First online: 20 September 2016

An Enhanced Bag-of-Features Framework for Arabic Handwritten Sub-words and Digits Recognition

Mohammed O. Assayony[*1], Sabri A. Mahmoud[2]

[1,2]Information & Computer Science Department, King Fahd University of Petroleum & Minerals, Dhahran 31261, Saudi Arabia

[*1]g201102150@kfupm.edu.sa; [2]smasaad@kfupm.edu.sa

***Abstract*-In recent years, feature learning approaches have gained substantial interest and are successfully applied to challenging problems in facial recognition, visual object retrieval and classification, document image analysis and, recently, in handwriting recognition. In this paper, we present a feature learning framework for Arabic handwritten text recognition based on the Bag-of-Feature (BoF) paradigm. Utilizing the characteristics of handwritten text, we developed two novel versions of SIFT that are discriminative and computationally efficient with half the size of the original SIFT descriptors. To evaluate the quality of the features learned by the framework and the efficiency of the proposed versions of SIFT, we conducted extensive experimental work on two Arabic handwritten text datasets, viz. the non-touching Arabic Indian digit and Arabic sub-words datasets of CENPARMI Bank check database. Our framework achieves state-of-the-art accuracies on both datasets. The recognition performance and the computational efficiency are the result of utilizing the unique properties of the handwritten text.**

Keywords- Feature Learning; Bag-of-Features; Arabic Handwriting Recognition; SIFT

I. INTRODUCTION

Handwriting recognition is a branch of pattern recognition concerned with automatic conversion of images of handwritten text into text representation. The advances in the area of handwriting recognition have assisted the automation of several demanding applications in daily life, including bank check processing, postal address reading and business form processing. Despite these successes, handwriting recognition remains a challenging task due to the large variability in human writings that produce visual differences in size, slant and pen-stroke of characters' shapes (Fig. 1). The effect of such variability and other types of noise are addressed by extracting robust features prior to the recognition. Deciding the type of features, however, is difficult and requires considerable effort. Vast number of features have been handcrafted by experts based on their prior knowledge and experience in the field, including structural features, e.g. skeleton representation, strokes, and loops, and statistical features like gradient histograms, projection profiles, energy of Gabor filters, and wavelets transforms coefficients, among others [1]. The new trend in computer vision and machine learning is to design techniques that can automatically learn robust features without the need for specific domain knowledge [2]. Several feature learning paradigms have been applied to handwriting recognition, e.g., Convolutional Neural Networks (CNN) [3-5], Recurrent Neural Networks (RNN) [6, 7] and Deep Believe Networks (DNN) [8, 9]. Despite their successes in learning robust features for handwritten text of several scripts, the aforementioned approaches rely on deep neural networks, which are computationally expensive and require large datasets for training. Bag-of-Features (BoF), however, is an alternative paradigm which is computationally efficient and can be trained with a reasonable number of samples [10]. The paradigm was utilized in visual object classification [11] and image retrieval [12] as well in document image processing applications, including handwriting recognition [13], word spotting [14] and writer identification [15].

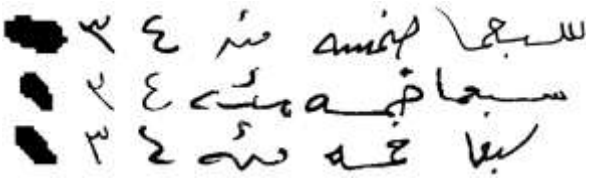


Fig. 1 Samples of Arabic digits and subwords written by different writers

In this work, we exploit the Bag-of-Features framework to learn robust feature representations for Arabic handwriting recognition. Though the framework has been applied before for Arabic handwriting recognition [13], we propose utilizing the characteristics of the handwritten text in order to enhance the quality of the learned features and improve the computational performance of the framework. Since the framework involves several steps that can be implemented by a variety of techniques, we thoroughly investigate several techniques, focusing on the options that have an impact on the quality of the learned features. Precisely, we propose two novel versions of SIFT that achieve the discriminative power of SIFT and are computationally efficient with half the size of the SIFT descriptors. To provide better representation for the handwritten text, we employ dense

sampling instead of the Harris detector used in [13]. As the size of the generated codebook plays important rule in the computational and discrimination performance, we built several codebooks of different sizes and selected the size that resulted in the optimal performance. To alleviate the quantization distortion associated with the traditional hard assignment, a soft assignment based on Gaussian Models is implemented and its performance is compared with the naïve hard assignment. To evaluate and analyse the efficiency of the proposed modifications, the framework is integrated with a holistic handwriting recognition system that we applied on two Arabic handwritten text datasets, the non-touching Arabic Indian digits and the Arabic sub-words datasets of CENPARMI Bank check database [16].

The rest of this paper is organized as the follows. Section II provides an overview of the Bag-of-Features framework and shows how the handwritten text characteristics are utilized to enhance the framework. The details of the handwriting recognition system we experimented with are presented in Section III. In Section IV, we present the experimental results, and Section V concludes the paper.

II. BAG-OF-FEATURES FRAMEWORK

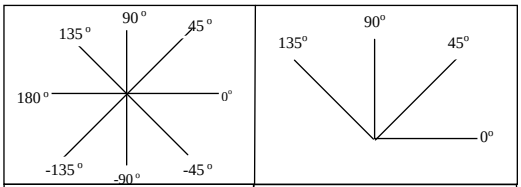
In the Bag-of-Features framework, an image is represented by the frequencies of occurrences of its local features [10]. The framework includes two-phases, viz. codebook generation and BoF vector construction. Each of the two phases involves several steps that can be implemented by variety of techniques. This pipeline organization gives BoF framework the flexibility to apply to different applications. The codebook is generated by extracting local features from the training data and clustering those features using off-the-shelf clustering algorithms. The codebook is the set of clusters' centroids where each centroid is called a codeword. The BoF vector of an image is constructed by extracting local features from the image. Each extracted feature is quantized to the closest codeword in the codebook, where the closeness relation is defined based on a distance metric. The frequency of each visual word is used to represent the image. The resulting BoF vector represents the statistics of the image local features. It can be used to train classifiers for image classification and categorization tasks, for object detection and recognition applications, or to index images in databases for efficient image retrieving applications.

