2014 27th SIBGRAPI Conference on Graphics, Patterns and Images

Vehicle License Plate Recognition With Random Convolutional Networks

D. Menotti, G. Chiachia, and A. X. Falcão
Institute of Computing (IC)
University of Campinas (UNICAMP)
Campinas, SP, Brazil
Email: menotti@iceb.ufop.br,{chiachia,afalcao}@ic.unicamp.br

V. J. Oliveira Neto
Computing Department (DECOM)
Federal University of Ouro Preto (UFOP)
Ouro Preto, MG, Brazil
Email: vantuiljose@gmail.com

***Abstract*—Despite decades of research on automatic license plate recognition (ALPR), optical character recognition (OCR) still leaves room for improvement in this context, given that a single OCR miss is enough to miss the entire plate. We propose an OCR approach based on convolutional neural networks (CNNs) for feature extraction. The architecture of our CNN is chosen from thousands of random possibilities and its filter weights are set at random and normalized to zero mean and unit norm. By training linear support vector machines (SVMs) on the resulting CNN features, we can achieve recognition rates of over 98% for digits and 96% for letters, something that neither SVMs operating on image pixels nor CNNs trained via back-propagation can achieve. The results are obtained in a dataset that has 182 samples per digit and 28 per letter, and suggest the use of random CNNs as a promising alternative approach to ALPR systems.**

***Keywords*-convolutional neural networks, random search, random filters, optical character recognition, vehicle license plate recognition.**

I. Introduction

Automatic license plate recognition (ALPR) has been actively investigated in view of numerous real-world applications such as automatic toll collection, traffic law enforcement, parking lot access control, and road traffic monitoring. In general, an ALPR system consists of four main modules: image preprocessing, license plate detection, character segmentation, and optical character recognition (OCR) [1], [2]. While OCR is currently considered a relatively solved problem, in the context of ALPR systems, it can be particularly challenging for the OCR module to recognize characters when they are not correctly segmented or when there are variations on the plate image perspective [3], [4], color [5], font, and occlusion [6], illumination [7], background [8], and country standards [9], [10].

In addition, the OCR module of ALPR systems must be highly accurate, provided that it is enough to miss a single character in the plate to miss the vehicle identification. In other words, assuming P_{acc} as the license plate recognition accuracy and C_{acc} as the character recognition accuracy, we have that

$$P_{acc} = C_{acc}^{c}, \tag{1}$$

where c is the number of characters per license plate. So, for example, if we have $c = 7$, we need $C_{acc} \approx 97.7\%$ in order to have $P_{acc} \approx 85\%$, which is considered a baseline performance in commercial ALPR systems. Surprisingly, Tesseract,[1] a well known public OCR system, can only achieve a C_{acc} of approximately 80% when operating on real-world license plate characters.

We address the problem of ALPR by using convolutional neural networks (CNNs) [11] for feature extraction and linear support vector machines (SVMs) [12] for classification[2]. While CNNs have been actively investigated with excellent results in text recognition [13], [14], to the best of our knowledge, this is the first time that *random* CNNs are used for vehicle identification.

Our CNN has four layer respectively implementing (i) filter bank convolution, (ii) rectified-linear activation, (iii) spatial pooling, and (iv) divisive normalization. Its architecture is chosen from thousands of random possibilities and its filter weights are set completely at random. The approach for searching for optimal network architectures is inspired on the work of [15], [16] and is guided by how good the architectures perform in a dataset with 2,548 real-world plate character samples. In addition to the architectural parameters, this search space comprehends hyperpameters defining the behavior of the activation, pooling, and normalization operations (Section II).

According to the methodology described in Section III, we compare the performance of our approach with two baselines: (i) linear SVMs operating directly in the image domain (without an intermediate CNN) and (ii) traditional CNNs also operating in the image domain, but with filter weights learned via back-propagation [11]. As we present in Section IV, our method can outperform the baselines, achieving highest accuracies in the recognition of both digits and letters.

