Journal of Scientific Exploration, Vol. 3, *No.* 2, pp. 185–200, 1989
Pergamon Press plc. Printed in the USA.

0892-3310/89 $3.00+.00

Searching for "Signatures" in Anomalous Human–Machine Interaction Data: A Neural Network Approach*

DEAN I. RADIN†

Department of Psychology, Princeton University, Princeton, NJ 08.544

Abstract—An artificial neural network was used to explore whether unique "signatures" could be found in **data** collected in experiments studying the effect of intention on the statistical behavior of random number generators. Results showed that a network trained with a back-propagation technique was able to learn to associate 32 different individuals with the data they generated, then successfully transfer that knowledge to new data. It is recommended that similar experiments studying anomalous human–machine interactions should attempt to identify person-specific patterns in data in addition to measuring the magnitude of effects; parallel processing analysis techniques are also recommended.

Introduction

One of the most conspicuous and frustrating aspects of the study of human behavior is the fact that people are different. This distinctiveness allows us to identify people based on properties such as fingerprints, handwriting, voice, gait, DNA, personality, and so on (Weisburd, 1988). The importance of individual differences has long been noted in psychological experiments (Barlow & Hersen, 1984), thus it should not be surprising to find that such differences have also been reported in parapsychological experiments (Babu, 1987; Berger, 1988; Jahn & Dunne, 1986, 1987; Jahn, Dunne, & Nelson, 1987; McConnell, 1989). The present study investigated the hypothesis of individual differences, called "signatures," in human–machine interaction data collected by the Princeton Engineering Anomalies Research (PEAR) laboratory (Nelson, Dunne, & Jahn, 1984, 1986).

In the PEAR studies, individuals attempted to influence the output statistics of electronic random number generators (RNG) solely via mental intention. In a typical RNG experiment conducted at PEAR, an RNG is set to produce a series of 200 truly random bits at the press of a button; this is called one **trial.** A person watches a digital display that shows how many times the random samples matched an alternating "target" bit over the course of 200 generated bits. By chance, one would expect an average of 100

* A version of this paper was presented at the 7th Annual Meeting of the Society for Scientific Exploration, Cornell University, Ithaca, NY, June 2–4, 1988.

† Present address: Intelligent Systems Laboratory, Contel Technology Center, 15000 Conference Center Dr., P.O. Box 10814, Chantilly, VA 22021-3808.

such matches, and under control conditions the distribution of trial scores closely matches the theoretically expected binomial distribution (Nelson, Dunne, & Jahn, 1986). When an individual is asked to aim for high numbers, he or she mentally tries to cause the RNG to produce trial scores greater than 100. In a low aim condition, trial scores less than 100 are intended; and in a control condition, no mental intention is applied. One *run* consists of 50 aim high, 50 aim low, and 50 control trials. One *series* consists of 50 such runs, which takes about five hours of data collection to complete.

Aggregate experimental results to date reveal statistically significant correspondences between the intentional "aim" and the shift of the RNG output statistics (Jahn & Dunne, 1986, 1987; Jahn, Dunne, & Nelson, 1987). Meta-analysis of over 600 similar RNG experiments conducted by some 67 other investigators indicates that the anomalous correlation is not due to methodological shortcomings or undetected artifacts in the PEAR RNG (Radin & Nelson, 1987; Radin & Nelson, in press).

One intriguing observation about this data (beyond the mere existence of an anomalous correlation) is that individuals seem to perform in consistently unique patterns. Nelson, Dunne and Jahn (1986) report that data produced in one run often bears resemblance to data produced in other runs, and such similarities appear to be unique to the individuals who produced the data. However, other than providing face validity based upon graphical representation of the data (Jahn & Dunne, 1987), some rudimentary statistical analyses (Babu, 1987), and corroborating observations in similar types of experiments (Berger, 1988; McConnell, 1989), a "signature" effect has not been rigorously demonstrated. If such an effect were confirmed, it would suggest that experiments on human consciousness would provide more useful and revealing information with single-subject designs rather than conventional multisubject designs (cf. Barlow & Hersen, 1984).

To explore the idea of person-specific signatures in the PEAR RNG data, I used a powerful computational technique that is proving to be exceptionally adept at discovering weak patterns in noisy data. As described below in more detail, the general term for this approach is *neural network analysis,* and the specific training procedure used in this study is called *back-propagation* (Jones & Hoskins, 1987; Rumelhart, Hinton, & Williams, 1986). The study involved training a network to associate given data with given individuals, then observing whether the trained network could successfully identify persons based upon data that the network had not "seen" before.

