8-2017

Better Text Detection through Improved K-means-based Feature Learning

Kardo Othman Aziz
koa5197@rit.edu

Better Text Detection through Improved K-means-based Feature Learning

by

Kardo Othman Aziz, B.E.

THESIS

Presented to the Department of Computer Science at the

Faculty of the Golisano College of Computer and Information Sciences

Rochester Institute of Technology

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Computer Science

Rochester Institute of Technology

August 2017

Better Text Detection through Improved K-means-based Feature Learning

APPROVED BY

SUPERVISING COMMITTEE:

Dr. Richard Zanibbi, Supervisor

Dr. Leo Reznik, Reader

Dr. Matthew Fluet, Observer

Acknowledgments

I would like to give a special thanks to my advisor Dr. Richard Zanibbi for helping and supporting me throughout my academic path. I also thank him for advising and encouraging me toward academic achievements.

Also I would like to thank my thesis committee Dr. Leo Reznik and Dr. Matthew Fluet for both serving on my thesis committee.

Also I would like thank all members of Document and Pattern Recognition lab at Rochester Institute of Technology. Firstly, the former DPRL lab Phd student, Siyu Zhu for making his system available for me and other students.Then I would like to give a special thank to current Phd student, Kenny Davila who helped me improve my understanding about the problems and finding right path to solve them. A very special I thank my beloved wife and my parents who motivated me and supported me toward my academic goals. ...

Abstract

Better Text Detection through Improved K-means-based Feature Learning

Kardo Othman Aziz, M.S.
Rochester Institute of Technology, 2017

Supervisor: Dr. Richard Zanibbi

In this thesis, we propose a different technique to initialize a Convolutional K-means. We propose Visual Similarity Sampling (VSS) to collect 8×8 sample patches from images for convolutional feature learning. The algorithm uses within-class and between-class cosine similarity/dissimilarity measure to collect samples from both foreground and background. Thus. VSS uses local frequency of shapes within a character patch and uses it as probability distribution to select them. Also, we show how that initializing Convolutional K-means from samples with high between-class and within-class similarity produce discriminative codebook. We test the codebook to detect text in the natural scene. We show that using representative property within and between class for each sample as the probability for selecting it as initial cluster center, helps achieve discriminative cluster centers, which we use as feature maps. One of the advantages of our work is; since it is not problem dependent, it can be applied for sample collection in other pattern recognition problems. The proposed algorithm

helped improve detection rate and simplify the learning process in both convolutional feature learning and text detection training.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Text Detection and localization is an open problem that is challenging due to variations in text size, font color, scene complexity, uneven lighting, blurring, aspect ratio and distortion [34]. Text detection is used in driver-less cars to locate signs and recognize them. It is a prerequisite for applications and systems that work on the content of images; for example, searching images by their textual content, localizing and retrieving math formula on a white-board[10]

Researchers try to work around these issues using different techniques. For the last decade, the primary focus of researchers has been on learning dictionaries or convolutional masks for object and text detection[7, 39], recognition[23, 39] and image restoration[13]. Researchers have tried to learn convolutional masks that can be used to separate text from background[7, 35, 39].

Algorithms for learning dictionaries often make use of unsupervised learning algorithms such as clustering or Convolutional Neural Networks (CNN). Unsupervised techniques are required when a labelled dataset is not available. CNN tries to generate a set of features (i.e. it is a generative method). It usually models how the data is generated in order to categorize them. On the contrary, clustering learns a set of representative features regardless

of how the data generated. Moreover, clustering is a straightforward and easy technique, that can achieve state-of-the-art performance.

Convolutional Neural Networks include convolutional mask learning. Due to nature of CNN, its architecture and training process is more complex than clustering algorithm. CNN has multiple convolutional and pooling layer network that connected to a fully connected neural network that learns the masks through back propagation. While in our work, we don't have any convolutional operations in the dictionary learning process. Since our dictionary has 1000 patches, we use 1000 convolutional operations in the feature extractions. Our clustering requires cosine similarity and mean computation using simple mathematical operations. Moreover, clustering finds a set of a representative convolutional patch, that are collected from training images. And these representative patches can be used as feature map to detect text. While Convolutional Neural network generates a set of patches for the same purpose. Fortunately, some researchers [7, 13, 39] have shown that convolutional feature maps can be produced using simple clustering techniques such as K-mean with different data representation and metrics.

However we use clustering algorithm in our work, but still, the whole process is not unsupervised; because At the sampling stage, we use labels for sample selection and equalization. On contrary, the labels are not used during the feature map learning in the Convolutional K-means.

The clustering algorithm tries to find some different groups in the data, such that

the cluster centers can represent the data that the goal is for. Each group contains samples that are similar to each other and different from other clusters' samples. In convolutional patch learning, the clusters represent shapes that are representative of the data. A good clustering algorithm provides some cluster centers that can represent the data (i.e each cluster center represents a set of similar samples). In our problem, since each cluster center, gets convolved with image and response is used a feature to classify the region; We look to achieve a set cluster centers that can represent most of the discriminative and representative characteristics of both foreground and background patches.

Due to nature of K-means algorithm, the quality of the clusters heavily depends on data representation, initialization, and the similarity metric used my the method. Measuring the quality of clusters is a subjective problem. One can use similarity/dissimilarity between cluster centers. In our problem, the quality of cluster centers is measured by their ability to split the text regions from non-text in the images. There are some algorithms for initialization and data representation that can be used to enhance the quality of cluster centers as discussed in Chapter 2.

Sampling data is an essential step for training any learning algorithm. More informative samples produce better learning outcomes. Random sampling is an effective method for text detection and has been used by both Coates et al.[7], and Zhu et al.[39] with some constraint. Both [7, 39] use an equal number of foreground and background samples. Coates et al.[7] and Zhu et al.[26] consider a sample as foreground if it has enough overlap with a character bounding in the ground truth and at least height or width should be within a small difference with the character bounding box height or width. See Chapter 2.

Initialization and a number of clusters are another two fundamental problems with clustering algorithms. Due to the high dimensionality of the data, distribution of the data is unknown and often complex. Thus there isn't a method that guarantees to provide a perfect initialization or an ideal number of clusters in the data. Random and incremental initialization algorithm is the most common used algorithms for initialization. Also, elbow technique is used to specify the number of clusters. see Chapter 2.

Zhu et al.[39] use an equal number of text/non-text patches from the gradient of grey images to learn a dictionary with 1000 patch using Convolutional K-means. They use cosine similarity as distance metric and select initial cluster centers randomly and from Support vectors (i.e. close to the decision boundary that separate text from non-text patches). The dictionary used in an experiment for detecting text in a natural scene.

Research Questions

In our thesis, we design experiments to answer the following research questions

- Does sampling based on visual within-class similarity and between-class dissimilarity improves clustering quality for text detection?
- Does initialization based on between-class similarity/dissimilarity helps improve text detection?

1.1 Contribution

We show that text detection rate can be improved by sampling with high cosine similarity and dissimilarity for training and initialization of the algorithms. We manage to use within-class cosine similarity and between-class similarity to collect training samples to initialize CK-means. In this thesis, a convolutional feature maps learning process for text detection is discussed. We sample data using within-class and between-class similarity/dissimilarity for convolutional patch learning. Then we compare our results with Zhu et al. [26]. We present our techniques and their effect on text detection accuracy.

The main contributions of the thesis are:

- Visual Similarity Sampling is presented to collect samples and initialize convolutional k-means. The features are learned automatically from the image data. The discussions are included in Chapter 3.
- Providing an open source system for sampling, feature learning and text detection.

1.2 Organization of This Document

The thesis is organized as follows. In Chapter 2, previous work about text detection is presented, including different ground truth, evaluation and feature learning methods. In Chapter 3, different data sampling and initialization for feature learning are proposed and analysed. In Chapter 4, results of the proposed techniques, and their effect on text detection is discussed.

Chapter 2

Background

Text Detection and localization is a challenging due to variations in text size, font color, scene complexity, uneven lighting, blurring, aspect ratio and distortion[34]. Text detection is a primary feature for systems that use textual information in images such as driver-less cars, searching images by content, etc. The detection needs to be accurate and efficient in term of where the text is (i.e localization), then extract the individual characters or Connected components CCs of the text.

Researchers try to solve this problem by detecting word, character or text-lines in images and generate bounding box around them. Text localization applies to natural scenes without any graphical texts (i.e. no text is added after the picture being taken). Figure 2.1a shows an example with the ground truth word bounding box. The detection process usually starts with a preprocessing step such as, smoothing, sharpening, edge detection, rotating, etc. Then the detection and the localization of word, character or text-line applied. The detection process usually uses a pre-learned dictionary as feature map to extract features from the image and train a classifier to detect regions with/without text. Clustering and Convolutional Neural Networks CNN are two most common algorithms to learn dictionary. The detection usually happens in multiple stages. The first stage is a raster scan of the image and classifies each sliding window as text/non-text. The steps after are; Connected component (CC) extraction or validation step to remove false positive examples.

(a) (b) (c)

Figure 2.1: Different ground truths that are available in ICDAR 2015. In (a) word bounding box are represented by the green boxes. (b) is a pixel level ground truth. In (c) character bounding boxes are represented by green box.

2.1 Data

ICDAR is the International Conference on Document Analysis and Recognition ICDAR[14]. We consider robust reading competition Focused Scene Text, which has been held in 2003, 2005, 2011,2013 and 2015.

ICDAR 2015 Focused Scene Text provides three types of ground truth. First, bounding box for words in each image. Second, bounding box for characters in the image. Third, character label images, which has a black background with white characters. It contains 228 training images and 233 testing images. Texts in those images are from the original scene. They contain handwritten text as well. The texts are different in color, font size orientation, etc. The image backgrounds are also complex. They include the natural scene, such as buildings, leaves, bricks, etc.

2.2 Ground Truth

To show text in an image, location of the text is an essential way to show where it is. In most of the text detection competitions, a (.txt or .xml) file is associated with each image, which indicates the location of the text.

Bounding Box Level is the most used method for text detection. At this level the associated file stores the bounding box information for each word, character or text line if exist in the image. Each line in the file represents a bounding box (i.e. X and Y coordinates, width and height). In our work, we use character bounding box to train a classifier for text localization, but word bounding box is used by ICDAR[14] to evaluate the final word detection. see Figure 2.2

Pixel level This level has a binary image associated with the original image. The binary image has a black background and white foreground which represents the text in the original image.

Bounding box level is not considered as the best method because it is not accurate enough. For example, if we have a character inside another character, their bounding box are overlapped. It is difficult to use the bounding box information to evaluate text localization performance correctly [26]. Both bounding box and pixel-level image can be used to define more meaningful text patches and remove overlapping characters.

2.3 Evaluation Methods

The most used metrics for text detection evaluation are precision, recall and f-measure. Precision is the percentage of retrieved instances that are relevant. Instance can be words, characters, text-lines or patches. Recall is the percentage of relevant instances that

Figure 2.2: Show some images from ICDAR dataset. The green box shows the bounding box for each word in the ground truth [26].

have been retrieved over total relevant instances in the image. F-measure is the harmonic mean of precision and recall.

$$Precision = \frac{|\{relevant\ instances\} \cap \{retrived\ instances\}|}{|retrived\ instances|} \tag{2.1}$$

$$Recall = \frac{|\{relevant\ instances\} \cap \{retrived\ instances\}|}{|relevant\ instances|} \tag{2.2}$$

$$F - Measure = 2 \times \frac{precisoin\ \times\ recall}{precision + recall} \tag{2.3}$$

The evaluation method uses overlap ratio detection protocol[34] from ICDAR robust reading competition.

These metrics are computed at following three levels to evaluate the system performance. both character and word level uses bounding box while pixel level uses pixel values for evaluations.

- **Character Level** The detected characters' and the ground truth bounding box are used to compute overlap ratio. Any character with 50% or less overlap are discarded [14], otherwise, the overlap rate is considered as detection rate. (i.e., both groundtruth and the detected character should have 50% overlap or higher) Then the metric are computed based on the detection rate.

- **Word Level** The bounding box of detected words are used to compute overlap with ground truth bounding box. Each detected word bounding box gets compared with the corresponding word in the ground truth. The overlapping region between them is computed as detection rate. See Figure 2.4. Any word that does not have 50% overlap or more with any words in the ground truth will be discarded[14], otherwise considered as true detection (i.e., both groundtruth and the detected word should have 50% overlap or higher). Some images from ICDAR dataset along with their bounding box ground truth is shown in Figure 2.2

- **Pixel Level** The detected characters pixel are used with pixel level image in the ground truth to compute true metrics in each image. The ground-truth in pixel-level are images with black background (i.e '0') and white(i.e '1') pixel in place of the text. At this level, the performance is computed using the fraction of overlapping correct detected pixels(i.e., the pixels that are marked to be text in the image) with white pixels in the ground truth.

