# Label Detection and Recognition for USPTO Images using Convolutional K-means Feature Quantization and Ada-Boost

Siyu Zhu
Center for Imaging Science
Rochester Institute of Technology, USA
sxz8564@rit.edu

Richard Zanibbi
Department of Computer Science
Rochester Institute of Technology, USA
Email: rlaz@cs.rit.edu

***Abstract*—We utilize Coates' unsupervised feature learning method and Ada-Boost to detect and recognize part label regions in patent drawings. Image patches are harvested from training data, and features are learned from patterns in image patches. Angle distances between samples and feature banks are computed, and used in Ada-Boost classifier. We extract image patches with different sizes to counter the scale problem. An ensemble Ada-Boost is used to classify pixels as text or background. Meta-Boost is introduced to improve performance. The pixel level detections are then grouped into 'Connected Components'. Several denoise methods are applied, followed by 'Tesseract' OCR. Our system achieves competitive performance without using strong prior knowledge.**

## I. Introduction

The text detection methods have draw a lot of attention in recent years. Locating text in natural scene images or patent drawings poses a much more complicated problem than for text lines in conventional documents. Classifiers which can discriminate text from background and noise are required. Considering texts in natural scenes or drawings may appear in different fonts, sizes, orientations and illuminations, text detection is a challenging problem.

To address this problem, several papers have been published. Epshtein etc. [1] used Stroke Width Transform to detect text in natural scene images. Lu etc. [2] used a linear object erasing method to detect texts in machine drawings. Zhang etc. [3] used Edge Gradients to extract character candidates and grouped candidates with Graph Spectrum.

Instead of designing features and tuning parameters by hand, an adaptive learning method might generate desirable features automatically. They are independent from prior knowledge of the target shape or color information. Despite the dataset variations, the classifier generated are adapted to the data.

Kavukcuoglu etc. [4] used the convolutional learning method to learn sparse codings from the data. Meanwhile, they also learned efficient encoders from the data to reduce the computation time. Coates etc. [5] tested several unsupervised learning methods, such as sparse auto-encoder, sparse restricted Boltzmann machine, K-means and Gaussian Mixtures. In another paper, Coates etc. [6] made the use of convolutional K-means to quantize the feature space into several typical patterns, and utilized the pattern bank to classify the target images. They claimed that convolutional K-means yields results comparable to other methods while being much simpler and faster to compute.

We propose an end-to-end system for part label detection and recognition for patent drawing images. We implement Coates's unsupervised learning method, and use Ada-Boost to extract labels from patent drawings. 'Tesseract' OCR [7] is used to recognize text information in the detected Region Of Interest (ROI).

## II. USPTO Algorithm Challenge

Text detection in patent drawings has a lot of practical interest but is difficult [8]. The algorithm is useful in patent image digitization and automation, so that we can for example automatically display the text associated with part label overtop of a diagram images as it is being read online. There were two competitions held to encourage novel and efficient text detection and recognition algorithms for patent, in 'USPTO Algorithm Challenge' [1]. There are totally 306 patent drawing images in competition 1, divided into 3 subsets. They contains 178, 35, 93 images as training, provisional test and system test sets, respectively. There are 139 additional images in competition 2. We use data from competition 1 only in our experiment.

The top 1 player team in challenge 1 used a Connected Components (CC) based algorithm for text detection. they examined the shapes of CCs. If a CC's size, aspect ratio, boundary length and etc. is out of a prefixed range, it will be discarded. Template matching method was used to recognize characters. A language model is used to filter recognized symbols, word length is limited to 4 characters, and only a, c, f, g are allowable English characters. Information from '.html' files were used to support recognition when confidence was low.

We propose a method starting from pixel level, based on the image patch patterns around label text and figure drawings. The first stage detection will discard undesirable pixels and select text candidates pixels. The second stage, it will group the candidate pixels and find text in ROI.

