

Text Zone Classification using Unsupervised Feature Learning

Nibal Nayef and Jean-Marc Ogier

L3i Laboratory, Université de La Rochelle, France
 {nibal.nayef, jean-marc.ogier}@univ-lr.fr

Abstract—Text zone classification is a vital step in the digitization process, without which OCR systems perform poorly. Prior methods to document zone classification have relied on large sets of hand-crafted features for training zone classifiers. Such features are usually database-dependent, and their computation is time consuming. In this work we propose a novel method for text zone classification that relies on the approach of unsupervised feature learning. Within our method, feature vectors of document zones are automatically learned by patches extraction, encoding and pooling, where feature encoding is based on a codebook of visual words. The training phase of the text classifier takes into consideration the unbalance between text zones and non-text zones of all types. The proposed method has been tested on publicly available standard databases, and achieved competitive or better results compared to state-of-the-art methods. The results show that our approach matches well the task of text classification, and is robust to zone shapes, orientations and size.

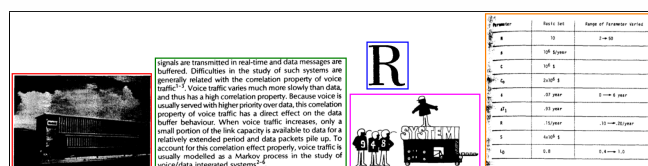
I. INTRODUCTION

Within the processing chain of document analysis systems, a document image is converted to a representation of structure and content, in order to allow later steps of automatic document understanding. One important subtask within this processing chain is zone classification. This task allows content type-specific processing, for example applying OCR on text zones, document classification based on logos, table reading, image processing for graphics etc.

A layout analysis step segments a document into a set of blocks (regions or zones), which are not necessarily rectangular. After that, it is necessary to have a zone classifier for identifying the types of the segmented zones as one of a set of predefined classes such as text, image, graph, halftone, line drawing, table, separator, noise, handwritten content etc. The misclassification of document zone types leads to high error rates in all subsequent processing in the document understanding chain. Moreover, classifying zone types is crucial to indexing and retrieval of large document databases.

In this work we focus on the identification of the content type of detected document zones by classifying them into text or non-text. Rather than investing a lot of effort in designing features that are tuned to specific zone types in specific databases, we present instead, an approach for automatically learning a powerful feature vector for the zone classification task. By unsupervised feature vector construction and a standard learning tool such as SVM classifiers, we are able to achieve competitive or better performance than state-of-the-art methods. The general approach of automatic feature learning has been successful in many computer vision tasks

[1]. We have adopted and specialized this approach to our zone classification problem to achieve high accuracy and robustness to zone shapes, orientations and size.



(a) Text and non-text zones from the UW-III dataset.



(b) Text and non-text zones from the PRImA dataset.

Fig. 1. Example segmented document zones from the datasets used in the evaluation of our system. Different zones are shown in different colored bounding boxes. The zones have variable sizes, orientations and shapes.

The main idea behind feature learning is to automatically find a set of features that are relevant to the task in hand. In contrast to all previous approaches in zone classification – which rely on features carefully designed based on prior knowledge of a document database –, our approach is able to find the relevant features in a completely unsupervised manner. Figure 1 shows example document zones that we consider as input in our text zone classification problem. The zones have large variability in size, shape and orientation. Text content for example may be at letter, word, line or paragraph level.

II. RELATED WORK

A review of prior work in document zone classification can be found in the survey of Okun et al. [2], and in the works of Wang et al. [3], [4]. According to the classification made by Wang et al. [4]; the two categories of zone classification methods either include segmentation [5] or not [3], [4], [6]. We review here the methods that focus on the already extracted zones, as our approach falls in this category. The approach followed in the vast majority of these methods is based on extracting a set of different types of features from zones, and then learning a model for classifying zones' types.