We hypothesize that we could achieve superior performance when using random filters because of the statistical properties enforced by the initialization of their weights and by the subsequent activation function. Given that they are mean-centered, unit-normalized, and uniformly distributed in the filter feature space, these filters compete evenly among themselves and are not prone to overfitting. Moreover, while searching for good

[1]https://code.google.com/p/tesseract-ocr/

[2]The first author, with DECOM-UFOP, has spent his sabbatical year (2013-2014) at the Institute of Computing, UNICAMP, Brazil, and was supported by FAPESP.

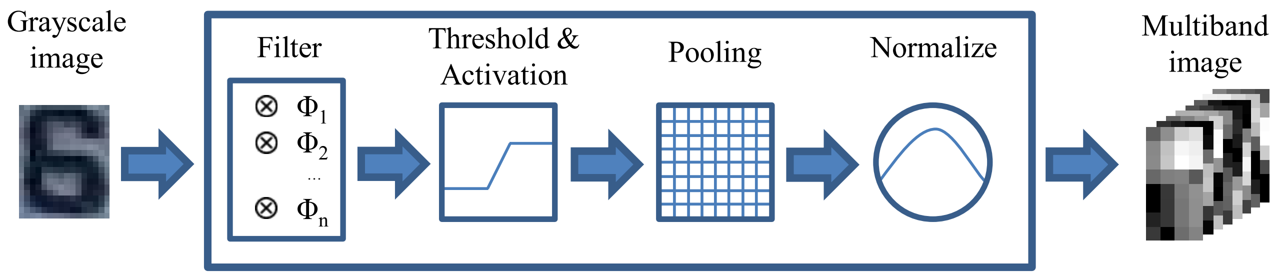

Fig. 1. A convolutional neural network (CNN) transforms a character image into a higher level representation that can be viewed as a multiband image.

architectures, the rectified linear activation enforces sparsity in the information processing flow, thereby strengthening the robustness of the features to the aforementioned variations in plate characters.

The introduction and validation of random CNNs as a promising alternative approach to ALPR systems is the main contribution of this paper.

II. Proposed Convolutional Network

In this work, convolutional neural networks (CNN) can be viewed as a sequence of four linear and non-linear image processing layers. Given an input image, our CNN essentially extracts a higher level representation, named *multiband image*, whose pixel attributes are concatenated into a high-dimensional feature vector for later pattern recognition (Figure 1).[3]

In order to describe the aforementioned operations, let $\hat{I} = (D_I, \vec{I})$ be a multiband image such that $D_I \subset Z^2$ is the image domain and $\vec{I}(p) = \{I_1(p), I_2(p), \dots, I_m(p)\}$ is the attribute vector of a pixel $p = (x_p, y_p) \in D_I$. When $\hat{I}$ is a grayscale image, $m = 1$ and $\hat{I} = (D_I, I)$. In addition, assume $\mathcal{A}(p)$ as a squared region centered at p of size $L_\mathcal{A} \times L_\mathcal{A}$, such data $\mathcal{A} \subset D_I$ and $q \in \mathcal{A}(p)$ iff $\max(|x_q - x_p|, |y_q - y_p|) \leq (L_\mathcal{A} - 1)/2$.

A. Filter Bank Convolution

Let $\Phi = (\mathcal{A}, W)$ be a filter with weights $W(q)$ associated with pixels $q \in \mathcal{A}(p)$. In the case of multiband filters, filter weights can be represented as vectors $\vec{W}_i(q) = \{w_{i,1}(q), w_{i,2}(q), \dots, w_{i,m}(q)\}$ for each filter i of the bank, and a multiband filter bank $\Phi = \{\Phi_1, \Phi_2, \dots, \Phi_n\}$ is a set of filters $\Phi_i = (\mathcal{A}, \vec{W}_i)$, $i = \{1, 2, \dots, n\}$.

