

# Handwritten word verification by SVM-based hypotheses re-scoring and multiple thresholds rejection

Laurent Guichard  
INRIA Bretagne Atlantique  
IRISA, Rennes, France  
lguichar@irisa.fr

Alejandro H. Toselli  
Instituto Tecnológico de Informática  
UPV, Valencia, Spain  
ahector@iti.upv.es

Bertrand Couasnon  
INSA de Rennes  
IRISA, Rennes, France  
couasnon@irisa.fr

## Abstract

*In the field of isolated handwritten word recognition, the development of verification systems that optimize the trade-off between performance and reliability is still an active research topic. To minimize the recognition errors, usually, a verification system is used to accept or reject the hypotheses output by an existing recognition system. In this paper, a novel verification architecture is presented. In essence, the recognition hypotheses, re-scored by a set of the support vector machines, are validated by a verification mechanism based on multiple rejection thresholds. In order to tune these (class-dependent) rejection thresholds, an algorithm based on dynamic programming is proposed which focus on maximizing the recognition rate for a given prefixed error rate.*

*Preliminary reported results of experiments carried out on RIMES database show that this approach performs equal or superior to other state-of-the-art rejection methods.*

## 1. Introduction

The interest in developing effective verification systems (VSs) for handwritten word recognition applications (HWR) that can distinguish when their outputs are not recognized with enough certainty (and consequently rejected) is still an active research topic. Such VSs are crucial and vital for several security-sensitive applications, as for example the case of recognition of handwritten postal-address, legal amounts handwritten in bank checks, etc.

Commonly, VSs involve two parts: the confidence measures computation (CMs), which gives an idea of the achieved recognition quality of each word image, and the thresholding-based procedure, which stands for trading off between errors and rejections.

In the literature we can find a wide diversity of VSs for HWR. On the one hand are the VSs directly applying a rejection rule to the HWR hypotheses scores [8, 7, 10]. For HWR based on Hidden Markov Models (HMMs), which is by far the most successfully employed statistical approach according to the state-of-the-art, VS rejection mechanisms rely usually on the same HMM decoding scores. Those approaches are limited by the intrinsic nature of the HWR, aimed at maximizing the recognition but not the rejection. On the other hand, there are some VSs which re-score HWR hypotheses independently from their decoding scores before performing the accept/reject action. This is the case described in [9], where a multi-layer perceptron (MLP) is employed to reevaluate the hypotheses. Because of this classifier is not specifically suitable for rejection tasks, the use of support vector machines (SVM) to re-score these HWR hypotheses emerges as a promising alternative, as they already proved their ability to verify isolated handwritten digits [1, 2].

As mentioned above, VS approaches rely on thresholding methods, which intend to adjust threshold values to decide whether accept or reject given recognized hypotheses. The formulation of the best error-reject trade-off and the related optimal reject rule is given in [3]. According to this, the optimal error-reject trade-off is achieved only if the *a posteriori probabilities* of the classes are known exactly. As they are always affected by errors, [4] suggests the use of multiple reject thresholds to obtain the optimal decision and reject regions. Nevertheless, in the field of HWR, most VSs do not take in account this and employ just one single threshold.

An inherent difficulty of the multi-threshold VSs, within the context of HWR based on HMMs, lies in how to defining the appropriate classes associated to each of the thresholds, which do not necessarily correspond to the lexicon words. Another difficulty is also the tuning of rejection thresholds, which has been already investigated in [4, 6, 8, 13], where different algorithms are

proposed but neither of them guarantee an optimal solution.

In this paper, two main contributions are presented which aims at improving both rejection and recognition capabilities of the verified HWR. The first one describes a new VS approach which employs an alternative SVM-based confidence measures relying on the grapheme segmentation information from the HMMs Viterbi decoding, and applies multiple thresholds to optimize the error-rejection trade-off. The second contribution focuses on presenting a new algorithm for computing multiple reject threshold values based on dynamic-programming which, unlike others approaches, guarantees an optimal solution.