At the first stage of the system, we evaluate the classifier performance by computing Area Under the Curve (AUC) and f-measure[7, 39]. AUC is been used by both Coates et al.[7] and Zhu et al.[26]. AUC evaluates the performance of the classifier on a set of isolated

text candidate patches. We follow their instruction in order to be able to compare my results with theirs. At this level the instances are individual 32×32 patches. To compute AUC, 50,000 patches with size 32×32 are extracted from training images randomly with equal number of foreground(i.e., patches with character) and background (i.e., patches without character)[7]. Then the cluster centers (i.e., the output of the CK-means) is used to extract features from each patch. Train an AdaBoost classifier with 1000 iterations using 80% of the data (i.e 40,000 samples)for training. Then use the other 20% of the data for testing and plot precision-recall PR curve by defining a set of thresholds. Finally, compute the area under the precision and recall curve. See Figure 2.3.

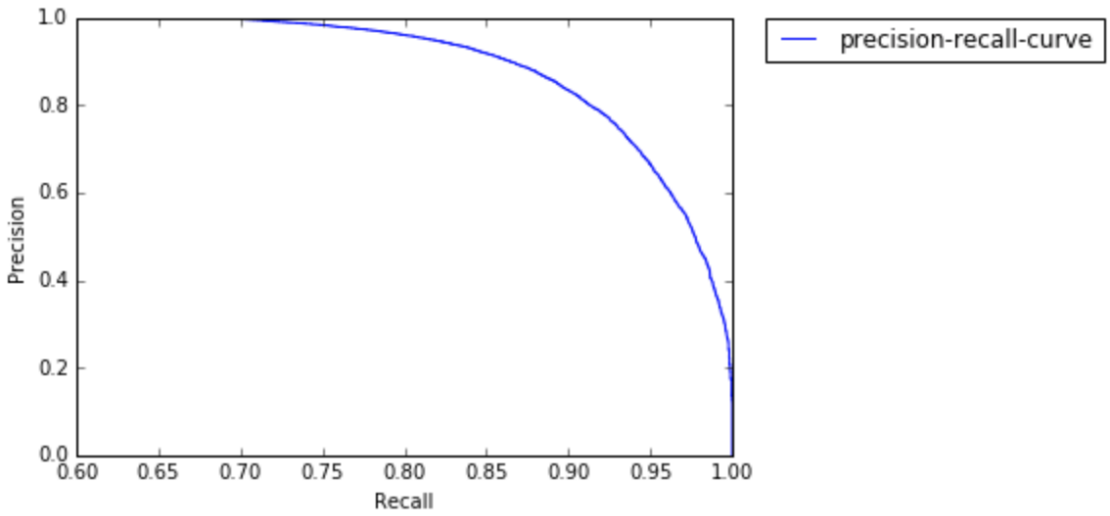


Figure 2.3: Show an example of precision-recall curve.

(a) Groundtruth (b) Detection

Figure 2.4: These images generated using on-line evaluation tool. (a) shows the bounding box of word in the ground truth in ICDAR Robust Reading Competition[1]. The green box shows the correct detection words but the red boxes shows the missed words in the detection. (b) shows the bounding box for the detected words. Red regions in the detection (if exist) indicates extra words that has been detected but it is not a word in the ground truth.

2.4 Text Detection using k-means for Feature Learning

Sliding window over images is the most commonly used method for text detection[8]. This method uses a fixed size window and slides over the image. At each position, we extract features and classify the window. Researchers use different methods for feature extraction from the window (i.e., patch). Using a dictionary as a feature map is a very common method for feature extraction from patches.Those dictionaries can be learned from clustering methods[7, 26] with different data samples and initializations [2, 13, 33]. The dictionary can be learned with/without the class label. Both [2, 9] exploit the class label to compute probability distribution for dictionary learning. Akhtar et al.[2] propose a Bayesian approach to learn discriminative dictionaries for sparse representation of data. Their approach adaptively builds the association between the dictionary atoms and the class labels such that this association signifies the probability of selection of the dictionary atoms in the expansion of

[1] `http://rrc.cvc.uab.es/?ch=2&com=introduction`

class-specific data. The produced dictionary is test on two face dataset and they show that the proposed Bayesian approach consistently outperforms the existing approaches.

Dahl et al[9] uses and iterative algorithm to initialize vector quantization algorithm. The initialization randomly select one sample and uses similarity to make sure that the next samples are different. The procedure makes sure that the initial samples are unique. Then Dahl uses a modified vector quantization to build a dictionary, but the class label information is included in the learning procedure. The dictionary samples are iteratively updated to be moved towards image patches that have similar labels as the ideal label atom and away from dissimilar patches. The final output of the algorithm provides a discriminative dictionary D along with associated label L of each atom in the dictionary.

On contrary, Zhu et al. [26], Coates et al. [7] and Juvonen [13] don't use the class label in the learning procedure (i.e., They use unsupervised clustering method). Juvonen extracts every possible 4×4 from training images and uses a regular k-means to learn the dictionary. The regular k-means use Euclidean distance as similarity measurement. The resulting dictionary is used in an experiment for image restoration. The goal of their dictionary Learning is to approximate a matrix factorization problem (i.e., for a dataset X Find D and A such that $X \approx DA$)

A variant of K-mean have been used by Zhu et al.[39] and Coates et al.[7] to learn a convolutional feature map. The algorithm uses cosine similarity to compute the distance between training samples and cluster centers. Cosine similarity is shown to be a good metric, as only non-zero elements are considered[24]. They use the produced dictionary to extract the feature from text and non-text patches then classify them. They have shown that the convolutional k-means achieve state-of-the-art performance at that time.

Both [7, 26] use random sampling to collect training data set to learn a dictionary. while [2, 9, 13] uses all the collected samples for training. Zhu et al. managed to use support vectors SV along with random selection to collect samples for training and initialization of the clustering algorithm. Zhu showed that the combination of SV and random selection improves the detection by (1-2%).

The closely related work to ours are Coates et al.[7] and Siyu et al.[26]. They collect 32×32 patches from the text and non-text regions, then use Convolutional K-means(CK-Means) to learn a convolutional dictionary from training image. They use the dictionary to extract features from 32×32 patches of an image and then classify them as text/non-text at the coarse and fine detector. See Figure 2.8.

They report that Sobel technique is used with gray images to get the gradient of the image as input for the system. However, we tried to replicate their results using Sobel edge on the gray image using codes from Zhu et al. [26], but our achieved result was not close to what they had. We found that the Sobel technique on the gray image will not emphasize character edges very well. We even lost some character edges (i.e we did not get any edge for the character). We managed to use Sobel on all three channels (luminance and colors) of L*a*b color image separately, L*a*b represent each pixel of the image by L luminance, and colors (a, and b). Then add the edges up together. It gave us a better edge image, character edges are emphasized and we don't lose any character edge in the images. The trick is that character edges tend to be represented by both intensity and color gradients. The results of different pre-processing are shown in Figure 2.5

(a) Original image

(b) Sobel on gray image

(c) Sobel on L*a*b image

Figure 2.5: Different pre-processing and edge detection technique that has been used during our replication process experiment

2.4.1 Sample Collection

To create the dictionary, one needs to collect samples directly from the image. The samples are 8×8 patches. For this purpose, Zhu et al. extract 32×32 patches from the image then label them as foreground/background. A patch is considered as foreground if it has 56% or higher overlap with any word bounding box in the image with 10% or less different in height or width. Zhu collects 50,000 32×32 samples randomly from both foreground and background. Then from each 32×32 four random 8×8 patches are selected then converted into a vector of length 64 and normalized into a unit vector. Totally, 200,000 8×8 samples from ICDAR2015 dataset are used to train the Convolutional K-means and learn a codebook of 1000 templates.

2.4.2 Convolutional K-means

It is originally used by Coates et al.[7] to learn convolutional features. It is originated from the simple k-means algorithm, updating sample cluster labels and shifting cluster centers iteratively to maximize inter-cluster distance and minimize intra-cluster distance. The only different is; in CK-means, instead of using Euclidean distance, cosine similarity is used as similarity measurement with all patches converted into vectors.

For sample dataset containing m samples with n features, we make the matrix size as $m \times n$, where each row stands for a sample vector and each column corresponds to a feature scalar, $X \in \mathbb{R}^{m \times n}$. For initialization, Zhu randomly picks k samples from the sample matrix X, normalize each vector into a unit vector, so cluster center matrix $D \in \mathbb{R}^{k \times n}$. k is the number

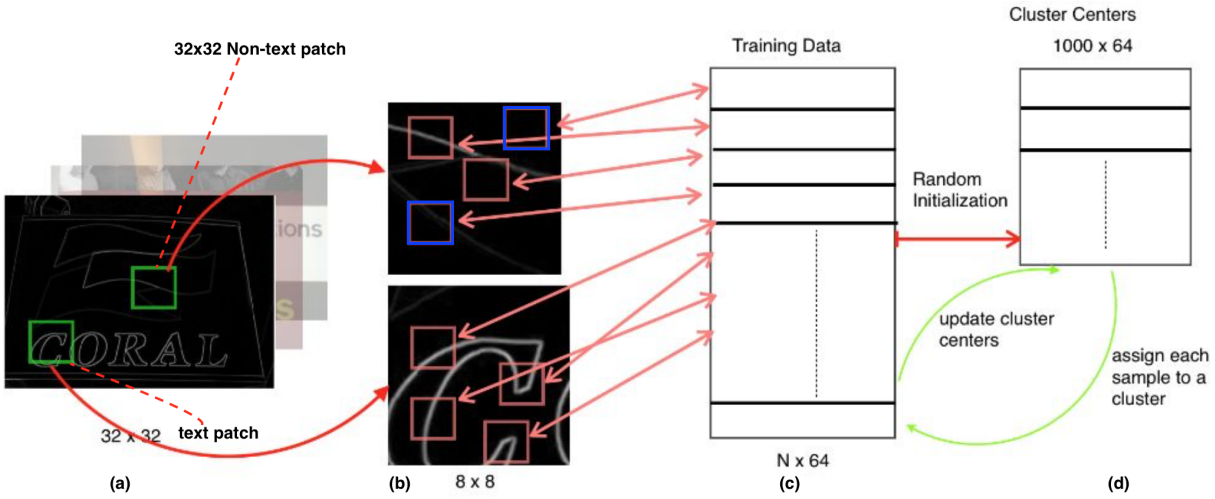


Figure 2.6: Show sample collection and the Convolutional K-means. (a) is a gradient of the image. (b) Shows two 32×32 sample patches from the image on top from non-text and bottom from text regions. it shows how four random 8×8 patches are selected. Note that there are many chances to select empty 8×8 patches. See the blue squares in (b). (c) is the training data set with size of $n \times 64$ where N is number of collected samples, each row is a vectorized sample, and columns shows features in the sample. (d) is the cluster centers randomly selected from training set. each row is a normalized center.

of clusters. The goal here is to minimize the equation:

$$\Sigma_i \left|\left| D_{s_i} - x_i \right|\right|^2 \tag{2.4}$$

Each cluster center (i.e. rows in D) is the normalized basis vector. s_i is a hot encoding which allows only one non-zero element representing the cluster center (row of D) training sample x_i belongs to. To find a matrix D with minimum overall distances from samples to cluster centers, we alternatively minimize D and s_i.

Having D fixed, we solve for s_i by letting $s_i[k] = D^{(k)^T} x_i$ for

$$k = argmax_j D^{(j)^T} x_i$$

where $s_i[k]$ means kth elements of s_i, $D(j)^T$ means the transpose of jth column of D. And other elements in s_i except $s_i[k]$ are set to zero. Having s_i fixed, we solve the minimum of equation 2.4 for D in closed form.

D and s are updated alternatively, and the learning process stops after a certain number of iterations or the overall distance is less than a threshold. Rows of D are then used as a feature map. These quantized features, unlike features designed with prior knowledge, are learned automatically from the data without any human interference. These feature maps are (8×8) patches. They represent patterns such as vertical/horizontal lines, corners, zebra, etc.

