SYSTEM DESIGN OF AN INTEGRATED PATTERN RECOGNITION SYSTEM OR HOW TO GET THE BEST MILEAGE OUT OF YOUR USED PATTERN CLASSIFIER

A. R. Hanson*
E. M. Riseman†

COINS Technical Report 73C-5
June 1973

This research was supported by the Office of Naval Research under Grant ONR 049-332

*Department of Computer, Information, and Control Sciences
University of Minnesota
Minneapolis, Minnesota

†Department of Computer and Information Science
University of Massachusetts
Amherst, Massachusetts

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*

University of Massachusetts, Amherst, Mass. 01002

2a. REPORT SECURITY CLASSIFICATION

UNCLASSIFIED

2b. GROUP

None

3. REPORT TITLE

System Design of an Integrated Pattern Recognition System
or How to Get the Best Mileage out of Your Used Pattern Classifier

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

Technical Report

5. AUTHOR(S) *(First name, middle initial, last name)*

Allen R. Hanson, Edward M. Riseman

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
June 1973	26	18

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
N00014-67-A-0230-0007	COINS Technical Report 73C-5
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d.	

10. DISTRIBUTION STATEMENT

Distribution of this document is unlimited

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
None	Office of Naval Research, Code 437 Washington, D.C.

13. ABSTRACT

This paper examines the conceptual design of a character recognition system. The overall classification system is viewed as a network of three subsystems: a feature extractor, a classifier, and a contextual postprocessor. Some of the design dependencies that exist between these subsystems are always utilized in the construction of a pattern classifier. However, there are many instances where the subsystems are designed independently, ignoring the dependencies existing between them. When viewed in perspective, these dependencies are not as complex as might be expected and their consideration can result in substantial improvements in system performance.

Integration of the network is achieved by two mechanisms: 1) introducing additional paths of data flow, and 2) modifying the design of each of the subsystems. This allows alternative methods for processing a pattern at different stages in the recognition process.

The second means of integrating the pattern classifier is the main contribution of this paper. An attempt is made to redesign each of the three subsystems on the basis of the design and performance of the other two subsystems. For example, the design of the feature extractor involves the selection of the particular features to be employed. These features can be selected to yield more information for those class pairs whose confusion will cause the contextual postprocessor difficulty in its attempt to correct the error. In this way the feature set is designed to compensate for the weaknesses in the postprocessor. This type of design dependency is examined for every pair of subsystems. Suggestions are made to incorporate knowledge of these dependencies into an overall design procedure.

ABSTRACT

This paper examines the conceptual design of a character recognition system. The overall classification system is viewed as a network of three subsystems: a feature extractor, a classifier, and a contextual postprocessor. Some of the design dependencies that exist between these subsystems are always utilized in the construction of a pattern classifier. However, there are many instances where the subsystems are designed independently, ignoring the dependencies existing between them. When viewed in perspective, these dependencies are not as complex as might be expected and their consideration can result in substantial improvements in system performance.

Integration of the network is achieved by two mechanisms: 1) introducing additional paths of data flow, and 2) modifying the design of each of the subsystems. The first technique involves providing several feedback paths between the subsystems. This allows alternative methods for processing a pattern at different stages in the recognition process.

The second means of integrating the pattern classifier is the main contribution of this paper. An attempt is made to redesign each of the three subsystems on the basis of the design and performance of the other two subsystems. For example, the design of the feature extractor involves the selection of the particular features to be employed. These features can be selected to yield more information for those class pairs whose confusion will cause the contextual postprocessor difficulty in its attempt to correct the error. In this way the feature set is designed to compensate for the weaknesses in the postprocessor. This type of design dependency is examined for every pair of subsystems. Suggestions are made to incorporate knowledge of these dependencies into an overall design procedure.

I. INTRODUCTION

The construction of many classification systems have proceeded by designing each component or subsystem very carefully. However, the optimal design of a system is not usually achieved by an independent optimal design of each of the subsystems unless they are actually performing independent processes. Independent subprocesses are usually a figment of the designer's imagination, but the complexity of existing dependencies has led many researchers to ignore these effects. However there are a number of ways to allow the the subsystems to interact in order to achieve more effective recognition rates. Even more important is the design of each subsystem with the knowledge that it is to be part of a system with other subsystems whose desired general operation is understood. The ideas presented in this paper apply to the problem of pattern recognition in general, but they will be more carefully discussed in the context of the design of an integrated character recognition system.

The character recognition systems that will be examined consist of three subsystems: a feature extractor, a classifier, and a contextual postprocessor. The feature extractor is the measurement subsystem; this component determines the values of the feature set when a pattern is presented. It should be noted that the design of the feature extractor involves the determination of the set of features by a suitable selection algorithm. Thus, the feature extractor should be distinguished from the process of feature selection. The classifier is the decision module to decide upon the identity of the sample. The contextual postprocessor (or CPP) then attempts to detect and correct errors produced by the classifier.

