Weakly Supervised Character Detection for License Plate Recognition

Luis Felipe Zeni
Informatics Institute
Federal University of Rio Grande do Sul
Porto Alegre, Brazil
Email: luis.zeni@inf.ufrgs.br

Claudio Jung
Institute of Informatics
Federal University of Rio Grande do Sul
Porto Alegre, Brazil
Email: crjung@inf.ufrgs.br

Abstract—**Automatic Licence Plate Recognition (ALPR) is an essential task in the context of intelligent transportation systems. In a typical ALPR pipeline, the last stage receives as input a cropped license plate region and outputs the string with the plate characters. This paper presents a Weakly Supervised Character Detection (WSCD) approach that requires only string-level annotations (as in generic text recognition methods) but is able to detect characters individually (as in detection-based methods, which require character-level annotations). The proposed method is evaluated in five distinct datasets and present very competitive results against other state-of-the-art methods.**

I. Introduction

Automatic License Plate Recognition (ALPR) has been playing an important role in a wide range of transportation applications such as detection of stolen vehicles, traffic law enforcement and automatic toll control, to name a few. Although ALPR has been an active field of research in the past decades, it still presents challenges, such as different license plate (LP) layouts, variability in image acquisition conditions and dealing with small annotated datasets.

The ALPR pipeline can be divided into two main steps: license plate detection (LPD) and license plate recognition (LPR). The first task aims to extract the LP region from the input image, while the second one is responsible for producing the LP string. While LPD methods have shown good generalization to different layouts, LPR methods are often trained for a specific country/region, such as in [1], [2].

LPR can be split into two main groups: methods that depends on costly bounding box character annotations, such as supervision [2]–[7]; and approaches that use only the plate string as supervision [8]–[10]. Methods require the bounding box as supervision tend to have a better adaptation to different plate layouts because they learn the appearance of each character separately. However, annotating bounding boxes is a costly, tedious, and error-prone task. To alleviate this annotation cost, some methods utilize only the string as supervision to learn a coarse localization of the characters and the final string. The drawback of these methods is that they tend to not generalize well to new plate layouts.

In this paper, we present a Weakly Supervised Character Detection (WSCD) method that explores the best of both worlds: it uses only the string-level annotations as training supervision, which is utilized to learn the bounding boxes of the characters in a weakly supervised fashion. We start with a baseline Multiple Instance Detection Network method [11] with Online Instance Refinement [12] and Knowledge Distillation [13] and present modifications to deal with the LPR problem. Our main contributions are: i) the introduction of an instance aware online refinement approach that can be used when the number of category is know; ii) a sub-network for estimating the number of characters that guides the final recognition result; and, iii) an extensive evaluation of the generalization capability of state-of-art text recognition methods applied to LPR. Next, we present the related work, and then describe the proposed methodology with the experimental results and conclusions.

II. Related Work

In this section, we start by briefly revising generic Optical Character Recognition (OCR), OCR focused on LPR and Weakly Supervised Object Detection.

A. Optical Character Recognition

Optical Character Recognition (OCR) has been studied by the computer vision community for a long time. In the past years, methods based on deep learning have shown to achieve the best results for generic text detection and recognition in natural images [9]. These approaches typically aim to handle a variety of fonts and explore some kind of language model (as n-grams). Although LPs do not present grammar-related rules, we briefly review next some generic text recognition methods that could be potentially applied to LPR and that require string-level annotation as supervision.

Shi et al. [14] presented an end-to-end approach by modeling scene text recognition as a sequence problem. They combined convolutional and recurrent layers and explored Connectionist Temporal Classification (CTC) to train the model. Other recurrent modules such as Bidirectional LSTMs (BLSTMs) have been using in conjunction with CTC [15], and non-recurrent layers such as CNNs have also been explored with CTC for modeling sequential data [16].

Attentional mechanisms have also been used to highlight and align textual information in several approaches, with promising results. Cheng et al. [8] proposed a Focusing Attention Network (FAN) that combines an attention model that highlights the characters and a focus correction module.

Bai and collaborators [17] presented an alternative correction module called edit probability that requires less supervision than [8]. In [9], the authors remedy the alignment problem using a decoupled attention network that explores visual information, avoiding the use of historical decoding information.

Baek and collaborators [10] showed the strong dependency between the used dataset and performance of text recognition methods, and also provided a generic framework that allows the combination of several individual modules typically used in OCR. As we will show later in this work, such dependency also exists in the context of LPR.

B. Licence Plate Recognition

Although LPR might seem a particular case of generic OCR, that is not exactly the case. On the one hand, the font and size variability in ALPR is smaller than a generic OCR problem; on the other hand, there is no grammatical or semantic information that relates the characters, and LPs captured "in-the-wild" might present strong shadows, non-uniform illumination and other special characters that might be mis-detected as a character.

Some LPR approaches follow a pipeline similar to generic OCR methods, exploring some kind of recurrent module. In a nutshell, they are mostly generic OCR approaches that are customized to the LPR problem, and sometimes combined with a License Plate Detection (LPD) module. For example, Li et. al [18] combined traditional features (such as HoG and LBP) with recurrent networks and CTC for character recognition, then used a deep feature extractor with a similar LPR module in [18]. In a subsequent work [19], the same authors presented an end-to-end network that explores CTC in the final task. Youssef and colleagues [20] presented a text recognition approach based on a fully convolutional network without any recurrent connections using a CTC loss. They show good results in applications such as Captcha recognition and ALPR (for a single dataset).

Another direction toward LPR is to perform OCR using object detectors. The core idea of these methods is to detect each character in the LP independently, and then exploring the set of detection results to build the LP string. The works of Silva and Jung [3]–[5] and Laroca et al. [6], [7] presented LPR modules based on modified YOLO-v2 architectures [21]. The main difference among them is the choice of the anchors, number of layers and the dataset used for training their models. Lee and colleagues [22] presented a network that segments the characters and regresses the expected number of characters in the LP, and recognition is performed using YOLOv3 [23]. Björklund et al. [2] presented a detection-based LPR method that is trained solely using synthetic data.

