# Computer Aided Cleaning of Large Databases for Character Recognition

N. Matić, I. Guyon, L. Bottou, and J. Denker
AT&T Bell Laboratories
Holmdel, NJ 07733

**Abstract**

A method for computer-aided cleaning of undesirable patterns in large training databases has been developed. The method uses the trainable classifier itself, to point out patterns that are suspicious, and should be checked by the human supervisor. While suspicious patterns that are meaningless or mislabeled are considered garbage, and removed from the database, the remaining patterns, like ambiguous or atypical, represents valid patterns that are hard to learn and should be kept in the database. By using our method of pattern cleaning, combined with an emphasizing scheme applied on the patterns that are hard to learn (increasing the frequency of presentation), the error rate on the test set has been reduced by half, in the case of the database of handwriten lowercase characters entered on a touch terminal. Our classifier is based on a Time Delay Neural Network (TDNN) and was trained on a large database of 11,500 characters from approximately 250 writers.

## 1 Introduction

A large number of methods in pattern recognition involve training by examples. Patterns are collected and labeled according to the class to which they belong by a human supervisor. In most practical cases, one has to deal with training databases which contain patterns inaccurately labeled and which may even include some meaningless patterns. Such *garbage* patterns in the training data result in performance degradation of the classifier. In particular, training procedures which emphasize atypical patterns (in an effort to capture the tail of the distribution) require eliminating substantially all garbage patterns from the database, otherwise procedures will emphasize garbage patterns as well.

In this paper, we present a method which uses the classifier itself in the process of training, to identify suspicious patterns so that a new human *super-supervisor* can be

Category: Pattern Recognition Methodology and Systems.
Copyright © 1991 AT&T (unpublished). For confidential review only.

queried. Based on that query, without major effort, the *super-supervisor* will browse through the suspicious patterns, eliminate garbage patterns and keep the valid ones. Experiments were carried out on the problem of writer independent on-line isolated handprinted character recognition. Characters were entered on a touch terminal. For this particular pattern recognition problem, a *neural network* was used as the *Learning Machine*, that incorporates the feature extractor and classifier in a single trainable system. By using our method of data cleaning, the error rate on the test set was reduced almost by half.

## 2 Problem Description

During data collection, several kinds of errors can be introduced into the database:

- Hardware failures can cause the insertion of *meaningless* patterns.
- Human failures can cause the insertion of *mislabeled* patterns.

Keeping these patterns in the training database might cause severe degradation in the performance of the classifier.

To get better reliability, one could have the database checked by several people. This procedure has the advantage of providing confidence levels for the labels, but it is impractically laborious for reasonably large databases.

There exist classical methods for handling the problem of outliers. So-called *robust statistics* [**?**] , in a broad interpretation includes different methods that are relatively insensitive to departure from distributional assumptions, outliers, sample censoring etc. This robustness is often obtained by modifying least squares schemes so that outliers have much less influence, or by discarding outliers. This is dangerous because suspicious patterns may not be garbage patterns; if they are valid patterns that happen to be difficult to learn, they are extremely valuable. Such a situation can arise in two different cases:

- The pattern is *atypical* but meaningful and well labeled.
- The pattern is *ambiguous*; reasonable questions can exist about the label.

Our goal, in terms of robustness, is to develop selective robust procedure, that will keep valuable outliers in the database. To do a good job at cleaning the database, it is therefore necessary to insert a human *super-supervisor* in the cleaning process. To alleviate this task, we propose to provide the assistance of the trainable classifier itself, which will point out a restricted number of suspicious patterns.

Examples of meaningless and mislabeled patterns that are present in our database of handwriten lowercase letters are given in figure 1. These patterns are considered garbage and removed from database. Examples of valuable outliers, in the same database, i.e. atypical and ambiguous, are given in figure 2. We want to keep them in order to capture well the tail of the distribution, and to put emphasis on these patterns in order to improve recognition rate.