Extracting local image features involves feature detection and description. Different detection techniques have been proposed in computer vision, e.g., Hessian, Harris, Hessian-Laplacian and Harris-Laplacian, and Difference-of-Gaussian, in addition to dense and random sampling that provide better coverage for image contents [17, 18]. Several approaches were proposed for interest region description [19, 20]. However, the most state-of-the-art results are obtained by gradient-based descriptors, especially SIFT descriptors [21]. For codebook generation, most BoF works are based on k-means clustering for codebook generation. In the quantization step, two approaches are common, viz. hard assignment, where a descriptor is assigned to the nearest codeword, and soft assignment, where a descriptor is assigned to the nearest k codewords in order to alleviate the quantization distortion associated with the hard assignment.

We believe that utilizing the characteristics of the handwritten text could enhance the quality of the learned features and reduce the overheads of several stages of the framework. We show below how the characteristics of the handwritten text could be employed in improving SIFT descriptors.

A. Reducing Descriptor Dimensionality

SIFT algorithm relies on the distribution of the gradient magnitude to describe the regions of interest. Once the gradient magnitude and orientation for each pixel in the region are computed, the region spatial area is divided into 4×4 regions and the 360 degree gradient orientation range is quantized into 8 orientation bins (Fig. 2 (a)). After that, the histogram of the gradient magnitudes at each region is computed, giving a descriptor vector of 4×4×8 = 128 elements for the patch. In text recognition, however, our concern is the line orientation rather than the individual pixel orientation. For instance, when a pixel shows -90^o orientation, this indicates that the pixel lies on the lower edge of a horizontal line, whereas +90^o orientation indicates that it lies on the upper edge of a horizontal line. In both cases, the line is horizontal. Therefore, the two symmetric orientation bins can be combined into a single bin, giving 4 orientation bins instead of 8 (Fig. 2 (b)) and the descriptor dimensionality is reduced from 128 to 64. Note that the distance between the two adjacent bins is still 45^o, similar to the original SIFT algorithm. The only difference is that the values of the negative orientation bins are accumulated to the corresponding symmetric positive orientations. This modification produces shorter vectors for the text patch with the same discriminative power of the original SIFT.



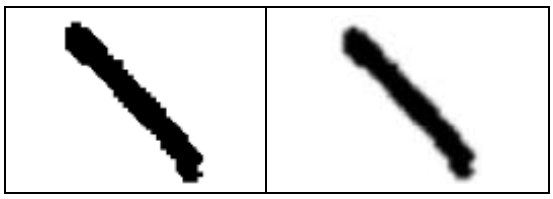
(a) The range [0^O, 360^O] is quantized into 8 bins

(b) Combining the symmetric orientation bins gives 4 bins

Fig. 2 Orientation quantization

B. Fast Computation of the Gradient Magnitude and Orientation

In order to compute the pixels' gradient magnitude and orientation, the original SIFT algorithm applies the basic derivative filters (h_x = [-1 0 1], h_y = [-1 0 1]^T) for evaluating the horizontal and vertical derivatives (d_x, d_y) at each pixel. The justification of using the basic derivative filters rather than large filters like Prewitt or Sobel filters is that the image has to be pre-smoothed by Gaussian kernels. We argue that Gaussian smoothing is computationally expensive, so several descriptors like SURF [22], BRIEF [23] and LDP [24] have avoided it by using efficient approximations, i.e., box filters that could be efficiently implemented using integral images. Moreover, for handwritten text where the images are binary, the noise associated with this type of images is due to the writing style and the binarization algorithm. The noise appears as sharp edges due to the transition from 0 to 1 or from 1 to 0. Applying Gaussian smoothing would attenuate these edges rather than permanently eliminating the notches on the text borders (Fig. 3). Further, smoothing affects the binary values of almost all pixels - even those pixels that are far from the edges of the text line. The changes in the values of the pixels in the middle of the text line generates noisy gradients.



(a) Original image (b) Smoothed image

Fig. 3 A Digit Sample smoothed by a Gaussian kernel. The original digit has 821 black pixels. After smoothing, only 8 pixels remained black

To avoid these effects, we eliminated the smoothing step and applied the basic derivative filters directly. Since the images are binary, the filter response at any image pixel would take one out of three values {-1, 0, 1} based on the intensity value (0/1) of the left/right and top/bottom neighbor pixels. Consequently, the gradient magnitude and orientation can be computed by building lookup tables instead of applying computationally expensive procedures for computing arctan and square-root functions. The lookup tables are shown in Fig. 4. The possible values of the gradient orientation in Fig. 4 (b) are exactly the 8 orientation bins shown in Fig. 2 (a), which implies that the gradient magnitude would be accumulated to exactly one bin according to its orientation. Utilizing these lookup tables would enable us to construct faster SIFT descriptors compared with the original algorithm. Using the reduction approach explained in the previous subsection, the descriptor dimensionality would be reduced to half by combining the corresponding symmetric orientation bins. The lookup table for computing pixel orientation after combining the corresponding symmetric orientations is shown in Fig. 4 (c).

d_y \ d_x	-1	0	1
-1	$\sqrt{2}$	1	$\sqrt{2}$
0	1	0	1
1	$\sqrt{2}$	1	$\sqrt{2}$

(a) gradient magnitude

d_y \ d_x	-1	0	1
-1	-135°	-90°	-45°
0	+180°	+90°	0°
1	+135°	+90°	+45°

(b) gradient orientation before combining the symmetric orientation bins

d_y \ d_x	-1	0	1
-1	+45°	+90°	+135°
0	0°	+90°	0°
1	+135°	+90°	+45°

(c) gradient orientation after combining the symmetric orientation bins

Fig. 4 Lookup tables for computing gradient magnitude and orientation

III. ARABIC HANDWRITING RECOGNITION SYSTEM

To assess the quality of the learned features as well as the performance of the proposed modifications, the BoF framework is integrated with a holistic handwriting recognition system. The general model of the holistic handwriting recognition system is shown in Fig. 5. Below, we give the details of the options we implemented for each step in the system.