LPR methods based on the specialization of text recognition methods [18], [20], [24] present the advantage of requiring only string-level annotation, but also produce as an output a sequence of characters with no clear spatial localization. On the other hand, LPR methods based on object detectors [2]–[7] require more detailed annotation data (the bounding boxes of each character), but are able to provide a spatial location for each detected character. This localization property is particularly interesting to deal with LPs that presents two rows of characters, such as motorcycles. This work tries to combine the benefits of both classes of methods.

C. Weakly Supervised Object Detection

WSOD is a category of methods that aim to learn the localization information (bounding box) of object classes using only the category annotations (presence or absence of the class in the image). Impressive advancements have happened in WSOD after the adoption of CNNs as an image feature extractor. Cinbis et. al. [25] proposed a multi-fold multiple instance learning procedure that combines Fisher Vector and CNN features as descriptors. Li et al. [26] introduced a two-stage adaptation algorithm that first collects class-specific object proposals with higher precision and uses the most confident object candidates to turn image classifiers into object detectors.

Bilen et al. [11] proposed a two-stream smooth MIL approach, where one stream performs classification and the other detection. Tang et al. [12] proposed an online instance classification refinement (OICR) module that can be plugged in a cascaded way to [11] with increases the detection performance. In [27], the authors improved the refinement process adding proposal clusters to select one or more supervision boxes during the training, whereas Zeni and Jung [13] proposed a refinement knowledge distillation module and an adaptive refinement supervision function to the OICR, called boosted OICR (BOICR).

Some methods explore the usage of Class Activation Maps (CAM) [28] to select the best boxes during the training process. Diba et al. [29] proposed a three-stage cascaded method that uses CAM to mine boxes. Wei et al. [30] employ a similar approach to mine tight object boxes by exploiting segmentation confidence maps to train a model based on [12]. The drawback of these CAM based methods [29], [30] is that the training process of it is overly complex and the accuracy improvements do not pay off.

Wan et al. [31] proposed a min-entropy latent model to measure the randomness of object localization. Wan et al. [32] introduced a continuation optimization method that uses a series of smoothed loss functions to approximate the target (desired) loss to alleviates the non-convexity problem in MIL. In contrast, Tang et al. [33] proposed a region proposal network that explores the responses in mid-layers of a network to create object proposals removing the need of using an external proposal generator such as Selective Search [34].

This work starts with BOICR [13] as the baseline WSOD method. It introduces an instance aware online refinement approach (since we know the number of instances per category), and a sub-network for estimating the number of characters to guide the final recognition result.

III. Weakly Supervised Character Detection

We present an overview of the proposed architecture and its modules in Fig. 1. The input to our method is a "rectified"

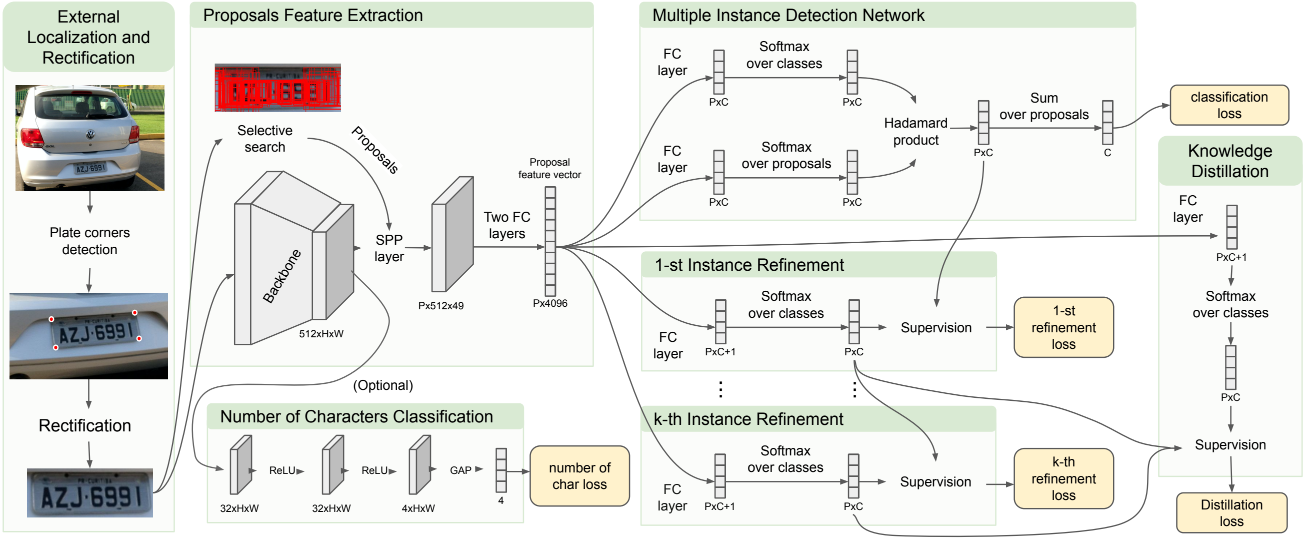


Fig. 1. An overview of the proposed architecture and its modules. The localization and rectification module localizes the corners of the car plate, and these points are used to apply an affine transformation to reduce the perspective deformations of the plate in the image. The rectified image is feed in the proposals feature extraction module, which uses an SSP layer to extract features from proposals generated by selective search. The image features extracted using the CNN backbone are used by the number of characters classification module to estimate the number of characters present in the plate. The multiple instance detection network module learns to select the best proposal instances and generates an image classification score. The instance refinement modules are cascaded into k instances, and each one learns to refine instances from its predecessor result. Finally, the knowledge distillation module combines the output of all the refinement modules to increment the detection accuracy.