Wang et al. [3], [4] extracted a large set of zone characteristics including run-length features, spatial features such as the pixel distribution of zone foreground, and autocorrelation features. With those features, they trained an optimized decision tree classifier. Contextual constraints are incorporated in the classification of some zones using HMMs. Keyzers et al. [6] extended the work of Wang et al. [4] by presenting a detailed analysis of the used features and their power in zone classification. They used a comprehensive set of features most of which are the features used by Wang et al. [4]. They used a nearest neighbor classifier with introducing a new a speckle noise class in addition to the other typical zones types. Lin et al. [5] used different texture-based GLCM (Grey Level Co-occurrence Matrix) features, and their method included a segmentation step. They divided a document into blocks of graphics, text and space zones, and used K-means for clustering the blocks into zones. They then used pre-learned heuristic rules for zone classification.

As many features of different types had already been tested for the zone classification task, other researchers proposed improving the learning part of methods. Abd-Almageed et al. [7] proposed a novel hybrid learning method which combines the benefits of both one-against-all and one-against-one voting schemes. They used typical structural features such as run length, and texture features such as Local Binary Pattern (LBP) and autocorrelation features among others. They also used partial least squares on pairs of classes to compute discriminating pairwise features. Bouguelia et al. [8] proposed to use an incremental learning method where the classification relies on a reject utility in order to reject ambiguous zones or documents. Their trained model is updated incrementally each time a new document and its extracted zones are added and learned. They used run-lengths and connected components features. Their zone classes do not contain printed text, but rather: handwritten annotations, stamps, signatures etc.

Other researchers presented special purpose zone classification methods. Kumar et al. [9] developed a method for classifying handwritten versus machine printed text zones. They used shape-based triple-adjacent-segment (TAS) features. They constructed two shape codebooks from those features for each class. Each zone is represented by a normalized histogram of codewords as a feature vector, which in turn is used to train an SVM classifier. Their approach is robust to background noise, and their features are invariant to text transformation. As we discussed in the previous section, our method differs from all the methods reviewed above, as we follow an automatic feature learning approach.

III. THE FEATURE LEARNING APPROACH

Our approach for text classification follows the global generic framework of feature learning employed in several computer vision and deep learning methods [1]. Figure 2 shows a block diagram of the proposed approach. It consists of three main parts: feature representation, feature vector construction at image-level and finally training and classification. The first part, feature representation, discovers features from unlabeled data by learning to map input data to features. The second part extracts data features and constructs fixed-size feature vectors based on feature pooling. Those two parts represent the complete process of feature learning, and they are carried

out completely unsupervised. The result of those two parts is a set of feature vectors that correspond to document zones in a database. Given a set of labeled training zones, the third part in our framework, trains a classifier to predict labels (i.e. zone types) of test zones. The details of each of the parts of our approach are described in the following subsections.

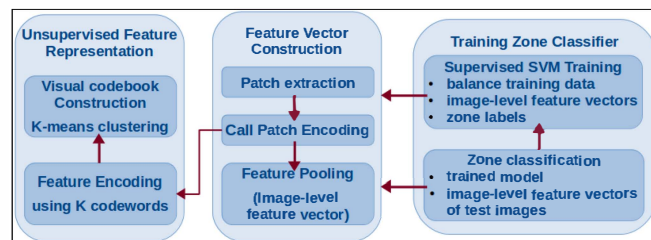


Fig. 2. Block diagram of the proposed zone classification system based on unsupervised feature learning.

A. Learning Feature Representation

In a high-level view, this part of the framework can be represented as a block that takes a dataset of images S as input, and outputs a function $f: \mathbb{R}^d \rightarrow \mathbb{R}^K$ that maps a dataset input vector x^i to a new vector of K features, where K is an input parameter. In order to learn such a function f , we perform the following steps:

- Randomly sample M patches as input vectors x^i from a dataset of unlabeled training images D
- Normalize and whiten the patches
- Apply an unsupervised learning algorithm (ex. k-means clustering) on the patches to learn the mapping function f

The dataset S in our case is a set of training images where each image is a zone (region) in a document image. We assume that the regions of the document images in the database have been identified by a layout analysis algorithm. In the first step, we randomly sample M patches from the unlabeled zone images: $S = \{x^1, \dots, x^m\}$ where $x^i \in \mathbb{R}^d$. A patch is a vector of d pixel intensity values.