Neural Networks

New computational techniques and models, variously called neural networks, parallel distributed processing, connectionism, and so on, are attracting wide interest within the disciplines of artificial intelligence, the cognitive sciences, and the neurosciences. These models, analogous to biological neural networks, are rapidly advancing the theory and development

of self-organizing, adaptive machines as well as solving previously intractable problems in artificial intelligence (Materna, 1987; Shriver, 1988).

Neural networks are a form of parallel processing based upon research about how the brain encodes and processes information. The power of these networks rests upon the finding that when numerous elementary processing units are richly interconnected under the right conditions, they can automatically learn to associate arbitrarily complex inputs with arbitrarily complex outputs. Properly configured, these networks can also implement self-organizing associative memories, automatically derive statistical descriptions of spatial and spatiotemporal data, and autonomously acquire knowledge by observation.

An essential idea underlying neural networks may be illustrated by analogy with **a** bee hive. A hive is a complex, dynamic community with intelligent organization and structure, created and supported by individually simple creatures. Instead of being controlled by a central, guiding intelligence, a hive seems to be maintained by the hundreds of thousands of interactions among bees. For another analogy, consider the collections of elementary cells that "cooperate" to form complex organic structures called organs.

In engineering terms, a neural network may be described as a "parallel dynamic system with the topology of a directed graph [which] can carry out information processing by means of its state response to continuous or initial input" (Materna, 1987). Information processing involves interactions among large numbers of artificial neurons. These neurons, called nodes or units, have four main components, as illustrated in Figure 1:

input connections, through which the unit receives activation from other units, a *summation function* that combines various input activations into a single activation, an

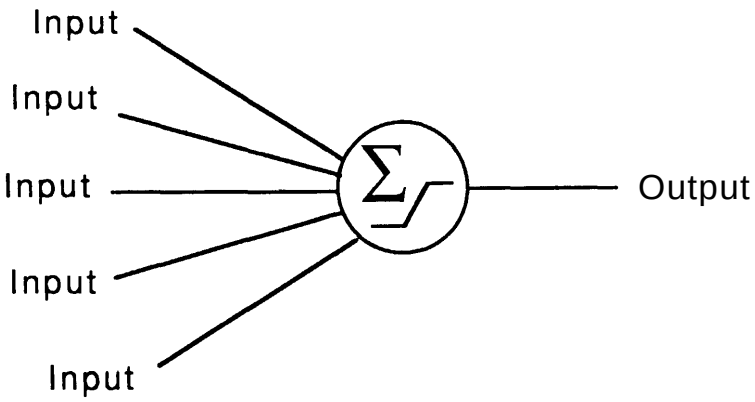

Fig. 1. Example of a typical node in a simulated neural network. A node may have an arbitrary number of input connections, and any number of output connections (only one output is shown).

output function that converts summation of input activation into output activation, and

output connections by which **a** unit's output activation arrives as input activation at other units in the system (Jones & Hoskins, 1987).

Such networks have been successfully taught to automatically recognize human faces, read text, make medical diagnoses, balance objects, and so on, without conventional algorithmic programming (Materna, 1987; Shriver, 1988; Widrow & Winter, 1988). Because a more complete description of the techniques and applications of neural networks is beyond the scope of this paper, I will proceed by concentrating on the present implementation.'

Method

The Data

Data used in this study were originally collected at the PEAR laboratory as part of their research on human–machine interactions with truly random event generators[2] The dataset consists of 87 series of data ("series" as defined above), produced by 33 different individuals over approximately a nine-year period. The data are in the form of run scores (average scores obtained over 50 successive trials). A typical series is represented in the form of 150 lines of data (one run score per line, and one set of 50 lines for each of the tripolar intentions).

Other types of information were available from computer archive files, including such items as whether the run was in "volitional" or "instructed" mode, whether the RNG was in an "automatic" or "manual" condition, and so on (Nelson, Dunne, & Jahn, 1984), but only run scores, intentional aim direction, and operator identity numbers were used in the present analysis.

Data Preparation

I chose a straightforward method of presenting data to a network, one that required only six numbers to characterize an individual's performance over one series. This had the advantage of simplicity, which was desirable in this exploratory study, but the disadvantage of compressing nearly a million bits of temporally collected data per person into only a few static summary statistics.