LP image that corresponds to (or emulates) a fronto-parallel view of an LP. In scenarios where the LP is already captured in a frontal pose, any approach based on bounding boxes can be used. However, there are approaches that can detect the four corners of the LP (such as [1], [2], [4]), so that a planar homography can be used to warp the detected LP to an approximately fronto-parallel view.

Using the rectified images we train a WSOD pipeline that learns the location of the plate characters using only the plate string as supervision. We build our method on top of [12] and the improvements recently proposed in [13]. We call this process Weakly Supervised Character Detection (WSCD).

The WSCD process starts by extracting candidate proposals from the input image using selective search [34]. We then extract deep image features using a CNN backbone and use these features in two streams. In the first stream, we use Spatial Pyramidal Pooling (SPP) [35] to obtain a fixed-sized subset of features related to each region proposal. The proposals feature maps are then fed to two fully connected (fc) layers and generate proposal feature vectors, which are branched into three different stages in the same way as in [12], [13]. Second, the CNN features feed another sub-network called “Number of Characters Classification” that aims to estimate the number of characters present in the image based on visual content. Although this task could be tackled as a regression problem, the number of characters in an LP does not vary much (typically from 4 to 8), so we adopt a classification-based approach. More precisely, we explore CAMs with Global Average Pooling (GAP) [28], so that images with different input resolutions can be used as input without resizing.

The proposal-based feature vectors are used as input by the remaining modules. The first two modules are similar to [12], where the first one trains a basic instance classifier, and the second stage trains a set of K cascaded refinement modules. We also included the refinement knowledge distillation module proposed by [13] to extract extra knowledge from all the other refinement modules, which has shown to improve the results in generic visual classes provided in the Pascal VOC dataset.

A limitation in the supervision process of most WSOD methods is that they are typically applied in a dataset where the class labels are known, but not the number of instances from each class (e.g., the annotation indicates that a person exists in a given image, but does not tell how many there are). In our adaptation to the LPR problem, we know exactly how many category instances are present in the image because the car plate string is given as supervision. Therefore, we extended the adaptive supervision function proposed in [13] to deal with more than one top-scoring supervision instance if needed. In the rest of this section, we will explain in detail all the employed modules and contributions.

A. Instance selection

Following [12], [13], we employ the multiple instance detection network proposed by [11] because it presents a good trade-off between results and difficulty of implementation. The instance selector scores all the candidate proposals according to the desired classes. The process starts by branching the extracted proposal feature vectors into two streams. Each stream classifies the proposals features using an fc layer that produces two matrices $\mathbf{x}^c$, $\mathbf{x}^d \in \mathbb{R}^{C\times|R|}$, where C is the number of classes and $|R|$ is the number of proposals generated by the region proposal module (in our case, selective search). A *softmax* function is applied to both matrices along different dimensions, yielding

$$\sigma(\mathbf{x}^c)]_{ij} = \frac{e^{x^c_{ij}}}{\sum_{k=1}^{C} e^{x^c_{kj}}}, \quad \sigma(\mathbf{x}^d)]_{ij} = \frac{e^{x^d_{ij}}}{\sum_{k=1}^{|R|} e^{x^c_{ik}}}. \tag{1}$$

The interpretation of these streams is that the first one learns the probability of proposal j belonging to class i, and the second learns the contribution of proposal j to classify the image as category i.

Both streams matrices are combined using element-wise product, that is $\mathbf{x}^R = \sigma(\mathbf{x}^c) \odot \sigma(\mathbf{x}^d) = [x^R_{cj}]$. Lastly, the classification score $\phi_c \in (0,1)$ for each class c is obtained by adding all values over proposal dimension, i.e,. $\phi_c = \sum_{j=1}^{|R|} x^R_{cj}$. Since each LP image tends to present different characters (classes), we use the multi-label cross entropy loss, defined in [12] as

$$L_{class} = -\sum_{c=1}^{C} y_c \log \phi_c + (1 - y_c)\log(1 - \phi_c), \tag{2}$$

where $y_c \in \{0, 1\}$ are the category annotations. More details in [11]–[13].

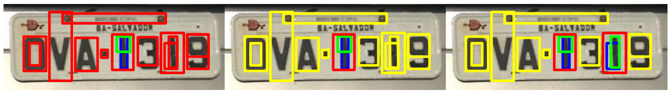
B. Instance aware online instance refinement

Proposed in [12], the online instance refinement strategy aims to iteratively refine the output of the instance classifier to improve the quality of the final detection results. In contrast with the instance classifier, each k^{th} refinement step adds an additional dimension for the background class. As such, the scores for each proposal j are coded as $\mathbf{x}^{R,k}_j \in \mathbb{R}^{(C+1)\times 1}$, $k \in 1, 2, ..., K$, where k is the index for the refinement, K is the total number of cascaded refinements, and the $C + 1^{th}$ dimension relates to the added background. The score vector from the instance classifier is represented here as $\mathbf{x}^{R,0}_j \in \mathbb{R}^{C\times 1}$, and is used as a starting supervision point to the online refinement process.

The vectors $\mathbf{x}^{R,k}_j$ for each refinement step (with $k > 0$) are obtained passing the proposal-related feature vectors through a single fc layer, and a softmax function is applied over all classes.

The supervision of each refinement step k is obtained from the scoring matrix $\mathbf{x}^{R,k-1}$ related to the previous step and a supervision label vector is built for each proposal j in the format $\mathbf{Y}^k_j = [y^k_{1j}, y^k_{2j}, \cdots, y^k_{(C+1),j}]^T \in \mathbb{R}^{(C+1)\times 1}$. As generic category-level annotations provide no information about the number of instances in the image, authors of [12] assume that if a category label is present in the annotation, *at least* one instance of that category must be present in the image. Therefore, the supervision process proposed by [12] starts by selecting the top-scoring proposal for each class. Since we know exactly the number of instances for each category, we changed this process to take into account this information.