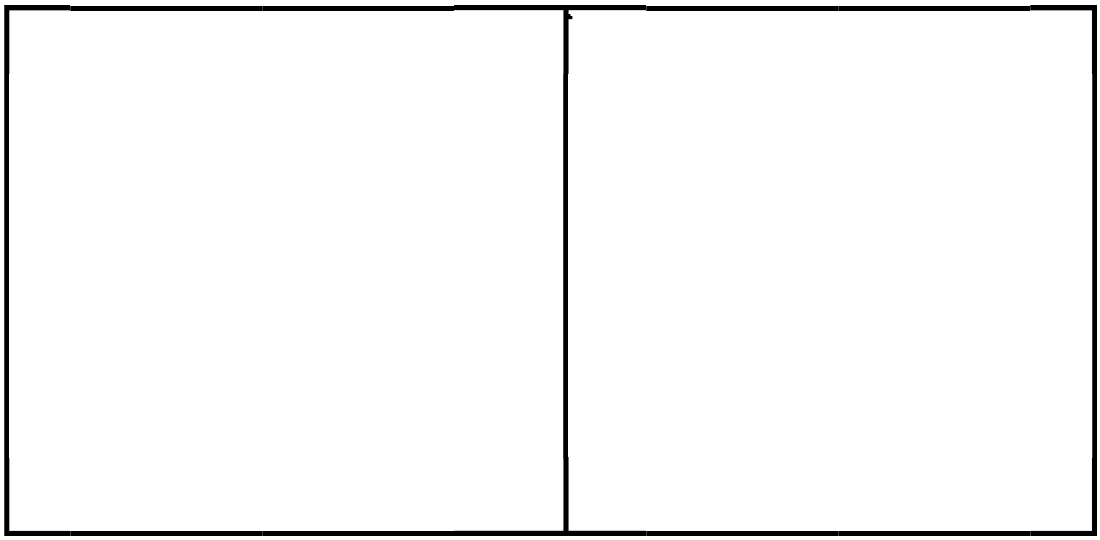

Figure 1: Examples of meaningless and mislabeled patterns in the lowercase letters database.

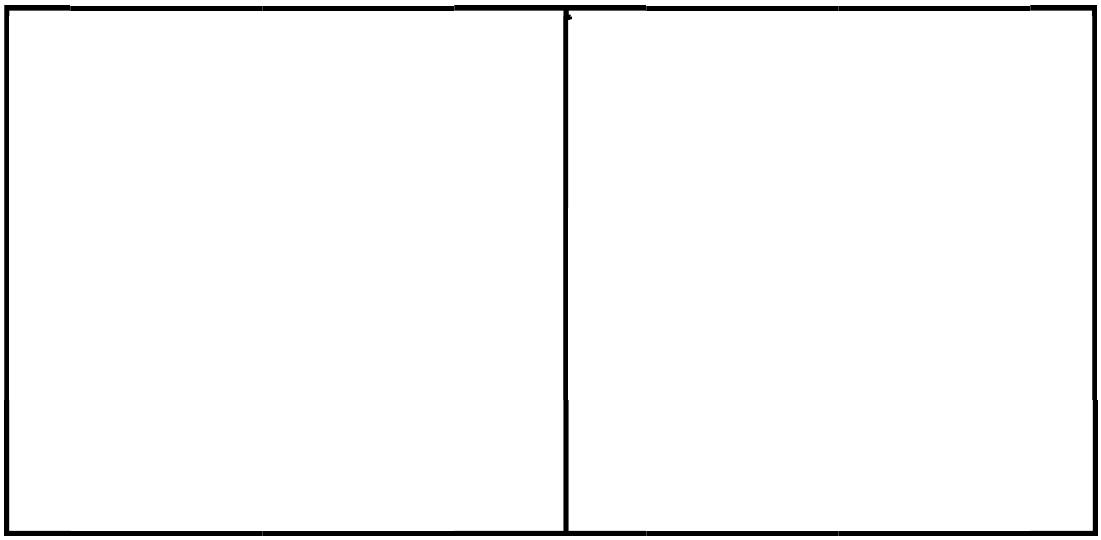

Figure 2: Examples of atypical and ambiguous patterns in the lowercase letters database.

# 3 Emphasizing Ambiguous and Atypical Patterns

Before describing the cleaning (query) system, we will outline the learning-with-emphasis procedure which is intended to capture the tail of the distribution of the training patterns.

An ideal classifier will: (i) perform very well on all *typical* patterns, and (ii) perform reasonably well on *atypical* and *ambiguous* patterns. Most algorithms lead to classifiers satisfying (i). In reference [**?**], it was shown that by using a training scheme which emphasizes difficult patterns, it is possible to get (ii) with no degradation of (i).

Emphasis can be applied to all adaptive training schemes which cycle several times through all the patterns before converging, such as gradient descent techniques. Typically, such techniques minimize an error function such as the mean-squared-error (MSE) cost function:

$$E = \frac{1}{p} \sum_{k=1}^{p} \sum_{l=1}^{c} (f_l(\mathbf{x}^k) - y_l^k)^2 \tag{1}$$

where $p$ is the number of training patterns, $c$ the number of output units (one per class), $f_l(\mathbf{x}^k)$ the state of the $l^{th}$ output of the classifier, when pattern $\mathbf{x}^k$ is presented, and $y_l^k$ the corresponding target value. The target values are binary; for instance $y_l^k = 1$ if $l = \text{class}(\mathbf{x}^k)$ and 0 otherwise. It is usually assumed that $f_l(\mathbf{x}^k)$ provides an approximation to the Bayesian classification function $P(y_l = 1|\mathbf{x}^k)$.