Because the main purpose of this study was to see whether a network could learn to identify an operator based solely upon his or her data, I actually needed *two* datasets associated with each operator—one would be used to train the network and the other would be used to see whether the trained network could transfer its knowledge to new data. Thus, each series

(a total of 50 runs) was split in half, using the first half (25 runs in each of the three aim conditions) as the training set and the second half (25 runs) as the transfer set. This half-split method was used, rather than creating training and transfer sets out of separate series, partially because only 20 individuals had produced two or more series, but more importantly, because I speculated that consistent human performance would be more evident within a given series rather than between different series.[3]

Individual run scores were transformed into standard normal deviates against chance expectation (i.e., Z scores), then for each series the following six data items were generated to produce the training set: (1) an overall (Stouffer) Z score for the first 25 runs under high aim intention, (2) the average Z^2 of those 25 high aim runs,[4] (3) a Stouffer Z for the first 25 runs under low aim intention, (4) the average Z^2 of those 25 runs, (5) the Stouffer Z for the first 25 control intention runs, and (6) the average Z^2 of those runs. For the transfer set, the same six data items were determined, except using data for runs 26–50.

This procedure produced 87 items in the training set and 87 items in the transfer set, representing data for 33 operators. To further simplify the interpretation of the training-transfer test, I used only an operator's first series, and only the first 32 operators (for reasons described below). This resulted in two datasets, each consisting of 32 lines of data.

Control *Datasets*

If the transfer test showed that say, 50% of the operators were correctly identified, the "signatures" results would be self-evident. But, if say, only 2–3% were identified, as expected given the very small magnitude effects reported by the PEAR lab, then a statistical assessment would be necessary. Therefore, two control datasets were generated to compare against the transfer test results. First, a random dataset was generated by simulating the PEAR experimental protocol with a pseudorandom number generator (PRNG). The random dataset was created in five steps: (a) the PRNG was used to generate one trial of 200 random bits, (b) this was repeated 50 times to generate one run, (c) a Z score was determined from this run, (d) this was repeated 25 times to simulate a half-series, and (e) steps a–d were repeated 32 times to simulate the 32 series used from the PEAR dataset.

Then a scrambled dataset was generated by using the original data, but with operator numbers chosen uniformly at random, with replacement, over the range 1–32. Note that I could not create a suitable scrambled dataset by simply shifting operator codes by one, because this would guarantee that at least half of the operators would be identified in that "scrambled" data.[5] The UNIX System V™ 48-bit multiplicative congruential PRNG, called *drand*48, was used to provide pseudorandom numbers in the control study. This generator has passed extensive first-order and higher-order randomness tests (Radin, 1985; Roberts, 1982).

Description of Network

The program used to implement the network was originally written in Fortran, later recoded into C, then adapted and revised by the author for the present application.[6] A Silicon Graphics IRIS **4D**™ workstation was used to run the program.

The network used for this study was based upon a three-layer model—input, hidden, and output—as illustrated in Figure 2. There were six input nodes, corresponding to the above six values associated with each operator; between 10 and 30 "hidden" nodes (so-called because they are not directly accessible to the outside world), depending on factors such as how fast the network was to learn, how complex the inputs were, and so on; and 5 output nodes, encoding 32 operator identities as binary codes.

For the present study, values applied to the six input nodes consisted of the Stouffer Z scores and average Z^2 associated with each operator's data, as mentioned above. The desired network output was the unique binary code associated with each operator. As is customary in such networks, the interconnecting links between the input and hidden, and hidden and output nodes were initially set to random values (typically using a uniform random range between $\pm$ 1.0).

Parallel Processing in the Network

The following events occur in one simulated parallel processing cycle: (a) The first line of the 32 line training dataset is read by the program, which applies the six data values for the first operator to the input nodes. These

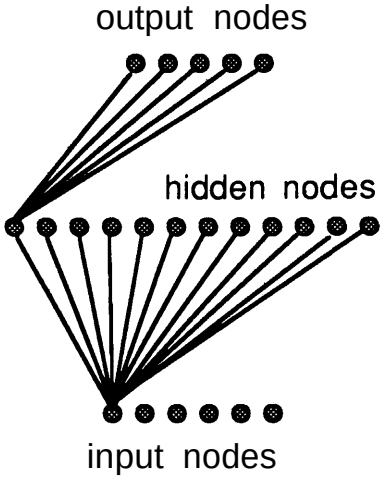

Fig. 2. Structure of the neural network used in this study. Links from the first input to first hidden node, and first hidden to output node are shown to illustrate the connectivity of the network. The actual network was fully interconnected between input and hidden, and hidden and output layers. Here we see 12 hidden nodes; the networks tested used 10, 16, and 30 nodes.

values cause the input nodes to spread activation out along their links towards the hidden layer in proportion to each link's strength or weight. (b) The hidden nodes sum the activation received from the input nodes, depending on the value of the connecting links, then, in accordance with an activation function they send activation towards the output nodes. (c) The output nodes sum the activation sent from the hidden nodes, ending the "feed-forward" phase of this cycle.

