Washington University in St. Louis
Washington University Open Scholarship

All Computer Science and Engineering Research | Computer Science and Engineering

Report Number: WUCS-92-17

1992

Energy-related Feature Abstraction for Handwritten Digit Recognition

Authors: Thomas H. Fuller Jr.

Most handwritten character recognizers use either graphical (static) or first-order dynamic data. Our research speculates that the mental signal to write a digit might be partially encoded as an energy profile. We used artificial neural networks (ANN) to analyze energy-related features (first and second time derivatives) of handwritten digits of 20 subjects and later 40 subjects. An experimenal environment was developed on a NeXTstation with a real-time link to a pen-based GO computer.
Although such an experiment cannot confirm an energy profile encoded in the writer, it did indicate the usefulness of energy-related features by recognizing 94.5% of the 600 test patterns after 29,000 random presentations of 800 training digits. This three-layer ANN had 54 input units (representing 29 trinary features), 4 hidden units (0, 2, 3, 4, 8, 10, 12, 15, 20, and 25 hidden units were tested), and 14 output units. Another ANN recognized 91.7% of the test digits after only 6,000 training presentations. Later testing with 40 subjects (including very erratic writing) resulted in 91% recognition.
This same feature abstraction was... **Read complete abstract on page 2.**

Follow this and additional works at: http://openscholarship.wustl.edu/cse_research

 Part of the Computer Engineering Commons, and the Computer Sciences Commons

Recommended Citation
Fuller, Thomas H. Jr., "Energy-related Feature Abstraction for Handwritten Digit Recognition" Report Number: WUCS-92-17 (1992). *All Computer Science and Engineering Research.*
http://openscholarship.wustl.edu/cse_research/528

Department of Computer Science & Engineering - Washington University in St. Louis
Campus Box 1045 - St. Louis, MO - 63130 - ph: (314) 935-6160.

Energy-related Feature Abstraction for Handwritten Digit Recognition

Complete Abstract:

Most handwritten character recognizers use either graphical (static) or first-order dynamic data. Our research speculates that the mental signal to write a digit might be partially encoded as an energy profile. We used artificial neural networks (ANN) to analyze energy-related features (first and second time derivatives) of handwritten digits of 20 subjects and later 40 subjects. An experimenal environment was developed on a NeXTstation with a real-time link to a pen-based GO computer.

Although such an experiment cannot confirm an energy profile encoded in the writer, it did indicate the usefulness of energy-related features by recognizing 94.5% of the 600 test patterns after 29,000 random presentations of 800 training digits. This three-layer ANN had 54 input units (representing 29 trinary features), 4 hidden units (0, 2, 3, 4, 8, 10, 12, 15, 20, and 25 hidden units were tested), and 14 output units. Another ANN recognized 91.7% of the test digits after only 6,000 training presentations. Later testing with 40 subjects (including very erratic writing) resulted in 91% recognition.

This same feature abstraction was tested in a Supervised Competitive learning (SCL) implementation which was free to create as many or few digit prototypes as was needed to recognize characters. Using the former data set (20 subjects), it created from 19 to 34 prototypes (depending on control parameters) and achieved 94.8% recognition.

This technical report is available at Washington University Open Scholarship: http://openscholarship.wustl.edu/cse_research/528

Energy-related Feature Abstraction for Handwritten Digit Recognition

Thomas H. Fuller, Jr.

WUCS-92-17

May 1992

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
Saint Louis, MO 63130-4899

This work was supported by the Kumon Machine Project.

Reprinted from the Proceedings of the Fourth Midwest Artificial Intelligence and Cognitive Science Society Conference.

Energy-related Feature Abstraction for Handwritten Digit Recognition

Thomas H. Fuller, Jr.
Department of Computer Science, Washington University[1]

Abstract: Most handwritten character recognizers use either graphical (static) or first-order dynamic data. Our research speculates that the mental signal to write a digit might be partially encoded as an energy profile. We used artificial neural networks (ANN) to analyze energy-related features (first and second time derivatives) of handwritten digits of 20 subjects and later 40 subjects. An experimental environment was developed on a NeXTstation with a real-time link to a pen-based GO computer.