In the following, we will also use the notation $e(k)$ for the contribution of pattern $k$ to the MSE $E$:

$$e(k) = \sum_{l=1}^{c} (f_l(\mathbf{x}^k) - y_l^k)^2 \tag{2}$$

We focus on "on-line" learning procedures; that is, a modification of the classifier parameters happens at each presentation of a pattern. On-line gradient descent is affected by the order and frequency of presentation of the patterns. The least predictable example (i.e. with largest squared error) is the one carrying most *new* information [**?**]. In character recognition, for instance, one should intermix examples of different classes and from different writers, as opposed to presenting batches of similar patterns. Proper shuffling of the examples greatly improves the speed of convergence.

Performance can be further improved by increasing artificially the frequency of presentation of the patterns that are least predictable. At the extreme, it is conceivable to train only on the least predictable pattern (in the squared error sense) every time one cycles through the training set. This time consuming scheme converges asymptotically to a solution that minimizes (over all training patterns) the maximum squared error (as opposed to the mean square error; see [**?**]).

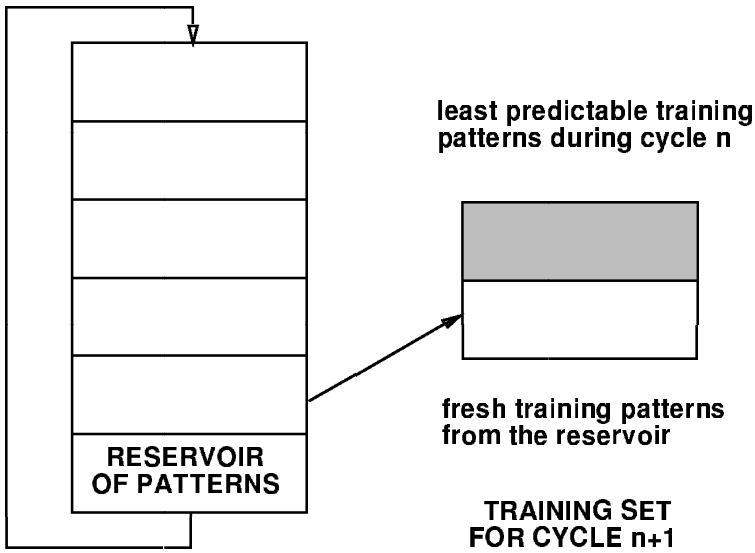


Figure 3: Emphasizing scheme. The least predictable examples are presented more often.

A more practical implementation of this idea was proposed in [?] (figure 3). Training patterns are put into a reservoir, from which only a small subset of training patterns is used at each training cycle. At the end of a cycle, patterns in this subset are sorted in order of increasing predictability. The least-predictable half is kept, and the second half is replaced by fresh patterns taken from the reservoir.

If we look at the examples with the largest squared error, we can see that garbage patterns will fit the description of the least predictable patterns. On the other hand, it is clear that they don't carry any new information. So, the danger of this scheme is that if the database contains *mislabeled* of *meaningless* patterns, emphasis will be put on them as well as on the highly important *atypical* and *ambiguous* patterns. The significant improvements obtained with the emphasizing scheme [?] compared with "classical" procedures, where all the patterns are presented at each training cycle, motivated the design of an efficient cleaning method.

# 4 Super-Supervised Learning

Super-supervised learning refers to filtering the suspicious patterns (meaningless, mislabeled, ambiguous, or atypical) using a trainable classifier and a human super-supervisor (supervisor II). In addition to the basic classification task, the classifier is used to divide the patterns, based on some predefined criteria, into two categories: the predictable versus unpredictable; the latter are declared suspicious and sent to supervisor II.

