Recognition of Gridletters: Probing the Behavior of Three Competing Models

Gary McGraw & Daniel Drasin
Center for Research on Concepts and Cognition
Department of Computer Science
Indiana University, Bloomington, Indiana 47405
gem@cogsci.indiana.edu ddrasin@silver.ucs.indiana.edu

1 Letter Spirit and Gridfont Recognition

This paper compares the performance of three different models of letter recognition in the Letter Spirit domain. The approaches reported here are rivals for the model of letter recognition that will actually be used in the Letter Spirit program[1]. Since preliminary work on Letter Spirit will deal almost exclusively with the recognition phase of the Letter Spirit project, it is important to build and test alternative architectures in order to assess the strengths and weaknesses of the one we have chosen to implement. The hope is to create some basis for inter-architecture comparison and analysis.

The Letter Spirit project is an attempt to model central aspects of human high-level perception and creativity on a computer, focusing on the creative act of artistic letter-design. The aim is to model the process of rendering the 26 lowercase letters of the roman alphabet in many different, internally coherent styles. Two important and orthogonal aspects of letterforms are basic to the project: the *categorical sameness* possessed by instances of a single letter in various styles (*e.g.,* the letter 'a' in Baskerville, Palatino, and Helvetica) and the *stylistic sameness* possessed by instances of various letters in a single style (*e.g.,* the letters 'a', 'b', and 'c' in Baskerville). Figure 1 shows the relationship of these two ideas.

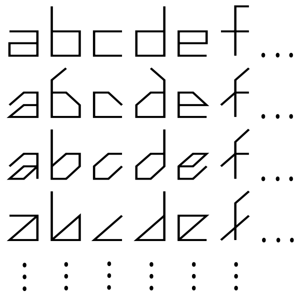

Figure 1: Items in any column have *letter* in common. Items in any row have *spirit* in common.

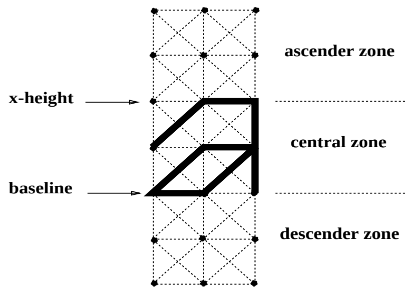

Figure 2: The Letter Spirit grid, with one of the possible sets of quanta instantiating an 'a' turned on.

Starting with one or more seed letters representing the beginnings of a style, the program will attempt to create the rest of the alphabet in such a way that all 26 letters share the same style, or *spirit.* To avoid the need for modeling low-level vision and to focus attention on the deeper aspects of letter design, we eliminated all continuous variables, leaving only a small number of discrete decisions affecting each letterform. Letterforms are restricted to short line segments on a fixed grid of 21 points arranged in a 3×7 array [Hofstadter, 1985]. Legal line segments, called *quanta*, connect a point to any of its nearest neighbors. There are 56 possible quanta, as shown in Figure 2. This restriction allows much of low-level vision to be bypassed and forces concentration on higher-level cognitive processing, particularly the abstract and context-dependent character of concepts.

Becauses quanta are either on or off, decisions on the grid are coarse. Surprisingly, the variety among letters of a given category is still huge — hundreds of versions of each letter and 600 full gridfonts have been designed by humans. Almost paradoxically, the domain's limitations engender this diversity.

As there are no surface features to manipulate at a fine-grained level, one ends up playing at the boundaries of the 26 categories. Consequently, many gridfonts are wild, sometimes having angular, blocky, spiky,

[1]For information about the Letter Spirit project see [Hofstadter and McGraw, 1993] and [McGraw and Hofstadter, 1993].

sparse, or otherwise bizarre letters. This makes the task of recognizing gridletters very hard.

2 Three Different Models

2.1 A symbolic approach — DumRec

DumRec is a messy-symbolic-AI gridletter-recognition program written in Scheme. DumRec compares a mystery letter with a number of stored letters, counting agreements and disagreements of all sorts (which are computed at various levels of complexity), and making a decision based on a weighted comparison of micro-features. DumRec's database of stored letters can be changed to include any number of different "training letters".