Now a "teacher signal" comes into play: (d) An arbitrary but unique 5-bit code (e.g., "00111" identifies operator number 7) associated with the input data is compared against the actual continuous-valued results residing at the output nodes, generated from step (c). (e) The difference between the observed activation from (c) and the desired "signal" from (d) is now propagated *backwards* through the network, changing the values of the interconnecting links so as to minimize that error on the next pass if the same input values were applied (thus the name of the learning rule, *back propagation*, see the Appendix for the basic equations).[7] This ends one parallel processing cycle.

Now, (f) the same process is applied for the remaining 31 operators' data: First a feed-forward phase, followed by back propagation of the error. Applying these 32 lines of input to the network is called one *pass.* With the present network and form of the data, it typically takes between 1,000 to 3,000 such passes to train the network to perfectly associate the observed data with the desired operator codes. (Some readers may find it surprising that a network can learn to perfectly associate random-looking input data with arbitrary output codes; this is one of the interesting properties of trainable neural networks.)

It is important to note that an untrained network always starts with random interconnection weights between nodes. The hidden nodes act as a kind of mathematical space in which to compute the problem of associating inputs with outputs, thus if we begin with say, 20 hidden nodes, we have a topological flexibility roughly equivalent to a 20-dimensional space. With a space this large, there are a vast number of "solutions" to the problem, and it is likely that a new solution will be found each time the network is trained from scratch. In other words, training the network once will not guarantee the "best" solution, only an acceptable solution (i.e., the network does not know that we want to transfer its knowledge to other data; it just solves the association problem). Because I was interested in exploring the possibility of knowledge transfer, the process described above in steps (a)–(f) was performed repeatedly, forming a *distribution* of transfer results against which to compare the random and scrambled dataset results.

Transfer Test

The transfer test consisted of 5 steps:

(a) The network was trained (from a random starting state) on the 32-line data file until the error between input and output values was less than 0.05

per output node at the end of a pass. From experience this was determined to take between 1,000 and 3,000 passes.

(b) The trained network was tested against the transfer dataset by sending each of the 32 items in the transfer dataset through the net in one feed-forward pass, then comparing the values obtained at the output nodes against the desired values. The observed value at an output node was considered to be 0 if the activation was less than 0.5, otherwise it was considered to be a 1.[8] The number of correct bit-matches was then used as a measure of success for the transfer test. For example, if all 5 bits of an operator code number were identified correctly after passing that operator's data through the network, the data-operator relationship learned by the network during the training phase would have transferred perfectly (in informational terms, we could think of the transfer test as having "transmitted" 5 bits of information). If 4 bits were correctly identified, this would constitute an 80% match (transmitted 4 bits of information), and so on.

(c) The same process was conducted for the random dataset and the scrambled dataset, producing for each of the three tests the number of perfectly correct matches (5/5 bits correct) out of 32 operators, 4 out of 5 correct matches (out of 32 operators), and 3 out of 5 (out of 32 operators). Thus, the score for a perfect transfer test would be 32 (all operators correctly identified with 5 out of 5 bits correct).

(d) Steps a–c were repeated 100 or more times to form a distribution of results for the transfer, random, and scrambled tests.

(e) The means of the distributions obtained in step d were compared with a *t*-test.[9] The null hypothesis is that there are no consistent patterns within the data, so there is nothing that can be transferred, and thus there should be no differences among the transfer, random, and scrambled dataset distribution means. The alternative hypothesis is that there is something consistent in the data that is associated with each operator; that these associations can be learned; and that these can be detected in independent data produced by those same operators. These effects would manifest in the present case by shifting the transfer distribution mean positive with respect to the random and scrambled distribution means, and the random and scrambled means should not differ significantly from one another. Thus, three t-tests were planned in advance: Transfer vs. Random, (TR) Transfer vs. Scrambled (TS), and Random vs. Scrambled (RS).

Results

In considering the results, it is important to note that the values at the output nodes of these networks resemble Bernoulli trials with $p = 0.5$. For example, we would expect to find approximately one operator correctly identified by chance (say, one "hit") out of 32, or approximately 5 hits out of 32 where 4 out of 5 bits were correctly identified, and so on. Actually, the

chance expected value at the (binary transformed) output nodes also depends on the initial interconnection weights in the network, on the form of the activation and learning rules, and on the input values. With these caveats in mind, by rough approximation we would expect to see about one operator completely identified by chance (i.e., all five code bits correct), and one completely missed (i.e., all five bits missed).

