Evaluating Retraining Rules for Semi-Supervised Learning in Neural Network Based Cursive Word Recognition

Volkmar Frinken and Horst Bunke

Institute of Computer Science and Applied Mathematics, University of Bern,
Neubrückstrasse 10, CH-3012 Bern, Switzerland
{frinken, bunke}@iam.unibe.ch

Abstract

Training a system to recognize handwritten words is a task that requires a large amount of data with their correct transcription. However, the creation of such a training set, including the generation of the ground truth, is tedious and costly. One way of reducing the high cost of labeled training data acquisition is to exploit unlabeled data, which can be gathered easily. Making use of both labeled and unlabeled data is known as semi-supervised learning. One of the most general versions of semi-supervised learning is self-training, where a recognizer iteratively retrains itself on its own output on new, unlabeled data. In this paper we propose to apply semi-supervised learning, and in particular self-training, to the problem of cursive, handwritten word recognition. The special focus of the paper is on retraining rules that define what data are actually being used in the retraining phase. In a series of experiments it is shown that the performance of a neural network based recognizer can be significantly improved through the use of unlabeled data and self-training if appropriate retraining rules are applied.

1 Introduction

The automatic recognition of handwritten text – starting from single letters and digits to sequences of letters in words, text lines and whole sentences - has been a focus of intensive research for several decades [1, 10]. Yet the problem is far from being solved, especially in the field of unconstrained handwritten word and sentence recognition.

Several types of recognizers for handwritten text have been developed, all of which need a large amount of written text and the corresponding ground truth for training. Creating this ground truth, however, is a costly and tedious task since it needs to be done by humans. As handwritten texts are ubiquitously available, one can raise the question weather it is possible to enhance an unconstrained handwriting recognition system by using texts without ground truth in the training phase. Making use of both labeled and unlabeled data for classifier training is also known as semi-supervised learning [9]. However, almost no related work has been reported for using unlabeled data in handwriting recognition [5].

Most work on semi-supervised learning deals with the standard classification scenario, where a mapping of single points in a feature space (patterns) to class labels is being computed [8, 9, 14]. However, in handwritten word and sentence recognition, a more general problem is considered in the sense that a sequence of feature vectors is to be mapped to a sequence of classes, i.e. a sequence of words or, as described in this paper, a sequence of letters. Nevertheless, basic semi-supervised learning frameworks exists, viz. self-training and co-training [4, 7], which are general enough to cope with sequential data. In self-training, recognizers decode a large set of unlabeled data and use the most confidently recognized patterns to create a new training set. A single recognizer is iteratively retrained by enlarging the original training set with this new set.

In contrast to [5], where the adaptation of a recognition system to a single person's writing style is performed, we are interested in the applicability of semi-supervised learning to general unconstrained handwriting recognition in this paper. We evaluate the impact of different forms of self-training on the recognition accuracy for single word recognition. The recognizer used in this paper is a combination of ten parallel neural networks [2], specifically designed for sequential data, that act as a single recognition system from which a recognition confidence can be deduced. Three deterministic, one probabilistic and one oracle retraining rule, all based on the recognition confidence, have been tested and compared. A significant increase in the recognition accuracy can be reported for all but one of the retraining rules.

The rest of the paper is structured as follows. In Sec-

978-0-7695-3725-2/09 $25.00 © 2009 IEEE
DOI 10.1109/ICDAR.2009.18

tion 2 the concept of self-training as well as the different retraining rules are described. Section 3 introduces the neural network based handwritten word recognizer. Experimental results are presented in Section 4. Finally, Section 5 concludes the paper.

2 Self-Training

2.1 Overview

The basic idea of semi-supervised learning is to enhance a recognizer's accuracy by using both labeled and unlabeled data for training. Handwritten words, text lines, and sentences, however, are usually represented as sequences of feature vectors, rather than by individual feature vectors. This is in contrast with most of the scenarios where semi-supervised learning was used in the past. Nevertheless, there exist two frameworks that are general enough to deal with sequences of feature vectors, rather than single vectors. These are co-training, introduced in [4], and self-training, which goes back as far as [6] and was put in context to co-learning in [7].