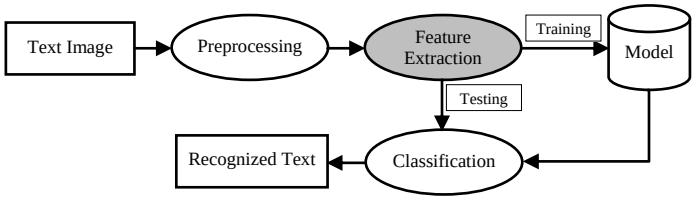


Fig. 5 General model of the handwriting recognition system

In the preprocessing step, image samples are normalized to 64 pixels in height while maintaining the aspect ratio. Height normalization is a common preprocessing step for reducing the side effect of the variety in text size. 64-pixel height normalization was used with the Non-Touching Digits Dataset in [25] and [26]. The image samples shown in Fig. 1 all have a height of 64-pixels. For feature extraction, we applied the BoF framework. As the framework involves several stages that can

be implemented in diverse approaches, we have implemented different approaches for each stage. The Harris detector, Harris-Laplace detector and dense sampling are implemented for selecting representative image regions. In the dense sampling approach, the image spatial area is divided into a grid of overlapping fixed-sized patches where the descriptor algorithm is applied on each patch. We use grids of four patch sizes (viz. 16, 24, 32 and 40 pixels) and a stride of 8 pixels in the four grids. The preliminary experiments showed that multi-size sampling has better performance than using single-size. Though a stride shorter than 8 pixels would produce better accuracy, this value makes a balance between the accuracy and the computational performance, especially when the GMM is applied in the quantization step, as presented below. As the Harris detector doesn't report a scale for the detected regions, we use a fixed-size patch of 24 pixels centered at each detected region. We found experimentally that this size is the best among the four sizes that are used in the dense sampling approach. SIFT produces a 128-D descriptor for each region. We apply PCA to reduce the SIFT descriptors to 64-D vectors. For codebook generation, we apply the K-Means clustering algorithm on a set consisting of one million descriptors selected randomly from the training samples. Moreover, Gaussian Mixture Models GMMs are estimated for the selected descriptors. Several codebooks of sizes ranging from 128 to 2048 codewords are generated. For quantization, hard assignment and soft assignments are implemented. The image final feature vector is obtained by averaging the occurrences of the codewords in the image. We use the GMM to compute a-posteriori probabilities of the descriptors and accumulate them to produce the final image vector in soft assignment. The classification step of our recognition system is implemented using the Support Vector Machine (SVM) with linear kernel. Table 1 summarizes the main configuration of our handwriting recognition system.

The system is implemented in MATLAB R2012b and run on a server with two Intel Xeon X5690 processors of 3.47 GHz and 88 GB RAM running Windows 7 Professional N. We used the *CornerDetector System object* of the MATLAB Computer Vision System Toolbox for Harris implementation and VLFeat library [27] for the other algorithms.

TABLE 1 CONFIGURATION OF THE HANDWRITING RECOGNITION SYSTEM

Stage	Value
Preprocessing	Height normalization
Local Feature Detector	Harris: *region size of 24 pixels.* Harris-Laplace: *the region scale specifies the region size.* Dense Sampling: *4 grids of sizes 16, 24, 32 and 40 with 8-pixels stride*.
Descriptor	SIFT *SIFT vectors are reduced to 64-D by applying PCA.*
Codebook generation	K-Means *applied on 1 million random descriptors.* GMM estimated over the codebook.
Codebook Size	128, 256, 512, 1024 and 2048
Assignment Approach	Hard Assignment Soft Assignment based on GMM
Classification	1-vs-all SVMs with linear kernel

IV. EXPERIMENTAL RESULTS

We have conducted several experiments to evaluate the features learned by the proposed framework. Our experiments are conducted on the non-touching Arabic Indian digits and non-touching Arabic subwords datasets of the CENPARMI database [16]. The non-touching Arabic/Indian digit dataset contains 10425 image samples of isolated Arabic/Indian digits (0-9) extracted from the courtesy amount field of the checks. The samples are divided into the training set (7390 images) and the testing set (3035 images), where the samples of each digit are grouped into separate class in each of the training and testing sets. Fig. 6 shows the statistics of the digit classes with a sample image example from each digit class. The non-touching Arabic subwords dataset contains 27985 image samples of Arabic subwords used in legal amounts of Arabic checks. The dataset has 87 classes corresponding to the Arabic subwords encountered in the legal amount filed of the collected checks' samples. As the samples are extracted from real checks, the distribution of the samples of the classes are unbalanced, with some classes having a single sample while the testing set is empty.

Class	Training	Testing	Total	Sample
0	3793	1574	5367	
1	782	304	1086	
2	545	225	770	
3	362	144	506	
4	307	133	440	
5	649	263	912	
6	279	111	390	
7	233	109	342	
8	246	98	344	
9	194	74	268	
Total	**7390**	**3035**	**10425**	

Fig. 6 Statistics of the non-touching digits dataset

A. Non-Touching Digits Dataset

The first set of experiments were conducted on the non-touching Arabic/Indian digits. The obtained recognition accuracies are shown in Fig. 7. The results show that, on average, dense sampling outperforms the two interest point detectors. This result is consistent with comparative studies conducted for visual object recognition tasks that show that dense sampling produces better recognition rates than interest point detectors [18]. The best results of the three sampling strategies are shown in Table 2.

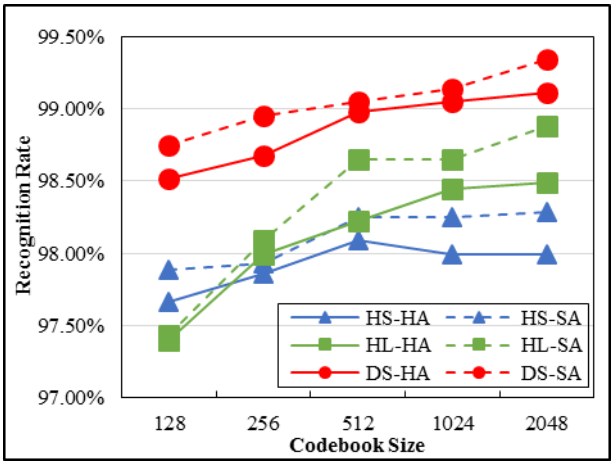


Fig. 7 Recognition rates for digit dataset with different local feature detection and quantization approaches
HS = Harris, HL = Harris-Laplace, DS = Dense Sampling
HA = Hard Assignment, SA = Soft Assignment

