Feature set evaluation for offline handwriting recognition systems: Application to the recurrent neural network model

Youssouf Chherawala, Partha Pratim Roy, and Mohamed Cheriet, Senior Member, IEEE

Abstract—The performance of handwriting recognition systems is dependent on the features extracted from the word image. A large body of features exists in the literature, but no method has yet been proposed to identify the most promising of these, other than a straightforward comparison based on the recognition rate. In this paper, we propose a framework for feature set evaluation based on a collaborative setting. We use a weighted vote combination of recurrent neural network (RNN) classifiers, each trained with a particular feature set. This combination is modeled in a probabilistic framework as a mixture model and two methods for weight estimation are described. The main contribution of this work is to quantify the importance of feature sets through the combination weights, which reflect their strength and complementarity. We chose the RNN classifier because of its state-of-the-art performance. Also, we provide the first feature set benchmark for this classifier. We evaluated several feature sets on the IFN/ENIT and RIMES databases of Arabic and Latin script respectively. The resulting combination model is competitive with state-of-the-art systems.

Index Terms—Feature set evaluation, word recognition, system combination, recurrent neural network, IFN/ENIT, RIMES.

I. INTRODUCTION

THE recognition of handwritten text is a challenging task, owing to the huge variation in writing styles of individual writers. Recognition systems mimic human reading by processing a text line image as a sequence of vertical frames. Features are extracted from each frame and fed to a system which convert them into a string of symbolic characters. In spite of extensive research with hidden Markov models (HMM) and hybrid neural network-HMM models for sequential data transcription [1], [2], [3], feature extraction remains a challenge.

The goal of features is to remove unnecessary variability, in the form of individual writing style, from a word image, and keep only the information relevant for recognition. Their use goes from word-spotting [4], [5], [6], [7], [8], [9], [10], where the goal is to retrieve specific keywords within a document, to word recognition [11], [12], [13], [3], [14], [15], [16], where a document is converted into a symbolic character string. Nevertheless, feature design [17], [18], [19], [20], [21] for handwritten word shape is a difficult task, because the requirements for good features of word images cannot be explicitly defined (i.e., by a set of rules) when the word image is degraded or the handwriting shows a large variability.

For this reason, a large body of features exists in the literature for handwriting recognition in Latin and Arabic scripts [22], [23], [24]. The search for better feature sets is far from over not only for handwriting recognition but also for the related tasks of image recognition, retrieval and annotation [25], [26], [27], [28], [29]. Existing features are based on models devised in various fields, such as pattern recognition, computer vision, and machine learning. Because of their different backgrounds, it is very difficult to compare them on a theoretical basis. Moreover, they are often used on different databases, with different protocols and recognition systems. This makes it difficult to decide which features should be used for a new application. The literature does not provide clear guidelines on relevant features and no principled feature design has emerged. Although existing features certainly make a significant contribution in their respective fields (computer vision, machine learning, etc.), that contribution is not clear in the context of handwriting recognition. Therefore, it is important to evaluate existing features first, and then identify the most promising ones. This could be achieved by comparing the recognition rate of classifiers trained with different feature sets. However, this approach only provides a partial insight into the features as it ignores their complementarity. What is needed are efficient tools for feature set evaluation, able to quantify their strength and complementarity, in order to guide the design of the next generation of features.

In this paper, we propose a framework for feature set evaluation in handwriting recognition. Feature sets are indirectly evaluated by means of a recurrent neural network (RNN) word classifiers. A classifier is trained for each feature set, and votes for its recognized word, then, all the votes are gathered using a weighted vote scheme (Figure 1). This scheme is given a probabilistic interpretation; the true word distribution is modeled as a mixture of classifier–specific word distributions in which the weights are optimized. Under this model, the weights are the prior probabilities that the true word label comes from a given classifier and provide insight into the corresponding feature sets strength and complementarity. We evaluated a total of five feature sets, including handcrafted features, which are designed based on expert knowledge, and automatically learned features based on machine learning models. Specifically, we considered the following categories of features: pixel distribution, concavity, direction distribution, and automatically learned.
The main contribution of this work is to provide a feature set evaluation framework quantifying the relative importance of each feature set within a combination model. The resulting combination system shows performance comparable with state-of-the-art recognition systems. This reflects the relevance of the estimated weights but it is not the main focus of this work. As a part of our feature set evaluation, we provide the first feature benchmark using the state-of-the-art RNN model. The RNN outperformed the classic HMM on several handwriting tasks [30], [31], however no feature benchmark is available yet for this recently proposed model.

This paper is an extension of the work published in [32]. In particular, that extension includes feature set evaluation based on classifier combination, and the use of a complete reference system, with the integration of the token passing algorithm. The experimental section has also been significantly improved by considering two databases to test our framework.

The rest of the paper is organized as follows. Related work is reviewed in Section II. We detail our classifier combination approach and the weight optimization procedure respectively in Section III and Section IV. Then, we describe the RNN recognition system in Section V and the evaluated feature sets in Section VI. The experimental setup is given in Section VII followed by the feature set evaluation in Section VIII. We study the impact of the estimated weights on several combination rules in Section IX and we compare our combination approach with other systems in Section X. Finally, we apply our methodology to HMMs in Section XI.

II. RELATED WORK

A. Handcrafted features

One way to design features is to benefit from expert knowledge. In this case, features are handcrafted by experts in the field based on their knowledge and experience. Handcrafted features exhibit the word shape structure, and combine the shape geometry and topology. However, these shape properties are difficult to capture explicitly, therefore, they are expressed as a count of specific patterns or through the spatial distribution of foreground pixels [17]. Pixel distribution features characterize the density of these pixels in an image frame [33]. These features typically relate to the number of foreground pixels, the number of foreground-to-background transition and to the lower and upper word shape profile. They capture the presence of ascenders and descenders in the word image, which are important cues for correct word recognition. For Arabic word shapes, the geometry is often extracted through concavity features, which provide stroke direction and concavity information [33], [34]. These are computed with a hit-or-miss transform, based on morphological patterns. Also, recent advances in computer vision have produced efficient direction distribution descriptors, such as SIFT [35], SURF [36], and HOG [37], which are based on local histograms of gradient orientation. These descriptors have inspired new features for word shape. For example, Rothacker et al. built bag-of-word features from SIFT descriptors [38]. These descriptors have also been adapted to the specificity of word images for word-spotting applications [39], [40].