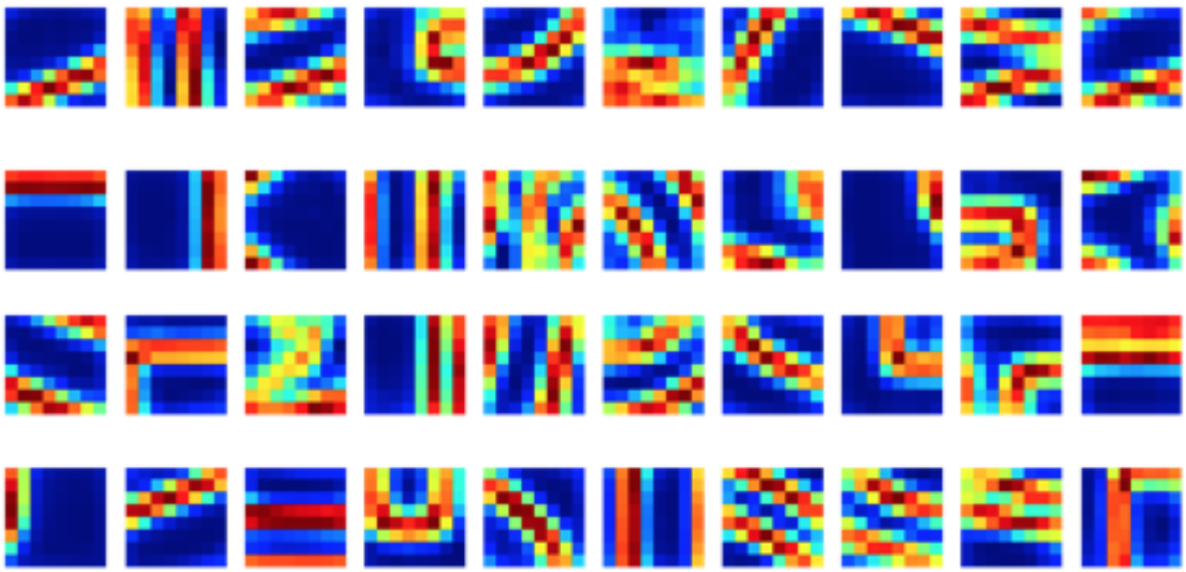


Figure 2.7: Shows some patches from the codebook that has highest confidence from Zhu et al.[26] coarse detector classifier. Red color shows highest value (i.e., one) and blue shows background (i.e., zero)

In Zhu's work, CK-Means is exploited and 1000 sample as the initial cluster centers

are selected randomly. Cosine similarity has been used to find the similarity between samples and the centers. At each iteration, samples get assigned to nearest center and then the mean of each cluster becomes a new center for the next iteration. The average movement of centers is captured at each iteration. The learning process stops after a certain number of iterations or the overall distance is less than a threshold. The final cluster centers can be used as a dictionary of patches (i.e. templates) to extract features from the image to separate text from non-text patches[7, ?]. Zhu et al used random selection for initialization. The sample collection and CK-means are shown in Figure 2.6.

Zhu improved detection simply by initializing CK-means from support vectors. He used SVM to find the discriminative samples (i.e., samples closer to decision boundary). The SVM finds a decision boundary that best separate two group of data by maximizing the margin between data points and boundaries. The closest points to the decision boundary are called support vectors, meaning they are supporting the maximum-margin hyperplane. The weight of support vectors are used as probability of selecting 1000 samples randomly as initial centers.

These patterns are sufficient for text detection because texts are a combination of these patterns. Some of the learned patches from Convolutional K-means are shown in Figure 2.7

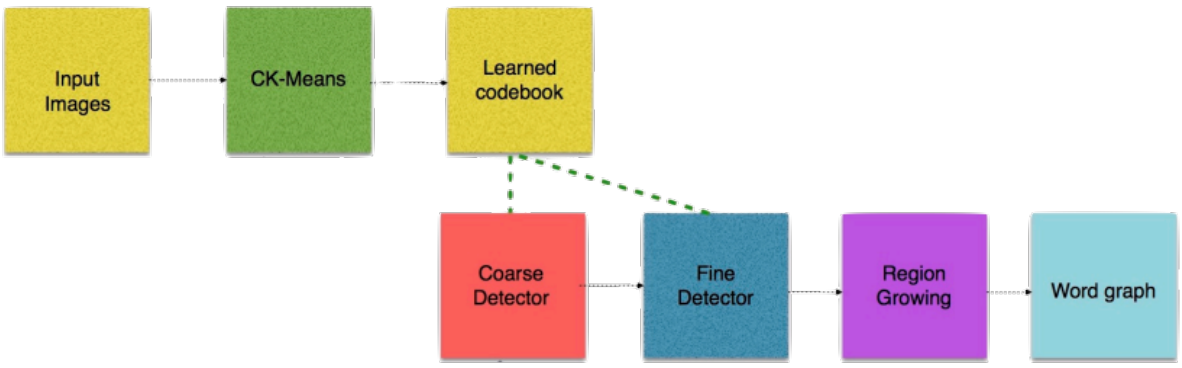


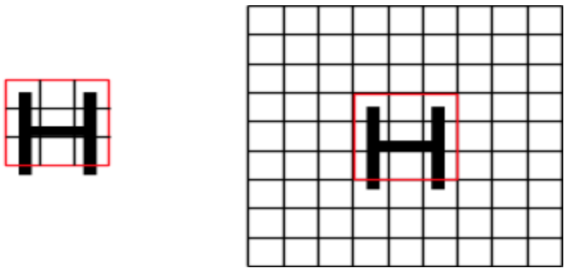
Figure 2.8: shows the overall structure of Siyu et al.[26] system.

2.4.3 Coarse-detector

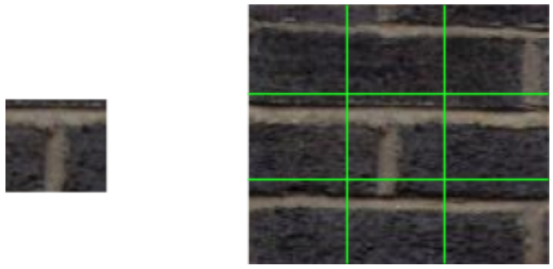
The dictionary is used to convolve with the image to extract features. Each sample from the dictionary gets convolved with the gradient of the image. Zhu et al.[26] use the codebook to convolve with the gradient of the image. They use mode 'valid' for the convolutional. This mode ignores the regions that don't fit the template, thus it produces an image with a smaller size than the original. The size of the output image will be $(M - m \times N - n)$, where $m \times n$ is the size of the template and $M \times N$ is the size of the original image.

32×32 patches are extracted from the image and spatial pooled to a grid of 3×3. Spatial pooling is basically dividing the patch into 3×3 regions and choose the maximum value within each region as a representative of the regions. Patches have 16-pixel overlap with patches aground it. The total number of features for a single 32×32 patch is equal to $(K \times 9)$ where K is the number of templates in the codebook. Zhu showed that using contextual information helps increase the accuracy of the detection. Contextual, includes

features from the 8 patches around each patch. Thus the total number of features becomes $M \times 9$ (blocks) $\times 9$ (features in each block). The difference between local and contextual are shown in Figure 2.9



(a) Local and contextual illustration



(b) local and contextual information for an ambiguous example

Figure 2.9: Shows the difference between contextual and local information, (a) show the local 3×3 and 9×9 contextual detection windows. (b) resolving ambiguous local features by context.

Training samples includes 72000 samples (32×32) with an equal number of foreground and background samples from ICDAR 2015 images; 80% of them are used for training and the 20% are used for evaluation. The equalization happens within each image within each scale. background patches are usually generated more frequent, thus we randomly select as much background as foreground samples in the current scale of the image. for the coarse

	Dataset				
Stage	#	**Background**	#	**Foreground**	**Classifier**
AUC	25,000	ICDAR	25,000	ICDAR	Regular AdaBoost
Coarse detector	36,200	ICDAR	36,200	ICDAR	Confidence-weighted AdaBoost
Fine detector	50,000	ICDAR	50,000	Tao Wang. [28]	Confidence-weighted AdaBoost
1st verification	4,500	ICDAR pixel-level	7,500	ICDAR pixel-level	Confidence-weighted AdaBoost

Table 2.1: The datasets and classifiers that are used in different stages of Zhu et al. [26] System. Note that regular AdaBoost classifier is used for AUC, because it is originally used by Coates et al. [7], then Zhu et al. [26] uses the same classifier to compare results. To have a fare comparison we used the same classifier to compute AUC. All the counts are number of 32×32 foreground/background patches.

detector, a patch is considered as foreground if it has 56% overlap and its width or height with 10% different with any character bounding box in the ground truth.

The raster scan happens in up to 30 different scales in the image. The size of the image reduces by 10% at each scale. We stop the downscaling if either height or width of the image became less than 50 pixels. The output of the classifier produces a heat map for each scale, based on the confidence. Higher confidence shows more likely to be a character in the image. For a final image, the heat-maps are combined using maximum pooling across all scales. See Figure 2.10

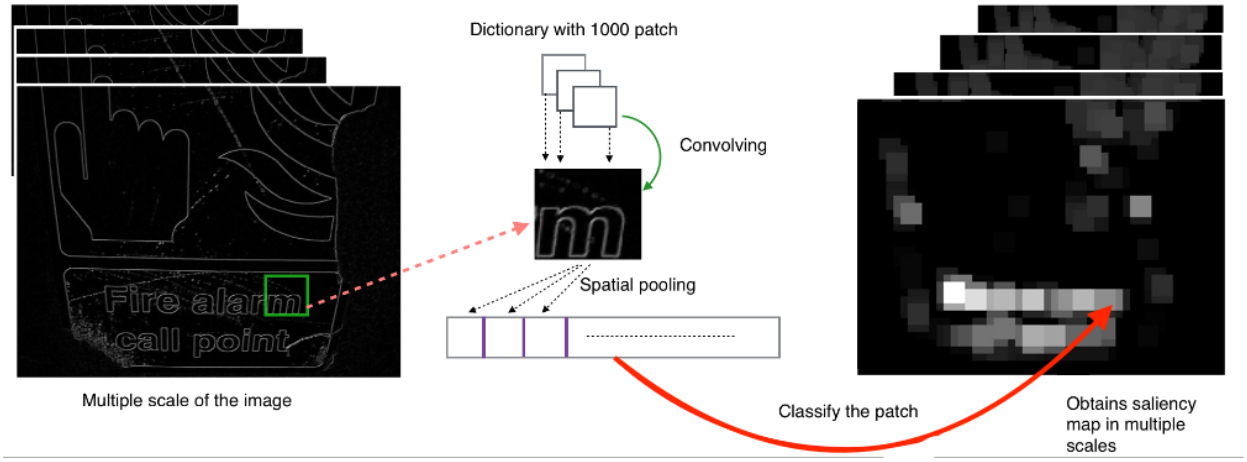


Figure 2.10: creating hot-map in multiple scales and combine them. The images are from Siyu et al's result using contextual information for coarse detector.

2.4.4 Fine detector

The next step Zhu managed to scan the region of interest provided by the Binarized hot-map from the previous step. They scan with higher resolution (step size of 1) to examine those areas. Any patch's center point that gets classified as text, used as a seed point for the next step. They call this step Fine-detector. Only local information is used to classify patches in the fine detector. The final output of the fine detector will be a list of seeds, which represent the center points of the text patches. Each region gets scanned in multiple rotation and aspect ration. Zhu et al.[26] showed that the rotation and aspect ratio help the system to keep recall higher without losing precision.

2.4.5 Region Grow

Then from each seed point, a flood filling algorithm is used to grow regions and get the whole body of the character. Both edge and color difference is considered to stop growing

the region. They use Sobel edge detection on RGB color channels. The cost function of The region growing criterion C is defined as follows:

$$C = \cos(\delta\Theta)\lambda + \frac{\Sigma_{c\in R,G,B}(|I_{c,q} - I_{c,seed}|)}{Z} \tag{2.5}$$

where the first term represents the edge intensity $\delta\Theta$ along the growing direction λ, and the second term represents the color difference between boundary pixel $I_{c,q}$ and the regions seed pixel $I_{c,seed}$. Z is a normalization factor, for the data type of 8-bit integers, the value of Z is 256[26]. Initially, seed pixels and region boundaries are labelled as foreground/background respectively. Unlabeled pixels with minimal cost are labelled iteratively. Region growing stops when no unlabeled pixels exist between foreground and background.

Many false positive characters will be generated in this step. Thus a verification step is used to filter out those CCs. They use the learned codebook again here to extract features from each CC along with some bounding box features.

2.4.6 Verification

Another confidence-weighted AdaBoost is trained using pixel-level ground truth. 4500 foreground and 7500 background samples are extracted from the pixel-level ground truth. We grow regions from some random seeds over the characters. The random seeds make sure all the characters within the images have at least 3 seeds. To over generate samples and achieve a various version of the same CC, the images are blurred using different gaussian filtering and edges are thresholded before growing the seeds. Gaussian filters with the size of 3,5,7,9,11 are used with sigma equal to half of the filter size. Edge thresholds of 50%,60%,70%,80%

and 90% are used. A CC is considered as foreground if it covers 80% or higher pixel overlap with any character in the ground truth. Then the gradient of the patch is computed using Sobel and convolved with the templates in the codebook. Finally, each convolved patch is spatial pooled into a grid of 3×3. A total number of features equal to $M \times 9$ where M is the number of templates in the codebook. Zhu used all templates in the codebook (i.e M = 1000).