Let us consider that s_c is the number of annotated instances for a given class c. When $s_c > 1$, we must select more than one proposal for class c – one for each instance. To avoid selecting more than one proposal for the same instance, we first apply non-maxima suppression (NMS) to all the proposals, obtaining a reduced set of proposals $\mathcal{R}$. We then sort them in descending order with respect to its score $x^{R,k-1}_{cj} \forall r \in \mathcal{R}$, and retrieve the s_c top-scoring proposals $j^{k-1}_{c1}, j^{k-1}_{c2}, \cdots, j^{k-1}_{cs_c}$ for the class c. The set of all top-scoring proposals is denoted by $\mathcal{R}_t$.



(a) OICR [12] (b) Knowledge Dist. [13] (c) Instance Aware

Fig. 2. Effect of different supervision strategies. Green denotes the supervision boxes, blue boxes pass the threshold (selected), red boxes fail (selected as background), and yellow boxes are ignored in loss.

For all the remaining proposals, we must either assign it to an existing class or to the background (class $C + 1$), which is done by geometrical similarity/proximity. For a candidate proposal j, we compute its Intersection over Union (IoU) with all elements in $\mathcal{R}_t$, and retrieve the proposal $j' \in \mathcal{R}_t$ with the largest IoU, which is assigned to class c'. The class c^*_j assigned to proposal j is given by

$$c^*_j = \begin{cases} c', & \text{if } \ \lambda \leq IoU(j, j'), \\ C + 1, & \text{if } \ \lambda_{ign} \leq IoU(j, j') < \lambda \ , \\ -1, & \text{otherwise} \end{cases} \tag{3}$$

where $0 < \lambda_{ign} < \lambda < 1$ are IoU thresholds. The largest threshold λ is used to consider the two proposals being compared as belonging to the same object, and hence presenting the same label. The smallest threshold is used to detect proposals with a partial overlap λ_{ign}, which are considered background to avoid growing the object region too much. If the overlap is smaller than λ_{ign}, we use a fake label $c^*_j = -1$ to indicate that the proposal should be ignored. The motivation for ignoring some proposals is that sometimes the best s_c proposals for a given class do not reflect the real location of the characters in the image (specially in early training stages), and labeling all proposals with $IOU < \lambda$ as background will probably cause the network to classify the real locations as background in future. We present an example of a plate with two instances of '1' and the effects of different supervision strategies in Fig. 2 .

We employed the same adaptive strategy proposed in [13] to select the values for λ and λ_{ign}. It explores a monotonically increasing function such that more candidates are aggregated in the beginning (since the top-proposals are typically small and represent only a part of the object) and less at the end of the training. This concept is also explored by [32] is a different context. More precisely, we defined

$$\lambda = \frac{1}{2}\frac{\log(s + l_b) - \log l_b}{\log(S + l_b) - \log l_b}, \tag{4}$$

where s is the current training step, S is the total of training steeps, and l_b defines the velocity that the curve grows.

Threshold λ_{ign} is given by

$$\lambda_{ign} = \lambda_{max} - \lambda, \tag{5}$$

where λ_{max} defines an upper bound for λ_{ign}.

After this step, all proposals j present an associated class label, and the supervision vector $\mathbf{Y}_j^k$ can be built. Each element y_{cj}^k is updated using c_j^*, that is, $y_{cj}^k = 1$ if $c = c_j^*$ and zero otherwise. Finally, the refinement loss function is defined as

$$L_{agent}^K = -\frac{1}{|R|}\sum_{j=1}^{|R|}\sum_{c=1}^{C+1} w_j^k y_{cj}^k \log x_{cj}^{R,k}, \tag{6}$$

where w_r^k is a weight regularization term introduced to reduce noise during the supervision and is obtained as $w_j^k = x_{cj_c^{k-1}}^{R,k-1}$, which is the classification score of the proposal at the previous refinment $k-1$. The weight w_j^k is set to zero if the class assigned to proposal j is -1 according to Eq. (3), meaning that such proposal is ignored by the loss function.

C. Knowledge distillation module

We also included the knowledge distillation module proposed in [13]. This distillation module learns using a combination of the outputs from the K sequential refinement modules, and is defined simply as

$$\mathbf{x}_{cj}^D = \frac{1}{K}\sum_{k=1}^{K}\mathbf{x}_{cj}^{R,k}, \tag{7}$$

where $\mathbf{x}_{cj}^D$ is the supervision matrix used by the distillation module. The module also outputs a score matrix in the format $\mathbf{x}_j^{Dk} \in \mathbb{R}^{(C+1)\times 1}$. To obtain $\mathbf{x}_j^{Dk}$, the proposals-related feature vector is passed through a single fc layer, and a softmax layer is applied over the class dimension. The remaining process is similar to the described in section III-B and the loss function $L_{destill}$ is the same as the weighted softmax loss in Eq. (6).

D. Classification of the Number of Characters

The number of characters in an LP typically depends on the country/region. For some regions (as the case of Brazil), the number of characters is fixed. However, European and North American LPs present a variable number of characters, which should be detected at inference time.

As in a typical object detector, our WSCD approach employs a detection threshold that intrinsically defines the number of objects (in our case, characters) that will be retrieved. The problem with a fixed threshold is that is hard to choose an ideal value, and probably the selected threshold will not work well for all different regions producing more or fewer characters than the ground-truth string.

In this work, we created a branch in the network that aims to classify the number of characters of the given image. As the string of each training sample is given, we can use the length of the string as the ground-truth class and train a classifier to produce the number of characters in the image. Since our method deals with image inputs of different sizes, we based our character number classification branch on the CAM-based classification presented in [28].

