

Neural Network Analyses of Consciousness-Related Patterns in Random Sequences

DEAN I. RADIN

Department of Psychology, University of Edinburgh, Scotland

Abstract — Researchers investigating the effects of mental intention on the output of random number generators have observed person-unique patterns or "signatures" impressed into the data. A previously reported study used an artificial neural network to analyze the data produced in these experiments and found evidence supporting the signatures hypothesis. The present study again used a neural network to search for patterns, this time using new data and new network configurations. Results of eight analyses confirmed the presence of person-specific signatures. Suggestions for creating practical applications from this phenomenon are outlined.

Introduction

For nearly four decades, more than seventy researchers have investigated hypothesized correlations between mental intention and the statistical output of electronic random number generators (RNG) (Radin & Nelson, 1989). Some of these researchers have claimed that the data reveals person-specific patterns, occasionally referred to as "signatures" (Berger, 1988; McConnell, 1989; Nelson, Dunne & Jahn, 1986).

These signatures may be described as unique graphical shapes that appear in summaries of individual datasets. For example, Figure 1 illustrates signatures for three hypothetical people.

In an earlier paper (Radin, 1989), I described the use of an artificial neural network to examine the signatures hypothesis using data collected in RNG experiments at the Princeton Engineering Anomalies Research (PEAR) Laboratory at Princeton University (e.g., Dunne & Jahn, 1992). Results of that study suggested that an artificial neural network could be trained to associate patterns in RNG data with the identity of individuals who produced the data. It was also shown that a network trained on repeated independent examples of the same individuals' efforts improved the network's ability to correctly identify the patterns.

In this paper, I describe the results of new neural network analyses designed to replicate and elaborate upon the previous findings, using additional data provided by the PEAR laboratory.' In particular, I attempted to

- (a) confirm the previous finding that person-specific signatures can be identified by a neural network;



Fig. 1. Illustration of the signatures idea for three hypothetical people. In a typical experiment involving RNGs, individuals are asked to mentally "cause" the output of an RNG to randomly walk *above* chance expectation (indicated by the "+"), to randomly walk *below* chance expectation ("-"), or to randomly walk around chance as a control ("c"). These graphs plot cumulative deviations (hence the Δ symbol) from chance expectation for each of three participants. The signatures claim is based upon the observation that individuals' cumulative data seem to show different characteristic shapes and that these shapes apparently persist over repeated experimental trials and different RNG devices.

- (b) explore whether a network could learn to associate RNG data with individual identities, the type of mental intention the individuals employed, the type of RNG used in the experiment, and then successfully transfer the learned associations to new data; and
- (c) explore whether a network could learn the associations described above, but under an experimental protocol where individuals generated the data with a single button press.

Neural Networks

This section provides a quick overview of artificial neural networks for readers not familiar with this technology. Refer to Radin (1989) for more details on neural network computational techniques as used in the present analysis.

Artificial neural networks encompass a class of relatively new computational techniques and models, variously called neural networks, parallel distributed processing, and connectionism. The technique has attracted broad interest within artificial intelligence, the cognitive sciences, and the neurosciences because neural networks (a) are advancing the theory and development of self-organizing, adaptive systems, (b) are providing solutions to previously difficult problems in pattern recognition and image analysis, and (c) are providing effective, computational ways of studying biological neural networks (Shriver, 1988).

Neural networks are a form of parallel processing roughly 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, they can automatically learn to associate arbitrarily complex inputs with arbitrarily complex outputs. Properly configured, these networks can also implement desirable features such as self-organizing associative memories, automatically derived statistical descriptions of

spatial and spatio-temporal data, and autonomously acquired knowledge by observation.

Information processing in a neural network involves interactions among large numbers of artificial "neurons." These neurons, also called nodes or units, have four main components, namely: input connections, through which the unit receives activation from other units, a summation function that combines various input activations into a single activation, an 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).