TABLE 2 THE BEST RECOGNITION ACCURACY ACHIEVED BY THE THREE DETECTION METHODS
CODEBOOK SIZE = 2048, QUANTIZATION = SOFT ASSIGNMENT

Class	Harris	Harris-Laplace	Dense Sampling
0	99.56%	99.68%	99.36%
1	96.71%	98.03%	97.70%
2	98.22%	97.78%	100.00%
3	96.53%	98.61%	100.00%
4	98.50%	97.74%	99.25%
5	96.21%	98.48%	99.24%
6	98.20%	100.00%	100.00%
7	93.58%	97.25%	100.00%
8	98.99%	97.98%	100.00%
9	94.59%	94.59%	100.00%
Average	**98.29%**	**98.88%**	**99.34%**

Comparing the hard and soft assignments, we found that soft assignments gave better accuracies in all cases, since the soft assignment reduces the side effect of quantization distortion associated with the hard assignment. The results show also the positive effect of increasing the codebook size up to 2048. The best recognition rate of 99.34% was achieved by dense sampling with soft quantization and a codebook size of 2048. The hard quantization with the codebook of the same size achieved 99.11%, while the codebook size of 1024 achieved 99.14% and 99.05% with the soft and hard assignments respectively. Similarly, the codebook size of 512 achieved a 99.05% recognition rate with the soft assignment. These results outperform the state-of-the-art recognition rates published in the literature on the same dataset (Table 3).

TABLE 3 RECOGNITION ACCURACIES REPORTED IN THE LITERATURE FOR
CIMPARMI NON-TOUCHING DIGITS DATASET VS. THE BEST ACCURACIES ACHIEVED IN THIS WORK

Features and Classifier	Accuracy	Statistical significance
Dense Sampling, 2048 codebook, Soft Assignment with SVM (This work)	99.34%	±0.29
Dense Sampling, 1024 codebook, Soft Assignment with SVM (This work)	99.14%	±0.32
Dense Sampling, 2048 codebook, Hard Assignment with SVM (This work)	99.11%	±0.33
Dense Sampling, 1024 codebook, Hard Assignment with SVM (This work)	99.05%	±0.34
Dense Sampling, 512 codebook, Soft Assignment with SVM (This work)	99.05%	±0.34
GSC Features with SVM [28]	99.04%	±0.34
Log-Gabor Filter Response Features with SVM [25]	98.95%	±0.35
Pixel Intensity Values with Bernoulli Mixtures [29]	98.10%	±0.45
Pixel Intensity Values with Bernoulli HMM [30]	98.00%	±0.46
Spatial Gabor Filter Response Features with 1-NN [26]	97.99%	±0.46

The table shows the statistical significance of the recognition rates at the 95% confidence level. The table shows that some of the results are not statistically significant, although they are higher than published work. This indicates the power of the framework in learning robust feature representation. Fig. 8 shows the confusion matrix of the tested samples for the configuration that gives the best accuracy (dense sampling with soft quantization and codebook size of 2048). We achieved a recognition rate of 100% in six classes (2, 3, 6, 7, 8 and 9), while most of the misclassified samples are in class 0 (10 samples) and class 1 (7 samples). 20 total samples were misclassified. These samples are shown in Fig. 9. It is clear that the misclassified samples of classes 0 and 1 are similar, making the differentiation between them hard even for a human. The large, wide hole in sample #10 makes digit 0 looks like 5. On the other hand, the thick border and small hole in sample #19 make digit 5 looks like 0. The soft mid-concavity of the misclassified sample of class 4 (sample #18) makes it similar to digit 2, especially when we compare it with some training samples of digit 2 that ended with sharp right tail as in (٢) and (٢). Our approach successfully recognized several challenging samples in this class, like (٤), (٤), (٥) and (٤) that were misclassified in other works [25, 28]. Our approach also tolerates the cut found in the middle of some samples like the ones in (٥), (٨) and (٨). Digit 3 is usually written with three upper strokes (٣) and sometimes with two (٣). While previous approaches [25, 28] have difficulties to recognize them, our approach achieved 100% accuracy in this class.

Actual Class	Predicted Class										Recognition Rate
	0	1	2	3	4	5	6	7	8	9	
0	1564	8	0	0	1	1	0	0	0	0	99.36%
1	7	297	0	0	0	0	0	0	0	0	97.70%
2	0	0	225	0	0	0	0	0	0	0	100.00%
3	0	0	0	144	0	0	0	0	0	0	100.00%
4	0	0	1	0	132	0	0	0	0	0	99.25%
5	1	0	0	0	0	262	0	1	0	0	99.24%
6	0	0	0	0	0	0	111	0	0	0	100.00%
7	0	0	0	0	0	0	0	109	0	0	100.00%
8	0	0	0	0	0	0	0	0	99	0	100.00%
9	0	0	0	0	0	0	0	0	0	74	100.00%
										Average	**99.34%**

Fig. 8 Confusion matrix of the tested digit samples for dense sampling with codebook of size 2048 and soft quantization

No.	Image	Actual Class	Predicted Class	No.	Image	Actual Class	Predicted Class
1		0	1	11		1	0
2		0	1	12		1	0
3		0	1	13		1	0
4		0	1	14		1	0
5		0	1	15		1	0
6		0	1	16		1	0
7		0	1	17		1	0
8		0	1	18		4	2
9		0	4	19		5	0
10		0	5	20		5	7

Fig. 9 Images of the misclassified digit samples

B. Non-touching Subwords Dataset

Two sets of experiments are conducted on the non-touching subwords dataset. The first is conducted on the most frequent 10 classes (Fig. 10) in order to compare our results with published work [31], while in the second set we consider all classes that contain at least one sample in the training set and one sample in the testing set (Fig. 11). In all experiments, SIFT descriptors are extracted densely using four scales viz 4, 6, 8 and 10 pixels. Five codebooks of sizes 128, 256, 512, 1024 and 2048 are used. Both the Hard and Soft Assignments are evaluated. The obtained accuracies are shown in Fig. 12.

