.

First, it is crucial that CNNs are trained on a sufficient number of sample images with a distribution of the variables contributing to the intra-class variance representative of real-world conditions. Sourcing and annotating a large training dataset with controlled

* Corresponding author.
E-mail address: enrico.magli@polito.it (E. Magli).

https://doi.org/10.1016/j.patcog.2019.04.007
0031-3203/

distribution of variables is a daunting task even considering data augmentation techniques.

Second, recent CNN architectures achieving state-of-the-art performance rely on deep topologies with hundreds of millions of learnable parameters. The requirement for moving the LPR processing (at least in part) over smart cameras, calls for a constrained complexity network architecture suitable for smart cameras with limited memory and computational resources.

The present work builds upon and substantially extends the preliminary results of [20] in several ways.

First, we propose a framework tailored to generate synthetic images of license plates. Synthetic training has been advocated before, e.g., for pose estimation [21], text spotting [22] or disparity and optical flow estimation [23]. In [22] it is proposed to generate realistic images by constraining the pose of some synthetic text to match geometry of a natural background. However, matching the background geometry constrains the samples ability to account for the large text variability that makes LPR so challenging. Moreover, realistic-looking synthetic images are not a guarantee that the image captures the key variables driving the plate aspect nor that they are effective at training a CNN. Instead, our framework generates images that, while not necessarily realistic, do account for the key variables controlling the plate aspect and thus are effective towards the goal of training a CNN for plate recognition. We achieve such goal by modeling the key variables responsible for the large intra-class variability of real plate images and allowing arbitrary, unconstrained combinations thereof. Our synthetic training method allows our LPR system to handle different plate types (e.g., country, shape) by simply adapting the synthetic templates and fonts. Recently, it has been proposed to use deep generative models such as Generative Adversarial Nets [24] to produce quasi-realistic synthetic training data. However, recent evidence [25] suggests that realistic looking training images may not be crucial to successfully train a discriminative network. Moreover, generative networks need to be trained on images as well, thus it does not totally solve the issue with the training samples that we are trying to address. Therefore, we do not claim that synthetic training is a better option to natural image based training, rather we point out that it is an effective solution in terms of coping with the lack of natural annotated sample images towards training a discriminative model [23].

Second, we propose a CNN architecture that is reusable for both plate and character detection and is suitable for limited-complexity devices. Concerning the first point, we define network design principles enabling generic joint object localization and classification (object detection, in the following). Then, by adapting the network hyper-parameters, we dimension it either as a plate or character detector, with the obvious advantage of shared training procedures and cost function definitions. Concerning the second point, while our system is not designed to meet the tight constraints of silicon implementations [26], yet its resource-wise design reduces the memory footprint to the point where it can be deployed over real embedded platforms, as we experimentally show.

Third, we propose a modular LPR design with one CNN for plate detection and one for character detection, both trained on synthetic images. The modular design allows for addressing the different requirements of real LPR systems, such as plate segmentation and character reading, in part or totally on a smart camera device. Finally, by means of a plate and character refinement classification, we further improve the performance of our system with a minimal increase in system and design complexity.

To demonstrate the validity of our methods and trained networks, we experiment on three distinct test sets of real world Italian, Taiwanese and Chinese plate images. That is, in all of our experiments we train our system on synthetic images but we test it on real world, natural, images. Such images are captured in challenging conditions of perspective and illumination, which makes them a challenging benchmark. The experiments on Italian plates show greatly improved performance over the preliminary results presented in [20]. The new experiments on Taiwanese and Chinese plates confirm consistent performance across multiple datasets, demonstrating the ability of our LPR system to achieve state-of-the-art performance at reading different plate types despite the relatively low number of parameters of our networks. Furthermore, we benchmark our networks on an embedded platform and a powerful desktop GPU, showing our CNNs are suitable for embedded platforms and achieve nearly real-time performance over server platforms.

The rest of this paper is organized as follows. Section 2 discusses the relevant literature. Section 3 describes our framework for generating synthetic license plate images for training purposes. Section 4 describes a CNN for joint object localization and classification (detection) and its tailoring for plate and character detection respectively. In Section 5 we describe the architecture of a complete LPR system designed around our CNNs and in Section 6 the relative network training and parameter tuning procedures. Section 7 provides an experimental evaluation of our architecture on different sets of images and compares it with the state-of-the-art. Finally, the conclusions of our research are presented in Section 8.

2. Related works

Over the last years, a number of designs based on different architectures and technologies have been proposed to solve the LPR problem.

Yu et al. [27] focus on the problem of detecting car plates under wildly changing illumination, background and perspective conditions. They propose a method based on wavelet transform and empirical mode decomposition that shows good accuracy at plate localization, albeit they do not propose a complete LPR pipeline.

Guo et al. [28] address both the problems of plate segmentation and character segmentation by means of classic image processing techniques and with a parameter-adaptive approach. However, such work does not solves the problem of character recognition, which is a key task towards a complete LPR system.

Giannoukos et al. [29] proposes a fast approach to reduce the plate detection time in full-HD, high-resolution, images. Namely, they propose a context scanning method capable to process a QVGA image in less than 50 ms on a CPU. Nevertheless, modern GPU-accelerated CNNs are able to process even larger images in comparable time.

Al-Ghaili et al. [30] propose a fast method for license plate by identifying vertical edges exploiting contrast, suppressing unwanted lines, and locating the plate in a binarized image. While such approach has very low complexity at plate detection, it does not provide a solution towards a complete LPR system

Hsu et al. [31] propose a complete three-staged LPR architecture designed around edge clustering for plate detection and maximally stable extreme region for character segmentation. They argue that different types of LPR applications (e.g., access control, road patrol, etc.) differ in the range of values relative to the variables (pan, tilt, etc.) characterizing typical application input data. They show that an ad-hoc solution accounting for an application range of variables outperforms other solutions agnostic of the particular application parameters. Clearly, the main drawback of such solution is that one distinct solution is required for each considered task.

Li et al. [32] employ a combination of different networks to tackle the LPR problem from a slightly different perspective. First they use a CNN trained on the datasets created by [33] and [34] to possibly detect characters in the input image. The candidate image areas are classified as plate or non-plate via a second CNN trained

on the AOLP dataset [31] via cross-validation to discard false positives. Finally, a long short-term memory recurrent network trained on the same character set is used to label the characters as a textual sequence, avoiding the otherwise critical character segmentation step. Despite the use of three different networks, this approach shows improved results on the same dataset used in [31].

Yuan et al. [35] propose a low-complexity plate detection architecture. First, a line density filter locates candidate license plates regions. Next, a license plate classifier discards false positive regions on a color salience basis. While the proposed approach achieves remarkable recall, precision may suffer due to lower resilience to false positives.

Jiao et al. [36] address the issue of localizing plates of different appearance (plate aspect, character format, etc.) across different nations via a tunable algorithmic approach. We point out however that our synthetically trained LPR system is able to cope with different plate aspects simply by changing the template used to generate the training images.

Gou et al. [37] propose a LPR algorithm based on Extremal Regions and Restricted Boltzmann Machines. First a coarse detection of license plates is performed using edge detection and image filtering. Characters regions are then extracted using Extremal Regions which in turn are used to refine the plate region. Finally the characters are recognized using hybrid discriminative restricted Boltzmann machines trained on character samples extracted from real photos augmented by rotation and noise. However, plate localization deals only with plates of specific size and aspect ratio and is not demonstrated to be robust to changes in perspectives.

Bulan et al. [38] propose to use a weak and sparse classifier followed by a strong CNN to sort out readable license plates. For character recognition, they avoid the segmentation step using a sweeping SVM classifier and a hidden Markov model to infer their locations. The character classifier is trained either with real samples labeled by an already working classifier or on synthetic data. However, the experiments show a performance loss when the network is trained on their synthetic data.