Throughout the following discussion, we assume that any required pre-

processing of the patterns, such as centering, digitizing, size and shear normalization, etc. has been accomplished. We shall ignore the problems associated with these preprocessing operations, while recognizing their importance.

There are two perspectives from which we will view the pattern classification systems considered: 1) the manner in which the subsystems interact through data flow and 2) the way each subsystem is affected by the design of each of the other two discrete subsystems. Integration of the design of the total system will be achieved both by introducing additional paths of data flow and by appropriately modifying the design or design procedure of each subsystem to take advantage of dependencies between the subsystems.

The first technique involves providing several feedback paths between the subsystems. This allows alternative methods for processing a pattern at different stages in the recognition process. For example, if the contextual postprocessor detects an error that it cannot correct, the following alternative feedback actions will be useful: 1) request the classifier to reclassify among the subset (hopefully small) of the classes specified by the postprocessor; 2) request the measurement of additional specific features to resolve confusion between a subset of the classes specified by the postprocessor.

The second means of integrating the pattern classifier involves redesigning each of the three subsystems on the basis of the design and performance of the others. As an example, consider the influence of the postprocessor on the design of the feature extractor. The contextual postprocessor has difficulty correcting some types of errors and little difficulty with others. The feature selection algorithm can be modified to produce a feature set which

yields more information for those class pairs whose confusion will cause the postprocessor difficulty. In this way the feature set is designed to compensate for the weaknesses of the postprocessor. Such design dependencies are examined for every pair of subsystems.

The paper initially examines the usual classification system consisting of just a feature extractor followed by a classifier. The general design considerations for each of these subsystems and the manner in which they interact is analyzed. Then an effort is made to integrate the system design of this model. Finally, a contextual postprocessor is added and an effort is made to integrate the entire design by the procedures that have been outlined.

II. A SIMPLE CLASSIFICATION SYSTEM

First let us examine the simplest model of a classification system shown in Figure 1.

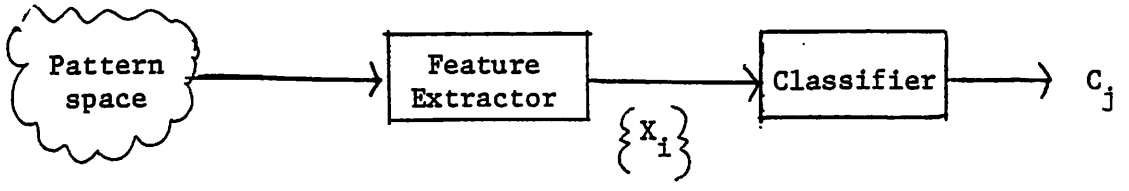

Figure 1. A Simple Classification System

For any pattern presented to the classification system the values of a set of features, previously selected by some appropriate feature selection algorithm, describe a transformed pattern vector $\overline{X} = (X_1, \ldots, X_n)$; the classifier's job is to output the proper class, C_j. Although there are many types of classification methods, the most frequently used is the Bayesian classifier which chooses the class j for which $\underset{j}{\text{Max}}\ P(\overline{X}|Cj)\ P(Cj)$ is satisfied.

This requires the estimation of the probabilities $P(Cj)$, which are easy to obtain, and the densities $P(\overline{X}|Cj)$, which are not. Because the number of measurements is normally large, determination of $P(\overline{X}|Cj)$ is usually prohibitive. The standard assumption, and one which is almost always not true, is that the measurements are independent. Thus, $P(\overline{X}|Cj)$ becomes $\prod_i P(\overline{X}_i|Cj)$ which is far easier to measure and store. Depending upon the degree of validity of the independence assumption, the result is almost always a varying increase in the error rate. One way to offset this undesirable deterioration of performance is the inclusion of additional features. However, if the features exhibit sufficiently strong dependencies there will be less improvement than expected by the addition of extra features; in the extreme, addition of enough features can actually result in an increased error rate due to the gross violation of the independence assumption [1-3]. Probably the most important factor limiting the inclusion of more features, though, is the additional expense incurred, in terms of either extra hardware and/or increased computation per pattern which could reduce the processing rate of the system. Obviously there is a balance to be reached between the system cost and the system error rate.

The paths of data flow of the simple system in Figure 1 are straightforward and devoid of feedback. A pattern is presented to the system, the feature measurement subsystem accepts the pattern, collects its information and passes this information (i.e., the transformed pattern $\overline{X}$) to the classification subsystem which produces as output the class to which the input pattern (hopefully) belongs. The next two sections will describe the design dependencies of each of the two subsystems upon the other.