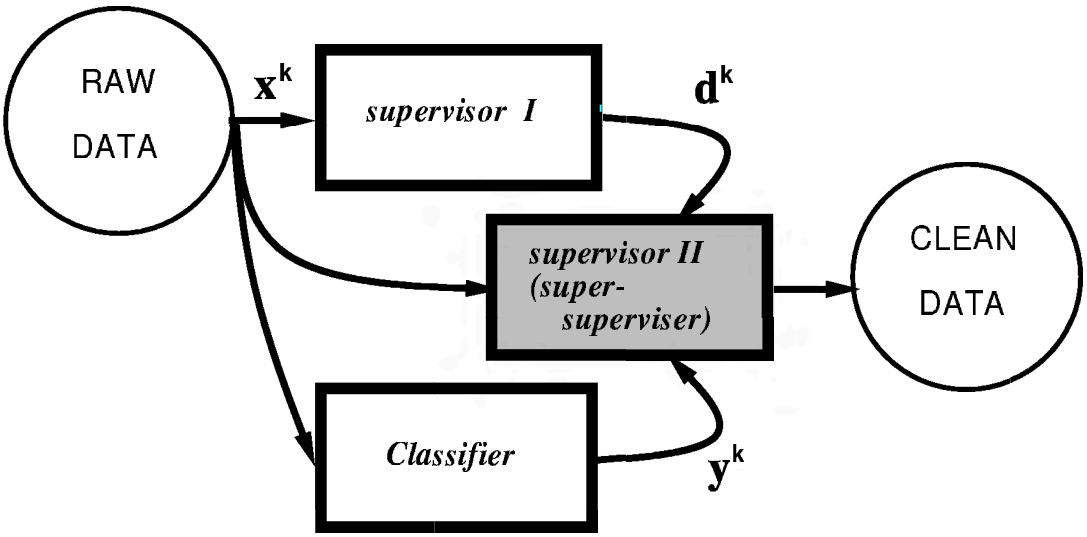


Figure 4: Block diagram of the cleaning scheme: super-supervision.

A block diagram of the procedure is presented in figure 4. The classifier has already received some preliminary training. Supervisor I provided the labels that appear in the training database before the cleaning operation is started. When a subsequent pattern is presented, an error (such as the one computed from equation 2) is obtained by comparing the output of the classifier with the label provided by supervisor I. If this error exceeds a certain threshold, supervisor II will be queried before the pattern is incorporated into the training set. The point is to get maximum information per unit effort from supervisor II, since most of the patterns will skip query.

We propose two complementary implementations of this scheme: interactive cleaning and batch cleaning.

### 4.1 Interactive Cleaning

"Interactive super-supervision" has the advantage of combining cleaning and training in a single session. A flow diagram of this method is presented in figure 5. Initialization is performed by training the classifier with a few clean examples.

After initialization, at step $k$ of the cleaning process, a new pattern $\mathbf{x}^k$ is presented to the classifier. The prediction of the classifier is compared to the label provided by supervisor I and the error $e(k)$ is computed. If the error is below a given threshold $\theta$, the pattern is immediately incorporated into the training set. Otherwise, the pattern is sent to supervisor II for checking. Depending on the decision of supervisor II, the pattern is either incorporated into the training set or discarded.

The advantage of interactive cleaning is that when the session ends, both training