Meyer et al. [23] explore the problem of what makes good synthetic training data for deep learning in the tasks of disparity and optical flow estimation. Their finding is that while sample diversity and camera knowledge helps, image realism is overrated, a result that is also supported by our experimental results in the considered case of license plate recognition.

3. Generation of synthetic training images

In this section, we describe a process to generate synthetic license plate images suitable to train a supervised learning algorithm for license plate detection such as the CNN architectures that we will introduce in the next section. We recall that the problem we intend to tackle by synthetic training is the cost associated with collecting a number of real license plate images large enough to train a CNN (in the order of several thousands of images) as well as annotating them by hand (for each image, annotating plate corners position, characters and characters positions). For this reason, we propose to resort to synthetic, computer generated, license plate images in the place of real images for training our CNNs. The advantages of resorting to synthetic training images are that i) an arbitrarily large number of images can be generated and ii) all the required annotations are implicit, lifting the need for manual annotation. In detail, this section focuses on the process of generating synthetic license plate images; the process for extracting the exact training *samples* used to actually train our CNNs for license plate and character detection will be detailed later on in Section 6 together with the training procedures.

The synthetic training images process generates *positive* or *negative* images and is organized into a sequence of steps. Each step models at least one of the plate geometric (size, perspective) or environmental (light conditions, acquisition noise) variables that are responsible for the high intra-class variance of real-world license plate images, which is the ultimate reason why LPR in natural light is so challenging. Our goal is not to model the variable distribution found in real world images, which is often unknown and may vary significantly depending on the scenario. Rather, our goal is to generate a large number of challenging training images suitable to train a network to generalize well across widely different scenarios. Therefore, for each image, we independently draw the variables driving the plate appearance from a uniform distribution. Our hypothesis, as supported also by recent research [23] is that photo realism is not strictly necessary to train a discriminative algorithm, as we will also experimentally verify in Section 7.

3.1. Positive images generation

Positive images contain a synthetic plate over some random background, the plate is entirely contained within the image. Negative images are such that they contain random text on the same random background. Keep in mind that we do not intend to produce realistic images, rather we aim to produce sufficiently challenging image set of enough variance that the trained classifiers also cover all variances of natural images.

2D Template Generation. The process to generate a positive image starts from a synthetic blank plate template, as shown for example in Fig. 1 (left). The template reflects the actual aspect ratio and color scheme of actual Italian car plates. Next, a random sequence of characters reflecting the actual plate numbering scheme is superimposed on the template, reflecting the actual text layout of real plates Fig. 1 (center).

2D Template Alteration. For each positive image, the font thickness is randomly altered to reflect the actual variability observed in photos of real plates, as shown in Fig. 1 (right). Also, each character is randomly shifted by a few pixels along the horizontal and the vertical axes to improve the plate variability. Next, plate properties are randomly altered to reflect changes in illumination angle and base color (Fig. 2 left). Coherent reflections and shadows are added around the character edges to account for the fact that characters in actual plates may be embossed, Fig. 2 (center). Then, a partial shading of variable intensity is randomly added to the top or to the side to account for partial occlusions generated from the vehicle body. An example of a resulting synthetic plate is shown in Fig. 2 (right).

3D Perspective Projection. The plate template is then rotated in a three-dimensional space to simulate the observation of the plate from a random perspective. The parameters governing the rotation over the three canonical axes are evenly distributed in ranges such that the plate is still readable by human eye. Namely, the pan in

Fig. 1. (left) Blank template. (center) Template with random license. (right) Font thickness altered.

Fig. 2. (left) Random color variation. (center) character shade. (right) partial shading due to occlusions.

Fig. 3. (left) Small rotation applied. (center) Rotation and pan independently applied (right) Rotation, pan and tilt independently applied.

Fig. 4. (left) Lightness channel after averaged with background. (center) Plate after color shifts in HSL color space. (right) Plate superimposed on background.

Fig. 5. (left) Synthetic and (right) negative plate images.

the [-80, 80] degrees range and the tilt in the [-40, 40] degrees range, while the rotation is limited to the [-5, 5] degrees range (due to mounting restrictions and the expectation that the camera is held horizontally, otherwise this range should be increased). The plate template is then projected back to a two-dimensional surface using perspective projection, so that the plate width ranges in the 75–200 pixels interval, an example of the projected plate after rotation, pan and tilt are applied is shown in Fig. 3.

Imaging Variations and Noise Addition. Next, we select a random background image of 768 × 384 pixels, convert from RGB to HSL color space and average the lightness channel of the image with the lightness of a heavily smoothed random background patch to simulate varied illumination on the image, similar to what occurs in the background image, see Fig. 4 (left). After that, all three HSL channels are randomly shifted and stretched or compressed slightly to simulate color variations from illumination, production, aging and different camera and acquisition settings before converting back to RGB, Fig. 4 (center). Then, the plate template is superimposed and centered on the background image, Fig. 4 (right). As a last variation, some motion blur to simulate a moving object is randomly added, as well as pixelated noise to simulate dirt and low light conditions and Gaussian blur to simulate imperfect camera focus, as shown in Fig. 5 (left).

3.2. Negative images generation

The process to generate a negative image is similar. However, negative images are generated superimposing a string of random text rather than a plate template over the background image, see Fig. 5 (right). The superimposed text aims to teach the network that not any text string represents a vehicle plate and thereby to decrease false positives. Each negative image uses the same background image as a positive to ensure that irrelevant background features are equally represented regardless of image class.

4. Convolutional networks design

This section first discusses the design principles for a Convolutional Neural Network (CNN) suitable for joint classification and localization (*detection*, in the following) of generic objects. Next, we propose distinct embodiments of such principles for the tasks of plate and character detection.

4.1. Convolutional networks design principles

The image is first processed by a common *feature* extraction stage with multiple convolutional layers, each followed by Rectified Linear Unit (ReLU) activation functions and max-pooling for feature

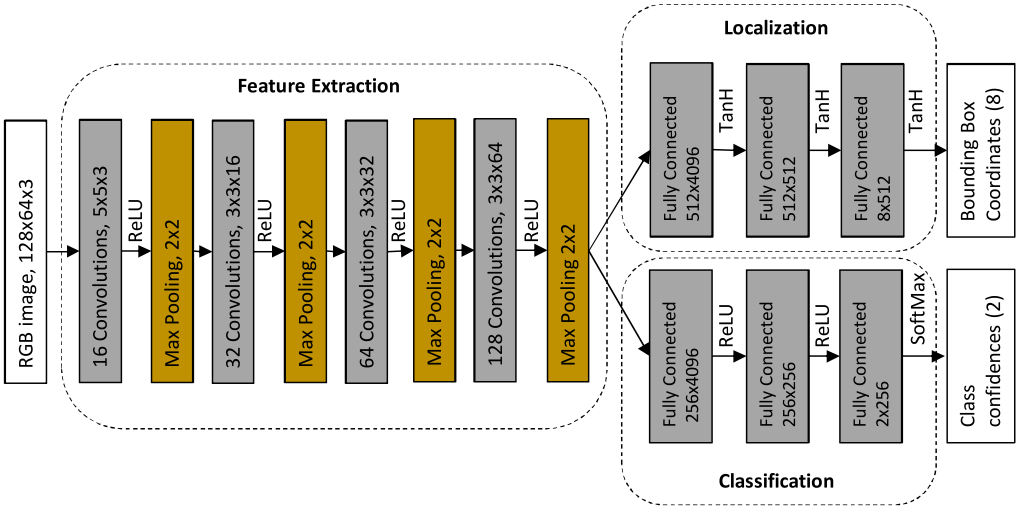

Fig. 6. Our proposed CNN architecture as dimensioned for plate detection.

map subsampling. The resolution of the feature map produced by each convolutional layer is downsampled via a 2×2 max-pooling [6].