The generated CCs from the flood filling algorithm is verified using this verification step. The survived CCs considered as a true character and get passed to the next step.

2.5 Confidence-weighted AdaBoost

In Zhu's system, confidence-weighted AdaBoost classifier is used to detect text/no-text[20]. Unlike the original AdaBoost which provide a hard label [-1,1], the classifier provides a confidence of how likely a patch is a text. Then a hard threshold is used to keep the recall higher than 95% with a maximum possible precision. A decision stump is used as a weak learner to find the best split at each iteration. The error reduction becomes very small after 1000 iteration, thus we stop the learning after 1000 iteration. The whole pipeline of Siyu et al's work is shown in Figure 2.8. The Datasets and classifier in each stage of Zhu's system is shown in table 2.1

2.6 Summary

In this chapter, we talked about text detection systems. Detecting text in the natural and wild scene has attracted many recent researchers attention. The texts vary in the font, size, color, lighting, etc. Also, the complexity of the backgrounds, make the problem

more challenging and interesting. Different algorithms for sampling, dictionary learning and text-detection are presented.

ICDAR Robust Reading competition is presented with some samples and their ground truth. Evaluation methods are discussed in both pixel and bounding box level. The Evaluation metrics are also presented. Text detection systems are usually evaluated using Recall, precision and f-measure. ICDAR 2015 evaluation includes three levels, bounding box level for word and characters and pixel-level. Precision, recall and F-measure are computed at each level. They subject to the overlapping area between detection and ground truth. The data sampling is presented. A sampling includes random selection with 80% overlap or 56% over along with height and width constraint are used to label a sample as foreground. We propose a new way of sampling to get more representative features, that can represent the discriminative characteristics of both text and non-text regions.

Chapter 3

Methodology

In our thesis, an existing system for feature learning and text detection, which use Convolutional K-Means is improved. From the experiments in Chapter 3, we show that sampling using representative and discriminative samples from each class helps improve text-detection. We show that how the text-detection rate changes by selecting samples with high visual similarity and dissimilarity in both within-class and between classes.

Convolutional K-mean is a simple and effective algorithm for feature learning, but the training set is essential input that needs to be representative of the problem. We proposed a general algorithm that can be used for any pattern recognition problem. Our algorithm tries to find the most representative samples in each class by selecting samples with highest within-class similarity and high between-class dissimilarity as training data set.The intuition behind the VSS is to increase the probability of the most frequent shapes and most unique shapes within a character patch. In Section 4.2.1, we present the results and effectiveness of our algorithm. We compare the results with Zhu's reported results and replicated results which we've produced from Zhu's re-implemented system.

We use Visual Similarity Sampling VSS to select samples from images as the training set. The most similar samples (i.e., most frequent) from each class and most similar samples between classes are used as initial cluster centers for the CK-Means. We also show how

we get the same result of SVM initialization used by Zhu, by using between-class similarity/dissimilarity. We show that how the detection rate changes after using samples closer to decision boundary (i.e., samples with high between-class similarity) and using samples farther from decision boundary (i.e., samples with high between-class dissimilarity).

3.1 Data

ICDAR 2015 Focused Scene Text data set is used. It contains 229 images for training and 233 images for testing. As described in section 2.1, they contain texts in natural scenes with complex backgrounds and various text size, color, and orientation. It provides color images and the bounding box of each character and words. It also provides label-image (i.e., pixel-level image). It is a binary image in which '1' represent character pixels and '0' represents background pixels. The gradient of the color image is used for Convolutional K-means sample collection and the detection. The label-images are used for verification classifier. see Section 2.4.6.

We are using the gradient of the images to collect training samples. As shown in section 2.4.1. Sobel edge detection is used on the L*a*b image to compute the gradient on all three channels and combine them as final gradient image. L*a*b image exploited because we believe character edges constitute of both color and luminance edge. K-means.

3.2 Feature Learning

We created a text detection system that uses convolutional features learned from convolutional K-means. We are trying to show whether we can improve the detection rate and/or simplify the problem using convolutional features combined from a set of representative and discriminative samples.

Features called representative because they have the highest within-class similarity and the discriminative patches are those with high within-class dissimilarity. The representative/discriminative patches are used as input of the CK-means. A set of the convolutional patch (i.e., templates) is learned from Convolutional -K-means. The same Zhu's CK-means is used in our work, we only changed the input patches. See Section 2.4.2 for more details about CK-means algorithm.

3.3 Visual Similarity Sampling

Data samples for the Convolutional K-Means are a set 8×8 patches collected from 32×32 patches from both foreground and background regions. see Figure 2.6(b). Coates et al.[7] managed to collect 60,000 8×8samples randomly from both text and non-text regions in the training images. In the meantime, Zhu et al.[26] collect 50,000 32×32 samples and from each, 4 random 8×8 patches are extracted. In total 200,000 8×8 patches are collected for CK-means training. see Section 2.4.1. Convolutional K-means is used to learn 1000 cluster centers from the collected 8×8 patches.

Zhu et al...[26] have selected four random patches in each 32×32, but they did not filter the random selected 8×8 patches. We found that a quarter of collected patches were empty even when the 32×32 patch is labeled as text. For example, The patch may be a

letter 'T', but if you select patches that lie at right or left side of the vertical bar under the top horizontal bar, Most likely the patch will be empty. Thus empty patch filtering is a fundamental step.

We collect the same number of samples but with more constraints. The first important constraint is filtering empty 8×8 patches. This can be done by counting the non-zero pixel after we extract the patch if all pixels are zero then discard it and select another one. This filter removes non-empty patches, but still, we may end up with many non-useful patches that represent nothing in text/non-text regions. For example, we may produce patches with a single non-zero pixel. Another constraint can be applied to select patches that are common or constitute a fundamental characteristic of a character. For the example in Figure 3.1, we aim to achieve patches with two vertical lines and patches with diagonal lines as a visual property of the character patch. Clearly, patches with two vertical lines are the most frequent shapes within the patch and patches with s small diagonal line on lower left are least frequent.

For this purpose, We consider the visual similarity and dissimilarity as properties of the samples before we select. We use average visual similarity as a score to computer the frequency of appearance of each sample in the patch. The similarity and dissimilarity value is used as the probability distribution of the random selected. Which means that we are not rejecting the random selection fully; we only add probability to each sample before we pick them randomly. Note that we are not selecting samples with highest visual similarity or dissimilarity, because we may end up with missing some useful shapes that did not get highest similarity/dissimilarity at any patch. Thus, using visual similarity as

probability leaves chance for all shapes to be selected within a shape. For example, if we have a letter 'm' selecting shapes with highest similarity would be vertical lines always and least frequent would be the arch shape always. The probability distribution makes sure that no most frequent patches have lower probability of selection than least frequent patches for the representative distribution.

The representativity show how common is a sample in that class. The similarity between samples can compute this and find the ones that have highest average in-class similarity. For $x_i \in T$ where T is the list of the samples with size $N \times 8 \times 8$, the visual similarity score of x_i is computed as follows.

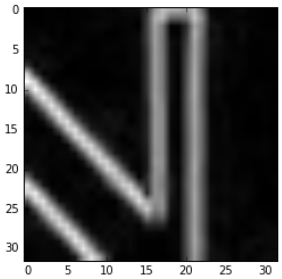
$$simi\_score(x_i) = \frac{1}{N} \times \sum_{j \in N} x_i \cdot x_j \tag{3.1}$$

Then we compute visual dissimilarity of each samples by subtracting the visual similarity from 1.

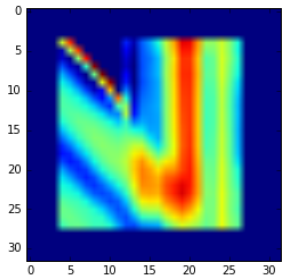
$$disi\_score(x_i) = 1 - simi\_score(x_i) \tag{3.2}$$

Samples with high similarity tend to be more representative of the patch. And samples with high dissimilarity shows the discriminative samples. We collect samples that are very common within-class and can be seen occasionally in other class. See Figure 3.1, (a) shows the similarity score map for every possible 8×8 patches while dissimilarity score map is shown in (b).

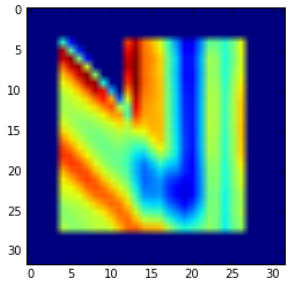
The aim of this patch collection from both background and foreground is to achieve a set of 8×8 patches that can represent and discriminate individual text/non-text patches. Thus, we try to capture the visual properties of each 32×32 character/non-character patch in four 8×8 patches. We select only four 8×8 from each 32×32, because Coates et al. [7]



(a) A 32×32 patch from text region.



(b) Similarity score map



(c) Dissimilarity score map

Figure 3.1: Similarity and Dissimilarity score map for every possible non-empty 8×8 patch from the 32×32 patch in (a). The red regions shows highest score, and blue regions shows lowest score in both (b and c). Notice that, the vertical line patches are the representatives because there are more patches with vertical line than diagonal.

and Zhu et al. [26] have used the selected four. Additional, we want to show the effectiveness of the sampling by keeping all variables the same as before except the distribution.
The gradient of the image is computed,Then passed through two stages to collect samples for the Convolutional k-means.

First stage is collecting 32×32 patches from both text/non-text regions in the image from 30 different scales. The second stage is selecting 8×8 patch from each 32×32 patches based on local cosine similarity and/or dissimilarity.

3.3.1 Stage One: Character Candidate Patch Sampling

This stage is for extracting 32×32 patches from both text and non-text regions. The gradient image is used to collect 32×32 patches with stride size of 16-pixel. Each patch has 16-pixel overlap with the patch. Zhu's restriction is used to label a patch as foreground; that is 56% overlap with 10% difference in height or width with character bounding box

in the ground truth. see Figure 2.6(a and b). Samples are collected in 30 different scales. An equal number of foreground and background samples are selected from each image. In practice, the equalization happens within each image and within each scale. At each scale, all the foreground samples are kept and same quantity samples are randomly selected from the background. For example, let's assume we have 10 foreground patches in a scale, so we will select 10 foreground samples at random in the same scale.

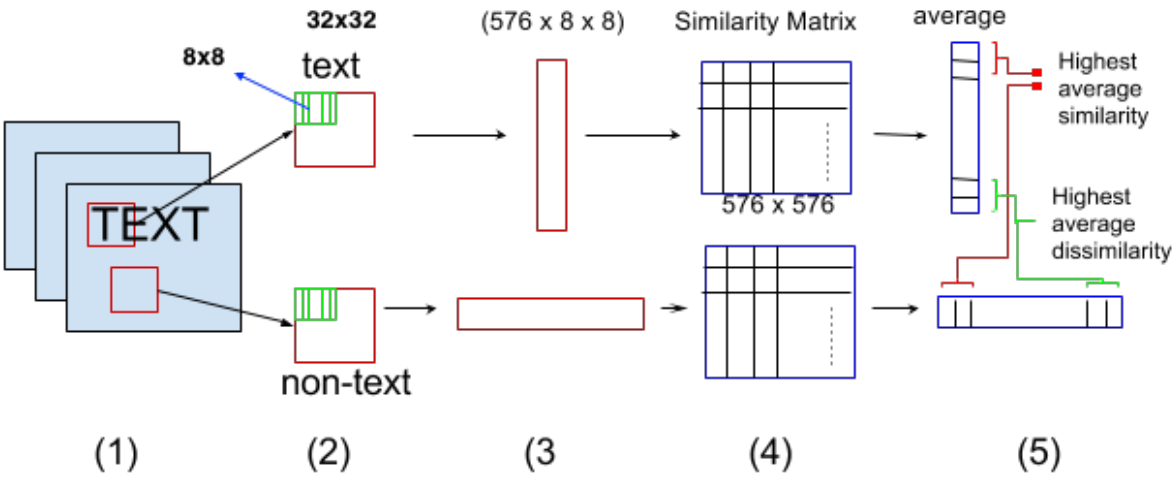


Figure 3.2: Illustration of how 8×8 patches are extracted and visual similarity score computed within a patch. The average similarity score in (5) is used as probability distribution to select samples in (3).

3.3.2 Stage Two : Character Candidate Sub-Patch (Feature) Sampling

This stage is for selecting four (8×8) samples using local visual similarity/dissimilarity score as the probability distribution (i.e., visual similarity/dissimilarity within a patch). From each 32x32 patch in the picture, we collect four 8×8 patches. We use every possible non-empty 8×8 patches from each 32×32 patch with stride size of 1; each patch has 7-pixel

overlap with patches next to it. Then compute the similarity/dissimilarity for all patches. using equation 3.1 and 3.2. Angular distance is used to compute similarity/dissimilarity between samples.

The algorithm for stage two sampling is as follows.