No	Training	Testing	Total	Sample
1	2061	859	2920	
2	1896	781	2677	
3	1726	724	2450	
4	1301	529	1830	
5	1175	490	1665	
6	1159	521	1680	
7	1077	442	1519	
8	1005	410	1415	
9	961	396	1357	
10	745	304	1049	
Total	**13106**	**5456**	**18562**	

Fig. 10 Statistics of the most frequent 10 classes of the non-touching subwords dataset

No	Train	Test	Total	Sample	No	Train	Test	Total	Sample	No	Train	Test	Total	Sample
1	1993	829	2822		24	12	4	16		47	71	28	99	
2	4	2	6		25	51	20	71		48	93	39	132	
3	10	4	14		26	2	1	3		49	1	1	2	
4	24	8	32		27	161	61	222		50	21	10	31	
5	2061	859	2920		28	80	37	117		51	83	35	118	
6	567	244	811		29	4	1	5		52	6	2	8	
7	1175	490	1665		30	362	147	509		53	64	26	90	
8	961	396	1357		31	319	121	440		54	40	18	58	
9	242	103	345		32	55	22	77		55	13	5	18	
10	1896	781	2677		33	11	5	16		56	277	116	393	
11	37	16	53		34	4	2	6		57	158	55	213	
12	821	330	1151		35	11	4	15		58	83	38	121	
13	6	3	9		36	1	1	2		59	69	33	102	
14	187	82	269		37	114	47	161		60	96	36	132	
15	92	36	128		38	62	27	89		61	4	1	5	
16	62	31	93		39	23	11	34		62	61	28	89	
17	2	1	3		40	473	198	671		63	33	15	48	
18	81	30	111		41	1077	442	1519		64	4	2	6	
19	1726	724	2450		42	1005	410	1415		65	87	29	116	
20	745	304	1049		43	130	49	179		66	183	80	263	
21	216	83	299		44	12	5	17		67	10	4	14	
22	69	25	94		45	2	1	3		68	78	40	118	
23	1301	529	1830		46	10	4	14		69	3	1	4	

Fig. 11 Statistics of the non-touching subwords dataset

Similar to the digit dataset, the soft assignment outperformed the hard assignment in all codebook sizes, and the best accuracies are achieved with a codebook of size 2048. In the most frequent 10 classes, the best recognition rate of 95.53% (±0.48) was achieved by the soft assignment with the 2048 codebook. This result is statistically significant and about 6.4% better than the result reported in [31], where a recognition rate of 89.10% (±0.71) was reported on the same classes with pixel intensity values and Bernoulli HMM. Fig. 13 shows the confusion matrix of the tested samples for the configuration that gives the best result (soft quantization and codebook of size 2048). The lowest accuracy was for the NOON (ن). This is attributed to

the large variety of the writing style of this letter (Fig. 14). The two letters RAA (ر) and WAW (و) have similar writing styles. We noticed that the confusion between these two classes constitutes about 36.5% of the total misclassification errors. Other misclassified samples have challenging writing styles that made them similar to other classes. In Fig. 14, we show samples of such challenging styles.

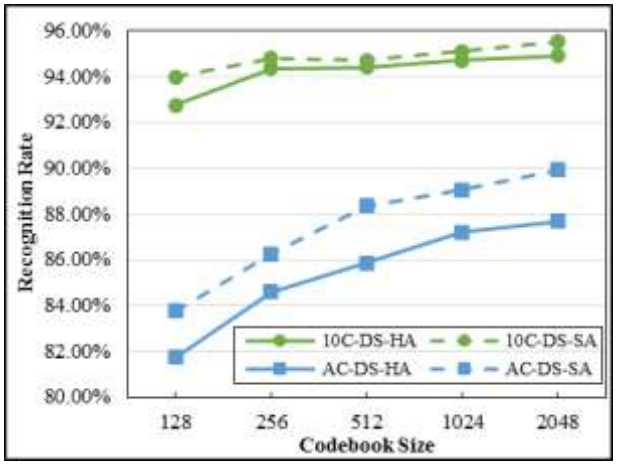


Fig. 12 Recognition rates for the non-touching subwords dataset

10C = The Most frequent 10 classes, AC = All Classes

	Predicted Class										Recognition Rate
	ا	ر	ل	ن	و	لا	ف	يا	نـ	مُقط	
ا	499	15	2	0	2	2	0	1	0	0	95.78%
ر	4	804	2	8	39	2	0	0	0	0	93.60%
ل	5	4	466	6	0	4	1	2	2	0	95.10%
ن	0	7	7	366	2	5	4	3	2	0	92.42%
و	0	50	2	0	722	5	0	0	2	0	92.45%
لا	0	2	5	1	2	710	2	1	0	1	98.07%
ف	0	0	2	2	0	3	297	0	0	0	97.70%
يا	1	0	1	2	0	5	0	518	2	0	97.92%
نـ	0	0	0	12	1	4	0	3	421	1	95.25%
مُقط	0	0	0	0	0	0	0	0	1	409	99.76%
										Average	**95.53%**

Fig. 13 Confusion matrix of the tested samples of the most frequent 10 classes of the non-touching subwords dataset

No.	Image	Actual Class	Predicted Class	No.	Image	Actual Class	Predicted Class
1		ا	ر	11		و	ل
2		ا	و	12		لا	ل
3		ر	و	13		لا	ف
4		ر	ا	14		لا	يا
5		ل	ا	15		ف	مُقط
6		ن	نـ	16		ف	ن
7		ن	ر	17		يا	لا
8		ن	لا	18		نـ	ن
9		ن	و	19		نـ	و
10		و	ر	20		مُقط	نـ

Fig. 14 Examples of misclassified subwords samples in the most frequent 10 classes of the non-touching subwords dataset

For the complete subwords dataset, the best accuracy we achieved was 89.93% (±0.56) with the Soft Assignment and 2048 codewords. These results are encouraging on such a challenging dataset containing a large number of classes, with many classes of few samples. This is still better and statistically significant than the recognition accuracy reported in [31] with the most frequent 10 classes. Fig. 15 shows the per-class accuracy using Soft Assignment and 2048 codewords. The main source

of error is attributed to the lack of training samples, as the classes with less than five training samples (e.g., classes #2, 17, 29, 36, and 61 ((ـتـ), (ـسـ), (ـنـ), (ـشـ), and (ـصـ)) have poor performance. The classes having similar writing style, e.g., classes #4 and 5 ((د) and (ر)), classes #6, 8 and 10 ((ـفـ), (و) and (ن)) and classes #40 and 41 ((عشر) and (نيف)) have large confusion. Class# 8 (ن) has several writing styles, so samples from other classes with unusual writing styles are sometimes assigned to this class. Fig. 16 shows examples of challenging samples. Despite that, we have achieved high accuracy (above 90%) in many classes containing several hundred test samples, like Classes #1, 19, 20, 23 and 42. Some samples in these classes have complicated writing styles which are hard for humans to recognize without context (Fig. 17).