The convolution between an input image $\hat{I}$ and a filter Φ_i produces a band i of the filtered image $\hat{J} = (D_J, \vec{J})$, where $D_J \subset D_I$ and $\vec{J} = (J_1, J_2, \dots, J_n)$, such that for each $p \in D_J$,

$$J_i(p) = \sum_{\forall q \in \mathcal{A}(p)} \vec{I}(q) \cdot \vec{W}_i(q). \quad (2)$$

The weights of Φ_i are randomly generated from $U(0,1)$ and are further normalized to zero mean and unit norm in order to ensure that they are spread over the unit sphere.

[3]In this paper, CNNs are described from an image processing perspective, with terms like image *domain*, image *band*, *etc.*, used throughout.

B. Rectified Linear Activation

The activation layer of our network consists of rectified linear units of the type present in many state-of-the-art CNN architectures [15], [17] and simply perform

$$J_i(p) = \max(J_i(p), 0). \quad (3)$$

Notwithstanding its simplicity, this activation function play an important role in the network information flow, specially when coupled with random filters initialized as described in the previous section, in which case it enforces sparsity in the network by discarding 50% of the expected filter responses, thereby improving the overall robustness of the features being extracted.

C. Spatial Pooling

Spatial pooling is a foundational operation in the CNN literature [11] that aims at bringing translational invariance to the features by aggregating activations from the same filter in a given region.

Let $\mathcal{B}(p)$ be pooling regions of size $L_B \times L_B$ centered at pixel p. In addition, let $D_K = D_J/s$ be a regular subsampling of every s pixels $p \in D_J$. We call s the *stride* of the pooling operation. Given that $D_J \subset Z^2$, if $s = 2$, $|D_K| = |D_J|/4$, for example. The pooling operation resulting in the image $\hat{K} = (D_K, \vec{K})$ is defines as

$$K_i(p) = \sqrt[\alpha]{\sum_{\forall q \in \mathcal{B}(p)} J_i(q)^\alpha}, \quad (4)$$

where $p \in D_K$ are pixels in the new image, $i = \{1, 2, \dots, n\}$ are the image bands, and α is a hyperparamter that controls the sensitivity of the operation. In other words, our pooling operation is the L_α-norm of values in $\mathcal{B}(p)$.

D. Divisive Normalization

The last operation of our CNN is divisive normalization, another mechanism widely used in top-performing CNNs [15], [17] that is based on gain control mechanisms found in cortical neurons [18].

This operation is also defined within a squared region $\mathcal{C}(p)$ of size $L_C \times L_C$ centered at pixel p such that

$$O_i(p) = \frac{K_i(p)}{\sqrt{\sum_{j=1}^{n} \sum_{\forall q \in \mathcal{C}(p)} K_j(q) \cdot K_j(q)}} \quad (5)$$

for each pixel $p \in D_O \subset D_K$ of the resulting image $\hat{O} = (D_O, \vec{O})$. Divisive normalization promotes competition among filters (input image bands j) such that high responses will prevail even more over low ones, further strengthening the robustness of the CNN output feature vector $\vec{O}$.

E. CNN Hyperparameters

Most operations described in the previous sections have hyperparameters that need to be determined according to the problem at hand. For example, the size of the filter region — *i.e.*, the *receptive field* of convolutional neurons — is an architectural hyperparameter of the network. In a nutshell, the filtering operation requires the definition of the filter size $L_{\mathcal{A}}$ and the number of filters n; the pooling operation requires the definition of the pooling size $L_{\mathcal{B}}$, the stride s, and the pooling sensitivity α; and the normalization operation requires the definition of $L_{\mathcal{C}}$. Indeed, $L_{\mathcal{A}}$, n, $L_{\mathcal{B}}$, s, α, and $L_{\mathcal{C}}$ can be seen as the architectural parameters of our network.