After reading in a data file containing several training letters and computing a property list for each letter, DumRec requests a mystery letter. When the mystery letter is presented, a property list is computed for it in exactly the same way. The mystery letter's property list is then compared, using a weighted metric, to the property lists of the training letters.

DumRec uses two types of comparison. The first is a direct approach in which the quanta of the mystery letter are directly compared to the quanta of each training letter. DumRec has two comparisons of this type: a rating based on Hamming distance, and an "on"-bit rate which is the number of quanta shared between the mystery letter and a known letter. Points are awarded to each training letter based on how well it matches the mystery letter[2].

All the other points awarded to training letters are based on comparison of the pre-computed feature lists of each training letter to the mystery letter's feature list. These features can be broken into three levels (based on the amount of abstraction involved in the computation).

Level-1 data involves a small amount of abstraction since level-1 properties are made up of multiple quanta. Level-1 data includes position information for a variety of simple angles and shapes. Numbers of certain types of shapes (like squares, triangles, lines, angles and closures) are also considered level-1 data.

Level-2 data consists of more abstract properties — namely, "mass", "gravity" and sectorization data. The grid has six sectors (the three zones divided into left and right halves), each of which may or may not include quanta from the letterform. A letter's "gravity" is the number of the sector with the most quanta.

Level-3 data is the most abstract of all in the DumRec program. Level-3 properties include "point" and "tip" information. The top of the bar on the left side of the letter 'b' is an example of what we mean by a tip. "Points" are the 21 intersection points of quanta on the grid.

Once the property data have been calculated for the mystery letter, the matching process begins. DumRec calculates a score for each training letter based on its relation to the mystery letter. The score is a weighted ranking of the match of the property lists. The weights play a crucial role in DumRec's performance. Modifying the weights is easy and is an important part of "tuning" DumRec. In the future we plan to adjust these weights using some sort of relaxation algorithm, but for now they are set by hand.

Interestingly, DumRec is fairly good at gridfont letter recognition, most of the time either guessing the correct letter or picking a "reasonable" wrong one. Results of DumRec's performance are discussed below.

2.2 Connectionist approaches

NetRec We have completed several experiments in gridfont letter recognition using two- and three-layer feedforward connectionist networks. The NetRec networks are trained (using backpropagation) on a variety of gridfonts and then tested on fonts they have not yet seen. Results show that connectionist networks can perform fairly well on recognition tasks. The ability to generalize about input patterns and create complex association maps between input patterns and their categories is especially useful in gridletter recognition.

Our networks use a localist representation for classification purposes. The input layer consists of 56 nodes, one for each quantum on the grid. If a quantum is "on" (*i.e.,* it has a line drawn in its position), the quantum's associated node gets high activation. If it is "off", its node gets little or no activation. The output layer is simply a vector of 26 nodes, one for each letter of the alphabet. Hidden layers consist of

[2] A simplified version of DumRec uses only the Hamming distance for recognition. Its performance is significantly degraded.

anywhere from 0 to 120 nodes. The network is trained to associate patterns on its input layer with patterns on its output layer, using standard backpropagation of error.

One of the major open problems in connectionism involves deciding how to pick both a learning rate for backpropagation and the number of hidden units to use[3]. We know of no theoretical results which help guide the choice of these two key parameter values. There are some fairly obvious rules of thumb which can be consulted for guidance, but they do not always apply. We opted to determine values for these two parameters experimentally. In the course of doing this we came across some interesting results.

