Region-based Discriminative Feature Pooling for Scene Text Recognition

Chen-Yu Lee†, Anurag Bhardwaj, Wei Di, Vignesh Jagadeesh, and Robinson Piramuthu
†University of California, San Diego
eBay Research Labs
ch1260@ucsd.edu, {anbhardwaj,wedi,vjagadeesh,rpiramuthu}@ebay.com

Abstract

We present a new feature representation method for scene text recognition problem, particularly focusing on improving scene character recognition. Many existing methods rely on Histogram of Oriented Gradient (HOG) or part-based models, which do not span the feature space well for characters in natural scene images, especially given large variation in fonts with cluttered backgrounds. In this work, we propose a discriminative feature pooling method that automatically learns the most informative sub-regions of each scene character within a multi-class classification framework, whereas each sub-region seamlessly integrates a set of low-level image features through integral images. The proposed feature representation is compact, computationally efficient, and able to effectively model distinctive spatial structures of each individual character class. Extensive experiments conducted on challenging datasets (Chars74K, ICDAR'03, ICDAR'11, SVT) show that our method significantly outperforms existing methods on scene character classification and scene text recognition tasks.

1. Introduction

With the rapid growth in large image collections, efficient ways of extracting useful information from them have gained immense research attention lately. The computer vision community has addressed these issues in a variety of ways, such as image retrieval, object detection and recognition tasks. In this work we are especially interested in recognizing text presents in natural images that contains rich source of information.

Although a variety of useful applications of scene text recognition exist today, it is still considered a largely unsolved problem due to a number of challenges such as large variation in fonts, lighting conditions, perspective transforms, cluttered background, etc. Existing approaches can be divided into two major groups: region-grouping-based and object-recognition-based approaches. Region-grouping-based methods [3, 10, 15, 29] usually involve image binarization or segmentation, and therefore have difficulty recognizing scene characters with noise and low resolution. On the other hand, object-recognition-based approaches typically employ a two-stage pipeline in form of character recognition and word recognition (Fig.3). Various techniques have been proposed in the past few years, which focus on word recognition module such as pictorial structures [23, 24], integer programming [20], conditional random fields [19, 14, 13], Markov model [26], or real-time commercial OCR systems [15].

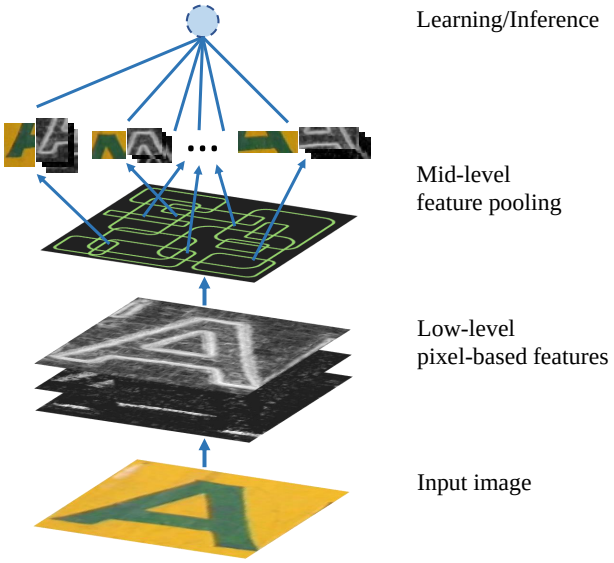

Figure 1: An overview of our region-based feature pooling approach. Low-level image features are automatically aggregated through learned sub-regions that are discriminative for each character class.

Most of these object-recognition-based works simply use off-the-shelf HOG-like features for character recognition at the first stage. A closer look shows that HOG divides input image data into several equally spaced square grids and then extract oriented gradient information in those predefined sub-regions. However, not all sub-regions contain useful information. Certain non-text regions may generate strong histogram values that pollute the feature representa-

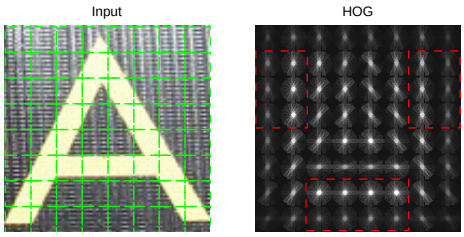

Figure 2: Example of HOG feature computed from a scene text image. Red rectangles highlight the non-text regions that often introduce strong noise.

tion of scene text images after histogram normalization as shown in Figure 2. Due to this drawback, existing character recognition systems are still considered unsatisfied and therefore limit the overall system performance in unconstrained natural settings.