III. Methodology

A. Dataset

The dataset of plate characters that we use in this work was obtained from Brazilian license plate images captured in a real-world setting, and are available in [19]. Given that we are interesting in evaluating the proposed CNN-based OCR system, the first two steps of the recognition pipeline — namely (i) plate detection and (ii) character segmentation — were respectively implemented according to the methods of Mendes *et al.* [20] and of Nomura *et al.* [21].

A total of 2,548 character images were extracted from the plate images by cropping the segmented regions and shrinking it to an aspect ratio of 5:4 and a size of 20×16 pixels. Given the small image size, in Figure 2 we are able to show the entire dataset, with 1,820 digits and 728 letters. Note that, in some cases, segmentation did not perform well and that we have variations in typesetting, scale, rotation, lightning, *etc.* This dataset will be made public in [22].

B. Evaluation Protocol

In general, license plates are usually composed by letters followed by numbers. Since this is a priori information that can be used in the context of automatic license plate recognition (ALPR), in our experiments we separately evaluate OCR modules for letter and for digit recognition.

In order to evaluate each OCR module, we randomly split the corresponding dataset samples into training and test sets by considering 90% of the samples for training and 10% for testing. We then use the training set to build the OCR module and the test set to measure its accuracy. Our evaluation protocol consist of repeating this random split procedure 30 times and reporting the resulting mean accuracy and standard deviation.

C. CNN Hyperparameter Random Optimization

As presented in Section II, the proposed random CNN that we use for feature extraction requires the definition of several hyperparameters that here we optimize by random search. This strategy has shown to be a very good choice to tackle this problem [15], [23], and in the context of our work can be described by the following steps:

1) Randomly — and uniformly, in our case — sample values from the CNN hyperparameter search space;
2) Extract features from the dataset images with this CNN;
3) Evaluate the CNN according to some performance measure;
4) Repeat steps 1–3 until some criterion is met.

In accordance to the notation of Section II, the hyperparameter search space that we explore in this work (Step 1) is a discrete one, and can be defined as lists of possible values for:

- filter size $L_A = \{3, 5, 7, 9\}$;
- number of filters $n = \{32, 64, 96, 128\}$;
- pooling size $L_B = \{0, 3, 5, 7, 9\}$;
- pooling stride $s = \{1, 2, 3\}$;
- pooling sensitivity $\alpha = \{1, 2, 3, 10\}$;
- normalization size $L_C = \{0, 3, 5, 7, 9\}$.

Once we extract features from all samples (Step 2), the evaluation criterion used to measure the performance of the candidate CNN (Step 3) is the mean accuracy obtained by training a hard-margin ($C = 10^5$) linear support vector machine (SVM) according to the protocol described in Section III-B. Note that we are dealing with high-dimensional features vectors ranging from 512 to 100k elements, and preliminar experiments showed that the use of other kernels such as Radial basis function (RBF) does not bring to us promising results and are too expensive to be evaluated in this context. Finally, our termination criterion (Step 4) is to simply iterate 2,000 times over Steps 1–3 and selecting the best performing CNN at the end.

An important remark is that some combinations of hyperparameter values are invalid due to the small size of the input images. In these cases, we simply ignore this combination and sample the next one. In addition, note that the optimization may result in CNNs with no pooling or normalization layers, since we considered zero as a possible choice for their sizes.

IV. Experiments and Results

We evaluate the performance of the proposed approach in the dataset described in Section III-A according to the protocol presented in Section III-B and following the hyperparameter optimization procedure detailed in Section III-C. This experiment took 3 hours for digits and half an hour for letters running in cluster with eight Intel Core i7 @3770GHz machines.

In Fig. 3, we present histograms summarizing the random search, with mean accuracies in the horizontal axis and number of random CNNs that achieved such accuracy in the vertical axis. It is possible to observe that a large number of CNNs produce accurate results, but just a few result in peak performance. Indeed, this is somehow in accordance to the findings of [15].