Concerning the number of filters per convolutional layer, a common design rule is that after each downsampling the following convolutional layer will have twice as many filters as the previous layer. The last convolutional layer eventually yields a number of feature maps that are provided as input to two different sets of fully connected layers acting as object localizer and classifier respectively. In fact, features relevant for object localization are also likely to be relevant for classification (and vice-versa), reducing complexity with respect to a design with distinct independent object localization and classification [39].

The *classifier* stage receives the extracted features and predicts which out of C possible classes the input image belongs to. The classification stage is composed of three fully connected layers with ReLU activation functions after the first two while the last is followed by a SoftMax activation function leading to the output of C units, to account for a one-hot output encoding scheme.

The *localizer* stage shares the inputs with the classifier stage and is a regressor that predicts the (x, y) coordinates of the vertices of a bounding box to localize the object in the input image. The localizer is also composed of three fully connected layers, all with hyperbolic tangent activations. The output consists of 8 units, each in the $[-1, 1]$ interval, corresponding to the 4 (x, y) pairs of coordinates describing a bounding box.

4.2. Plate detector network

We now adapt the above design principles to jointly predict whether or not a 128×64 patch (suitable to accommodate EU plates) contains a plate (binary classification problem) and the corresponding coordinates of the bounding box (localization problem). The feature extraction stage includes 4 convolutional layers, with 16, 32, 64 and 128 filters of sizes 5×5, 5×5, 3×3 and 3×3 respectively. Therefore, the last convolutional layer yields 128 8×4 feature maps for a total of just 4096 features, keeping complexity bounded. The classifier stage hidden layers have 256 units each and the output layer has $C = 2$ units, accounting for plate and background. The localizer stage hidden layers have 512 units each and the output layer has 8 units accounting for the 4 pairs of coordinates of a plate located at an arbitrary position in the image. Overall, the plate detector network has about 100k learnable parameters in the feature extraction stage, 1.1M parameters in the classifier and about 2.4M in the localizer, for a total of about 3.6M parameters as shown in Fig. 6.

4.3. Character detector network

Similarly, we adapt our design principles for the complementary problem of character detection. Characters used in vehicle plates are typically taller than wider, therefore we use a 24×40 input window. The feature extraction stage includes 3 convolutional layers with 32, 64 and 128 filters of sizes 5×5, 3×3 and 3×3 respectively The last convolutional layer yields 128 feature maps of size 6×10 each, for a total of 7680 features. The classifier stage has two hidden layers with 512 and 256 units respectively, whereas the output layer $C = 33$ units for the Italian plates, $C - 1$ for the encodable plate characters plus one to indicate that no character was found in the image (C_{null}, in the following). Typical C values for actual plates are usually between 10 and 40. The localizer stage consists of two hidden layers having 128 and 64 units respectively, while the output layer has only 4 neurons. With respect to the 8-neurons plate localizer, the character localizer only determines a rectangular bounding box around a character by its upper-left and lower-right corners. In fact, once the exact location of the plate in the image is known, the plate is transformed to a pose where the plate and the letters inside are parallel to the normal axes, as we detail in Section 5.3. Overall, the character detector network has about 95k learnable parameters in the feature extraction stage, about 4.1M parameters in the classifier and about 1M parameters in the localizer, for a total of about 5.2 million parameters.

5. LPR system architecture

Fig. 7 illustrates the architecture of our LPR system, which is organized as a cascade of five distinct processes (and relative subprocesses).

5.1. Scale-pyramid representation

Our LPR system is designed to achieve invariance even to extreme scale changes (e.g., surveillance cameras resolution ranges from full HD to VGA, and a plate size in the image may range from tens to hundreds of pixels depending on the camera-plate distance; to this end, we rely on a scale-pyramid approach where we repeatedly downsample the image such that the height and

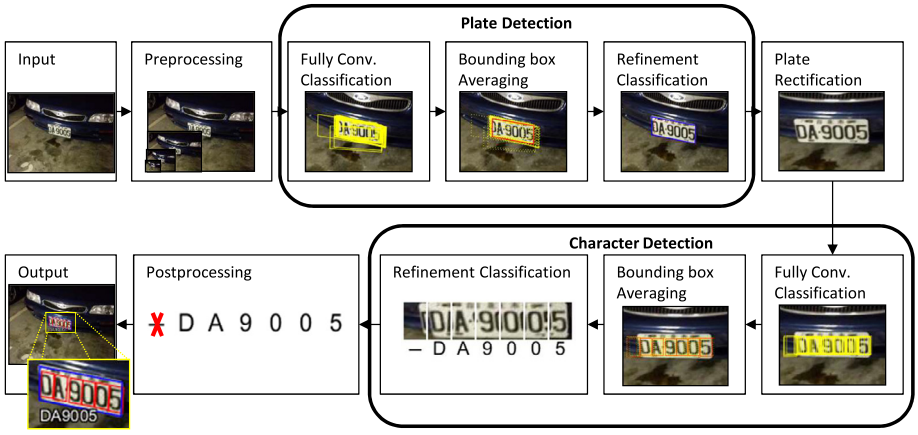

Fig. 7. The proposed LPR architecture pipeline as a cascade of five different processes.

width of the $(n+1)^{th}$ layer is half that of the n^{th} layer, so that each plate has the right size to be detectable at least in one scale. Notice that R-CNNs [40] provide some robustness to scale variations via ROI pooling. However, the scale ranges at which they can reliably localize objects is bounded and depends on the network topology and on the training samples. Conversely, our design is scale-invariant without depending on the network topology and training samples. Furthermore, such architectures are designed for producing loose, category agnostic, bounding boxes, while our approach yields plate-tight bounding boxes enabling rectification, described in Section 5.3. finally, the scale-pyramid approach increases the complexity only slightly, as the total additional area of the n downsampled layers does not exceed 33% of the original image area.

5.2. Plate detection

Plates are detected via three main sub-processes.

Localization and classification. Each layer of the pyramid is independently fed to the plate detector CNN described in Section 6.3. The network includes 4 pooling layers, so input windows are shifted by 16 pixels, vertically or horizontally. The network returns one response for each of the possible 128 × 64 windows spanning over each layer. For each window, the network predicts if it contains a plate as well as the coordinates of its vertices. In the following, we define as *platebox* the output of the network for a given input window. A platebox defines i) the confidence of the window contains an entire plate and ii) the predicted coordinates of its vertices.

Redundancy reduction. As input windows overlap (the stride is 16 pixels), a plate may appear within adjacent windows in a layer and within up to two adjacent layers, i.e. we may have multiple overlapping plateboxes for the same plate. Therefore, partial platebox views are discarded and redundant plateboxes are merged. First, plateboxes with a confidence lower than a threshold Θ_1^p are discarded. Next, for the remaining plateboxes, we compute for each pair the overlap defined as the Jaccard index, i.e. Intersection over Union (*IoU*). The pair of plateboxes with highest overlap above an ω^p threshold is replaced with a new platebox with averaged coordinates. The procedure is iterated until no pair of plateboxes overlapping more than ω^p exists.

Refinement. Finally, the vertices coordinates of each platebox are refined via a further application of the plate detector network. Plates detected in the (scaled) image were likely not centered in the plate detector CNN receptive field (the plate detector stride is 16 px). CNNs such as that in Fig. 5 are more precise at object classification or localization if objects are well centered in the CNN receptive field. So, for each platebox, we crop a rectangular patch from the image and rescale it to 128 × 64 (i.e. the original network input size), assuring that the plate is centered inside the platebox and feed it to the plate detector CNN again. If the updated platebox prediction is below a threshold Θ_2^p, the platebox is discarded. The complexity of this refinement is negligible compared to the fully convolutional search over the image. Our experiments revealed that refinement is especially effective towards improving plate localization accuracy, the latter being of pivotal importance for accurate plate rectification as described below.