The process starts receiving the image feature map $S \in \mathbb{R}^{\lfloor\frac{H}{2^p}\rfloor\times\lfloor\frac{W}{2^p}\rfloor\times L}$, where $\lfloor\frac{H}{2^p}\rfloor \times \lfloor\frac{W}{2^p}\rfloor$ are the image height and width adjusted to the final size after p max-pooling layers in the backbone and L is the number channels. The feature map S is passed through two convolutional layers of 512 channels with the kernel size of 3×3 and Rectified Linear Units (ReLU) as the activation function. In the sequence, a convolutional layer with D channels and kernel size of 1×1 (no padding), where D is the number of classes in this module. Then, the output is fed into a Global Average Pooling (GAP) layer followed by a softmax activation to generate a D-dimensional score vector $\boldsymbol{\eta}$ for the number of characters in the LP.

As the loss function L_{num_char} we used the same function defined in Eq. (2) based on the cross-entropy, with the obvious difference that the classes are not the same, and the ground-truth labels are given by $y_c = 1$ if $c = \#\text{plate\_string}$.

E. Final loss function

The classification, refinement, distillation, and number of characters modules present individual loss functions. However, we train our model using a single loss that integrates all individual loss functions defined by

$$L = L_{num\_char} + L_{class} + L_{distill} + \sum_{k=1}^{K} L_{agent}^k. \tag{8}$$

.

F. Extracting the LP string from the detections

To extract the final LP string, we first combine the output of all the K refinement and distillation modules, i.e., we compute the scores

$$\mathbf{x}_{cj}^F = \frac{1}{K+1}(\mathbf{x}_{cj}^D + \sum_{k=1}^{K}\mathbf{x}_{cj}^{Rk}), \tag{9}$$

restricting the analysis to $c \in \{1, 2, \cdots, C\}$ (we discard the background class introduced in the refinement step). We then select the indices of the top-scoring class for each proposal

$$b_j = \arg\max_c \mathbf{x}_{cj}^F, \tag{10}$$

and apply NMS to discard highly overlapping results, resulting in a set of pruned detection proposals b_j' sorted in decreasing order w.r.t. the detection score.

Next, we obtain the predicted number of characters N from score vector $\boldsymbol{\eta}$ as

$$N = \arg\max_n \eta_n, \tag{11}$$

and use it to select the N top-scoring proposals b_j' for $j = 1, 2, \cdots, N$. Note that if the number of characters in the LP is known *a priori* (for instance, if we analyze only Brazilian LPs, we know that the LP will present $N = 7$ characters), then the prediction of the number of characters can be ignored. Finally, we sort the retrieved boxes in increasing order with respect to the horizontal coordinate x_{min} of the top-left corner and

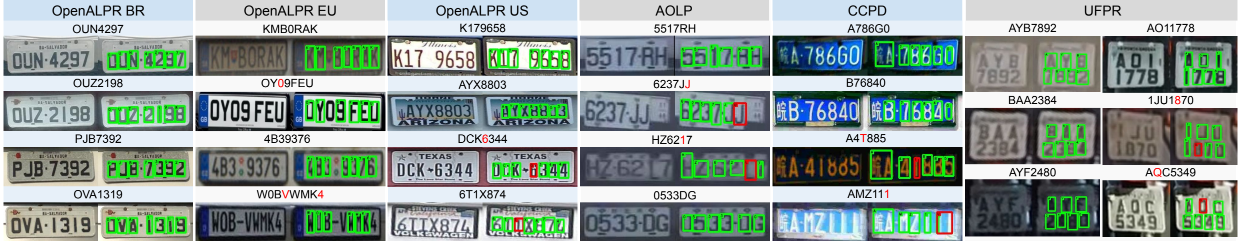


Fig. 3. Some examples of the character detected in different plate layouts. Green rectangles represents proposals that have the correct character detected, and red rectangles are proposals with wrong character.

generate the LP string by concatenating the class predictions of the boxes.

To handle LPs that present two rows of characters (such as motorcycle LPs in Brazil), we first identify the detections related to each row. We first retrieve the central vertical coordinate v_i of each detected box and compute the mean value μ of v_i in a robust manner using an α-trimmed mean that ignores the lowest and highest values (which might be related to outliers). Then, characters presenting v_i smaller than μ are assigned to the first row, and the others to the second row. For each row, detections are scanned from left to right (as described before).

IV. Experimental results

We conducted our experiments using datasets with LPs from different regions, namely: Brazil, China, Taiwan, United States and European Union. We chose to use only a subset of these regions to train our method, and test using all regions. This allows us to investigate how well our method can generalize to unseen LP regions. We start describing each dataset, its characteristics, and what modifications we made. Next, we perform an ablation experiment to compare the effects of our contributions to [12] when applied to the LPR problem. Finally, we compare our model with other state-of-art methods.

A. Datasets

We use five different datasets in our experiments. Three of them were used for both training and testing (AOLP, Fazenda, CCPD), and the other two (OpenALPR and UFPR-ALPR) are used to test the model. Due to the LP fonts, we used the pairs 'I'-'1' and '0'-'O' as a single label each, yielding 34 classes for character detection.

We ran an LPD method (unpublished) to extract the four LP corners (except for CCPD, which already provides such annotation) and rectify them to a fronto-parallel view using a planar homography, keeping a 3×1 aspect ratio for cars and 1.17×1 for motorcycles[1]. **CCPD** [1] contains Chinese plates. We use 100k images to train and 20k images from the validation set to test. **AOLP** [36] contains Taiwanese plates, 1,162 images were used to train and 776 images to test. **Fazenda** is a private dataset of Brazilian LPs, we used 34,584 images to train and 20k images to test. **OpenALPR**[2] contain 112 Brazilian, 173 American and 107 European plates. **UFPR-ALPR** [37] have Brazilian plates. We selected only the motorcycle subset as a test set, which contains 783 images after running the LPD method.