This work is organized in the following way. Section 2 and 3 addresses in detail the two contributions above-mentioned. Experimental results and conclusion are presented in sections 4 and 5.

## 2 Proposed Verification system approach

The proposed VS is suitable for HWR based on grapheme/character-segmentation (explicit or implicit). For a given word image input  $s$ , the HWR outputs the  $N$ -best recognized hypotheses along with their corresponding grapheme segmentations and recognition scores. This list of  $N$ -best hypotheses serves as input of our VS approach. To represent this list, we employ the following notation:  $\langle h_1 = (w_1, r_1), \dots, h_N = (w_N, r_N) \rangle$ , where  $w_i$  and  $r_i$  denote respectively the transcription and grapheme segmentation of the  $i$ th recognized hypothesis  $h_i$  of word image  $s$ . In turn, each hypothesis  $h_i = (w_i, r_i)$  is associated with a sequence of grapheme-label and sub-image pairs:  $\langle (c_{i,1}, g_{i,1}), \dots, (c_{i,n_i}, g_{i,n_i}) \rangle$ , where  $n_i$  is the number of recognized (grapheme/character) labels of the corresponding hypothesis transcription  $w_i$ . Furthermore, each  $h_i$  has an associated probability  $P_{HWR}(h_i)$  emitted by the HWR.

Our VS approach is composed of three different modules: *grapheme feature extraction*,  *$N$ -best hypotheses re-scoring* and *hypothesis selection and verification*.

The first module makes use of the segmentation information provided by HWR to split input word image into the corresponding grapheme sub-images (i.e. character images in our case). Then, a feature extraction process transforms each of these sub-images into a 95-dimensional real-value vector composed of the following set of features:

- 8th order Zernike moments (45 components);
- 8-contour directions histogram using Freeman chain code representation (48 components);

- Normalized pixels distributions within grapheme image area lying above the word upper line and grapheme image area lying between word base and upper lines (2 components).

The second module performs a re-scoring of each  $N$ -best recognized hypotheses by using SVM classifiers, each of which modeling a specific grapheme class  $c$  from the whole grapheme classes set considered in the recognition. In this way, given a pair  $(c_{i,j}, g_{i,j})$  with  $i \in [1, N]$  and  $j \in [1, n_i]$ , the corresponding SVM assigns to it a new score  $P_{SVM}(c = c_{i,j}|g_{i,j})$ . The SVM output score is approximated to a *posterior probability* by using the *softmax* function, as described in [12]. Once all individual grapheme probabilities have been computed, a global SVM score of hypothesis  $h_i$  is calculated as the geometric mean of their respective grapheme scores:

$$P_{SVM}(h_i) = \sqrt[n_i]{\prod_{j=1}^{n_i} P_{SVM}(c = c_{i,j}|g_{i,j})} \quad (1)$$

We realized after some informal experiments that this way of computing the SVM global score works properly well for this case. Moreover, this makes the SVM score independent from hypothesis length (number of graphemes) and thereby comparable across different length hypotheses.

The final confidence measure (CM) of hypothesis  $h_i$  is then computed by linearly combining their respective global HMM score (given by the HWR system) and SVM score:

$$P(h_i) = \alpha P_{SVM}(h_i) + (1-\alpha) P_{HMM}(h_i) \quad \forall i \in [1, N] \quad (2)$$

This linear combination of classifier scores aims at balancing their effect by the empirically tuned coefficient  $\alpha$ .

Once all hypotheses of the  $N$ -best list have been re-scored, the third and last module is in charge to select the best one (i.e. with the maximal CM score) and to perform the accept/reject action on it. In order to do this, the hypotheses are first re-ordered according to their new CM scores, defining a new list:  $\langle \hat{h}_1, \dots, \hat{h}_N \rangle$ , such that  $P(\hat{h}_i) \geq P(\hat{h}_j) \quad \forall 1 \leq i < j \leq N$ . Then, the reject/accept action decision is performed by the thresholding mechanism using the computed difference between the two best re-scored hypotheses

$$d_{12} = P(\hat{h}_1) - P(\hat{h}_2)$$

as a value to be compared with the corresponding threshold. Experiments conducted by other works [9] have shown that this strategy gives the best results.

