Automatic license plate recognition with convolutional neural networks trained on synthetic data

Conference Paper · October 2017

DOI: 10.1109/MMSP.2017.8122260

5 authors, including:

A. Fiandrotti
Politecnico di Torino
33 PUBLICATIONS **165** CITATIONS
SEE PROFILE

Enrico Magli
Politecnico di Torino
272 PUBLICATIONS **3,263** CITATIONS
SEE PROFILE

Automatic License Plate Recognition with Convolutional Neural Networks Trained on Synthetic Data

Tomas Björklund
Department of Electronics and Telecommunications
Politecnico di Torino
Turin, Italy
tomas.bjorklund@polito.it

Attilio Fiandrotti
Department of Electronics and Telecommunications
Politecnico di Torino
Turin, Italy
attilio.fiandrotti@polito.it

Mauro Annarumma
Telecom Italia
Turin, Italy
mauro.annarumma@telecomitalia.it

Gianluca Francini
Telecom Italia
Turin, Italy
gianluca.francini@telecomitalia.it

Enrico Magli
Department of Electronics and Telecommunications
Politecnico di Torino
Turin, Italy
enrico.magli@polito.it

***Abstract*— We present an Automatic License Plate Recognition system designed around Convolutional Neural Networks (CNNs) and trained over synthetic plate images. We first design CNNs suitable for plate and character detection, sharing a common architecture and training procedure. Then, we generate synthetic images that account for the varying illumination and pose conditions encountered with real plate images and we use exclusively such synthetic images to train our CNNs. Experiments with real vehicle images captured in natural light with commodity imaging systems show precision and recall in excess of 93% despite our networks are trained exclusively on synthetic images.**

Keywords— CNN, ALPR, license plate

I. INTRODUCTION

Automatic License Plates Recognition (ALPR) nowadays finds application in a number of different applications such as toll collection, bus lane occupancy detection, and speeding fining, to name the new. In ALPR, first the captured image is analyzed to detect the presence of plates and to localize their bounding boxes (plate detection). Then, the characters in the plate are read, yielding a textual representation thereof (character detection).

ALPR is a challenging application due to variations in vehicle scale, perspective and brightness, together with motion blur, occlusions and clutter typically encountered in practical applications. Such issues do not affect conceptually similar OCR applications [1], thus prompting for ad-hoc solutions such as infrared illuminators and cameras. In this work, we tackle the challenging problem of ALPR in the case of natural illumination and commodity imaging systems (smartphones, IP-based surveillance cameras) with the goal to practically enable low-cost, large-scale deployment.

Convolutional Neural Networks (CNNs) recently emerged as the cornerstone of a number of record-breaking image classification systems [2]. First, a *feature extraction stage* extracts relevant features from the image via multiple *convolutional layers*. Then, the features are fed to *fully-connected* layers which classify the features among a set of classes. CNNs have also been shown to be suitable for joint object classification and localization in [3]. Further performance improvements were then achieved via smaller filters and deeper topologies [4].

ALPR systems designed around CNNs have been proposed before. In [5] it is proposed to feed the fully connected layers with features extracted at different scales in the convolutional layers to achieve robustness against scale variance. They show good recall, however at the expense of precision. Menotti et al [6] experimented with CNNs and randomly initialized filters, however addressing only the character detection problem and the reported performance barely exceeds 95% and 98% for letters and digits respectively. Interestingly, in all the above approaches, the number and type of available annotated training images constrains the CNN ability to recognize plates and characters, thus limiting the system performance. The authors of [7] address the particular problem of motion blur in character recognition training a CNN over synthetic samples where motion blur has been added on purpose. However, they do not consider other problems typical of ALPR systems nor generalize their approach to train a full ALPR system. Therefore, designing a high-performance yet lightweight

ALPR system around CNNs trained over natural images is still an open issue.

The present work provides several contributions towards an ALPR system designed around CNNs and trained exclusively over synthetic data.

First, we introduce a CNN architecture for generic *object detection*: in the following we use the term object detection to indicate the joint task of *object classification* and *object localization*. The network includes one single feature extraction stage whose output is shared by two separate fully connected stages for object classification and localization respectively, reducing training time and deployment complexity. The network is then adapted to the complementary yet different problems of plate and character detection.