In the current paper we focus on self-training. It is based on the idea that a recognizer is retrained on its own most confident output produced from unlabeled data. First, a conventional supervised training on the labeled set initializes the recognizer. Afterwards, several self-training iterations are performed to incorporate the unlabeled data until some stop criterion is met, e.g. the converging of the recognition accuracy. In each of the self-training iterations, the unlabeled set is decoded. Some elements of that decoding are then used for retraining the recognizer.

Self-training as introduced in [7] uses only the most confidently recognized element to retrain the recognizer. In this paper we extend this idea. A conservative approach is to ensure that as least incorrectly labeled data as possible are used for retraining. However, if retraining is done with only those elements whose correctness can nearly be guaranteed, the retraining set does not change significantly and the classifier may remain nearly the same. Enlarging the retraining set, on the other hand, is only possible at the cost of increasing noise, i.e. adding mislabeled words to the training set. With only few correctly recognized words and large amounts of possible misrecognitions, the challenge of successful self-training lies in finding the optimal tradeoff between data quality and data quantity for retraining.

2.2 Retraining Rules

We investigate three deterministic, one probabilistic, and one oracle rule to select the elements used for retraining. To estimate the quality of the recognition output, additional information about the recognition is needed that indicates how confident the recognizer is. The retraining rules compared in this paper are based on this confidence. To compute the confidence measure, an independent validation set is used to transform a given confidence c into a correctness probability $p_{\text{validation}}(c)$ value. It indicates the probability of a word being correct if it was recognized with this given confidence c (for more details see Section 3.3).

The three deterministic retraining rules compute a confidence threshold each and select all elements recognized with a confidence equal to or above that threshold. Additionally, they select the whole original training set for retraining. The conservative deterministic retraining rule tries to ensure that no missclassified samples are included in the set used for retraining. It places its associated confidence threshold t_{high} to the lowest value so that $p_{\text{validation}}(t_{\text{high}}) = 1$. Since this is the highest threshold considered, it is termed *High Confidence* retraining rule. To obtain a larger retraining set, the *Medium Confidence* retraining rule selects all words that are more likely to be correct than wrong by placing its threshold t_{med} at the lowest value with $p_{\text{validation}}(t_{\text{med}}) \geq 0.5$. As the third deterministic and most relaxed rule, the *Low Confidence* retraining rule selects all words for retraining, regardless of their recognition confidence.

To ensure a large retraining set as well as some level of data quality, a probabilistic retraining rule is additionally introduced. The *Weighted Random* retraining rule randomly samples elements with replacement, where the probability of an element being chosen increases with its recognition confidence. Furthermore, elements from the original training set are also sampled with replacement and added to the retraining set.

Finally, the *Oracle* retraining rule selects all correctly recognized words together with the original training set for retraining. This rule reflects the ideal case and is used to obtain a theoretical upper bound on the system's performance.

3 Recognition System

3.1 Preprocessing

To transform the word images into feature sequences that can be fed into the recognizer for training and testing, several preprocessing steps are applied. The words used in the experiments come from the IAM database [13]. They are extracted from pages of written texts, which were scanned and separated into individual text lines. After binarizing the image with a threshold on the grey scale value, the slant and skew of each textline was corrected and the width and height were normalized. For details on these steps, we refer to [12]. Next, text lines are split into individual words. Then features are extracted. For this purpose, each word was transformed into a sequence of feature vectors with a horizontally sliding window. A window with a width of one pixel was used to extract nine geometric features at each position, three global and six local ones. The global features are the 0^{th}, 1^{st} and 2^{nd} moment of the black pixel's distribution within this window. The local features are the position of the topmost and bottommost black pixel, the inclination of the top and bottom contour of the word at that position, the number of vertical black/white transitions and the average grey scale value between the topmost and bottommost black pixel. Once again, we refer to [12] for more details.

3.2 Bidirectional Long Short-Term Memory Neural Network

The recognizer used in this paper is a recently developed recurrent neural network, termed *bidirectional long short-term memory* (BLSTM) neural network [2]. Each hidden layer is made up of so called *long short-term memory* blocks instead of simple nodes. These memory blocks are designed specifically to address the *vanishing gradient problem* which describes the exponential increase or decay of values as they cycle through recurrent network layers. This is done by nodes that control the information flow in and out of each memory block. For details about BLSTM networks, we refer to [2, 3].