Although such an experiment cannot confirm an energy profile encoded in the writer, it did indicate the usefulness of energy-related features by recognizing 94.5% of the 600 test patterns after 29,000 random presentations of 800 training digits. This three-layer ANN had 54 input units (representing 27 *trinary* features), 4 hidden units (0, 2, 3, 4, 8, 10, 12, 15, 20, and 25 hidden units were tested), and 14 output units. Another ANN recognized 91.7% of the test digits after only 6,000 training presentations. Later testing with 40 subjects (including very erratic writing) resulted in 91% recognition.

This same feature abstraction was tested in a Supervised Competitive Learning (SCL) implementation which was free to create as many or few digit prototypes as was needed to recognize characters. Using the former data set (20 subjects), it created from 19 to 34 prototypes (depending on control parameters) and achieved 94.8% recognition.

Energy-related features

The heart of handwritten character recognition is the selection of features to train and test the recognizer — in our case, an artificial neural network (ANN). Generally the extraction of features is based on static data fixed by the character's representation (Suen et al. 1980, Tappert et al. 1990, Rubine 1991). Some systems consider dynamic data related to the static representation such as the energy needed to transform a prototypical spline into the observed shape (Hinton et al. 1990) and derivatives of the shapes themselves (Rodin et al. 1992). We are not aware of feature extraction drawn from the energy imparted to the pen in the two-dimensional writing plane while drawing the character. This is the feature abstraction considered in this work.

In discussing the process by which characters are created with Dan Kimura, it occurred to us that humans write characters as easily with their eyes closed as with them open. The mental signal that creates a character apparently rests on more than the final representation of that character. We speculate that the generative signal sent to the hand muscles is — at least partly — an energy profile of the character.

---

[1] Assistant Professor, Department of Mathematics and Computer Science, Principia College

An energy profile may be characterized as force acting through distance. More formally, if **F**, **a**, and **v** are the force, acceleration and velocity vectors of the pen motion, the energy is given by

$$\int \mathbf{F} \cdot \mathbf{v}\, dt = \int m\mathbf{a} \cdot \mathbf{v}\, dt.$$

A presumed energy profile would be detectable in the velocities and accelerations of the pen in the two dimensions of the tablet surface. There are several approaches to seeking such a profile. One could attempt to derive specific relations between characters and the instantaneous values of the vectors **a** and **v**, or one could simply offer some representation of the **a** and **v** vectors to a artificial neural network and observe if it derives predictively effective relationships on its own.

This latter approach underlies the derivation of features in our feature preprocessing. Although it is undeniable that graphic features do influence the creation of characters, we chose to omit all graphic content (except the *number* of strokes, and, for early tests, the *extent* of the finishing stroke — i.e. distance from the starting point), and to limit the ANN to energy-related features to test the energy-profile hypothesis.

