Towards Feature Learning for HMM-based Offline Handwriting Recognition

Nils Y. Hammerla[1], Thomas Plötz[2], Szilárd Vajda[1], Gernot A. Fink[1]
[1]*TU Dortmund, Computer Science Department, 44221, Dortmund, Germany*
[2]*Newcastle University, School of Computing Science, NE7 1NP, Newcastle upon Tyne, UK*
{*nils.hammerla,szilard.vajda,gernot.fink*}*@udo.edu*
t.ploetz@ncl.ac.uk

Abstract

Statistical modelling techniques for automatic reading systems substantially rely on the availability of compact and meaningful feature representations. State-of-the-art feature extraction for offline handwriting recognition is usually based on heuristic approaches that describe either basic geometric properties or statistical distributions of raw pixel values. Working well on average, still fundamental insights into the nature of handwriting are desired. In this paper we present a novel approach for the automatic extraction of appearance-based representations of offline handwriting data. Given the framework of deep belief networks – Restricted Boltzmann Machines – a two-stage method for feature learning and optimization is developed. Given two standard corpora of both Arabic and Roman handwriting data it is demonstrated across script boundaries, that automatically learned features achieve recognition results comparable to state-of-the-art handcrafted features. Given these promising results the potential of feature learning for future reading systems is discussed.

1 Introduction

Offline handwriting recognition (HWR) techniques rely, like other pattern recognition methods, on a compact representation of its high-dimensional input data - images of written script. The recognition capabilities of these reading systems are largely dependent on the extracted features, i.e., on measures that unveil the characteristics of the analyzed handwriting.

In domains like computer vision and speech recognition, research has led to theoretical insights that can be considered when designing appropriate feature representations. So far, comparable theoretic background knowledge regarding the (physical) nature of handwriting was insignificantly exploited. Instead, mainly heuristically handcrafted extraction procedures were proposed in the last few decades. Allowing impressive reading systems, still the desire for deeper fundamental insights remains, e.g., for features that generalize across script boundaries.

One possible way to circumvent this issue is to automatically *learn* a suitable feature representation directly from the data. The usage of these learned features has been proved to be appropriate for tasks like isolated digit recognition [6]. In contrast to fully holistic settings, where labels are readily available for each input-image and supervised methods can be applied easily, many state-of-the-art systems for the recognition of handwritten text rely on a segmentation-free approach [14]. Here the extracted, often tiny frames just show segments of characters and, consequently, lack a general semantic that would lead to proper labeling. Therefore, just the appearance of the frames is available for training, which is probably the reason why methods to automatically discover features are rare and mostly limited to the application of Principal Component Analysis (PCA).

This paper proposes to extend the automatic feature learning introduced by Hinton and proves its efficiency for more complex pattern recognition tasks like handwriting recognition. The extracted features resulting from a complex learning process involving no prior knowledge are integrated transparently in an Hidden Markov Model (HMM) based HWR framework. The method introduced in [15] is used to further improve the fully appearance based features learned initially by using labels that are obtained by aligning the trained HMMs on state-level.

The fully automatic feature learning method proposed here is evaluated on Arabic and Roman script using well-known benchmark datasets. A direct performance comparison is provided with a state-of-the-art HMM based system [14] that employs heuristic features as input for its recognizer. The comparable results achieved by this strategy show the great potential of such automatic feature extraction schemes and motivate their possible deployment on more complex recognition task in the future.

2 Related Work

In offline HWR static, two-dimensional images of handwritten text are processed to obtain a hypothesis about the text shown. Among the many methods used in state-of-the-art HWR systems, segmentation-free approaches that rely on Hidden Markov Models (HMMs) form a significant share [14]. In contrast to holistic approaches, here a given image is transformed into a sequence of observations using a sliding window.

2.1 Features for offline HWR systems

As stated by LeCun [7], better recognition systems can be built by relying more on automatic learning and less on hand-crafted heuristics. In a case study for isolated digit recognition the authors showed that those feature extraction techniques can be replaced by carefully designed learning machines operating only on pixels.