5.3. Plate rectification

Plates in images may appear under any arbitrary perspective, which would complicates character recognition later on. So, we rectify the plate, i.e. we apply a geometric transformation which maps each platebox to a rectangular area of predefined size and perspective.

First, we estimate the corresponding transformation parameters: borrowing the notation of [41], let us indicate the (x, y) coordinates of the four corners of the platebox detected by the network in the image (platebox) as

$$X^s = \begin{bmatrix} x_1^s & x_2^s & x_3^s & x_4^s \\ y_1^s & y_2^s & y_3^s & y_4^s \end{bmatrix},$$

where the subscripts indicate the corner indices and the coordinates are normalized with respect to the image size. Now, let us indicate the normalized coordinates of the corners of a corresponding target platebox in homogeneous coordinates as

$$X^t = \begin{bmatrix} x_1^t & x_2^t & x_3^t & x_4^t \\ y_1^t & y_2^t & y_3^t & y_4^t \\ 1 & 1 & 1 & 1 \end{bmatrix},$$

where X^t is such that i) the target platebox corners are located at predefined, known a priori, positions ii) the target platebox edges are parallel to the canonical axes. Then, we find the transformation matrix

$$A_t = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{32} \end{bmatrix},$$

such that $X^s = A_t X^t$. We recall that the source platebox coordinates, X^s are known (predicted) and the target platebox coordinates X^t are fixed in advance. Hence, $X^s = A_t X^t$ represents an overdetermined system of 8 equations and 6 unknown variables (the elements of A_t). Therefore, we find the least squares solution given by

$A_t = X^s(X^t)^+$, where $(\bullet)^+$ indicates the Moore-Penrose pseudoinverse.

Once the parameters of the geometric transformation are known, the RBG values of each pixel in the target platebox can be computed from corresponding pixels from the source image as follows. Let us indicate the position of the i-th pixel in the source platebox $[x_i^s\ y_i^s]^T$ and the position of the corresponding pixel in the target platebox as $[x_i^t\ y_i^t\ 1]^T$. For each pixel within the target platebox, we find the corresponding coordinates in the source platebox as $[x_i^s\ y_i^s]^T = A_t[x_i^t\ y_i^t\ 1]^T$, linearly interpolating nearby pixels.

5.4. Character detection

Each rectified plate patch is independently fed to the character detector CNN described in Section 5.3. The network input window is 24 × 40 pixels so the 20 × 30 characters produced during plate rectification fit within the CNN window with some tolerance.

Localization and classification. we feed each rectified plate patch to the character detector CNN, which returns a localization and classification response for each of the input windows spanning over the rectified plate. We define as charbox the network response at a given input window position. A charbox defines i) the confidence that the window (entirely) contains each of the considered C character classes and ii) the predicted bounding box.

Redundancy reduction. As for plateboxes, we discard charboxes for which the highest confidence class is below threshold Θ_1^c or is the C_{null} class (no complete character is found inside the input window). Next, redundant boxes are merged by means of the same iterative process described in Section 5.2, until no pair of charboxes exist such that $IoU > \omega^c$.

Refinement. charboxes are refined via a second application of the character detector network similarly to Section 5.2. For each charbox, we crop a patch from the rectified plate, centered on the (merged) charbox center. The patch is cropped to include some margin relative to the height and width of the charbox and is then resized to 24 × 40 pixels. Each patch is fed to the character detector CNN and charboxes with confidence below Θ_2^c are discarded. Charbox refinement is especially effective towards improved character classification accuracy and ruling our false positives (e.g., background signs or marks in the image), as we experimentally show later on.

5.5. Postprocessing

Finally, we exploit the knowledge of i) the number N of characters in the plate and ii) of the letters-digits pattern (if any) to rule out false positives. Let us assume that the character detector returns a *word* of M characters. If $M > N$, up to $M - N$ spurious characters may have been detected in the plate (for example, background clutter). Namely, we have M *choose* N possible words of N characters each, called words in the following. Let us indicate as w_i the i^{th} word of N characters, where $w_i = \{c_{i,1}, c_{i,j}, c_{i,N}\}$ and $c_{i,j}$ indicates the j^{th} character in the i^{th} word, where the characters are sorted in left-to-right order as localized in the image. Let $P\{c_{i,j}\}$ indicate the maximum probability that $c_{i,j}$ is a valid character, as predicted by the plate detector. Furthermore, let $l_{i,j}$ be 1 if $c_{i,j}$ character is allowed to appear at the j^{th} position in the i^{th} word according to the considered coding scheme, otherwise 0. For example, for Italian and other EU plates, N=7, the three central characters shall be digits and the remaining characters shall be letters. Let

$$P\{w_i\} = \prod_{j \in 1,N} c_{i,j} l_{i,j}.$$

be the probability that the i^{th} word is correct. So, for each plate we return the word with top score. Note that for $M = N$, we just have one possible reading. Conversely, if $M < N$, the character detector has not detected enough characters in the plate and the classification is discarded as incomplete (e.g. occlusions in the image). Such postprocessing scheme is key to improve the LPR system performance by ruling out false positives, as we experimentally show later on.

6. Sample selection and training

In this section we first describe how we extract the samples required to train the plate and character detector CNNs from the synthetic images we generated. Next, we propose a cost function for jointly minimizing the localization and classification error at training time. Finally, we detail the procedures for training the plate and character detector CNNs.

6.1. Sample generation for training the plate detector

The synthetic images we generate according to the procedure in Section 3 have a resolution of 768 × 384 pixel and the width of the plates ranges between 75 and 200 pixels. However, our CNNs for plate detection is conceived to process images sized 128 × 64. Therefore, we extract sample patches of 128 × 64 pixels from such images as follows. We preliminarily draw pairs of sample images, such that each pair contains one positive and one negative image. Then, we downsample each pair of images so that the width of the plate in the positive image ranges between 37 and 100 pixels. We downsample the negative image by an identical scale factor. Next, one positive sample is generated by cropping a patch at random from the positive image such that the plate is entirely contained within the sample image with a 5 pixels margin from the sample edge. Similarly, one negative sample is generated from either the positive or the negative image so that it contains either a non-complete view of a plate or no plate at all (e.g., only background). So, for each negative sample, we randomly crop with equal probability a random patch either from the negative or the positive image. Patches from the positive images are cropped at random, with the constraint that no more than 50% of the plate is included within the patch. In this way, for each pair of synthetic images we generate one positive and one negative 128 × 64-sized training sample suitable for training the plate detector network. Fig. 8 shows examples of the samples used to train the plate detector.

6.2. Sample generation for training the character detector

In order to train the character detector CNN described in Section 4-C, we extract 24 × 40 character sample patches from the previously generated 768 × 384 synthetic images. Such samples are extracted to reflect the characteristics of the images the character detector is going to actually process when deployed on the field within our LPR system. Namely, in Section 5.3 we explain how we rectify the detected plates according to the location of the plate corners as predicted by the plate detector network. To this end, we first rectify our synthetic images according to the bounding boxes produced by our trained plate detector network rather than according to the ground truth boxes. Therefore, the character detector will be trained to detect characters samples more similar to those it will actually see when deployed on the field within our LPR system. After rectifying the synthetic plate images, we crop positive character patches from each positive images. In this context, a character sample is positive if the character is entirely contained in the sample. The patches are cropped and rescaled so that the character in the patch are 16–40 pixels high and 10–24 pixels

Fig. 8. Examples of (top) positive and (bottom) negative samples used for training the plate detector CNN.