Create two empty lists ($S_{background}$ and $S\_foreground$) to store 8×8 patches. $S_{background}$ holds 8×8 patches that are extracted from non-text 32×32 patches and $S\_foreground$ stores 8×8 patches that are extracted from text 32×32 patches. For each image and for each 32×32 patch repeat the following steps.

1. Collect all N non-empty 8×8 patches in the 32×32 patches and save it in Q where Q is a matrix with size $N \times 8 \times 8$.

2. Compute average similarity for each sample and store it in matrix *simi*

$$\underset{i\in N}{simi(i)} = \frac{1}{N} \times \sum_{j\in N} x_i \cdot y_j \quad where\ x_i, y_j \in Q$$

   where *simi* is a matrix with size of $N \times N$.

3. Then normalize the vector.

$$P = \frac{simi}{\Sigma\ simi}$$

   P will be used as the probability distribution for selecting samples. To use dissimilarity, subtract the probability P from one.

$$P = 1 - P$$

4. Randomly select z samples according to the distribution given by P, then add them to ($S_{background}$ if it is a foreground patch, if it is not a foreground, add them to $S_{foreground}$). In our experiment $z = 4$

The illustration of using similarity as the probability of selecting patches randomly is shown in Figure 3.2 and an example in Figure 3.1. Higher the average in-patch similarity tends to be more representative. Any patch that has the highest similarity means it is very common and can be used as a representative sample of the character and vice versa. In the previous example, In the letter 'T' patch, most probably we get the highest similarity in those patches with a horizontal or vertical bar. Then we find the average dissimilarity from the average similarity and use it as the probability of selecting the other two sample. The pipeline for 8×8 patch extraction is shown in Figure 3.2.

The products up to this point will be two lists of samples ($S\_foreground$ and $S\_background$). $S\_foreground$ stores $N1$ 8×8 patches, which are collected from foreground 32×32, while $S\_background$ stores $N2$ patches from background regions.

The number of samples from each class can be very large. We are looking for roughly $N = 100,000$ sample for each class. Thus we use random selection with equal distribution to select the final set of samples.

3.3.3 Stage Three: : Seeding Initial k-means Cluster Centers (Patches)

This stage uses between-class and within-class similarity/dissimilarity to select initial cluster centers for the convolutional K-means. To select those number of samples, we exploit

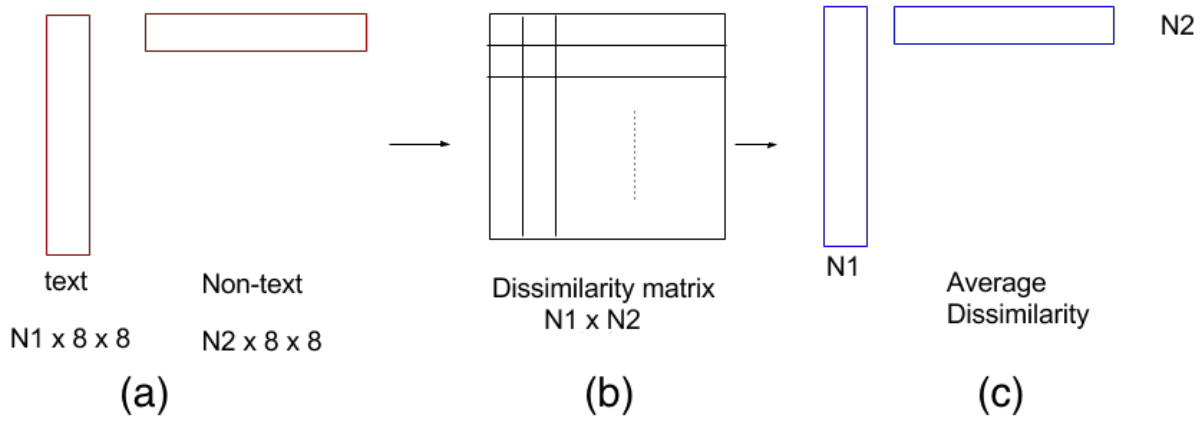


Figure 3.3: (a) shows the list of patches in each class where $N1$ and $N2$ are the number of patches in each class. (b) is the dissimilarity matrix between samples of the two classes. (c) Is the average dissimilarity for each sample. N1 is average row wise of matrix in (b). N2 is average column-wise of the matrix in (b). The values in (c) is used as probability of selecting samples randomly in (a)

within-class and between-class similarity/dissimilarity.

1. **Between-class similarity**.

   In this case, we create one big similarity or dissimilarity matrix with size about $(N1 \times N2)$, where $N1$ is a number of samples in $S_{foreground}$, N2 is the number of samples in $S_{background}$. Then finding the average column and row wise, gives us two vector $disi\_text$ and $disi\_non\_text$ with size of $(1 \times N1)$ and $(1 \times N2)$ respectively. $disi\_text$ stores between-class similarity/dissimilarity for foreground patches while $disi\_non\_text$ stores between-class similarity/dissimilarity for background patches. Then we can use the average column and row wise and use them as the probability of selecting a N sample from each class randomly. The illustration is shown in Figure 3.3. Clearly, if similarity is used in this stage, samples that are appear most, gets high probability

and vise versa. In contrary if dissimilarity is used, samples that are appear least, get high probability.

2. **Within-class similarity**.

   To find the most common patches in each class, the similarity between all the samples of the same class are computed. This option will generate two large matrices with the size of $N1 \times N1$ and $N2 \times N2$. Then averaging them give us two vectors $simi\_txt$ and $simi\_txt$ with size of $N1$ and $N2$ respectively. the within-class similarity of foreground patches are save in $simi\_txt$ and within-class similarity for background patches are saved in $simi\_non\_txt$.

3. **Harmonic mean**.

   We can combine these two similarity values into a single value by using the harmonic mean method, and use it as the probability of selecting samples from each class. See Equation 3.3 and 3.4. We are trying to select training patches that can represent its class (High similarity), and discriminate from other class (High dissimilarity). Harmonic mean is used because it assigns higher value for those inputs that are high in both similarity and dissimilarity. For example, let's say we are looking for samples with high between-class similarity and high within class similarity, The sample that has 0.5 in both are more preferable that having 0.8 in one and 0.2 in another.

$$Harmonic\_mean = \frac{2 \times simi_{wclass} \times simi_{btw}}{simi_{wclass} + simi_{btw}} \tag{3.3}$$

$$Harmonic\_mean = \frac{2 \times simi_{wclass} \times diss_{btw}}{simi_{wclass} + diss_{btw}} \tag{3.4}$$

The similarity/dissimilarity and the harmonic mean can be exploited for both training set collection and for initial cluster center selection. One can select 1000 samples from the training set using average dissimilarity/similarity. The average between-class similarity will provide samples closer to decision boundary while average between-class dissimilarity provides samples further to decision boundary. Furthermore, within-class similarity provides common or representative samples in the class, while, within class-dissimilarity provides rare samples in the class. A similar technique is used by Zhu et al[26]. They train an SVM to get samples close to the boundary between text and non-text samples. Then 1000 samples selected from both achieved support vectors and random. See Chapter 2. Similarly, we try to collect using representativity within class, and combine it with similarity and/or dissimilarity between class. Combining within-class similarity and between class similarity using harmonic mean, provides samples closer to decision boundary and preferring those that are more frequent within a class. We can achieve representative samples from each class and far from decision boundary by combining within-class similarity and between-class dissimilarity

Convolutional K-mean uses the samples to find 1000 cluster centers. we use stage three to achieve initial cluster centers. In order to be able to show the effectiveness of our technique, we compare our result to both Zhu et al.[26] and Coates et al.[7]. We create a codebook with 1000 samples which represent the final cluster centers. The final cluster centers are used as feature map to extract features from images.

The training set X containing m samples with n features. In our experiment, $m = 200,000$ and $n = 64$. Each row is a sample vector, and each column corresponds to a feature, $X \in \mathbb{R}^{m \times n}$. After initialization, normalize each vector into a unit vector. Cluster center matrix $D \in \mathbb{R}^{k \times n}$, where k is the number of cluster centers. The goal is to minimize Equation (3.5) (see Coates et al.[7]):

$$\Sigma_i ||D_{s_i} - x_i||^2 \tag{3.5}$$

Each cluster center (i.e. rows in D) is the normalized basis vector. s_i is a hot encoding which allows only one nonzero element representing the cluster center (row of D) that training sample x_i belongs to. To find a set of cluster centers (i.e. matrix of D) that minimizes the total distance between samples and cluster centers, we alternatively minimize D and s_i. In our experiment, $k = 1000$ cluster centers are learned.

The convolutional K-mean with cosine similarity is used to learn codebook with 1000 centers. Then we extract features from each 30×30 patch in the image and then classify them. This step is very close to Zhu et al.[26] (See Section 2.4.2)

3.4 Coarse detector

The coarse detector is used to detect and tell where likely is text in an image. The coarse detector provides a hot map, where the peaks represent the text regions in the image. It uses the produced codebook from CK-means as convolutional feature map to convolve with the gradient of images and extract features. Patches from the codebook get convolved with the image and every possible 30×30 patch is extracted with stride size of 15. Every

patch has 50% overlap with the patches around it. Then each patch is spatial pooled into a grid of 3×3 using maximum pooling. Zhu et al were using a patch of 32×32. clearly, when you apply 3×3 spatial pooling you need to have at least 2-pixel overlap between pooled regions. see Figure 3.4 a. Thus, if a pixel with z large value lies in the overlapped regions will be used as the maximum value for both regions. We manged to used smaller window size (i.e. 30×30) so when we apply spatial pooling, there will not be any overlap between pooled regions which accelerates computation greatly. We pool maximum value from each 10×10 regions. see Figure 3.4 b. For each 30×30, we achieve 9 features (local only) multiplied by a number of templates in the codebook. Since Zhu showed that contextual information helps increase detection, we use contextual information as well in our experiment. Contextual information uses the 9 local feature with features from 8 neighboring blocks around it. see Chapter 2

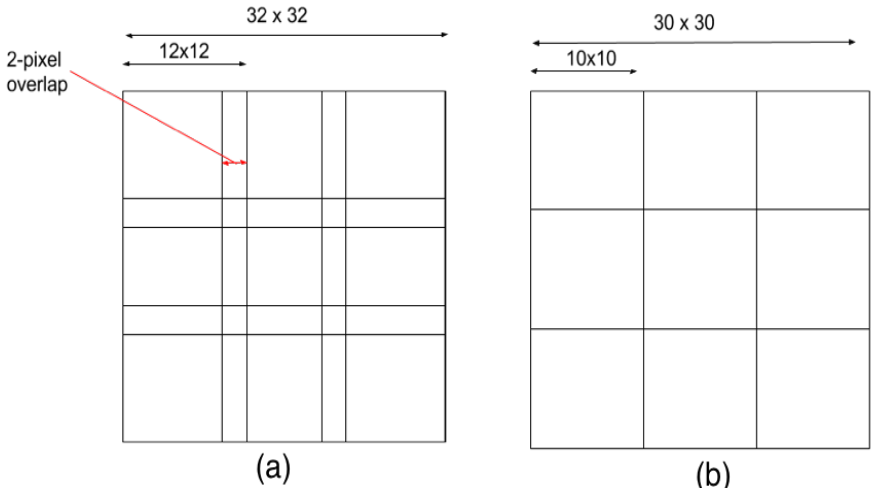


Figure 3.4: (a) is showing the pooling procedure applied by Zhu et al.[26], which has 2-pixel overlap between 2 pooling regions. (b) shows our technique to avoid the overlap.

We will test both local and contextual information. Local information will provide a

training set of $N \times 9000$. The contextual information will produce a training set of $N \times 81,000$ where N is the number of 30×30 samples.

A confidence-weighted AdaBoost is used to classify individual patches. As described in Section 2.5. Decision stump is used as the weak learner of the classifier. Confidence-weighted AdaBoost classifier provides real positive/negative value as output. The value shows the confidence of how likely the patch text(positive) or non-text(negative). We stop the classifier at 1000 iterations (i.e., up to 1000 decision stumps). The performance of the classifier monotonically increases on the training data with the number of iterations, but the difference between 1000 stump and 2000 stump is very small which does not worth to go more than 1000. As shown by Zhu et al.[26] the difference in performance between 1000 and 2000 stumps is less than 1%.

The coarse detector is used to classify every possible 30×30 patch in 30 different scales of the image. The value of each patch is replaced by the output of the classifier to create a heat map for each scale. A hard threshold is applied to the output hot map images to get one binarized image for each individual scale. Each blob in the binary images is called the region of interest where it most-likely contains text. Only the region of interests are fed into the consequent stages of the system.