No	#Samples	Correctly Classified	Mis-classified	Accuracy	No	#Samples	Correctly Classified	Mis-classified	Accuracy	No	#Samples	Correctly Classified	Mis-classified	Accuracy
1	829	807	22	97.35%	24	4	0	4	0.00%	47	28	15	13	53.57%
2	2	0	2	0.00%	25	20	13	7	65.00%	48	39	21	18	53.85%
3	4	2	2	50.00%	26	1	1	0	100.00%	49	1	0	1	0.00%
4	8	3	5	37.50%	27	61	52	9	85.25%	50	10	3	7	30.00%
5	859	797	62	92.78%	28	37	29	8	78.38%	51	35	28	7	80.00%
6	244	217	27	88.93%	29	1	0	1	0.00%	52	2	0	2	0.00%
7	490	459	31	93.67%	30	147	138	9	93.88%	53	26	9	17	34.62%
8	396	341	55	86.11%	31	121	100	21	82.64%	54	18	14	4	77.78%
9	103	83	20	80.58%	32	22	19	3	86.36%	55	5	0	5	0.00%
10	792	721	71	91.04%	33	5	2	3	40.00%	56	116	106	10	91.38%
11	5	1	4	20.00%	34	2	0	2	0.00%	57	55	48	7	87.27%
12	330	278	52	84.24%	35	4	0	4	0.00%	58	38	24	14	63.16%
13	3	1	2	33.33%	36	1	0	1	0.00%	59	33	21	12	63.64%
14	82	31	51	37.80%	37	47	44	3	93.62%	60	36	31	5	86.11%
15	36	24	12	66.67%	38	27	21	6	77.78%	61	1	0	1	0.00%
16	31	24	7	77.42%	39	11	8	3	72.73%	62	28	20	8	71.43%
17	1	0	1	0.00%	40	198	189	9	95.45%	63	15	8	7	53.33%
18	30	25	5	83.33%	41	442	397	45	89.82%	64	2	0	2	0.00%
19	724	706	18	97.51%	42	410	406	4	99.02%	65	29	24	5	82.76%
20	304	291	13	95.72%	43	49	40	9	81.63%	66	80	77	3	96.25%
21	83	76	7	91.57%	44	5	1	4	20.00%	67	4	0	4	0.00%
22	25	13	12	52.00%	45	1	0	1	0.00%	68	40	32	8	80.00%
23	529	508	21	96.03%	46	4	0	4	0.00%	69	1	0	1	0.00%
Total Samples: 8172; Correctly Classified: 7349; Misclassified: 823; Average Accuracy: 89.93%														

Fig. 15 Per-class accuracy achieved by soft assignment and 2048 codewords on the complete subwords dataset

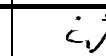
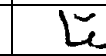
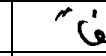
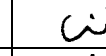
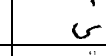
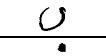
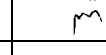
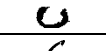
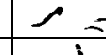
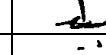
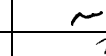
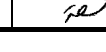
No.	Image	Actual Class	Predicted Class	No.	Image	Actual Class	Predicted Class
1		ـتـ	ن	11		نتة	نية
2		د	ر	12		ثنا	لغا
3		د	و	13		لفا	لف
4		ر	و	14		تنى	ئة
5		و	ر	15		نما	ن
6		ن	ف	16		تنين	نيف
7		ف	ن	17		نيف	عشر
8		ي	نيف	18		فقط	لا
9		ين	ن	19		مئة	ن
10		ين	ن	20		تسعة	سبعة

Fig. 16 Examples of misclassified subwords samples of the complete subwords dataset

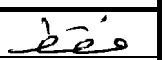
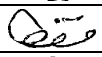
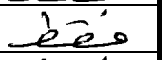
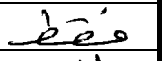
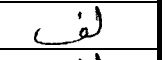
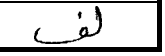
No.	Image	Class	No.	Image	Class
1			6		
2			7		
3			8		
4			9		
5			10		

Fig. 17 Examples of challenging samples that were correctly recognized

C. *Experimental Evaluation of the Proposed Approaches*

Based on the modifications presented in Section II, we have implemented two novel versions of SIFT, named *unsignedSIFT* and *binarySIFT*. *UnsignedSIFT* smooths the input image by a Gaussian kernel proportional to the descriptor spatial area similar to the one used by the VLFeat *phow* command. Gradient magnitudes and orientations are computed using the same procedures as VLFeat *dsift* command. However, instead of generating 8 bins in each of the 4×4 spatial regions, the contributions of the corresponding symmetric orientation bins are accumulated, giving 4 bins per region. Accordingly, *unsignedSIFT* produces a 64-D descriptor for the input patch. In *binarySIFT*, the Gaussian smoothing step is ignored and the gradient magnitude and orientation are computed for the binary image using the lookup tables shown in Fig. 4. Similar to *UnsignedSIFT*, the contributions of the corresponding symmetric orientation bins within each spatial region are accumulated, giving a 64-D descriptor for the patch. The two versions are implemented based on the open-source VLFeat library by modifying *dsift* command, which is an efficient implementation for extracting dense SIFT descriptors for gray-scale images.

To evaluate the performance of the novel versions, we integrated them with our feature learning framework and conducted several experiments on the non-touching Arabic Indian digit and non-touching Arabic subwords datasets. In these experiments, we used dense sampling with four patch sizes (16, 24, 32 and 40 pixels) and a stride of 2 pixels in the four grids. The original SIFT as well as the two novel versions were applied in the description step. The obtained descriptors are de-correlated by applying PCA. Five codebooks of sizes 128, 256, 512, 1024 and 2048 are generated by k-means clustering, and the hard assignment is utilized in the quantization step. Fig. 18 compares the recognition accuracies of the three SIFT versions.