Table 1 shows results of tests using three network configurations. Each of the simulations took about 6 hours to complete on a Silicon Graphics IRIS 4-D workstation.[10]

Discussion

If the signatures hypothesis is true, it would manifest in this study by making the number of correctly identified operators greater in the transfer condition than in either the random or scrambled conditions. We would expect to find significant differences between the transfer vs. random and transfer vs. scrambled conditions, and a nonsignificant difference for the random vs. scrambled condition. This is what the results show (Table 1). The two histograms in Figure 3 display the distributions for the "4/5 bits identified" results.

Another way of illustrating transfer in this study is by examining the difference between means of perfectly identified (5/5 bits) and perfectly *unidentified* (0/5 bits) operators for each of the three conditions. Under the signatures hypothesis, we would expect to find more cases of 5/5 code bits identified than 0/5 code bits identified in the transfer condition, but not in the random or scrambled conditions. Because the 5/5 and 0/5 means within the three conditions are not fully independent, the statistics shown in Table 2 are based on t-tests for differences between correlated pairs of means. Figure 4 shows the separation between the distributions for the training-transfer test.

Given these results, one may wonder—under the assumption that within-person performance is reasonably consistent—whether repeatedly presenting a network with multiple examples of the same operator's data (i.e., from different experiments) would improve the transfer rate. Table 3 shows test results for correctly identified operators (5/5 code bits correct) in a 20 hidden node network trained on three independent datasets for each of nine people. (Nine persons in the PEAR database had completed three separate series.)

Comparing results in Table 3 with those in Table 1 suggests that there may be an advantage in presenting networks with repeated, independent views of the same operators' data. More research is needed to determine ways of training networks to recognize this data, as well as to select optimum statistical parameters to use as inputs. In addition, even though the transferred information in the present study was statistically significant, the ab-

TABLE 1
Means, standard deviations, and t-tests for transfer, random, and scrambled tests, using actual and random training-transfer data; all results are based upon N = 100 training-transfer repetitions

			10 Hidden Nodes[a]			
Bit matches (#/5)	0	1	2	3	4	5
Transfer						
Mean	0.61	3.96	9.07	10.88	6.02[b]	1.46[c]
SD	0.69	1.78	2.49	2.68	2.12	1.13
Random						
Mean	1.13	5.38	10.17	9.39	4.91	1.02
SD	1.00	1.97	2.53	2.60	2.11	0.91
Scrambled						
Mean	0.93	4.91	10.20	10.02	5.04	0.90
SD	0.96	2.22	2.34	2.65	1.88	0.81
			16 Hidden Nodes[d]			
Bit matches (#/5)	0	1	2	3	4	5
Transfer						
Mean	0.74	3.79	8.86	11.28	6.19	1.14
SD	0.74	1.51	2.40	2.52	1.91	0.92
Random						
Mean	0.95	4.86	10.08	10.20	5.03	0.88
SD	1.01	2.29	2.48	2.81	2.09	0.90
Scrambled						
Mean	1.12	4.88	9.76	10.17	5.05	1.02
SD	1.05	2.03	2.66	2.45	2.04	0.96
			30 Hidden Nodes[d]			
Bit matches (#/5)	0	1	2	3	4	5
Transfer						
Mean	0.94	3.52	8.28	11.62	6.52	1.12
SD	0.82	1.62	2.27	2.54	1.70	1.00
Random						
Mean	0.87	5.01	10.10	10.23	4.78	1.01
SD	0.92	1.87	2.61	2.98	2.05	1.02
Scrambled						
Mean	0.98	5.07	9.96	10.07	4.89	1.03
SD	0.94	2.21	2.58	2.63	2.37	1.06
			t-tests			
10 Hidden nodes						
t(T-R)[e]	−4.26	−5.32	−3.08	**3.97**	**3.69**	**3.02**
t(T-S)	−2.69	−3.32	−3.29	**2.27**	**3.44**	**4.01**
t(R-S)	1.44	1.58	−0.09	−1.69	−0.46	0.98
16 Hidden nodes						
t(T-R)	−1.67	−3.88	−3.52	**2.85**	**4.08**	**2.01**
t(T-S)	−2.94	−4.29	−2.50	**3.14**	**4.06**	0.90
t(R-S)	−1.16	−0.07	0.88	0.08	−0.07	−1.06
30 Hidden nodes						
t(T-R)	0.57	−5.99	−5.24	**3.53**	**6.50**	0.77
t(T-S)	−0.32	−5.63	−4.86	**4.22**	**5.56**	0.61
t(R-S)	−0.83	−0.21	0.38	0.40	−0.35	−0.14

[a] Trained 4,000 passes.
[b] The network identified 4 out of 5 operator code bits for an average of 6.02 operators out of 32.
[c] The network identified all 5 operator code bits for an average of 1.46 operators out of 32.
[d] Trained 2,000 passes.
[e] All t-tests for differences between independent means, 198 degrees of freedom. Significant results, one-tailed, are highlighted in bold for emphasis.