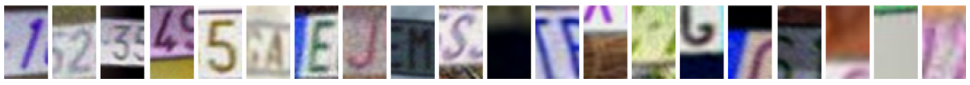

Fig. 9. Examples of positive (10 leftmost) and negative (10 rightmost) samples for training the character detector CNN.

wide and randomly positioned within the crop without any margin. Negative character samples are also cropped from the positive image. In this context, a character sample is negative if it contains no character at all or just a partial view of the character, which the character detector will frequently encounter when deployed on the field. To ensure a character is not randomly included we check that the IoU with any character never exceeds 50%. We extract the character samples so that the number of positive samples is identical to the number of negative samples and each class of characters is evenly distributed in the training set. Overall, we extract 450,000 positive and 450,000 negative samples. Examples of positive and negative training character samples are shown in Fig. 9.

6.3. Target cost function definition

The cost function we optimize for training our CNNs is defined as follows. Let us define the i^{th} training sample as x_i, the network response (output) to x_i as y_i and the corresponding expected (target) output as t_i. The output of our proposed CNN architecture described in Section 4 is composed by the output of the classification branch and the output of a localization branch, we further define the network response to x_i as $y_i = \{y_i^c, y_i^l\}$, where y_i^c and y_i^l are the classification and localization components of the network response respectively. Similarly, we define $t_i = \{t_i^c, t_i^l\}$. Next, we define the cost function for the i^{th} sample as (henceforth we omit the i subscript for the sake of readability)

$$J(w, y, t) = aL^c(w, y^c, t^c) + (1-a)L^l(w, y^l, t^l) + \lambda R(w), \quad (1)$$

where w represents the learnable parameters (weights and biases) of the network.

The component $L^c(w, y^c, t^c)$ represents the network error (or loss) on the c-classification branch outputs of the network from sample x_i, and is defined as

$$L^c(w, y^c, t^c) = -\sum_{k=1}^{C} t_k \log y_k.$$

That is, the error on the classification branch of the network is defined as the cross-entropy between y_i^c and t_i^c. Likewise, the component $L^l(w, y^l, t^l)$ represents the corresponding network error on the localization branch, defined as

$$L^l(w, y^l, t^l) = 1/L \sum_{k=1}^{L} (y_k - t_k)^2.$$

That is, the error on the localizer branch is defined as the Mean Square Error (MSE) between y_i^1 and t_i^1. The factor a drives the balance between the contributions of the localization and the classification errors to the overall cost function. Finally, the component $R(w)$ represents an optional regularization term that helps preventing overfitting to the training samples and is defined as the squared L2-norm of all the weights in the network.

6.4. Training procedures

In order to train the plate detector CNN, we first generate 40,000 synthetic images (20,000 positive and 20,000 negative images) of Italian vehicle plates using the procedure we described in Section 3. We also generate 5000 distinct additional images (2,500 positives and 2500 negatives) for validation purposes. Each pixel in the image is normalized according to the pixel mean and standard deviation values computed over the training images to accelerate learning.

We feed batches of 128 samples (64 positives, 64 negatives) to the network and then we compute the gradient deltas minimizing the cost function 1 for each i-th training sample. Next, we compute an average set of weights differences over the batch and update the weights using the AdaGrad [42] algorithm for learning rate adaptation. Our experiments suggested that an initial learning rate of 10^{-4} is a good compromise between convergence speed and probability to overshoot the optimal solution. After each epoch, we test the trained network on a validation set of 5000 synthetic images: the experiments showed that the average validation error derivative becomes zero after about 400 epochs. Concerning the cost function parameters, we experimentally found that $a = 0.1$ and $\lambda = 10^{-6}$ best minimize the overall validation error when training the plate detector CNN.

The CNN responsible for character detection is trained according to the same procedure described above for the plate detector network. However, in this case we use batches of 512 samples each and we experimentally found that the error function derivative becomes zero after about 300 epochs. Concerning the cost function parameters, we set $a = 0.6$ and $\lambda = 10^{-6}$ when training the character detector CNN.

Finally, to prevent the units in the convolutional layers from learning features that are useful only in combination with other features (feature co-adaptation), we found a dropout [43] factor of 0.5 useful in the last convolutional layer. Also, we found it useful to add a batch normalization layer after each convolutional layer to speed up the convergence of the training process.

Notice the trained networks shall be converted to a fully convolutional form [5] prior to their practical deployment. Namely, units in the first fully connected layer in the classification and localization branches are rearranged into a convolutional layer with filter

Fig. 10. A few samples from the platesmania.com database. Original resolution: 0.3 - 3 Mpixels.

Table 1
Classification performance of our LPR system with different modules disabled for Italian plates [%].

Task	LPD			LPR		LPDR		
Method	PR	RE	IoU	Char	Lic	PR	RE	ACC
Proposed	99.9	99.0	85.1	99.7	99.6	99.6	98.7	98.3
No plate refinement	94.7 (−5.2)	94.4 (−4.6)	75.4 (−9.7)	99.4 (−0.4)	99.5 (−0.1)	94.2 (−5.3)	94.0 (−4.7)	88.9 (−9.4)
No plate rectification	–	–	–	99.0 (−0.8)	99.1 (−0.5)	98.9 (−0.6)	96.4 (−2.3)	95.4 (−2.8)
No character refinement	–	–	–	82.6 (−17)	62.3 (−37)	60.7 (−39)	47.1 (−52)	36.1 (−62)
No post-processing	–	–	–	95.6 (−4.1)	80.2 (−19)	76.1 (−23)	98.5 (−0.2)	75.2 (−23)

size equal to the dimension of the output features from the last feature extraction layer and the following fully connected layers are converted to 1×1 convolutions.

7. Experiments

In this section, we experiment with a Torch [44] implementation of our synthetically-trained[1] LPR system on different sets of real images using the three tasks defined in [31,32].

1. The *License Plate Recognition* (LPR) task aims at assessing the LPR system performance at character recognition and is further subdivided in two sub-tasks. The *Character Recognition* subtask consists in recognizing each single character in each plate. Considering the Character Recognition subtask, let us indicate the number of true positive character recognitions as *TP* and any incorrect classifications (wrong character, missed character or background as a character) as *F*. Then, we define the license plate recognition accuracy as $TP/(TP+F)$. The *License Recognition* subtask consists in recognizing all the characters in each plate. Let us define a true positive (*TP*) if all the actual plate characters and only the actual plate characters are correctly recognized, otherwise the recognition is false (*F*). The overall license recognition accuracy over each subset is defined as $TP/(TP+F)$.
2. The *License Plate Detection and Recognition* (LPDR) task aims at assessing the overall, end-to-end, LPR system performance. For this task, we define a true positive plate detection and recognition (*TP*) as i) the plate has been correctly localized within the image with $IoU > 50\%$ and ii) all and only the characters actually in the plate have been correctly recognized. Each positive classification (i.e. a positive plate classification with a recognized character string of the correct format) such that $IoU < 50\%$ or incorrect license classification is counted as an false positive (*FP*). For each correct classification (*TP*) not found, we count it as a false negative classification (*FN*). The LDPR performance is also measured in terms of *Precision* and *Recall*, or as *Accuracy*, as defined in the LPD task.

Notice that the LPD and LPR task definitions are borrowed from [31,32], allowing us to compare results. Finally, to show robustness in the training method, we will use the thresholds $\Theta_1^p = \Theta_2^p = \Theta_1^c = \Theta_2^c = 0.5$ to avoid tailoring the thresholds to a specific trained model instance or a specific application, while also eliminating the need to experiment with real image samples. Similarly, in the following we also use $\omega^p = \omega^c = 0.5$. After all, an $IoU > 50\%$ is a well accepted condition for agreeing localizations. In all of the following experiments we will evaluate our proposed system when trained on synthetic images exclusively. While we would have liked to evaluate our system trained on real images as well, the lack of enough real annotated training samples (which is the motivation originally motivating our work) unfortunately prevented us from performing such a comparison.