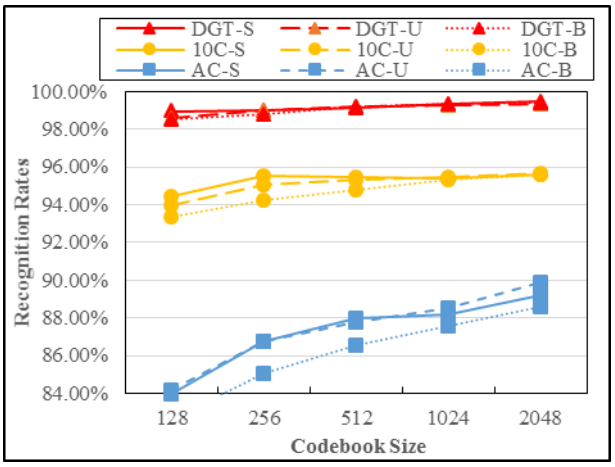


Fig. 18 Comparing the Recognition accuracies of the three versions of SIFT
DGT = Digits dataset, 10C = Most frequent 10 classes, AC = All Classes
S = Original SIFT, U = UnsignedSIFT, B = BinarySIFT

The results show that *unsignedSIFT* and *binarySIFT* achieve comparable performance to the original SIFT, although their descriptors have only 64 elements. The two versions achieved better performance with larger codebooks. In addition to the promising recognition accuracies, the two versions take less time to compute. Due to their lower dimensionality, the clustering and quantization are faster than those generated by the original SIFT. Table 4 shows the CPU times of computing the descriptors of a digit sample image (*dg003082.tif*) using the original SIFT and the two novel versions. We show also the CPU times of clustering three sets, each with one million random descriptors generated by one of the three versions into 1024 clusters. The two versions achieved up to 2.2X speedup in description generation and clustering steps, which indicates that utilizing the characteristics of the handwritten text reduces the computational overhead.

TABLE 4 CPU TIME (IN MILLISECONDS) ELAPSED IN COMPUTING THE DESCRIPTORS OF A SAMPLE IMAGE AND IN CLUSTERING 1 MILLION DESCRIPTORS INTO 1024 CLUSTERS

	SIFT	**UnsignedSIFT**	**BinarySIFT**
Descriptors Computation	15.51	8.88	6.92
Clustering	507.13×10^3	341.07×10^3	356.37×10^3

V. CONCLUSIONS

We presented, in this paper, a feature-learning framework for Arabic handwritten text recognition based on Bag-of-Feature (BoF) paradigm. Several alternatives involved in the framework are investigated, and their effects in the learned features are evaluated. We used the Harris detector, Harris-Laplace detector and dense sampling. The selected regions are described using SIFT that produces 128-d descriptors for each region. The codebook is built by applying k-means clustering on sample vectors selected randomly from the training set. The feature vector of a given sample is obtained by quantizing its reduced SIFT descriptors using the codebook and constructing the normalized histogram of size equal to the codebook size. Two quantization approaches are evaluated, viz. hard and soft assignment. SVM is used in the classification phase. The system is evaluated on the non-touching Arabic Indian digits and Arabic subwords datasets of CENPARMI database.

The extensive experiments show that dense sampling outperforms the interest point detectors for handwritten text recognition. Codebooks with large sizes had a positive effect on the performance in all datasets. The best accuracies are achieved using a codebook of size 2048. Soft assignment reduces the side effect of quantization distortion associated with the hard assignment and improves the performance of the learned features. The developed framework achieved 99.34% recognition accuracy on the non-touching Arabic/Indian digit dataset, 95.53% recognition accuracy on the 10 most frequent classes of the non-touching Arabic subwords dataset and 89.93% on the full non-touching Arabic subwords dataset. All these results outperformed the state-of-the-art published work on the same datasets. The main source of errors in both datasets is the challenging writing styles. The two datasets contain samples written with complicated styles that are difficult even for human recognition without context. The non-touching subwords dataset lacks enough training samples for some classes. Classes containing less than 10 training samples achieved poor performance. The accuracy of the 10 most frequent classes of this dataset support this argument. Despite that, our system achieved 100% accuracy on 6 digit classes, and successfully tolerated common problems encountered in published works such as imperfect and disconnected samples. On the full subwords dataset, our system achieved accuracy above 90% in many classes containing several hundred test samples, though some samples in these classes have complicated writing styles.

Utilizing the characteristics of the handwritten text images resulted in two novel versions of the SIFT algorithm that are computationally efficient and produce descriptors of half the size of the original SIFT. The two versions have achieved comparable recognition performance to SIFT on the non-touching Arabic Indian digits and Arabic subwords datasets.

As future work, we will investigate improving the feature learning framework we developed in this work by adapting novel low-level features other than SIFT to utilize the characteristics of Arabic text. We intend to adapt our framework to segmentation-free open-vocabulary Arabic handwritten text recognition.

ACKNOWLEDGEMENT

The authors would like to acknowledge the support provided by King Fahd University of Petroleum & Minerals (KFUPM) for funding this work through project number RG 1313-1/2. The first author is supported by Hadhramout Establishment for Human Development, Yemen Graduate Scholarship. The authors would also like to thank the anonymous reviewers whose comments helped in improving the paper.

REFERENCES

[1] M. T. Parvez and S. A. Mahmoud, "Offline arabic handwritten text recognition: A Survey," *ACM Comput. Surv.*, vol. 45, no. 2, pp. 1-35, 2013.

[2] Y. Bengio, A. Courville, and P. Vincent, "Representation Learning: A Review and New Perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798-1828, 2013.

[3] A. AbdulKader, "A Two-Tier Arabic Offline Handwriting Recognition Based on Conditional Joining Rules," in *Arabic and Chinese Handwriting Recognition*, Springer Berlin Heidelberg, 2008, pp. 70-81.

[4] Y. LeCun, L'. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278-2324, 1998.