	Learning rate													
HU	.001	.01	.05	.1	.15	.2	.25	.3	.35	.4	.45	.5	.8	.99
10	2.28	1.91	2.00	1.27	0.61	0.39	0.30	0.26	0.22	0.20	0.18			
20	2.17	1.94	1.57	0.47	0.27	0.21	0.17	0.15	0.14	0.12	0.11	0.09	0.00	0.06
30	13.4	1.96	1.28	0.36	0.23	0.18	0.15	0.13	0.12	0.11	0.10	0.08	0.06	0.05
40	max	10.8	1.5	0.34	0.21	0.16	0.14	0.12	0.11	0.10	0.09	0.08	0.06	0.05
50			16.1	1.03	0.29	0.19	0.15	0.13	0.15	0.10	0.09			
55	16.5	14.2	2.15	0.44	0.26	0.20	0.17	0.15	0.13	0.12	0.09			
58	max	max	max	23.1	18.6	14.5	0.53	0.21	0.15	0.13	0.11			
60		max	max	24.1	20.4	16.7	0.46	0.29	0.17	0.14	0.11			
65		16.0	14.7	9.32	2.54	0.34	0.20	0.16	0.14	0.13	0.12			
70		max	max	max	max	max	max	max	max	max	max			

Table 1: Average error on various networks trained and tested on *standard square*. HU stands for hidden units. In general, an error greater than 2 signifies very bad performance (non-convergence) and an error less than 0.2 signifies almost perfect convergence. Note that a very low value may also signify overlearning.

Table 1 shows a convergence chart for three-layer networks trained on a particular gridfont (known as *standard quare*) for 50,000 cycles. We actually tested more networks than those shown on the chart but none of them performed better than the ones shown. The numbers on the chart represent average error over all output nodes for a test run. The lower this number, the better the convergence. A graphical interpretation of part of the chart is shown in Figure 3. We were surprised to find that none of the networks with over 70 hidden units converged no matter what the learning rate was! We are not sure why this is and would like to explore this further in the future.

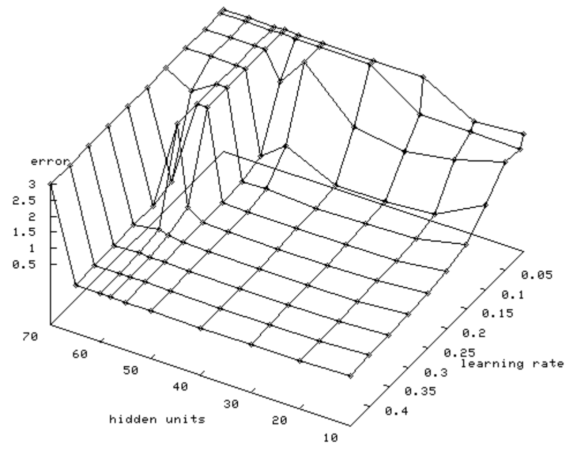

Figure 3: Convergence on standard square.

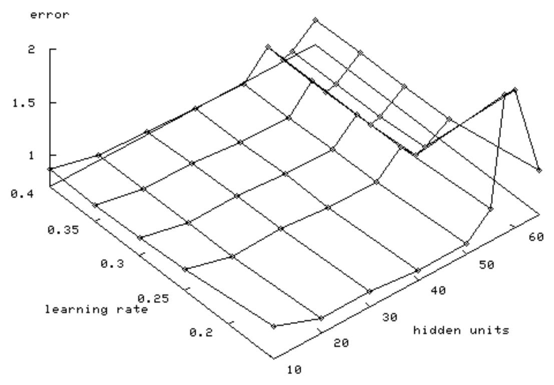

Figure 4: Convergence on the experimental data set.

A similar convergence pattern (having the same weird hidden-unit—nonconvergence property) is shown in Figure 4. This data is from a three-layer network run on our experimental test set (explained below). Based on the results of experimentation, we decided to use a learning rate of 0.4 and 50 hidden units for the comparison tests reported below.

FnetRec After finding that the performance of NetRec was generally not as good as DumRec's (see below), we decided to try a "feature-based" connectionist approach. The hypothesis was that an architecture which forces the network to pay more attention to certain features (as determined by us) will perform better. Similar modular approaches to letter-recognition and/or other categorization tasks include [Fukushima, 1989],

[3]Actually these are just two of the many parameters of a backpropagation network that must be tuned by the user.

[Guyon et al., 1989], and [Regier, 1991]. Our approach was to train a number of small "subnets" to detect certain features, and then combine these subnets together into a letter-recognizer.