7.1. Pipeline process assessment

First, we study how each process of the LPR pipeline in Fig. 7 affects the overall system performance. We extract 1152 images of Italian vehicles captured under a wide range of conditions of light and perspective such as those in Fig. 10 from the *platesmania.com* database, and we assess how plate refinement, plate rectification, character refinement and postprocessing affect the overall system performance. Table 1 shows our LPR system performance for the three experimental tasks defined above. The *Proposed* reference scheme corresponds to our complete LPR pipeline. The four following schemes represent the cases where each process has been independently disabled (bypassed).

Disabling plate refinement in Section 5.2 impairs the LPD task plate localization accuracy by -9.7%. However, the character classifier is robust enough for such dislocations and the LPR accuracy is only decreased by 0.1%.

Disabling plate rectification (projection of the platebox onto fixed coordinates) has obviously no impact on the plate detec-

[1] Our synthetic training datasets are available at: https://ipl.polito.it/software/

tion LPD task, whereas the character reading LPR task performance worsens a lot. When plate rectification is disabled, the character detector CNN is presented perspective-deformed and rotated characters; although the character detector was replaced in this experiment with one trained on un-rectified characters, it is only partially robust. So, plate rectification is confirmed to be important towards simplifying the stint of the character detector CNN. Conversely, disabling the character classification refinement yields a severe performance loss in the LPR and LPDR tasks. Our investigations revealed that character classification refinement is crucial to reject false positive characters, such as lines, bars and partial character views. Also, the classification accuracy of characters improves significantly when characters are properly centered in the CNN input window.

Finally, we disable the postprocessor described in Section 5.5 replacing it with a naïve scheme that discard plates for which less or more than the expected number of characters have been detected. Concerning the LPR task, the single character detection accuracy decreases by 4.1%. As a consequence, the LDPR classification precision (and thus accuracy) decrease by 23%. Such experiment shows that postprocessing is effective at rejecting false positives, i.e. detections that are implausible due to an incorrect syntax for the detected plate type.

7.2. Synthetic variables assessment

Then, we experiment to assess the effect of some of the synthetic images variables described in Section 3 on the LPR system performance. Preliminary experiments confirmed the intuition that some variables such as those related to plate perspective (i.e., pan, tilt, rotation) cause a dramatic drop in performance when their value is fixed rather than being drawn from a random distribution. Therefore, we focus on variables controlling image and variations, whose effect on the overall system performance is less understood. To this end, we generated an extra set of 80k synthetic images where each of such variables was active only for 50% of the images, while the distribution of the other features remain unchanged within each subset. Plate and character detectors were then trained from scratch on each subset. For reference, we also trained on a subset of 40k randomly drawn images with such variables evenly represented.

Table 2 shows the LPDR task accuracy for our Italian plates dataset. Disabling color shifting and motion blur variables yields the largest performance drops, showing the importance of such variables towards accurate plate detection. Disabling pixelated noise accounting for sensor noise yields a negligible loss whereas disabling illumination variation accounting for artificial lights even slightly improves performance. This result can be well explained observing that both variables account for low-light images (e.g., taken at night) and that such images are only seldom found in our dataset. This suggests that our scheme could be further improved if an accurate knowledge of the statistics of the variables of real-world images were available.

Table 2
LPDR accuracy [%] of our LPR system with different synthetic features excluded for Italian plates.

LDPR accuracy [%]	ON	OFF	Δ ON-OFF
Reference	97.6		–
HSL color shift	97.6	95.6	1.97
Motion blur	97.5	96.8	0.71
Pixelated noise	97.6	97.5	0.09
Illumination variance	97.2	97.8	−0.51

7.3. Experiments with AOLP Taiwanese plates

Next, we repeat our experiments with our LPR system on Taiwanese vehicle plates from the *Application Oriented License Plate* (AOLP) database [31]. The database contains 1891 Taiwanese vehicle license plates images annotated with license strings and plate positions. The database is subdivided in three application-oriented subsets, Access control (AC), Traffic Law Enforcement (LE) and Road Patrol (RP) as shown in Fig. 11. The AC subset contains QVGA images of front plates, acquired from slightly above but with little horizontal angle (e.g. by a camera mounted above an access gate). The LE subset contains VGA images of vehicles in traffic, acquired with a large horizontal angle (e.g. by a roadside camera). Finally the RP subset contains QVGA images acquired from a patrolling vehicle (e.g. a mounted or hand held camera).

To classify the AOLP plates, we generated new synthetic training samples following the same procedure described in Section 3 for generating the Italian synthetic plate images, with the results shown in Fig. 12. Next, we train a new plate detector network and character detector network on such images using the procedure described in Section 4.

License Plate Detection Task. Table 3 shows the performance of our system on the AOLP dataset in the LPD task and compares it with [31] and [32]. Our proposed approach outperforms the approach of [31], where the plate location is determined via a single CNN, for every AOLP subset. Interestingly, our approach also outperforms that of [32], which relies on a double plate detection approach where the characters are first detected and then the plate is localized based on the character detector output. The high precision values reported in the table are confirmed by the actual measured *IoU*, which is equal to 85% on average over the three AOLP subsets.

License Plate Recognition Task. Table 4 shows the performance of our proposed system over the AOLP dataset in the LPR task and its two character recognition (Char) and license recognition (Lic) subtasks. The results of [31] and [32] are extracted from Table 8

Fig. 11. *Application Oriented License Plate* (AOLP) database samples: (left) Access Control subset (center) Traffic Law Enforcement subset, (right) Road Patrol subset. Original Resolution: QVGA - VGA.

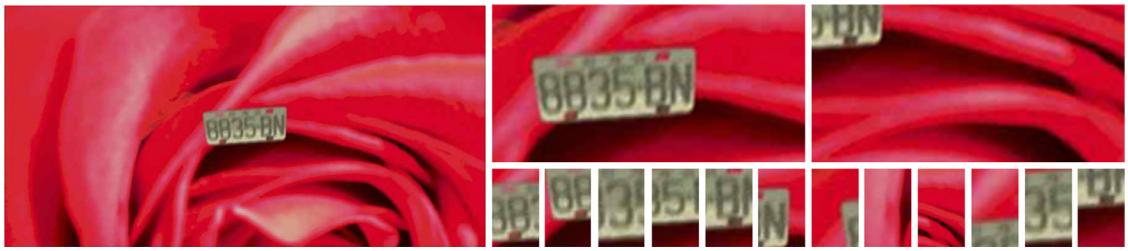

Fig. 12. Taiwanese synthetic training image (left) with corresponding plate training crops (positive: top center, negative: top right) and character training crops (positive: bottom center, negative: bottom right).

Fig. 13. Examples of AOLP images processed by our LPR system. Blue boxes correspond to detected plates, red boxes to detected characters (textual reading is superimposed). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 3
License plate detection (LPD) task, measured as precision (PR), recall (RE) and accuracy (ACC) on the AOLP dataset [%]. A classification is considered positive if $IoU > 50\%$.

Subset	AC			LE			RP		
Method	PR	RE	ACC	PR	RE	ACC	PR	RE	ACC
Hsu [31]	91	96	87.7	91	95	86.8	91	94	86.0
Li [32]	98.5	98.4	97.0	97.8	97.6	95.5	95.3	95.6	91.3
Proposed	**100**	**99.3**	**99.3**	**99.8**	**99.0**	**98.8**	**99.8**	**99.0**	**98.8**

Table 4
License plate recognition (LPR) of single character recognition and full license recognition accuracy [%] on correctly localized plates for the AOLP dataset.

