ARABIC TEXT DETECTION IN VIDEOS USING NEURAL AND BOOSTING-BASED APPROACHES: APPLICATION TO VIDEO INDEXING

Sonia Yousfi*  Sid-Ahmed Berrani*  Christophe Garcia†

* Orange Labs – France Telecom, 35510 Cesson-Sévigné, France
† University of Lyon, INSA-Lyon, LIRIS, UMR5205 CNRS, 69621 Villeurbanne, France

ABSTRACT
Text detection in videos is a primary step in any semantic-based video analysis systems. In this work, we propose and compare three machine learning-based methods for embedded Arabic text detection. These methods are able to detect Arabic text regions without any prior knowledge and without any pre-processing. The first method relies on a convolution neural network. The two other methods are based on a multi-exit asymmetric boosting cascade. The proposed methods have been extensively evaluated on a large database of Arabic TV channel videos. Experiments highlight a good detection rate of all methods even though neural network-based method outperforms the other ones in terms of recall/precision and computation time.

Index Terms— Arabic text detection, Convolutional Neural Network, multi-exit asymmetric boosting, news video indexing

1. INTRODUCTION
Structuring and indexing news videos require the use of all the information that is available outside and within the video. In particular, embedded text in the frames of the video provides valuable information on what is being shown. It gives for instance the name and the position of a person who is speaking on the screen.

This paper focuses on the detection of embedded text within the frames of Arabic videos, i.e. the localization of bounding boxes that delimit the embedded text. It is a very challenging task due to text variations (e.g. size, style) and acquisition conditions (e.g. background complexity and variability). The focus on Arabic text is motivated by many reasons. First, this language is used by more than half of a billion people in the world and many big Arabic news channels appeared in the last two decades. Second, Arabic text has many specific properties that make its detection, and recognition in a second step, very challenging. Arabic text is cursive. In a single word, characters are often connected by a baseline thereby forming sub-words. This text has also different texture characteristics compared to Latin or Chinese ones: more stokes in different directions, different aspect ratios...

Finally, there are only very few works that have addressed the problem of arabic text detection in videos.

In this study, we propose three machine learning-based methods. The first one is based on Convolutional Neural Network (ConvNet). It performs both text features extraction and classification. The two other methods rely on a multi-exit asymmetric boosting cascade. The first one makes use of Multi-Block Local Binary Pattern representation for feature extraction and performs classification using the Gentleboost algorithm. The second one relies on Haar-like features and Adaboost. Unlike many other methods, these methods perform a robust localization of the text lines without applying any tedious geometric constraints or local image processing.

The rest of the paper is organized as follows. Section 2 gives an overview of related works. Section 3 describes the proposed solutions for Arabic text detection. Experiments and results are then presented and discussed in Section 4. Section 5 concludes the paper and outlines our future work.

2. RELATED WORK
Many methods have been proposed for embedded text detection in videos. They often rely on features like edge, texture, intensity and color distribution. These features are processed using heuristic-based methods, machine learning-based methods or hybrid methods.

Heuristic-based methods apply a set of manually inferred rules directly on the low-level features. They are based on the observation of text characteristics. For instance, edge specificity such as distribution, density and strength is used. Shivakumara et al. [1] apply a Fourier-Laplacian filter to smooth noise and extract text edges followed by k-means to construct text regions. Text lines separation is based on the skeleton of each obtained connected component. Anthimopoulos et al. [2] extract an edge map with the Canny edge detector. Text regions are constructed by applying morphological operations (dilation and opening) on edges. These techniques are very sensitive to variable background and image quality. Therefore, they are not very effective in detecting text from video frames. Moreover, they heavily rely on estimating a set of parameters, which makes their performance very dependent on the processed dataset.
Many machine learning-based methods have been proposed for text detection. They aim at learning discriminative features from a training image dataset in order to build a text/non-text classifier. Kim et al. [3] proposed a method that rely on Support Vector Machines (SVM) in order to directly classify video pixels without any prior knowledge. Another method from Li et al. [4] is based on a Multi-Layer Perceptron that learns mean, second- and third-order moments of the image wavelet decomposition. To localize text, these methods usually use a sliding window technique. They generally show good discriminative and generalization abilities.