Second, we propose a complete framework to generate synthetic training images accounting for a wide range of illumination and perspective conditions. That is, we avoid the time-consuming task of hand-annotating large numbers of real training images, avoiding copyright and privacy issues. We show that it is possible to train our CNNs over synthetic images via standard backpropagation and well-known optimization algorithms.

Third, we describe a complete ALPR system designed around our two CNNs for plate and character detection. The pipeline leverages fully convolutional CNN embodiments for efficient detection of plates of arbitrary size. We tune the parameters of each component of our ALPR system over a synthetic dataset, separately assessing each component's performance.

Finally, we evaluate the whole system over a dataset of natural images acquired with commodity imaging systems, showing the ability to correctly read plates even in challenging conditions with a precision-recall performance in excess of 93%.

II. Convolutional Networks Design

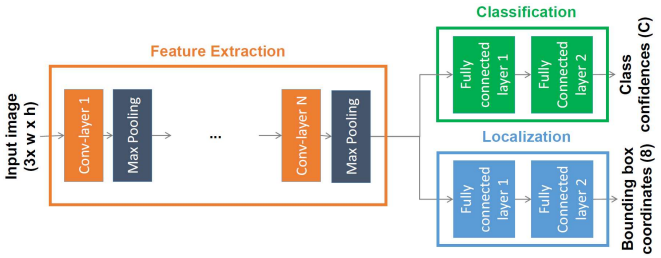

Fig. 1. General CNN architecture for object detection.

In this section we first describe our CNN for generic object detection, where object detection consists in the two sub-tasks of object classification and localization. Then, we detail how to adjust the network hyperparameters to turn it into a plate or a character detector.

A. General Architecture

The network receives as input a fixed size RGB image that is fed to a feature extraction stage composed by a cascade of convolutional blocks. Each convolutional block includes i) one convolutional layer with 3×3 filters and Rectified Linear Units (ReLU) activation functions and ii) max-pooling layers for feature spatial downsampling [4]. The extracted features are forwarded to the object classifier and the object localizer, as features useful for object classification are expected to be useful also for object localization and the other way around.

The classifier stage is a fully connected network predicting if the input image contains an object belonging to the i^{th} class out of C possible classes. This stage has C outputs and includes a *softmax* layer that yields normalized *confidence ratings* for each class. The localizer stage is a linear regressor and predicts the position of the object within the image (namely, the surrounding bounding box vertices). This stage has 8 outputs for the 4 (x,y) coordinate pairs describing the object bounding box vertices.

We now discuss how to adapt the above CNN to the distinct applications of plate detection and character detection.

B. Plate Detector Architecture

The term *plate detection* here indicates the joint tasks of i) predicting if a plate is present in the input image (classification) and ii) localizing the 4 corners of the plate. We consider 128×64 input images to accommodate for the aspect ratio of EU-format plates. The plate detector feature extraction stage consists of 10 3×3 convolutional layers with ReLUs and with 4 Max poolings after the 2^{nd}, 4^{th}, 7^{th} and 10^{th} convolutional layer. The classifier stage consists of 3 fully connected layers and has C=2 outputs, corresponding to the confidence that a plate is present or not in the input image. The localization stage consists of 3 fully connected layers connected to the 8 regression units defining the bounding box surrounding the plate.

The plate detector has 10 convolutional layers in place of 8~16 layers of [4] as the input image size is smaller (128×64 vs. 224×224). Furthermore, the number of convolutional filters is equal to 16, 32, 64, 128 rather than 64, 128, 256, 512 respectively. The dimensions of the plate classification problem are in fact intrinsically different than in [3][4] (C=2 in place of C=1000). Thus, as fewer features have to be learned, we reduce the number of trainable parameters reducing the training complexity (our experiments showed little performance gains when increasing the filter sizes).

C. Character Detector Architecture

Similarly, the term *character detection* here indicates the joint tasks of i) identifying characters within an image assigning a class confidence to each of them and ii) localizing the character bounding box corners. The character detector CNN architecture is similar to the plate detector with some key differences. The input image is smaller, 24×40 pixels, suitable to account for the wide range of characters used in EU plates. The feature extraction stage consists of 7 3×3 convolutional layers with ReLUs and 3 Max Pooling after the 2^{nd}, 4^{th} and 7^{th} convolutional layer.