B. Automatic feature extraction

A popular alternative to handcrafted features is the use of dimensionality reduction methods [41] for automatic feature extraction. In such settings, new features are extracted either in a supervised fashion (using the target label information) or an unsupervised one. Principal component analysis (PCA) performs linear dimensionality reduction, and is among the most popular unsupervised feature extraction methods. Nonlinear methods, such as kernel PCA [42] and autoencoder neural networks [43], can explain nonlinear dependencies among the input variables.

Feature extraction is also performed in a supervised fashion, where the target recognition task has a direct influence on the extraction process. This is typically the case in the Multi-Layer Perceptron (MLP) neural network. The output of each hidden layer consists of features extracted by a nonlinear combination of the features of the previous layer, and the weights of the combination are learned during the training phase. For handwriting recognition, however, the MLP lacks the ability to deal with unsegmented data, unlike HMM. Therefore, it has been associated with the HMM in the so-called hybrid neural network/HMM system, where the HMM observation probabilities are based on the output of the MLP, instead of the classical Gaussian mixture model. This idea has been extended to tandem systems, where the MLP is used as a feature extraction module [44], [45]. The training of the tandem system involves several steps. First, the word slices are given the label of their characters, either manually or by using
a previously trained HMM in forced alignment mode. Then, the MLP is trained to recognize the label of the image slices without feature extraction. Finally, the output of the MLP followed by dimensionality reduction is considered as the extracted features for a new HMM model. Another approach is based on vision and image recognition, where neural networks are given a specific architecture to emulate the behavior of the visual cortex. In convolutional neural networks [46], the weights act as local image filters and produce multiple feature maps at each layer. Each feature map is a 2D image, produced in two steps. First, the output of the previous layer is convolved with a set of weights, and then it is usually subsampled with max-pooling. The activation of the feature maps of the first layer typically corresponds to the image edges. When multiple hidden layers are stacked – forming a deep neural network – a hierarchy of more and more abstract features is created. This architecture has been combined with RNN [47] and provides an alternative model for automatic feature extraction from temporal data.

C. Feature evaluation and combination strategies

Feature evaluation has been proposed for handwritten numeral recognition [48] based on the features class separation and recognition capabilities. However, this approach is not applicable to word recognition classifiers because of the large number of classes (words). Another natural way to evaluate features is through the combination of classifiers trained with a single feature. But for the same reason, the combination methods proposed for OCR based on a Bayesian approach [49] or the Dempster-Shafer theory [50] are not applicable. Therefore, classifier combination approaches for word recognition are more straightforward, with a focus on the overall performance instead of the class specific performance.

Such a system has been proposed in [51] for feature evaluation. Each classifier is trained with a single feature set, and the performance of a feature set is based on the recognition rate of the classifier combination at the decision level. Nevertheless, this approach doesn’t measure the individual contribution of each feature set.

Several other combination methods exist in the literature [52]. The simplest one is the plurality vote, where each classifier votes for a word hypothesis, and the one with the largest number of votes is selected. One of its variants is the sum rule, where the vote of each classifier is weighted by its confidence. One drawback of the sum rule is that the confidence of the classifier must be well scaled for good performance. Other famous approaches are based on ranking, in which each classifier provides an N-best list. The Borda count method selects the word candidate with the highest average rank. However, none of these methods provides an evaluation of the classifier. Re-ranking methods have been proposed in [33], [53], where an MLP is trained to select the true word hypothesis, given the confidence of various classifiers over their N-best list as input. Unfortunately, the MLP is not an explicit model and can’t be used to evaluate the relative strength of the base classifiers. In [54], the voting weight of each classifier is learned in a supervised scheme, such an approach can be derived to evaluate the classifiers explicitly.

III. CLASSIFIER COMBINATION

In this section, we present our strategy for classifier combination. Given \( N \) feature sets \( F_i \), \( i \in [1..N] \), we train \( N \) classifiers \( C_i \), each with a given feature set, that is, \( C_i \) is trained with \( F_i \). We assume without loss of generality that the output of the classifiers takes values from a lexicon \( L \).

A. Weighted vote

We combine the classifiers using the weighted vote approach, where each \( C_i \) votes for its recognized word hypothesis for a given word image \( x \). The word \( \hat{w} \) with the highest number of votes is selected as the final recognition:

\[
\hat{w} = \arg \max_{w \in L} \left( \sum_{i=1}^{N} \alpha_i \cdot P_i (w) \right)
\]  

(1)

where \( \alpha_i \) is the weight assigned to \( C_i \) and \( P_i (\cdot) \) is an indicator function which takes the value 1 if its argument is equal to the word recognized by \( C_i \), otherwise the value 0. This approach differ from the plurality vote, which is a special case where all the weights are equal. Carefully chosen weights can improve the performance of the combination system, and at the same time provide insight on the strength of a classifier \( C_i \), as stronger classifiers would have larger weights. In this framework, the only difference between the classifiers is the feature sets, and we relate the weights directly to the feature sets for their evaluation. In the following, we formulate the weighted vote in a probabilistic framework and describe two methods to optimize the weights.