Subset	AC		LE		RP	
Method	Char	Lic	Char	Lic	Char	Lic
Hsu [31]	96	–	94	–	95	–
Li - CNN [32]	98.2	94.0	98.4	92.9	95.6	87.7
Li - LSTM [32]	–	94.9	–	94.2	–	88.4
Proposed	**99.4**	**97.6**	**99.8**	**99.5**	**99.6**	**99.0**

of [32]. Our character detector CNN outperforms both the character detector of [32] based on a CNN and the character detector of [32] based on a LSTM recurrent network. The reason for our system superior performance is the improved ability to correctly classify characters by refinement of already classified characters, as our experiments in Section 7.3 revealed. The LE dataset shows highest performance due to higher image resolution (VGA for LE vs. QVGA for AC and RP) and thus higher plate resolution (85×40 vs. 75×30). For similar image and plate resolution, AC shows lower performance than RP since several images are corner cases where the plate is on the edge of the frame (see Fig. 11, left) and is thus discarded being deemed not completely reliable.

License Plate Detection Recognition Task. The performance of our system over the AOLP dataset in the LPDR task is shown in Table 5. A comparison with the corresponding results over the *PlatesMania.com* database of Italian plates in Table 1 show comparable performance and similar inter-subsets trends as for the LPR task results above due to the LDPR task depending on LPR task. Therefore, this experiment shows that our LPR system is capable of accurate plate detection across widely different types of vehicle plates. Fig. 13 shows a few examples of final classifications of the LPDR task.

7.4. Experiments with PKU Chinese plates

For a final benchmark, we consider the PKU dataset [35], containing about 4000 Chinese plate images divided into 5 subsets captured in increasingly difficult conditions (daylight, daylight with sunshine glare, nighttime, day- and nighttime with reflective glare, and in the final set, images with multiple plates at intersections with crosswalks). An example from each of the first 4 subsets is shown in Fig. 14. To classify these images, we generated new sets of synthetic images of the Chinese plates types contained in the PKU set (blue background and 6 white characters, yellow background and 6 black characters, white background and 5 black characters) as documented in Section 3. Then, we trained plate and character detectors as described in Section 6. We evaluated only the LPD task performance, since the dataset is provided with plate position annotations but not with character annotations. Table 6 shows the accuracy in the LPD task as defined in [35], i.e. a plate is correctly detected if $IoU > 50\%$ with respect to the closest ground truth annotation. While only being on par on the easiest subset,

Table 5
License plate recognition and detection (LPDR) precision (PR), recall (RE) and accuracy (ACC) [%] for our system on AOLP dataset.

Subset	AC			LE			RP		
Method	PR	RE	ACC	PR	RE	ACC	PR	RE	ACC
Proposed	97.6	96.9	94.6	99.3	98.5	97.8	98.8	98.0	96.9

Fig. 14. Examples of PKU images in daylight (G1), daylight with sun-glare (G2), nighttime (G3), nighttime with reflective glare (G4).

Table 6
License plate detection (LPD) task accuracy ($IoU > 50\%$) over the Chinese plates PKU dataset.

Subset	G1	G2	G3	G4	Average
Yuan et al. [35]	98.76	98.42	97.72	96.23	97.78
Proposed	98.77	99.00	98.92	97.74	98.61

our method shows an increased margin down to the reference in the more difficult subsets, with an average of 0.83% better than the reference.

7.5. Computational complexity

As a final experiment, we measure the GPU computational of our LPR architecture in two different scenarios. In the first scenario, a smart surveillance camera with an *NVIDIA Jetson TX1* embedded board (256 CUDA cores) captures mid-resolution images and performs LPR aboard the camera. In the second scenario, a high-resolution camera streams the captured images to a remote LPR server equipped with a *GeForce GTX 1080* GPU with 2560 CUDA cores. We measure the time required by the GPU to apply the plate and character detection CNNs. As a reference, we consider the VGG16-based *Faster R-CNN* [40] architecture for the plate detection process. While such architecture is designed to compete in the large scale *ImageNet* challenge addressing a fundamentally different problem, it is a widely accepted reference for complexity comparisons. Similarly, the R-CNN architecture enables robustness to scale variations and thus it can serve as a reference for our scale-invariant pyramid-based architecture. Table 7 shows that most of the complexity lies in the plate detection process and in the processing of the first, high-resolution, layer of the pyramid of images. Therefore, the relative penalty of our pyramid-based approach to scale invariance with respect to architectures such as [40] is only marginal, *whereas it enables complete scale invariance*. Therefore, our experiments show that our ad-hoc, parameter-savvy, CNN design actually allows deployment over embedded devices and about 1–2 fps frame rate in an embedded scenario and above 20 fps in a server scenario.

8. Conclusion

In this work we proposed a License Plate Recognition (LPR) system designed around Convolutional Neural Networks (CNNs) trained on synthetic samples. Our extensive experiments on three different datasets (Italian, Chinese, Taiwanese) of real images provide a strong experimental evaluation of the effectiveness of training a CNN on synthetic images towards different object localization and recognition tasks. We attribute our system's ability to outperform comparable systems trained on natural images to the complete control over the distribution of the geometric parameters of the training samples that our synthetic approach allows, including the control over elements such as camera noise, and number of images. By comparison, with natural training images the training sample parameter distribution is constrained by the actual acquisition process conditions, which may not necessarily match the deployment scenario conditions, jeopardizing the network's ability to generalize on previously unseen images. We thus conclude that, in terms of training a neural network towards object localization and classification, photo-realistic synthetic training samples are not strictly required to learn the meaning features and distribution of the key parameters for classification of real samples. As a final remark, we add that our ad-hoc network design is lightweight enough to be deployed on devices such as smart cameras, as our experiment with real embedded hardware demonstrated. Our ongoing and future research activities include a large-scale industrial experimentation and validation of the proposed LPR system

Table 7
GPU processing time [ms].

Platform:	Jetson TX1		GeForce GTX 1080	
Resolution	320×240	640×480	640×480	1280×960
Plate classif. (base layer)	101	457	12.5	47.5
Plate classif. (other layers)	26.4	127	5.5	18.0
Plate refinement	23.9		2.2	
Char classific.	191		3.2	
Char refinement	46.2		2.2	
Total proposed	389	845	25.5	72.7
Faster R-CNN (Plate classif.)	–	–	82	96

and improving the analysis regarding which image and noise parameters have most weight towards driving the performance of our system to optimize the sample synthesis process.

Acknowledgment

This work was done at the Joint Open Lab VISIBLE and was supported by a fellowship from TIM.