As was mentioned in section 1, the proposed verification mechanism is based on multiple class-dependent thresholds. To define these classes, we have clustered into different length-classes all word transcriptions from the HWR lexicon according to their length. It is worth noting that the use of length-class-dependent thresholds serves to compensate the inaccuracy of the *a posteriori probabilities* mentioned earlier and also somewhat to mitigate the problem related to the empirical normalization that does not make fully comparable, for example, 10-characters words with 2-characters words. Formally, the set of length-classes is defined as:

$$\Omega = \{length(w) : w \in \text{Lex}\}$$

where *length* is a function returning the number of graphemes of word transcription *w*. We also employ  $\omega_j \in \Omega$  with  $j \in [1, |\Omega|]$  to denote an element belonging to  $\Omega$ . Thus, each of the length-classes:  $\omega_1, \omega_2, \dots, \omega_{|\Omega|}$  has been linked to a respective threshold:  $t_1, t_2, \dots, t_{|\Omega|}$ , whose values are set up during the tuning phase. The description of this tuning phase is detailed in 3.

The verification process performs for a given selected hypothesis  $\hat{h}_1$  and its associate threshold  $t_j$  ( $t_j \rightarrow \omega_j = length(\hat{h}_1)$ ) the accept/reject action of word image *s*, according to:

$$\text{if } d_{12} \geq t_j \text{ then accept } \hat{h}_1 \text{ else reject } \hat{h}_1$$

### 3 Multiple thresholds tuning algorithm

As was seen, the verification system presented here rely on a set of previously set-up thresholds. Looking for the best thresholds is not a trivial problem, involving a combinatorial optimization over all their possible values.

Let *S* be a validation set of word images samples on which threshold values tuning is carried out. Likewise, let  $S_i \subseteq S$ ,  $i \in [1, |\Omega|]$  be sets of word samples with the same lengths:

$$S_i = \{w : length(w) = \omega_i, w \in \text{Lex}, \omega_i \in \Omega\}$$

Additionally, the following definitions for *performance* (PFR), *error rate* (ER) and *rejection rate* (RR) for our VS will be adopted:

$$\text{PFR} = \frac{Corr}{|S|} \quad \text{ER} = \frac{Err}{|S|} \quad \text{RR} = 1 - \text{PFR} - \text{ER} \quad (3)$$

where *Corr* and *Err* are respectively the number of words correctly and incorrectly classified.

In similar way as described [4], the problem of tuning a set of thresholds  $t_1, \dots, t_{|\Omega|}$  can be formulated in

terms of PFR and ER as follows:

$$\begin{cases} \max_{t_1, \dots, t_{|\Omega|}} \text{PFR}(t_1, \dots, t_{|\Omega|}) \\ \text{ER}(t_1, \dots, t_{|\Omega|}) \leq \text{ER}_{max} \end{cases} \quad (4)$$

where  $\text{ER}_{max}$  is a prefixed maximal error rate. The final goal here is to find the threshold values that maximize the performance of the system without exceeding a given  $\text{ER}_{max}$ .

Existing state-of-the-art algorithms for multiple thresholds tuning are not optimal [4, 13, 6]. The new tuning-threshold algorithm presented here is inspired from the *0-1 KnapSack problem* resolution based on dynamic programming [11]. Actually, this dynamic-programming-based approach leans on expression (5) rather than (4), where absolute values *Corr* and *Err* are used instead of PFR and ER:

$$\begin{cases} \max_{t_1, \dots, t_{|\Omega|}} \text{Corr}(t_1, \dots, t_{|\Omega|}) \\ \text{Err}(t_1, \dots, t_{|\Omega|}) \leq \text{Err}_{max} \end{cases} \quad (5)$$