However, the classical features can be classified either as low-level pixel based analytical features or as high-level information, the so-called perceptual features. They heavily rely on the quality/precision of the extractors and the semantics behind them.

Analytical features like intensity, vertical and horizontal derivatives of densities [1], proportion of black pixels in a sliding window, gravity centers, second order moments, the number of black-white transitions, the fraction of black pixels between the upper and the lowermost black pixel [4], the pixels and their different neighbourhoods in a non-symmetric Markov Random Field scenario [17], or projection profiles [12] are among the most popular ones. The perceptual (structural) features being a higher level decription of the different scripts deal with more sophisticated measures like ascenders, descenders, diacritics and their relative position in the word, the estimated word length, horizontal, vertical strokes [3, 16] return points [9] or the analysis of convexities and concavities [2, 17], are some of the commonly used structural features.

The selection of a suitable feature set when designing an automatic recognition system is still a challenging issue and is, unfortunately, largely dependent on the script to be recognized.

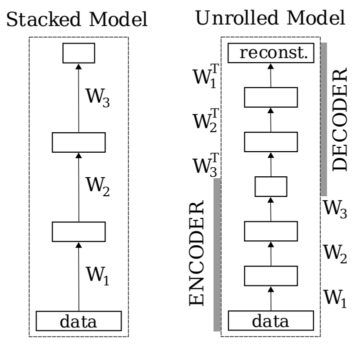


Figure 1. Two stages of autoencoder training. Left: trained deep generative model, right: unrolled, initialized autoencoder network.

2.2 Autoencoder networks

In the literature autoencoder networks have been described as a powerful tool for generic semi-supervised discovery of features. An autoencoder network is a feed-forward neural network that consists of one input layer, one output layer and an odd number of hidden layers. Both the input layer and the output layer share the same dimensionality and semantic. The innermost, *code*-layer is of much lower dimensionality and forms an intentional bottleneck. Each layer is fully connected to the subsequent one and usually a non-linear activation function is used. The autoencoder network transmits a low-dimensional representation of the input-vector through the bottleneck in order to reconstruct it at the output layer. Two parts can be identified, an *encoder* and a *decoder* (see figure 1).

Learning the parameters of such a deep network poses a challenge to common gradient based methods as described in [5]. In [6] it was proposed to learn the layers of an autoencoder network greedily from the bottom to the top by treating each pair of subsequent layers in the encoder as a Restricted Boltzmann Machine (RBM). In short an RBM is a fully connected bipartite two-layer graphical model able to generatively model binary data. One RBM is trained for each pair by treating the activation probabilities of the hidden units of one RBM as input-data for the next (stacking). The trained deep generative model is unrolled to obtain an initialized network. Further fine-tuning can be applied using common gradient based methods (for an overview see [7]). Figure 1 illustrates the deep generative model trained with

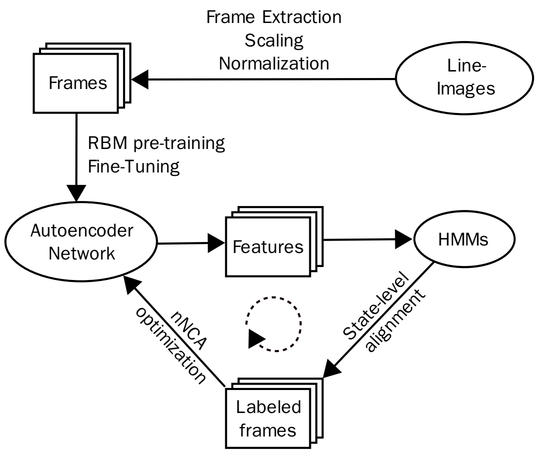


Figure 2. Overview of the feature learning and extraction process.

RBMs and how it is unrolled to obtain an initialized autoencoder network.