References

[1] C.-N.E. Anagnostopoulos, License plate recognition: a brief tutorial, IEEE Intell. Transp. Syst. Mag. 6 (1) (2014) 59–67.
[2] B.-H. Chen, S.-C. Huang, An advanced moving object detection algorithm for automatic traffic monitoring in real-world limited bandwidth networks, IEEE Trans. Multimedia 16 (3) (2014) 837–847.
[3] J. Schmidhuber, Deep learning in neural networks: an overview, Neural Netw. 61 (2015) 85–117.
[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE, 2009, pp. 248–255.
[5] M. Mathieu, Y. LeCun, R. Fergus, D. Eigen, P. Sermanet, X. Zhang, Overfeat: Integrated recognition, localization and detection using convolutional networks, 2013.
[6] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[7] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A.Y. Ng, Reading digits in natural images with unsupervised feature learning, in: NIPS Workshop on Deep Learning and Unsupervised Feature Learning, 2011, 2011, p. 5.
[8] M. Jaderberg, K. Simonyan, A. Vedaldi, A. Zisserman, Synthetic data and artificial neural networks for natural scene text recognition, arXiv:1406.2227 (2014).
[9] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg, L. Fei-Fei, Imagenet large scale visual recognition challenge, Int. J. Comput. Vision (IJCV) 115 (3) (2015) 211–252, doi:10.1007/s11263-015-0816-y.
[10] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.
[11] N. Gunji, T. Higuchi, K. Yasumoto, H. Muraoka, Y. Ushiku, T. Harada, Y. Kuniyoshi, The Univ. of Tokyo, ilsvrc2012 scalable multiclass object categorization with fisher based features.
[12] K. Simonyan, Y. Aytar, A. Vedaldi, A. Zisserman, Oxford_vgg @ ilsvrc 2012, 2012. http://image-net.org/challenges/LSVRC/2012/oxford_vgg.pdf.
[13] D.G. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vision 60 (2) (2004) 91–110.
[14] X. Bai, P. Ren, H. Zhang, J. Zhou, An incremental structured part model for object recognition, Neurocomputing 154 (2015) 189–199.
[15] H. Zhang, X. Bai, J. Zhou, J. Cheng, H. Zhao, Object detection via structural feature selection and shape model, IEEE Trans. Image Process. 22 (12) (2013) 4984–4995.
[16] Y. Bengio, I.J. Goodfellow, A. Courville, Deep learning, Nature 521 (2015) 436–444.
[17] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3431–3440.
[18] A.T. Lopes, E. de Aguiar, A.F. De Souza, T. Oliveira-Santos, Facial expression recognition with convolutional neural networks: coping with few data and the training sample order, Pattern Recognit. 61 (2017) 610–628.
[19] X. Bai, B. Shi, C. Zhang, X. Cai, L. Qi, Text/non-text image classification in the wild with convolutional neural networks, Pattern Recognit. 66 (2017) 437–446.
[20] T. Björklund, A. Fiandrotti, M. Annarumma, G. Francini, E. Magli, Automatic license plate recognition with convolutional neural networks trained on synthetic data, in: Proceedings of the IEEE 19th International Workshop on Multimedia Signal Processing, 2017.
[21] H. Su, C.R. Qi, Y. Li, L.J. Guibas, Render for CNN: Viewpoint estimation in images using CNNS trained with rendered 3d model views, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 2686–2694.
[22] A. Gupta, A. Vedaldi, A. Zisserman, Synthetic data for text localisation in natural images, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2315–2324.
[23] N. Mayer, E. Ilg, P. Fischer, C. Hazirbas, D. Cremers, A. Dosovitskiy, T. Brox, What makes good synthetic training data for learning disparity and optical flow estimation? Int. J. Comput. Vision (2018) 1–19.
[24] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in Neural Information Processing Systems, 2014, pp. 2672–2680.
[25] A. Rozantsev, V. Lepetit, P. Fua, On rendering synthetic images for training an object detector, Comput. Vision Image Understanding 137 (2015) 24–37.
[26] H. Caner, H.S. Gecim, A.Z. Alkar, Efficient embedded neural-network-based license plate recognition system., IEEE Trans. Vehicular Technol. 57 (5) (2008) 2675–2683.
[27] S. Yu, B. Li, Q. Zhang, C. Liu, M.Q.-H. Meng, A novel license plate location method based on wavelet transform and EMD analysis, Pattern Recognit. 48 (1) (2015) 114–125.
[28] J.-M. Guo, Y.-F. Liu, J.-D. Lee, License plate localization and character segmentation with feedback self-learning and hybrid-binarization techniques, in: TENCON 2007-2007 IEEE Region 10 Conference, IEEE, 2007, pp. 1–4.
[29] I. Giannoukos, C.-N. Anagnostopoulos, V. Loumos, E. Kayafas, Operator context scanning to support high segmentation rates for real time license plate recognition, Pattern Recognit. 43 (11) (2010) 3866–3878.
[30] A.M. Al-Ghaili, S. Mashohor, A.R. Ramli, A. Ismail, Vertical-edge-based car-license-plate detection method, IEEE Trans. Veh. Technol. 62 (1) (2013) 26–38.
[31] G.-S. Hsu, J.-C. Chen, Y.-Z. Chung, Application-oriented license plate recognition, IEEE Trans. Veh. Technol. 62 (2) (2013) 552–561.
[32] H. Li, C. Shen, Reading car license plates using deep convolutional neural networks and LSTMS, arXiv:1601.05610 (2016).
[33] M. Jaderberg, A. Vedaldi, A. Zisserman, Deep features for text spotting, in: European Conference on Computer Vision, Springer, 2014, pp. 512–528.
[34] K. Wang, B. Babenko, S. Belongie, End-to-end scene text recognition, in: Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE, 2011, pp. 1457–1464.
[35] Y. Yuan, W. Zou, Y. Zhao, X. Wang, X. Hu, N. Komodakis, A robust and efficient approach to license plate detection, IEEE Trans. Image Process. 26 (3) (2017) 1102–1114.
[36] J. Jiao, Q. Ye, Q. Huang, A configurable method for multi-style license plate recognition, Pattern Recognit. 42 (3) (2009) 358–369.
[37] C. Gou, K. Wang, Y. Yao, Z. Li, Vehicle license plate recognition based on extremal regions and restricted Boltzmann machines, IEEE Trans. Intell. Transp.Syst. 17 (4) (2016) 1096–1107.
[38] O. Bulan, V. Kozitsky, P. Ramesh, M. Shreve, Segmentation-and annotation-free license plate recognition with deep localization and failure identification, IEEE Trans. Intell. Transp. Syst. (2017).
[39] R. Girshick, J. Donahue, T. Darrell, J. Malik, Region-based convolutional networks for accurate object detection and segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 38 (1) (2016) 142–158.
[40] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, in: Advances in Neural Information Processing Systems, 2015, pp. 91–99.
[41] M. Jaderberg, K. Simonyan, A. Zisserman, et al., Spatial transformer networks, in: Advances in Neural Information Processing Systems, 2015, pp. 2017–2025.
[42] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, J. Mach. Learn. Res. 12 (Jul) (2011) 2121–2159.
[43] N. Srivastava, G.E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting., J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[44] R. Collobert, K. Kavukcuoglu, C. Farabet, Torch7: A matlab-like environment for machine learning, BigLearn, NIPS Workshop, EPFL-CONF-192376, 2011.

Tomas Björklund received an M.Sc. in Scientific Computing, an M.Sc. in Medical Imaging and a Ph.D. Electronical Engineering in 2007, 2012 and 2018 respectively from Uppsala University, Royal institute of Technology and Politecnico di Torino. His research interests include computer vision and machine learning applications.

Attilio Fiandrotti received his M.Sc. and Ph.D. degrees in Computer Science in 2005 and 2010 respectively. Currently, he is Maître de Conférences at Télécom Paristech. His research interests include signal processing with deep learning techniques for vehicular and mobile applications.

Mauro Annarumma received the M.Sc. degree from Politecnico di Torino, Italy, in 2014. He was a research engineer in the VISIBLE Joint Open Lab of Telecom Italia until May 2016 and is now a machine learning analyst at Kings College London. His research interests include computer vision and machine learning.

Gianluca Francini received the M.Sc. cum laude in Computer Science from the University of Torino, Italy. He is a senior researcher actively involved in image processing. He is currently leading VISIBLE, the lab opened by Telecom Italia. He is co-inventor of 19 patents in the field of image and video analysis.

Enrico Magli received the M.Sc. and Ph.D. degrees from the Politecnico di Torino, Torino, Italy, in 1997 and 2001, respectively. He is currently an Associate Professor with Politecnico di Torino. His research interests include compressive sensing, image and video coding, and vision.