A Hybrid System for the Recognition of Hand-Written Characters

Rudolf Freund, Markus Neubauer, Martin Summerer, Stefan Gruber,
Jürgen Schaffer, and Roland Swoboda

Institute for Computer Languages
Vienna University of Technology

Karlsplatz 13,
A-1040 Wien, Austria

udi@logic.at

Tel.: ++43 1 58801 18542, Fax: ++43 1 58801 18597

Abstract. In this paper we introduce a new hybrid system for the automated recognition of hand-written characters - we combine the most promising approaches of the last decade, i.e., neural networks and structural/syntactical analysis methods. The given patterns represent hand-written capital letters and digits stored in arrays. The first part of the hybrid system consists of the implementation of a neural network and yields a rapid and reliable pre-selection of the most probable characters the given pattern may represent. Depending on the quality and the special characteristics of the given pattern a flexible set of characters is communicated to the second part of the hybrid system, the structural analysis module. The final decision is based on the evaluation of the presence of features, being characteristic for a specific character, in the underlying pattern. Basically, the structural analysis module consists of graph controlled array grammar systems using prescribed teams of productions. We describe the main parts of the implemented hybrid system and demonstrate the power of our approach.

Key words: array grammars, character recognition, hybrid systems, neural networks, structural analysis

1 Introduction

The recognition of handwritten characters is still an important issue in automatically processing forms and vouchers. Although many systems are already used in various fields and sometimes are considered as hardly improvable, recognition rates in existing applications show that additional human review and postprocessing still plays an important and expensive rôle. Especially for recognition tasks where no context information can be taken into account for verifying the result of the automated recognition system, improving the reliability of the recognition procedure still constitutes a significant step forward. Thus our system is especially designed to be used in banks for automated processing of vouchers and recognition of hand-written data in fill-in-forms.

As the last years were dominated by research in neural networks (e.g., cf. [2]) showing great new ideas, we wanted to go one step further and integrate them in a hybrid system together with a model using structural analysis (e.g., see [13]). As started in previous works (e.g., see [7]), our main intention was to find a way of enforcing the advantages of both approaches and compensating their shortcomings. While in [7] syntactical and statistical features were used as input for a statistical classifier, we implemented a backpropagation neural network (cf. [2]) for a fast pre-selection to speed up the final classifier component, which was implemented based on the theoretical model of graph controlled array grammars with prescribed teams of productions; these array grammars are designed in such a way that they come up with an exact analysis of the detected features. The main idea behind our model is to emphasize on the reliability of the final classifier and to overcome the lack of speed of the syntactical analysis by the pre-selection carried out by the neural network.

The paper is organized as follows. In the next section we describe the main parts of the hybrid system and explain our motivation for its design. In Section 3 we present the statistical pre-selection module based on neural networks and describe the flexible interface between this statistical pre-selection module and the final syntactical classifier (the structural analysis module). Section 4 deals with the structural analysis module and presents its main ideas and advantages. Finally, in Section 5 we give a short overview on the results obtained with the prototype of our hybrid system as well as an outlook to future work.

2 The Hybrid System

The complete system was especially designed for the recognition of hand-written capital letters and digits as they are used in fill-in-forms and vouchers, where the single letters are found on prescribed fixed positions and no overlapping has to be taken into account.

2.1 Data Acquisition and Preprocessing

The data basis of hand-written characters we used was acquired by Arnold ([1]) from hundreds of persons on specific forms. The scanned characters first were normalized to fill out a 320×400 grid in order to get comparable patterns. After the elimination of noisy pixels, the resulting arrays on the 320×400 grid were mapped on a 20×25 grid. These arrays on the 20×25 grid then were subjected to a thinning algorithm (e.g., see [11], [14]) which finally yielded unitary skeletons of the digitized characters.

2.2 Motivation for Designing a Hybrid System

Regarding the set of characters we consider and the sub-patterns they consist of, obviously there are different degrees of similarity among characters. The quality of classifiers also varies depending on this degree of similarity. Hence,

our intention was to combine two different methods, the first one specialized on excluding all characters with low similarity to the underlying pattern, and the second one concentrating on working out very detailed informations from the pattern and its closeness to the remaining candidate characters.

The neural network module. The task of the statistical module is to provide a group of candidate characters with high similarity to the given pattern. As this goal should be achieved very quickly, a neural network seemed to be appropriate for this task. We require that it must not exclude the correct solution, therefore its primary task is not to provide the correct result but to eliminate all characters that significantly differ from the given pattern.