Another class of hybrid methods that makes use of both heuristics and machine learning techniques has also been proposed. A coarse text detection is performed using heuristic rules in a first step. Then, a feedback scheme takes place to reject false alarms using a proper classifier. Anthimopoulos et al. [5] propose a technique where text line regions are produced using edge filters and some heuristic rules (dilation, smoothing, projections...). Obtained results are then refined by a SVM classification. Thilagavathy et al. [6] combined binarization for text region extraction and classification with a Multi-Layer Perceptron.

All previously presented techniques are dedicated to Latin text detection. Existing methods dedicated to Arabic text detection in video images are very few. Ben Halima et al. [7] use Multi Frame Integration process to decrease background variations and extract lines and columns that probably contain text. Then, they apply a three-layer perceptron to refine previous results. Moradi et al. [8] propose a purely heuristic method. It is based on the Sobel operator to extract edges and morphological operations (dilation and erosion) are applied on edge maps. Text region identification is performed by histogram analysis. Ahmad et al. [9] apply almost the same strategy but rely on the Laplacian operator to extract edges and the k-means algorithm to cluster text regions.

### 3. THE PROPOSED METHODS

#### 3.1. Convolutional Neural Network-based method

Our first method is based on ConvNet [10] [11]. It is inspired from our previous work presented in [12]. A ConvNet is a hierarchical multilayered neural network that performs feature extraction and classification jointly in a single integrated scheme. It is based on three main hierarchical aspects in order to ensure scale, distortion and shift invariance: local receptive fields, weight sharing and spatial sub-sampling.

##### 3.1.1. Architecture

As shown in Figure 1, the ConvNet we propose is a pipeline of six layers. It receives training labeled images with a retina of 32×64 pixels. The first four layers perform feature extraction. The two last ones are dedicated to feature classification. Each layer contains feature maps resulting from convolution, sub-sampling or neuron unit activation from previous layer outputs. The first layer C1 contains $n_{C1} = 5$ maps performing convolution over the input image with $5 \times 5$ trainable masks. Multiple maps lead to the extraction of multiple features (end points, corners, oriented layers...) directly from the input image pixels. Each of these maps is then sub-sampled in the second layer S1 which reduces by 2 their spatial resolution. This layer contains $n_{S1} = n_{C1}$ feature maps reducing the sensitivity to shifts, distortions and variations in scale and rotation. The next two layers C2 and S2 perform feature combination in order to produce more complex sets of feature maps. Layer C2 is a convolution layer. It contains $n_{C2} = (n_{S1} \times 2) + n_{S1}$ maps: each map of S1 gives two maps (convolution with two different $3 \times 3$ weighted masks). Each pair of maps of S1 are combined in one map after applying a convolution with two different $3 \times 3$ kernels. Layer S2 is the sub-sampling layer of C2 maps. The last two layers are a simple MLP that perform feature classification. They are composed of standard sigmoid neurons. Each neuron in layer N1 is connected to only one feature map of S2. Layer N2 contains one neuron indicating the class of the input image (1/-1).

##### 3.1.2. Training

The network is trained to extract appropriate text image features and classify text and non-text images. The database used for training is divided into two sets: training and validation. Both sets contain images of text and non-text (see Section 4.1) with the same size of the network retina. At each iteration, the ConvNet is trained with an equal number of positive and negative examples randomly selected from the training set. For each example $i$, a mean square error $MSE = (o_i - d_i)^2$ is computed, where $o_i$ is the network response and $d_i$ is the desired output. The error is then back-propagated throw all layers to update weights. The validation set is used to check the generalization ability of the network during training and to avoid over-fitting. To boost the rejection ability of false
alarms, we use a bootstrapping technique. After each training epoch, a set of false alarms are gathered by running the ConvNet on various images that do not contain any text. These examples are added as negative examples to the training set. At each epoch, a grabbed example is considered as false alarm if the network response is greater than a threshold \( T_{hr} \). Initially, \( T_{hr} = 0.8 \) when the network is still a weak classifier. This threshold is then gradually reduced.

### 3.2. Boosting-based methods

The two other proposed methods are based on multi-exit boosting cascade. They learn to distinguish text and non-text areas using Multi-Block Local Binary Patterns (MBLBP) and Haar-like features.