B. Mixture model

The weighted vote classification can be reformulated in a probabilistic framework. For this, we first show that \( P_i (\cdot) \) is a probability mass function over \( L \). From its definition:

\[
P_i (w) \geq 0, \forall w \in L
\]  

(2)

Because all words of the lexicon are different, \( P_i (\cdot) \) takes the value 1 only for the word recognized by \( C_i \) and 0 for all others. Therefore:

\[
\sum_{w \in L} P_i (w) = 1
\]  

(3)

Hence \( P_i (\cdot) \) is a probability mass function, and will also be noted \( P_i (\cdot|x) \) when the input image \( x \) needs to be explicit. The classifiers \( C_i \) are based on different feature sets, and therefore, provide complementary views of \( x \). A better estimate of the true word can be obtained by combining the decision of all the \( C_i \). The true probability \( P \) can be approximated by a mixture model \( \hat{P} \), which is a linear combination of the \( P_i \), called mixture components, with specific bounds on the mixture weights:

\[
\hat{P} (w) = \sum_{i=1}^{N} \alpha_i \cdot P_i (w)
\]  

(4)
where \( \alpha_i \) is the mixture weight assigned to \( P_i \), such that:
\[
\sum \alpha_i = 1 \text{ and } \alpha_i \geq 0, \forall i
\]  
(5)

The weights form a probability distribution where each \( \alpha_i \) is the prior probability that the true word label is given by the component \( P_i \). Therefore, under this model, the weights have a clear interpretation. Classification can be carried out with the same procedure as in Eq. 1. The main challenge is to optimize the weights in order to maximize the recognition rate; it will be studied in the next section.

IV. WEIGHT OPTIMIZATION

We provide two optimization procedures to estimate the mixture weights. The first one is a straightforward least squares optimization (LS), while the second is an empirical estimation (EE). Before discussing them in detail we give some preliminary notations. The weights domain, as defined by Eq. 5, is the standard \((N-1)\) simplex denoted \( \Delta^{N-1} \). Let \( X = \{x_1, \ldots, x_K\} \) be a set of \( K \) word images and \( W = \{w^1_1, \ldots, w^K_1\} \) the corresponding ground truth. Together they form the training dataset \( D = \{X, W\} \) that is used for the weight optimization.

A. Least squares optimization

The mixture model (Eq. 4) can be directly optimized. Given a word image \( x \), and its ground truth \( w^* \), we want \( \hat{P}(w^*) \) to approximate the true probability \( P(w^*) = 1 \). In fact, we want to find the weights that perform the best approximation not only for a specific pair \( \{x, w^*\} \), but for the whole training dataset \( D \). We rewrite the right hand side of Eq. 4 with matrix notation. Let \( \alpha = [\alpha_1, \ldots, \alpha_N]^T \) be the weight vector, \( A \) a \( K \times N \) matrix such that each entry takes the following value: \( A_{ji} = P_i(w^*_j|x_j) \). The \( j \)th entry of the vector \( A \cdot \alpha \) correspond to \( \hat{P}(w^*_j) \), for \( j \in \{1, K\} \). Finally, \( b \) is a column vector of length \( K \) with all entries equal to 1, corresponding to the target probability of each \( w^*_j \). Therefore, we want to approximate \( b \) by \( A \cdot \alpha \) in the least squares sense by optimizing the weights over \( \Delta^{N-1} \):
\[
\hat{\alpha} = \arg \min_{\alpha \in \Delta^{N-1}} \| A \cdot \alpha - b \|^2
\]  
(6)

This is a linear least squares problem with linear and nonnegativity constraints [55]. We assume that \( N \ll K \) so that the problem is over determined, which is a typical case in practice. This formulation has many advantages. First, \( \hat{\alpha} \) is a global minimizer because the feasible region is convex. Second, if the rank of \( A \) is \( N \), the objective function is strictly convex and, therefore, the minimizer is unique. This condition corresponds to the case where the functions \( P_i(w^*|x) \) are linearly independent for \( \{x, w^*\} \in D \). The minimizer can be found using a numerical solver or the gradient-descent algorithm proposed in the Appendix. We make an important distinction between two types of data \( \{x, w^*\} \):

- type 1 - the probability of the true word is the same for all the classifiers:
  \[
P_i(w^*|x) = c, c \in [0, 1], \forall i \in [1..N]
\]

- type 2 - the probability of the true word is not the same for all the classifiers:
  \[
\exists (i, j) \in [1..N], P_i(w^*|x) \neq P_j(w^*|x)
\]

For type 1 data, any weights will lead to the same result, so they don’t play any role in the optimization. They includes the case where all the classifiers outputs are wrong or when they all output \( w^* \). For type 2 data, some of the classifiers are rights and some are wrong, and the optimization should favor the correct ones by giving them larger weights. These data are essential for the optimization.

B. Empirical estimation

We propose a second method to estimate the weights based on an empirical approach. It is motivated by the following shortcoming of the LS method, which finds a solution fitted to the behavior of \( \{C_i\} \) on a particular dataset \( D \). Nevertheless, \( D \) is not necessarily representative of the true data distribution \( \mathcal{D} \) due to the limited data available in practice. Moreover, if all the classifiers are competitive over \( D \), the amount of type 2 data will be very small, potentially leading to generalization problem over a test dataset. Therefore, we propose an empirical weight estimation method based on the following idea. The optimal \( \alpha \) for the (unknown) distribution \( \mathcal{D} \) must provide good classification results for \( D \), although not necessarily optimal. In this method, we test several hypotheses for \( \alpha \), and provide an estimation of the true \( \alpha \) based on the most efficient candidates. To generate the weights hypotheses, we sample \( \alpha \) from the Dirichlet distribution \( \text{Dir}(\beta) \), where \( \beta \) is a parameter vector with entries \( \beta_i \) of positive real numbers. It is a distribution over the domain \( \Delta^{N-1} \):
\[
f(\alpha, \beta) = \frac{1}{B(\beta)} \prod_{i=1}^{N} \alpha_{i}^{\beta_{i}-1}
\]  
(7)