The structural analysis module. The output of the neural net - a reduced number of candidate characters - is transferred to the final classifier. The method used there has to provide the possibility to check for very small details, because the differences between similar characters tend to consist of only some pixels. Structural analysis based on array grammars can be a powerful approach to this task (e.g., see [13], [6]); hence, we developed a special array grammar for each character that analyses the pattern with respect to its inherent features of the character and the differences to other ones. The main problem of syntactical analysis usually is the inherent non-determinism leading to unpredictable processing times in the worst-case behaviour. Thus we enriched the array grammars with several control mechanisms, which allowed us to eliminate the undesirable non-determinism.

3 The Neural Network Module

The reason for needing a fast pre-selection is the way how structural analysis works: As the final classifier analyses the pattern with respect to each possible character, it also tries to find characters in patterns that do not show the characteristic features of these characters, which results in a big overhead in processing time. Neural networks provide the solution for this problem, because we get a fast elimination of characters that are obviously not represented by the given pattern and therefore need not be analysed in every little detail. For the remaining candidate characters we can use the more reliable and exact - but slower - structural recognition module to make a final decision.

The most important aspect is the reliability of the recognition achieved by the neural net. As we design a hybrid system, there are two recognition results, and the final result is the combination of both. Although the motivation of the network was a speed-up of the system, we had to put emphasis on the reliability of the output, to guarantee the possibility of a correct classification by the structural analysis. This means that it is not so important, how many characters are excluded, but that the searched one is within the result of the pre-selection. There have been made a lot of studies about which recognition rate

can be reached by using neural network classifiers, but the very special interest in our neural net is the reverse performance, i.e., the reliability not to exclude the character most likely shown in the pattern.

3.1 Input Vector

Normal backpropagation with one input unit for every pixel leads to very poor results, as it is very hard for the net to develop internal representations for structures like lines and edges. Therefore we use structured backpropagation where we pre-process the shown pattern to detect features and to reduce the dimension of the input. The task is to reduce the number of input units from 500 (each pixel representing one input unit) down to a reasonable number without losing essential information about the pattern.

Further we have to try to structure the input vector in such a way that information about the positions of pixels and their neighbourhood relations to other pixels are extracted, which is a kind of information neural nets are hardly able to extract from pixel-dependent inputs. Therefore we choose the following inputs: One part of the input vector consists of the pixel sums per column and row. This strategy brings good values for characters with straight lines, as H or F, but it is weak in extracting the features of characters consisting of curves as C or O. Therefore the second part of the input vector consists of the number of contiguous pixel groups per line for each column and row, which is an indicator for the number of lines cut in the corresponding row or column, respectively. Applying these ideas we reduce the dimension of the input vector from 500 to 90 and achieve a spatial structuring of the input data.

3.2 Communicating the Neural Network Result

The only interface between the neural network component and the array grammar component consists of a simple linked list of records containing pre-selected characters and their probability. There is no need for sorting, because all the selected characters are processed anyway.

The probability that the shown character is a member of the selected group has to be very close to hundred percent. If this were not guaranteed, the exact analysis would make no sense, as it could never bring a correct result. The result of the neural network analysis is a value (= probability) of each output unit (= character). When using a softmax activation function, those values can be interpreted as percentages. Depending on those percentages we have to make a decision, which units (characters) are fitting best and therefore have to be handed over to the final analysis. For the selection of the candidate characters computed by the neural network we considered several different strategies:

Selection down to a percentage limit. The simplest possibility is to select all units whose percentage is higher than a fixed bias. The advantage of this method is its simplicity and clearness. The average number of chosen letters can

easily be alternated by shifting the limit, results are easy to predict. In addition, the values need not to be sorted before we make our decision, as it only depends on the predefined limit. If no unit reaches this limit, we simply choose all the characters as possible candidates. But there are some serious disadvantages, as the distribution of the values is ignored, e.g., if there is a flat distribution with percentages around the limit, small differences can have an unacceptable great influence on the result.

Selection by summing up to a percentage sum. This method avoids the problems of the one above in a simple way. We first sort the output-values in descending order, and then sum up the percentages assigned to the selected characters, until we reach a certain percentage sum limit. The average number of chosen characters can easily be altered by shifting the limit again, but a flat distribution results in a larger set of possible candidates.

