A PHOC Decoder for Lexicon-free Handwritten Word Recognition

Giorgos Sfikas, George Retsinas and Basilis Gatos

Computational Intelligence Laboratory, Institute of Informatics and Telecommunications
National Center for Scientific Research "Demokritos", GR-15310 Agia Paraskevi, Athens, Greece
{sfikas, georgeretsi, bgat}@iit.demokritos.gr

***Abstract*—In this paper, we propose a novel probabilistic model for lexicon-free handwriting recognition. Model inputs are word images encoded as Pyramidal Histogram Of Character (PHOC) vectors. PHOC vectors have been used as efficient attribute-based, multi-resolution representations of either text strings or word image contents. The proposed model formulates PHOC decoding as the problem of finding the most probable sequence of characters corresponding to the given PHOC. We model PHOC layers as Beta-distributed observations, linked to hidden states that correspond to character estimates. Characters are in turn linked to one another along a Markov chain, encoding language model information. The sequence of characters is estimated using the max-sum algorithm in a process that is akin to Viterbi decoding. Numerical experiments on the well-known George Washington database show competitive recognition results.**

I. Introduction

Two main families of methods dominate the field of unconstrained offline handwriting recognition: Hidden Markov [1], [2] and, more recently, Neural Network-based models [3], [4]. After preprocessing of the scanned input document image, the document is typically segmented into text areas up to the level of text line or word [5]. As a rule, methods of either family require large amounts of annotated training data, i.e. manually transcripted text lines and/or word images, in order to obtain an acceptable recognition rate.

Recently, Almazán et al. [6] have proposed a learning-based model that is suitable for the task of word spotting [7], as well as word recognition. An important merit of this model is that it requires only a (comparatively) moderate amount of training data [6]. The model encodes word images, as well as transcription strings, as fixed-length vectors that represent a set of attributes of the word. The encoding method has been named Pyramidal Histogram Of Characters (PHOC). The PHOC vector is a fixed-length, attribute-based representation [8] of a word. Each variate of the PHOC vector is related to a specific letter (unigram) or bigram and its relative position in the word. For example, the related attributes can be answers to questions like "does the word contain the letter 'd' "? or "does the word contain the bigram 'in' on its second half"? After estimating the PHOC vector for the word to be recognized, PHOC vectors are computed for all words in a pre-existing lexicon. The correct transcription is assigned after comparing functions of the PHOC vectors of the unknown word versus the representations of the lexicon words. Hence, a lexicon of possible words is necessary to perform recognition.

In this work, we propose a model that can decode PHOC vectors and produce an estimate of the true transcription without requiring a lexicon. The implication of this is that the word to be recognized is neither required to be part of the training set, nor part of a preset vocabulary of possible words. Therefore, with the proposed decoding model we can take advantage of the efficiency of recent methods that produce PHOC representations of word images [6], [9], [10] and use them to perform lexicon-free recognition.

PHOC generation is formulated as a hierarchical probabilistic model [11]. The observed PHOC estimate is modelled as the instance of a Beta-based probability density function that depends on the unobserved word transcription. Hence, decoding the input PHOC vector is formulated as finding the most probable word transcription given the observed PHOC. We propose and employ a suitable reparametrization and decomposition of the Beta-based emission model, so that it becomes tractable. The overall model is solved with a novel procedure that is based on the max-sum algorithm and is akin to Viterbi decoding [11]. Numerical results show that the proposed model has competitive recognition performance.

The remainder of this paper is organized as follows. In Section II we provide a brief overview of the structure of PHOC vectors and their use in the related literature. In Section III we present the proposed model and we show how to use it for word recognition in Section IV. In Section V we evaluate our model with numerical experiments and we conclude with a brief discussion of the paper's contribution and perspectives of future work in Section VI.

II. PHOC vector structure

The PHOC vector representation was introduced by Almazán et al. [6]. A PHOC vector is a fixed-length vector representation of a word. PHOCs can be computed exactly as binary histograms given a text string, or estimated as non-binary histograms given a word image [6], [9]. [1] We shall first

[1] In [6], the representation given a word image is referred to as an attribute vector and in [9] the same vector is referred to as a PHOC estimate. In the current work, we shall refer to representations given either a word image or a string as a PHOC vector or PHOC estimate.

examine the former case, as it is somewhat simpler.