II.1 DESIGN OF THE CLASSIFIER SUBSYSTEM

Let us clearly understand how the classifier, in this simple system, is designed. It is obvious that any classifier must be heavily designed on the basis of knowledge concerning the feature measurements. In our previous discussion we pointed out that the classifier fundamentally employs class conditional probability densities $P(\overline{X}|Cj)$ in the decision process. Much has appeared in the literature concerning the statistical estimation of the means and variances of the underlying probability densities (see, for example, Maisel [4], Patrick [5]). Thus, the classifier both receives processed information from the feature extractor (path of data flow) during its operation and is designed on the basis of the estimated statistics of this feature measurement process (design dependency).

An examination of the error characteristics of the classifier yields further insight into the relationship between the feature selection algorithm and the classification scheme. Any classifier will be prone to errors i.e., the classifier will mistakenly identify a particular class as another class. This error manifests itself as a probability of confusing various classes; thus, for every class pair C_iC_j, there is some probability (perhaps zero) of confusing C_j for C_i. Those class pairs for which the probability of confusion is high are those pairs for which there is a reasonably large overlap of the class conditional probability densities. Thus, the resulting error probability for any class pair is a function of the amount of "separation" or discrimination between the members of that pair in the feature space. This distance may be estimated by a "figure-of-merit" which is the quantity that an automated feature selection algorithm can be based upon.

II.2 PROBLEMS IN FEATURE SELECTION

If one examines how the design of the feature extractor utilizes information from the classification subsystem, it becomes apparent that the process is ill-defined. Clearly, the major design consideration is the selection of the particular features that should be employed. However, no theory exists to guide us in determining which features maximize the ratio of the information that discriminates between the set of classes and the cost of taking these measurements. To further complicate matters, we have already pointed out that the utility of a feature cannot be assessed independently of the entire feature set to be employed. Even if the individual features are good, they are not necessarily non-redundant; twenty good features could be worse than five fair features if each of the twenty features is measuring approximately the same characteristic in the pattern space. Thus, the results of Mucciardi and Gose [6] are not surprising. They considered two characteristics for feature selection, 1) the probability of error produced by using a feature by itself and 2) the average correlation coefficient with the features selected already. The most effective weighting of these two characteristics was .1 for the first and .9 for the second. This seems to imply that the selection of non-redundant features is more important than the selection of inherently "good" features.*

Now we pose the question: How do we determine the optimal set of features? If the novice still expresses exuberance for this quest, let him consider that

---

*The first characteristic, however, is such a weak measure of the inherent worth of an individual feature that these results must be interpreted cautiously.

there are usually an infinite number of possible measurements and that even if we restricted the pool of possible measurements to only two hundred, there are practically an infinite number of different subsets of these measurements (2^{200} different subsets) each of which must be separately evaluated as an entire subset if we are to guarantee optimality. We hope that no one is upset by replacing our objective of optimality with the objective of finding a "good" subset where "good" is a coarse criteria that can be interpreted by the equally ill-defined rule of thumb: "it is very difficult to find another set that is more than marginally better for approximately the same range of costs."

II.3 DESIGN OF THE FEATURE EXTRACTOR

There have been two basic approaches to the problem of feature selection. The first involves an iterative process of heuristic selection by intuition and somewhat random guesses; this is followed by experimental testing to determine overall error rates and to pinpoint the pattern classes producing the difficulty, and the process repeats in an attempt to obtain a better set of features. The human designer somehow selects a set and then iteratively changes and tests the set. Clearly he is using the performance of the classifier to aid in his deletion of old features and selection of new features. Thus, in this heuristic manner, the classifier affects the design of the feature extractor.

The other mechanism that has been less widely used is the automated selection of the set of features from a pool of features that has been generated by the human designer and/or randomly. These algorithms require some

figure of merit for the expected utility of a feature and some means of determining redundancy between measurements [7,10-12]. The figure of merit employed in the two class problem has often been a measure of the "distance" or "separability" of the pattern classes by that feature.* Here again the classifier affects the selection of the features because this measure is an estimate (direct or indirect) of the overlap of the class-conditional probability densities and thereby the error rate that will be produced by the classifier [4]. The automated selection algorithms might also use the selection-test cycle to improve the quality of the feature set.

It is at this point that we should realize what is being done in each of these cases and attempt to unify and integrate the subsystems further. Both of the approaches actually reduce to just an informed heuristic selection procedure. The first is obviously so and the second, while founded on a strong theoretical basis, only heuristically relates the figure of merit and error rate unless strongly restrictive assumptions are made (such as Gaussian distributions and feature independence).**

III. INTEGRATION OF THE SIMPLE CLASSIFICATION SYSTEM

III.1 UNCERTAINTY AND ERRORS

One way that this initial system can be integrated and thereby made more

---

*The usual N-class case is handled by considering it to be either the set or the sum of pairwise dichotomization problems; in the first case one can then maximize the worst of these pairwise cases while in the second maximize the sum of the separate pairwise cases.