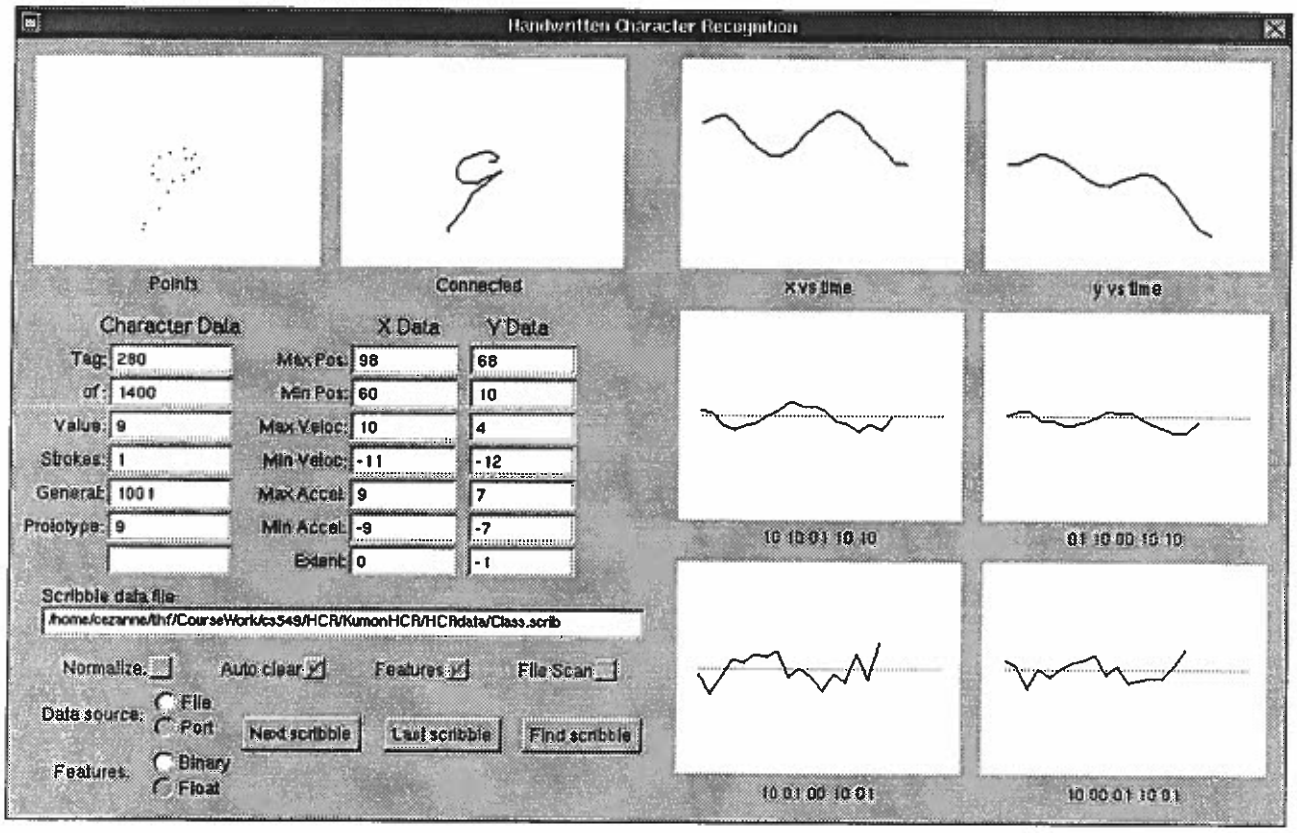
Feature extraction preprocessing

The data is captured by the **x** and **y** movement of the pen on a digitizing tablet. A series of points is combined into a *stroke*, and a series of strokes is combined into a *scribble* (after GO 1991, Chapter 15, and Morita et al. 1991). The algorithm assumes that the point sampling rate is constant over time. This is not a perfect assumption, but averaging makes it workable.

The feature extractor divides the scribble into several (4, 5, 6 or 7) intervals. It then calculates the average velocity and acceleration in the **x** and **y** directions for each of the intervals. The differences between successive points are recorded in velocity arrays (**x** and **y**). The differences between successive velocities are recorded in acceleration arrays (**x** and **y**). Average velocity or acceleration in each interval is compared to the maximum for the character. Each region was encoded as a trinary feature — positive, negative, or zero. (Zero indicates a value below an arbitrary threshold, say one third of the maximum.) In actual testing, the trinary features were encoded with two short integers — 10 01 00 represent negative, positive, and zero.

In the first four experiments shown in Figure 1 below, velocity and acceleration were divided into five intervals leading to 22 trinary features or 44 binary inputs. In the rest of the experiments, velocities were divided into seven regions, accelerations were divided into six regions, and extensions were omitted leading to 27 trinary features or 54 binary inputs. In all tests, the number of strokes composing the scribble was encoded as one or more-than-one.