The network is *bidirectional*, i.e. a sequence is fed into the network in both the forward and the backward mode. The input layers with one node for each feature is each connected to separate, recurrent hidden layer. Both hidden layers are in turn connected to a single output layer. One hidden layer deal with the forward sequence, the other hidden layer with the backward sequence. At each position p of the input sequence of length l, the output layer sums up the values coming from the hidden layer that has processed positions 1 to p and the hidden layer that has processed position l down to p.

The output layer contains one node for each possible character in the sequence plus a special ε node, to indicate "no character". At each position, the output activations of the nodes are normalized so that they sum up to 1 and are treated as probabilities that the node's corresponding character can occur at this position. In a subsequent step only the nodes with the highest probability are considered while the others are discarded. This sequence of one output activation at each time step is further considered. Eliminating repeated activations of the same node and activations of the ε node (in that order), yield the desired character sequence.

3.3 Confidences

To apply the retraining rules given in Section 2, a reliable confidence measure is needed. A simple approach uses the intensity of the output activations as a confidence measure. If the normalized output activations at each timestep are interpreted as a probability for each letter, the product of the most active ones over the whole sequence can be seen as the recognition likelihood for that sequence. However, initial experiments have shown that this approach does not result in a good confidence measure.

To obtain better results, we took an alternative approach. Ten neural networks, initialized with different random weights were trained separately in each step. For each sequence, the ten networks produce ten outputs. A simple count of how many networks agree gives a preliminary confidence value n for that output y. To break a tie in the voting, the network with the best performance on an independent validation set is used. In step two, the recognition accuracy on the validation set with a preliminary confidence value n is computed separately for all values of n and stored as $p(n)$. In the third step, the output itself is additionally taken into account. The recognition accuracy for each different output y and each value of n is computed on the validation set and stored as $p(y, n)$ if enough samples exist to estimate it robustly. Finally, the recognition's confidence value of the word's output is set to $p(y, n)$ if that value exists, otherwise to $p(n)$. For more details on these confidence values, we refer to [11].

4 Experiments

4.1 Setup

To evaluate the applicability of self-training in handwritten word recognition, several test runs with different settings were conducted. The words used are the 4000 most frequently occurring words from the IAM Handwriting Database [13]. The database is split up in a test set (5342 words), a validation set (5590 words), and a work set (38127 words) with different writers each[1]. To simulate the existence of few labeled data and large amounts of unlabeled data, the work set was randomly split up into a labeled set to be used for training the initial recognizer and an unlabeled set to be used for retraining. Clearly, the label of each word in the unlabeled set is known, but it was ignored during th experiments. All retraining rules were tested with 2000, 4000, 6000, 8000, and 10000 labeled words.

To initialize the networks, they were trained on the labeled part of the working set. In each self-training iteration, the networks decoded a) the validation set from which the confidence mapping was computed, b) the unlabeled set from which the new training sets were created and c) the test set to compute the recognition accuracy. The new training sets were then used to retrain each network. For all de-

[1] The splitting has be done according to the standard writer independent recognition task `http://www.iam.unibe.ch/fki/databases`