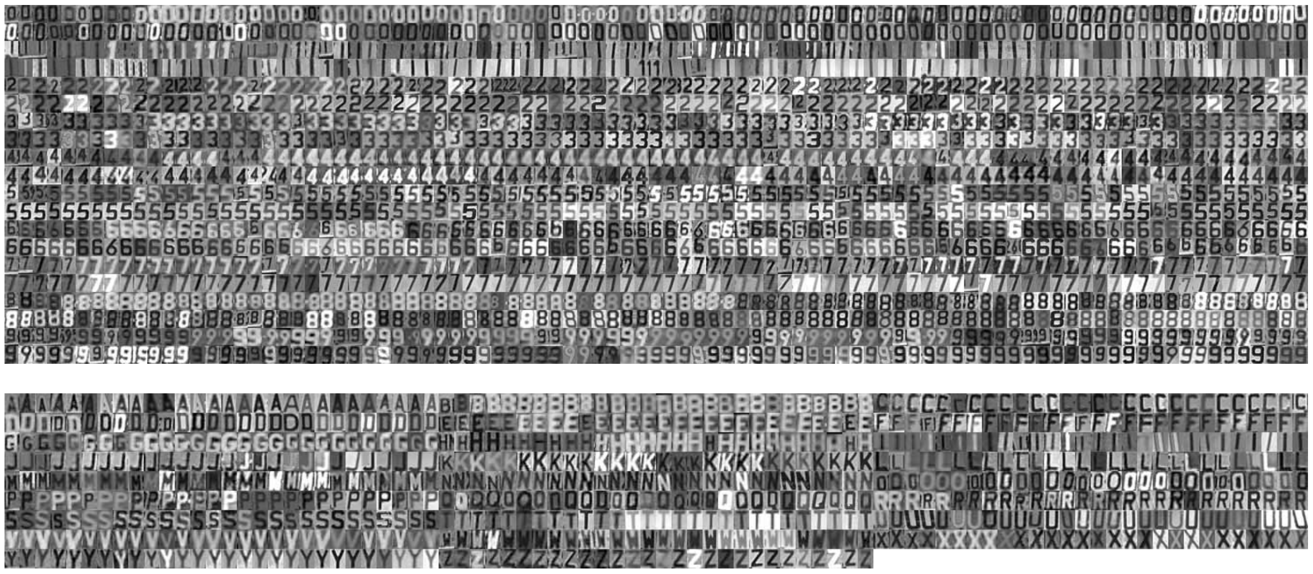

Fig. 2. Dataset used in the experiments with 2,548 characters split into 1,820 digits (0-9, 10 classes) and 728 letters (A-Z, 26 classes). A total of 182 samples per digit and 28 samples per letter are available.