Selection of a predefined group of characters. Recognition by neural networks tends to determine groups of characters having significant similarities between them. To be sure that all the characters that are rather similar for the neural network and therefore hard to distinguish, are communicated to further structural analysis, we implemented a kind of group selection. One method for determining suitable groups is obtained by doing a statistical evaluation of the results of the structural analysis. Another method is to create a test set for the trained neural network consisting of very clearly and correctly written characters and to evaluate which characters look similar for the trained network. Combining the results of these two methods resulted in a system with pretty good reliability. These results can also be used to create a new network structure, where we do not use one output-unit for exactly one character, but one for each group of similar letters. Thus one character can be member of more than one group, which brings more flexibility into the groups (e.g., F is similar to E; in a different way F is also similar to P, but E and P are easy to distinguish).

4 The Structural Analysis Module

If emphasis is put on reliability, the most promising method for recognizing characters is to detect their structural features in a very exact way. This becomes the more important the sloppier the characters are written. When a given pattern is very similar to several characters, distinguishing them can depend on tiny lines or small differences in distances between lines and endpoints. Hence, the structural analysis module is designed to decide between quite similar characters by using very detailed information about the pattern and its analysed lines. As shown in [6], array grammars are a very promising method for syntactical character recognition; we further developed the ideas discussed there and could reduce parsing complexity especially by eliminating non-determinism (see [8]).

4.1 Arrays and Array Grammars

Following [3], [5], [6], [10], and [12], we introduce the following definitions and notations for (two-dimensional) arrays and array grammars.

An *array* over the alphabet V is a function from Z^2 to $V \cup \{\#\}$; $\#$ is the blank symbol. The set of all arrays over V be denoted by V^{*2} . An *array grammar* is a structure $G = (V_N, V_T, \#, P, \{(v_0, S)\})$, where V_N is the alphabet of *non-terminal symbols*, V_T is the alphabet of *terminal symbols*, $V_N \cap V_T = \emptyset$, $\# \notin V_N \cup V_T$; P is a finite non-empty set of array productions over $V_N \cup V_T$ and $\{(v_0, S)\}$ is the *start array (axiom)*, v_0 is the start vector, and S is the *start symbol*. We say that the array $\mathcal{B}_2 \in V^{*2}$ is *directly derivable* from the array $\mathcal{B}_1 \in V^{*2}$ by the array production $p \in P$, denoted $\mathcal{B}_1 \Rightarrow_p \mathcal{B}_2$, if and only if the application of p to $\mathcal{B}_1$ yields $\mathcal{B}_2$.

Control mechanisms. As shown in [5], control mechanisms (cf. [4]) as control graphs are a suitable way to enhance the power of a grammar and can be applied in the field of character recognition as well (see [6]). Nevertheless our work showed that by using graph controlled array grammars in combination with sets of prescribed teams it is possible to eliminate non-determinism, especially in crucial situations with crossing points of lines.

Graph controlled array grammars. A *graph controlled array grammar* is a construct $G_P = (V_N, V_T, \#, (R, L_{in}, L_{fin}), \{(v_0, S)\})$, where V_N and V_T are disjoint alphabets of non-terminal and terminal symbols, respectively; v_0 is the start vector, $S \in V_N$ is the start symbol; R is a finite set of rules r of the form $(l(r) : p(l(r)), \sigma(l(r)), \varphi(l(r)))$, with $l(r) \in Lab(G_P)$, $Lab(G_P)$ being a set of labels associated (in a one-to-one manner) to the rules r in R , where $p(l(r))$ is a set of array productions over $V_N \cup V_T$, $\sigma(l(r)) \subseteq Lab(G_P)$ is the *success field* of the rule r , and $\varphi(l(r))$ is the *failure field* of the rule r ; $L_{in} \subseteq Lab(G_P)$ is the set of initial labels, and $L_{fin} \subseteq Lab(G_P)$ is the set of final labels. For $r = (l(r) : p(l(r)), \sigma(l(r)), \varphi(l(r)))$ and $v, w \in (V_N \cup V_T)^{*2}$ we define $(v, l(r)) \Rightarrow_{G_P} (w, t)$ if and only if

- **either** an array production in $p(l(r))$ is applicable to v , the result of the application of this array production to v is w , and $t \in \sigma(l(r))$,
- **or** no array production in $p(l(r))$ is applicable to v , $w = v$, and $t \in \varphi(l(r))$.

Attribute vectors. In order to be able to store information about lines and crossing points (e.g., length, positions, etc.) we use additional attribute vectors assigned to each sentential form and therefore have to add a corresponding manipulator function f (to each array production) which changes the attribute vector. Formally this means that instead of $p(l(r))$ we now have $(p(l(r)), f(l(r)))$; $(p(l(r)), f(l(r)))$ is applicable to a configuration $(\mathcal{A}, v)$ yielding $(\mathcal{B}, w)$, i.e., $(\mathcal{A}, v) \Rightarrow_{(p(l(r)), f(l(r)))} (\mathcal{B}, w)$, if and only if $\mathcal{A} \Rightarrow_{p(l(r))} \mathcal{B}$ and $f(l(r))(v) = w$. With the information stored in the attribute vector it is possible to transparently

implement conditional branching according to the applicability of the manipulator function f .