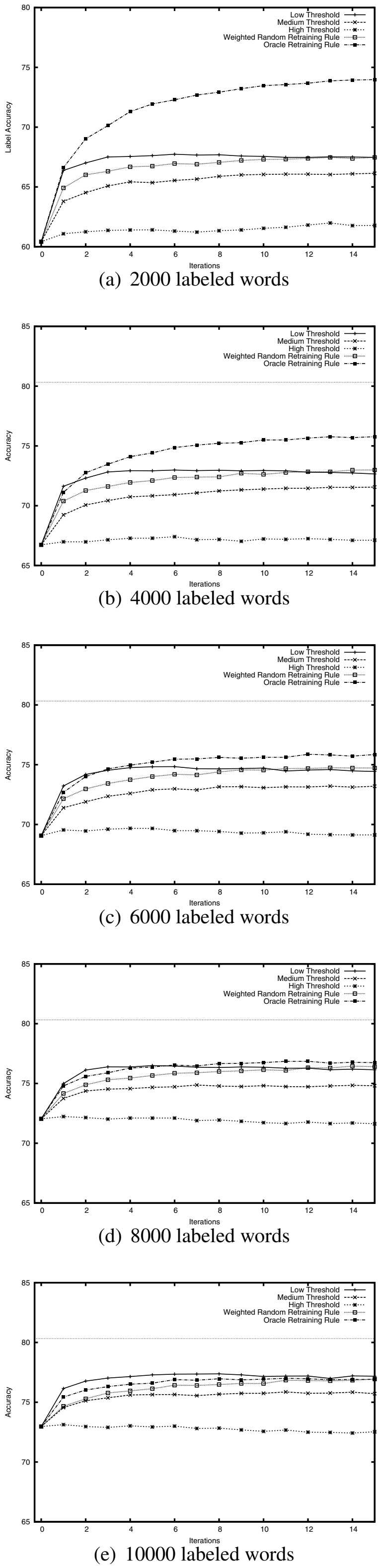
terministic retraining rules, the retraining sets were exactly the same for all ten networks in each step, while in case of the probabilistic retraining rule *Weighted Random* the retraining sets differed. Retraining was bounded to 10 backpropagation iterations to keep the computation time within reasonable bounds.

4.2 Results

Self-training has been performed for 15 iterations. In a single test run, the same retraining rule has been applied constantly. Furthermore, one test run for each rule and each size of labeled data has been conducted. Each network decoded the test set after each iteration separately. The averaged label accuracy after each iteration can be seen in Fig. 1. The average accuracy of neural networks that are trained on the entire labeled workset is 80.32% and is indicated by the horizontal line in Fig. 1, (b)-(e). The test set label accuracy is the average of each word's label accuracy, which is given by the number of correctly recognized letters minus the number of added and missing letters divided by the number of letters in the word's ground truth.

First of all, it can be seen that the initial as well as the asymptotic accuracies increase with the size of the labeled set. Secondly, the *High Confidence* retraining rule did not achieve a constant increase and at some points even decreased the recognition accuracy. The other retraining rules increased the accuracy substantially during the first few iterations and stabilized on a significantly higher level than the initial accuracy[2]. The *Medium Threshold* retraining rule performed better than the *High Confidence* retraining rule, but still not as good as the *Low Confidence* retraining rule. This behavior can be explained with the number of elements used for retraining. The lower the confidence threshold, the more elements lie above that threshold and are chosen for retraining. The *High Confidence* retraining rule added in the first iteration around 1,000 elements to the new training set and even less additional elements in the following iterations. The *Medium Threshold* initially added between 10,000 and 14,000 elements to the retraining set and later on much less (between 500 and 1,000 elements). The very few elements above the *High Confidence* threshold barely make any difference to the original training set and exhaustive retraining can then lead to overfitting. The same argument, but to a lesser degree, holds for the *Medium Confidence* retraining rule.

The *Weighted Random* retraining rule ensures that the retraining set is sufficiently large and that good data is preferred for retraining. The resulting curves follow closely the form of the curves of the *Oracle* retraining rule. In all five cases, the *Weighted Random* retraining rule reached the best recognition accuracy among all non-oracle retraining



(a) 2000 labeled words

(b) 4000 labeled words

(c) 6000 labeled words

(d) 8000 labeled words

(e) 10000 labeled words

Figure 1. The averaged accuracies of all ten networks on the test set for each self-training iteration.

[2] All results are statistically significant on the $\alpha = 0.05$ level.

rules after all 15 iterations, except for the case of 10,000 labeled words (Fig. 1(e)) where the difference to the best one is only 0.2%. Among the deterministic retraining rules, the *Low Confidence* threshold performed best. However, the *Medium Threshold* retraining rule kept on improving the accuracy for more iterations than the *Low Confidence* retraining rule which started to decline after few iterations. The *Oracle* retraining rule uses much less elements than the *Low Threshold* rule but overtakes it after a few iterations or comes very close. Their difference in Fig. 1(e) is not statistically significant after the 8th iteration. It occurs that, initially, data quantity is most important for self-training, while data quality gains more impact the more self-training iterations are performed.