All nine subnets require the usual 56 input units and one output unit. Output values are in the range [0..1] and correspond to a boolean or scaled value. There were subnets for each of the following features: height (using 7 hidden units), descenders (5), tips1 (7), tip2 (9), tips3 (11), tips4 (13), weight (5), and ascenders (7).[4] The tips1-4 features responded positively depending on whether the letter had the number of tips specified in the name. The subnets were treated as input to a classification net. The classification network had a hidden layer or 20 units with direct connection to the 56 input units as well as inputs from the subnets.

More detailed information about NetRec, FnetRec, and Dumrec is available in [McGraw, 1990].

3 Comparing Performance

In order to compare the performances of DumRec, NetRec, and FnetRec, we created a data set with 398 gridfont letters. The data set included letters from each of the 26 categories with 9 to 23 instances of each letter (depending on the letter). The first 5 instances of a given letter were used as "training letters". These 130 letters were used either as a training set for the networks or to calculate initial property lists by DumRec. The remaining 259 letters were the "test set". Once a recognition model had been trained, its performance was evaluated by determining how many of the previously unseen test letters it could correctly categorize.

A graph of the performance of all three models on the test set is shown in Figure 5. To determine the percentages for NetRec, 20 runs with different initial weights were done and averaged together. For the FnetRec results, 11 runs were averaged. Since Dumrec is deterministic, its results are always the same. Overall DumRec's performance is best. It sucessfully recognized 74.3% of the test letters. FnetRec did almost as well, recognizing 72.84% letters corectly. NetRec's performance was not far behind with 70.45% letters classified corectly.

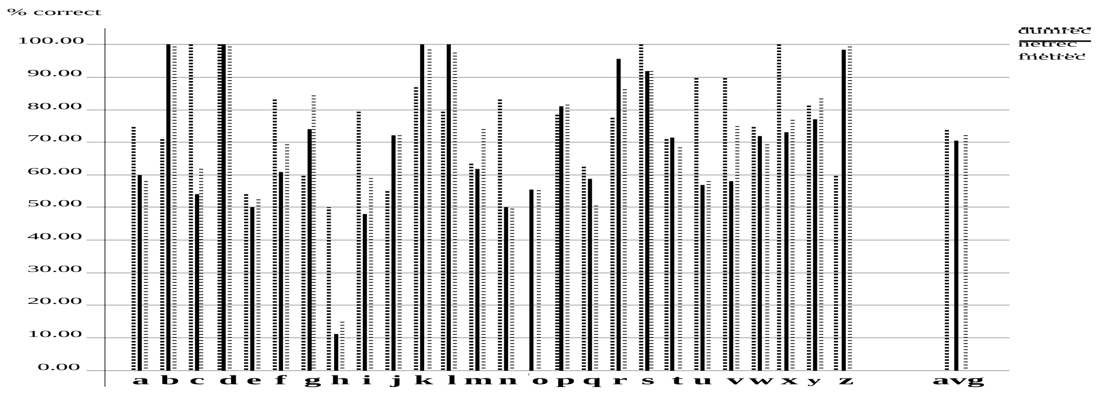

Figure 5: Percentage of correct classification for the three models on the testing set.

A few interesting trends are apparent in the results. One is that DumRec cannot properly classify any of the 'o's in the test set. This probably means that its weight system for closure is incorrect. There was not any one particular letter that it consistently mistook the 'o's for. On the other hand, the networks are incredibly bad at 'h's, usually mistaking them for 'b's or 'k's. The next largest discrepency comes with 'z's which both networks are able to recognize far better than DumRec.

In general the performance of NetRec and FnetRec are very similar. FnetRec is better in general, but not much, and both are "bad" and "good" at the same letters. This result was unexpected. We had though that FnetRec's performance would be significantly better than NetRec's since we were forcing it to consider certain features. We need to do more analysis (considering especially the weights to certain features) to find out why this happened. We are also interested in statistical analysis of the data.

[4]We are in the process of testing a version of FnetRec with a closure (6) subnet. Surprizingly its performance is only very slightly better than FnetRec without closure.