Several other research areas have utilized low-level image features computed using linear and non-linear transformations of the input image [8] and then extract mid-level feature representations using sub-regions based pooling schema. Perona and Malik [12] used Gaussian smoothing kernel to perform spatial integration/pooling from low-level features in early 90s. Methods in [9, 17] aggregated low-level statistics via histogram representation. Viola and Jones proposed integral image technique to compute more expensive bandpass kernels for Haar features. The idea of using integral image to naturally integrate different sources of low-level information has been executed in several systems such as object recognition [21], image categorization [27], and pedestrian detection [30, 7, 6], etc. Such feature mining approaches try to automatically find meaningful feature spaces to improve the system performance.

In this paper, we study the effectiveness of mid-level feature pooling for scene text recognition task based on Integral Channel Features [6], which are a set of pixel-wise low level feature. The proposed feature representation mechanism can automatically learn a compact and discriminative feature space from a set of randomly generated sub-regions, thus leads to better classification performance and light computational load over HOG-like features. We demonstrate state-of-the-art performance through comprehensive experiments on challenging Chars74K [5], ICDAR 2003 [11], ICDAR 2011 [18], and SVT [23] datasets for scene character classification and scene text (word) recognition problems. Figure 3 demonstrates the flow chart of our system pipeline. Section 2 describes the proposed character classification algorithm. Section 3 details feature construction and character re-scoring steps. We use the classic PLEX [23] as our word recognition component.

2. Proposed Framework

Figure 1 illustrates the proposed discriminative mid-level feature pooling algorithm. The following sections describe each step in detail.

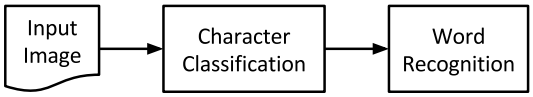

Figure 3: The flow chart of our system.

2.1. Low-level Feature Computation

Given an input image $I(x, y)$, a corresponding feature channel set $C(x, y)$ can be represented as $C = \{\Omega_1(I), \ldots, \Omega_D(I)\}$, where Ω_i denotes a channel generation function over all pixels (x, y) [6]. Each channel function Ω_i takes a pixel location as input and generates a real valued channel response. In this paper, three types of channel features are used, including: gradient histograms (6 orientations), gradient magnitude, and color (LUV), resulting in total $D = 10$ response values per pixel location. Figure 4 shows an example of the generated channel set C for an image of character "A". Notice that these low-level features are pixel-wisely computed and will be further integrated in next step.

2.2. Region-based Feature Pooling

To naturally aggregate low-level feature channels and allow better modeling for various character structures, we randomly generate a large amount of templates (T) in different sizes, locations, and aspect ratios. Let (x_i, y_i) denote the position of the upper left corner and (w_i, h_i) the width and height of the template τ_i. For a given image I with size $N \times N$, the generated template τ_i is a rectangular region $\mathbf{R}$ parameterized by position and shape:

$$\tau_i = \mathbf{R}(x_i, y_i, w_i, h_i), \quad i = 1, \ldots, T \tag{1}$$

where the parameters are randomly sampled from a discrete uniform distribution U:

$$x_i, y_i, w_i, h_i \sim U(1, N)$$

We then generate the first-order channel feature vector $\mathbf{s}_i$, where each entry corresponds to the sum of all pixel values within template τ_i from each channel layer in C:

$$\mathbf{s_i} = [\sum_{x,y \in \tau_i} C_1(x, y), \ldots, \sum_{x,y \in \tau_i} C_D(x, y)] \tag{2}$$

Finally the image I is represented as a concatenation of all $\mathbf{s}_i$ from all rectangular regions in a fixed order:

$$\mathbf{f} = [\mathbf{s}_1\ \mathbf{s}_2\ \mathbf{s}_3\ \ldots\ \mathbf{s}_T]^T \tag{3}$$

Such feature representation can greatly explore spatial characteristics of each character class (text) and naturally integrate heterogeneous sources of information (channels) through the random templates (sub-regions).