A PHOC vector is defined as the concatenation of a smaller set of histograms of attributes, to which we shall refer to as T. Each of the histograms of T relates to either encoding information about unigrams or bigrams, respectively forming sets T_u and T_b (hence $T = T_u \cup T_b$). Also, each histogram in T encodes information about a specific part of the word, at a specific scale. In this respect, histograms in T can be grouped in layers. All histograms in layer x refer to the same scale and all together span the whole word in x same-length horizontal zones. For example, there are 3 histograms in layer 3, with each one encoding a different third of the word. In this work we represent histograms as $\phi_{X_{a.b}}$, where $X \in \{U, B\}$ and $a, b \in \mathbb{N}$. X corresponds to whether the histogram encodes information about unigrams or bigrams, located in space and scale in layer a, zone b. For example, $\phi_{U_{5.3}}$ is a histogram that encodes information about unigrams found in layer 5, horizontal zone 3, or more simply the third fifth of the word. All unigram or bigram histograms have the same length D_u or D_b, respectively. These histograms encode the presence or absence of all possible unigrams or bigrams.

As each variate of the PHOC vector is related to the presence or absence of a token, given an input text string each variate takes a binary value. An estimate can be computed given a word image, instead of a text string, as input. This vector differs to the PHOC representation given a text string, only to the domain of the vector variates. The representation vector variates are related to the same semantic information (i.e. existence of a unigram or bigram at a specific zone in the word). In that case, PHOC variates will in general be non-binary, as they are each directly or indirectly outputs of soft binary classifiers.

In this work, we shall assume the existence of a mechanism that can produce fixed-length PHOC vectors of dimensionality D, given either text strings or word images. The PHOC vector given a text string is assumed to be $\phi \in \{0, 1\}^D$, while the PHOC vector given a word image is assumed to be $\phi \in [0, 1]^D$. In the two following Sections we examine the proposed model and method to decode PHOCs, in order to produce the most likely word that generated them.

III. Proposed model

A. Overview

Given the PHOC vector estimate of an input word image, our goal is to use it to infer the related word text string. We formulate the process of PHOC generation as an hierarchical probabilistic graphical model [11]. The model defines two random processes, dependent one on the other. These are the word transcription to be estimated, denoted as γ, and the PHOC vector, denoted as ϕ. The PHOC vector ϕ is an observed variable, while the transcription is a latent variable (cf. fig. 1).

The word transcription γ is made up of an ordered set of M letters, and we can write $\gamma = \{\gamma_1, \cdots, \gamma_M\}$. We use a one-hot encoding to represent each letter γ_i. Specifically, for all $i \in [1, M]$, all variates of vector γ_i are zero except variate k, which is equal to one if γ_i represents the k^{th} letter in the alphabet of possible unigrams.

Fig. 1. A generic view of the proposed graphical model. γ stands for the word transcription to be estimated and ϕ stands for the observed PHOC vector. We aim to find the most probable transcription γ for the given encoding ϕ.

The PHOC vector ϕ consists of $|T| = |T_u| + |T_b|$ layers, where $|T_u|$ encode information about unigrams in the word and $|T_b|$ encode information about bigrams. Formally, ϕ is a concatenation of layer histograms $\{\phi_\tau | \tau \in T\}$. Model evidence can be written as

$$p(\phi, \gamma) = p(\phi|\gamma)p(\gamma) \tag{1}$$

Under this formulation, we aim to compute

$$\arg\max_{\gamma} p(\gamma|\phi) = \arg\max_{\gamma} p(\phi|\gamma)p(\gamma) \tag{2}$$

The function to be optimized is hence decomposable in two terms, namely the emission likelihood $p(\phi|\gamma)$ and the prior $p(\gamma)$ on possible transcriptions.

B. Emission likelihood

We model the emission likelihood $p(\phi|\gamma)$ as independent, identically distributed (i.i.d.) observations following the Beta probability distribution function (pdf). The Beta distribution, defined over $[0, 1]$, is often used to model binary events [11]. Formally the emission likelihood is written as

$$p(\phi|\gamma) = \prod_{\tau \in T} \prod_{d \in \tau} \beta(\phi_{\tau d} | \{\gamma_i^d | i \in \lambda(\tau)\}), \tag{3}$$

where $\phi_{\tau d}$ denotes the d^{th} variate of the PHOC layer histogram ϕ_τ, and $\beta(.)$ stands for the Beta distribution [11]. γ_i^d denotes the d^{th} binary variate of the one-hot representation of letter i. The function λ maps a layer histogram to the letter positions it relates to. For example, assuming a 6-letter word, layer histogram $\phi_{U_{3.2}}$ would encode information about the second third of the word, hence $\lambda(\phi_{U_{3.2}}) = \{3, 4\}$ (see also fig.2). In order to cover cases where layer zone limits "fall between" letters, we have used the convention of [6], assigning letters according to their letter-region area overlap.