**In fact, one does not know how heuristically because the magnitude of the error of the assumption is unknown--what are the real underlying distributions? How dependent are the features?

powerful, is to allow options for the classifier during the decision process. Rather than be forced to always output its best guess as to the identity of the pattern class, the subsystem has further information that should be used, namely the confidence of its guess. Consider the classification procedure that was proposed: select j such that j maximizes the product $P(Cj)P(\overline{X}|Cj)$. There is certainly a difference in the confidence of the decision between the cases where this product is non-zero for only a single class as opposed to the uncertainty where there are several classes which are closely vying for the maximum likelihood. In the second case, one can allow the system to reject the pattern, i.e., refuse to make a decision. One simple rule that can be employed is that the ratio of the most likely class to the second most likely class must be greater than some constant or else it is rejected. Although this requires a certain number of patterns to be processed by some other means, the possibility of rejections allows the error rate to be reduced, sometimes a great deal. The discouraging note is that the attempt to reduce the error rate by an increasing amount requires an even larger percentage increase in the patterns rejected, some of which would have been correctly classified. Thus, eventually, one reaches the point where a modest decrease in error rate is obtained only at the expense of a very large increase in reject rate.

III.2 FEEDBACK FROM THE CLASSIFIER TO THE FEATURE MEASUREMENT SUBSYSTEM

Now that the option of uncertainty has been introduced to the classifier, one can add another path to the data flow to integrate the system further. Rather than immediately reject the pattern, the classifier can signal

the feature extractor that the information provided is not sufficient and more features should be measured. The additional features measured should have the capability to greatly reduce or remove the uncertainty in the decision. The design of the feature extractor now involves forming a secondary set of features to be employed. Since there is a high probability that the correct class is one of the top several most likely classes in the statistical decision process, we can view the feedback to the measurement subsystem as an attempt to eliminate confusion between several class pairs. Each of the features in this secondary set should be "specialists" at dichotomizing one or more class pairs. This is in contrast to features in the primary set which must be more "general purpose". The latter necessity can be seen by noting that in the case of just the 26 letters of English (ignoring differences of fonts, capital and small letters, etc.), there are $\binom{26}{2} = 325$ pairs of classes. If the number of features employed is not large compared to 325, and we make the reasonable assumption that the separation of every pair of classes will require information from several features, then each feature will have to be useful in separating more than one pair of classes, in fact probably many pairs. Thus features in the primary set are "general purpose" or "global" over the entire set of class pairs. The requirements of the feature specialists in the secondary set changes radically. If the number of classes that have a high likelihood are reduced to 4, there would be only 6 class pairs to dichotomize. Thus, if more features are to be measured, each feature chosen can be a specialist on one or possibly a few of the class pairs. Consequently, the secondary set of features will be very large with the expectation that only a few of these specialists will be required in those cases of uncertainty we have described.

The design of the secondary set can be accomplished by modifying the feature selection process accordingly. The same criteria of separation can be used but now we must be sure that there are one or more features that highly separate each class. There are many heuristic selection procedures that can be imagined. One simple alternative is to choose the best two features for each class pair to be available if further separation is required between any two classes.

III.3 THE SIMPLE INTEGRATED CLASSIFICATION SYSTEM

Let us summarize the system [Figure 2] that we have so far. The solid lines represent paths of data flow and should be quite clear at this point.

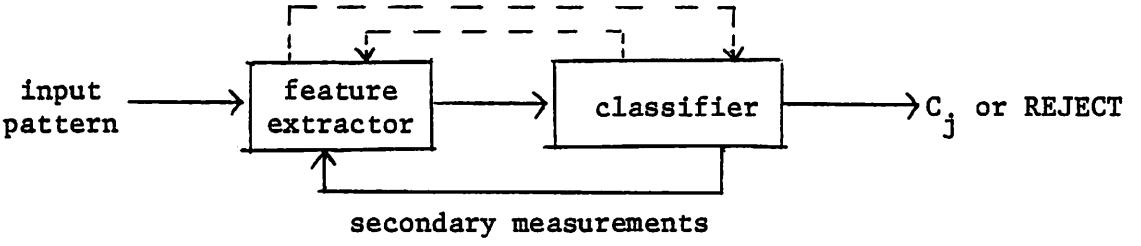

Figure 2. Paths of Data Flow and Design Consideration in an Integrated Classification System.

The dotted lines represent design considerations with the subsystem at the origin of the arrow influencing the design of the subsystem being pointed at. The system has been integrated by trying to select features carefully and in those cases where the confidence of classification is low, allow further specific measurements to be taken.