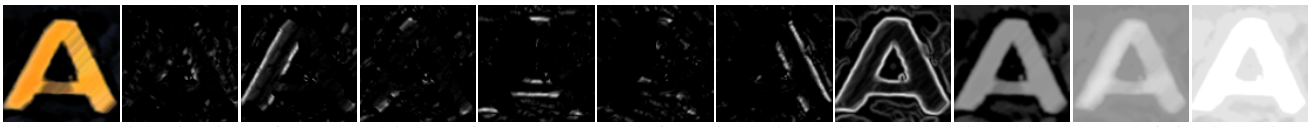

Figure 4: Visualization of low-level image features. From left to right: input image, gradient histograms of 6 orientations, gradient magnitude, and LUV color channels are shown respectively.

2.3. Discriminative Feature Selection

Individual character class varies significantly in terms of shape, contour and geometry. Given the feature vector $\mathbf{f}$ from a large number of randomly generated templates, our goal is to automatically discover important spatial layout of each character class from the training data in a discriminative way. To do so, we propose a method for feature ranking and selection using Support Vector Machines (SVMs) with linear kernels, which has the advantage of naturally sorting feature relative importance through trained weights. We perform feature ranking for all character classes and learn the most informative sub-regions from a classification perspective as described below.

Given data example $\mathbf{f}_i = (f_1, f_2, \ldots, f_d)_i$, where $d = D \times T$ is the dimensionality of the feature space. In general, the decision function ℓ of SVMs can be represented as:

$$\ell(\mathbf{f}) = \text{sgn}(\boldsymbol{\omega}^T \phi(\mathbf{f}) + b) \tag{4}$$

For linear kernel case, where

$$\kappa(\mathbf{f}_j, \mathbf{f}_k) = \phi(\mathbf{f}_j)^T \phi(\mathbf{f}_k) = \mathbf{f}_j^T \mathbf{f}_k \tag{5}$$

the decision function can be rewritten as:

$$\ell(\mathbf{f}) = \text{sgn}(\boldsymbol{\omega}^T \mathbf{f} + b) \tag{6}$$

where $\boldsymbol{\omega} = (\omega_1, \omega_2, \ldots, \omega_d)^T$ are the weights trained by SVMs. Geometrically, $\boldsymbol{\omega}$ can be interpreted as the normal vector of the hyperplane that best separates positive and negative instances. Since the final decision value $\ell(\mathbf{f})$ is a linear combination $\boldsymbol{\omega}^T \mathbf{f}$, mathematically we can see that the bigger the value of $|\omega|$ is, the more the corresponding feature contributes to the final decision value. Therefore, the absolute values of weights $\boldsymbol{\omega}$ indicate the importance of corresponding features towards the final decision. We can therefore automatically identify key/relevant features from the training dataset [2].

In order to select features that are compact but contain the most discriminative information, we first train a linear SVMs for each individual character class using one-vs-all schema. Features of each class are ranked based on the learned SVM weights, and only the top K most important ones are kept. Features from all classes are then merged to obtain the final feature set. Given a set of d dimensional features $\mathbf{f} = (f_1, f_2, \ldots, f_d)$ from the training data, where each dimension corresponds to the first-order feature computed from one of the initially randomly generated T templates with one of the D channel layers (see eq.2), we first rank them according to the absolute learned weights for a given the class m:

$$\begin{aligned} &f^m_{rank} = \{\hat{f}^m_1, \hat{f}^m_2, \ldots, \hat{f}^m_d\} \\ &\text{s.t. } |\omega^m_i(\hat{f}^m_i)| \geq |\omega^m_j(\hat{f}^m_j)|, \text{ and } i < j \end{aligned} \tag{7}$$

Then, we only select the top K features with the highest absolute weights as representative features for the class m:

$$\mathbf{f}^m = \{\hat{f}^m_t, \text{ where } t \leq K\} \tag{8}$$

The final feature vector $\mathbf{F}$ is the union set of all the top K features across all M classes:

$$\mathbf{F} = \bigcup_{m=1:M} \mathbf{f}^m \tag{9}$$

This discriminative feature selection procedure allows the model to focus only on important features and discard unnecessary features. We then retrain the linear SVMs in the new feature space $\mathbf{F}$ to obtain the final model. This procedure will help to improve classification performance while reducing the computation time during testing [2]. Also note that since each template is convolved with multiple channels, among the top K chosen features, the same template may be selected more than once. Therefore, essentially, our proposed approach searches for the best discriminative features that jointly optimize from both spatial and low-level feature perspectives. We will detail the different types of regularization schemes for the classifier in section 4.1.