For sake of convenience, we define the auxiliary function:  $\mathbf{F} : s \mapsto (\text{Corr}_s, \text{Err}_s, P_s)$   $s \in S_i, \forall i \in [1, |\Omega|]$ , which for each  $s \in S_i$ , returns the associated  $P_s$  (CM of sample *s*), as well as the  $\text{Corr}_s$  and  $\text{Err}_s$  (number of samples correctly and incorrectly classified) computed on the samples  $s' \in S_i$  whose  $P_{s'} \geq P_s$ . Furthermore, we introduce the accumulator function  $A(l, \text{Err})$ , which returns the maximal number of well recognized samples that can be attained with a number of errors equal or lower than *Err* considering only samples belonging to the class sample sets:  $S_1, \dots, S_l$  where  $l \in [1, |\Omega|]$ . Thus,  $A(l, \text{Err})$  can be recursively defined as follows:

$$\begin{cases} A(0, \text{Err}) = \max_{s \in S_1, \text{Err}_s \leq \text{Err}} \text{Corr}_s \\ A(l, \text{Err}) = \max_{s \in S_l, \text{Err}_s \leq \text{Err}} A(l-1, \text{Err} - \text{Err}_s) + \text{Corr}_s \end{cases} \quad (6)$$

The algorithm 1 finds the optimal solution for  $A(|\Omega|, \text{Err}_{max})$  using dynamic programming. Computation of  $A(l, \text{Err})$  is made iteratively until  $l = |\Omega|$  and  $\text{Err} = \text{Err}_{max}$ . On each iteration, the sample that maximizes  $A(l, \text{Err})$  is stored in the auxiliary variable  $B(l, \text{Err})$  to make possible to recover the threshold values set which has maximized  $A(|\Omega|, \text{Err}_{max})$ . Basically, the running time of this algorithm depends on the size of validation set and the prefixed maximal *error rate*:  $O(|S| \times \text{Err}_{max})$ . Algorithm 2 recovers the threshold values by backtracking through the information stored in  $B(l, \text{Err})$ .

---

**Algorithm 1** Forward pass: Compute  $A(|\Omega|, Err_{max})$ 

---

$s_0$  : default sample defined by  $F(s_0) = (0, 0, 1.0)$   
// Initialization  
**for**  $Err = 0$  **to**  $Err_{max}$  **do**  
     $A(0, Err) \leftarrow 0$   
**end for**  
// Fill the accumulator A  
**for**  $l = 1$  **to**  $|\Omega|$  **do**  
    **for**  $Err = 0$  **to**  $Err_{max}$  **do**  
         $A(l, Err) \leftarrow A(l-1, Err)$   
         $B(l, Err) \leftarrow s_0$   
        **for all**  $s \in S_l$  **do**  
             $(Corr_s, Err_s, -) \leftarrow F(s)$   
            **if**  $Err_s \leq Err$  **then**  
                 $aux_s \leftarrow A(l-1, Err - Err_s) + Corr_s$   
                **if**  $aux_s > A(l, Err)$  **then**  
                     $A(l, Err) \leftarrow aux_s$   
                     $B(l, Err) \leftarrow s$   
                **end if**  
            **end if**  
        **end for**  
    **end for**  
**end for**

---

---

**Algorithm 2** Backward pass: Track back the thresholds

---

$t$  : set of thresholds to be tuned  
// Initialization  
 $l \leftarrow |\Omega|$   
 $Err \leftarrow Err_{max}$   
// Get the thresholds  
**while**  $l > 0$  **do**  
     $s \leftarrow B(l, E)$   
     $(-, Err_s, P_s) \leftarrow F(s)$   
     $t(l) \leftarrow P_s$  {Threshold for class  $\omega_l$ }  
     $E \leftarrow Err - Err_s$   
     $l \leftarrow (l-1)$  {Next class}  
**end while**

---

## 4 Experiments

### 4.1 Experimental setup

Experiments have been carried out on the RIMES database used at the ICDAR 2009 competition [5]. The database contains a total of 59 202 running words with their transcriptions and a vocabulary-size of 1 612 different words. Table 1 presents basic statistical information of the corpus along with the partition definition employed to carry out the experiments.