Each patch x^i is then normalized by subtracting the mean and dividing by the standard deviation of its elements. This step is an essential factor in feature learning of visual data, as it results in the normalization of local brightness and contrast. After that, the entire dataset S is whitened to achieve decorrelation. Whitening transforms input data to have a diagonal covariance matrix. This step is crucial to clustering algorithms since they are blind to correlations in the data.

After that, k-means clustering algorithm is used an unsupervised learning algorithm to learn f . The normalized input patches are mapped into K clusters, and hence, K centroids c^k are learned. Then the feature mapping f of an input data vector x is computed as follows:

$$f_k(x) = \max\{0, \mu(z) - z_k\} \quad (1)$$

where $f_k(x)$ denotes the k^{th} feature in the output vector, $z_k = \|x - c^k\|_2$, and $\mu(z)$ is the mean of z . This step performs a non-linear mapping from an input data patch to a feature

vector representation of that patch. This mapping is also called encoding. We can view this process as feature encoding, where a visual codebook (the K centroids c^k) is learned by k-means clustering. Then each data patch is encoded in terms of the codebook c^k visual codewords. There are many choices for encoding function $f_k(x)$. For example, there is the typically used *1-of-K hard assignment*, where an input patch is assigned only to one of the K codewords. Our choice belongs to *soft encoding* functions, because it has shown better performance in previous computer vision works [1].

B. Feature Vector Construction at Image Level

Now we have a feature representation function that can represent image patches into features. However, we do not have a feature vector of the entire image (the image of a document zone). Moreover, feature vectors at image level have to be of fixed size to be eligible as input for later steps of training and classification. In order to construct such feature vectors, we perform the following three steps:

- Extract N random patches from an input image X , $X = \{x^1, \dots, x^N\}$
- Encode each of the extracted patches using Eq.(1) to get the encoded image representation $E_{N \times K}$
- Pool the encoded matrix of features E into a feature vector $F_j = \max\{E_{j1}, E_{j2}, \dots, E_{jN}\}$, where $j \in \{1, 2, \dots, K\}$

In the first step we randomly extract N equally sized patches from an image. Note that the sizes of input images are different, hence N is not the same for all images. Note also that N is not necessarily equal to M mentioned in the previous subsection. M should be chosen such that we have enough representative patches for constructing the visual codebook, whereas N is the number of patches that are used to construct a feature vector of an image. After patch extraction, we have a matrix $X_{N \times d}$ of patches for each input image. In the second step, we encode each patch $x^i \in \mathbb{R}^d$ according to Eq.(1) to get a patch representation $f^i \in \mathbb{R}^K$. For the whole image X , the encoding process results in a matrix $E_{N \times K}$ where each row is an encoded patch. Finally we perform maximum pooling along the columns of matrix E to get a feature vector $f_{1 \times K}$ for each image. This vector represents the features of an image, and it has the same fixed size $1 \times K$ for all the images.

C. Training of a Zone Classifier

The above two subsections have shown how to automatically compute a feature vector of a certain size from any image. This makes it easy to use any available supervised learning and classification method for solving the task for zone classification. In this work, we are concerned with the classification of text zones versus all other types of zones, because finding and classifying text zones is a very important step for OCR systems and other document analysis processes. Having a set of zone images of a dataset (as shown in Figure 1), we compute a feature vector for each zone image. From the ground-truth of a dataset, we have the labels of each zone type (each document region type). The problem is represented as a binary classification problem with two labels: “text” to represent documents text zones, and “non-text” to represent

all other zone types such as: image, table, drawing, graph, equation, separator etc.

An SVM classifier is trained on the zones’ feature vectors with their corresponding labels. Note that in most databases, the number of text zones is much larger than the number of non-text zones, hence the training data is unbalanced. To that end, we increase the number of training samples of non-text zones as follows. We perform the step of random patch sampling two to three times for each zone image of type non-text, to get different sets of patches, and hence different feature vectors of a zone image. For a zone image of size $n \times n$ pixels, and a patch size of $w \times w$, there are $(n - w + 1) * (n - w + 1)$ overlapping patches with a sliding step of one pixel. For a small image of 512×512 pixels, there are 256036 patches. This is a huge number of patches that is much more than needed for creating a representative rich feature vector. In many feature learning research works as well as in this work, N is set to a number between 1000 and 10000 patches that are randomly extracted from an image to construct the image-level feature vector as described in subsection III-B. If we extract two different sets each of size 2000 of random patches, we can get two different feature vectors of the same training sample. We additionally set a larger learning weight in the SVM for the class “non-text”. In the classification phase, we construct a feature vector for each test zone image, and use this vector as an input to the trained classifier, in order to predict the label of a zone image as text or non-text.