3. Implementation Details

Fast Feature Construction: To enable fast computing of feature pooling step, integral images for each channel in C are computed first:

$$CC_j(x, y) = \sum_{x' \leqslant x, y' \leqslant y} C_j(x', y'),\ j = 1, \ldots, D \tag{10}$$

This equation can be easily implemented using one line of code in Matlab: `CC = cumsum(cumsum(C),2);`

Using this formulation, the jth entry in $\mathbf{s}_i$ (see eq. 2) can be computed efficiently using integral image technique [22] with three linear operations:

$$\begin{aligned} \mathbf{s}_i(j) &= \sum_{x,y \in \tau_i} C_j(x, y) \\ &= CC_j(x_i, y_i) + CC_j(x_i + w_i, y_i + h_i) \\ &\quad - CC_j(x_i + w_i, y_i) - CC_j(x_i, y_i + h_i) \end{aligned}$$

At testing stage we only need to compute the $\mathbf{s}_i$ that are selected by the final model (re-trained model), which usually contains less than the half of the original feature dimensions as discussed in section 4.1.

Parameter Selection: The procedure of the proposed algorithm is described in Algorithm 1. In the experiment, the value of K is chosen by 5-fold cross validation.

Character Rescoring: English alphabets tend to have certain aspect ratios for each character. For example, character "I" is usually narrower than character "W". We incorporate this shape prior as a post-processing step to re-score the output from the trained SVMs model. First, we learn 62 different Gaussian distributions to model aspect ratios of each character class. These models are then used to re-score the confidence scores for candidate character from a large number of sliding windows operating at multiple scales and aspect ratios:

$$Score(l_i)' = \mathcal{N}(a_i; \mu_m, \sigma_m) \cdot Score(l_i)$$

where μ_m and σ_m are the mean and variance of the aspect ratio (computed from training data) for character class m for a window l_i with aspect ratio a_i, and $Score(l_i)$ is the original confidence score.

Computation Time: We analyze the computation time for the low-level features for a VGA resolution (640×480) input image on a standard PC using fps (frames per second) unit: LUV color channels at 193 fps, gradient magnitude at 118 fps, and gradient histograms (6 orientations) at 97 fps. Computing all 10 channels requires 91 fps computation time. Besides, the proposed region-based feature representation has the advantage of lighter computational cost as compared to sliding window based feature extraction that uses HOG feature. This is because we only need to compute the low-level features for the whole input image once and then perform feature pooling from the informative regions in linear time. However, multiple computation for each sliding window is needed for HOG feature. In our experiment, HOG feature requires average 0.24 seconds feature extraction time for all sliding windows on a VGA input image, compared with the proposed method only needs average 0.08 seconds (both in MATLAB Executable files).

4. Experiments and Evaluation

In this section, we conduct extensive experiments on four public datasets: Chars74K [5], ICDAR03 [11], ICDAR11 [18] Robust Reading Competition datasets, and Street View Text (SVT) [23] dataset. We present detailed results and evaluations on scene character classification and scene text recognition tasks using standard training and testing data splits.

Algorithm 1 Feature Ranking and Selection

Input: Training sets, $(\mathbf{f}_i, labels)$

Output: Sorted feature ranking list and re-trained model

1: **Initial Train:** Train L2-regularized L2-loss linear SVMs with parameters selected by grid search.
2: **Feature Ranking:** Sort features by the absolute values of weights in the model for each class.
3: **Feature Selection:** Choose top K features from the sorted list of each class.
4: **Model Retrain:** Re-train the model with the union set of the top K features from all classes.

4.1. Scene Character Classification

Two standard datasets are used for scene character classification task: Chars74K-15 and ICDAR03-CH dataset. We treat the problem as a multi-class classification problem with 62 different character classes, including 10 number digits and 52 English characters (upper and lower cases). For the purpose of fair comparison, the split of training and testing images is consistent with previous work in [5, 11]. There are in total 930 training and testing examples in Chars74K-15 respectively (15 training and testing examples per character class). ICDAR03-CH dataset is split into 6113 for training and 5369 for testing. We resize each image into a canonical size (24×24) and then extract proposed feature representation. Notice that we only need to initialize the random templates once. The final configuration of the templates will be learned automatically during the training stage, where only templates that carries the most discriminate information will be kept.

Figure 5 shows character classification experiments using L1/L2 regularized L2-loss SVM/logistic regression models for both datasets. SVM based approaches gain extra accuracy improvement after re-train step while regression based models do not have this trend. In addition, L2-regularized SVMs outperform L1-regularized SVMs both before or after model re-train. The results also show that proposed feature selection method outperforms L1-regularized models which is often used as a principled way for feature selection. Besides the better accuracy performance, re-trained model also has significantly smaller feature dimension. This reduces the feature extraction time at the testing stage. Thus, we choose L2-regularized SVMs with re-training step as our character recognition model.