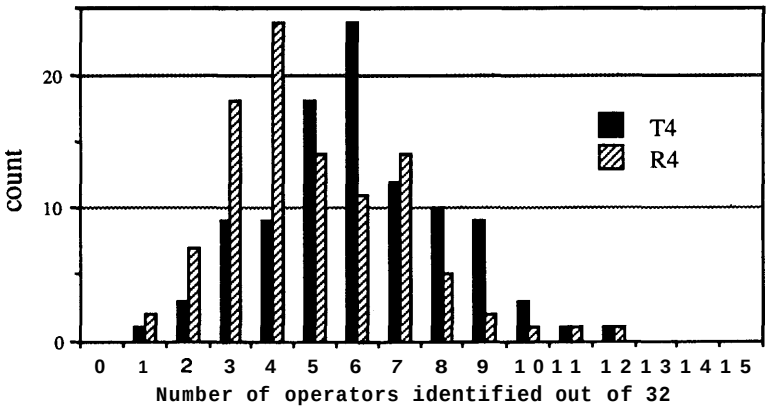

Fig. 3. Histogram for the number of times that four out of five operator code bits were correctly identified, out of a total of 32 operators, using a 10 hidden node network. T4 refers to the training-transfer test and R4 refers to the training-random test. Thus, in 100 independent repetitions of the training-transfer test, the mean number of operators in which 80% of the operator code was correctly identified was 6.02, whereas the equivalent mean in the random test was 4.91. Compare these figures with 5.0, which is the number expected by chance assuming that the output nodes are Bernoulli trials with $p(\text{hit}) = .5$.

solute magnitude was miniscule. Obviously, if the original "signal" (i.e., influence on an RNG output) were more robust, then the neural network would be more likely to recognize a potential signature. Thus, research is also needed on ways of statistically increasing the magnitude of these anomalous influences.

Conclusion

This study suggests that a neural network may learn to associate data produced by a random number generator with the identity of individuals who attempted to "mentally influence" the output statistics of the generator. Results of tests with different network configurations suggest that it may

TABLE 2
Paired t-tests (99 df) for differences between correlated means, for 5 vs. 0 correctly identified operator code bits, in the training-transfer (T), training-random (R), and training-scrambled (S) conditions; significant results are highlighted in bold

	Paired t-tests		
Hidden Nodes	T5-TO	R5-RO	S5-S0
10	**6.06**	−0.75	−0.24
16	**3.17**	−0.51	−0.64
30	1.35	1.03	0.35

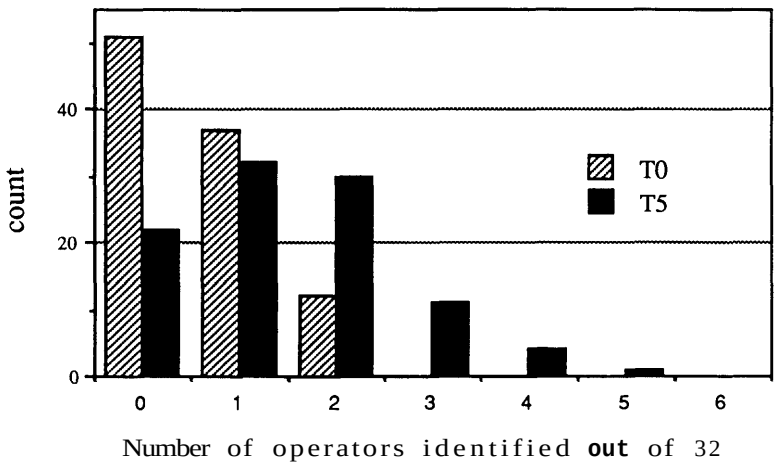

Fig. 4. Histogram for five and zero operator code bits correctly identified, out of a possible 32 operators, for the training-transfer (TO & T5) condition, using a 10 hidden node network. In other words, in 100 training-transfer test repetitions, the mean number of times that zero operator code bits were identified was 0.61, whereas the mean number of times that all five operator code bits were identified was 1.4. Results shown in Table 2 show that this separation is seen only in the transfer datasets, and not in the random or scrambled datasets.

be possible to develop experimental protocols that are better suited for network analysis than protocols currently used to generate data in typical random event generator experiments.