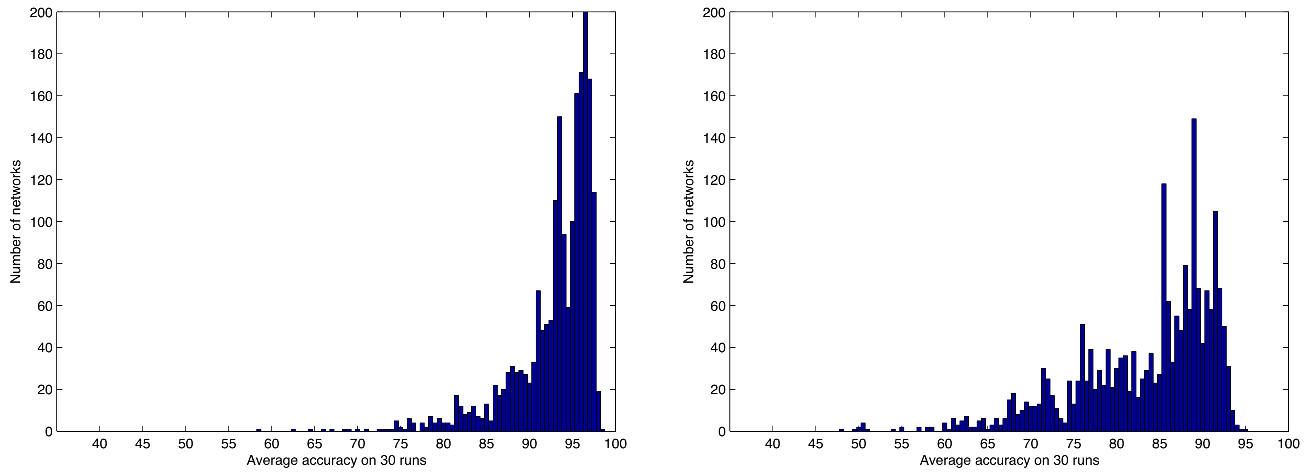

Fig. 3. Histograms summarizing the hyperparameter random optimization of CNNs for digits (left) and letters (right).

In Tables I and II, we present the hyperparameter sets that led to the five best performing random CNNs for digits and letters, respectively. Considering filter sizes, apparently $L_{\mathcal{A}} = 9$ performs best for digits and $L_{\mathcal{A}} = 5$ for letters. Also, we can see a trend towards the highest number of filters allowed ($n = 128$) in both cases. It is not possible to infer anything concrete about pooling regions $L_{\mathcal{B}}$ for digits, but $L_{\mathcal{B}} = 7$ always performed best for letters. The opposite can be said regarding pooling strides, where $s = 1$ was always the best choice for digits and no tendency was observed for letters. Other regularities such as $\alpha = 10$ for letters, $L_{\mathcal{C}} = 3$ for digits, and $L_{\mathcal{C}} = 0$ can also be observed. Taken together, the diversity in performance shown in Fig. 3 combined with the values shown in Tables I and II allow us to conclude that hyperparameter optimization is indeed an important step to building effective random CNNs.

TABLE I
FIVE BEST CNNs FOR DIGITS. SEE SECTION III-C FOR DETAILS.

#	$L_{\mathcal{A}}$	n	$L_{\mathcal{B}}$	s	α	$L_{\mathcal{C}}$	acc. ($\mu \pm \sigma$)
1	9	128	3	1	2	3	98.3±1.05
2	9	128	5	1	1	3	98.2±0.86
3	9	128	3	1	3	3	98.2±1.13
4	9	96	5	1	1	3	98.2±1.01
5	9	128	7	1	10	0	98.1±1.20

We now compare the best systems from Tables I and II with two baseline approaches. The first baseline takes raw

TABLE III
COMPARISON WITH TWO BASELINE APPROACHES.

approach	background norm.	accuracy ($\mu \pm \sigma$)	
		digits	letters
raw pixels	no	69.2±3.54	66.3±4.62
raw pixels	yes	95.2±1.53	88.5±3.01
traditional CNN	no	73.1±2.68	77.8±3.83
traditional CNN	yes	95.1±1.55	86.0±4.21
random CNN	no	98.3±1.05	95.1±1.64
random CNN	yes	98.5±0.72	96.8±1.74

TABLE II
FIVE BEST CNNS FOR LETTERS. SEE SECTION III-C FOR DETAILS.

#	$L_{\mathcal{A}}$	n	$L_{\mathcal{B}}$	s	α	$L_{\mathcal{C}}$	acc. ($\mu \pm \sigma$)
1	5	128	7	2	10	0	95.1±1.88
2	5	128	7	1	10	0	94.7±2.25
3	5	96	7	3	10	0	94.0±2.22
4	5	96	7	3	10	0	94.0±2.22
5	5	64	7	1	3	0	93.9±2.49

pixel values as input to classifiers of the same type that we use in our approach, *i.e.*, hard-margin linear SVMs. Note that our goal here is to compare the features generated by CNN with those of raw pixels, and the optimization of the classifier hyperparameters (in our case C) is out of scope. The second baseline approach, in turn, is a "traditional" CNN whose filter weights are learned via back-propagation and whose architecture was determined as the best performing one out of several possible configurations of single hidden layer CNNs. The best architecture found for this baseline CNN consisted of 5 filtering neurons of size 7×7 and pooling unists of size 2×2. Training such traditional CNN required 100 epochs for digits and 200 for letters using the implementation of [24]. Additionally, we also compare the methods using a preprocessing step for background normalization known to substantially improve performance. This step is done by simple analyzing the intensity of the red channel - so we need to have the input image as a color one. If it is above a threshold, the image is then inverted.