The classifier stage has 3 fully connected layers and C=33 outputs, one for each of the 32 eligible characters in Italian plates plus one output for the case where no character is detected. The localization stage consists of 3 separated fully connected layers connected to the 4 regression units defining the bounding box surrounding the plate.

The character detector feature extraction stage has 7 convolutional layers, fewer than the plate detector, due to the smaller input image size (24×40 vs 128×64). The number of filters is 32, 64, 128, 256; larger than the plate detector but lower than [4]. In fact, the character detector network needs to learn more features to correctly differentiate images belonging to C=33 classes in part similar to each other (e.g., characters 7, T and 1).

Finally, the CNNs may be converted to fully convolutional for efficient multiple, arbitrary-sized, object detection at arbitrary positions in the image (e.g., plates typically appear with arbitrary size in an image and multiple characters are typically found in a plate) [3]. The first fully connected layer in the classifier and in the localizer are rearranged into a convolutional layer with a filter size equal to the dimensions of the output from the last feature extraction stage (8×4 for the plate detector, 3×5 for the character detector). Next, the following two fully connected layers are converted to 1×1 convolutions. So, the plate and character detection networks are capable of object detection within a number of overlapping *input windows* in the input image of size 128×64 and 24×40 respectively. Table 1 shows that the number of trainable parameters of our CNNs is one order of magnitude lower than [3][4].

TABLE I. NUMBER OF TRAINABLE PARAMETERS AND LAYERS IN PARENTHESES FOR PROPOSED AND SOME REFERENCE ARCHIETCTURES.

Number of parameters	***OverFeat* (accurate model)**	***VGGNet* (conf. C)**	**Proposed (plate detection)**	**Proposed (character detection)**
Convolutional layers	~ 19M (6)	~ 10M (13)	478k (10)	434k (7)
Fully connected layers	~ 125M (3)	~ 124M (3)	~ 9M (3+3)	~12M (3+3)
All layers	~ 144M (9)	~ 134M (16)	~ 10M (16)	~12M (13)

III. DATASET GENERATION AND TRAINING PROCEDURES

In this section, we first describe the procedure to generate synthetic car plate images. Then, we describe how to extract synthetic samples suitable for training the plate and the character detector CNNs introduced in the previous section. Finally, we detail the CNN training procedure.

A. Synthetic Images Generation

First, we generate synthetic images containing either one plate (*positive* images) or no plates at all (*negative* images) .

Positive images are 1024×768 pixels in size and contain one synthetic plate in the center of the image. First, we generate a synthetic 2D plate template matching the EU EC 2411/98 recommendation. Then, we superimpose a random 7-characters string matching the IT numbering scheme (2 letters, 3 digits, 2 letters) using the appropriate font and spacing. We account for the fact that real plates are embossed by adding random reflections and shadows around the characters and adjusting the font thickness. Next, we randomly add different levels of shading to simulate partial or total plate occlusions by other car parts (e.g., plates spotted from narrow angles).

The plate is then projected over a different 2D surface to simulate camera capture from a random angle. The image is then scaled so that the plate width ranges between 80 and 200 pixels, such that the smallest plates are still readable by naked eye while the largest are at least twice as large as the smallest. Since a camera capture will alter the colors (due to e.g. light conditions) the image is converted to HSL-color space and each channel is slightly altered at random. The plate is finally superimposed on a random natural background image with variable transparency factor.

Negative images are equally sized and contain either random natural background or a randomly generated text string with random font and color over the random background to make the plate detector robust to false positives (e.g., spurious text cluttering the scene).

Positive and negative images may be randomly altered adding some blur to simulate motion and noise to simulate scarce illumination conditions.

Notice that by generating synthetic training images, we overcome the problem of acquiring a sufficient number of fully-annotated samples (plate readings and exact position of each plate and character bounding box) for training our CNNs.