where \( B(\beta) \) is a normalization constant. We set all \( \beta_i \) to 1 in order to sample uniformly over \( \Delta^{N-1} \), without favoring any weight configuration. Given a set \( \{\alpha^j\} \) of \( M \) weight vectors sampled from \( \text{Dir}(\beta) \), we compute the recognition rate using Eq. 1 for each \( \alpha^j \) (\( j \in [1..M] \)) over \( D \). Then, we sort \( \{\alpha^j\} \) in descending order based on their corresponding recognition rate and only keep the top \( m \) weight vectors whose indices form the set \( \text{Top-}m \). The estimation of the true weight vector is finally given by the empirical expected value of the top \( m \) weight vectors:
\[
\hat{\alpha} = \frac{1}{m} \sum_{j \in \text{Top-}m} \alpha^j
\]  
(8)

V. RNN-BASED WORD CLASSIFIERS

We have chosen the recurrent neural network (RNN) as the reference recognition system for our framework for two reasons. First, RNNs have been shown to perform better than HMMs for several sequence-decoding problems, in particular handwriting recognition [56]. One of the possible reason is that RNNs are discriminative models, while standard HMMs are generative. Second, RNNs are able to seamlessly learn
features from the input image in a supervised fashion, which HMMs can’t. This makes the RNN a good representative for a system based on learned features. The RNN classifier is made up of two distinct neural networks. The first is the long short-term memory (LSTM) network, which can access a long range temporal context. The second is the connectionist temporal classification (CTC) output layer, which is able to transcribe unsegmented data.

The architecture of the system differs, depending on whether the features are handcrafted or learned. Handcrafted features are first extracted from the input image, and then fed to the LSTM neural network. Finally, the CTC decoding layer provides the recognized character sequence as output. For learned features, the input image is directly fed to a multidimensional LSTM (MDLSTM) neural network, and then to the CTC decoding layer. In fact, the MDLSTM network replaces both the handcrafted feature extraction module and the LSTM neural network of the handcrafted feature system. The architecture of the system for both types of features is illustrated in Figure 2. Below, we describe the core of our recognition system, that is, the LSTM and CTC layers. The MDLSTM layer is described in Subsection VI-D.

A. Long short-term memory (LSTM) layer

The LSTM layer is made up of nodes with a specific architecture called a memory block, which is capable of preserving contextual information over a long period of time. Each memory block contains a memory cell, and its interaction with the rest of the network is controlled by three multiplicative gates: an input gate, an output gate, and a forget gate. For example, if the input gate is closed, the block input has no influence on the memory cell. Similarly, the output gate has to be open, so that the rest of the network can access the cell activation. The forget gate scales the recurrent connection of the cell. The gate behavior is controlled by the rest of the network. For the specific task of handwriting recognition, the ‘past’ and ‘future’ contexts are necessary for better performance. Therefore, the bidirectional LSTM (BLSTM) layer is used, where one LSTM layer processes the feature sequence in the forward direction, while another layer processes it in the backward direction. The output of the two layers is combined at the next layer as a feature map. As with the convolutional neural network architecture, it is possible to have multiple forward and backward layers in each LSTM layer, as well as multiple feature maps at the output layer, and to stack multiple LSTM layers using max-pooling subsampling.

B. Connectionist temporal classification (CTC) layer

Usually, most RNNs require pre-segmented training data or postprocessing to transform their output into transcriptions. To avoid this process, the CTC output layer has been designed to label unsegmented sequences. This layer is trained to predict the probability $P(w|O)$ of an output character sequence, that is, a word $w$, given an input feature sequence $O$, making the training discriminative. The output activation function provides the probability of observing each character for each time of the sequence. Once the network is trained, the labeling of an input sequence $O$ involves a decoding process of the network output, where the word $\hat{w}$ from a lexicon $L$ that generates the most probable path $\pi$ is chosen:

$$\hat{w} = B \left( \arg \max_{\pi \in \bigcup_{w \in L} B^{-1}(w)} P(\pi|O) \right)$$

where $B$ is a function that maps a path on the network output to a word.

VI. Word features

In this section, we present the image feature sets evaluated for word recognition. We provide justification for their selection, and we detail their extraction procedure. They have been organized into four categories: pixel distribution, concavity, direction distribution and automatically learned feature sets. These categories have been chosen either because of their state-of-the-art performance (pixel distribution and concavity), or because they represent recent trends in feature design, inspired by computer vision and machine learning. The first three categories correspond to handcrafted features, and, when one of the feature set overlaps several categories, we assign it to the most relevant one. The handcrafted features are obtained by sliding a frame window horizontally over the word image and computing the features in each frame.

A. Pixel distribution features

Two pixel distribution feature sets are described here. They are both extracted in a column–wise fashion. The first feature set was proposed by Rath and Mannmaha (R-M feature set) for handwritten word–spotting in historical manuscripts [17]. Each word image is described as a sequence of 4D feature vectors: the upper and lower profiles, the projection profile, and the background–to–foreground transition profile. The minimum and maximum positions of the foreground pixels are considered as the lower and upper profiles. The projection
profile is the number of foreground pixels in the corresponding column. The number of transitions between the foreground and background pixels is used as the transition profile. In word-spotting, the features extracted from two word images are matched using Dynamic Time Warping for similarity measurement. This feature set is popular because it is simple and robust to image degradation.

The second feature set was proposed by Marti and Bunke [57] (M-B feature set), and has been used by many researchers for handwritten text recognition with an HMM. Nine features are computed from the foreground pixels in each image column. Three global features capture the fraction of foreground pixels, the center of gravity, and the second order moment. The remaining six local features are the position of the upper and lower profiles, the number of foreground–to-background transitions, the fraction of foreground pixels between the upper and lower profiles, and the gradient of the upper and lower profile with respect to the previous column, which provides dynamic information.