The following illustration depicts a scribble for the digit nine. The left two graphs represent the input points as dots and connected dots. The top two right graphs represent x *versus* time and y *versus* time. The others represent velocity and acceleration respectively. The feature encoding for velocity and acceleration appears below each respective graph. In this illustration, velocity and acceleration were divided into five intervals each. For example, in the **x** velocity graph, the first section of the graph dips downward (since the nine is first formed by moving in the negative **x** direction). Then it rises in the middle and dips down again. Therefore the x velocity encoding is 10 10 01 10 10 — negative negative positive negative negative. None happened to be zero.



The artificial neural net simulator

Training and test patterns were assembled from the twenty subjects (after duplicating two of the 3's to make up for omissions by two subjects). Features were extracted and stored. Eight hundred patterns were used for training (four of each individual's patterns for each character). The remaining 600 patterns were used to test the generalization of the network. Later this data was combined with a similar set for twenty other subjects giving a training file of 1600 scribbles and a testing file of 1200 scribbles.

The three-layer ANN had input units representing the trinary features. In all but the first experiment, an additional input was added whose value was always one. The hidden layer consisted of 0, 2, 3, 4, 8, 10, 12, 15, 20, or 25 hidden units. The final layer had (originally) 10 output units. The output unit with the highest activation value identified the digit. As described below, it was later observed that expanding the prototypes available to the ANN improved its recognition ability. (For example, some writers draw five with one stroke, some with two; a prototype was created for each. Some writers draw an eight as two independent circles; most draw the eight with one interleaved stroke. Again prototypes were made available for each.)

The input layer was fully connected to both the hidden layer and the output layer. The hidden layer was fully connected to the output layer. There were no lateral connections, that is, no connections within the same layer. Error values were backpropagated to the hidden and input layers using the gradient descent method as developed in Rumelhart and McClelland (1986).

We used the activation and error functions of Kalman and Kwasny (1991, 1992), namely,

$$a = \lambda(x) = (1 - e^{-x}) / (1 + e^{-x}) \qquad \text{(activation function)}$$

$$e = \Sigma (c^2 / (\lambda'(x)) = \Sigma (c^2 / (1 - a^2) \qquad \text{(error function)}$$

Where c is the difference between the *training value* (1 if correct, -1 if incorrect) and the actual activation value of the unit. Each trial consisted of backpropagating a single randomly-selected pattern through the ANN. The *learning rate* (**Eta**, how much change in weights per back propagation) and *momentum* (**Alpha**, the degree to which the last change in weight contributes to the current) varied as shown. The individual digit performance is at the right.

Figure 1 — Summarized data for nine tests

Units in	hid	out	Eta	Alpha	Seed	Trials	<---------- Performance on test data ----------> Total	0	1	2	3	4	5	6	7	8	9
(5 velocity intervals and 5 acceleration intervals, x-extension and y-extension)																	
50	20	10	.0005	.3	1	25,000	92.0%	57	48	60	57	60	51	54	57	57	51
51	20	10	.0005	.9	30	6,000	91.7%	57	52	59	56	59	48	57	54	56	52
51	20	10	.0005	0	30	80,000	91.8%	55	51	59	55	59	49	59	55	56	53
51	12	14	.0005	.3	90	50,000	91.8%	59	52	59	56	59	48	58	53	54	53
(7 velocity intervals and 6 acceleration intervals, no extension data)																	
55	4	14	.0010	.6	1	35,000	93.0%	55	54	55	58	58	57	48	57	58	58
55	4	14	.0005	.6	1	26,000	93.0%	48	53	57	57	58	59	52	56	60	58
55	4	14	.0005	.3	1	470,000	94.7%	50	53	58	58	58	59	56	58	60	58
(Same, but using inline *tanh* function instead of the *exp* function: slower per trial, but *fast* convergence)																	
55	4	14	.0010	.4	1	29,000	94.5%	55	54	58	59	58	58	51	57	59	58
(2800 scribbles from 40 subjects, 7 velocity intervals and 6 acceleration intervals, no extension data)																	
55	6	17	.0007	.4	1	682,000	91.0%	91	108	117	117	111	114	96	114	114	110

For all but the last run, the training set was 800 patterns and the testing set was the other 600 patterns. (A perfect score for one digit is 60.) The digits one and five were consistently more difficult for the 22 trinary feature version to recognize. Zero and six were most difficult for the 27 trinary feature version. Increasing the number of prototypes (to 12, 14, and 17) allowed higher recognition rates. The last experiment represents 40 subjects (7 samples * 10 digits * 40 subjects = 2800 scribbles) including some very poor writing samples. For this experiment, 1600 digits were used for training and the other 1200 were used for testing.