[1] http://community.topcoder.com/longcontest/?module=ViewProblemStatement&rd=15027&pm=11645

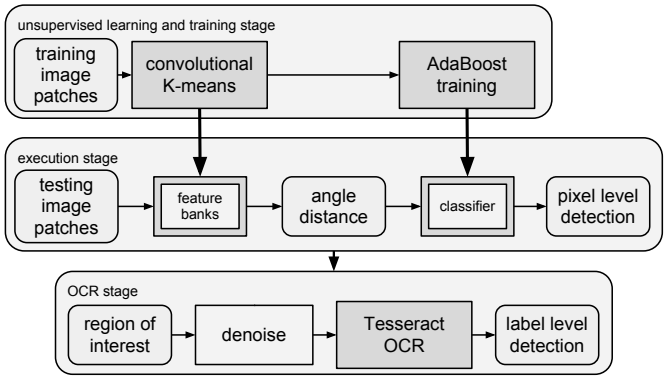



Fig. 1: USPTO patent drawing part label detection and recognition system.

## III. Algorithm

Our system consists of two parts: Text detection and character recognition. The detection system contains two components: Unsupervised feature learning and Ada-Boost classification, as in Fig. 1.

### A. Convolutional K-means Feature Learning

The features used for different kinds of images and different tasks usually varies accordingly. The feature selection is usually done by human with expert knowledge and image processing experience. However, the possibility of making the feature selection automatic and adaptive is attractive. Coates etc. [6] proposed unsupervised feature learning method which generates feature patterns from training images. Then they used a Support Vector Machine to detect texts inside natural scenes. The feature generation and classification are both automated.

In our system, the image pixel values are binarized to black and white. Then the pixels are extracted from the images in raster scan mode. Vast areas of the images are plain white pixels, they are discarded to save time. The pixels are used only when they are black. We extracted their neighbourhood pixels as image patches. For example, if we extract $9\times9$ pixels image patch, the $9 \times 9$ neighbourhood will then be reshaped to a $1 \times 81$ vector used as feature vector.

To address the scale problem, the image patches are produced with different sizes. We use $5 \times 5$, $9 \times 9$, $15 \times 15$, and $19 \times 19$, to cover sizes from small to large. Different image patch sizes are used to address structures in different scales. Small image patch presents local structures while large patch presents relatively global structures in the image.

We use Coates' Convolutional K-means to learn features from the extract feature vectors. The Convolutional K-means method is similar to standard K-means, updating sample cluster labels and shifting cluster centers iteratively to maximize inter cluster distance and minimize intra cluster distance. However, Convolutional K-means uses inner product distance instead of Euclidean distance, as described below:

For sample matrix containing $m$ samples with $n$ features, we make the matrix size as $n \times m$, where each column corresponds a sample vector and each row corresponds a feature, $X \in \mathbb{R}^{n\times m}$. For initialization, we randomly pick $k$ samples from the sample matrix $X$, normalize each vector, so cluster center matrix $D \in \mathbb{R}^{n\times k}$. $k$ is the number of clusters.

The goal here is to minimize the equation:

$$\sum_i ||Ds_i - x_i||^2 \tag{1}$$

as in [6]. Each column of $D$ is the normalized basis vector. $s_i$ is the hot encodings vectors with only one non-zero element. $s_i$ denoted which column of $D$ that current sample $x_i$ is belonging to, and its magnitude is the dot product distance between the current sample and cluster center. $x_i$ is the corresponding training sample. To search for 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 $s_i[k] = D^{(k)\top}x_i$ for $k = argmax_j D^{(j)\top}x_i$. Here , $s_i[k]$ means $k$th elements of $s_i$, $D^{(j)\top}$ means the transpose of $n$th column of D. And other elements in $s_i$ except $s_i[k]$ are set to zero.

Having $s_i$ fixed, we solve for the minimum of equation (1) for $D$ in closed form. Notice that each column of $D$ could be solved independently. So for each column, we solve:

$$\sum_p ||ya_p - x_p||^2 \tag{2}$$

Here $y$ is a column of $D$, which is a center vector of a cluster. $p$ is the indices of samples that are assigned to column/cluster $y$. $a_p$ is the single non-zero element in $s_p$. $x_p$ are samples that are classified to $y$. Compute the first derivative and set it to zero, we have:

$$y = \frac{\sum_p a_p x_p}{\sum_p a_p^2} \tag{3}$$

We now could compute each column $y$ in matrix $D$ using the classified samples $x_p$ and their corresponding hot encodings $s_p$. Notice second derivative of (2) is always greater than or equal to zero, so we are minimizing the distance.