Observe that the best CNN architectures found by our random optimization procedure use much more filters than the traditional CNNs. For such traditional CNNs, the higher the number of filters, the higher both the number of weights to be learn through backpropagation algorithm and also the number of samples required to such learn are. If the number of samples are not enough for such learning process, the generalization capability of the learned network is seriously compromised. In contrast, our proposed approach uses randomly predefined filters as stated earlier.

The results are presented in Table III. While we can observe that both baselines are greatly benefited by background normalization, our approach performs well with and without normalization, suggesting that random CNNs are inherently able to correct for problems in absolute pixel intensities. Considering the results without background normalization, both the method based on raw pixels as well as the traditional CNNs perform badly, while random CNNs perform well. With background normalization, raw pixels and traditional CNNs present better performances, but random CNNs still perform over 3% better for digits and over 8% better for letters. These results clearly support the use of random CNNs as a promising alternative approach to ALPR systems.

V. CONCLUSIONS

In this work, we proposed the use of random convolutional neural networks (CNNs) to extract features for vehicle license plate character recognition. We presented two networks, one for the recognition of digits and the other for letters, whose optimal architectures were chosen by random optimization from thousands of possibilities.

We demonstrated in our experiments that when we train linear SVMs with features extracted with the proposed random CNNs, we can achieve a significantly better performance as compared to training linear SVMs on image pixels or learning the filters weights with back-propagation instead of setting them at random. Indeed, we believe that the statistical properties of random filters coupled with rectified linear activation units were essential in achieving superior results.

While other methods proposed in the literatue can achieve comparable levels of performance [25], [26], the datasets used are different, and therefore it is not clear whether or not they impose the same challenges. For this reason, as future work, we intend to compare our approach with these methods in a challenging dataset such as the one used in this paper. We also plan to evaluate other filter learning techniques, specially unsupervised ones as in [27], [28], [29], but using a few training samples. Finally, we also intend to investigate how different strategies of data augmentation from this dataset might contribute in the conception of even better ALPR systems.

ACKNOWLEDGMENT

Thanks to UFOP, CNPq (Procs. 303673/2010-9 & 479070/2013-0) and FAPESP agency for funding (grants 2013/4172-0 & 2013/11359-0).

References

[1] S. Du, M. Ibrahim, M. S. Shehata, and W. M. Badawy, "Automatic license plate recognition (ALPR): A state-of-the-art review," *IEEE Transactions on Circuits and Systems for Video Technologie*, vol. 23, no. 2, pp. 311–325, 2013.

[2] C.-N. Anagnostopoulos, I. Anagnostopoulos, I. Psoroulas, V. Loumos, and E. Kayafas, "License plate recognition from still images and video sequences: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 3, pp. 377–391, 2008.

[3] H. Hegt, R. de la Haye, and N. Khan, "A high performance license plate recognition system," in *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, vol. 5, 1998, pp. 4357–4362.

[4] S.-L. Chang, L.-S. Chen, Y.-C. Chung, and S.-W. Chen, "Automatic license plate recognition," *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, no. 1, pp. 42–53, 2004.

[5] E. R. Lee, P. K. Kim, and H. J. Kim, "Automatic recognition of a car license plate using color image processing," in *IEEE International Conference Image Processing (ICIP)*, vol. 2, 1994, pp. 301–305.

[6] J. Matas and K. Zimmermann, "Unconstrained licence plate and text localization and recognition," in *IEEE Intelligent Transportation Systems (ITS)*, 2005, pp. 225–230.