The HWR used here is a standard HMMs-based recognizer which extracts feature vectors using a sliding window, models lexicon words by a concatenation

**Table 1.** Basic statistics of the RIMES-DB words corpus and its standard partition.

| Num. of: | Training | Valid. | Test   | Total   | Lex.  |
|----------|----------|--------|--------|---------|-------|
| words    | 44 196   | 7 542  | 7 464  | 59 202  | 1 612 |
| charact. | 230 259  | 39 174 | 38 906 | 308 339 | 65    |

tion of continuous left-to-right grapheme HMMs and employs the Viterbi algorithm to look for the HMM-concatenated models that maximize the probability to produce the given feature vector sequence.

To assess our contributions (VS and multiple thresholds tuning algorithm), comparisons have been made with other methods already published:

**SVM-ST:** VS presented in section 2 using SVM-rescoring and just a global single reject threshold.

**MLP-ST:** VS employing MLP classifier-based grapheme re-scoring (see [9]). As **SVM-ST**, it uses just a global single reject threshold.

**HMM-ST:** as described in [8], a global single reject threshold is applied with a CM defined as the difference between the recognized scores of the first and second HWR best hypotheses.

**SVM-MT-DPR:** our VS explained in section 2 using SVM-rescoring and multiple reject thresholds tuned with the dynamic-programming algorithm detailed in 3.

**SVM-MT-FUM:** verification mechanism explained in section 2 using multiple reject thresholds tuned with the algorithm employed in [4] based on an iterative procedure.

**SVM-MT-MTL:** verification mechanism explained in section 2 using multiple reject thresholds tuned with the *Automatic Multiple-Thresholds Learning algorithm* [13], which is based on an iterative procedure faster and more robust during initialization phase than the one used in **SVM-MT-FUM**.

The SVM classifiers employed to re-score graphemes use a Gaussian kernel and were trained with the one-against-all strategy for multi-class SVM classification. In this sense, grapheme samples to train SVM and MLP classifiers were obtained through segmenting the word images of the training set with our HMMs-based HWR in forced alignment mode.

The RIMES-DB partition sets employed in the experiments are highlighted in table 1. While HMMs, SVMs and MLPs parameters learning is carried out on the training set, multiple thresholds tuning is performed on the validation set using an algorithm derived from [13]. Finally, reported results of the comparisons

among the different approaches have been obtained on the test set.

For the VS using multiple reject thresholds, a number of 17 thresholds were set according to the number of classes produced by regrouping the RIMES lexicon words with the same lengths, (i.e. RIMES lexicon contains words varying from 1 to 17 characters). The number of hypotheses generated by the HWR for each recognized word image was set to 10.

First experiments aim at assessing our VS with its SVM-based CM and its novel multiple thresholds computation mechanisms. The performance is measured through comparisons against others approaches: **SVM-ST**, **MLP-ST** and **HMM-ST**. Second set of experiments seek to show through experimental results that the proposed dynamic-programming-based multi-threshold tuning algorithm performs equal or better than other published algorithms solving the same issue: **SVM-MT-FUM** and **SVM-MT-MTL**.

For the experimental comparisons, we employ the *Receiver Operating Characteristic* (ROC) curve, which plots the *True Rejection Rate* (TRR) versus the *False Rejection Rate* (FRR). The TRR (resp. FRR) is defined as the number of wrong (resp. well) recognized words that are rejected divided by the number of well (resp. wrong) recognized words. In addition, the area under a ROC curve provides an adequate overall estimation of the rejection capabilities. This area is denoted as AROC. The *Performance* (PFR) versus *Error Rate* (ER) curve is also plotted to demonstrate the increase of well recognized words brought by the VS.