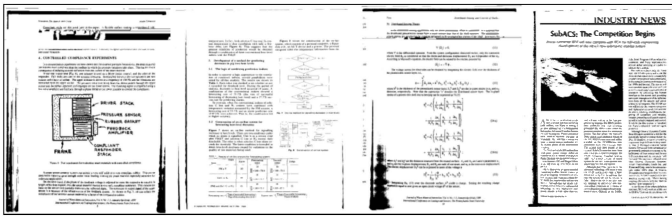
IV. EXPERIMENTAL EVALUATION

We have implemented our feature learning approach in a zone classification system, and evaluated this system on standard databases in document layout analysis. In the following subsections we detail the databases, the settings of our experiments and analyze the results.

A. Databases

Two datasets have been used in our experiments. The first is the University of Washington III (UW-III) dataset [10]. The database contains 1600 English binary scanned document images along with their zone-level ground truth. The ground truth is represented by bounding boxes of 24254 page zones (page regions). Those bounding boxes are labelled as either text or non-text. A non-text region could be an equation, line drawing, halftone, table, image, chemical formula etc. This ground truth makes the dataset very suitable for layout analysis, page segmentation and segmented zone classification purposes. The documents in the UW-III dataset have different degradation levels due to photocopy process, so, the same document could appear multiple times with different degradation. This dataset is rather old.

The second dataset used in our evaluation is the ICDAR-2009 competition dataset called “PRImA” [11]. The main purpose of the PRImA dataset is layout analysis tasks like page segmentation and zone classification. The set contains 55 contemporary colored documents. The ground truth contain precisely located zones using polygons rather than bounding boxes, as the zones might not always be rectangular. The documents’ zones are labelled as text, separator, graph, image or line art. We relabel those zones as text or non-text, due to the



(a) Documents from UW-III dataset.

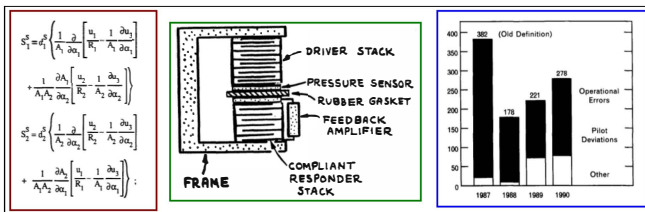


(b) Documents from PRImA dataset.

Fig. 3. Example document images from the datasets used in the evaluation of our system.

few available examples of each type of non-text zones, and also because we are mainly concerned by the ability to distinguish text from non-text regardless of the type of non-text. Example documents from both datasets are shown in Figure 3.

Note that in both datasets, non-text zones could contain text, for example, like annotations inside images, or text in tables. This makes it challenging to classify non-text regions, specially for bottom-up layout analysis methods. Figure 4 shows examples of non-text regions that contain text.



(a) UW-III dataset document zones.



(b) PRImA dataset document zones.

Fig. 4. Challenging document zones: Text appears in non-text zones. However, they are correctly classified by our system. (Different zones are shown in different colored bounding boxes.)

B. Experimental Settings

We use the same parameter settings for testing both databases. The train/test data are 60% and 40% respectively. This is a common train/test ratio as used with typical learning-based approaches that do not use feature learning to construct feature vectors. As the construction of the codebook and

feature vectors is based on random extraction of patches, we have executed the system 200 runs per one parameters' setting. Unless otherwise stated, the results that we show are the average across the 200 runs.

Now we come to the parameters that are directly related to the feature learning algorithm itself. The number of patches that are randomly sampled from each image – denoted M or N in previous sections –, is set in each experiment to one element in the set: $\{1000, 1500, 2000, 2500, \dots, 5000\}$. Next, we have varied the number of clusters K – i.e the size of the image-level feature vector – to one element in the set: $\{32, 49, 64, 128, 256, 512, 1024\}$. Finally, we have varied the patch size to one element in the set: $\{5, 7, 8, 9\}$.