We update $D$ and $s$ alternatively, and after certain number of iterations (512 iterations in experiment) or the overall distance is less than a threshold, we stop the learning process. Columns of $D$ is then used as a bank of quantized features. These quantized features, unlike features designed with prior knowledge, are directly learned from the data.

### B. Ada-Boost Classification

The bank of quantized features we learned from the data is used to classify the samples into two different classes, foreground (part label) and background. For each sample, we first compute the similarity to each of the cluster centers (quantized features). This similarity is represented by the angle distance between sample feature and bank of quantized features. The angle distances are used as features for Ada-Boost.

For unsupervised quantization using $k$ clusters, the new feature is a $k\times1$ vector with each element as an angle distance. For a training sample set with $m$ samples, the training data for Ada-Boost is a matrix with size $k \times m$. The training dataset

is labeled by $\{-1, 1\}$, indicating samples are background or foreground correspondingly.

We use confidence-rated weak prediction Ada-Boost classifier. The weak hypothesis of our Ada-Boost is based on a single threshold decision stump. The confidence of the classifier on each sample is the scalar distance from current sample to the threshold [9]. Thus at each iteration of boosting, we find the single threshold which divides the training samples with minimum weighted error. And we reweight each sample based on the classification correctness and confidence-rate. The misclassified sample weight will increase in the next iteration. The final classifier is a weighted combination of classifiers in each iteration.

### C. 'Tesseract' OCR

As patent labels are usually drawn using typeset characters, we expect it could be recognized by a typical OCR engine with fairly high accuracy. Our system utilizes Tesseract [7] to recognize the symbols, meanwhile, locate the texts more precisely based on the OCR output, once we found the bounding boxes of the part labels.

Tesseract is known for its accuracy on clearly printed text documents. To be able to use Tesseract on patent drawings, we need to firstly localize the text position. Connected foreground image patches are grouped as ROIs. We extract black pixels from ROIs and group them into CCs.

Several denoising methods are used to enhance recognition results. To counter the case where one symbol is separated due to noise or detector's missing, we perform morphological close, which is a morphological dilation followed by erosion, capable of closing small gaps between CCs. We also implement a boundary object remover, where objects touching the boundaries of the current ROI are removed. As most true detections are in the center of the ROI, boundary objects are mostly neighbouring objects. This process reduces false positives.

Because some patent drawings have part label written vertically with 90 degrees counter-clockwise rotation, preprocessing is needed to rotate them back before recognizing. The aspect ratio of each object is defined as its height divided by its width. As horizontally written characters usually have aspect ratio greater than 1, and vertically written characters usually have aspect ratio less than 1. We count the numbers of objects having aspect ratio greater than 1 or less than 1 in each patent drawings. If the final vote decision suggests that they are written vertically, we then rotate all ROIs in this patent drawing 90 degrees clockwise. Then the OCR is performed for each ROI.

We set page segment mode of Tesseract to treat each ROI as a single text line. This setting is important as other settings will produce much lower recognition results. The output of OCR consists of two parts, the recognized symbol class and the symbol location. Then symbol location from OCR output is used above ROI position from detection to localize the part label more accurately.

## IV. Experiment

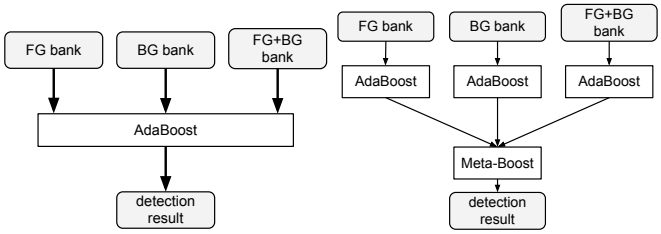
We use USPTO images provided by USPTO Algorithm Challenge I for training and testing. The training set from the competition is used for unsupervised learning and Ada-Boost training, the 'system' set is used to evaluate our system. There are totally 178 training and 93 testing images. Because images are very large, to reduce computation time, they are resampled to 1/6 of the original size before generating image patches. Resample is done by bilinear interpolation, the aspect ratio is preserved. In OCR, the images with original size are used, in order to get sufficient recognition accuracy from Tesseract.