B. Plate Samples Selection

The plate detector is trained on 128×64 samples extracted at random from the scaled synthetic images to avoid overfitting. Positive samples are cropped at random so that each sample contains the entire plate. Negative samples are either random crops from negative images or random crops of positive images such that less than 50% of the plate is present in the sample. Such approach makes the plate detector robust to partial plate views, reducing the expected number of false positive detections. We eventually generate 25,000 positive and 25,000 negative samples to train the plate detector.

C. Character Samples Selection

The character detector is trained on 24×40 samples extracted at random from the synthetic images. Positive samples consist in random crops of the synthetic images so that each crop contains one character fitting entirely into the 24×40 character detector input window size. Moreover, crops are randomly rescaled so that the actual character height ranges between 28 and 34 pixels to account for slight character size variations due to perspective changes. Negative samples are either random crops from negative images that do not contain characters at all, or positive images crops such that less than 50% of a character is contained, making the character detector robust to partial character views. Finally, the character detector training set consists of about 175,000 positive and 175,000 negative training samples.

D. Training Procedures

The plate and the character detectors are separately trained for 100 epochs each with random parameter initialization and backpropagation of the learned parameters. The minimized loss function is the Mean Square Error (MSE) over all localization

and classification outputs. We consider batches of 128 samples and at the end of each batch we move towards the averaged steepest gradient direction. The parameters update process is optimized via Adagrad[9], where the initial learning rate value is equal to 5×10^{-2} and the learning rate decay factor is equal to 10^{-3}.

Fig. 2 shows some synthetic training images (top), crops used for training the plate detector (center) and the character detector (bottom).

Fig. 2. Synthetic training images (top) and relative synthetic samples for training the plate dtector (middle) and character detector (bottom) CNNs.

IV. Proposed ALPR System Architecture

This section details an ALPR system architecture (Fig. 3) designed around the two fully convolutional networks for plate and character detection described in Sec. II.

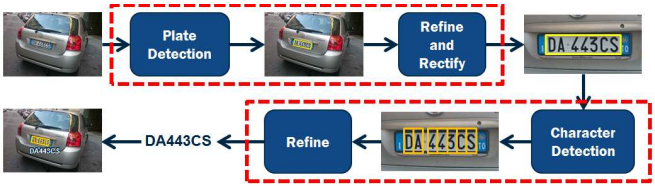

Fig. 3. The proposed ALPR architecture pipeline; dashed boxes represent plate and character recognition stages respectively.

The plate detection stage (dotted box in the top half of the image) predicts the presence of plates in the query image (classification) and the respective positions (localization). The query image is normalized according to mean and standard deviation values computed at training time.

A. Plate Detection

First, the query image is repeatedly rescaled to detect plates of arbitrary size, yielding a pyramid of images such that the $(n+1)^{th}$ layer area is $1/4$ of the n^{th} layer area. The plate detector is in fact trained to reject any plate not fitting entirely within the 128×64 input window. Thanks to rescaling, plates rejected in the n^{th} layer may be detected in the $(n+1)^{th}$ layer. Each image is then independently fed to the plate detector network. For each input window, the network returns the relative bounding box coordinates (*platebox*, in the following). Each platebox is associated with the confidence that it entirely contains a plate or not (C=2). If a confidence is below a threshold θ_1^p, its platebox is discarded (e.g., a partial plate view).

B. Averaging Redundant Classifications

As the plate detector network input windows overlap, the same plate is likely to appear within multiple input windows, yielding redundant plate classifications. We define the *overlap* between two plateboxes ω as their intersection over union (*Jaccard index*). First, we compute ω among all pairs of detected plateboxes, looking for the pair with highest ω and replacing them with an averaged platebox. The algorithm iterates until no pair of plateboxes overlaps more by than a minimum threshold ω^p. Depending on ω^p, the number of detected plates may vary.

C. Refining and Rectification

Finally, we refine the plate detector accuracy with a further re-detection of each (averaged) platebox. Whereas the networks are trained to detect objects over a wide scale, our experiments revealed that the localizer output is most accurate when the object is well centered in input window and scaled above an average training sample. So, we crop the query image around each (averaged) detected platebox, adding some frame around and scale the crop to match the plate detector input size (128×64). Crops containing a plate with a confidence lower than a threshold θ_2^p are discarded.