The trained network can be used to extract low-dimensional features for each input vector that are purely based on pixel information. In [15] a method is introduced, namely regularized non-linear Neighbourhood Component Analysis (reg. nNCA), that allows further improvement of the features, given that class-labels are available for at least a fraction of the input-data. By introducing constraints on parts of the code layer, gradient based methods can be used to explicitly encode class-related features and irrelevant transformations.

3 Feature Learning for HMM-based offline Handwriting Recognition

The feature learning and extraction process proposed here can be divided into two subsequent parts. In the initialization phase the extracted frames are used to train an autoencoder network. The features that are extracted with it are used to train a set of character HMMs in the subsequent optimization phase. The trained HMMs are used to obtain labels for each extracted frame which form the basis for a recursive feature improvement procedure. An overview of the general layout is shown in fig. 2.

3.1 Preprocessing

Autoencoder networks, as they are used here, can just process input-data of fixed dimensionality. The sliding window used to extract frames is therefore constrained to both a fixed height and width. Input images in offline HWR usually show a large variety in absolute height that is a result of variations in writing style and prior normalization steps (e.g. letter-height/-width normalization, ascender-length, etc). This variance has to be coped with by performing appropriate scaling of the absolute height.

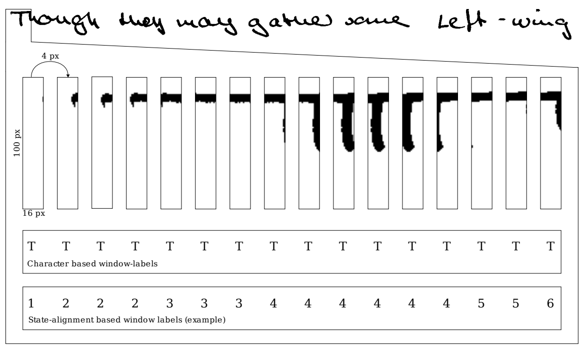
Typical scaling methods for word images rely on a segmentation of an image into different areas based on a number of estimated baselines. The different areas are fitted to a heuristic percentage, where less *important* areas are cropped or linearly scaled. In order to avoid the prior knowledge about the script that is implicitly used, here a novel scaling method is employed. The smoothed, cumulative transition histogram h^i (rising edges in a line) of an input-image x_i is projected onto the mean histogram h^M that is calculated on a training-set. For each line s of a new image (with the height of the sliding window) the corresponding value h_s^M is obtained. The line o of the original image whose value h_o^i lies closest to h_s^M is fitted into the new image at position s. The result is a scaled, centered word-image that is used during frame extraction.

3.2 Initialization

In the initialization phase the frames that are extracted from the preprocessed input-images are used to train an autoencoder network. The frames from an input-image form one *batch* that is processed as a whole throughout the training. Depending on the width of the image and the step-size chosen for the sliding window the number of frames in each batch varies from a few hundred to a couple of thousand.

After a deep generative model has been trained with the RBM pre-training it is unrolled to obtain an initialized network which is subject to further fine-tuning of the reconstruction error, as described in [6]. As a distance measure between a frame and its reconstruction the *cross-entropy-error* is minimized using Conjugate Gradients (CG) although other methods and loss-functions are possible.

The features extracted using the autoencoder network are purely based on the appearance of the frames. Therefore they are similar to the projection into the principal subspace, discovered by PCA. They do not fulfill the basic requirement to solely represent *relevant* variance since they focus on that which is largest in the input-space, missing a way to judge the relevancy. The great advantage of features extracted with an autoencoder network compared to those produced by PCA is that they can be further improved using label information as it is shown below.



Though they may gather some Left - wing
4 px
100 px
16 px
T T T T T T T T T T T T T T T T
Character based window-labels
1 2 2 2 3 3 3 4 4 4 4 4 4 5 5 6
State-alignment based window labels (example)

Figure 3. Character vs. state-based labels for a few extracted windows.