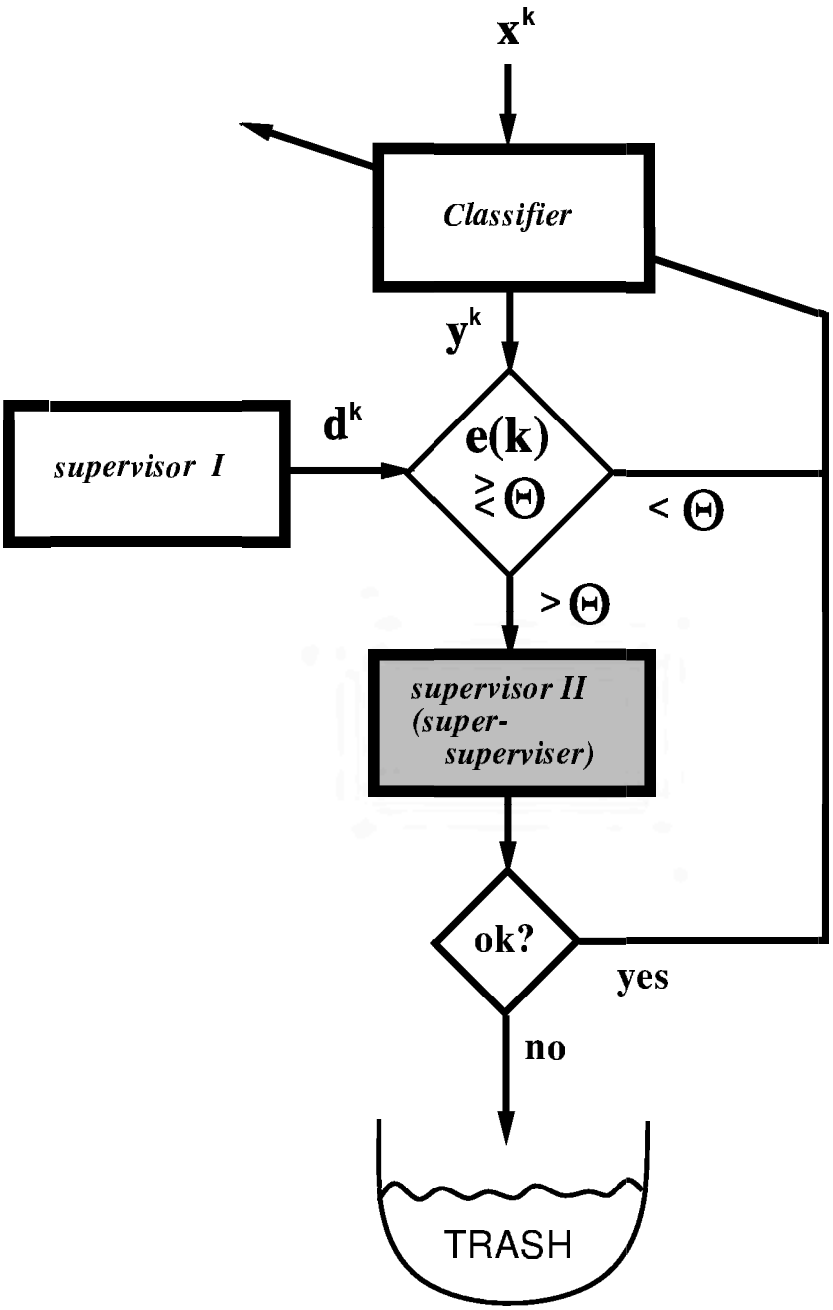


Figure 5: Flow diagram of the interactive version of the cleaning scheme.

and cleaning are completed. Efficiency requires knowing an appropriate value for the threshold $\theta$.

### 4.2 Batch Cleaning

The batch cleaning scheme, in contrast, does not require knowing $\theta$. Training is performed first on all the patterns. An index of predictability of the patterns is then computed, perhaps based on the remaining error $e(k)$ after training. Batch cleaning can be easily combined with the emphasising scheme. In that case, a good index of predictability is the time each pattern spent among the patterns that are hardest to learn (shaded area in figure 3). Patterns are sorted in order of increasing predictability. Supervisor II checks the patterns starting with the least predictable ones. Cleaning can be stopped when undesirable patterns become very rare.

One drawback of batch cleaning is that after cleaning, a new classifier has to be trained with clean data. In principle, this new classifier should be checked to make sure no new "outliers" have turned up.

### 4.3 Bootstrap Cleaning

Third version of the cleaning scheme combines the best features of the interactive and batch versions, is called the bootstrap method. Cleaning and training are performed using a small part of the available data. The resulting network constitutes a partial solution that is highly selective at identifying outliers in a larger subset of the training data. The process can be iterated as necessary, with ever larger amounts of training data, until the whole reservoir of patterns is cleaned. To make supervisor II task more comfortable, we have developed user-interface for bootstrap cleaning, where different cleaning parameters, such as size of pattern chunks, threshold, or number of training iterations can be easily defined.

## 5 Experimental Results

In our application, we consider the problem of writer independent recognition of lowercase isolated letters, entered on a *Touch Terminal* that provides pen trajectory information. A data set of 11513 letters from approximately 250 writers was collected among the staff at AT&T. The contributors were instructed to write a certain set of letters, so the task of supervisor I (preliminary labeling of the data) was trivial — which explains our concern with minimizing the burden on supervisor II. The data set was partitioned into a training set of 9513 characters and a test set of 2000 characters from disjoint set of writers. For batch cleaning, the training session was stopped after approximately 250,000 example presentations.

Our classifier is a Time Delay Neural Network (TDNN) which was described in reference [?]. The TDNN makes use of the pen trajectory information. It is a convolutional multi-layer feed-forward network trained with a gradient descent method minimizing the MSE.

In figure 6 we present the results given by the bootstrap cleaning procedure. The cleaning was done while training the TDNN with plain back-propagation (no emphasizing scheme). The performance on the test set was measured according to the *raw performance*, where the output unit with highest output value is considered to be the decision of the network.

Both training set and test set were cleaned, but, a distinction was made between these two sets. We removed from the test set only *garbage* patterns, i.e. patterns that supervisor II deemed totally meaningless — typically resulting from a hardware failure during data collection. Patterns that were poorly written, highly ambiguous, or mislabeled were kept in the test set, to make our test results as conservative as reasonably possible. The supervisor II was free to decide whether or not questionable characters should be removed from the training set.

Starting with the "uncleaned" database, 3 stages of bootstrap cleaning were applied by the super-supervisor. Each stage was more strict, i.e. he lowered the tolerance for marginal-quality characters. To test the power of the cleaning technique, we also tried to corrupt the initial, uncleaned database, by artificially mislabeling 5% of the characters in the training set.

The results of figure 6 show the evolution of the error rate both on the training set and the test set. While the error rate keeps decreasing on the training set, the error rate on the test set goes through a minimum for a certain level of cleaning. After that point, any additional cleaning will result in an increase of the test error rate. This indicates that the super-supervisor has removed all meaningless and mislabeled patterns and starts removing useful atypical or ambiguous patterns. It is important to be able to detect this phenomenon, to avoid overcleaning. This minimum can be detected with a validation-set (independent from the test-set so as not to bias the final performance result). Another way is to use the predictions of learning theory [?]. In that case, no validation set is needed. This technique is further examplified in section 6.