For each image, the competition holder has prepared ground truth. They are polygon vertices values surrounding part labels. The images contain figures, part labels, figure labels, tables and document headers. The patent drawing header is removed easily, because they are all at a fix position on top of the image.

### A. Convolutional K-means Experiment

The ground truth is created for each pixel, if the pixel is inside foreground polygonal bounding boxes, it is considered as a foreground sample. Otherwise, it's background.

The image patches are extract as introduced in section III. Feature vectors are then used to form training dataset matrix. The unsupervised learning method is used to generate quantized feature vectors from the training dataset. Here in experiment, we use 128 clusters in each case, and we train convolutional K-means for 512 iterations.




(a) Type I (b) Type II

Fig. 2: Two types of system architectures. Type I uses all features in one single boost. Type II boosts using different feature banks, and in a cascaded second stage, previous results with confidence ratings are used to generate a final decision.

### B. Ada-Boost and Meta-Boost Experiment

There are three sets of feature banks unsupervised trained from different sample data. One feature bank is trained using foreground samples only, another feature bank uses background, the third one uses both foreground and background. We compare the performance of the classifier using different feature banks. Moreover, we would like to use features from all three banks to create a large feature ensemble. Therefore, we could boost on features from different feature banks and achieve higher performance. Fig. 2 shows two possible Ada-Boost architecture alternatives using different feature banks. In our experiment, we test both structures shown in Fig. 2, within each type of structure, we use image patch sizes from $5 \times 5$ to $19 \times 19$. Thus, from each case, we produced 4 different patch sizes by 128 number of clusters, totally 512 features. In Fig. 2a, we concatenate all features from different feature banks and

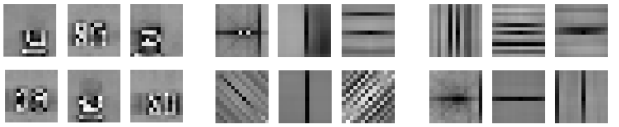
different patch sizes together to form a single feature matrix with each feature length equaling $512 \times 3 = 1536$. In Fig. 2b, we train three Ada-Boost classifiers with different feature banks, then use a Meta-Boost to combine the results of the three.

The Meta-Boost structure utilizes the classification results from each branch classifier. The branch classifier's output is actually the weighted sum of basic classifiers, at the final stage, weighted sum is binarized to produce classification decision, positive or negative. However, if we omit binarization, Ada-Boost can provide a real valued estimation of each sample, the sign is classification decision and magnitude is the fuzzy confidence-rate of the classification. Utilizing this real number output allows designing another Ada-Boost classifier using previous classification results as features and combining different classifiers' decisions to improve accuracy. Because Ada-Boost classification results in branches are used as meta-data for the final classifier, so final classifier is called 'Meta-Boost'.

Meta-Boost will allow faster training in each branch classifier, as they are shorter in feature length comparing to Type I ensemble and could be computed simultaneously. Instead of allowing classifier to freely choose best features in each iteration, Meta-Boost structure forced final classifier to include features from foreground, background and FG+BG bank. In our experiment, as foreground, background and FG+BG banks are describing different aspects of image patterns, classifiers could benefit from this forced inclusion. In Ada-Boost, confidence-rated weak hypothesis is used as in [9]. The Ada-Boost runs to 256 iterations for each architecture in Fig. 2.

### C. Tesseract Experiment

The estimated foreground pixels are then grouped into CCs. A morphological close with $3 \times 3$ pixels kernel is performed beforehand to remove gaps. Other processes, such as boundary objects remover, auto-rotation and underscore remover (see section III C) are also implemented. The bounding boxes are then extracted from the original images. The bounding boxes in detection map are then used as masks to extract ROI from original images. Tesseract OCR is used towards each region to recognize symbol class and locate symbol position more precisely. The Tesseract OCR results are filtered. Only English characters and numbers will be kept, other symbols like punctuation marks, Greek letters, math symbols are discarded. Comparing to other systems (see Section II), our system used extremely simple language model.