C. Results and Analysis


Table I shows the average accuracies of the proposed zone classification method on the complete datasets: UW-III and PRImA. The shown results are for the experimental setting that achieved best results: $K = 128$, patch size = 7, number of patches = 2000. Note that a small size of the visual codebook is enough to achieve highly accurate results. Small codebook means a similarly small-sized feature vector, which means very fast computation time per sample at test phase, and also less memory requirements for storing the trained model and the codebook. This is an advantage over what is commonly encountered in computer vision applications that use a somewhat similar feature learning approach. We argue that this indicates the suitability of feature learning approaches for text zone classification, as only a small number of visual codewords is needed to represent patches of text characters, whereas in natural images, a large number of visual words is required for representing images of some content type like flowers or animals. The same argument can be made for the number of patches extracted from each zone image.

TABLE I. ZONE CLASSIFICATION ACCURACIES OF THE PROPOSED METHOD. THE SHOWN ACCURACIES ARE AVERAGE FOR 200 RUNS.

| Dataset | Zone Type | # Zones | Accuracy |
|---------|-----------|---------|----------|
| UW-III | Text | 21705 | 98.9% |
| | Non-text | 2549 | 82.3% |
| | All zones | 24254 | 97.4% |
| PRImA | Text | 952 | 97.2% |
| | Non-text | 294 | 93.2% |
| | All zones | 1246 | 96.3% |

We can see from the results' table, that the non-text class is challenging to learn. This is due to two reasons: first, it contains multiple different sub-classes such as separators, line drawings and natural images which have very different visual appearances. The second reason is that some of the subclasses such as table and graph actually contain text, which might cause confusion with the text class. Figures 5 and 6 show examples of error cases by our system for both classes.

Now we turn to discussing the effect of varying the parameters settings explained in subsection IV-B. We have carried out experiments for all combinations of parameters' settings. It is worth mentioning though, that in previous research works in deep learning and computer vision, the effect of varying such parameters have been extensively studied for different problems and databases. Hence, we will just comment on



| Technique | Management overhead | Network overhead | Router overhead | DDoS handling | Postmortem capability | Traceback speed |
|---------------------|---------------------|------------------|-----------------|---------------|-----------------------|-----------------|
| Ingress filtering | Moderate | Low | Moderate | N/A | N/A | N/A |
| Input debugging | High | Low | High | Good | N/A | Slow |
| Controlled flooding | Low | High | Low | Poor | N/A | Slow |
| Logging | High | Low | High | Excellent | Excellent | Slow |
| ICMPTraceback | Low | Low | Low | Good | Excellent | Moderate |
| Packet marking | Low | Low | Low | Good | Excellent | Moderate |
| SIPT | Low | Very low | Low | Good | Excellent | Fast |

(a) Non-text classified as text. Note that all the zones contain text, specially the last one with a lot of text and few separators.



(b) Text classified as non-text. On the left, the zone is exactly like non-text zones in other documents. On the right, the misclassification happened due to the very large font used, which was not available in the training, the size of this zone is 984x1090 pixels.

Fig. 5. Misclassification cases by the proposed system in the PRImA dataset.

those parameters briefly. Our system is in fact robust to at least a range of parameter values: in all the experiments with different parameters' values combinations, the overall zone classification accuracy for PRImA has varied between 92.2% and 96.4%, with the lower values achieved for smaller numbers of patches and smaller sizes of codebook. However, starting from 2000 patches per image, and 128 for codebook size, the performance stabilizes with $\pm 1\%$. Similar behavior has been noticed for the UW-III dataset but with much less variance, where the overall zone classification accuracy has varied between 95.9% and 97.5%. The better results are naturally due to the availability of more training samples, and that the region shapes of the UW-III dataset are less complicated than those of the PRImA dataset.