3.3 Optimization

When labels are given for each extracted frame, the autoencoder network trained in the initialization phase can be further improved using regularized nNCA as described in [15]. Although the method has been applied to other problems such as action recognition [10], the number of classes never exceeded the ten from the initial publication. Applied to offline HWR at least one class for each symbol has to be considered. For the case of Arabic handwriting, where some letters occur in up to 4 different shapes, the number of classes already exceeds 150. When more detailed classes are considered (as shown below) their number grows to several thousands. This paper describes the first attempt to apply reg. nNCA to a problem of such a complexity.

In general terms all benchmark databases for handwriting are accompanied by the reference transcription (ground-truth). In a segmentation-free context, the different images are split into a sequence of tiny frames that are unlikely to show a complete character. When each frame is labeled with the character of which it shows a small part, this information is not based on the appearance of the frame but rather on semantic information. When these character labels are used for reg. nNCA, the network is forced not only to remove irrelevant variance from the feature representation (like vertical shift, squeeze, stretch, rotate, etc.) but also to encode the *position* of the extracted frame along the character. This implicit classification of a frame during feature extraction is beyond the scope of the feature learning and extraction process and should instead be left for the subsequent HMM recognizer.

Therefore different labels are needed that allow the reg. nNCA training to focus on the irrelevant variations present in the frames. The features extracted after the initialization phase, accompanied by the ground-truth, can be used to train a set of character HMMs. After their parameters have been estimated, an alignment of each HMM can be calculated that is most likely to have produced an observed feature-vector sequence (see [14] for an overview). Fig. 3 shows an example state-level alignment compared to the corresponding character labels.

The total number of states in the HMMs and, therefore, also the amount of classes can be very large (more than $2,000$ in our experiments). When regularized nNCA was applied in [15], the training was based on batches that, although just containing a fraction of sampled examples, still hold samples from every class. For the huge amount of classes here this is infeasible since the batch-size is limited by the available computational resources. Hence just a limited number of classes can be considered in each batch. After choosing a random order of the classes, one batch for each position is constructed. Each contains a fixed number of examples from the corresponding class and its n successors in the precomputed random order, where just a fraction is considered as *labeled* and the rest acts as regularization. The batches are processed in the precomputed order, ensuring that each class is part of n subsequent batches.

The gradient of the NCA objective function reported in [15] was reformulated to ease a vectorized implementation. Since the computation of the difference vector d_{ab} between the feature representation of any two frames a and b is symmetrical ($d_{ab} = -d_{ba}$), the derivative can be expressed as

$$\frac{\partial O_{\text{NCA}}}{\partial f(x^a, W)} = -2 \sum_{b:c^b=c^a} (p_{ab} + p_{ba}) d_{ab}$$
$$+2 \sum_{z \neq a} \left[p_{az} \sum_{b:c^b=c^a} p_{ab} + p_{za} \sum_{b:c^b=c^z} p_{zb} \right] d_{az} \qquad (1)$$

where $f(x^a, W)$ is the feature representation of frame a defined by the weights W, c^x is the class of frame x and p_{ab} as defined in [15]. The amount of required sums is reduced and furthermore the sums over p_{xy} in the second term can be precomputed. An overall speedup of 25% was observed during experimentation.

The experiments show that regularized nNCA can be applied successfully to the domain of handwriting recognition. To the authors best knowledge no attempt has yet been made to apply autoencoder networks as well as further feature optimization to automatic reading. The obtained recognition results indicate, that the methods can nevertheless be extended to this real-world scenario.

4 Experimental Evaluation

Experiments on Arabic script were performed using the IFN/ENIT-database [13]. It consists of scanned forms of 400 writers containing $26,400$ tunesian city names with more than 210k+ characters. The set is split into 4 groups (a,b,c,d) where the d-subset was used as test-set in the experiments. It contains $6,735$ samples.

Experiments were also performed on Roman script using the IAM-database [11] that consists of $1,539$ forms filled out by approximately 657 writers. The database contains roughly $115,000$ samples of a lexicon with more than $10,000$ distinct words. The official training-set contains $6,161$ lines of text while the official test-set consists of $1,861$ lines.