If ϕ_τ is a histogram over unigrams, we have $\phi_\tau \in [0, 1]^{D_u}$. If it is a histogram over bigrams, $\phi_\tau \in [0, 1]^{D_b}$. The domain of the Beta distribution is over $[0, 1]$, and its parameters control the number of effective prior observations [11]. For more details on the Beta distribution and the parametrization used here, see the Appendix.

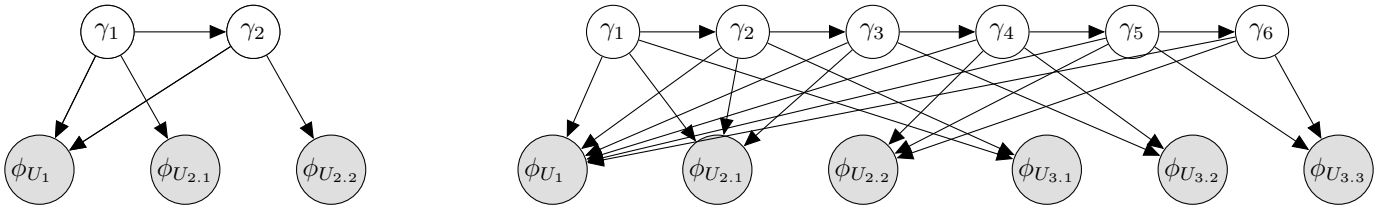

Fig. 2. An analytical view of two example use-cases of the proposed graphical model. PHOC size and word length size fixed to values suitable for the illustration of the model mechanism. (a) On the example on the left, the word length is fixed to a specific value ($\alpha = 2$, i.e. a word of two letters) and the observed PHOC vector only comprises 3 layers ($T = \{\phi_{U_1}, \phi_{U_{2.1}}, \phi_{U_{2.2}}\}$). Note that PHOC layer ϕ_{U_1} encodes information about the whole word (both letters), hence its value is conditioned on both letters γ_1, γ_2. Layers $\phi_{U_{2.1}}, \phi_{U_{2.2}}$ encode information each on a different half of the word (one letter each), hence they are conditioned respectively to letters γ_1, γ_2. (b) On the example on the right, we assume a larger word length ($\alpha = 6$) and PHOC size ($|T| = 6$).

We further decompose the Beta distribution terms into simpler terms (again, see the Appendix for details) that are conditioned only over single binary observations as

$$\beta(\phi_{\tau d}|\{\gamma_i^d|i \in \lambda(\tau)\}) = \hat{Z}(\{\gamma_i^d|i \in \lambda(\tau)\}) \prod_{i \in \lambda(\tau)} \beta(\phi_{\tau d}|\gamma_i^d), \tag{4}$$

where the $\hat{Z}(\cdot)$ function is defined in the Appendix. After combining eqs. (3) and (4) and dropping the normalization terms $\hat{Z}$, we get

$$\hat{p}(\phi|\gamma) = \prod_{\tau \in T} \prod_{d \in \tau} \prod_{i \in \lambda(\tau)} \beta(\phi_{\tau d}|\gamma_i^d). \tag{5}$$

In eq. (4), the normalization term $\hat{Z}$ depends only on the number of non-zero elements of the set $\{\gamma_i^d|i \in \lambda(\tau)\}$ and the set cardinal number, and not on the exact values of the set members *per se*. We shall assume that we can ignore the normalization term for the decoding step of the method, and use eq. (5) to approximate the emission likelihood during model solution. This choice comes with the benefit that the emission likelihood decouples in terms that depend only to one letter at a time. In terms of model optimization, this translates to a tractable and comparatively simple and fast decoding mechanism.