Fine-detector is used to scan the region of interests from the previous stage. 50,000 background patches are collected from ICDAR2015 dataset and 50,000 foreground samples are collected from Wang's Synthetic dataset [30]. Edge detection and feature extraction

are applied to generate features from samples. Sobel edge detection is applied to the grey image, and spatial pooled to a grid of 3×3. The total number of features is $M \times 9$ where M is a number of templates in the codebook. The final features are used to train another confidence-weighted AdaBoost for the fine detector with 1000 iteration (i.e., 1000 decision stumps as a weak learner). The classifier performance and learning curve are shown in Figure 2.4.4. Each region is scanned with step-size 1 and different orientation and aspect ratio. The same edge detection and local feature extraction of the coarse detector are used to classify each 30×30 patch at the regions of interests. The center point of each patch, which classified as the text is used as a seed point for the consequent stage. Aspect ratios is considered from 0.6 to 1.4, using step size 0.2. Small rotations are also considered from -6 to 6, using a step size of 2. The final output of the fine-detector is thresholded to maximise f-measure. Surviving pixels provide seeds for the next region growing step[26]. Each seed represents the centre of a patch that is classified as text.

3.5 Region Grow

A flood filling algorithm is used to grow regions to form CCs. The algorithm uses the seeds from fine-detector as a starting point and grows region till the grown region hits an edge point or a large shift in color. see Section 2.4.4.

Zhu uses the seed point from the fine-detector as foreground and one pixel around the region as background. Then at each iteration labels the one pixel that uses the least cost to foreground/background. Even though we may have all the adjacent pixels with least

cost, but the algorithm takes one at a time to label. See Section 2.4.4 for the cost function. the process continues until no pixel left in the region unlabeled. Clearly, this process is the slowest process, because it is possible for the algorithm to iterate as many as pixel exists in the region. For example, if we have a region with a seed point in the middle. which lies on an edge point; The background points will be labeled before the foreground pixels because they cost less than foreground pixels.

We managed to accelerate the region grown that implement by Zhu et al. [26]. We change the algorithm to grow all the pixels that cost less than the threshold using breadth-first search. We use the same threshold used by Zhu, which is the 80th percentile of the edge values in the image. Clearly, we label all the dilatable pixel that is below the threshold in an iteration. Furthermore, we showed that we can achieve the same Connected Components CCs by ignoring the background pixels that can be grown. Thus we reduce the number of iterations from thousands to tens and the time consumption from tens of minutes to seconds.

3.6 Evaluation

The evaluation is done using ICDAR off-line tool[14]. created for evaluation word detection in images. It accepts an individual file for each image, it should contain the bounding box of words or characters in the image (i.e., one bounding-box per line). It computes overlap between detected words/characters bounding box and ground truth. See Section 2.3

The evaluation metrics based on recall and precision of bounding box in each image and average of all images. The bounding box includes x and y coordinates, with width and

height of the word in an image.

3.7 Summary

In this chapter, we talked about the improvement that we did to an existing text detection system. It includes, reducing sliding window size, removing pooling region overlap, and improving time consumption of the regions growing algorithm.

We also talked about the proposed sampling techniques, that has been used to collect samples for the convolutional K-means. The proposed methods include selecting the most common shape within a local patch (i.e., high visual similarity), or selecting the unique shape or both instead of random sampling, or exploiting both similarity and dissimilarity as the probability distribution of each sample. Angular distance is used to compute similarity/dissimilarity score between patches.

The evaluation methods are also described. The evaluation includes both bounding box and pixel-level. We use ICDAR off-line tool to evaluate our results. It computes overlap between detected words/characters and the ground truth. The overlap ratio is used to compute evaluation metrics, which include precision, recall, and f-measure.

Chapter 4

Results and Discussion

We have applied a series of experiments with different sampling techniques to create a codebook for text detection from the natural scene. The dataset used is ICDAR 2015 as described in section 2.1. First, Zhu et al's system re-implemented and result replicated, then different sampling technique is used to create codebook and use it for text detection. Detection results are compared to the replicated results and documented results of Zhu et al.[26]. The evaluation metrics are computed in three levels, word bounding box, character bounding box, and pixel-level as described in section 2.3.

The experiments look at the samples that are fed to the convolutional K-means to create a dictionary; Then it used as a feature map for text detection from images. Our experiment designed to show how the CK-means sampling affects the detection performance. Our sampling algorithm looks at local visual similarity/dissimilarity (i.e. in-patch) and also global visual similarity/dissimilarity. At the local level, we aimed to show that how selecting similar, dissimilar or both within a patch can affect detection. Then we use global visual similarity and dissimilarity to select the final set of training samples for the convolutional K-means.

The produced codebook is used to extract a feature from a fixed-size sliding window

from images. Then a confidence weighted AdaBoost is used to classify the windows. The value of the window will be replaced with the output of the classifier, which shows how likely a region is a text in the image.

4.1 Implementation

We started by re-implementing Zhu et al's system and replication their results up to the fine-detector. See Figure 2.8 for full system structure. Since we are dealing with images, the processing time was our main concern. To make the process faster, we needed a GPU implementation to extract feature and a classifier that can be trained fast. We started by implementing sample extraction for the convolutional K-means, and feature extraction for the coarse detector using GPU. The convolution and pooling process on CPU was taking a week to finish all the images. Once we produced the GPU implementation, we reduce feature extraction time for one image from tens of minutes to seconds. The GPU-implementation helped the system to avoid all the nested loops that were needed for the process.
The next step was to produce a classifier, that can be trained fast for the coarse, fine detector as well as verification stage. The classifiers were not available in Zhu's system. The classifier is confidence-weighted AdaBoost, which produces real value for classification instead of a hard thresholded value. See Section 2.5. Since we had a high dimension data with roughly 80,000 samples, using classifiers from libraries such as Sklearn wasn't a good idea. It took days to train one classifier on such libraries. We managed to implement the Confidence-weighted AdaBoost using Python and Cython for the weak learner (i.e., decision stump). Exploiting Cython helped decrease the training time from days to hours.

Coarse-detector results are replicated using re-implemented version of feature extrac-

tion(i.e., patch size of 30×30, with GPU implementation). see Section 3.4.

4.1.1 Experiments

We trained two classifiers, one with local features and one with contextual features. Both classifiers were using a codebook that was provided by Zhu. We run both classifiers for 1000 iteration. Then we produced the coarse detector heat-maps for all images using both classifiers separately. The produced heat-map of the coarse detector is binarized using a hard threshold; by selecting the best threshold that produces highest f-measure.

Replicated results are evaluated based on the binarized heat-map. Each blob in the image is considered as a word, its bounding box is taken to account for the evaluation. Since Zhu et al.[26] provided their final binarized coarse detector heat-map, we also evaluated their heat-map and compared to our replicated results. The results are shown in Table 4.1.

We produce Connected Components (CCs) by growing region from each provided seed from preceding steps. As described in Section 2.4.4, the algorithm applies a flood filling technique starting from the seeds and considers edge intensity and colour difference to compute cost function. see Section 2.4.4.

Each grown CC need to go through a validation step before we consider it as a true detected CC. The Validation classifier uses individual characters from the pixel-level ground truth for training. Any Connected Component that has 90% pixel overlap with any character in the ground truth is considered as foreground. 5500 foreground and 7800 background samples are generated to train the validation classifier. Each CC is resized to 32×32 with keeping the original aspect ratio. Then applies edge detection and feature extraction using

the codebook with 1000 template. The feature extraction convolves the templates with the gradient of the resized CC and uses maximum polling into a grid of 3×3. A total number of features is $(M \times 9)$ where M is the number of templates in the codebook. The algorithm takes one seed at a time; it grows the region and verifies it. For faster execution, any seed that has been covered by previous grown CCs will be ignored; because they tend to generate same CC.

4.2 Visual Similarity Sampling

Up to this point, we replicated Zhu's result from beginning till the first verification. Then we can test our proposed sampling algorithm. We start by collecting samples for the convolutional K-means algorithm using visual similarity and/or dissimilarity.

In our experiments, We started from stage one to collect 32×32 patches and then extracting 8×8 from them. Stage one is identical to Zhu et al. [26]. Non-empty 32×32 patches are collected from 30 scales of the images and labeled as foreground/background. see Section 3.3.1. Then at stage two, our main contribution is applied. For each 32×32 patch visual similarity/dissimilarity is computed for all non-empty 8×8 and used as probability distribution. Totally, $400,000$ patches (8×8) are collected for each class. Then randomly $100,000$ sample is selected from each class. Thus, $200,000$ samples with equal number of of foreground/background are fed to the CK-mean to create a codebook. Then it is used as feature map for the coarse detector. The coarse detector results are evaluated to show the

	Pixel-level						Char-level		
	Coarse			Fine & Verification					
methods	recall	precision	f-measure	recall	precision	f-measure	recall	precision	f-measure
baseline	82.21	10.7	18.14	79.58	74.63	77.02	76.07	81.8	78.83
replicated-baseline	83.5	10.83	19.17	67.71	86.78	76.06	51.92	81.99	63.57
baseline-empty (local)	84.4	12.2	21.31	70.60	88.08	78.37	63.03	83.58	71.86
local similarity	78.43	11.21	19.61	64.34	76.2	69.81	50.9	79.9	62.18
local dissimilarity	73.51	10.2	17.91	62.88	74.7	68.28	50.23	78.78	54.08
local both	85.58	13.23	22.91	71.9	88.15	79.2	63.14	83.4	71.87
simi_init	86.1	13.50	23.34	71.5	88.05	78.9	63.18	83.45	71.91
simi_diss_init	81.1	10.8	19.06	61.25	79.23	69.08	51.1	79.76	62.29

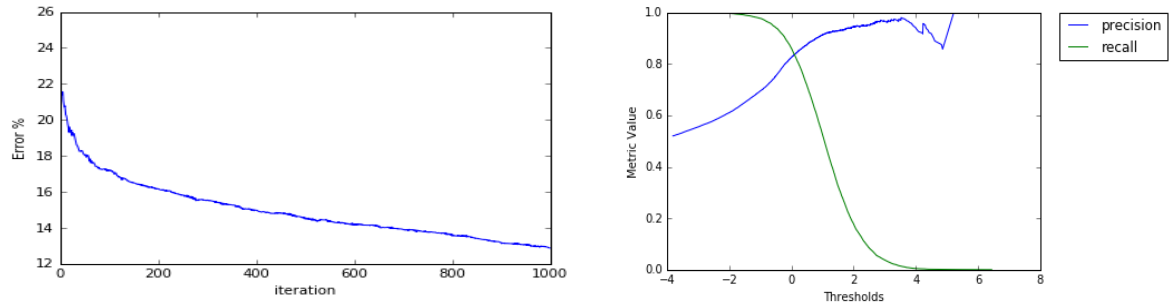
Table 4.1: Performance of our techniques vs previous methods on test data set

behavior of different sampling technique.

At stage two, we use two different approaches to select samples.

4.2.1 Stage One: Character Candidate Sub-Patch (Feature) Sampling

In this approach, we extract four 8×8 patches based on local visual similarity. We use similarity score as the probability distribution for the random selection. The result shows that if we select all four patches based on local similarity the detection metrics on coarse detector hot-map will decrease. See Figure 4.1 The local visual similarity, gives chance to most common shape in the character to be selected as representative of the character patch. We collect about 400,000 samples from each foreground and background, then for the final



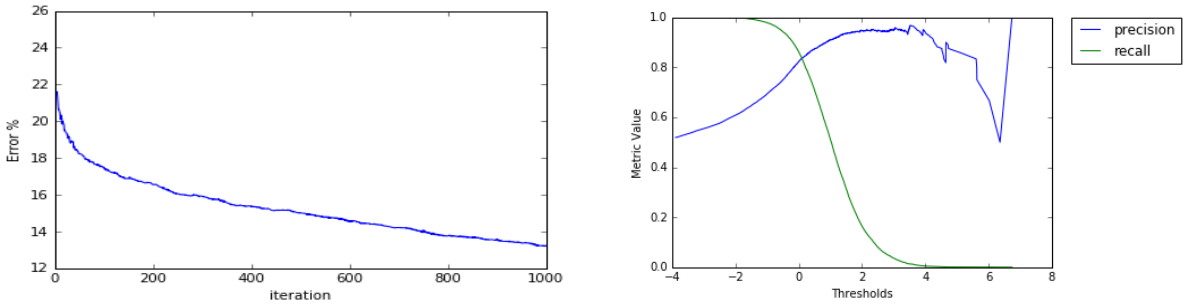
(a) classifier learning curve (b) classifier performance)

Figure 4.1: In-patch similarity and random in global

training data set we randomly select 100,000 from each class. However, it did not help improve the detection, but still, it contains some important shapes that are needed for the detection.

The Classifier performance on the test data set shows that there are many strong false positives in the data that cannot be classified correctly. See Figure 4.1. We think the only, cause for losing those samples were a lack of useful information which helps detect them because The local visual similarity only finds the common shapes but tells nothing about discriminating from others.