Since reclassification does not entail a fully convolutional search over the input image, but rather a single classification on a 128×64 patch, it is a lightweight operation from a computational complexity perspective.

After plate detection, we perform an affine transform on each detected platebox to improve the character detector performance. This rectifies the query image so that the predicted corners are at predefined locations, fitting a centered, rectangular template. This process is similar to Spatial Transformer Networks[8], which predicts the parameters of the transformation matrix in an unsupervised manner. However, as synthetic samples have perfect ground truth data for all the plate corners, it is more accurate to train the classifier to predict the corner locations, and then use them to compute the same transformation matrix as [8] predicts.

Depending on θ_1^p, θ_2^p and ω^p, the number of plateboxes returned by the plate detection stage for each plate actually present in the query image may vary.

D. Character Detection

The character detection stage (dotted box in the bottom half of the image) localizes and classifies the characters in each detected platebox independently. To start with, we crop each detected platebox from the rectified image. Next, we resize the crop four times so that the characters height is about 32, 34, 36 and 38 pixels (the character detector input window size is 24×40) to guarantee robustness to varying character size in plates observed from narrow angles. The result is a set of crops of each detected plate, each containing a differently sized plate representation. Then, we separately feed each crop to the character detector network, which returns a set of bounding boxes potentially containing a character (*charbox*, in the following). Each charbox is associated with a confidence ratings of the character belonging to each of the C=33 considered characters. If the highest confidence rated character

class corresponds to the negative class, the charbox is discarded because it is expected to contain no character or a partial view. Otherwise, if the highest rated character class is above a threshold θ_1^c, the charbox is added to a shortlist of candidate charboxes.

As for plateboxes, we reduce the number of redundant detected characters averaging same-class overlapping characters. Namely, we iteratively average the pair of charboxes having the highest overlap ratio given a minimum character overlap threshold ω^c.

Finally, we refine the character detector accuracy re-detecting each (averaged) charbox. We crop each detected charbox from the rectified image leaving some frame around and resize it to match the network input window size (24×40). Charboxes whose highest rating belongs to the empty class or is below a threshold θ_2^c are dropped.

Eventually, plateboxes with more or less than exactly 7 charboxes are dropped (noisy or incomplete). Otherwise, charboxes are sorted left-to-right, yielding the sought textual representation of each detected plate.

V. Experiments

As a first experiment, we find suitable thresholds θ_1^p, θ_2^p for plate detection experimenting on a synthetic dataset with 128×64 sample images.

First, we explore the performance of the plate detector network in isolation as a function of θ^p. Let χ^p be the confidence that the plate detector input window entirely contains a plate, as predicted by the network. We define *True Positives Rate* (TPR) as the ratio of *positive* sample images such that $\chi^p \geq \theta^p$. We define *False Positives Rate* (FPR) as the ratio of *negative* samples such that $\chi^p \geq \theta^p$ (i.e., the ratio of negative samples predicted as positives). Also, let ω^{gt} indicate the plate localizer accuracy defined as the overlap between one detected platebox and the ground truth.

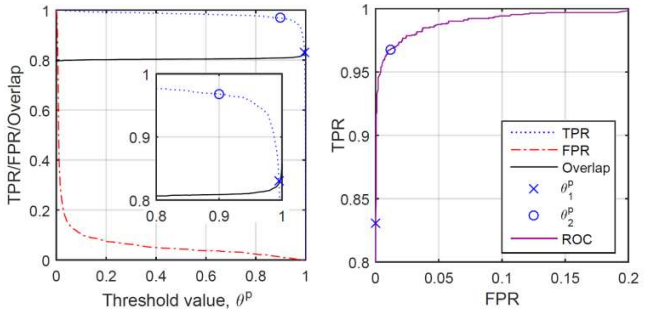

Fig. 4. Plate detector network performance as a function of θ^p (left) and relative ROC curve (right).