C. Prior on possible transcriptions

The prior on possible transcriptions $p(\gamma)$ is defined as a first-order Markov chain [11]. Formally this is written as a product of unigram and bigram state transition probabilities:

$$p(\gamma) = p(\gamma_1) \prod_{i=2}^{M} p(\gamma_i|\gamma_{i-1}) \tag{6}$$

Chain states represent possible letters for each letter position i. Hence, $p(\gamma)$ is in effect a language model prior. Unigram and bigram probabilities are computed offline on a suitable corpus and remain fixed throughout decoding.

IV. Decoding using the proposed model

A. Decoding given a specific word length

By combining equations 2, 5, 6 and taking logarithms, we can see that the objective we aim to optimize can be written as:

$$\gamma^\star = \arg\max_{\gamma} \sum_{\tau \in T} \sum_{d \in \tau} \sum_{i \in \lambda(\tau)} \ln \beta(\phi_{\tau d}|\gamma_i^d) + \\ + \sum_{i=2}^{M} \ln p(\gamma_i|\gamma_{i-1}) + \ln p(\gamma_1), \tag{7}$$

where we assume that word length M is known *a priori* (we shall examine how to estimate the true length M in Subsection IV-B). We note that all terms in the objective eq. (7) depend at most on two consecutive letters (this conveniently happens because we dropped the normalization terms in eq. 5). Hence, we can use a dynamic programming scheme to estimate $\gamma^\star$. In the context of graphical models, this can be performed using the max-sum algorithm [11].

In the max-sum algorithm, messages are passed between nodes of an extension of the basic graphical model. This extended graphical model is undirected, and includes all nodes of the original model as *variable* nodes plus one node for each term of the likelihood function, called *factor* nodes. The algorithm comprises a phase where *messages* are passed between variable and factor nodes. Factor nodes send messages to variable nodes, and vice versa. An example factor graph can be examined in fig. 3.

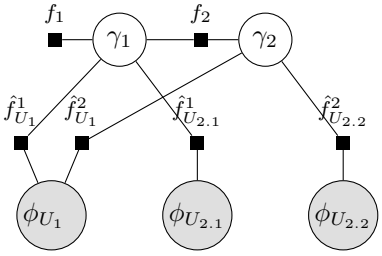

Fig. 3. Factor graph corresponding to graphical model presented in fig.2a.

For the proposed model, the messages to be passed can be computed as follows:

$$\mu_{f_1 \to \gamma_1}(\gamma_1) = \ln p(\gamma_1) \tag{8}$$

$$\mu_{\hat{f}^i_d \to \gamma_i}(\gamma_i) = \ln \beta(\phi_d|\gamma_i) = \ln \beta(\phi_d|\gamma_i^d) \tag{9}$$

$$\mu_{\gamma_i \to f_{i,i+1}}(\gamma_i) = \mu_{f_{i-1,i \to \gamma_i}}(\gamma_i) + \sum_d \mu_{\hat{f}^i_d \to \gamma_i}(\gamma_i) \tag{10}$$

$$\mu_{f_{i-1,i \to \gamma_i}}(\gamma_i) = \max_{\gamma_{i-1}} [\ln p(\gamma_i|\gamma_{i-1}) + \mu_{\gamma_{i-1} \to f_{i-1,i}}(\gamma_{i-1})] \tag{11}$$

where $\mu_{a\to b}$ stands for message sent from node a to node b. Factor nodes are denoted by letters f and $\hat{f}$. The nodes denoted by $f_{i,i-1}$ are the factor nodes that correspond to terms $p(\gamma_i|\gamma_{i-1})$, and f_1 corresponds to term $p(\gamma_1)$ of eq. (7). Nodes denoted by $\hat{f}_d^i$ correspond to the β factors of eq.(2). Note that in the above message-passing formulae we have intentionally omitted $\mu_{\phi_d\to\hat{f}_d}(\phi_d)$. Messages of that type could be modelled as δ Dirac pdfs centered on the observed values. We have instead chosen to combine these with $\mu_{\hat{f}_d^i\to\gamma_i}$ and present the combined message only, in order to simplify notation.

Messages are propagated from graph leaves towards the graph root, in our case the final letter γ_M. The first letter node γ_1 is the first to receive all incoming messages. After having done so, messages incoming to the next letter node γ_2 can be computed, and so on, until the final letter node γ_M is reached. The intuition behind this procedure is that, during each message passing, nodes propagate their "belief" about what letter is found at a particular word position. Emission nodes propagate their belief according to the PHOC observation; Markov chain nodes propagate their belief according to the language model. The end-result is a trade-off between these two factors.