[7] C.-N. Anagnostopoulos, I. Anagnostopoulos, V. Loumos, and E. Kayafas, "A license plate-recognition algorithm for intelligent transportation system applications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 3, pp. 377–392, 2006.

[8] T. Nukano, M. Fukumi, and M. Khalid, "Vehicle license plate character recognition by neural networks," in *International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, 2004, pp. 771–775.

[9] J. Jiao, Q. Ye, and Q. Huang, "A configurable method for multistyle license plate recognition," *Pattern Recognition*, vol. 42, no. 3, pp. 358–369, 2009.

[10] N. Thome and L. Robinault, "A cognitive and video-based approach for multinational license plate recognition," *Machine Vision Applications*, vol. 22, no. 2, pp. 389–407, 2011.

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[12] C. Cortes and V. N. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[13] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng, "End-to-end text recognition with convolutional neural networks," in *IEEE International Conference on Pattern Recognition (ICPR)*, 2012, pp. 3304–3308.

[14] A. Coates, B. Carpenter, S. S. C. Case, B. Suresh, W. T, D. J. Wu, and A. Y. Ng., "Text detection and character recognition in scene images with unsupervised feature learning," in *IEEE International Conference on Document Analysis and Recognition (ICDAR)*, 2011, pp. 440–445.

[15] N. Pinto and D. D. Cox, "Beyond simple features: A large-scale feature search approach to unconstrained face recognition," in *IEEE International Conference on Automatic Face and Gesture Recognition (FG)*, 2011, pp. 8–15.

[16] N. Pinto, Z. Stone, T. Zickler, and D. D. Cox, "Scaling-up biologically-inspired computer vision: A case study in unconstrained face recognition on facebook," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2011, pp. 35–42.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, 2012.

[18] W. S. Geisler and D. G. Albrecht, "Cortical neurons: Isolation of contrast gain control," *Vision Research*, vol. 32, no. 8, pp. 1409 – 1410, 1992.

[19] P. R. Mendes, J. M. R. Neves, A. I. Tavares, and D. Menotti, "Vehicle license plate location (VLPL) algorithms," 2011, https://github.com/pedrormjunior/vlpl.

[20] ——, "Towards an automatic vehicle access control system: License plate location," in *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2011, pp. 2916–2921.

[21] S. Nomura, K. Yamanaka, T. Shiose, H. Kawakami, and O. Katai, "Morphological preprocessing method to thresholding degraded word images," *Pattern Recognition Letters*, vol. 30, no. 8, pp. 729–744, 2009.

[22] Omitted, "Vehicle license plate characters," 2013, https://github.com/omitted/vlpc.

[23] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," vol. 13, pp. 281–305, 2012.

[24] R. B. Palm, "Prediction as a candidate for learning deep hierarchical models of data," Master's thesis, Technical University of Denmark, DTU Informatics, Lyngby, Denmark, 2012, https://github.com/rasmusbergpalm/DeepLearnToolbox.

[25] J.-M. Guo and Y.-F. Liu, "License plate localization and character segmentation with feedback self-learning and hybrid binarization techniques," *IEEE Transactions on Vehicle Technology*, vol. 57, no. 3, pp. 1417–1424, 2008.

[26] P. Comelli, P. Ferragina, M. N. Granieri, and F. Stabile, "Optical recognition of motor vehicle license plates," *IEEE Transactions on Vehicle Technology*, vol. 44, no. 4, pp. 790–799, 1995.

[27] A. Coates and A. Y. Ng, "Learning feature representations with K-means," in *Neural Networks: Tricks of the Trade*, 2nd ed. Springer, 2012, pp. 561–580.

[28] A. Coates and A. Ng, "The importance of encoding versus training with sparse coding and vector quantization," in *International Conference on Machine Learning (ICML)*, 2011, pp. 921–928.

[29] A. Coates, A. Y. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," *Journal of Machine Learning Research - Proceedings Track*, vol. 15, pp. 215–223, 2011.