For example, consider an experiment requiring an operator to attempt to simultaneously influence multiple RNGs. The multiple, parallel outputs of

TABLE 3
Means, standard deviations, and t-tests for transfer, random, and scrambled distributions; N = 50 training-transfer repetitions of a 20 hidden node network, trained for 2,000 passes; this network was presented with data from three independent series for each of nine operators, for a total of 27 lines of input;" significant tests are emphasized in bold

	5/5 Correct Matches
Transfer	
Mean	2.42
SD	1.09
Random	
Mean	1.24
SD	1.21
Scrambled	
Mean	1.64
SD	1.12
t(T-R)	**7.62**
t(T-S)	**5.13**
t(R-S)	−2.56

the RNGs would be fed into a network for analysis. With this form of parallel data, plus additional parameters such as direction of intention (e.g., high aim or low aim), feedback presentation style, age, gender, environmental variables, and so on, a network may be able to learn to associate operator identity with his or her data at a transfer rate approaching practical utility. Trial-by-trial parallel feedback displayed to operators could be more interesting than the typical display of numbers of hits, or linear graphs, and instead provide graphical representations of multidimensional phase relationships among the outputs of the RNGs.

Such displays would have the advantage of conveying more information to operators at a single glance, would allow more interesting forms of feedback (e.g., dynamically shifting coherent vs. chaotic shapes and colors), and would allow experimental protocols in which shift of a theoretical mean of a distribution would be secondary as compared to the relationships of means of independent binomial distributions. It may be that phase relationships are "easier" to influence than mean shifts because changes in phase do not require pushing a device against its normal operating probabilities, that is, different phase relationships among the outputs of independent random generators can be achieved without shifting the individual distribution means beyond chance expectation.

Endnotes

[1] For an introduction to the capabilities and theories of neural networks, see the special issue of IEEE ***Computer,*** 21, 1988.

[2] I am indebted to Robert Jahn, Roger Nelson, and Brenda Dunne for providing a copy of this dataset, which was retrieved from computer tape archives maintained by the PEAR laboratory.

[3] Note that under the null hypothesis, it should not matter when or by whom the data was collected, so the present method would offer no advantage in identifying "signatures."

[4] In other words, $V = (\sum Z^2/N)$, where $N = 25$.

[5] I will allow the interested reader to figure out why this is so, with the hint that successive operator codes in the training set are sequential binary numbers.

[6] The program allows a wide variety of parameters to be set, including variables related to how fast the network learns, how many nodes there are, the range for initial random weights used to interconnect the nodes, and so on. I am indebted to Stephen Hanson and Robert Masterson for their gracious assistance with this software. See the Appendix for the activation and learning rule equations.

[7] The back-propagation method used here is an extension of the well-known generalized delta rule (Rumelhart, Hinton, & Williams, 1986), allowing the use of non-euclidian error signals. Hanson & Burr (1987) have shown that this method is particularly effective with noisy data.

[8] In this network, output activation levels were continuous-valued numbers between 0 and 1.

[9] The variance in the transfer dataset comes from the randomness in the neural network's initial state, whereas the variance in the random and scrambled datasets come from two sources: randomness in the network and randomness in the pseudorandomly generated data. Thus, in statistical terms, the t-test is conditional on the transfer dataset.

[10] This computer runs at about 12.5 million instructions per second (MIPS), which suggests why neural network analyses have only recently become practical.

[11] It should be noted that the number of training passes, hidden nodes, training-transfer repetitions, and so on, were chosen by experience with these networks rather than by algorithmic or formal criteria. This is because the state-of-the-art of neural networks remains more of an art than a science.

References

Babu, S. (1987). Analysis of variance of REG data. In D. Weiner & R. D. Nelson (Eds.), *Research in parapsychology1986* (pp. 1–5). Metuchen, NJ: Scarecrow Press.

Barlow, D. H., & Hersen, M. (1984). *Single case experimental designs: Strategies for studying behavior change* (2nd ed.). New York: Pergamon Press.

Berger, R. E. (1988). In search of "psychic signatures" in random data. In D. Weiner & R. Morris (Eds.), *Research in parapsychology 1987* (pp. 81–85). Metuchen, NJ: Scarecrow Press.