B. Implementation Details

All experiments were performed using PyTorch 1.2, and VGG16 [38] pre-trained on ImageNet as backbone. We used a Nvidia Titan XP GPU to train the model, being able to process 32 rectified images per second in a single batch. The object proposals of the rectified images are extracted using Selective Search [34] and we filter out proposals that present width larger than the height to remove false candidates. For the module that estimates the number of characters in the LP, we used five classes (4 to 8 characters). For data augmentation, the rectified images were re-sized into five scales $\{480, 576, 688, 864, 1200\}$ concerning the smallest dimension.

We also added other random modifications, such as colour changes (in HSV colour space), morphological grayscale dilations or erosions (that thicken or thin the LP characters), blur and noise during the training process. We chose the SGD algorithm with momentum 0.9, weight decay $5e^{-4}$, and batch size 4 to optimize the weights. In the adaptive aggregation function, we set $l_b = 100$ and $\lambda_{max} = 0.51$ given in Eqs. (4) and (5). The learning rate is set to 0.001 for the first 30K iterations and then decreases to 0.0001 in the following 20K iterations. In test time, all the five augmented images are passed in the network, and the outputs are averaged.

C. Ablation Experiments and Classification of the Number of Characters

We conduct some ablation experiments to illustrate the effectiveness of the proposed improvements over the baseline method OICR [12] and Bosted-OICR [13]. We evaluate four different scenarios in all datasets, and for each dataset we evaluate three detection thresholds: "0.5" and "0.6" mean that proposals with a score lower than this threshold are pruned after the NMS step, and "-1" means that the top S proposals are selected after NMS. In the first three experiments, S is the number of characters in the annotation file, and in the last experiment, S is the output of the network that estimates

[1] The train/val/test sets and detections utilized in this paper are available at http://github.com/luiszeni/WSCD

[2] Available at https://github.com/openalpr/benchmarks.

		OpenALPR BR			OpenALPR US			OpenALPR EU			AOLPR			Fazenda			CCPD val 20k			UFPR Mbike		
Error	**Method**	**-1.0**	**0.5**	**0.6**	**-1.0**	**0.5**	**0.6**	**-1.0**	**0.5**	**0.6**	**-1.0**	**0.5**	**0.6**	**-1.0**	**0.5**	**0.6**	**-1.0**	**0.5**	**0.6**	**-1.0**	**0.5**	**0.6**
0	OICR	94.64%	41.07%	35.71%	72.83%	48.55%	45.09%	41.12%	10.28%	9.35%	98.20%	69.72%	65.46%	93.86%	42.27%	38.71%	99.48%	69.02%	67.39%	78.93%	45.08%	38.83%
	OICR+KD	100%	100%	100%	83.82%	**76.88%**	**75.72%**	87.85%	73.83%	77.57%	98.20%	97.55%	97.55%	**94.79%**	**93.41%**	92.76%	**99.85%**	**99.43%**	99.51%	87.10%	**74.58%**	**75.22%**
	OICR+KD+MS	100%	100%	100%	**84.97%**	72.83%	74.57%	**98.13%**	**82.24%**	**83.18%**	**98.71%**	**98.20%**	**98.07%**	94.58%	93.23%	**92.78%**	99.81%	**99.43%**	**99.52%**	86.08%	70.37%	73.31%
	OICR+KD+MS+cont	100%	100%	100%	53.76%	72.25%	71.68%	5.61%	68.22%	75.70%	98.07%	97.81%	97.42%	94.55%	93.12%	92.50%	99.82%	99.13%	99.25%	**86.97%**	**74.58%**	72.29%
1	OICR	100%	84.82%	78.57%	89.60%	82.66%	77.46%	63.55%	48.60%	41.12%	99.48%	97.04%	95.62%	98.41%	82.75%	80.01%	99.76%	95.44%	94.88%	92.46%	85.31%	80.33%
	OICR+KD	100%	100%	100%	**98.27%**	96.53%	97.11%	97.20%	95.33%	96.26%	**100%**	**99.87%**	**99.87%**	**98.93%**	98.82%	98.56%	99.97%	99.96%	**99.96%**	**97.32%**	**96.30%**	**95.53%**
	OICR+KD+MS	100%	100%	100%	95.95%	97.69%	97.69%	**99.07%**	**98.13%**	99.07%	99.87%	99.74%	99.74%	98.66%	**98.90%**	**98.75%**	99.96%	99.96%	**99.96%**	96.04%	95.27%	95.02%
	OICR+KD+MS+cont	100%	100%	100%	93.06%	**99.42%**	**98.27%**	96.26%	97.20%	97.20%	99.87%	99.74%	99.74%	98.87%	98.83%	98.58%	**99.98%**	**99.97%**	99.95%	96.42%	95.79%	94.25%

TABLE I

ABLATION STUDY ACCURACY PERFORMANCE ON ALL THE DATASETS

OpenALPR BR	**OpenALPR US**	**OpenALPR EU**	**AOLP**	**Fazenda**	**CCPD val 20k**	**UFPR Mbike**
100.00%	64.16%	5.61%	99.61%	99.99%	100.00%	99.74%

TABLE II

ACCURACY OF THE PROPOSED NUMBER OF CHARACTERS CLASSIFICATION MODULE.