Many of the character recognition systems that have been reported in the literature produce character error rates in the range of 5 to 25%. Let us

put these systems in perspective for text recognition; assuming 6-letter words as an average, the word error rate would vary from 26 to 82%. Even a 5% character error rate would make the classifier impractical to operate. This does not mean these systems are inoperable; even humans have a 4% error rate if they classify without the use of context [13]. However, humans do use a great deal of context with a great reduction in the resultant error rate.

IV. A CONTEXTUAL POSTPROCESSOR

IV.1 GENERAL COMMENTS ON POSTPROCESSING BY CONTEXT

The usefulness of one further subsystem, a contextual postprocessor, has been clearly established in prior research. This subsystem can follow the classifier as an error detection and correction process. A review of the literature including these types of subsystems is available in Harmon [14]. Recently Riseman and Hanson [15] showed in simulations that simple contextual information [16] could be used to detect 99% of the errors in 6-letter words; depending on the size of the expected word input set, varying from 2755 6-letter words (which encompasses almost all such words that an individual uses) to 300 6-letter words, computationally efficient procedures were used to correct 65% to 95% of these errors, respectively. These particular procedures, as in many of the other error and detection systems, were designed independently of the rest of the character recognition system that it follows. The advantage of such a characteristic is that one type of contextual postprocessor might be used with many different classification systems with equal ease. A major point of this paper, however, is that we have much

to gain by giving up such standardization.

Utilization of a contextual postprocessor allows the first two subsystems of our classifier to be simpler and simultaneously makes the whole system more powerful. The assumption of the independence of the measurements can be made because many of the errors induced will be detected. By keeping the contextual information separate from the classification process, (i.e., not combining them into a single decision procedure), the overall system gains flexibility by having different options in the processing of characters as the following discussion will show. In the ensuing discussion we will incorporate into our system the postprocessor previously referred to [15] although other systems could easily be employed (possibly with some modifications in the design about to be presented).

IV.2 EXTENDED CLASSIFICATION SYSTEMS EMPLOYING POSTPROCESSING

The contextual postprocessor (CPP) operates by accepting as input the classifier's decision of each letter's identity until the end of the word; it will be assumed that the scanner will accurately determine the end of each word. At this time the CPP will attempt to detect if an error was made in one or more letters of the word, and then correction of any detected error will be attempted. If an error goes undetected we have no recourse; however, if an error is detected there are several options for further processing. Figure 3 depicts the pattern classifier with separate feedback loops from the CPP to the classifier and to the feature extractor.

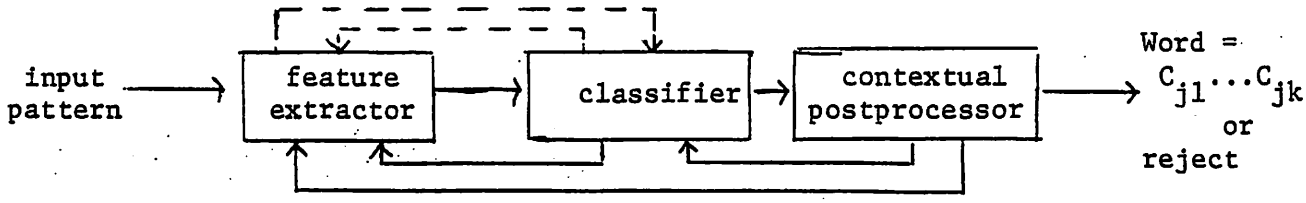

Figure 3. Addition of a Contextual Postprocessor to the Integrated System of Figure 2.

When an error is detected the CPP may:

1) correct the word directly, if it is confident, and output it;

2) determine the position(s) of the letter(s) in error and

   a) if the number of choices for the correct letter(s) is "small", then refer back to the classifier for reclassification among the few remaining choices; or

   b) if the number of choices for the correct letter(s) is "large" casting doubt upon the efficacy of reclassification directly, the CPP feeds back to the feature extractor for additional measurements before reclassification is attempted;

3) not be able to fix the position(s) of the error(s) and either

   a) rejects the word as uncorrectable; or

   b) chooses the subset of most likely positions of the error(s) and requests additional measurements for the letter(s) in these position(s).

A few comments on these options for processing are called for. We view the "small" number of choices that the CPP may narrow down to in the correction process to be roughly 2 to 5. The rationale is similar to one used previously, namely that the complexity decreases exponentially with the size of the subsets. Instead of attempting to separate the correct letter from the entire 26-letter set which involves all of the possible 325 pairwise dichotomies, the problem would be reduced to at most 10 pairwise dichotomies. Thus, initial erroneous classification has a far greater chance of being cor-

rected by carrying out the identical classification procedure in the restricted pattern subspace. One should also keep in mind that the pattern classes that are confused on the basis of the measurements employed usually are not the same pattern classes that the CPP confuses. For example, although "O's" and "C's" appear visually to be similar and might be mistaken for each other by the classifier, in many cases the contextual statistics would rule out one or the other as not being likely (or even possible). Thus, the expectation is that this procedure might boost the correction rate significantly.