B. Concavity features

Azeem and Ahmed [34] proposed a set of concavity features (CCV feature set) for Arabic word images, which has proved to be effective for Arabic text recognition with an HMM. First, the stroke thickness is normalized to a 3-pixel width by a thinning operation followed by dilation. Then, the response of the normalized image to 8 directional morphological filters is computed, leading to 8 binary directional images. Vertical frames of 6-pixel width are then used to extract the features, with an overlap of 3 pixels between two consecutive frames. In each frame and for each directional image, the number of ‘1’ pixels, as well as the normalized gravitational center of these pixels, is extracted as feature. Therefore, the final feature set contains 16 features per frame. The original feature set also includes dynamic features (delta and acceleration), but we exclude them in our framework as we expect the LSTM network to capture the temporal dependencies.

C. Direction distribution features

Rodriguez and Perronin developed the LGH features [39] inspired by SIFT. The image is divided into overlapping frames. The frame is fitted vertically to the text region, and then divided into 4×4 regular cells. Next, a histogram of gradients (8 bins) is computed in each cell, and the final vector represents the concatenation of the 16 histograms, which results in a 128D feature vector for each frame. Finally, the feature vector is scaled to unit norm. Note that the construction of the LGH can be summarized in two steps: image filtering followed by local sum-pooling for subsampling. These steps are typical of vision-based features. The LGH features have been successfully applied to handwritten text word-spotting [39]. The frame width is optimized empirically, and the frame overlap is set to 3, similarly to the concavity features.

D. Automatically learned features

The automatically learned features are based on the MDLSTM neural network [47]. This network is a multidimensional extension of the LSTM network. In this setting, the multidimensional data are scanned as multiple 1D sequences, by setting the scanning directions and the priority of the dimensions during scanning. For example, in a 2D image, we can choose to scan forward along the x dimension and backward along the y dimension, with a higher priority for x than for y, so that, during the scan, the x index will be updated before the y index. Each hidden layer memory block has a recurrent connection with the memory blocks one step back, according to the scanning direction for every dimension. One such layer provides the network with the context along the scanning direction. As there are 4 possible directions in 2D images (i.e., forward x and y, backward x and forward y and so on), 4 layers are necessary to capture the full context (Figure 3). As with the LSTM layer, it is possible to have multiple layers scanning in the same direction, and to combine them to form multiple feature maps at the output layer. Moreover, a hierarchy of the MDLSTM layer can be built, with 2D subsampling between layers. Because of this architecture, specifically at the first layers (image filtering with MDLSTM layers followed by subsampling), the MDLSTM can also be considered as a vision-based feature set.

VII. EXPERIMENTAL SETUP

A. Databases

We used two databases for our experiments. The first is the IFN/ENIT database [58] of Arabic script, and the second is the RIMES database [59] of Latin script (Figure 5).

The IFN/ENIT database is composed of 32,492 images of Tunisian city and village names written by several hundred different writers. It is divided into five sets: A, B, C, D, and E. From each of the first 4 sets, we randomly chose 500 images as the validation set, and the remaining images as the training set. We used the set E for testing.

The RIMES database is composed of more than 12,000 mails written in French, all annotated at the word level. We used the 2009 version of the database, which is divided into training, validation, and test sets, containing 44,195, 7,542 and 7,464 images respectively. The images are in gray level, so we binarized them using the Otsu algorithm [60] for all the features, except for the MDLSTM model. The decision with respect to the MDLSTM model is justified by the results in [54], which show similar performance using binarized or gray-level images. Moreover, we kept the distinction between
Fig. 4. Character recognition error rate during neural network training for different features on the IFN/ENIT database (first row) and the RIMES database (second row). The best model on the validation set for each feature set is shown.

Fig. 5. Sample images from the experiment databases. (a) IFN/ENIT. (b) RIMES.

characters with and without accents, for example e, é, and è are considered as different characters.

B. Experimental protocol

For both the handcrafted and learned feature sets, the network architecture is made up of a hierarchy of three LSTM/MDLSTM layers. In Table I, we provide the details of each level of the hierarchy. The layers of the last level are directly fed to the CTC network. For the MDLSTM feature set, we use the same network architecture as in [47]. For further details, please refer to [47]. For all the networks, the learning rate has been set to $10^{-4}$ and a momentum of 0.9 has been used. The training stops after 20 iterations without improvement for the character level error rate on the validation set. The experiment is reproduced 5 times for each feature set, because the random initialization of the neural network during the training phase leads to different performances.

Therefore, we have 5 classifiers $C_{ij}$ for each feature set $F_i$, with $j \in [1..5]$.

Several frame widths have been tested for the LGH features, and we selected the one providing the best performance on the validation set. The selected values are 24 for IFN/ENIT and 16 for RIMES, and have been chosen among the set {8, 16, 24, 32}. During our experimentation on the IFN/ENIT database, we noticed that the training of the MDLSTM architecture was prone to be trapped into local minima due to the higher number of weights to learn. Therefore, we trained 15 networks with this architecture and selected the 5 best based on their performance on the validation set. We used the RNNLIB implementation of the recurrent neural network [61].

We distinguish here two types of classifier. The first type is trained with the complete training set, and it is used to evaluate the individual performance of each feature set. We refer to these classifiers as $C_{ij}^{complete}$. The second type is trained with 80% of the training set, the remaining 20% being used for the weights estimation that will be detailed in Section VIII. We refer to these classifiers as $C_{ij}^{partial}$.