The emphasizing scheme was run on a freshly initialized TDNN. Experiments were performed with no cleaning and with the optimal cleaning. The resulting improvement is shown by the shaded bar in figure 6. We see that the cleaning plus emphasis reduces the raw error rate almost by half. With unclean data, emphasis is even a little bit worse than the plain vanilla training algorithm. The full advantage of cleaning is therefore reached with the emphasizing scheme.

Batch cleaning was also tried. The index of predictability was the time each pattern spent in the "hard to learn" category during training (shaded area in figure 3). The results are comparable to those obtained with the interactive cleaning, and it has the advantage that the super-supervisor need not sit in front of the machine during training, nor set any *a priori* threshold.

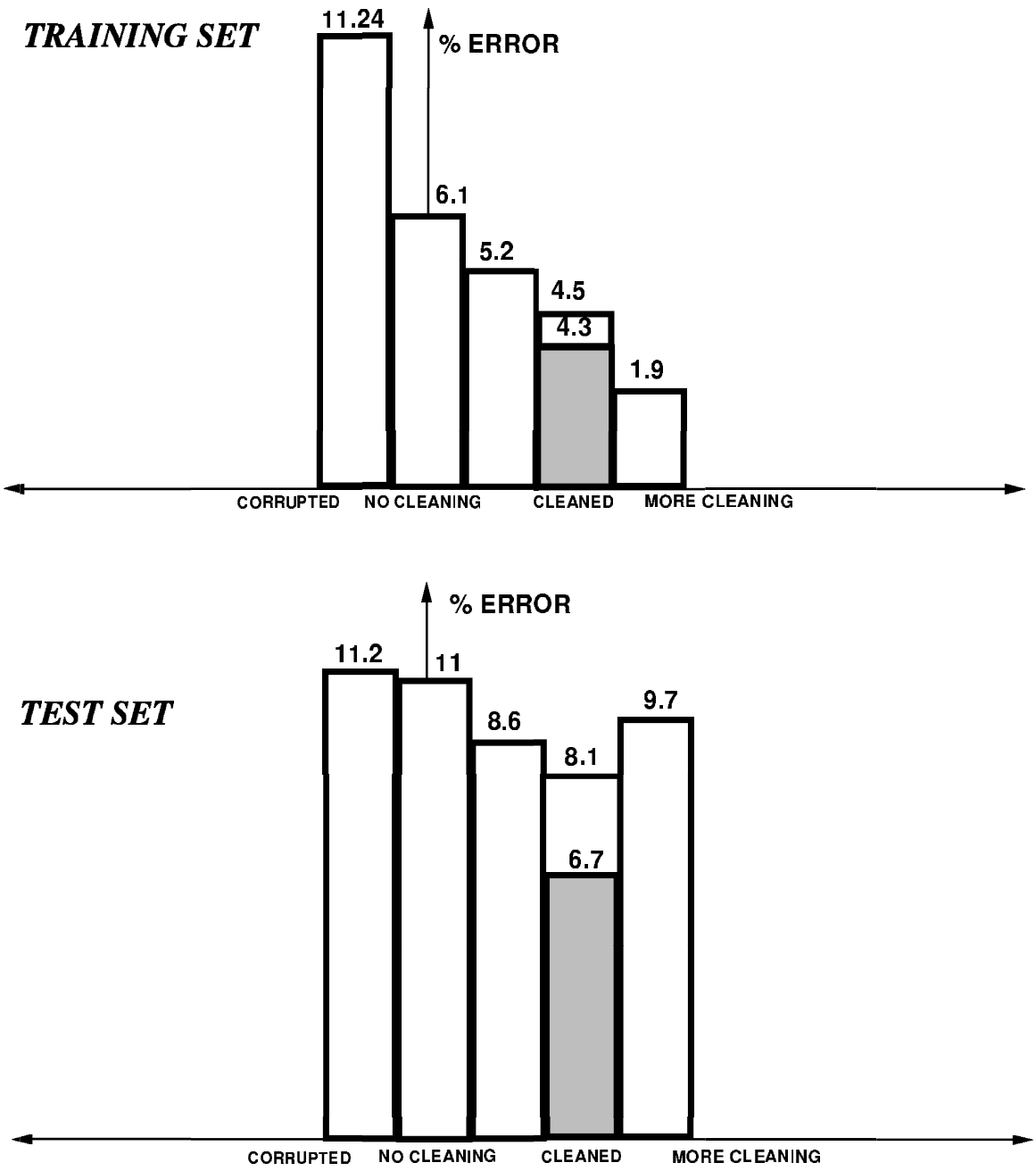


Figure 6: Recognition results for handwritten lowercase letters, for various levels of cleaning. The shaded area indicates the results obtained with the emphasizing scheme.