In the case where further measurements are taken because the CPP cannot narrow down to a few alternatives, reclassification should proceed on the basis of all the measurements. One should not ignore any of the available information. Just because the initial measurements led to an error, it does not imply that the information is worthless. Rather, it means that there was not enough information. If the primary large set of measurements is ignored, this pattern would probably have a greater chance of producing another erroneous classification.

Finally, in option (3) where the position of the error(s) cannot be fixed, we can carry out requests for further information on the several possibilities of errors. It is hoped that collection of further information will lead to the correction of erroneous letters and the reinforcement of correct choices.

This completes the description of the paths for data flow of the entire classification system. Let us continue the development of our integrated classifier by examining the ways in which the first two subsystems can influence the design of the CPP and vice versa.

IV.3 INFLUENCE OF THE POSTPROCESSOR ON THE CLASSIFIER

It has been pointed out that the CPP we are referring to is an independently operating subsystem. For any given subset of the language and assuming a uniform error distribution from a hypothetical classifier, data can be collected on the probability that this CPP can detect and correct the error of the classifier outputting $\wedge$ C_j given that the correct class is C_i, i.e., determine $P'(C_j|C_i)$ for all $i \neq j$ and $1 \leq i,j \leq 26$. This information is a clear summary of the strengths and weaknesses of the CPP independent of the first two stages of the classifier that precedes it. A high value of $P'(C_j|C_i)$ implies that one need not be concerned about this type of error because this subsystem will detect most of the occurrences of erroneous substitutions of C_j for C_i in the given set of input words. Conversely, a relatively low value says that one should avoid this type of classifier error. No longer are the errors equal; some initial classifier errors will never become system errors because of the CPP. By designing the first two stages as a function of the CPP, the weaknesses of the post processor can be compensated for by the strength of the feature extractor and classifier.

A reasonable alteration in the system design involves the classification procedure. The classifier used a Bayes decision procedure which is optimal if there is equal loss for all types of errors. Since that is no longer the case, the optimal decision procedure must include the loss incurred for the various types of errors, $L(C_j|C_i)$. Now the goal is to choose that class which minimizes the risk (or expected loss),

$$\min_j \sum_{i=1}^{26} P(\overline{X}|C_i)\, P(C_i) L(C_j|C_i).$$

By properly choosing the loss functional we can relate the risk to the probability of giving an incorrect system output (as opposed to a classifier output). Let $L(C_j|C_i) = 1 - P'(C_j|C_i)$, then the minimal risk is the minimization of the probability of making an error that will go uncorrected by the contextual postprocessor. This straightforward modification immediately extends the optimality of the decision criteria (assuming no known underlying probability distribution) from the clsssifier subsystem alone to the whole system.

IV.4 INFLUENCE OF THE POSTPROCESSOR ON FEATURE SELECTION

The presence of the contextual postprocessor even allows improvement in the process of feature selection. In order to demonstrate this let us outline a feature selection algorithm that will allow design dependencies from the CPP to be naturally incorporated [17]. By employing a figure of merit (refer to section II.3) as a measure of the separation of class pairs by each feature, we can associate with each class pair, C_i and C_j, a threshold, θ_{ij}. This threshold will indicate the minimum desired amount of discrimination between classes i and j. Features may then be selected sequentially on the basis of their total contribution to the separation of all class pairs. At any point in the selection process when some subset of features has already been chosen, separation of class pairs which have already surpassed their desired threshold can be ignored. The objective, then, is to select a set of features which achieves the "best" separation between all class pairs as specified by the threshold vector. "Best" may mean, if θ is the threshold vector and F_i is the figure of merit vector for the i^{th} feature set:

$$\min_i \quad ||F_i - \theta||$$

where $||\bar{v}||$ is an appropriate vector norm. Note that when it is not possible to exceed all thresholds one cannot be certain of achieving the best set except by examining all possible subsets of features, computing the figure of merit vector, and comparing this with θ. However, various methods for achieving a "good" feature set by a process of sequential selection may be employed.

Now we can consider a straightforward modification of this procedure by utilizing the information from the contextual postprocessor. Clearly the desired separation of a class pair is a function of the ability of the CPP to rectify any confusion of the two classes. Thus, the threshold of separation for each of the 325 class pairs in English can be individually varied from some initial value as a function of $P'(C_j|C_i)$ for all i,j. In particular, θ_{ij} can be varied inversely with $P'(C_j|C_i)$, lowering the threshold for some and raising it for others; e.g. one could use the function $\theta ij = K_1[1 + K_2 - P'(C_j|C_i)]$ where K_2 is that value of $P'(C_j|C_i)$ below which we want the threshold raised above K_1. Although the desired separation may not be achieved for all of the class pairs, it should result in large separations for many more of the class pairs where the CPP is weak [17]. The process of altering the thresholds can be viewed as a focusing process in the feature selection algorithm.