The first experiment required about four and a half minutes on our 68040-based NeXTstation (25MHz) . The second experiment (6,000 trials) took about a minute. The last (682,000 trials) ran about two hours. Once trained, the system derives a recognition in about 6 milliseconds (including extracting features and forward propagation through the ANN).

On-line testing with other subjects

Later the system was tested by various right and left-handed users (recognizing well above 90%). The following pictures (both ostensibly eight) illustrate the difficulties created by poor writing.

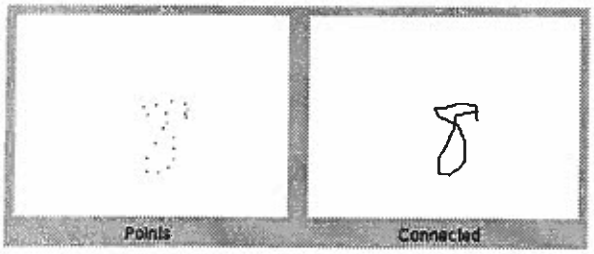


Figure 2 — Tag 88

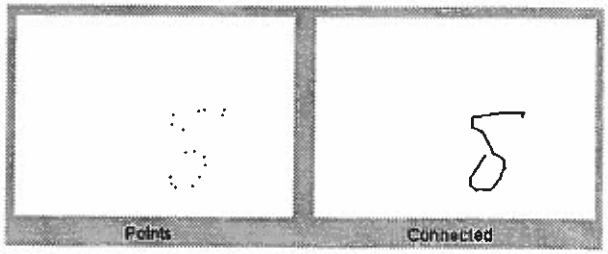


Figure 3 — Tag 89

Below are the output activations for the two "eights." Notice how strongly the eighth output unit dominates in the first one (Tag 88). In Tag 89, observe that no output unit is even positive, and that 5 wins, 1 is the second choice, and 8 is a weak fourth choice. In fairness, though, Tag 89 might be mistaken for a single-stroke five even by some humans.

```
Tag:Value:Guess -> Activation (lambda) for digits 0 to 9
                0      1      2      3      4      5      6      7      8      9
88:8:8 ->  -0.992 -0.958 -0.881 -1.000 -1.000 -0.973 -0.983 -0.999 +0.797 -0.990
89:8:5 ->  -0.999 -0.592 -0.968 -0.999 -1.000 -0.466 -0.960 -1.000 -0.829 -0.802
```

The system proved resilient to many ambiguous features. The presence or absence of crossing strokes did not significantly reduce the recognition rate of sevens. It also recognized "curly nines" made without cusps (exactly as an upside-down 6). Differences in size by a linear factor of four had no appreciable effect on the recognition rate. This is due to the implicit normalization in the velocity and acceleration data divided by maxima.

Supervised Competitive Learning (SCL) — "unlimited" prototypes

An analysis of the items missed by the network revealed that it was defeated by the various ways in which we make our numbers. The two worst cases derive from the seven members of the first test group who make a five with one stroke, and the four members who make an eight with two (circular) strokes. We hypothesized that the ANN was handicapped by the effort to converge to such disparate *prototypes* of the digits five and eight. To overcome this, we borrowed the concept of prototypes from the Adaptive Resonance Technique (ART) model. We created two additional prototypes for five and eight as noted above, and later two more.