As a reference system the segmentation-free recognition system described in [14] was used. Here, a 8-pixel wide analysis window is used to extract frames from the line-images that were normalized with respect to slant, skew and baseline orientation. Furthermore the average character width is normalized using the distance between local minima of the text-contour. Nine geometric features are extracted from the frames and an approximation of the first order derivative is added as feature. 178 semi-continuous HMMs with Bakis topology are trained for Arabic script (75 for Roman respectively) using a code-book with $1,500$ densities ($2,048$ for Roman) and diagonal covariance matrices. In the experiments the normalization, post-processing and the recognition backend are taken directly from the reference system to allow direct comparison of the recognition performance. During recognition the use of a language model is deliberately avoided. For the IFN/ENIT database the results are reported as *Word Error Rate* (WER) that is computed based on the reference transcription. For the IAM database the *Character Error Rate* (CER) is computed instead.

The HMM backend contains more than $2,000$ distinct states after training which also corresponds to the amount of classes present. During the optimization phase classes containing less than 100 samples are omitted which reduces their total amount to around $1,000$ for Arabic script and $1,500$ for Roman. Each batch contains 100 samples of 30 classes where the middle 500 are considered as labeled examples and the rest acts as regularization as described in [15]. The chosen batch-size and the fraction of labeled examples corresponds to the maximum possible that is computationally feasible.

Since the training and evaluation of the feature learning and extraction system is computationally costly, a few parameters were fixed during experimentation. In all experiments an analysis window with a height of 100 pixels was used which was determined in initial experiments. 24 features were extracted from each frame as it is the maximum number for which both the evaluation and the training in the optimization phase is time- and memory-efficient. All hidden layers of each different architecture used hold the same number of units which is determined by a fixed ratio of $1 : 0.625$ between the input-layer and the first hidden layer, similar to the one used in [6]. For a window-width of 8 pixels this yields hidden layers with 500 units each. A number of different widths for the analysis window are explored for both Arabic and Roman script. For Roman script, different network-layouts are trained in the initialization phase. The results are reported in table 1. Except for the smallest window-width, adding hidden layers improves the error rates. The window-width of 8 pixels seems to be the one most suited for Roman script and obtains a CER of 33.7%. For Arabic script a width of 12 pixels obtains the best WER of 11.7%.

	Number of hidden layers			
Width	1	2	3	3
4 pixels	34.2	33.9	34.4	-
8 pixels	34.1	33.9	**33.7**	12.8
12 pixels	37.1	36.6	36.1	**11.7**
16 pixels	-	-	-	12.6
	Roman (CER)			Arabic (WER)

Table 1. CER/WER for different architectures

Phase	Roman (CER)	Arabic (WER)
Initialization	33.7	11.7
Optimization	31.4	10.8
Reference	30.6	5.4

Table 2. CER/WER after initialization and optimization compared to the reference system.

The best performing networks (3 hidden layers, 12 pixel window width for Arabic, 8 for Roman) were subject to further improvement in the optimization phase. Table 2 shows the obtained results compared to the recognition performance of the reference system. The result after feature optimization on Roman script of 31.4% shows a relative improvement of more than 7% and is comparable to the reference value. On Arabic script the error-rate improves similarly by 8%, although the WER of 10.8% is still considerably worse than the reference.

One possible reason for the missing performance on Arabic script is the larger dependency on diacritic marks compared to Roman script. For many Arabic characters, the amount of diacritic marks influences their identity while their absence just has a small impact on the reconstruction error that is minimized during training. Therefore, some small markings may be omitted in the intermediate feature representation found during initialization. An alternative loss-function that puts an increased weight on these details would probably lead to an improved performance.