Since boosting approaches have been proven effective on several applications, such as face and pedestrian detection. We therefore compare AdaBoost with our model. As shown in Figure 6, for the 62-way character classification case, our model significantly outperforms AdaBoost by 17%. We also experiment the 2-way classification (text or non-text)

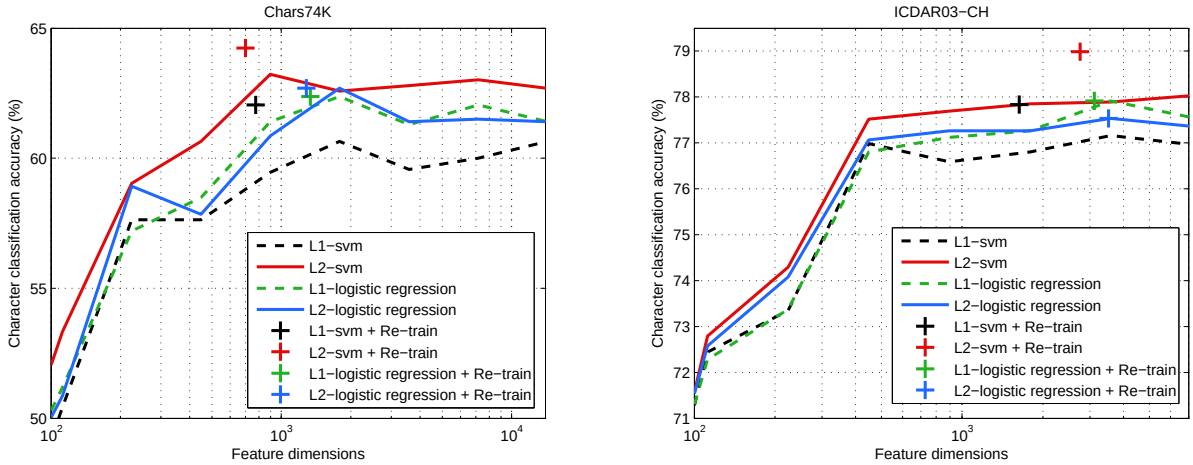

Figure 5: Character classification accuracy on Chars74k and ICDAR03-CH dataset using L1/L2-regularized SVMs/logistic regression. We also perform proposed discriminative feature selection and re-training procedure for the four different methods as shown in crosses. Proposed L2-regularized SVMs + re-train achieves the highest performance on both datasets. Notice that feature selection + re-train procedure not only improves classification accuracy but largely reduces feature dimensions.

task[1] to simulate detection based problems while there is no significant performance difference in this situation.

Table 1 lists the character classification results. Proposed feature representation outperforms all existing methods with classification accuracy up to 0.64 for Chars74K-15 and 0.79 for ICDAR03-CH. Especially, it shows much better accuracy as compared to human-designed features like shape context (SC-SVM, SC-NN), as well as histogram of orientated gradients (GHOG+SVM, LHOG+SVM, and HOG+NN). The possible explanation for this is that equally spaced square grids (e.g. in HOG) might not always carry useful information. Inclusion of non-informative regions does not benefit the recognition task, but instead introduces unnecessary or even harmful noise after histogram normalization.

We also compare our approach with the most recent work in [19] in Table 2, which only considers 49 character classes. Again our mid-level feature pooling algorithm outperforms their tree-structured part-based character model that requires human to pre-define "parts" of each character class. Notice that class merging in [19] leads to unclear performance increase for character recognition stage and the overall word recognition system.

Figure 7 shows the most informative sub-regions learned from proposed algorithm using ICDAR03-CH training examples. We can see that our mid-level feature pooling scheme can discover and catch distinctive regions of each character class: large templates extract global information, while small templates encode local details. These distinctive sub-regions do not necessarily mimic the shape of the character, but are rather located at areas that are capable of separating different character classes.

[1]We randomly sample 5k image patches from background as non-text examples.

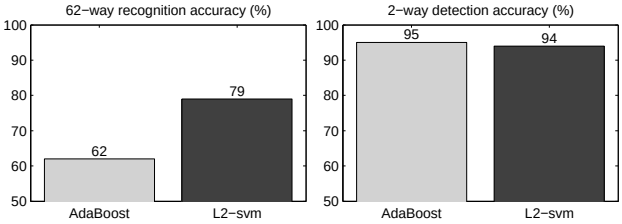

Figure 6: Comparison between AdaBoost and SVMs on ICDAR03 dataset in two scenarios.