[5] R. Anil, K. Manjusha, S. S. Kumar, and K. Soman, "Convolutional Neural Networks for the Recognition of Malayalam Characters," in *Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA)*, vol. 247, pp. 493-500, 2014.

[6] A. Graves and J. Schmidhuber, "Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 21, pp. 545-552, 2009.

[7] T. Bluche, J. Louradour, M. Knibbe, B. Moysset, M. F. Benzeghiba, and C. Kermorvant, "The A2iA Arabic Handwritten Text

Recognition System at the Open HaRT2013 Evaluation," *11th IAPR Int. Work. Doc. Anal. Syst.*, pp. 161-165, 2014.

[8] U. Porwal, Yingbo Zhou, and V. Govindaraju, "Handwritten Arabic text recognition using Deep Belief Networks," in *21st International Conference on Pattern Recognition (ICPR)*, 2012, pp. 302-305.

[9] P. P. Roy, Y. Chherawala, and M. Cheriet, "Deep-Belief-Network Based Rescoring Approach for Handwritten Word Recognition," in *2014 14th International Conference on Frontiers in Handwriting Recognition*, 2014, pp. 506-511.

[10] S. O'Hara and B. A. Draper, "Introduction to the Bag of Features Paradigm for Image Classification and Retrieval," *arXiv Prepr. arXiv1101.3354*, 2011.

[11] G. Csurka, R. Dance, Christopher, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Workshop on statistical learning in computer vision (ECCV)*, 2004, pp. 1-22.

[12] Sivic and Zisserman, "Video Google: a text retrieval approach to object matching in videos," in *Proceedings Ninth IEEE International Conference on Computer Vision*, vol. 2, pp. 1470-1477, 2003.

[13] L. Rothacker, S. Vajda, and G. a. Fink, "Bag-of-Features Representations for Offline Handwriting Recognition Applied to Arabic Script," in *2012 International Conference on Frontiers in Handwriting Recognition*, 2012, pp. 149-154.

[14] M. Rusinol, D. Aldavert, R. Toledo, and J. Llados, "Browsing Heterogeneous Document Collections by a Segmentation-Free Word Spotting Method," in *2011 International Conference on Document Analysis and Recognition*, 2011, pp. 63-67.

[15] S. Fiel and R. Sablatnig, "Writer Identification and Writer Retrieval Using the Fisher Vector on Visual Vocabularies," in *2013 12th International Conference on Document Analysis and Recognition (ICDAR)*, 2013, pp. 545-549.

[16] Y. Al-Ohali, M. Cheriet, and C. Suen, "Databases for recognition of handwritten Arabic cheques," *Pattern Recognit.*, vol. 36, pp. 111-121, 2004.

[17] T. Tuytelaars and K. Mikolajczyk, "Local Invariant Feature Detectors: A Survey," *Found. Trends® Comput. Graph. Vis.*, vol. 3, no. 3, pp. 177-280, 2007.

[18] E. Nowak, F. Jurie, and B. Triggs, "Sampling Strategies for Bag-of-Features Image Classification," in *ECCV*, vol. 3954, 2006, pp. 490-503.

[19] K. Mikolajczyk and C. Schmid, "Performance evaluation of local descriptors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 10, pp. 1615-30, 2005.

[20] J. Hu, X. Peng, and C. Fu, "A comparison of feature description algorithms," *Opt. - Int. J. Light Electron Opt.*, vol. 126, no. 2, pp. 274-278, 2015.

[21] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91-110, Nov. 2004.

[22] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-Up Robust Features (SURF)," *Comput. Vis. Image Underst.*, vol. 110, no. 3, pp. 346-359, 2008.

[23] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua, "BRIEF: Computing a Local Binary Descriptor Very Fast," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 7, pp. 1281-1298, 2012.

[24] X. Yang and K. T. T. Cheng, "Local difference binary for ultrafast and distinctive feature description," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 1, pp. 188-194, 2014.

[25] S. A. Mahmoud and W. G. Al-Khatib, "Recognition of Arabic (Indian) bank check digits using log-gabor filters," *Appl. Intell.*, vol. 35, no. 3, pp. 445-456, 2010.

[26] S. A. Mahmoud, "Recognition of Arabic (Indian) check digits using Spatial Gabor filters," in *5th IEEE-GCC Conference & Exhibition*, 2009, pp. 1-5.

[27] A. Vedaldi and B. Fulkerson, "VLFeat: An open and portable library of computer vision algorithms," in *18th ACM international conference on Multimedia*, 2010, pp. 1469-1472.

[28] S. Awaida and S. A. Mahmoud, "Automatic Check Digits Recognition for Arabic Using Multi-Scale Features, HMM and SVM Classifiers," *Br. J. Math. Comput. Sci.*, vol. 4, no. 17, pp. 2521-2535, 2014.

[29] V. Romero, A. Giménez, and A. Juan, "Explicit Modelling of Invariances in Bernoulli Mixtures for Binary Images," in *IBPRIA '07 Proceedings of the 3rd Iberian conference on Pattern Recognition and Image Analysis*, 2007, vol. 4477, pp. 539-546.

[30] A. Giménez, J. Andrés-Ferrer, A. Juan, and N. Serrano, "Discriminative Bernoulli Mixture Models for Handwritten Digit Recognition," in *2011 International Conference on Document Analysis and Recognition*, 2011, pp. 558-562.

[31] A. G. Pastor, "Bernoulli HMMs for Handwritten Text Recognition," Ph.D. Thesis, Polytechnic University of Valencia, Spain, 2014.

Mohammed O. Assayony received the MS degree in computer science from the Universiti Sains Malaysia in 2009 and he is currently pursuing his Ph.D. in computer science and engineering at King Fahd University of Petroleum & Minerals, Saudi Arabia. His research interests include Arabic Document Analysis and Recognition, Automatic Feature Learning for Arabic handwriting as well as parallel algorithms for scientific applications.

Sabri A. Mahmoud is a Professor of computer science in Information & Computer Science Department, King Fahd University of Petroleum & Minerals, Saudi Arabia. His research interests include Arabic Document Analysis and Recognition, Arabic NLP, Image Analysis and applications of Pattern Recognition. Dr. Mahmoud is a life senior member of IEEE. He published over 90 papers in refereed journals and conference proceedings.