We plan to use the data reported here to compartively analyze the conceptually-based letter-recognition model used in Letter Spirit. Recognition in Letter Spirit is strongly based on recognizing conceptual parts (called roles) of letters (*e.g.,* two roles in a 'b' could be a "post" on the left side with a sideways "bowl" connected to the right of it) and then activating the proper letter based on the roles. Supplying further information about recognition in Letter Spirit is impossible here due to space constraints. See [Hofstadter and McGraw, 1993] for a detailed discussion of our approach.

Also planned is a set of psychology experiments using the same data on human subjects. A categorization task using limited presentation times (to give the correctness measure more sensitivity) will be run.

4 What is missing in these models?

4.1 DumRec

Although the DumRec model is fairly good at recognizing gridletters, its performance is not good enough for use in the Letter Spirit program. It makes far too many mis-categorizations. This is probably because its features are for the most part very syntactic, low-level features. Better recognition requires the use of higher-level features (*e.g.,* roles). It is at this abstract level that the creative play of Letter Spirit will take place. Comparing DumRec to the recognizer in Letter Spirit should shed light on this hypothesis.

4.2 Connectionist approaches

A connectionist approach to letterform recognition would probably be better at recognizing letters without regard to their style than at recognizing letters while considering style. A network can classify its input pattern as a strong or weak member of a given letter-category by varying the degree of activation on the appropriate letter-node of its output layer. If a letterform is ambiguous, more than one node can be activated. Such a network could probably make good "objective" judgements regarding a mystery letter's category.

One problem with a connectionist approach is that although a connectionist network can classify letterforms as strong or weak, it cannot be judge them in terms of style. It is possible that the stylistic aspects of a proposed 'a' are completely wrong (with respect to other letters) even though the shape itself is a reasonable member of the category 'a'. The only way around this would be to train another network on style and somehow coordinate the two. But since an entire gridfont in the target style does not yet exist, and there is no way of talking about what style is in terms of key defining concepts; not enough information exists for style training (as evidenced by the results of [Grebert et al., 1991] who attempted, unsucessfully, to tackle all of Letter Spirit with a simple network). What is needed is a network that does analogy. Unfortunately, analogy has yet to be successfully addressed in the connectionist paradigm. Until then, high-level cognitive activities like gridfont design will remain out of reach to connectionism.

References

[Fukushima, 1989] Fukushima (1989). Analysis of visual pattern recognition by the neocognitron. *Neural Networks*, 2:413-20.

[Grebert et al., 1991] Grebert, I., Stork, D., Keesing, R., and Mims, S. (1991). Network generalization for production: Learning and producing styled letterforms. In *Proceedings of the Neural Information Processing Systems Conference (NIPS) 1991.*

[Guyon et al., 1989] Guyon, I., Poujaud, I., Personnaz, L., and Dreyfus, G. (1989). Comparing different neural network architectures for classifying handwritten digits. In *Proceedings of the IJCNN-89*, pages II, 127–132, Washington, D.C.

[Hofstadter, 1985] Hofstadter, D. (1985). *Metamagical Themas: Questing for the Essence of Mind and Pattern.* Basic Books, New York. See especially Chapters 10, 12, 13, 23, 24, and 26.

[Hofstadter and McGraw, 1993] Hofstadter, D. and McGraw, G. (1993). Letter spirit: An emergent model of the perception and creation of alphabetic style. Technical Report 68, CRCC, 510 N. Fess, Bloomington, IN 47405.

[McGraw, 1990] McGraw, G. (1990). Dumrec lives — birth of a project with multiple programalities. Center for Research on Concepts and Cognition, Indiana University, 501 North Fess, Bloomington, IN 47405 (internal document).

[McGraw and Hofstadter, 1993] McGraw, G. and Hofstadter, D. (1993). Perception and creation of alphabetic style. Technical report, AAAI. Selected papers from the 1993 AAAI Symposium on Creativity and Artificial Intelligence. (in press).

[Regier, 1991] Regier, T. (1991). Learning spatial concepts using a partially-structured connectionist architecture. Technical Report TR-91-050, International Computer Science Institute, Berkeley, California.