While passing messages from node to node, we keep track of the maximizing per-node (i.e. per-letter position) value:

$$\phi(\gamma_i) = \arg\max_{\gamma_{i-1}}[\ln p(\gamma_i|\gamma_{i-1}) + \mu_{\gamma_{i-1}\to f_{i,i-1}}(\gamma_{i-1})] \quad (12)$$

After the last letter is reached, the optimal decoding can be computed. Optimal per-node letter values, starting from the end ($i = M$) and moving progressively back to the beginning ($i = 1$) can be computed using a back-tracking procedure [11]. First we compute the best value for the last letter as:

$$\gamma_M^\star = \arg\max_{\gamma_M}[\mu_{f_{M-1,M}\to\gamma_M}(\gamma_M) + \sum_d \mu_{\hat{f}_d^i\to\gamma_M}(\gamma_M)] \quad (13)$$

In order to compute best values for the rest of the letters ($\forall i \in [1, M-1]$) we use the $\phi(\cdot)$ function we defined earlier (eq. 12):

$$\gamma_i^\star = \phi(\gamma_{i+1}^\star) \quad (14)$$

Therefore, under these considerations the best estimate for the word transcription is computed as the word $\gamma_1^\star\gamma_2^\star\cdots\gamma_M^\star$.

B. Choosing the word length

In Subsection IV-A we have seen how to produce a word transcription estimate when the word length M is known. We propose a two-part score function $s_{wl}(\cdot)$ to estimate M for each word to be recognized. The first function component gives a coarse estimate $s_{wl}^C(\cdot)$ of M, which is refined when combined with the second function component $s_{wl}^F(\cdot)$.

For the coarse estimate, we use the word image length in pixels to estimate M. We perform linear regression over pairs of { word length in pixels, word length in letters } of the training set. The parameters of the straight line that is estimated, are used to project pixel word length to a mean estimate M^C. This regression is meant to give a first estimate that is assumed to be sufficiently close (i.e. up to a margin of 1-2 letters) to the actual word length. We proceed to define:

$$s_{wl}^C(M) = \begin{cases} 1, & \text{if } |M - \text{round}(M^C)| \le 2 \\ 0, & \text{otherwise} \end{cases} \quad (15)$$

In order to compute the refined word length estimate, we first decode the word using all non-zero s_{wl}^C score candidate word lengths. We re-encode these decoded character sequences, and perform L2-normalization on the resulting PHOCs. Subsequently we compute Euclidean distances of the normalized PHOC decodings to the normalized initial input PHOC estimate. We map these distances to similarity scores $s_{wl}^F(\cdot)$ using a standard Gaussian distribution kernel. We choose the final estimate $M^\star$ as

$$M^\star = \arg\max_M s_{wl}(M) \triangleq \arg\max_M s_{wl}^C(M) s_{wl}^F(M) \quad (16)$$

V. Numerical experiments

We have used the GW20 manuscript collection for numerical experiments [12]. The GW20 collection is comprised of 20 pages of text, handwritten by George Washington and his associates in the 18^{th} century. The manuscripts have been manually segmented into 4860 word images, fully transcribed. We have used two-fold and four-fold cross-validation to partition the set in training/validation and testing folds. At each case, all folds are of equal size, 10 and 5 pages respectively. Again in both cases, one fold is used as the testing fold and the rest for training and validation. We have trained the language model prior on the word transcription over unigrams and bigrams found in the respective training and validation folds only.

We have used the original attribute SVM-based model of Almazán et al. [6] to produce PHOC estimates of the test word images. As suggested in that work, we use a PHOC representation with unigram layers of levels 2, 3, 4, 5 and bigram layers of level 2. After having performed Platts scaling [6] on the attribute vector, the output vector variates are $\in [0, 1]$. The resulting PHOC vector is of dimensionality $D = 604$. Regarding the proposed decoding model, we have used a likelihood function that is defined over unigram emissions only, as preliminary results have shown that including messages from bigram emissions results in somewhat worse performance. We attribute this result to the structure of our model, which represents the latent process as a series of unigrams, each linked to emission nodes separately.