#### 3.2.1. Feature extraction

MBLBP features [13] consist in encoding rectangular regions using the Local Binary Pattern operator. Then, a binary code is produced by comparing the average intensity of the central rectangle \( r_c \) with its \( 3 \times 3 \) neighborhood \( X = \{ r_0, \ldots, r_8 \} \). Haar-like features, popularized by Viola and Jones [14] for face detection, are based on the difference between the average intensities in rectangular regions. To improve the computational efficiency of these two descriptors, the integral image technique is used [14].

#### 3.2.2. Multi-exit asymmetric boosting

For classification, we propose to use the multi-exit asymmetric boosting cascade introduced in [15]. We propose two methods derived from this general method. In the first one, we combine MBLBP features and a GentleBoost based multi-exit asymmetric cascade. In the second one, relevant Haar-like features are selected at each node of the cascade by the AdaBoost algorithm. In the cascade, intermediate strong classifiers are represented by a set of nodes having indices \( \mathcal{N} \). Each classifier (node) makes a decision to pass or reject the input image. The strong classifier is constructed from a sequence of \( n \) weak classifiers. Unlike conventional cascade proposed in [14], nodes are able to use overlapped sets of weak classifiers, i.e. each node exploits these weak hypotheses from the beginning to a number \( n \) which corresponds to a fixed training detection rate and a fixed false positive rate. The final classifier resulting from the cascade is expressed as follow:

\[
F(x) = \begin{cases} 
+1, & \text{if } \sum_{i=1}^{n} w_i h_i(x) \geq 0, \forall n \in \mathcal{N}; \\
-1, & \text{otherwise}
\end{cases}
\]

where \( h_i \) are weak classifiers and \( w_i \) their weights. In order to improve the false alarm rejection ability, we use a bootstrapping procedure (similar to the one used for ConvNet).

### 3.2.3. Multiscale text search

In order to detect text areas of different sizes, a sliding window scans a multi-scale pyramid of the input image. Each of our classifiers is then applied on each window. Produced text candidates at different scales are then back-projected into the original input image. A k-means-like algorithm is used to cluster them and construct text regions according to their proximity in space and scale. Each region contains the average scale and density of the cluster. By applying a threshold on the density, we can eliminate a considerable number of false alarms. For ConvNet, we apply convolution and subsampling filters to the whole image at each scale instead of repeating them at each position which strongly reduces computation time. In addition, our schema is robust to multi-line detection giving that classifiers are trained to reject grouped lines, by adding such examples in the negative set.

### 4. EXPERIMENTS

#### 4.1. Dataset

Training sets have been gathered from three different Arabic TV channels: Al Jazeera, Al Arabiya and France24 Arabic. Negative examples have been extracted from a set of images that do not contain text. We have used 30,000 text images for both methods. During training, the methods gather, by bootstrapping, a set of false alarms at each epoch/node which is added to their initial non-text sets. At the end of training, we have reported 111,000 negative examples for the boosting training set and 80,000 negative examples for ConvNet. This inequality is explained by two reasons: (1) boosting-based methods use more nodes/epochs than ConvNet, so more bootstrapping iterations; (2) at early iterations the boosting classifier is weaker than the ConvNet one; it thus gathers more false alarms. Positive and negative examples are given in Figure 2. Two test sets for evaluating the proposed methods have been collected:

1. ES1: A set of 201 images from Al Arabiya, Al Jazeera and France24 Arabic with 959 annotated text areas.

2. ES2: A set of 164 images collected from the BBC Arabic channel containing 1017 annotated text areas.

Images in ES1 and ES2 have not been used to construct the training set. Moreover, images in ES2 come from a TV channel that has not been used at all for collecting training examples.
4.2. Evaluation results

Hereafter, “HAARada” refers to the model based on Haar-like feature and Adaboost. “MBLBPgentle” refers to the model with MBLBP features and Gentleboost. As for the evaluation of the final detectors in terms of recall and precision, we use the metrics proposed in [16]. The detection rate (DR) and the number of false alarms (#FA) are also computed for each of the three methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Recall</th>
<th>Precision</th>
<th>F-measure</th>
<th>DR</th>
<th>#FA</th>
<th>Resp. time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvNet</td>
<td>0.75</td>
<td>0.8</td>
<td>0.77</td>
<td>90%</td>
<td>45</td>
<td>7.25</td>
</tr>
<tr>
<td>HAARada</td>
<td>0.77</td>
<td>0.72</td>
<td>0.74</td>
<td>92%</td>
<td>170</td>
<td>14.75</td>
</tr>
<tr>
<td>MBLBPgentle</td>
<td>0.70</td>
<td>0.32</td>
<td>0.44</td>
<td>85%</td>
<td>220</td>
<td>46</td>
</tr>
</tbody>
</table>

Table 1. Experimental results on ES1.