		OpenALPR					**CCPD**	**UFPR**
Error	**Method**	**BR**	**US**	**EU**	**AOLPR**	**Fazenda**	**v. 20k**	**Mbike**
0	LPR of [3]	98.21%	69.36%	85.98%	94.20%	73.85%	58.45%	0.26%
	LPR of [7]	98.21%	**94.80%**	**99.07%**	**99.10%**	86.57%	68.28%	84.93%
	pretrained STR [39]	4.46%	0.57%	0%	31.82%	0.33%	0.15%	0%
	retrained STR [39]	98.21%	34.68%	4.67%	97.68%	**97.23%**	**99.86%**	0.26%
	OICR+KD+MS+cont	**100%**	53.76%	5.61%	98.07%	94.55%	99.82%	**86.97%**
	OICR+KD+MS th=0.5	**100%**	72.83%	82.24%	98.20%	93.23%	99.43%	70.37%
1	LPR of [3]	100%	94.80%	98.13%	99.61%	93.74%	89.66%	0.26%
	LPR of [7]	100%	**99.42%**	**100%**	**99.87%**	97.66%	94.69%	95.66%
	pretrained STR [39]	9.82%	3.46%	9.34%	54.12%	1.71%	34.58%	0%
	retrained STR [39]	100%	69.36%	28.97%	99.09%	**99.84%**	**99.98%**	12.13%
	OICR+KD+MS+cont	100%	93.06%	96.26%	**99.87%**	98.87%	**99.98%**	**96.42%**
	OICR+KD+MS th=0.5	100%	97.69%	98.13%	99.74%	98.90%	99.96%	95.27%

TABLE III

COMPARISON WITH OTHER STATE OF ART METHODS.

the number of characters. For all methods, we evaluate the accuracy w.r.t. the edit error between the detected and GT strings: Error 0 demands exact string match, and Error 1 allows a one-character error tolerance.

Table I shows that applying OICR [12] to LP images does not yield good results, particularly for datasets that were not present in the training set. Including the knowledge distillation and the adaptive function proposed in [13] (OICR+KD) improves the results in OpenALPR BR (5.36%), OpenALPR US(10.98%), OpenALPR EU (46.73%), Fazenda (0.92%), CCPD (0.36%) and UFPR (8.17%) when considering Error 0 and threshold: -1. Including the number of instances in the supervision as described in Sec. III (OICR+KD+MS) yields a small improvement in unseen datasets OpenALPR US (1.16%) and AOLPR (0.52%), and a higher gain for OpenALPR EU (10.28%). Finally, we evaluate the results using our character count classification module to select the top s proposals, instead of relying on a fixed threshold (or knowing *a priori* the number of characters in the LP). The full accuracy results (Error 0) were very good for datasets present in the training step and also for OpenALPR BR and UFPR (which contains motorcycle LPs), being better than using a fixed threshold in most of these datasets. Results were bad for OpenALPR US and EU when evaluating full accuracy, but considerably better when using the one-character tolerance. In fact, Table II shows the accuracy of the classifier that estimates the number of characters for all datasets. The accuracy is great for datasets present in the training set and also for OpenALPR BR and UFPR, which were not present in the training data but contain Brazilian LPs, as dataset Fazenda. On the other hand, results for ALPR EU and US were bad, which indicates overfitting (we believe that the network is actually learning the LP region/layout instead of effectively counting the number of characters).

D. Comparison with state-of-the-art

We compare our method with three other state-of-the-art methods for both generic text recognition and LPR, and results are shown in Table III. "pretrained STR" is the pre-trained generic text recognition approach presented in [39], which performs really bad in all datasets. We believe that this behavior is related to the LSTM memorizing the sequence of the datasets where it was trained. We also retrained this model with our datasets, which considerably improves the accuracy. However, the generalization abilities in unseen datasets is disappointing, and it is unable to handle motorcycle LPs (characters in two rows). Quoting the authors of [39], "Dataset Really Matters in STR". We also evaluate the fully-supervised methods [4] and [7] in all datasets, which are based on object detectors and trained with character bounding box annotations. In [4], LPs from different regions (Brazilian, Taiwanese, European) were used to train the model with the help of synthetic images. In [7], the authors also explore LPs from different regions (Brazilian, Taiwanese, English, Chinese) and data augmentation to increase variability. Using thresold: -1, our method overperforms all other methods in the UFPR dataset and has very competitive results in the AOLPR, Fazenda and CCPD datasets. As discussed before, the classification of the number of characters is bad for Open ALPR EU and US, which compromises our final results for these datasets. However, if using threshold: 0.5 (OICR+KD+MS th=0.5) our results are competitive as well in these datasets.

V. CONCLUSIONS

We presented a weakly supervised character detection (WSCD) method for License Plate Recognition (LPR). Our approach requires string-level annotation (as [39]) but is able to produce detection-level outputs (as [4], [7]). Our experimental results indicate that the proposed method is able to produce state-of-the-art results for several datasets, and shows good generalization capabilities for WSCD. However, the module that classifies the number of characters seems to be overfitting.

As future work, we intend to explore other approaches for selecting region proposals as alternatives to Selective Search. One possibility is to use Maximally-Stable Extremal Region (MSER), which was already explored in the context of text spotting [40]. We also plan to include the errors done in the rectification phase into the evaluation metrics and use the bounding boxes predicted by our model to train object detectors, so that our WSOD would bridge the gap between string-level and character-level annotations.

Acknowledgment

The authors would like to thank the funding agency CAPES (Finance Code 001), as well as NVIDIA Corporation for the donation of the GPU used in this research.

References

[1] Z. Xu, W. Yang, A. Meng, N. Lu, and H. Huang, "Towards end-to-end license plate detection and recognition: A large dataset and baseline," in *European Conference on Computer Vision*, 2018, pp. 255–271.

[2] T. Björklund, A. Fiandrotti, M. Annarumma, G. Francini, and E. Magli, "Robust license plate recognition using neural networks trained on synthetic images," *Pattern Recognition*, vol. 93, pp. 134–146, 2019.

[3] S. M. Silva and C. R. Jung, "Real-time brazilian license plate detection and recognition using deep convolutional neural networks," in *Conference on Graphics, Patterns and Images*, Oct 2017, pp. 55–62.

[4] ——, "License plate detection and recognition in unconstrained scenarios," in *2018 European Conference on Computer Vision (ECCV)*, Sep 2018, pp. 580–596.

[5] ——, "Real-time license plate detection and recognition using deep convolutional neural networks," *Journal of Visual Communication and Image Representation*, p. 102773, 2020.