Method	Chars74K-15	ICDAR03-CH
Proposed	**0.64**	**0.79**
GHOG+SVM [28]	0.62	0.76
LHOG+SVM [28]	0.58	0.75
HOG+NN [23]	0.58	0.52
MKL [5]	0.55	-
NATIVE+FERNS [23]	0.54	0.64
GB+SVM [5]	0.53	-
GB+NN [5]	0.47	-
SYNTH+FERNS [23]	0.47	0.52
SC+SVM [5]	0.35	-
SC+NN [5]	0.34	-
ABBYY [5]	0.31	0.21

Table 1: Scene character classification accuracy on Chars74k and ICDAR benchmarks.

4.2. Scene Text Recognition

To demonstrate the effectiveness of the proposed feature pooling algorithm, we further extend our experiments for scene text recognition task. We use trained character classifiers as described in section 4.1 and the word spotting approach PLEX from [23]. Notice that we only use ICDAR03-CH training set to learn our character detector,

Method	Chars74K-15	ICDAR03-CH
Proposed	**0.74**	**0.81**
TSM [19]	0.72	0.78
HOG+KNN [19]	0.64	0.66

Table 2: Scene character classification accuracy on Chars74k and ICDAR benchmarks with 49 character classes, in order to make fair comparison of the recent work in [19], which uses part-based tree-structured character classifier that requires human designs of the parts of each character.

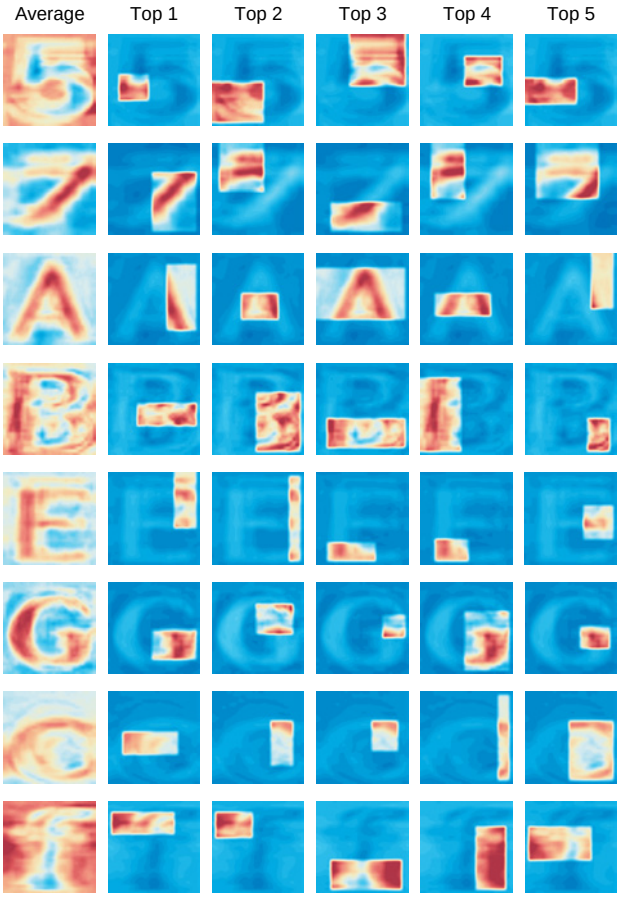

Figure 7: Visualization of the learned templates on ICDAR03-CH dataset. The first column shows the average images of each character class (from top to bottom: 5, 7, A, B, E, G, c, f). Each row shows the top K most informative sub-regions for that character class. Red and blue colors denote different image intensities from high to low. Notice that the learned top K sub-regions capture the most discriminative parts for each class.

which contains only 6113 training images for 62 character classes. Three standard datasets are used in this experiment: ICDAR03-WD, ICDAR11-WD and SVT-WD datasets. For a fair comparison, we use the same training and testing splits as provided in [23] as well as the lexicon. "FULL" refers to experiment that uses a lexicon created from all the words in the test set, while "50" refers to a setup using 50 words for lexicon.

