Recognizing Structured Forms Using Neural Networks

Jon A. Pastor
Suzanne Liebowitz Taylor

Unisys Center for Advanced Information Technology
70 E. Swedesford Road
Paoli, PA 19301
USA

Abstract

The ability to understand complex printed forms automatically is of considerable value for applications which must process large volumes of submitted paperwork. When processing batches of heterogeneous forms, it is essential to first recognize a given form as an instance of a particular type, so that its contents can be interpreted correctly. Performing this recognition presents some technical problems that may not be apparent initially. We review these problems, and describe a Neural Network implementation that uses a novel feature set to perform reliable identification of U. S. Internal Revenue Service forms.

1 Introduction

With rapid decreases in the cost of scanners and high-capacity storage media such as optical disk drives, the capture of large volumes of digitized textual information is becoming a reality. Advances in printed character recognition have made it possible, in some cases, to "read" documents and store them as text files rather than as digitized images. Given the sophistication of the available hardware and software, it is reasonable to expect that structured printed forms, of the kind typically used by bureaucratic organizations, would be simple to read and interpret. We have found that this is not always the case, particularly when the forms were designed for use in a manual-entry system.

In Section 2, we present a case study on a particular group of forms, the income-reporting forms used by the Internal Revenue Service in the United States, to introduce the form recognition problem. In Section 3, we enumerate approaches to this problem, and justify the use of Neural Networks. As in most Neural Network problems, preprocessing is a critical issue in our approach, and in Section 4 we discuss various preprocessing strategies. In Section 5, we describe the implementation of the network, and in Sections 6 and 7, we present and discuss our results.

2 IRS Forms

The Internal Revenue Service (IRS) is the central tax collection agency in the United States; all income and expenses must be reported to the IRS by both individuals and corporations. The IRS publishes a large number of different forms and schedules to accommodate a wide range of income sources and expenses; a typical tax return will contain from one to several dozen of these. Although tax returns may be filed electronically, the majority are still filed manually on forms obtained directly from the IRS or photocopied from IRS forms. In addition, the availability of inexpensive tax-preparation software for personal computers has led to a proliferation of computer-generated forms, the quality of which depends on the type and condition of the printer used.

The IRS receives hundreds of millions of forms and schedules annually, which are presently read by human operators and keyed into the IRS computer system. Since this process is error-prone and expensive, automatic entry of these forms would be of considerable benefit. Each of the more than 160 IRS forms and schedules has a unique set of data fields. The semantics of a form are dependent on its identity, and identification is thus pre-requisite to locating data fields, which is in turn pre-requisite to extracting, segmenting, and interpreting them. Since all subsequent processing depends on the recognition stage, and the processing of forms must be done in real-time, form recognition must be accurate and rapid.

While the above discussion focuses on IRS forms, it applies to any set of forms with similar characteristics. Such a set would include a variety of forms that arrive in a heterogeneous pool, are not designed for machine-readability, and are so numerous that manual entry is unacceptably time-consuming and inaccurate.

3 Form Recognition Strategies

A superficial analysis of the IRS form recognition problem might lead to the conclusion that it is appropriately addressed by a template-matching strategy: there is a fixed set of distinct forms, and one can identify an unknown form by comparing it with each of them. Implementing such a strategy, however, presents a number of practical problems.

First, one can make few assumptions about the layout of any given instance of an IRS form, since forms may be obtained from the IRS, photocopied, or printed by a tax preparation package. Even forms obtained from the IRS are not necessarily identical: there are several "official" printers used by the IRS, and each of these appears to use different artwork. Therefore, different images of the same form can vary not only in size, scale, and position on the page, but also in content and structure. Any recognition strategy must therefore be based on features of the form that are relatively insensitive to these variations.

Second, some IRS forms are quite similar in structure, and thus require fine comparisons, necessitating high resolution in the scanned images of the forms. High resolution gray-scale images are also required by subsequent processing of the form (e.g., character extraction and recognition), once it has been identified. We have found that, in order to preserve sufficient detail in handwritten characters and small distinguishing features, it is necessary to scan forms at 150 DPI, with 64 gray-levels, resulting in images that are on the order of 1200 by 1700 (eight-bit) pixels. Since there are over 160 different IRS forms and schedules, it is clearly impractical to attempt to match on full-size images, if the recognizer is to be used in a real-time system.