In fig. 4, we show decoding results for a number of words found in the GW20 dataset. We used results from the 2-fold GW20 setting. All of the selected words were found only in the test fold. In other words, no example of a same-transcription word image had been observed during the training phase. Yet, the proposed decoder succeeds at succesfully coping with such cases. This model characteristic is due to two factors: the zero-shot learning trait, related to the attribute-based character of the PHOC representation, and the fact that the proposed decoder does not depend on a lexicon of possible words, that might or might not include the target word.

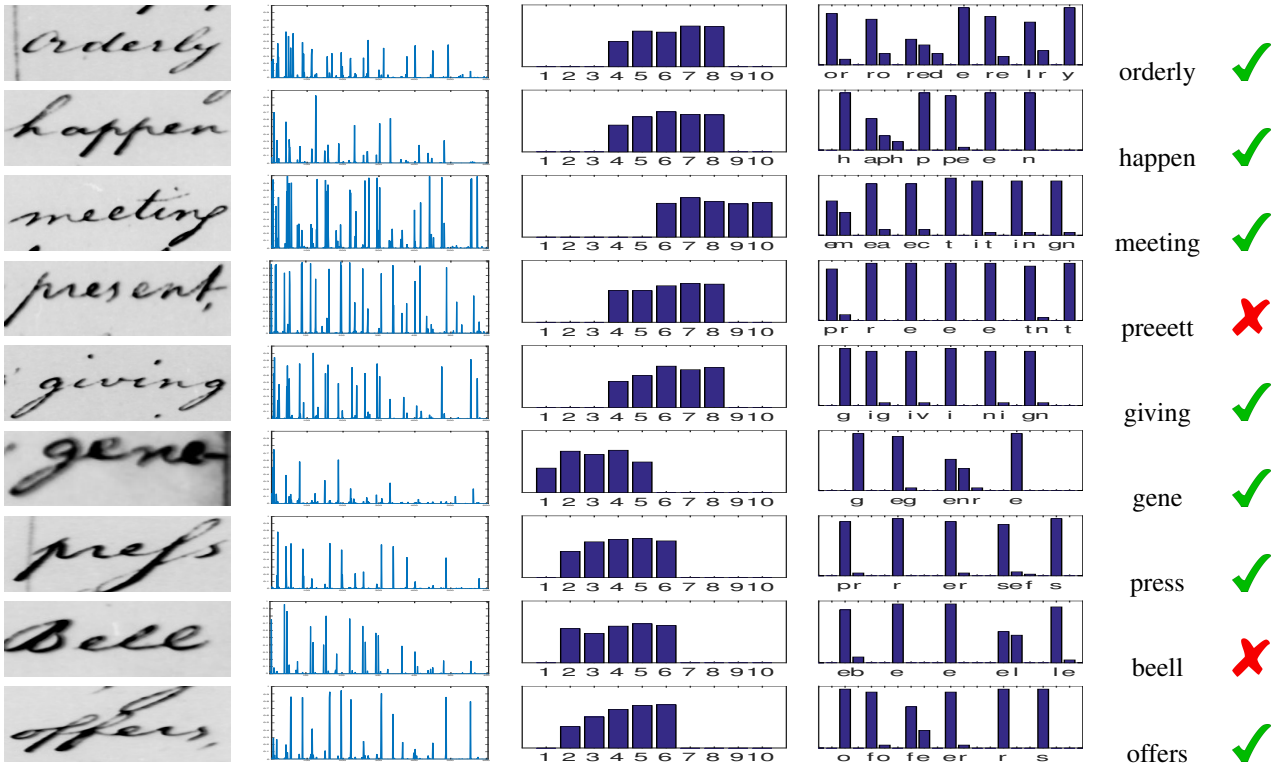

Fig. 4. Example decodings using the proposed model. On each row, from left to right, we see: The input word image, the observed PHOC vector shown as a signal of fixed length $D = 604$, the word-length estimator function s_{wl}, the most likely letters per sum of incoming messages per letter position, the decoded word. All depicted values on columns 2,3,4 are shown normalized to a scale $[0, 1]$. Per-letter position messages are combined with prior language model information to give the decoded end-result. Note that *no same-transcription examples were observed during the training phase*, for any of the words shown in this example.