Prescribed teams. Although from a theoretical point of view the power of graph controlled array grammars cannot be enhanced further (cf. [5]), we decided to use prescribed teams (e.g., see [9]) as additional control mechanism in order to eliminate non-determinism as well as to reduce parsing complexity while analysing parallel lines of equal lengths. For our purposes we could restrict ourselves to the case of (graph controlled) array grammars with prescribed teams of finite index:

Let M be an arbitrary set; then any object $\{(x, n_x) \mid x \in M\}$ with n_x being a natural number for every $x \in M$ is called a *multiset* over M . Now let $G = (V_N, V_T, \#, P, \{(v_0, S)\})$ be an array grammar. Any finite multiset s over P is called a *prescribed team over P* ; we shall also write s in the form $\langle p_1, \dots, p_m \rangle$, where the p_i , $1 \leq i \leq m$, are exactly the array productions in s occurring in the right multiplicity. The array productions p_i of the team $\langle p_1, \dots, p_m \rangle$ are applied in parallel to the underlying sentential form. In our special case of character recognition we especially use teams to define lines that are analysed in parallel, i.e., in this case the number m represents the number of lines analysed in parallel in one derivation step.

For adding prescribed teams of finite index to our graph controlled array grammars with attribute vectors we allow $p(l(r))$ in a rule r to be a set of prescribed teams of array productions of the form $\langle p_1, \dots, p_m \rangle$.

Our final model of array grammars. The final model of array grammars we used is that of a *graph controlled array grammars with attribute vectors and prescribed teams of finite index*

$$G_P = (V_N, V_T, \#, (R, L_{in}, L_{fin}), \{(v_0, S)\}, a),$$

where V_N and V_T are disjoint alphabets of non-terminal and terminal symbols, respectively; v_0 is the start vector, $S \in V_N$ is the start symbol; a is the initial attribute vector; R is a finite set of rules r of the form $(l(r) : \{(p_i(l(r)), f_i(l(r))) \mid 1 \leq i \leq k\}, \sigma(l(r)), \varphi(l(r)))$, where $l(r) \in Lab(G_P)$, $p_i(l(r))$ is a prescribed team of array productions over $V_N \cup V_T$ of the form $\langle p_1, \dots, p_m \rangle$, $f_i(l(r))$ is a manipulator function that operates on the attribute vector, $1 \leq i \leq k$; $\sigma(l(r)) \subseteq Lab(G_P)$ is the *success field* of the rule r , and $\varphi(l(r))$ is the *failure field* of the rule r ; $L_{in} \subseteq Lab(G_P)$ is the set of initial labels, and $L_{fin} \subseteq Lab(G_P)$ is the set of final labels.

Based on this theoretical model of array grammars defined above, we obtained a powerful and efficient tool for character recognition (see [8]) with rather small parsing complexity especially due to the avoidance of non-deterministic decisions at crossing points and at holes because of missing pixels in lines; moreover, the information stored in the attribute vector about lines and crossing points allowed us to make a detailed comparative analysis of the patterns with respect to a character and to define suitable error measures.

4.2 Implementation

Practically our approach intends to process every characteristic line and to store its values. Therefore we first have to look for a reliable start point (SP), where the analysis of the whole character has to begin from. When we process a line, we go on until we reach its end, storing the values as its length and the positions of the starting and the end point as well as the attributes of all crossing-points found along the way in the attribute vector. Following this strategy, we start the first set of prescribed teams of array productions representing the first lines to be analysed until no team of productions is applicable any more. Then the start points for the next set of prescribed teams of array productions are calculated using the values stored in the attribute vector, and then the array grammar continues until all lines have been analysed. At the end, an evaluation function calculates an error value for the character using the information stored in the attribute vector.

As an example we depict the control diagrams of the array grammars implemented for the characters F and P:

The advantage of our approach is that we get very exact values about the different parts of the characters. Hence, if within a group of several similar letters it is necessary, we can concentrate our analysis on very small details and draw sharp borders between different characters, which in this special case of characters F and P are the distances of the end points of the horizontal lines on the right hand side. The final evaluation takes into account the coordinates of these most interesting end points and calculates the exact distance, which has a great influence on the calculation of the error value and therefore on the final decision.