[6] R. Laroca, E. Severo, L. A. Zanlorensi, L. S. Oliveira, G. R. Gonçalves, W. R. Schwartz, and D. Menotti, "A robust real-time automatic license plate recognition based on the YOLO detector," *CoRR*, vol. abs/1802.09567, 2018.

[7] R. Laroca, L. A. Zanlorensi, G. R. Gonçalves, E. Todt, W. R. Schwartz, and D. Menotti, "An efficient and layout-independent automatic license plate recognition system based on the YOLO detector," *arXiv preprint*, vol. arXiv:1909.01754, pp. 1–14, 2019.

[8] Z. Cheng, F. Bai, Y. Xu, G. Zheng, S. Pu, and S. Zhou, "Focusing attention: Towards accurate text recognition in natural images," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5076–5084.

[9] T. Wang, Y. Zhu, L. Jin, C. Luo, X. Chen, Y. Wu, Q. Wang, and M. Cai, "Decoupled attention network for text recognition." in *AAAI*, 2020, pp. 12 216–12 224.

[10] J. Baek, G. Kim, J. Lee, S. Park, D. Han, S. Yun, S. J. Oh, and H. Lee, "What is wrong with scene text recognition model comparisons? dataset and model analysis," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 4715–4723.

[11] H. Bilen and A. Vedaldi, "Weakly supervised deep detection networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2846–2854.

[12] P. Tang, X. Wang, X. Bai, and W. Liu, "Multiple instance detection network with online instance classifier refinement," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, pp. 3059–3067, 2017.

[13] L. F. Zeni and C. R. Jung, "Distilling knowledge from refinement in multiple instance detection networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 768–769.

[14] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 11, pp. 2298–2304, 2016.

[15] T. M. Breuel, "High performance text recognition using a hybrid convolutional-lstm implementation," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 1. IEEE, 2017, pp. 11–16.

[16] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[17] F. Bai, Z. Cheng, Y. Niu, S. Pu, and S. Zhou, "Edit probability for scene text recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1508–1516.

[18] H. Li and C. Shen, "Reading Car License Plates Using Deep Convolutional Neural Networks and LSTMs," *arXiv preprint arXiv:1601.05610*, jan 2016.

[19] H. Li, P. Wang, and C. Shen, "Toward end-to-end car license plate detection and recognition with deep neural networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 3, pp. 1126–1136, 2018.

[20] M. Yousef, K. F. Hussain, and U. S. Mohammed, "Accurate, data-efficient, unconstrained text recognition with convolutional neural networks," *Pattern Recognition*, p. 107482, 2020.

[21] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jul 2017, pp. 6517–6525.

[22] Y. Lee, J. Lee, H. Ahn, and M. Jeon, "Snider: Single noisy image denoising and rectification for improving license plate recognition," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[23] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[24] K. Li, Z. Wu, K.-C. Peng, J. Ernst, and Y. Fu, "Tell Me Where to Look: Guided Attention Inference Network," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, jun 2018, pp. 9215–9223.

[25] R. G. Cinbis, J. Verbeek, and C. Schmid, "Weakly supervised object localization with multi-fold multiple instance learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 1, pp. 189–203, 2016.

[26] D. Li, J.-B. Huang, Y. Li, S. Wang, and M.-H. Yang, "Weakly Supervised Object Localization with Progressive Domain Adaptation," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3512–3520, 2016.

[27] P. Tang, X. Wang, S. Bai, W. Shen, X. Bai, W. Liu, and A. L. Yuille, "Pcl: Proposal cluster learning for weakly supervised object detection," *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[28] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*. IEEE, 2016, pp. 2921–2929.

[29] A. Diba, V. Sharma, A. Pazandeh, H. Pirsiavash, and L. Van Gool, "Weakly supervised cascaded convolutional networks," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 5131–5139, 2017.

[30] Y. Wei, Z. Shen, B. Cheng, H. Shi, J. Xiong, J. Feng, and T. Huang, "Ts2c: Tight box mining with surrounding segmentation context for weakly supervised object detection," in *European Conference on Computer Vision (ECCV)*, 2018, pp. 434–450.

[31] F. Wan, P. Wei, Z. Han, J. Jiao, and Q. Ye, "Min-Entropy Latent Model for Weakly Supervised Object Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2019.

[32] F. Wan, C. Liu, W. Ke, X. Ji, J. Jiao, and Q. Ye, "C-MIL: Continuation Multiple Instance Learning for Weakly Supervised Object Detection," *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 2199–2208, 2019.

[33] P. Tang, X. Wang, A. Wang, Y. Yan, W. Liu, J. Huang, and A. Yuille, "Weakly supervised region proposal network and object detection," in *European conference on computer vision*, 2018, pp. 352–368.

[34] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[35] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[36] G.-S. Hsu, J.-C. Chen, and Y.-Z. Chung, "Application-oriented license plate recognition," *IEEE transactions on vehicular technology*, vol. 62, no. 2, pp. 552–561, 2012.

[37] R. Laroca, E. Severo, L. A. Zanlorensi, L. S. Oliveira, G. R. Gonçalves, W. R. Schwartz, and D. Menotti, "A robust real-time automatic license plate recognition based on the YOLO detector," in *International Joint Conference on Neural Networks (IJCNN)*, July 2018, pp. 1–10.

[38] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[39] J. Baek, G. Kim, J. Lee, S. Park, D. Han, S. Yun, S. J. Oh, and H. Lee, "What is wrong with scene text recognition model comparisons? dataset and model analysis," in *International Conference on Computer Vision (ICCV)*, 2019.

[40] W. Huang, Y. Qiao, and X. Tang, "Robust scene text detection with convolution neural network induced mser trees," in *European conference on computer vision*. Springer, 2014, pp. 497–511.