In another approach, we are using local-dissimilarity as the probability distribution. We increase the probability of those samples that are unique or rare within a character. Thus, we have less chance to select the representative (i.e., common) shapes in the character. The resulting codebook reduced the performance. See Figure 4.2. The local visual dissimilarity increases the chance of those samples that are very rare in the character, to be selected at random. This technique collects all the unique shapes within each patch. How ever it did

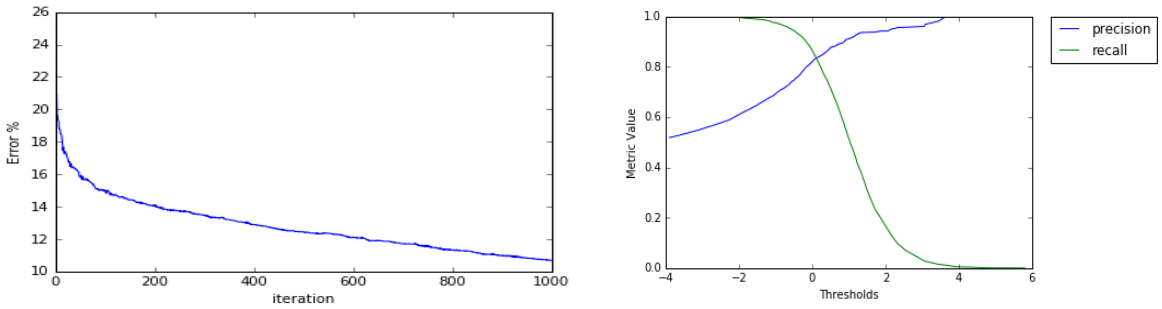


(a) classifier learning curve (b) Classifier performance)

Figure 4.2: In-patch dissimilarity and random in global

not help, but still we need those shape, because we may not find them in the first approach. For example, let's say we have a letter 'T', the highest dissimilarity goes to patches closer to the intersection of the horizontal and vertical lines. thus, the most common shapes, which are horizontal and vertical lines in the letter may be missed. Clearly, this technique gives very low chance to select vertical lines and horizontal lines from the previous examples. The results show that those shapes are not enough to discriminate foreground from background. see Table 4.1

Similar to the previous method, local dissimilarity is also miss classifies a lot of samples in the test dataset. Using local-dissimilarity misses more samples that local-similarity. We think it is because most of the information can be captured at local-similarity but we miss a lot of information that can not be captured by local-dissimilarity. This method tells if a character has unique shape but it ignores the most important information, which is the representative shape of the character.



(a) coarse classifier performance (b) testing performance)

Figure 4.3: inpatch similarity and dissimilarity and random in global

Finally, we managed to select two samples based on local-similarity and two samples based on local-dissimilarity. These methods collect 2 samples as representative of the character or the patch, and two more patches as discriminative or unique shapes within the character. The results show that it helps improve recall,precision and f-measure by 1%. Since we use both similarity and dissimilarity, we give chance to the very common shapes and very rare shapes to be selected as representative of the character patch. We think the improvement comes from the combination of the representative and the discriminative samples.

After we decided to select both visual similar and dissimilar samples as characteristics of the patch, we see the performance of the classifier increased. Clearly, the combination of those samples helped remove most of the false positive samples in the test dataset. see Figure 4.3

We see the error rate goes down lower than other two methods by the end of the learning. Also we see precision and recall curve acts like it suppose to be, we don't see any strong false positive where recall close to zero. Figure 4.4 shows some replicated results of

using VSS.

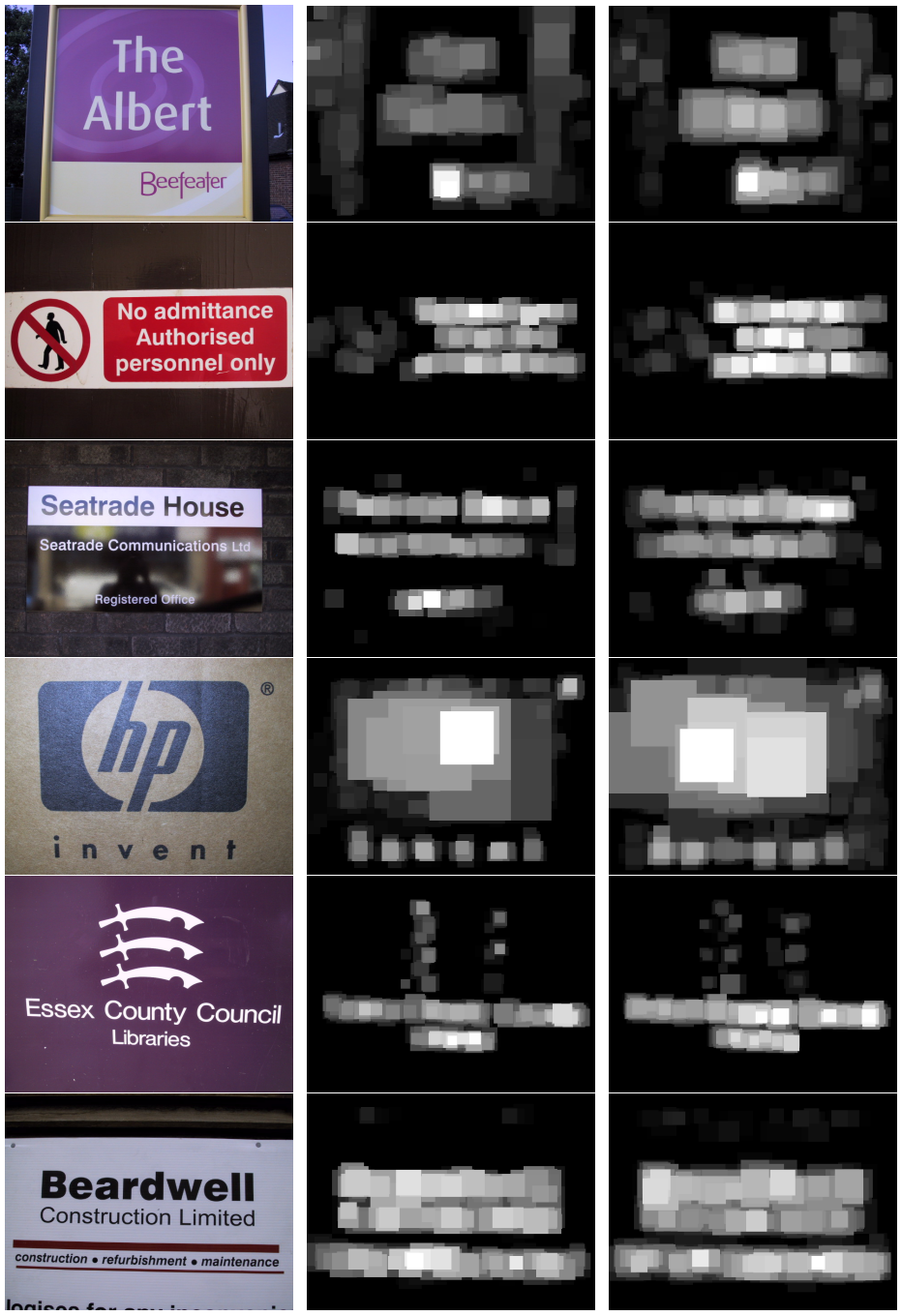
4.2.2 Stage Three: Seeding Initial k-means Cluster Centers

At this stage, we compute within-class and between-class similarity for samples of both classes, then we compute between-class dissimilarity. We aimed to collect samples that are very similar (i.e., common in a class) and very dissimilar between classes. See Section 3.3.3. The computed scores are combined using harmonic mean and used as probability distribution to select samples randomly for initialization of ck-means.

4.3 Initialization

The proposed method is exploited to initialize the convolutional -K-means algorithm. We use computed visual scores from global stage as probability distribution to initialize the convolutional K-means. We selected 1000 samples based on the distribution randomly that are close to decision boundary as initial cluster centres. A similar technique is used by Zhu et al. [26]. They managed to improve detection by exploiting SVM to select samples closer to the decision boundary as initial cluster centres for the convolutional K-means. We combine within-class similarity matrices and between-class similarity to select 1000 samples that are close to the decision boundary while preferring representative samples. We managed to increase the detection performance by initializing CK-means using samples closer to the decision boundary. It tends to provide samples that are close to samples of other class and able to discriminate.

We also tried to initialize the algorithm using harmonic mean of within-class similarity and between class dissimilarity. This approach uses samples that are close to the centre of



(a) Original (b) Local (c) Contextual

Figure 4.4: Difference between replicated hot-maps using (a) local information, (b) contextual.

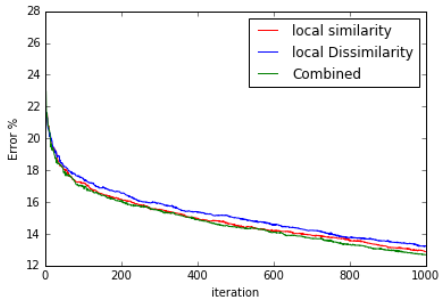


Figure 4.5: Learning curve for three different sampling approach at local stage

	AUC
Coates et al. [7]	62%
Zhu et al. [26]	71.2%
local_both	76.3%

Table 4.2: Area Under The Curve AUC for previous methods and our best achieved AUC when we use both similarity and dissimilarity at local level for patch collection.

the within the class distribution and far from decision boundary. However, it did not help improve detection, but it was expected. The approach collects representative samples from each class without considering their ability to discriminate from other class. The results in Table 4.1 shows that both recall and precision reduces.

4.3.1 Coarse Detector

The produced codebooks are used to extract features from 30×30 patches from images. Then a confidence -weighted AdaBoost is trained to detect text in the patches. The classifier is used to classify every 30×30 patch in the images. The patches have 15-pixel overlap with the surrounding patches. See Section 3.4

Area Under Curve AUC is used to evaluate the classifier performance. Zhu et al. [26]

and Coates et al. [7] use AUC for evaluation at this stage. Thus, we use the same algorithm to compute AUC and compare our results with theirs. To create a fair comparison we train regular AdaBoost classifier and compute its AUC for the comparison. see Section 2.3. 50,000 individual patches are selected randomly from training images. 80% of the data is used for training while 10% is used for testing.

Zhu et al. [26] achieved 72% of AUC, while Coates et al. [7] achieved 62. Our proposed method increased their result to 76.3. Removing the overlapped pooling regions in the coarse detector and the proposed sampling method helped to increase AUC. The AUC increased to 73.7 after reducing the patch size into 30×30 and removing the overlap between pooled regions. see Section 3.4 Additionally, the sampling technique with the initialization based on visual similarity increased the AUC even more to 76.3. see Table 4.2.

The confidence-weighted AdaBoost is used to produce the heat-map for the detection. A hard threshold is applied to the confidence of the classifier to create a binary image. See Section 3.4. The binairzed image is taken to account for evaluation.

The results show that the recall and precision by reducing patch size into 30×30 and removing overlap between pooling regions. see Section 3.4. Then using VSS increase the results even more. see Section 4.2.1 for details about different sampling approaches.

The recall is high enough for the coarse-detector in all experiments because even if we don't cover full characters in this region, we still have a chance to generate full character body after fine-detector and growing region stage. We also need to keep recall high and However, precision is very low, but it is expected. Since we compare coarse detector binarized image, we cover a lot of non-text pixels within the blob in the binary image.

In order to increase precision and get more accurate character detection, Fine-detector, region growing and verification step are applied. see Section 2.4.4, and Section 2.4.5.

4.4 Fine-Detector and Verification

All the proposed methods are tested after applying the fine-detector and verification step. see Section 2.4.4. The fine-detector applied on a region of interests of the coarse-detector in multiple scales to collect seeds (i.e., the center point of the true classified patches). The seeds from all scales combine into one image with an original size of the image. From each seed, a flood filling algorithm is applied in order to generate the whole full of the characters. All the generated CCs are fed into a verification classifier in order to reject the false positive CCs. The verification classifier is another confidence-weighted AdaBoost, which uses only visual features that are extracted using the generated codebook. See Section 2.4.6.

Any CC that passes through the verification classifier, is considered as a true character in the image. The resulting image is a binary image with black background and white pixels which represent the true characters in the image.

The output after verification step can be evaluated on two levels. First pixel-level, which computes the evaluation metrics based on true detected pixels against the ground truth. Second, uses the bounding box of each detected CC and considered as a character. Their bounding box is compared with character bounding box in the ground truth to compute evaluation metrics. See Table 4.1 for results.

The results show that the precision goes up while we lose recall. The recall reduces

because there are many characters that are very small and the region growing algorithm can not detect the whole body of the character. Thus the verification classifier rejects them. Some of the characters contain multiple colours or lightings makes the flood filling algorithm generate partial characters and then gets rejected by the verification. While some other characters are very thin, and their edge touches so there is no room for the flood filling algorithm to grow through the body of the character. There are even chance to have error in the implementation.