Furthermore a very fast convergence (around the 5th epoch) of the recognition rates during the reg. nNCA training was observed. A possible explanation is that, instead of learning inner-class similarities, irrelevant variations are removed which are the result of transformations, influenced by writing style and the general script investigated. These variations are shared between all classes and can therefore be considered after just a few epochs of training.

5 Discussion

In this paper we proposed a novel feature learning approach for offline HWR that applies the experience gathered in isolated digit recognition to the real-world domain of unconstrained offline handwriting recognition. We showed how the HMM recognition backend can be exploited to obtain labels for every frame that form the basis for the improvement of an intermediate representation. The evaluation of the approach on Arabic and Roman script yielded that the resulting features perform well and are – for Roman script – comparable in their recognition performance to a heuristic feature set. The results indicate that an alternative loss function could improve the recognition performance on Arabic script by increasing the impact of small details during the initialization phase. The completely data-driven method does not require any expert knowledge that usually guides the complicated feature-selection process in offline HWR. Furthermore it can be extended easily by incorporating prior geometric knowledge (e.g. pixel neighborhood) and explicit shift invariance (e.g. convolutional RBMs [8]). Therefore, the method's potential is high with respect to learning rather than designing future handwriting recognition systems.

Acknowledgment

This work has been supported by the German Research Foundation (DFG) within project **Fi799/3**.

References

[1] I. Bazzi, R. M. Schwartz, and J. Makhoul. An Omnifont Open-Vocabulary OCR System for English and Arabic. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 21(6):495–504, 1999.

[2] C. Freitas, A. E. Yacoubi, F. Bortolozzi, and R. Sabourin. Isolated word recognition in brazilian bank check legal amounts. In *DAS*, 2000.

[3] D. Guillevic and C. Y. Suen. Cursive script recognition applied to the processing of bank cheques. In *ICDAR*, pages 11–14, 1995.

[4] S. Günter and H. Bunke. Fast Feature Selection in an HMM-Based Multiple Classifier System for Handwriting Recognition. In *DAGM*, pages 289–296, 2003.

[5] G. Hinton. To recognize shapes, first learn to generate images. *Progress in brain research*, 165:535, 2007.

[6] G. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-Based Learning Applied to Document Recognition. In *Intelligent Signal Processing*, pages 306–351. IEEE Press, 2001.

[8] H. Lee, R. Grosse, R. Ranganath, and A. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 609–616. ACM, 2009.

[9] S. Madhvanath and V. Govindaraju. The role of holistic paradigms in handwritten word recognition. *IEEE Transaction on Pattern Analysis and Machine Intelligence.*, 23(2):149–164, 2001.

[10] M. Marin-Jimenez, N. de la Blanca, M. Mendoza, M. Lucena, and J. Fuertes. Learning action descriptors for recognition. pages 5 –8, may 2009.

[11] U. Marti and H. Bunke. The IAM-database: an English sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*, 5(1):39–46, 2002.

[12] H.-S. Park and S.-W. Lee. Off-line recognition of large-set handwritten characters with multiple hidden markov models. *IEEE Transaction on Pattern Analysis and Machine Intelligence.*, 29(2):231–244, 1996.

[13] M. Pechwitz, S. Maddouri, V. Märgner, N. Ellouze, and H. Amiri. IFN/ENIT-database of handwritten Arabic words. *Institute for Communications Technology (IFN)*, 2:1.

[14] T. Plötz and G. A. Fink. Markov Models for Offline Handwriting Recognition: A Survey. *Int. Journal on Document Analysis and Recognition*, 12(4):269–298, 2009.

[15] R. Salakhutdinov and G. Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *AI and Statistics*, volume 1, 2007.

[16] S. M. Touj and H. A. N. E. Ben Amara. Modélisation markovienne planaire pour la reconnaissance de l'écriture arabe. In *CIFED*, pages 104–109, 2004.

[17] S. Vajda, K. Roy, U. Pal, B. B. Chaudhuri, and A. Belaïd. Automation of Indian postal documents written in Bangla and English. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 23(8):1599–1632, December 2009.