5 Conclusion

To investigate semi-supervised learning for handwriting recognition, we split a work set randomly into a large unlabeled set and a small labeled set that was used to initially train the neural network recognizer. Then we performed fifteen self-training iterations with five different retraining rules on five different sizes of the labeled set. In each self-training iteration, the networks decode the set of unlabeled data, and assign a pseudo-label and a confidence to the output. Depending on the retraining rule, certain recognized words from the unlabeled set are added to the original training set for retraining.

In this paper we demonstrated that semi-supervised learning, here in the form of self-learning, can be successfully applied to the task of handwriting recognition and made evident that it is possible to substantially increase the recognition accuracy with unlabeled data. Furthermore, we introduced different retraining rules to go beyond the idea of retraining only with the best elements. It turned out that during the first self-training iterations, data quantity is more important than quality, from which we draw the conclusion that in this setup and network architecture some implicit form of unsupervised learning takes place.

In future work, we will experiment with dynamically changing retraining rules to further exploit the observation that the impact shifts from data quantity to quality in the course of the iterations. Another line of research will include self-training with Hidden Markov Models (HMM) as well as co-training, where two different recognizers retrain each other. Furthermore, we may consider extending single word recognition to text line recognition.

Acknowledgments

This work has been supported by the Swiss National Center of Competence in Research (NCCR) on Interactive Multimodal Information Management (IM2).

References

[1] Alessandro Vinciarelli. A Survey On Off-Line Cursive Word Recognition. *Pattern Recognition*, 35(7):1433–1446, 2002.

[2] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A Novel Connectionist System for Unconstrained Handwriting Recognition. *IEEE Transaction on Pattern Analysis and Machine Intelligence, Accepted for publication*.

[3] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist Temporal Classification: Labelling Unsegmented Sequential Data with Recurrent Neural Networks. In *23rd Int'l Conf. on Machine Learning*, pages 369–376, 2006.

[4] Avrim Blum and Tom Mitchell. Combining Labeled and Unlabeled Data with Co-Training. In *COLT' 98: Proc. of the 11th annual Conference on Computational Learning Theory*, pages 92–100, New York, NY, USA, 1998. ACM.

[5] Gregory R. Ball and Sagur Srihari. Prototype Integration in Off-Line Handwriting Recognition Adaptation. In *Proc. Int'l. Conf. on Frontiers in Handwriting Recognition*, pages 529–534, 2008.

[6] H. J. Scudder. Probability of Error of Some Adaptive Pattern-Recognition Machines. *IEEE Transaction on information Theory*, 11:363–371, 1965.

[7] Kamal Nigam and Rayid Ghani. Analyzing the Effectiveness and Applicability of Co-Training. In *9th Int'l Conf. on Information and Knowledge Management CIKM*, pages 86–93, 2000.

[8] Matthias Seeger. Learning with Labeled and Unlabeled Data. Technical report, University of Edinburgh, 5 Forest Hill, Edinburgh, EH1 2QL, 2002.

[9] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

[10] Réjean Plamondon and Sargur N. Srihari. On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 22(1):63–84, 2000.

[11] Roman Bertolami, Matthias Zimmermann, and Horst Bunke. Rejection Strategies for Off-Line Handwritten Text Line Recognition. *Pattern Recognition Letters*, 27(16):2005–2012, 2006.

[12] Urs-Victor Marti and Horst Bunke. Using a Statistical Language Model to Improve the Performance of an HMM-Based Cursive Handwriting Recognition System. *Int'l Journal of Pattern Recognition and Artificial Intelligence*, 15:65–90, 2001.

[13] Urs-Victor Marti and Horst Bunke. The IAM-Database: An English Sentence Database for Offline Handwriting Recognition. *Int'l Journal on Document Analysis and Recognition*, 5:39–46, 2002.

[14] Xiaojin Zhu. Semi-Supervised Learning Literature Survey. Technical Report 1530, Computer Science, University of Wisconsin-Madison, 2005. `http://www.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf`.