5 Conclusion

Our experiments with a prototype have shown that combining statistical and structural methods in a hybrid system allows for making use of the advantages of both approaches and for overcoming most of their shortcomings. The neural network together with the chosen selection policies turned out to provide a reliable and fast exclusion of obviously different characters. The graph controlled array grammar systems using prescribed teams represent a powerful tool for the detection and analysis of structural features in hand-written uppercase letters and digits. The main advantages concern the lack of non-determinism thus avoiding backtracking during processing, the exactness of capturing detailed features of patterns, and the powerful methods for evaluation.

5.1 Results

With the selection policy using the group model, the neural network module achieved a nearly hundred percent reliable exclusion of a set of characters obviously not representing the underlying pattern. The number of characters communicated to the structural analysis module varied with the quality of the pattern but never exceeded 9. Using the policy of summing up to a given percentage sum, the set of candidate characters could be reduced to 4 even in the worst case, but then the reliability was reduced to 94 percent.

With the subsequent structural analysis we could obtain a final recognition rate of nearly hundred percent; failures only occurred with patterns that were hardly recognizable even for human beings without any further context informations, which is exactly the situation our system was defined for. Due to the elimination of characters with little similarity by the neural network module, we could overcome problems occurring with the structural analysis component in the case of patterns with large holes in characteristic lines where sometimes due to these holes the whole group of characters with high similarity was classified worse than other characters with less similarity (e.g., misclassification of a pattern as character 3 instead of B because of holes in the vertical line).

5.2 Future Research

As the hybrid system is mainly designed for banking applications, the set of characters has to be extended by special symbols like those for special currencies. Another interesting idea is to automatically retrain the neural network using the results of the final classification of the underlying patterns by the structural analysis module.

References

1. H. Arnold, *Erkennung von handgeschriebenen Buchstaben mittels Entscheidungsstrategie, Neuraler Netze und BP-Untergruppen*, Thesis, Technical University Vienna, 1994.
2. C.M. Bishop, *Neural Networks for Pattern Recognition*, Clarendon Press, Oxford, 1995.
3. C.R. Cook and P.S.-P. Wang, A Chomsky hierarchy of isotonic array grammars and languages, *Computer Graphics and Image Processing* **8** (1978), pp. 144–152.
4. J. Dassow and Gh. Păun, *Regulated Rewriting in Formal Language Theory* (Springer, Berlin, 1989).
5. H. Fernau, R. Freund, and Markus Holzer, Regulated array grammars of finite index, Part I: Theoretical investigations. In: Gh. Păun and A. Salomaa (eds.), *Grammatical Models of Multi-Agent Systems*, Gordon and Breach, Reading, UK, 1998.
6. H. Fernau, R. Freund, and M. Holzer, Character recognition with k-head finite automata. In: A. Amin, D. Dori, P. Pudil, H. Freeman (eds.), *Advances in Pattern Recognition (Proceedings SSPR'98)* (Springer, Berlin, 1998), pp. 282–291.
7. L. Heutte, J.V. Moreau, T. Paquet, Y. Lecourtier, and C. Oliver, Combining Structural and Statistical Features for the Recognition of Handwritten Characters, *Proceedings of the thirteenth international Conference on Pattern Recognition*, Vol. **2**, pp. 210–214, ed. IEEE Comp.Soc., Vienna, Austria, 1996.
8. M. Neubauer and M. Summerer, *Character recognition based on structural feature detection using array grammar systems*, Thesis, Vienna University of Technology, 1999.
9. Gh. Păun and G. Rozenberg, Prescribed teams of grammars, *Acta Informatica* **31**, 6 (1994), 525–537.
10. A. Rosenfeld, *Picture Languages* (Academic Press, Reading, MA, 1979).
11. J. H. Sossa, An improved parallel algorithm for thinning digital patterns, *Pattern Recognition Letters* **10** (1989), pp. 77–80.
12. P. S.-P. Wang, Some new results on isotonic array grammars, *Information Processing Letters* **10** (1980), pp. 129–131.
13. P. S.-P. Wang, An application of array grammars to clustering analysis for syntactic patterns, *Pattern Recognition* **17**, 4 (1984), pp. 441 – 451.
14. Y.Y. Zhang, P.S.P. Wang: A Parallel Thinning Algorithm with Two-Subiteration that Generates One-Pixel-Wide Skeletons, *Proceedings of the thirteenth international Conference on Pattern Recognition*, Vol. **4**, pp. 457–461, ed. IEEE Comp.Soc., Vienna, Austria, 1996.