(a) Examples of FG feature bank (b) Examples of BG feature bank (c) Examples of FG+BG feature bank

Fig. 3: Quantized feature pattern bank using unsupervised learning algorithm from different sample pools.

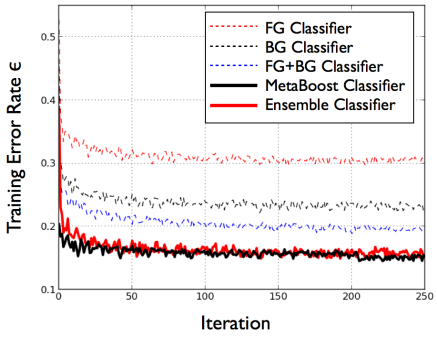
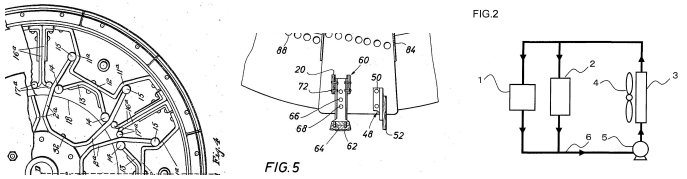
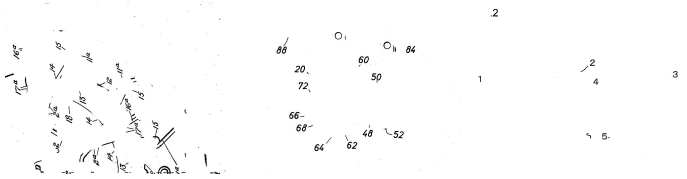


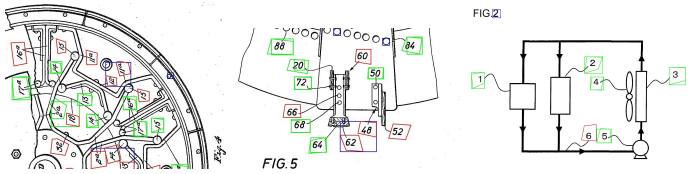

Fig. 4: Error rate through iterations for training using different Ada-Boost classifiers.



(a) Original Image



(b) Text detection map



(c) Tesseract OCR result

Fig. 5: Original image (a), text detection map (b) and Tesseract OCR output (c) examples.

## V. Results and Discussion

We learned the quantized feature banks from FG, BG and FG+BG images patches. From Fig. 3 we could see that feature banks using BG and FG+BG are relatively similar, meanwhile, feature banks using only foreground have larger distance from the other two. The background bank usually catches the horizontal, vertical and diagonal line patterns. The foreground bank catches the patterns where objects are laying in the center and homogeneous regions on the boundaries. This is easy to be imagined, as background commonly contains figures with lines and curves, and label texts are isolated symbols.

The pixel level detection map examples are illustrated in Fig. 5b. The texts are extracted with very high recall value. Detection false positives are showing as noise in the image, most of them can be easily removed by OCR.

We use two types of Ada-Boost architectures shown in Fig. 2. The converge speed is actually very similar among different branch classifiers in Type II. However, the error rates have significant differences. Fig. 4 shows the unweighted error rate variation with iterations for training samples. The FG branch produced worst performance with final error rate 30.5% on training, while the FG+BG branch has best error rate 18.7% on training for pixel-level detection.

The performance of text detectors are also evaluated using precision, recall and F-metric, as listed in Table I. The Meta-Boost classifier achieves slightly better precision and F-measure comparing to Ensemble Ada-Boost for testing samples. All precision, recall and F-metric is evaluated in pixel-level. We count the number of true positive pixels, total number of detected pixels and total number of foreground pixels to compute precisions and recalls.

The examples of OCR errors are illustrated in Fig. 5c. The bounding boxes detected are drawn overtop the original testing images. Correct detections are marked as green, while false positives and missing are marked as blue and red. The detection results will affect OCR performance. False positives as noise CCs around symbols will reduce accuracy of OCR. The false negatives will also make the symbol hard to be recognized. In evaluation, part labels that are partially recognized are assumed as error and marked as red. This evaluation method could be improved in the future.