4 Preprocessing

The preceding discussion suggests that a recognition strategy must include some means of reducing the size of the form images; alternative approaches to size reduction include:

1. semantic (extraction of a region of the image that provides adequate discrimination among forms),
2. pixel-level (e.g., image compression),

3. geometric (analysis at a higher level of abstraction than raw pixels, but below the level of semantic features).

As in most Neural Network applications, preprocessing is a key to successful implementation. We evaluated all three reduction strategies, and experimented with several variants of two of them.

4.1 Semantic Feature Analysis

Region extraction is impractical as a primary technique, since there is no single distinguishing feature that is present on all IRS forms. Many – but not all – forms have identifying information in the upper-left or lower-right corner, but even among forms that do include identifying information, it is typically variable-sized text, and does not appear in the same position from form to form, thus making it difficult to locate, segment, and recognize. While local high-level semantic features are thus impractical as a primary means of form recognition, they can be useful given a tentative identification of the form, since features may be sought in known locations. We plan to use local feature analysis to confirm identifications where confidence in the original identification is low, or where the network is known to have difficulty distinguishing among particular forms.

4.2 Pixel-level Feature Analysis

Our first approach to compression used block reduction to reduce the size and resolution of the image. Block reduction replaces each N-by-N block of pixels in the original image with a single pixel in the reduced image; the value placed in the reduced image is some function (typically an average) of the source pixels. Block reduction will blur and eliminate fine features while retaining the gross structure of the image. A reduction factor of 30 provided a reasonable compromise between dimensionality-reduction and preservation of distinguishability among forms, but still resulted in a feature vector nearly 2400 elements long. A second representation that depended on a pixel-level analysis, but with a somewhat smaller feature vector, performed as well as the first representation, but took significantly longer to train.

4.3 Geometric Feature Analysis

We examined several types of geometric features before choosing one based on the distribution of line-intersections. We have developed an algorithm for locating data fields on a form, once it has been identified[8]; during the computation of this algorithm, the location of each occurrence of each of ten different types of line crossings is recorded. Hypothesizing that the distribution of these crossing types might be distinctive, we began experiments with this representation. Partitioning the form into nine bins (three horizontal by three vertical), and counting the number of occurrences of each type of crossing in each bin, resulted in a 90-element feature vector; the choice of three vertical and three horizontal bins was based on the results of hierarchical cluster analysis. Partitioning the form captures the spatial distribution of the features, without placing undue emphasis on precise location, thus providing some shift-, scale-, and rotation-invariance.

When the network was trained on forms encoded in this manner, we found that a mean square error (MSE) comparable to that obtained with the block-reduced images was attained after 30% fewer presentations of the training set, and the time to train was reduced by nearly an order of magnitude, with recognition accuracy comparable to that obtained with block reduction and superior to that obtained with the alternative pixel-based representation. An added benefit – quite significant in a system that is intended for use in a real-time system – is that computing this feature vector has virtually no incremental cost, since the features that it uses are derived via simple arithmetical operations from features that are already being computed for use by another component of our system.

5 Implementation

Images are captured offline on a Fujitsu 64-gray-level flatbed scanner, and stored on optical disk. Preprocessing is performed using a set of C routines, some of which rely on an Androx signal-processor for image operations. Initial experiments were performed using the Rochester Connectionist Simulator [4], via its X Window System interface, on a SUN 3/60 workstation. The success of these experiments prompted us to acquire a dedicated accelerator board (the HNC AnzaPlus^{TM} DP) for one of our SUN 3/160 workstations, and re-implement on this platform. In production, the scanner output would be fed directly to the preprocessor, whose output would then be fed to a co-processor hosting the Neural Network.

Since we could see no specific need for special architectures, we elected to employ a simple feed-forward network, trained using back-propagation[6, 7, 9]; we have experimented with one- and two-hidden layer models. We are using floating-point state values, and a one-of-N output coding. We are not thresholding outputs, since we anticipate that some forms will be functionally indistinguishable by the NN, and plan to use the scores to permit checking of the second- and lower-ranked forms via other techniques. Preliminary results have been quite satisfactory, and we will explore other models only if the current strategy proves not to scale well as additional forms are added to our database.

6 Results

Unlike speech- and character-recognition efforts, for which large bodies of live data are common and easily accessible, there is no public database of IRS forms. As a consequence of this, we have been forced to generate our own training and test data. Since each scanned image occupies an average of about 1.4 megabytes of storage, generation and maintenance of a database have presented a significant challenge. Our initial database contained an average of 10 samples of 13 different forms; we have subsequently expanded this to an average of 11 samples of 21 different forms. As might be expected, we have observed a substantial difference in the strength of classifications for forms that have a relatively large number of examples in the training set, and performance has improved as the number of training samples increased. Two samples of each form are used for testing.