Method	ICDAR03 (FULL)	ICDAR03 (50)	SVT
Proposed+PLEX	**0.76**	**0.88**	**0.80**
TSM+PLEX [19]† (49 classes)	0.70	0.81	0.70
SYNTH+PLEX [23]	0.62	0.76	0.57
ICDAR+PLEX [23]	0.57	0.72	0.56

Table 3: Cropped word recognition accuracy. Different methods use the same word recognition procedure (PLEX) but with different character recognition methods. †Notice that TSM method in [19] only considers 49 character classes instead of 62.

Table 3 lists the cropped word recognition accuracy and directly compares all character detectors that use PLEX word spotting procedure but with different features. Our approach outperforms all the other methods on both ICDAR03 and SVT datasets. Especially, even for the more challenging dataset SVT, our approach significantly improves the baseline accuracy of SYNTH+PLEX by 23%, and is about 10% better than TSM [19], which is built upon a sophisticated tree-structured part-based model with only 49 character classes.

Table 4 includes comparisons to the most recent scene text recognition methods that are under the relative same size of training and testing data splits. These methods includes conditional random fields [19, 14], Markov model [26], and a commercial OCR system [23]. Results clearly show that simply using classic PLEX procedure, proposed character recognition method can still achieve the best word recognition accuracy on all datasets. Notice that result of TSM+CRF [19] is based on 49 character classes instead of 62, as the authors merged few confusion classes, which largely reduce the complexity of the lexicon. This is why their performance is close to ours on "FULL" setup. Examples of word recognition results can be seen in Figures 8a and 8b. It shows that the proposed method is able to recognize words with low contrast, or significant transform in font or size, or clutter background. Our experiment results demonstrate the importance of fundamental feature representation for character recognition as part of the system pipeline.

Some recent works exploit convolutional or deep neural networks for scene text recognition and demonstrate promising results. These methods explore the potentials of using larger scale of training data to fully train each neuron in the networks. For example, methods such as [4, 25] use 12k training images, and method in [1] requires manually labeled 2.2 million training images that are not publicly available. Thus, given our experiment only using standard 6k training images from ICDAR03-CH dataset to train character recognition system, it is hard to decouple the per-

Method	ICDAR03(FULL)	ICDAR03(50)	ICDAR11(FULL)	ICDAR11(50)	SVT
Proposed+PLEX	0.76	**0.88**	0.77	**0.88**	**0.80**
Weinman et al. [26]	-	-	-	-	0.78
TSM+CRF [19]† (49 classes)	**0.79**	0.87	**0.83**	0.87	0.74
Mishra et al. [13]	0.68	0.82	-	-	0.73
Mishra et al. [14]	-	0.82	-	-	0.73
Novikova et al. [16]	-	0.83	-	-	0.73
TSM+PLEX [19]† (49 classes)	0.70	0.81	0.74	0.80	0.70
SYNTH+PLEX [23]	0.62	0.76	-	-	0.57
ABBYY [23]	0.55	0.56	-	-	0.35

Table 4: Scene text recognition accuracy comparing with existing benchmark methods. Our method achieves state-of-the-art performances, which validates the effectiveness of the proposed feature representation. †Notice that TSM method in [19] only considers 49 character classes instead of 62, which largely simplifies the complexity of the lexicon.

formance gain due to the larger training data size. Nevertheless, we believe that some of the observations in this paper can be used in the context of neural network based approaches.

5. Conclusion

In this paper, we proposed a mid-level feature pooling algorithm which incorporates both pixel-wise low-level image features and an automatic discriminative sub-region mining schema. The proposed feature is compact, computationally efficient, and able to effectively model distinctive spatial structures of each individual character. Through extensive experiments on four public datasets, we demonstrate state-of-the-art performance on character classification and word recognition tasks. Future work includes exploring inter-relationship between randomized templates for better interpretation of shape information.

Acknowledgements

The first author thanks Tsung-Yi Lin for valuable discussion and feedback, and also thanks eBay research labs for supporting the project.