SCL lets the system freely create its *own* prototypes from the stream of randomly-selected inputs. The first sample becomes the first prototype on which it is trained; the second may become a second prototype if its teaching value is different or if it differs *significantly enough* from the first. An individual ANN for each new prototype is trained positively on its exemplar and negatively on other exemplars. Each is free to add similar input samples to its exemplar set. This permits better generalization. The control thresholds (for deciding that a difference is sufficient to create a new prototype, and for deciding the sufficiency of training) influence the number of prototypes created. The following figure gives the results of early experimentation with 7 velocity intervals and 6 acceleration intervals, and 4 hidden units in each prototype. (A trial here consists of backpropagation through *only one* prototype's ANN, so each trial requires less than a tenth the time of the earlier ANNs.) A fuller report is in Kimura and Fuller (1992).

Figure 4 — Summarized data for SCL (early results)

Prototypes created	Eta	Alpha	Seed	Trials	<---------- Performance on test data ----------> Total	0	1	2	3	4	5	6	7	8	9
19	.0005	.4	1	211,761	92.7%	51	53	57	57	59	59	49	57	59	55
35	.0005	.4	1	1,510,670	94.7%	55	53	59	60	56	58	50	58	60	59
26	.0005	.4	1	50.790	91.7%	55	52	56	59	56	57	42	57	60	56
26	.0005	.4	1	2,791,989	94.8%	56	53	57	59	59	58	53	59	59	56

Conclusions

This work was part of our effort to develop pen-based workstations for teaching mathematics to children. The limitations of earlier character recognition algorithms led us to search in new directions for feature extraction. Energy-based features are one promising avenue.

Although this research could never confirm an energy profile encoded in the writers of digits, the strong results indicate the usefulness of such energy-related features. Our future research, combining energy-based features with static positional data and with angular displacement data, should result in better recognition and a larger "vocabulary" including letters and gestures. The Supervised Competitive Learning (SCL) approach appears very promising for the demands of more flexible recognition including letters, command gestures, and even user-defined gestures.

References

GO Corporation. *PenPoint API Reference*. Foster City, California: GO Corporation, 1991.

Hinton, Geoffrey E., Christopher K. I. Williams and Michael D. Revow. "Adaptive Elastic Models for Hand-Printed Character Recognition." University of Toronto, Canada, 1990.

Kalman, Barry L. and Stan C. Kwasny. "A Superior Error Function for Training Neural Nets," *Proceedings Of the International Joint Conference on Neural Networks*, Seattle, Washington, 1991, Vol. 2, 49-52 .

Kalman, Barry L. and Stan C. Kwasny. "Why Tanh: Choosing a Sigmoidal Function", (submitted to) *International Joint Conference on Neural Networks,* 1992.

Kimura, Takayuki D. and Thomas H. Fuller, Jr. "Supervised Competitive Learning (SCL) with Backpropagation Networks." Washington University Technical Report, WUCS-92-xx, 1992.

Morita, Toshihiro, and Shinji Mori. "An Internal Model for Pen-based Input Data and the Program Libraries used with this Model."(Japanese) *Human Interface*, Vol. 6 (1991), 261-268.

Rodin, Ervin Y., Yuanlan Wu, and S. Massoud Amin. "Character Recognition:Qualitative Reasoning and Neural Networks." *Mathematical Computer Modelling*, Vol. 16, No. 2 (1992), 95-102.

Rubine, Dean. "Specifying Gestures by Example." *SIGGRAPH 91*. ACM, 1991.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams. "Learning Internal Representations by Error Propagation," in Rumelhart, David E. and James L. McClelland (eds.) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol 1*. Cambridge, MA: MIT Press, 1986.

Suen, Ching Y., Marc Berthod, and Shinji Mori. "Automatic Recognition of Handprinted Characters — The State of the Art." *Proceedings of the IEEE*, Vol. 68, No. 4 (April 1980) 469-487.

Tappert, Charles C., Ching Y. Suen and Toru Wakahara. "The State of the Art in On-Line Handwriting Recognition." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 12, No. 8 (August, 1990), 787-808.