IV.5 EFFECT UPON ERROR RATE

We have designed the feature set to yield more information for those class pairs whose confusion will cause the CPP difficulty. The classifier has

also been modified to bias its decision against those decisions in which an error will cause the CPP difficulty. Actually the error rate at the output of the classifier might increase; quite possibly we are designing the system to make more errors, but errors of a specific type, namely errors correctable by the CPP. This design should lead to an error rate for the entire system that is lower.

The system design can now be depicted with the additional design dependencies shown in Figure 4.

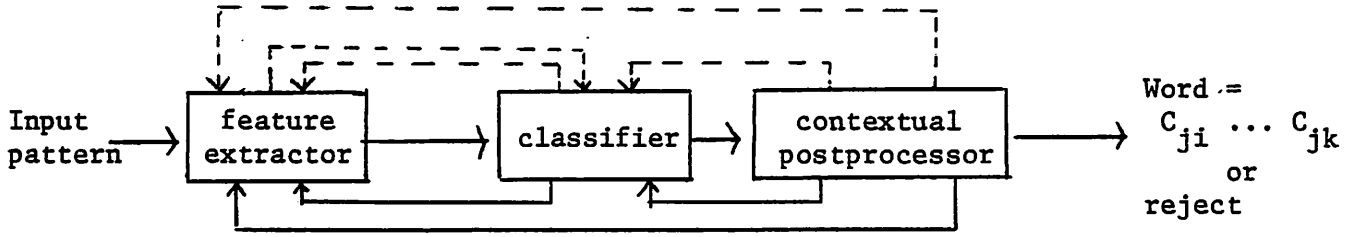

Figure 4. The Integrated Classification System

V. DESIGN OF THE POSTPROCESSOR

V.1 INFLUENCE OF THE CLASSIFIER ON THE POST PROCESSOR

The final subsystem design considerations to be discussed involve the contextual postprocessor. When the CPP determines that some character is in error, it attempts to correct this error. The erroneous output of the classifier can be useful in the determination of the proper letter since any classification system only confuses some subset of the letter pairs [18]. Thus, the first change involves collecting and utilizing statistical information on the manner in which the first two subsystems operate. This can be achieved in

the following way. Assume the first two stages are designed on the basis of the statistics of the unmodified CPP (the CPP defined up to this point). Now, a test set of patterns can be employed to determine the error matrix at the output of the classifier. Then before the CPP expends any effort toward correction, the classifier error matrix can be employed to narrow down the set of possible input letters that could have produced this error. In many cases this reduction might be dramatic, yielding only a few alternatives for correction.* In such instances one might expect an order of magnitude improvement in the correctability rates of the CPP. The contextual postprocessor that was discussed utilizes only probabilities of letter n-grams that have been quantized to 0 or 1 on the basis of whether the probability was zero or non-zero, respectively (these have been referred to as binary n-grams [15,16]). For this postprocessor the only information necessary is a 26 x 26 matrix of bits, each specifying whether a particular letter error substitution by the classifier is possible. In those cases where there is still ambiguity between several candidates for the correct letter, the probabilities of the errors can be employed. If one of the candidates is much more likely to have produced the error than any of the others, the correction can be made. The procedure that has been described should only make use of the statistics collected from the first attempt at classification by the classifier since that is the point at which the CPP will detect and then attempt to correct the error.

---

*This belief is based upon the expectation that each input letter will only be confused with a relatively small subset of letters.

V.2 INFLUENCE OF THE FEATURE SELECTION PROCESS

It is not at all clear how the CPP should be designed on the basis of the features that have been selected for the measurement subsystem. The features that have been selected indirectly affect the design of the CPP already since they play an important part in determining the error matrix. But this does not lead to any new insights at present. We have already utilized this information in the preceding section to improve the performance of the CPP. Therefore we will not examine this relationship any further.

VI. CONCLUSION: THE TOTALLY INTEGRATED SYSTEM

The design of our integrated character recognition system is now complete. Let us summarize all the design dependencies between the three subsystems as shown in Figure 5. We have discussed the direct design dependencies of each subsystem upon the other two except the dependency of the CPP upon the feature extractor.