The left graph of Fig. 4 shows that the average overlap, ω^{gt} is independent from θ_1^p so, we select θ_1^p and θ_2^p accounting only for plate classification accuracy. Regarding the choice of θ_1^p (plate detection in the query image) most input windows do not contain complete plates views, so we select an high θ_1^p=0.995 to reject such windows and keep the FPR low. Regarding the choice of θ_2^p (plate detection refinement), the need to reject false positives is lower, so we select θ_2^p=0.9, just before the TPR curve starts to drop. The TPR-FPR tradeoff for the selected θ_1^p and θ_2^p is shown in the right graph.

Next, we evaluate the performance of the entire plate detector stage (Fig.3, top dotted box) on 1024×768 synthetic images. We compare our plate detector to the *VLPL* detector proposed by the authors of [6], which relies on techniques such as edge detection, erode and dilate, to filter out plate features such as high contrast text. Fig. 5 shows the ω^{gt}distribution for positive predictions, i.e. input windows such that $\chi_2^p \geq \theta_2^p$. For *VLPL*, 58.8% predictions are such that $\omega^{gt} > 0.5$ (correct prediction) and 27.8% predictions are such that $\omega^{gt} = 0$. The corresponding ω^{gt} distributions for our proposed method are 79.42% and 5.6% respectively. Our method makes no assumptions on the number of plates are made (an average of 1.23 detections per sample were made) and most of the predictions with $\omega^{gt} < 0.5$ are additional classifications in samples where a good prediction has also been acquired (these will most likely be rejected by the character detector), only 3.2% of the samples receive no prediction with $\omega^{gt} \geq 0.5$.

As a second experiment, first we find suitable thresholds θ_1^c and θ_2^c for character detection by experimenting on a synthetic 24×40 characters database. Namely, we explore the performance of the character detector network in isolation using the same method used for the plate detector, finding the thresholds $\theta_1^c = 0.9$ and $\theta_2^c = 0.75$ ($\theta_1^c > \theta_2^c$ for the same reasons).

Next, we evaluate the performance of the whole character detector stage (Fig.3, bottom dotted box) over crops of the synthetic plate images. Fig. 6 shows the accuracy in single character classification as a function of how accurately the plate detector stage has previously localized the plate. The reference case $\omega^{gt} = 1$ assumes totally correct plate localization: in this case, character classification is 97.0% correct. We then add different amounts of uncertainty to the predicted plate localization ($\omega^{gt} < 1$ in the figure). In fact, Fig. 4 (left) shows that the plate detector localizations would be only 80% precise. Fig. 6 shows that the character classifier has little classification accuracy loss for any $\omega^{gt} > 0.65$: comparing such finding with the mean ω^{gt} in Fig. 4 (left) and the actual ω^{gt} distribution in Fig. 5, this experiment suggests that our character detector is expected to be robust to plate localization errors.

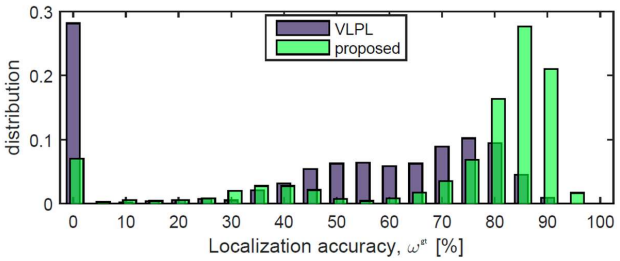

Fig. 5. Plate localization accuracy by average overlap with ground truth.

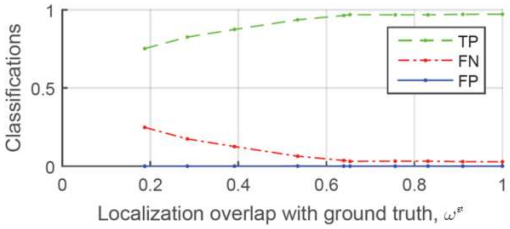

Fig. 6. Character detector classification accuracy depending on the average plate localization overlap with ground truth , ω^{gt} .

As a third and final experiment, we evaluate our entire ALPR system over images from platesmania.com, a categorized database of user-contributed plate images captured using smartphones, and digital cameras with different resolutions, quality, perspective and illumination. Several images are affected by blur due to insufficient illumination or motion. Images may include cluttering due to sharp shades being projected over the plate (Fig. 7). For our tests, we downloaded over 1000 images from Italian cars back plates. Most images include a partial view of the car back and side.