<table>
<thead>
<tr>
<th>Handcrafted feature system architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hierarchy</strong></td>
</tr>
<tr>
<td>Input</td>
</tr>
<tr>
<td>Level 1</td>
</tr>
<tr>
<td>Level 2</td>
</tr>
<tr>
<td>Level 3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Automatic feature system architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hierarchy</strong></td>
</tr>
<tr>
<td>Input</td>
</tr>
<tr>
<td>Level 1</td>
</tr>
<tr>
<td>Level 2</td>
</tr>
<tr>
<td>Level 3</td>
</tr>
</tbody>
</table>
C. Neural network capacity

We verify that all the networks have enough capacity, that is, they have enough neurons to learn complex recognition models. The learning curve of the best $C^{\text{complete}}_{ij}$ for each feature set is shown in Figure 4. Note that all the networks have enough capacity, as the error on the training set keeps decreasing, even after the error on the validation set has reached its minimum.

VIII. FEATURE SET EVALUATION

A. Feature set ranking

We provide here a feature set benchmark solely based on the classifiers recognition rate. Table II shows the average recognition rate of the $C^{\text{complete}}_{ij}$ for each feature set. We note that the features ranking order is the same for both databases. The M-B feature set provides the best performance, closely followed by the LGH one. Their performance are around 1% and 2% higher respectively for IFN/ENIT and RIMES, over the MDLSTM feature set which is the third best. Finally, the R-M and CCV feature sets are ranked last.

<table>
<thead>
<tr>
<th>Feature set</th>
<th>IFN/ENIT</th>
<th>RIMES</th>
</tr>
</thead>
<tbody>
<tr>
<td>M-B</td>
<td>93.2 ± 0.5</td>
<td>90.1 ± 0.3</td>
</tr>
<tr>
<td>R-M</td>
<td>91.6 ± 0.4</td>
<td>88.3 ± 0.5</td>
</tr>
<tr>
<td>CCV</td>
<td>88.5 ± 0.6</td>
<td>87.6 ± 0.5</td>
</tr>
<tr>
<td>LGH</td>
<td>92.8 ± 0.2</td>
<td>90.1 ± 0.4</td>
</tr>
<tr>
<td>MDLSTM</td>
<td>91.9 ± 0.5</td>
<td>88.8 ± 1.0</td>
</tr>
</tbody>
</table>

B. Feature set weighting

First, we describe the protocol used for the weight estimation for both methods. For the LS method, the ‘lsqlin’ MATLAB function has been used for the optimization. For the EE method, we sampled $M = 10^3$ weight vectors from the Dirichlet distribution, and selected the top 1% of samples.

We report the performance of both the LS and EE weights on the data of type 2 of the test set. For this purpose, we randomly selected 3 classifiers for each feature set among the following sets: $\{C_{ij}^{\text{partial}}\}$ and $\{C_{ij}^{\text{complete}}\}$, and repeated the experiment 100 times to assess the statistical significance of the results. We equally distributed $\alpha_i$ to the selected $C_{ij}$. The amount of data of type 2 is approximately 30% of the complete test set. Table IV shows the recognition rate for a baseline vector of equal weights (EW), the LS and EE weights. First, we note that both the LS and EE approaches perform better than the EW weights on both databases and on both sets of classifiers. Although the LS and EE weights provide similar performances on RIMES, the LS weights perform better on the IFN/ENIT database for $\{C_{ij}^{\text{partial}}\}$. Therefore, we consider it a better estimate of the true weights and interpret them. On both databases, the LGH and M-B feature sets obtain the highest weights, while the R-M and CCV feature sets obtain one of the lowest weights. This is in accordance with the feature set ranking results. We note that the weight of the MDLSTM features is considerably less on the IFN/ENIT database than on RIMES. The results from both databases highlight that the M-B and LGH feature sets are the most prominent, and complementary to each others.

<table>
<thead>
<tr>
<th>Database</th>
<th>Classifiers</th>
<th>EW</th>
<th>LS</th>
<th>EE</th>
</tr>
</thead>
<tbody>
<tr>
<td>IFN/ENIT</td>
<td>$C_{ij}^{\text{partial}}$</td>
<td>89.9 ± 0.4</td>
<td>90.7 ± 0.4*†</td>
<td>90.3 ± 0.4*</td>
</tr>
<tr>
<td></td>
<td>$C_{ij}^{\text{complete}}$</td>
<td>89.9 ± 0.3</td>
<td>90.2 ± 0.4*</td>
<td>90.2 ± 0.4*</td>
</tr>
<tr>
<td>RIMES</td>
<td>$C_{ij}^{\text{partial}}$</td>
<td>88.3 ± 0.3</td>
<td>88.5 ± 0.3*</td>
<td>88.5 ± 0.3*</td>
</tr>
<tr>
<td></td>
<td>$C_{ij}^{\text{complete}}$</td>
<td>88.0 ± 0.3</td>
<td>88.3 ± 0.3*</td>
<td>88.2 ± 0.3*</td>
</tr>
</tbody>
</table>

We show in Table V the results of the combination of all the classifiers on all the test data and the relative improvement obtained with the estimated weights. They confirm that both the LS and EE weights are better than equal weights, and that the LS method is better than EE. As expected, the performance of the combination with EW of $\{C_{ij}^{\text{partial}}\}$ is less than that of $\{C_{ij}^{\text{complete}}\}$, because they were trained with less data. Unfortunately, even the LS weights cannot make it for this loss of performance. Therefore, transferring the weights estimated from $\{C_{ij}^{\text{partial}}\}$ to $\{C_{ij}^{\text{complete}}\}$ is the only way to achieve an overall gain. The advantage of this approach is to use all the training data for the classifiers, without introducing any bias in the weight estimation. In Figure 6, we show some images incorrectly recognized by all the $C_{ij}^{\text{complete}}$. They are characterized by slanted handwriting or ambiguous character shapes.