# 6 Theoretical Foundations of Super-supervised Learning

We now make connections with Bayesian classification and the Vapnik-Chervonenkis (VC) theory [**?**]. We justify our method for cleaning the data based on an index of predictability. We describe a method to avoid overcleaning and check its validity experimentally.

## 6.1 Probabilistic Formalism

Our training problem can be formalized as follows. In an environment characterized by a probability distribution $p(\mathbf{x})$, examples $\mathbf{x}^k$ are drawn randomly and independently. Supervisor I provides class labels $y_l$ with probability $P(y_l|\mathbf{x})$.

In this framework, *atypical* patterns are drawn from regions of input space with small $p(\mathbf{x})$; *ambiguous* patterns correspond to regions of input space with small $P(y_l|\mathbf{x})$ (for instance $P(y_1|\mathbf{x}) \simeq P(y_2|\mathbf{x}) \simeq 0.5$).

In the following, $p(\mathbf{x})$ will be referred to as the "natural" prior distribution and $P(y_l|\mathbf{x})$ as the "natural" posterior probability.

Let us introduce now sources of distortion of these "natural" probabilities so as to account for hardware and human failures in the data collecting process. Hardware failures (*meaningless* patterns) cause distortions of $p(\mathbf{x})$, leading to the "distorted" prior distribution $p'(\mathbf{x})$. Human failures (**mislabeled** patterns) cause distortions of $P(y_l|\mathbf{x})$, leading to the "distorted" posterior probability $P'(y_l|\mathbf{x})$.

*Meaningless* patterns are drawn from regions of input space with eventually large $p'(\mathbf{x})$, but small $p(\mathbf{x})$. *Mislabeled* patterns correspond to regions of input space with eventually large $P'(y_l|\mathbf{x})$, but small $P(y_l|\mathbf{x})$.

Therefore, the four kind of suspicious patterns that we have inventoried correspond to either small $p(\mathbf{x})$ (atypical or meaningless, i.e. outliers) or to small $P(y_l|\mathbf{x})$ (ambiguous or mislabeled). But all the functionals $p(\mathbf{x})$, $p'(\mathbf{x})$, $P(y_l|\mathbf{x})$ and $P'(y_l|\mathbf{x})$ are unkmown and we have to rely only on the empirical data contained in the database itself to perform the cleaning.

## 6.2 Justification of the Cleaning Method

In our approach, we train a classifier $f_l(\mathbf{x})$ in order to obtain an approximation of $P(y_l|\mathbf{x})$. We detect a suspicious pattern $\mathbf{x}^k$ based on an index of predictability involving for instance the error $e(k)$ (see equation 2) which reflects the decrepancy of the opinion of the classifier, $f_l(\mathbf{x}^k)$, and that of supervisor I, $y_l^k$.

---

Using our previous notations, $l$ is the class index. Class labels are $y_l = 1$ if $l = \text{class}(\mathbf{x})$ and 0 otherwise.

From the considerations of the previous section, it is clear that such a method will eventually detect ambiguous and mislabeled patterns $\mathbf{x}^k$ which correspond to small $P(y_l^k|\mathbf{x}^k)$. Less evidently, outlying patterns such as atypical and meaningless patterns, which correspond to small $p(\mathbf{x}^k)$ can also be detected through the same mechanism. The justification comes as follows. It is very difficult to obtain reliable predictions of $P(y_l|\mathbf{x})$ in regions of small $p(\mathbf{x})$, because of the lack of training data. Indeed, with only mild assumptions about the problem at hand, a bound on the error bar is given by:

$$|\delta P(y|\mathbf{x})| \leq \frac{\alpha}{p(\mathbf{x})} |\delta \mathbf{x}| \tag{3}$$

where $\alpha$ is a constant (see appendix 8). This bound shows that the error-bar is bounded by a quantity inversely proportional to $p(x)$. A totallt random prediction would lead to predict the label $y^k$ provided by supervisor I with probability $1/c$ ($c$ being the number of classes). Therefore, $1/c$ is approximately the probability of not detecting an outlying pattern with our method. The method can further be refined by training $N$ claccifiers providing independent predictions, which reduces the probability of not filtering outlyiers to approximately $1/c^N$.

### 6.3 VC-predictions of Optimal Cleaning

Our experimental results indicate that there is a point of optimal cleaning yielding best generalization (see figure 6). In this section, we check experimentally the predictions of optimal cleaning obtained using the VC-bounds **??**.

According to the VC-theory, the probability of making classification errors with our trained classifier is smaller than the following *guaranteed risk*:

$$G = \nu(m) + \beta \frac{(ln\frac{p-m}{d} + 1) + \frac{m}{d}(ln\frac{p}{m} + 1)}{p - m} \tag{4}$$

where $\beta$ is a constant which was experimentally determined by an independent study [**?**] ($\beta = 0.5$), $\nu(m)$ is the frequency of training errors when $m$ suspicious patterns have been removed from the training set, $p$ is the number of training patterns before cleaning and $d$ is the VC-dimension.