We explore the thresholds ω^p and ω^c controlling when the detected overlapping plateboxes and charboxes are averaged, respectively. We define recall the ratio of images for which our system returned at least one 7-characters textual reading and precision the ratio of correct readings. Fig.8 shows four precision-recall curves, each corresponding one different $\omega^c \in \{0.1, 0.3, 0.5, 0.7\}$. Each curve has 9 points, corresponding to $\omega^p \in \{0.1, ..., 0.9\}$ (0.1 is towards the top-left corner, 0.9 towards the bottom-right corner). As a general trend, ω^p drives the precision-recall tradeoff of the system. We recall that each (averaged) platebox is independently processed by the character detector.

Fig. 7. PlatesMania images in angled perspective, motion blur, poor lightning conditions (top row) with corresponding classifications (bottom row).

High ω^p result in lower chances that plates are averaged and multiple readings of the same plate, which improves recall (more chances that the correct reading is output). However, also the chances that at least one reading is wrong increase and so precision drops. Opposite considerations hold for low ω^p values. The same figure also shows that when ω^c decreases, i.e. when charboxes are more likely to be averaged, the performance increases. In fact, averaging the detected charboxes improves localization and classification of each character during the following refinement step. For $\omega^p = 0.5$ and $\omega^c = 0.25$ our system boasts average precision and recall tradeoff in excess of 93%.

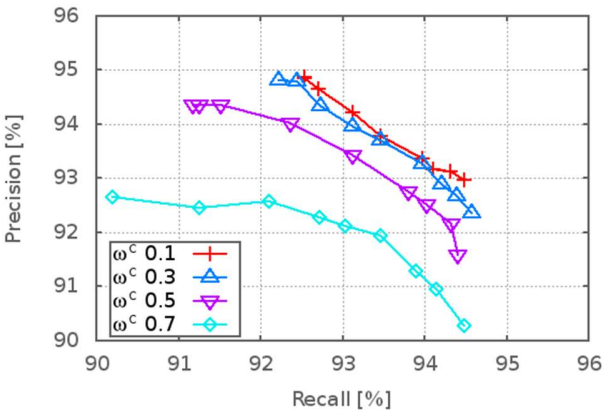

Fig. 8. Overall ALPR performance as a function of the platebox and charbox averaging thresholds ω^p and ω^c.

VI. Conclusions

This work describes an automatic license plate recognition system designed entirely around CNNs trained over synthetic images. A single CNN architecture is described and tuned to solve the complementary problems of plate and character detection. The networks are trained over a synthetic dataset, avoiding labor-intensive real images annotating. We evaluate our system over a dataset of real images taken with commodity imagery systems in natural light conditions. Our preliminary tests show that it is possible to attain precision-recall performances in excess of 93% while training the system over exclusively synthetic images.

References

[1] Y. LeCun, L. Bottou , Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” In Proceedings of the IEEE, 86(11), pp. 2278-2324, 1998.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” In Advances in neural information processing systems, pp. 1097-1105, 2012.

[3] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, “Overfeat: Integrated recognition, localization and detection using convolutional networks,” arXiv:1312.6229, 2013.

[4] K. Simonyan, and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” arXiv:1409.1556, 2014.

[5] J Li, C. Niu, and M. Fan, “Multi-scale convolutional neural networks for natural scene license plate detection,” In Advances in Neural Network, pp. 110-119, 2012.

[6] D. Menotti, G. Chiachia, A. X. Falcão, and V. J. Oliviera Neto, “Vehicle license plate recognition with random convolutional networks,” In 27th SIBGRAPI Conference on Graphics, Patterns and Images, pp. 298-303, 2014.

[7] P. Svoboda, M. Hradis, L. Marsik, and P. Zemcik, “CNN for license plate motion deblurring,” arXiv:1602.07873, 2016.

[8] M. Jaderberg, K. Simonyan, and A. Zisserman, "Spatial Transformer Networks," In Advances in Neural Information Processing Systems, pp. 2017-2025. 2015.

[9] J. Duchi, E. Hazan, and Y. Singer. “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization.” In Journal of Machine Learning Research, 12, pp. 2121-2159, 2011