## 4.2 Evaluation

### 4.2.1 VS approaches comparison

The following results were all obtained on the test set partition. Figure 1-(left) presents the ROC curves obtained for the four different VS approaches: **SVM-MT**, **SVM-ST**, **HMM-ST** and **MLP-ST**. It can be observed that **SVM-MT** and **SVM-ST** are the best performing approaches in the FRR range between 0% and 30%. Clearly in that range, **SVM-ST** outperforms **HMM-ST** and **MLP-ST**, corroborating the proposed CM quality. Furthermore, **SVM-MT-DPR** approach outperforms all of the others, including **SVM-ST**, confirming the usually good results of the multiple-thresholds-based VSs with respect to the single-threshold ones.

Figure 1-(right) also plots the VS performance versus error rate for each of the proposed approaches. Once again, it is notable specially for the ER range between 0% and 2.5%, the good performance achieved by **SVM-MT-DPR** and **SVM-ST** compared with the others. Additionally for each VS approach, table 2 gives some

more specific results.

**Table 2.** AROC values, TRR values for a constant FRR set to 10%, PFR values without rejection ( $PFR_1$ ) and PFR values for a constant ER set to 2.5% ( $PFR_2$ )

| Approach   | AROC  | TRR(%) | $PFR_1$ (%) | $PFR_2$ (%) |
|------------|-------|--------|-------------|-------------|
| SVM-MT-DPR | 0.899 | 73.3   | 83.7        | 68.4        |
| SVM-ST     | 0.874 | 68.9   | 83.7        | 63.1        |
| MLP-ST     | 0.864 | 64.5   | 82.3        | 58.4        |
| HMM-ST     | 0.822 | 56.3   | 78.6        | 53.6        |

One important feature to notice is the improvement in term of performance even without rejection. Indeed, the performance of the HWR (**HMM-ST**) increases from 78.6% to 83.7% when multi-threshold-based scheme is incorporated.

### 4.2.2 Algorithms comparison

Figure 2-left plots the ROC curves for our VS approach on the validation set using the three different multiple thresholds tuning algorithms: **SVM-MT-DPR**, **SVM-MT-FUM** and **SVM-MT-MTL**. In this plot can be observed the optimal nature of the algorithm **SVM-MT-DPR**, where its corresponding curve remains above the others for all FRR values.

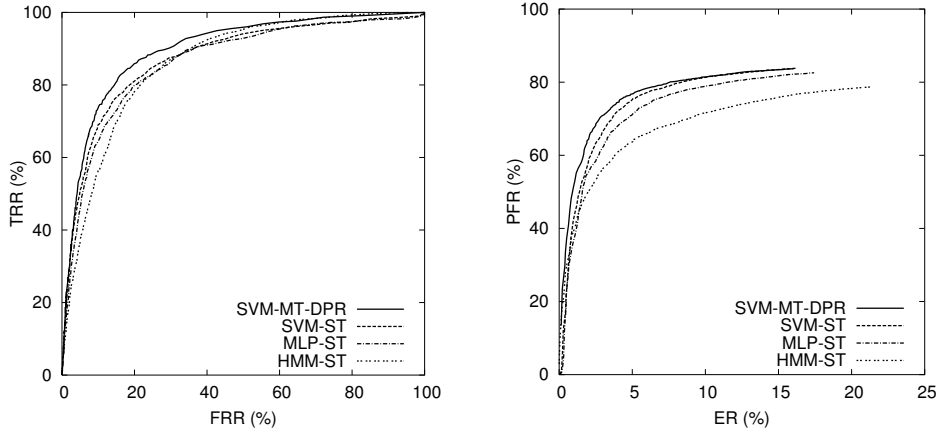
Likewise, the table in the figure 2-right reports the AROC values for all tuning algorithms obtained on the validation and test set partitions. Here, the good performance of the proposed algorithm can be observed **SVM-MT-DPR** as well as its good generalization ability.

In addition, this table reports running times of the 3 algorithms. Algorithm **SVM-MT-DPR** is clearly the fastest as it is about 6 times less time-consuming than **SVM-MT-MTL** and 7 times than **SVM-MT-FUM**.