Additionally, we evaluate a weight estimation method based solely on the individual performance (IP weights) of each feature set. The weight of each feature set is equal to the


### TABLE V

<table>
<thead>
<tr>
<th>Classifier combination results and relative improvement with the estimated weights.</th>
</tr>
</thead>
<tbody>
<tr>
<td>IFN/ENIT</td>
</tr>
<tr>
<td>Classifier</td>
</tr>
<tr>
<td><strong>C\text{partial}_{ij}</strong></td>
</tr>
<tr>
<td><strong>C\text{complete}_{ij}</strong></td>
</tr>
</tbody>
</table>

### TABLE VI

<table>
<thead>
<tr>
<th>Feature set combination using IP weights. The recognition rate over the weight training set is also shown. (Best values highlighted.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IFN/ENIT</td>
</tr>
<tr>
<td>Feature set</td>
</tr>
<tr>
<td>M-B</td>
</tr>
<tr>
<td>R-M</td>
</tr>
<tr>
<td>CCV</td>
</tr>
<tr>
<td>LGH</td>
</tr>
<tr>
<td>MDLSTM</td>
</tr>
<tr>
<td>IP Comb.</td>
</tr>
</tbody>
</table>

### TABLE VII

<table>
<thead>
<tr>
<th>Comparison of the recognition rate (%) of different combination methods. (∗: Difference statistically significant under the t test.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IFN/ENIT</td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>Plurality vote</td>
</tr>
<tr>
<td>Sum rule</td>
</tr>
<tr>
<td>Max rule</td>
</tr>
</tbody>
</table>

### Table VIII

<table>
<thead>
<tr>
<th>Comparison of the recognition rate (%) with other systems. (Best values highlighted.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IFN/ENIT</td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>TUM MDLSTM [31]</td>
</tr>
<tr>
<td>Hybrid-HMMs/MDLSTM comb. [54]</td>
</tr>
<tr>
<td>HMMs comb. [53]</td>
</tr>
<tr>
<td>LSTM-HMM Tandem [62]</td>
</tr>
<tr>
<td>MDLSTM [47]</td>
</tr>
<tr>
<td>HMM [38]</td>
</tr>
<tr>
<td>Proposed method</td>
</tr>
</tbody>
</table>

### IX. Weights impact on combination rules

In this section, we study the influence of the weights for the sum rule and max rule in addition to the plurality vote. In the weighted version, the classifier confidence (path probability) or vote is weighted by \( \alpha \) before applying the rule. For this purpose, we randomly select 3 \( C\text{complete}_{ij} \) classifiers for each feature set, and combine them using the weights estimated by the LS method and repeat this process 100 time. We show in Table VII the average combination results. The weights provide a slight improvement for all the methods on both databases, with the exception of the max rule on the RIMES database. This further shows the relevance of the estimated weights. We note that the confidence–based methods (sum rule and max rule) don’t perform as well as the proposed weighted plurality vote.

### X. Comparison with other systems

We also compare our combination approach of all the \( C\text{complete}_{ij} \) (with the weights estimated by the LS method) against state–of–the–art systems (Table VIII). All of these are based on either the HMM or LSTM model, or both. The TUM MDLSTM [31] is the classic MDLSTM model, but with hyper parameters optimization (the size of hidden layers, etc.). The system in [54] is based on a weighted combination of 7 systems: one hybrid MLP-HMM, two tandem GMM-HMMs, and four MDLSTMs. The system in [53] combines HMMs with and without context–dependent models using an MLP. The system in [62] is based on a tandem LSTM-HMM with horizontal positioning normalization. The system in [47] is the original MDLSTM architecture. Finally, the HMM in [38] is based on the bag–of–features model. The results show that the proposed approach is competitive with state–of–the–art methods, and actually obtains the best recognition rate on the IFN/ENIT database. Also note that, unlike most of the other systems, our approach is based on a single recognition model (RNN), and that no preprocessing is performed at image level, except for binarization.

### XI. Application to the hidden Markov model

In this section, we test our framework using the HMM classifier. The main motivation is to know if the feature set weighting is classifier dependent. We evaluate the M-B, R-M,
CCV and LGH feature sets. We adopt the HMM architecture proposed in [34]. Each alphabet symbol is modeled by an HMM with the Bakis topology and 6 emitting states. In order to limit the computational complexity, a mixture of 32 Gaussians per state is used to model the emission probability. During our preliminary test, we obtained poor results with the M-B, R-M and LGH feature sets as described in Section VI. Therefore, we extract the M-B and R-M feature sets once every three columns, similarly to the CCV feature set. Also, the LGH features frame is divided into $2 \times 2$ regular cells. We obtain better results with the delta and acceleration features for the M-B, R-M, CCV feature sets and without for LGH.

The individual recognition results, the combination weights obtained by the LS optimization, as well as the combination results are shown in Table IX. We note that the best recognition result is obtained by the LGH feature set on both databases. This is reflected by its very large weight. The CCV feature set obtains the second largest weight, and the M-B and R-M feature sets have the smallest weights. Once again, we obtain better combination results using learned weights than equal weights. More precisely, the EW combination provides a worse performance than the best individual classifier (LGH), while the LS weights provide an equal performance. The very large weight of the LGH feature set (more than 0.5) makes it impossible for the other classifiers to overrule its classifier results, and avoids the drop of performance. Nevertheless, the IP weights estimation method is able to improve the combination performance (93.9% and 82.1% respectively for the IFN/ENIT and RIMES databases). This suggests that some form of regularization is needed for the LS optimization method for the task of classifier combination and it will be investigated in future work.

Overall, the feature set ranking based on the recognition rate and the combination weights are different than those obtained using the RNN classifier. Therefore, the feature set weighting is classifier dependent, but we believe this is mainly due to the limitation of some classifiers to properly exploit the input features. For example, there are many parameters to manually adjust for HMMs, such as the topology, the number of states, the frame window shift etc. Unlike the HMM, the RNN classifier has shown the capability to perform well without excessive parameter tuning, which makes it a better candidate to evaluate the complementarity of different feature sets. Finally, as noted by one reviewer, testing our framework on more classifiers will help determine the classifier-independent weights.