Obtained results on ES1 are reported in Table 1. The last column gives the average response times for 576 × 1024 images\(^1\). These results reflect the good detection capacity of the proposed methods. We can notice the excellent precision rate of the ConvNet-based method. This is due to the good rejection ability of false alarms (cf. the 5\(^{th}\) column of the table). Although HAARada outperforms the other methods in terms of recall, the ConvNet-based detector still realizes the best compromise in terms of recall/precision and detection rate/number of false alarms. We notice, however, the low amount of precision for the MBLBPgentle method. This can be explained by the nature of the chosen features that capture mostly large scale structures in the image which increases the number of false alarms. The ConvNet-based detector outperforms also boosting-based methods in terms of response time. ConvNet applies features filters on the whole image before applying the sliding window.

In order to further evaluate the generalization of the proposed methods, we have used ES2. Obtained results are reported in Table 2. The table shows a good generalization of the three methods. There are almost no significant difference between these results and the ones obtained on ES1 (cf. Table 1). Some detection results of the ConvNet method are shown in Figure 3. We can see in image (b) that some scene texts are also detected. This shows the generalization ability of our methods. However, it is worth to point out that these detections are considered as false alarms by our evaluation procedure. We have annotated only embedded text in our test sets.

<table>
<thead>
<tr>
<th>Method</th>
<th>Recall</th>
<th>Precision</th>
<th>F-measure</th>
<th>DR</th>
<th>#FA</th>
<th>Resp. time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvNet</td>
<td>0.77</td>
<td>0.75</td>
<td>0.76</td>
<td>97%</td>
<td>66</td>
<td></td>
</tr>
<tr>
<td>HAARada</td>
<td>0.75</td>
<td>0.66</td>
<td>0.70</td>
<td>94%</td>
<td>134</td>
<td></td>
</tr>
<tr>
<td>MBLBPgentle</td>
<td>0.72</td>
<td>0.25</td>
<td>0.37</td>
<td>85%</td>
<td>563</td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Experimental results on ES2.

\(^1\)Experiments have been conducted on a machine running Intel(R) Core(TM) i5, 2.67 GHz, 4Gb of RAM.

\(^2\)The number of frames corresponds to the minimal duration of text appearance in the video.

Fig. 3. Examples of detection results using ConvNet-based method.

4.2.1. Application to video indexing

Embedded text appearance in a video often indicates the beginning of an interesting sequence (e.g. the beginning of a new subject in news or the appearance of a person on screen). Text detection can be then directly used to detect such key-moments. In this work, we apply the ConvNet-based detector (our best method), in order to detect Arabic embedded text in a football match (new scores, penalty...). The ground-truth is a set of segments \(G = \{G_i, i = 1 \ldots n\}\) indicating embedded text. Detection results consist in a set of segments \(D = \{D_i, i = 1 \ldots m\}\) each containing frames with detected texts. Recall and precision are computed as follow:

\[
R = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} \omega(G_i, D_j)}{n} \quad P = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} \omega(G_i, D_j)}{m}
\]

where \(\omega(G_i, D_j)\) equals 1 if the \(D_j\) and \(G_i\) overlapping is over 29 frames\(^2\), and 0 otherwise. Obtained results show that 100% of embedded texts in the ground-truth are detected, that is our method is able to identify 100% of the highlights (recall=1). As for precision, it is equal to 79%. It is worth pointing out that 50% of reported false alarms contains sequences with scene texts that have been detected.

5. CONCLUSION

We have presented in this study three different machine learning-based methods for embedded Arabic text detection in news videos: neural and boosting based methods. The built detectors are able to extract text lines without any pre-processing or tedious heuristic constraints. Experimental results highlight the good detection abilities of our methods specially for the ConvNet-based method with very few false alarms. The obtained detector is then exploited in an application for video indexing. In our future work, we will address Arabic text recognition in videos. In addition to the difficulties related to the characteristics of Arabic text, the low quality of text extracted from videos is another big challenge to address.
6. REFERENCES