References

[1] A. Bissacco, M. Cummins, Y. Netzer, and H. Neven. Photoocr: Reading text in uncontrolled conditions. In *ICCV*, 2013. 6

[2] Y.-W. Chang and C.-J. Lin. Feature ranking using linear svm. *Journal of Machine Learning Research - Proceedings Track*, 2008. 3

[3] X. Chen and A. L. Yuille. Detecting and reading text in natural scenes. In *CVPR*, 2004. 1

[4] A. Coates, B. Carpenter, C. Case, S. Satheesh, B. Suresh, T. Wang, D. Wu, and A. Ng. Text detection and character recognition in scene images with unsupervised feature learning. In *ICDAR*, 2011. 6

[5] T. E. de Campos, B. R. Babu, and M. Varma. Character recognition in natural images. In *International Conference on Computer Vision Theory and Applications, Lisbon, Portugal*, 2009. 2, 4, 5

[6] P. Dollár, Z. Tu, P. Perona, and S. Belongie. Integral channel features. In *BMVC*, 2009. 2

[7] P. Dollár, Z. Tu, H. Tao, and S. Belongie. Feature mining for image classification. In *CVPR*, 2007. 2

[8] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 1980. 2

[9] D. J. Heeger and J. R. Bergen. Pyramid-based texture analysis/synthesis. In *SIGGRAPH*, 1995. 2

[10] K. Kita and T. Wakahara. Binarization of color characters in scene images using k-means clustering and support vector machines. In *ICPR*, 2010. 1

[11] S. M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young. Icdar 2003 robust reading competitions. In *ICDAR*, 2003. 2, 4

[12] J. Malik and P. Perona. Preattentive texture discrimination with early vision mechanisms. *Journal of the Optical Society of America A, 7:923932*, 1990. 2

[13] A. Mishra, K. Alahari, and C. V. Jawahar. Scene text recognition using higher order language priors. In *BMVC*, 2012. 1, 7

[14] A. Mishra, K. Alahari, and C. V. Jawahar. Top-down and bottom-up cues for scene text recognition. In *CVPR*, 2012. 1, 6, 7

[15] L. Neumann and J. Matas. Real-time scene text localization and recognition. In *CVPR*, 2012. 1

[16] T. Novikova, O. Barinova, P. Kohli, and V. Lempitsky. Large-lexicon attribute-consistent text recognition in natural images. In *ECCV*, 2012. 7

[17] J. Puzicha, T. Hofmann, and J. Buhmann. Non-parametric similarity measures for unsupervised texture segmentation and image retrieval. In *CVPR*, 1997. 2

[18] A. Shahab, F. Shafait, and A. Dengel. Icdar 2011 robust reading competition challenge 2: Reading text in scene images. In *ICDAR*, 2011. 2, 4

[19] C. Shi, C. Wang, B. Xiao, Y. Zhang, S. Gao, and Z. Zhang. Scene text recognition using part-based tree-structured character detection. In *CVPR*, 2013. 1, 5, 6, 7

(a) ICDAR

(b) SVT

Figure 8: Examples of word recognition results on (a) ICDAR and (b) SVT datasets. It shows that the proposed method is able to recognize words with low contrast (e.g. *taqueria, allen, muzeo, saks, watercourse*), or significant transform in fonts or size (e.g *shogun, icebox, polish, watercourse*), or clutter background (e.g. *michoacana, icebox*).

[20] D. L. Smith, J. Field, and E. Learned-Miller. Enforcing similarity constraints with integer programming for better scene text recognition. In *CVPR*, 2011. 1

[21] Z. Tu. Probabilistic boosting-tree: learning discriminative models for classification, recognition, and clustering. In *CVPR*, 2005. 2

[22] P. Viola and M. Jones. Robust real-time object detection. In *International Journal of Computer Vision*, 2001. 3

[23] K. Wang, B. Babenko, and S. Belongie. End-to-end scene text recognition. In *ICCV*, 2011. 1, 2, 4, 5, 6, 7

[24] K. Wang and S. Belongie. Word spotting in the wild. In *ECCV*, 2010. 1

[25] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng. End-to-end text recognition with convolutional neural networks. In *ICPR*, 2012. 6

[26] J. J. Weinman, Z. Butler, D. Knoll, and J. Feild. Toward integrated scene text reading. *PAMI*, 2013. 1, 6, 7

[27] B. Yao, A. Khosla, and L. Fei-Fei. Combining randomization and discrimination for fine-grained image categorization. In *CVPR*, 2011. 2

[28] C. Yi, X. Yang, and Y. Tian. Feature representations for scene text character recognition: A comparative study. In *ICDAR*, 2013. 5

[29] C. Zeng, W. Jia, and X. He. An algorithm for colour-based natural scene text segmentation. In *Camera-Based Document Analysis and Recognition*, Lecture Notes in Computer Science. 2012. 1

[30] Q. Zhu, M.-C. Yeh, K.-T. Cheng, and S. Avidan. Fast human detection using a cascade of histograms of oriented gradients. In *CVPR*, 2006. 2