### Table IX

<table>
<thead>
<tr>
<th>Feature Set</th>
<th>IFN/ENIT (%)</th>
<th>RIMES (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>M-B</td>
<td>88.5</td>
<td>8.9</td>
</tr>
<tr>
<td>R-M</td>
<td>89.0</td>
<td>14.7</td>
</tr>
<tr>
<td>CCV</td>
<td>97.8</td>
<td>26.0</td>
</tr>
<tr>
<td>EW Comb</td>
<td>92.3</td>
<td>80.7</td>
</tr>
<tr>
<td>LS Comb</td>
<td>93.6</td>
<td>80.8</td>
</tr>
</tbody>
</table>

**Conclusion**

Features are a crucial component of handwriting recognition systems. A large body of feature sets is available in the literature, but no method is capable of quantifying both their efficiency and complementary nature. To fill this need, we used a weighted combination scheme and put it into a probabilistic framework. We tested five feature sets from four different categories: pixel distribution, concavity, direction distribution, and automatically learned. The results on Arabic and Latin word databases show that the Marti-Bunkes and the LGH feature sets are the most efficient, with good complementarity with each other. In future work, this framework will be applied to guide the design of novel features, and it will be extended to compare the nature of various recognition methods in terms of strength and complementarity, HMM with LSTM models, for example.

### Appendix

#### Gradient-descent LS optimization

We present here a gradient-descent algorithm for the least-squares optimization, as an alternative to numerical solvers. We follow the notations from Section IV and reformulate the objective function as:

$$F = \sum_{(x, w^*) \in D} \left( \sum_{i=1}^{N} \alpha_i \cdot P_i (w^* | x) - 1 \right)^2 \tag{10}$$

Algorithm 1 is the standard batch gradient-descent algorithm, with the difference that the solution $\alpha$ is projected back on $\Delta^{N-1}$ after each update (line 8). We used the fast simple algorithm proposed by Chen and Ye [63] for the projection operator $\pi_{\Delta}$. The algorithm will converge toward the global minimizer given a sufficiently small learning rate $\eta$.

**Algorithm 1** Gradient-descent LS optimization

**Input:** $N$ classifiers and the training database $D = \{X, W\}$

**Output:** Combination weight vector $\alpha$

**Parameters:** Learning rate $\eta$

1. $\alpha \leftarrow [1/N, \ldots, 1/N]^T$
2. repeat
3. $\nabla F \leftarrow 0$
4. for all $\{x, w^*\}$ in $D$
5. $A \leftarrow [P_1 (w^* | x), \ldots, P_N (w^* | x)]$
6. $\nabla F \leftarrow 2A^T (A \cdot \alpha - 1) + \nabla F$
7. end for
8. $\alpha \leftarrow \pi_{\Delta} (\alpha - \eta \nabla F)$
9. until convergence

### Acknowledgment

The authors thank the NSERC and SSHRC of Canada for their financial support and the reviewers for their constructive and helpful suggestions.

REFERENCES


Youssouf Chherawala received his M.Sc. and his Ph.D. degrees from the École de Technologie Supérieure (University of Quebec) in 2007 and 2013, respectively. In 2014, he was a postdoctoral researcher at the Synchromedia Laboratory for Multimedia Communication in Telepresence. Also, he worked on several industrial projects across different countries. His research interests include Pattern Recognition, Machine Learning, Shape Analysis and Handwriting Recognition.

Partha Pratim Roy received his Ph.D. degree in computer science under the supervision of Dr. Josep Lladós (Universitat Autònoma de Barcelona) and Dr. Umapada Pal (Indian Statistical Institute) in 2010 from the Universitat Autònoma de Barcelona, Barcelona (Spain). He worked as a postdoctoral research fellow in the Computer Science Laboratory (LI, RFAI group), France (2010-2012) and in Synchromedia Lab, Canada (2013). He has qualified for the ‘Assistant Professor’ position in France in 2012. Presently, Dr. Roy is working as an Assistant Professor at the Indian Institute of Technology (IIT), Roorkee. His main research area is Pattern Recognition. Dr. Roy has participated in several national and international projects funded by the Spanish and French governments. In 2009, he won the best student paper award at the International Conference on Document Analysis and Recognition (ICDAR). He has gathered industrial experience while working as an Assistant System Engineer in TATA Consultancy Services (India) from 2003 to 2005 and as Chief Engineer in Samsung, Noida from 2013 to 2014.

Mohamed Cheriet (SM95) received the M.Sc. and Ph.D. degrees in computer science from the University Pierre et Marie Curie, in 1983 and 1988, respectively. Since 1992, he has been a Professor with the Automation Engineering Department, Ecole de Technologie Supérieure, University of Quebec, Montreal, where he was appointed as a Full Professor in 1998. He co-founded the Laboratory for Imagery, Vision and Artificial Intelligence, ÉTS, and was the Director from 2000 to 2006. He has been the Founder and Director of the Synchromedia Laboratory since 1998, which targets multimedia communication in telepresence applications. He is an Expert in computational intelligence, pattern recognition, mathematical modeling for image processing, cognitive and machine learning approaches, and perception. He acquired an extensive experience in cloud computing and network virtualization. In addition, he has authored over 350 technical papers in the field. He has co-authored a book entitled Character Recognition Systems: A Guide for Students and Practitioners (John Wiley and Sons, 2007). He is a recipient of the Queen Elizabeth II Diamond Jubilee Medal in light of his significant contributions to knowledge improvement in computational intelligence and mathematical modeling for image processing. He serves on the Editorial Boards of several renowned journals and international conferences. He holds the Tier 1 Canada Research Chair on Sustainable Smart Eco-Cloud. He was the Founder and Chair of the IEEE Montreal Chapter of Computational Intelligent Systems.