The final performance evaluation is generated in label-level using tools provided by competition. The method of computing precision and recall can be found in their website (see Section II). Denoise process is important, the OCR accuracy without denoise is much lower. Our system achieves 70.10% precision, 69.33% recall for testing data, as listed in Table II.

We evaluated the performance of detection and recognition part separately by computing precision and recall while making OCR or detection perfect, as shown in Table II. When detection is perfect, Tesseract gets 83.72% precision and 83.72% recall. When OCR is perfect, our system produces 78.26% precision and 96.12% recall on label-level. Our detection has found most true positives while having many false positives. The real system gets both lower precision and recall. Since some part labels are handwritten, they are difficult to be recognized. In contrast, some false positives, like texts in tables or figures, are easier to be read by OCR system. The handwritten part labels and texts in tables or figures that do not belong to part labels caused most of the errors. The results are very close to the Top 1 player of USPTO Competition I, while we are not using high level prior assumptions, language model or text information to the system.

TABLE I: A comparison of our system using different architectures, evaluated using pixel-level precision, recall and F-measure.

| | Precision | Recall | F-measure |
|---|---|---|---|
| FG | 35.4 | 85.3 | 50.03 |
| BG | 60.1 | 92.9 | 72.98 |
| FG+BG | 63.8 | 94.1 | 76.04 |
| Ensemble | 65.7 | **95.2** | 77.75 |
| MetaBoost | **65.9** | 95.1 | **77.85** |

TABLE II: A comparison of different systems, evaluated using label-level precision and recall.

| | Precision | Recall |
|---|---|---|
| Perfect Detection + Tesseract OCR | 83.72 | 83.72 |
| Proposed Detection + Perfect OCR | 78.26 | 96.12 |
| Proposed System | 70.10 | 69.33 |
| Top1 competitor | 72.14 | 69.87 |

## VI. Conclusion

A part label text detection and recognition system for patent drawing image is proposed. Our system learns features from data directly, using Convolutional K-means. An Ada-Boost classifier is trained to automatically classify pixels as foreground or background. A Meta-Boost algorithm is proposed to shorten training while increasing accuracy. 'Tesseract' OCR is used to recognize symbols and refine symbol locations. Our system can produce competitive performance for a difficult dataset without high level prior knowledge. The feature learning, classification, scale selection are all automated, which makes our system very easy to adapt to datasets with different features and scales, e. g. natural scene images. In the future, we will add attentional shift feedback to feature learning and classifier training, focusing on mis-classified samples, and implement Viola and Jones [10] cascaded classifier.

## VII. Acknowledgement

Our thanks go to Marti Hearst for discussion about the algorithm. Our thanks also go to Christoph Riedl for providing the USPTO competition data, results and evaluation tool.

## References

[1] B. Epshtein, E. Ofek, and Y. Wexler, "Detecting text in natural scenes with stroke width transform," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, june 2010, pp. 2963 –2970.

[2] Z. Lu, "Detection of text regions from digital engineering drawings," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, no. 4, pp. 431 –439, apr 1998.

[3] J. Zhang and etc., "Text detection using edge gradient and graph spectrum," in *ICPR*, 2010.

[4] K. Kavukcuoglu, P. Sermanet, Y. Boureau, K. Gregor, M. Mathieu, and Y. LeCun, "Learning convolutional feature hierarchies for visual recognition," *Advances in Neural Information Processing Systems*, pp. 1090–1098, 2010.

[5] A. Coates, H. Lee, and A. Y. Ng, "An analysis of single-layer networks in unsupervised feature learning."

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

[7] R. Smith, "An overview of the tesseract ocr engine," in *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, vol. 2, sept. 2007, pp. 629 –633.

[8] T. Kanungo, R. M. Haralick, and D. Dori, "Understanding engineering drawings: A survey," in *In Proceedings of First IARP Workshop on Graphics Recognition*, 1995, pp. 217–228.

[9] Y. F. Robert E. Schapire, *Boosting: Foundations and Algorithms*. The MIT Press, 2012, ch. 9.2.

[10] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *CVPR 2001*, vol. 1, 2001, pp. I–511 – I–518 vol.1.