We are loosing some of the recall at pixel, because the algorithm fails to detect dots of the text. All of the dots in the images are rejected by the verifications classifier. If we force the verification to keep all those dots, then we keep the detected regions in side characters such as, inside letter 'O','B', 'R' ,...etc. Keeping those false positive CCs, will reduce precision at pixel level, but may help to keep recall and precision higher at character level.

Additionally, The resulting images with the CCs are evaluated based on the bounding box of CCs. We compare the bounding box of each detected CC in the Image with the character bounding box in the ground truth. The evaluation metrics are computed based on the overlap ratio between the bounding box. The same overlap rule as word bounding box is applied here. Characters with less than 50% overlap with the ground truth are discarded, otherwise, the overlap ratio is considered as detection ratio. The results of all experiments are shown in Table 4.1.

Figure 4.6 shows the precision and recall in which we set the hard threshold.

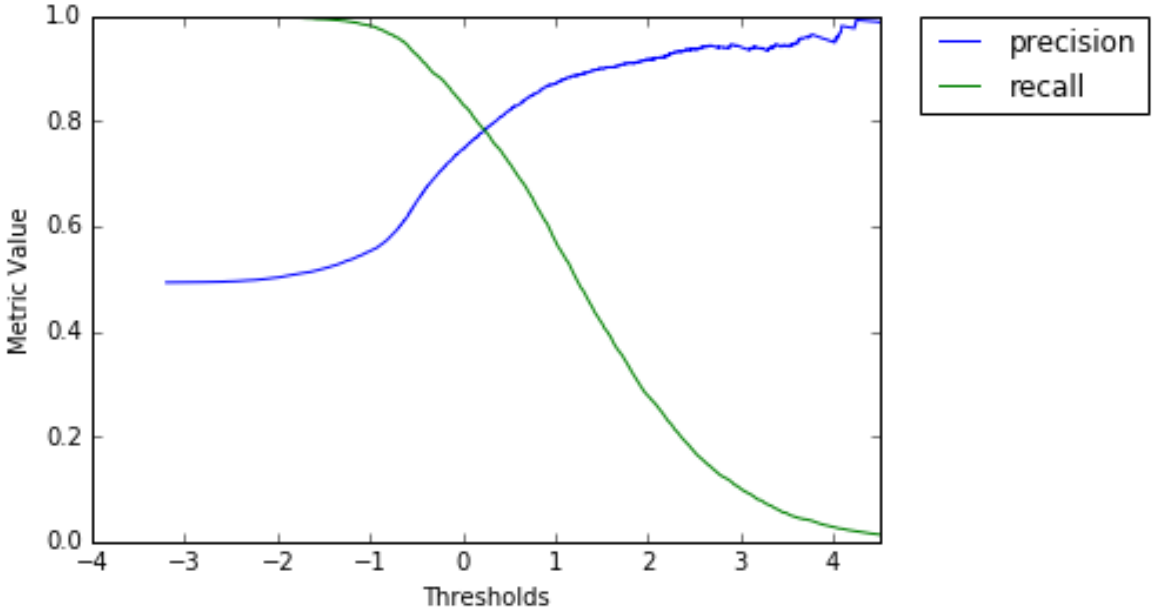


Figure 4.6

4.5 Summary of the Experiments

The experiments show that detection performance improves by sampling the convolutional K-means based on the visual similarity. The sampling uses visual similarity and dissimilarity score as the probability distribution for the random selection at the local level. The proposed methods show that samples with high visual similarity can represent its cluster and samples with high visual dissimilarity contains useful information along with visual similarity to discriminate foreground regions from background regions.

It appears that the detection increased by collecting non-empty samples for convolutional K-means training. Rejecting empty samples tends to force the algorithm to provide more information and increase detection performance.

The experiments also show that different initialization of the convolutional K-means affects the detection performance. We showed initializing the algorithm from the samples that are close to the decision boundary and high within-class similarity increases the results. The results show that the samples with high between-class similarity tend to be more discriminative than samples close to the centre of within-class distribution.

However, the results after fine-detector and region growing didn't increase, but we think that the region growing and verification has big share in results at this stage.

Chapter 5

Conclusion and Future Work

Our results suggests a text detection system for the ICDAR2015 dataset can be improved using visual similarity for sample collection. Those samples are fed to a convolutional K-mean algorithm to create a convolutional codebook, which is later used as feature map for text detection. The visual similarity score represents how common is a shape within a patch; while visual dissimilarity score represents how unique is a shape. Both scores (i.e., visual similarity and dissimilarity) are exploited in both local and global stages. The local is mostly to collect samples for the convolutional K-means, while global is to select 1000 samples as initial cluster centres.

At the local stage, neither most similar samples nor most dissimilar samples helped improve the detection. Thus, we select an equal number of samples from both similar and dissimilar samples and improved the performance. Furthermore, this technique supports both samples that are very common in a character or very rare. In which, we improved the detection performance by feeding those samples into the convolutional K-means.

Additionally, we exploit the visual similarity to initialize the convolutional K-means in order to produce more discriminative codebook. The results showed that initializing CK-means from samples closer to the decision boundary between text patches and non-text patches produces more discriminative codebook. These samples have high within-class

similarity and high between-class similarity. These two scores are combined using harmonic mean. The experiments showed that harmonic mean of within class similarity and between class similarity gives a higher probability to the discriminative samples.

5.0.1 Contributions and Future Works

Zhu et al.[26]'s system is re-implemented ans most of the results are replicated. We also proposed Visual Similarity Sampling to collect samples for the convolutional k-means. The re-implemented system source code is available for public.

Since we have re-implemented more than 80% of Zhu's system, the very first step would be reimplementing the word-graph and second verification in Zhu's system. Thus we would be able to have a full pip-line system and we can generate results and compare with other methods in ICDAR competition.

Also, the region growing stage may need more tuning and refactoring in order to make sure that we generate Zhu's grown regions. However, we grow region based on cost function mentioned in his thesis, but still there are many situations that the algorithm fails to generate full body of the character. For example, small and tide characters won't be able to grow through the body, because the edges are touching and cause the flood filling to stop.

To improve the detection even more, it really worth modify the learning process. Currently, in both local and contextual information the AdaBoost tries to find best threshold at each iteration. Thus, in to contextual situation, the AdaBoost may find best threshold in the 8-contextual blocks without even considering the actual block in the center. Modifying the algorithm should make the process find best split withing the local information and another threshold in the contextual blocks at each iteration.

We think that Visual Similarity Sampling can be used as general algorithm and worth testing it in other pattern recognition problems. Additionally, The algorithm can be modified into an iterative VSS. The iterative VSS can selected one sample at initial distribution, then remove the selected one and recompute the distribution for next iteration.

In term of implementation, Cython can be exploited in most of the part of the system. Cython accelerates most of the nested loops and operation within the system.

Bibliography

[1] Charu C Aggarwal and Chandan K Reddy. *Data clustering: algorithms and applications*. Chapman and Hall/CRC, 2013.

[2] N. Akhtar, F. Shafait, and A. Mian. Discriminative bayesian dictionary learning for classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(12):2374–2388, Dec 2016.

[3] D Arthur. Vassilvitskii (2007)k-means++: The advantages of careful seeding. In *Proceedings of the 18th Annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035.

[4] M Emre Celebi, Hassan A Kingravi, and Patricio A Vela. A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Systems with Applications*, 40(1):200–210, 2013.

[5] Datong Chen, Jean-Marc Odobez, and Herve Bourlard. Text detection and recognition in images and video frames. *Pattern recognition*, 37(3):595–608, 2004.

[6] Hojin Cho, Myungchul Sung, and Bongjin Jun. Canny text detector: Fast and robust scene text localization algorithm. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[7] Adam Coates, Blake Carpenter, Carl Case, Sanjeev Satheesh, Bipin Suresh, Tao Wang, David J Wu, and Andrew Y Ng. Text detection and character recognition in scene

images with unsupervised feature learning. In *2011 International Conference on Document Analysis and Recognition*, pages 440–445. IEEE, 2011.

[8] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223, 2011.

[9] Anders Lindbjerg Dahl and Rasmus Larsen. Learning dictionaries of discriminative image patches. In *22nd British Machine Vision Conference*, 2011.

[10] Kanny Davilla and Richard Zanibbi. Whiteboard video summarization via spatio-temporal conflict minimization. In *Proc. Int'l Conf. Document Analysis and Recognition (ICDAR), Kyoto, Japan*, 2017.

[11] Dian Gong, Xuemei Zhao, and Gérard Medioni. Robust multiple manifolds structure learning. *arXiv preprint arXiv:1206.4624*, 2012.

[12] Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

[13] Markus Juvonen. *Patch-based image representation and restoration*. PhD thesis, University of Helsinki, Finland, 2017.

[14] Dimosthenis Karatzas, Lluis Gomez-Bigorda, Anguelos Nicolaou, Suman Ghosh, Andrew Bagdanov, Masakazu Iwamura, Jiri Matas, Lukas Neumann, Vijay Ramaseshan Chandrasekhar, Shijian Lu, et al. Icdar 2015 competition on robust reading. In *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, pages 1156–1160. IEEE, 2015.

[15] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[16] Jiri Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761–767, 2004.

[17] Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, Jiri Matas, Frederik Schaffalitzky, Timor Kadir, and Luc Van Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1-2):43–72, 2005.

[18] Shaina Race. *Linear algebra*. Institute for Advance Analytics, 2015.

[19] Vikas Chandrakant Raykar. Spectral clustering and kernel principal component analysis are pursuing good projections. *Technical Report*, 2004.

[20] Robert E. Schapire and Yoav Freund. *Boosting: Foundations and Algorithms*. The MIT Press, 2012.

[21] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *International Conference on Artificial Neural Networks*, pages 583–588. Springer, 1997.

[22] Gregory Shakhnarovich. *Learning Task-Specific Similarity*. PhD thesis, Massachusetts Inistitute of Technology, September 2005.

[23] Saurabh Singh, Abhinav Gupta, and Alexei A Efros. Unsupervised discovery of mid-level discriminative patches. In *Computer Vision–ECCV 2012*, pages 73–86. Springer, 2012.

[24] Amit Singhal. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4):35–43, 2001.

[25] Lawrence Sirovich and Michael Kirby. Low-dimensional procedure for the characterization of human faces. *Josa a*, 4(3):519–524, 1987.

[26] Zhu Siyu. *A Text Detection System for Natural Scenes with Convolutional Feature Learning and Cascaded Classification*. PhD thesis, Rochester Institute of Technology, 2016.

[27] Matthew A Turk and Alex P Pentland. Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*, pages 586–591. IEEE, 1991.

[28] Kai Wang, Boris Babenko, and Serge Belongie. End-to-end scene text recognition. In *2011 International Conference on Computer Vision*, pages 1457–1464. IEEE, 2011.

[29] Tao Wang, David J Wu, Adam Coates, and Andrew Y Ng. End-to-end text recognition with convolutional neural networks. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 3304–3308. IEEE, 2012.

[30] Yong Wang, Yuan Jiang, Yi Wu, and Zhi-Hua Zhou. Multi-manifold clustering. In *Pacific Rim International Conference on Artificial Intelligence*, pages 280–291. Springer, 2010.

[31] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, April 2004.

[32] Hailiang Xu and Feng Su. Robust seed localization and growing with deep convolutional features for scene text detection. In *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, ICMR '15, pages 387–394, New York, NY, USA, 2015. ACM.

[33] Jieping Ye, Zheng Zhao, and Mingrui Wu. Discriminative k-means for clustering. In *Advances in neural information processing systems*, pages 1649–1656, 2008.

[34] Qixiang Ye and David Doermann. Text detection and recognition in imagery: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(7):1480–1500, 2015.

[35] Qixiang Ye and David Doermann. Text detection and recognition in imagery: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(7):1480–1500, 2015.

[36] Xuwang Yin, Xu-Cheng Yin, Hong-Wei Hao, and Khalid Iqbal. Effective text localization in natural scene images with mser, geometry-based grouping and adaboost. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 725–728. IEEE, 2012.

[37] O. Zayene, J. Hennebert, S. Masmoudi Touj, R. Ingold, and N. Essoukri Ben Amara. A dataset for arabic text detection, tracking and recognition in news videos- activ. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 996–1000, Aug 2015.

[38] Oussama Zayene, Sameh Masmoudi Touj, Jean Hennebert, Rolf Ingold, and Najoua Essoukri Ben Amara. Data, protocol and algorithms for performance evaluation of text detection in arabic news video. In *Advanced Technologies for Signal and Image Processing (ATSIP), 2016 2nd International Conference on*, pages 258–263. IEEE, 2016.

[39] Siyu Zhu and Richard Zanibbi. A text detection system for natural scenes with convolutional feature learning and cascaded classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 625–632, 2016.