Hanson, S. J., & Burr, D. J. (1987). Minkowski-R back-propagation: Learning in connectionist models with non-euclidian error signals. To appear in *Proceedings of First IEEE Conference on Neural Networks.*

Jahn, R. G., & Dunne, B. J. (1986). On the quantum mechanics of consciousness, with application to anomalous phenomena. *Foundations of Physics, 16,* 721–772.

Jahn, R. G., & Dunne, B. J. (1987). *Margins of reality.* New York: Harcourt Brace Jovanovich.

Jahn, R. G., Dunne, B. J., & Nelson, R. D. (1987). Engineering anomalies research. *Journal of Scientific Exploration, 1,* 21–50.

Jones, W. P., & Hoskins, J. (1987, October). Back-propagation: A generalized delta learning rule. *Byte,* pp. 155–162.

Materna, T. (1987, June). Neural networks enter high speed marketplace. *Computer Technology Review,* pp. 1–30.

McConnell, R. A. (1989). Psychokinetic data structure. In L. Henckle & R. Berger (Eds.), *Research in parapsychology1988* (pp. 16–19). Metuchen, NJ: Scarecrow Press.

Nelson, R. D., Dunne, B. J., & Jahn, R. G. (1984). *An REG experiment with large data base capability, III: Operator related anomalies* (Technical Note PEAR 84003). Princeton Engineering Anomalies Research, Princeton University, School of Engineering/Applied Science.

Nelson, R. D., Dunne, B. J., & Jahn, R. G. (1986). Operator-related anomalies in physical systems and information processes. *Journal of the Society for Psychical Research, 53,* 261–286.

Radin, D. I. (1985). Pseudorandom number generators in psi research. *Journal of Parapsychology, 49,* 303–328.

Radin, D. I., & Nelson, R. D. (1987). *Replication in random event generator experiments: A meta-analysis and quality assessment* (Technical Report HIP 87001). Human Information Processing Group, Department of Psychology, Princeton University.

Radin, D. I., & Nelson, R. D. (in press). Evidence for consciousness-related anomalies in random physical systems. *Foundations of Physics.*

Roberts, C. S. (1982, January 20). Implementing and testing new versions of a good 48-bit pseudo-random number generator (Bell Laboratories Technical Memorandum, publication no. TM-82-11353-1).

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel distributed*

processing, Explorations in the microstructures of cognition, Volume I: *Foundations* (pp. 318–362). Cambridge, MA: MIT Press.
Shriver, B. D. (1988). Artificial neural systems. *Computer, 21*(3), 8–10.
Weisburd, S. (1988). Fingerprinting DNA from a single hair. *Science News,* 133, p. 262.
Widrow, B., & Winter, R. (1988). Neural nets for adaptive filtering and adaptive pattern recognition. *Computer, 21*(3), 25–40.

Appendix

This semilinear function used to calculate activation in the forward pass is from Rumelhart, Hinton and Williams (1986, p. 329):

$$o_{(pj)} = \frac{1}{1 + e^{-\Sigma_i w_{(ji)} o_{(pi)}}},$$

where

o is the output activation,
p is a pointer referring to the hidden or output layer of nodes,
w are the interconnection weights from input to hidden, or hidden to output nodes,
i ranges over the number of input or hidden nodes, and
j ranges over the number of hidden or output nodes.

For the backwards pass, the back-propagation error **(A)** is computed for output nodes as:

$$\Delta_j = \mathrm{SGN}\,|t_j - o_j|^{(r-1)} * o_j * (1 - o_j),$$

where

SGN is the sign of $(t_j - o_j)$,
t is the teacher signal (the desired output),
o is the observed activation level at the output node,
and *r* is a real number (chosen as $r = 3$ for all network simulations reported here).

Now A for hidden nodes is calculated as:

$$\mathrm{A}, = \sum_{k=1}^{\mathrm{Nout}} \Delta k * w_{(jk)} * h_j * (1 - \mathrm{h},),$$

where

Nout is the number of output nodes,
Δ_k is the activation on an output node,
w is the interconnection weights between hidden and output nodes, and
h is the activation on hidden node **j.**

We then modify the weights in the network according to the rules:

$$\Delta_{ji}(t + 1) = \alpha * \Delta ji(t) + \epsilon * \Delta j * \mathrm{activation}_i,$$

and

$$w_{ji} = w_{ji} + \Delta_{ji},$$

where

α controls the "inertia" of learning in the network,
ϵ controls how fast the network learns,
t is the time or processing cycle number, and
activation_i is the amount of activation on node i.

Note that weights are updated after each presentation of an input/output pair, and the amount of change depends on how much the weights were changed on the previous cycle (and the value of α).