We have compared our method with an HMM-based model [2] and a NN-based model [4]. The HMM-based model of Lavrenko et al. [2] uses a holistic representation to describe word images. Word transcriptions are modelled as the hidden states of a Markov chain, and the holistic representations correspond to the observed process. In the work of Frinken et al. [4], a neural network based model for handwriting recognition is employed, and show that the same model can be adapted for the task of keyword spotting. In both works, recognition results on GW20 are reported. We can examine the recognition accuracy of these works versus the accuracy of the current model in table I. Let us note that the compared HMM-based model of Lavrenko et al. [2] uses a language model that had been trained on a corpus of over 4 million words. The language model of the proposed decoder is trained on the available training folds only, i.e. in our case in less than $3,000$ words in all cases. Moreover, Lavrenko et al. use a fixed lexicon of words. The proposed decoder outperforms the compared methods, albeit being at a disadvantage by having no information about the search space in the form of a lexicon.

We also show results assuming a perfect word length estimate. This "cheat" scenario is presented for the purpose of decoupling the performance of the word decoding and the word length estimation components of the proposed model. Correct word length estimation is crucial for our model, as a wrong length estimate will lead to wrong recognition. Results show that there is room for improvement for word length estimation, as about 9% of recognition accuracy is "lost" due to word length estimation errors.

TABLE I
COMPARISON OF RECOGNITION ACCURACY OF THE PROPOSED DECODER VERSUS STATE-OF-THE-ART WORKS TESTED ON THE GEORGE WASHINGTON DATABASE. IN ORDER TO EVALUATE THE PROPOSED WORD LENGTH ESTIMATION SCHEME, RESULTS ARE ALSO REPORTED FOR THE HYPOTHETICAL ("CHEAT") SCENARIO WHERE WORD LENGTH IS KNOWN.

	Accuracy
GW - 2 folds	
Lavrenko et al. [2]	47.0%
Proposed decoder	**49.81%**
Proposed decoder + known word length	58.04%
GW - 4 folds	
Lavrenko et al. [2]	53.0%
Frinken et al. [4]	52.92%
Proposed decoder	**53.44%**
Proposed decoder + known word length	61.62%

VI. CONCLUSION AND FUTURE WORK

We have proposed and solved a novel probabilistic model that can be used for lexicon-free HTR. With the proposed model, PHOC vectors, that are the direct or indirect product of a number of recent important models [6], [9], [10], can be efficiently decoded. The output of the PHOC decoder is an

estimate of the transcription of the input word image. While PHOCs have been used for word recognition before, this is the first work where this is done without using a lexicon of possible words. After modelling observations as following a Beta-based distribution, we have shown how to reformulate it properly so that it can be tractable. The proposed decoder is finally solved with the max-sum algorithm.

In perspective, a straightforward extension to the current model would be to use PHOCs with a higher number of layers, instead of the numbers and types used in the work that introduced the PHOC representation [6]. More PHOC layers would entail an encoding that is more detailed and richer in information, hence we can conjecture that this could lead to better decoding accuracy. Related to this point, another perspective would be to use more advanced models for producing PHOCs. The recent models proposed in [9], [10] for example, compute PHOCs by utilizing deep learning techniques while retaining the efficiency, in terms of training set size, of the original attribute-based model [6]. Combining the PHOC output of either model with the proposed PHOC decoder would be entirely straightforward. Testing the model on more challenging datasets, comprising larger collections, or written using different scripts [9], [13] can be envisaged.

With respect to the model itself, future work may concern the more suitable integration of bigram information in the model, as PHOC bigram information is only partially exploited in the current model. Integration of the model with a word-level Markov chain can also be envisaged, as well as integrating a possible knowledge of lexicon words as a model prior.

APPENDIX: DEFINITION, PARAMETRIZATION AND DECOMPOSITION OF THE BETA DISTRIBUTION

The Beta distribution is defined over a continuous variable $x \in [0, 1]$. A standard parametrization often used in the literature [11] is over two parameters $a, b > 0$, with:

$$\beta(x|a, b) = \frac{\Gamma(a + b)}{\Gamma(a)\Gamma(b)} x^{a-1}(1 - x)^{b-1}. \tag{17}$$

where $\Gamma(\cdot)$ stands from the Gamma function. Parameters a, b can be interpreted as the effective prior number of observations at $x = 0$ and $x = 1$ respectively [11]. We can reparametrize eq. (17) as:

$$\beta(x|\delta) \triangleq \beta(x|\delta + 1, 2 - \delta) = 2x^{\delta-1}(1 - x)^{2-\delta}. \tag{18}$$

where the new parameter δ is $\in \{0, 1\}$. We can further generalize the above reparametrization, again with respect to the parametrization of eq. (17), as follows:

$$\beta(x|\Delta) \triangleq \beta(x|1 + \sum_{i=1}^{M} \delta_i, 1 + M - \sum_{i=1}^{M} \delta_i), \tag{19}$$

where Δ is a set of M binary variables $\{\delta_1, \cdots, \delta_M\}$.