The training and test data images were scanned from a combination of sources, including both original IRS forms and photocopies, and both blank and hand-filled-in forms. For hand-crafted forms, we have used only semantically correct data – in other words, each form was filled out correctly, with all numbers in the correct arithmetic relationships, so that subsequent stages of processing (e.g., digit recognition) would be able to use form semantics to validate contents. Scanning was deliberately haphazard, with shift and rotation introduced randomly. Photocopies made at scales ranging from 0.9 to 1.1 were also included in our data set.

We have tested networks with one and two hidden layers, and with various numbers of nodes in each layer. In all cases, training was stopped when the change in MSE from one epoch to the next was less than 0.0001; further training was found to provide virtually no improvement

Figure 1: IRS Forms 1120 (left) and 1120-A (right); the quality of these images is typical of photocopied forms, which must be recognized with the same accuracy as official IRS printed versions of the same form.

in performance. With the 21-form database, we have had the most success using nets with two hidden layers containing 40 nodes each. With this architecture, we are able to achieve 98% (41 out of 42) accuracy in the best training runs. This figure is, however, somewhat optimistic, since some of the identifications are made with relatively low confidence, as measured by strength of response of best-responding output node, or discrimination, as measured by signal-to-noise (S/N) ratio": some "correct" identifications are made with the activation level of the best output node at or below 0.5, or with the ratio of the highest activation level to the second-highest little better than 1:1. In practice, weak identifications such as these would be considered tentative, and high-level semantic features would be sought to confirm or disconfirm the identification.

The forms that are typically mis-identified, or identified weakly, are generally either very similar in structure to those for which they are mistaken, or have a structure that is difficult for the line-crossing algorithm to interpret. An example of the former is illustrated in Figure 1: forms 1120 and 1120-A are visually quite similar, and their line-crossing profiles will obviously be almost identical. An example of the latter is form 1040EZ, which uses shaded boxes rather than lines to define input areas, and so poses a difficult challenge to the line-crossing algorithm.

7 Discussion

The work presented here is still in a preliminary state; we continue to build our training and test databases and to experiment with other architectures and parameterizations. Initial experiments with scaling have been encouraging, and we remain optimistic; however, scaling up to the full set of 160 IRS forms and schedules may necessitate changing the architecture of our system. One technique that we will consider, as the size of our database increases, is an hierarchical approach such as that described in [5]. If overall performance is acceptable, but specific clusters of forms prove difficult to distinguish, an architecture that permits class-by-class tuning of sensitivity, such as ART [1, 2, 3], may be useful. Since we anticipate that some forms will be indistinguishable by the NN, we already plan to augment the NN recognizer with local high-level feature analysis; in cases where the NN's identification is weak, the same tools can be applied to provide supporting evidence.

Acknowledgements

We would like to thank Richard Fritzson and Rick Rudolph for valuable suggestions during the development of our system. The algorithm that generates line crossing profiles of forms was developed by Richard Fritzson.

References

[1] Gail A. Carpenter and Stephen Grossberg. Art2: Self-organization of stable category recognition codes for analog input patterns. *Applied Optics*, 26:4919–4930, 1987.

[2] Gail A. Carpenter and Stephen Grossberg. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing*, 37:54–115, 1987.

[3] Gail A. Carpenter and Stephen Grossberg. Art3: Hierarchical search using chemical transmitters in self-organizing pattern recognition architectures. *Neural Networks*, 3(2):129–152, 1990.

[4] Nigel H. Goddard, Kenton J. Lynne, Toby Mintz, and Liudvikas Bukys. Rochester connectionist simulator. Research Report TR233 (revised), University of Rochester, Rochester, NY, October 1989.

[5] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1), 1991. To appear.

[6] David Parker. Learning logic. Invention Report S81-64, Stanford University, File 1, Office of Technology Licensing, 1982.

[7] Donald E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. In Donald E. Rumelhart and James L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, chapter 8, pages 318–362. MIT Press, Cambridge, Mass., 1986.

[8] Suzanne Liebowitz Taylor, Richard Fritzson, and Jon A. Pastor. Structured document processing, 1991. In preparation.

[9] Paul J. Werbos. *Beyond Regression: New Tools for Predictions and Analysis in the Behavioral Sciences.* PhD thesis, Harvard University, 1974.