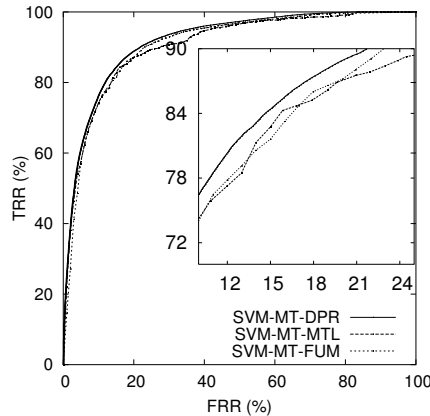
## 5 Remarks and Conclusion

This paper introduces an alternative independent verification system using a confidence measure based on SVMs rescoring and multiple rejection thresholds to verify handwritten word recognized hypotheses. The experimental results obtained show that the proposed approach boosts the rejection capabilities of the HWR as, for example, the performance increases from 53.6% to 68.4% for an error rate set to 2.5%. It also improves the global recognition performance which rises from 78.6% to 83.7% when rejection is disabled.

A new algorithm to tune multiple rejection thresholds has also been presented. It was confirmed experimentally that this tuning algorithm based on dynamic-



**Figure 1.** Left: ROC curve for each VS. Right: performance (PFR %) versus error-rate (ER %) for the different VS.



| Algorithm  | AROC       |       | Running time (in seconds) |
|------------|------------|-------|---------------------------|
|            | Validation | Test  |                           |
| SVM-MT-DPR | 0.920      | 0.899 | 0.65                      |
| SVM-MT-FUM | 0.912      | 0.890 | 4.7                       |
| SVM-MT-MTL | 0.910      | 0.888 | 4.1                       |

**Figure 2.** Right: ROC curves for the three multiple thresholds tuning algorithms on the validation set. Left: AROC values corresponding to all ROC curves produced by the multiple thresholds tuning algorithms on the validation and test sets.

programming produces very optimum results and is less time-consuming than other published algorithms.

**References**

- [1] A. Bellili, M. Gilloux, and P. Gallinari. An mlp-svm combination architecture for offline handwritten digit recognition. *IJDAR*, 5(4):244–252, July 2003.
- [2] C. Chatelain, L. Heutte, and T. Paquet. A two-stage outlier rejection strategy for numerical field extraction in handwritten documents. In *ICPR*, volume 3, pages 224–227, 2006.
- [3] C. K. Chow. On optimum error and reject tradeoff. *Information Theory Society*, 16(1):41–46, Jan. 1970.
- [4] G. Fumera, F. Roli, and G. Giacinto. Reject option with multiple thresholds. *Pattern Recognition*, 33:2099–2101, 2000.
- [5] E. Grosicki and H. E. Abed. Icdar 2009 handwriting recognition competition. In *ICDAR*, 2009.
- [6] M. N. Kapp, C. O. de A. Freitas, and R. Sabourin. Methodology for the design of nn-based month-word recognizers written on brazilian bank checks. *Image and Vision Computing*, 25(1):40 – 49, 2007. SIB-GRAPI.
- [7] M. N. Kapp, C. Freitas, and R. Sabourin. Handwritten brazilian month recognition: An analysis of two nn architectures and a rejection mechanism. *IWFHR*, 0:209–214, 2004.
- [8] A. L. Koerich. Rejection strategies for handwritten word recognition. In *IWFHR*, pages 479–484, 2004.
- [9] A. L. Koerich, R. Sabourin, and C. Y. Suen. Recognition and verification of unconstrained handwritten words. *TPAMI*, 27(10):1509–1522, 2005.
- [10] S. Madhvanath, E. Kleinberg, and V. Govindaraju. Holistic verification of handwritten phrases. *TPAMI*, 21(12):1344–1356, 1999.
- [11] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, Chichester, NY, revised edition, 1990.
- [12] J. Milgram, M. Cheriet, and R. Sabourin. Estimating accurate multi-class probabilities with support vector machines. In *IJCNN*, volume 3, pages 1906–1911, 2005.
- [13] H. Mouchere and E. Anquetil. A unified strategy to deal with different natures of reject. In *ICPR*, volume 2, pages 792–795, 2006.