This bound is valid for small values of the frequency of training errors $\nu(m)$ and assumes that patterns are removed only from the training set. $G$ predicts an upper boung on the frequency of errors on the original uncleaned test set.

$G$ is the sum of two terms: the frequency of training errors $\nu(m)$, which decreases as the number of removed examples $m$ increases, and a second term which characterizes the confidence with which we know the probability of classification error from $\nu(m)$. This last term increases when we remove training examples. The minimum of $G$ indicates the point of optimal cleaning guaranteed by the VC-theory.

The VC-dimension is difficult to estimate for large neural networks trained with learning schemes involving uncontrolled regularization. In our case, regularization happens in several different ways, including stopping the training session before reaching the minimum of the training error. When initialization is performed with small random weights, this is equivalent to a weight decay regularization [**?**]. If training consited in merely minimizing the number of errors on the training set, a good approximation of the VC-dimension would be the number of free parameters $N_{free}$. In the case of the TDNN, this number is the number of independent weights, not the number of connections. Because of regularization, the *effective* VC-dimension is only a fraction of $N_{free}$. Based on experiments performed in [**?**], we use the following heuristic formula:

$$d = \frac{N_{free}}{3} \ . \tag{5}$$

We performed the following experiments to check the validity of this result. Training of the TDNN was performed with all the training data (including garbage patterns) using the emphasizing scheme. Patterns were then ranked according to decreasing index of predictability. A set of suspicious patterns selected among the top ranked patterns (least predictable) was removed from the training set. A freshly initialized TDNN was trained with the remaining patterns. Then, suspicious patterns were progressively incorporated in the training set, starting from the most predictable ones (least suspicious). A each step, partial retraining of the previous TDNN was performed with the clean training set augmented with some suspicious patterns.

Two different definitions of the set of suspicious patterns were considered: (i) the top 25% of the least predictable patterns (dumb cleaning); (ii) the top 25% of least predictable patterns that had been identified as suspicious by supervisor II in our previous study (smart cleaning). In figure **??** we show the results obtained in both cases.

The minimum of $G$ coincides with the best generalization performance measured with the test error ?!?!?!?!?!?!? The best test error is however larger than the one presented in section 5: this is due to the fact that the test set contains here all the garbage patterns, including hardware failures.

# 7 Conclusions

By filtering suspicious patterns and querying a super-supervisor, we have demonstrated clear improvement in our handwritten lowercase letter training database. For the writer independent task, the final recognition rate on the test set was 93.1% (raw performance; no rejection allowed), and 95% when 3.2% of the patterns were rejected as unclassifiable, and 97% with 6.5% rejection. These results are better than the ones reported in a recent review paper [**?**] on a comparable task. Such high recognition rates could not have been achieved without the learning scheme

which emphasizes ambiguous and atypical patterns, or without accurate cleaning of the training database.

## 8 Acknowlegments

The authors thank all the Neural Network research group headed by L. Jackel at Bell Labs, Holmdel, for their suggestions regarding this paper. The simulations were performed using Neural Network simulator SN of Leon Bottou and Yann Le Cun.

In this appendix, we justify that the error-bar on the prediction of the posterior probability $P(y_l|\mathbf{x})$ is bounded by an quantity inversely proportional to the prior distribution $p(\mathbf{x})$.

According to the bayes formula:

$$P(y_l|\mathbf{x}) = \frac{p(\mathbf{x}|y_l)P(y_l)}{p(\mathbf{x})} \ . \tag{6}$$

By differentiating:

$$dP(y_l|\mathbf{x}) = \frac{P(y_l)}{p(\mathbf{x})^2} dp(\mathbf{x}|y_l)p(\mathbf{x}) - dp(\mathbf{x})p(\mathbf{x}|y_l) \ . \tag{7}$$

We perform the following majoration:

$$\max_l \left( \left| P(y_l) \frac{dp(\mathbf{x}|y_l)}{d\mathbf{x}} \right| \right) \leq \alpha \ . \tag{8}$$

Because

$$p(\mathbf{x}) = \sum_{l=1} cp(\mathbf{x}|y_l)P(y_l) \ , \tag{9}$$

we have also

$$\left| \frac{dp(\mathbf{x})}{d\mathbf{x}} \right| \leq c\alpha \ , \tag{10}$$

and, for all $l$,

$$p(\mathbf{x}|y_l)P(y_l) \leq p(\mathbf{x}) \ . \tag{11}$$

Hence, from equations 7, 10 and 11,

$$|dP(y_l|\mathbf{x})| \leq \frac{(c+1)\alpha}{p(\mathbf{x})} |d\mathbf{x}| \ . \tag{12}$$