The Beta pdf form of eq. (19) can be decomposed as a product of Beta pdfs of the form of eq. (18). This can be proved as follows:

$$\prod_{i=1}^{M} \beta(x|1 + \delta_i, 2 - \delta_i) \propto \prod_{i=1}^{M} x^{\delta_i}(1 - x)^{1-\delta_i} =$$

$$= x^{\sum_{i=1}^{M} \delta_i}(1 - x)^{M - \sum_{i=1}^{M} \delta_i} \propto$$

$$\propto \beta(x|1 + \sum_{i=1}^{M} \delta_i, 1 + M - \sum_{i=1}^{M} \delta_i) = \beta(x|\Delta), \tag{20}$$

or simply $\beta(x|\Delta) \propto \prod_{i=1}^{M} \beta(x|1 + \delta_i, 2 - \delta_i)$, which can be written as $\beta(x|\Delta) = \hat{Z}(\Delta) \prod_{i=1}^{M} \beta(x|\delta_i)$, where $\hat{Z}(\cdot)$ is a factor with a value independent of x. We can compute it by taking into account the normalizing factors that are implied in eq. (20). Thus, finally we have

$$\hat{Z}(\Delta) = \frac{M + 1}{2^M} \binom{M}{\sum_{i=1}^{M} \delta_i}.$$

ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation Programme (H2020-EINFRA-2014-2015) under grant agreement n°674943 (project READ).

REFERENCES

[1] M. Villegas, J. A. Sánchez, and E. Vidal, "Optical modelling and language modelling trade-off for handwritten text recognition," in *Proceedings of the 13^{th} International Conference on Document Analysis and Recognition (ICDAR)*, 2015, pp. 831–835.

[2] V. Lavrenko, T. M. Rath, and R. Manmatha, "Holistic word recognition for handwritten historical documents," in *Proceedings of the 1^{st} International Workshop on Document Image Analysis for Libraries*, 2004, pp. 278–287.

[3] G. Leifert, T. Strauß, T. Grüning, W. Wustlich, and R. Labahn, "Cells in multidimensional recurrent neural networks," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 3313–3349, 2016.

[4] V. Frinken, A. Fischer, R. Manmatha, and H. Bunke, "A novel word spotting method based on recurrent neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 2, pp. 211–224, 2012.

[5] M. Diem, F. Kleber, and R. Sablatnig, "Text line detection for heterogeneous documents," in *12th International Conference on Document Analysis and Recognition*, 2013, pp. 743–747.

[6] J. Almazán, A. Gordo, A. Fornés, and E. Valveny, "Word spotting and recognition with embedded attributes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 12, pp. 2552–2566, Dec 2014.

[7] A. P. Giotis, G. Sfikas, B. Gatos, and C. Nikou, "A survey of document image word spotting techniques," *Pattern Recognition*, vol. 68, pp. 310 – 332, 2017.

[8] C. H. Lampert, H. Nickisch, and S. Harmeling, "Attribute-based classification for zero-shot visual object categorization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 3, pp. 453–465, 2014.

[9] S. Sudholt and G. A. Fink, "PHOCNet: A deep convolutional neural network for word spotting in handwritten documents," in *Proceedings of the 15^{th} International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2016, pp. 277–282.

[10] P. Krishnan, K. Dutta, and C. V. Jawahar, "Deep feature embedding for accurate recognition and retrieval of handwritten text," in *Proceedings of the 15^{th} International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2016, pp. 289–294.

[11] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[12] A. Fischer, A. Keller, V. Frinken, and H. Bunke, "Lexicon-free handwritten word spotting using character HMMs," *Pattern Recognition Letters*, vol. 33, no. 7, pp. 934–942, 2012.

[13] G. Sfikas, A. P. Giotis, G. Louloudis, and B. Gatos, "Using attributes for word spotting and recognition in polytonic greek documents," in *Proceedings of the 13^{th} International Conference on Document Analysis and Recognition (ICDAR)*, 2015, pp. 686–690.