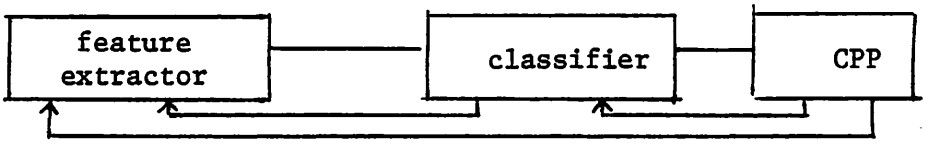

Figure 5. Design Dependencies in the Integrated Classification System.

The design dependencies can be summarized as follows:

I. the feature extractor is influenced by

   a) the classifier--the selection of features may involve an iterative process of the select-test cycle to experimentally pinpoint class pairs that cause difficulty;

b) the classifier--the figure of merit in the automated feature selection algorithm is used to provide an estimate (directly or indirectly) of the expected confusion between class i and class j;

c) the contextual postprocessor--the desired threshold of separation of the class pairs C_i and C_j are altered inversely proportionally to the probability that the CPP can correct an error when C_i is substituted for the correct letter C_j;

d) both the classifier and the contextual postprocessor--a secondary set of measurements (feature specialists) is selected for any class pairs that may be confused in the classification and/or correction processes.

II. the <u>classifier</u> is influenced by

a) the feature extractor in the normal manner--the feature measurements form the base upon which the classifier operates;

b) The contextual postprocessor--the decision procedure incorporates a loss function which is a measure of the inability of the postprocessor to correct classifier errors.

III. the <u>contextual postprocessor</u> is influenced by

a) the classifier--the CPP employs the error matrix of the classifier to determine the letters which most probably could have produced the observed error.

REFERENCES

[1] R. Bakis, N.M. Herbst, & G. Nagy, "An Experimental Study of Machine Recognition of Hand-Printed Numerals", IEEE Transactions on Systems Science and Cybernetics, Volume SSC-4, pp. 119-132, July, 1968.

[2] D.C. Allais, "The Selection of Measurements for Prediction", Systems Theory Laboratory, Stanford University, Stanford, California, Tech. Rept. 6103-9, November, 1964.

[3] C.K. Chow, "A Class of Nonlinear Recognition Procedures", IEEE International Conv. Rec., vol. 14, pt.6, pp. 40-50, March, 1966.

[4] W.S. Maisel, Computer Oriented Approaches to Pattern Recognition, Academic Press, N.Y., 1972.

[5] E.A. Patrick, Fundamentals of Pattern Recognition, Prentice-Hall, Englewood Cliffs, N.J., 1972.

[6] A.N. Mucciardi and E.E. Gose, "A Comparison of Surn Techniques for Choosing Subsets of Pattern Recognition Properties", IEEE Trans. Computers, Vol. C-20, Ho.9, September, 1971, pp. 1023-1031.

[7] L.A. Kamentsky and C.N. Liu, "Computer Automated Design of Multifont Print Recognition Logic", IBM Journal of Research and Development, pp. 2-13, January, 1963.

[8] C.H. Chen, "Theoretical Comparison of a Class of Feature Selection Criteria in Pattern Recognition", IEEE Trans. Computers, Vol.C-20, Ho.9, September 1971, pp. 1023-1031.

[9] H. Kobayashi and J.B. Thomas, "Distance Measures and Related Criteria", in Proc. 5th Annual Allerton Conf. Circuit and System Theory, Oct. 1967, pp. 481-500.

[10] R.P. Heydorn, "Redundancy in Feature Extraction", IEEE Trans. Comput., Vol. C-20, pp. 1051-1054, Sept. 1971.

[11] R.C. Ahlgren, H.F. Ryan, and C.W. Swonger, "A Character Recognition Application of an Iterative Procedure for Feature Selection", IEEE Trans. Comput., Vol. C-20, pp. 1067-1075, Sept. 1971.

[12] S.K. Das, "Feature Selection with a Linear Dependence Measure", IEEE Trans. Comput., (Corresp.) Vol. C-20, pp. 1106-1109, Sept. 1971.

[13] U. Neisser and P. Weene, "A Note on Human Recognition and Hand Printed Characters", Inf. & Control, Vol. 2, 1960, pp. 191-196.

[14] L.D. Harmon, "Automatic Recognition of Print and Script",

[15] E.M. Riseman and A.R. Hanson, "A Contextual Post Processing System for Error Detection and Correction in Character Recognition", Computer and Information Science Tech. Rept., 72C-1, Univ. of Mass., Amherst, Oct. 1972.

[16] E.M. Riseman and R.W. Ehrich, "Contextual Word Recognition Using Binary Diagrams", IEEE Transactions on Computers, Vol. C-20, pp. 397-403, April, 1971.

[17] E.M. Riseman and A.R. Hanson, paper in preparation.

[18] C.M. Vossler & N.M. Branston, "The Use of Context for Correcting Garbled English Text", in Proc. ACM 19th Nat. Conf., pp. D2.4-13, Aug., 1964.