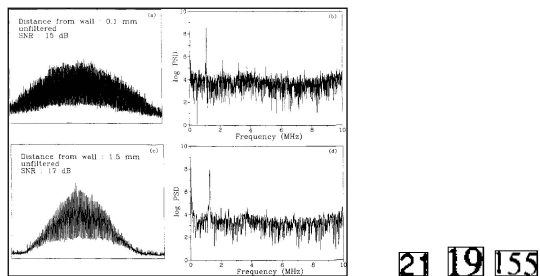


Fig. 6. Misclassification cases by the proposed system in the UW-III dataset. The first from left is a non-text zone classified as text. The next three are text zones classified as non-text. Due to their very small size, the extracted features are not really representative of any zone type.

Finally, we would like to compare our method to state-of-the-art zone classification methods that are based on large sets of hand designed features. Table II shows this comparison with the two best performing methods for the UW-III dataset. Note that an exact comparison among the methods is not possible, as different zone types (classes) are used. As for the PRImA

dataset, we are not aware of available state-of-the-art results.

TABLE II. COMPARISON OF OUR METHOD TO STATE-OF-THE-ART ZONE CLASSIFICATION METHODS ON THE UW-III DATASET.

| Method | # pages | # zones | accuracy | comments |
|---------------------|---------|---------|----------|-------------------------------|
| Wang et al. [4] | 1600 | 24177 | 98.5% | different zone types are used |
| This work | 1600 | 24254 | 97.4% | |
| Keyesers et al. [6] | 713 | 13811 | 98.5% | 11804 + |
| This work | 713 | 11804 | 99.8% | 2007 noise zones |

V. CONCLUSIONS AND FUTURE DIRECTIONS

This paper has presented a novel method for text zone classification based on automatic learning of features rather than hand-crafted features. The part of feature learning is completely unsupervised and without domain knowledge. We have shown how to adopt the generic approach of feature learning to the problem of document zone classification, and that our approach fits very well this problem in terms of high performance and robustness on different datasets.

For future work we would like to investigate the use of feature learning on other problems in text document analysis. Additionally, we will extend the zone classifier to multiple non-text subclasses. We expect this to be an easy problem as non-text subclasses have distinct visual properties, and the feature vector size can be increased to accommodate multi-class discrimination. Finally we would like to use more advanced algorithms for encoding patches, such as sparse coding.

REFERENCES

- [1] A. Coates, H. Lee, and A. Ng, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, vol. 15, 2011, pp. 215–223.
- [2] O. Okun, D. Doermann, and M. Pietikainen, "Page Segmentation and Zone Classification: The State of the Art," Tech. Rep. LAMP-TR-036,CAR-TR-927,CS-TR-4079, 1999.
- [3] Y. Wang, I. T. Phillips, and R. M. Haralick, "A study on the document zone content classification problem," in *Document Analysis Systems*, 2002, pp. 212–223.
- [4] —, "Document zone content classification and its performance evaluation," *Pattern Recognition*, vol. 39, no. 1, pp. 57–73, 2006.
- [5] M.-W. Lin, J.-R. Tapamo, and B. Ndovie, "A texture-based method for document segmentation and classification," *South African Computer Journal*, vol. 36, pp. 49–56, 2006.
- [6] D. Keyesers, F. Shafait, and T. M. Breuel, "Document image zone classification - a simple high-performance approach," in *VISAPP (2)*, 2007, pp. 44–51.
- [7] W. Abd-Almageed, M. Agrawal, W. Seo, and D. Doermann, "Document-zone classification using partial least squares and hybrid classifiers," in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, 2008, pp. 1–4.
- [8] M.-R. Bouguelia, Y. Belaid, and A. Belaid, "Document image and zone classification through incremental learning," in *Image Processing (ICIP), 2013 20th IEEE International Conference on*, 2013, pp. 4230–4234.
- [9] J. Kumar, R. Prasad, H. Cao, W. Abd-Almageed, D. S. Doermann, and P. Natarajan, "Shape codebook based handwritten and machine printed text zone extraction," in *DRR*, 2011, pp. 1–10.
- [10] I. Phillips, "Users reference manual. cd-rom, uw-iii document image database-iii," 1995.
- [11] A. Antonacopoulos, S. Platschacher, D. Bridson, and C. Papadopoulos, "Icdar 2009 page segmentation competition," in *ICDAR*, 2009, pp. 1370–1374.