Neural networks typically consist of several layers of nodes. All networks used in the present analysis had three layers: input, middle, and output. The middle layer is sometimes called the hidden layer because it is not "seen" directly by the external world. That is, the input and output nodes form the explicit association problem to be solved, whereas the hidden nodes and the rich set of interconnections among the nodes are used to increase the computational space of the network and to allow complex, nonlinear, nonparametric relationships to be learned (Ripley, 1992).

Method

The Data

Data from four individuals were used in the present analysis (Nelson & Dobyns, 1991), provided in the form of trial scores (the number of times a random bit matched an alternating target bit in a sequence of 200 random bits; mean expectation = 100). One experiment was called a series, consisting of 1000 trials in each of three mental "aim" conditions: aim for high trial scores (i.e., numbers greater than 100), for low scores (numbers less than 100), and a "no aim" baseline condition.

In the PEAR Laboratory protocol, the "manual mode" of operation generates one trial at each press of a button. All of the present series were generated using various types of "automatic mode," in which runs of 50 trials, 100 trials, or 1000 trials were automatically generated with a single button press. Thus, each series of 1000 trials in the present database was produced with 10, 20, or a single button press per aim condition (depending on which run-length was selected). The database consisted of a grand total of 65 series generated with 50 or 100 trials per aim condition, and 82 series generated with 1000 trials per aim condition.

Besides the raw trial scores, the database also included information about the identities of the persons who produced the data (identity arbitrarily coded as 1, 2, 3 and 4), intentional aim (high, low, baseline), type of RNG (pseudo-random algorithm or truly random), mode of instruction ("volitional," in which participants chose the direction of intentional aim themselves, or "instructed," in which participants were randomly assigned the direction of aim), and several other parameters which were not used in the present analysis.

Data Preparation

The previous neural network analysis (Radin, 1989) employed a simple way of summarizing an individual's performance over one series: a measure of the overall shift of the mean from chance expectation per aim condition, calculated by a *t* score, and a test comparing the observed standard deviation per aim condition with the chance expected standard deviation, calculated by a chi-squared statistic and then transformed into a normal *Z* score. Thus, the raw data in each of three aim conditions was transformed into two normalized statistics, for a total of six summary statistics per series, per person. Note that the precise forms of the summary statistics do not matter provided that the method of determining them is consistently applied to all data.

To evaluate whether an artificial neural network has learned to identify a person based upon his or her data, two datasets are needed for each individual: one is used to *train* the neural network (called the training dataset) and the other is used to examine the extent to which the trained network has *transferred* (called the transfer dataset) its learned knowledge to new data. In the previous study (Radin, 1989), each series was split in half due to limited data, using the first half of the data as the training set and the second half as the transfer set.

In the present analyses, because the database provided sufficient numbers of repeated series per person, the training and transfer datasets consisted of summary statistics from completely independent series. This has the disadvantage of losing potential psychological, physical, and environmental similarities in a given individual's performance over a short period of time, but it also provides an advantage: If the signatures hypothesis is genuine, and appears *across* datasets, it would help support the notion that signatures are more than short-term drifts in the RNG equipment, or some other short-lived methodological artifacts that are somehow related to specific individuals.

Control Datasets

The previous study indicated that because of the high noise inherent in data produced by random number generators, to adequately judge the extent of knowledge transferred we need to compare transfer test results using actual series data against some form of control dataset. In the first analysis described below, two control datasets were employed: (a) a *random* dataset generated by computer-simulating the PEAR experiment without any human intervention using a pseudorandom number generator (PRNG),² and (b) a *scrambled* dataset generated by using the original PEAR data, but where the subject identity codes have been scrambled. In the remaining analyses reported here, only random datasets were used for controls (for reasons that are described later).

Implementation Software and Hardware

A three-layer neural network using the conventional backpropagation learning scheme was used in these analyses (Jones & Hoskins, 1987; Rumel-

hart, Hinton & Williams, 1986). The program was written by the author in the C language and a Silicon Graphics IRIS 4DTM workstation was used to run the parallel processing simulations.³

Results of Analyses 1 - 4

Analysis 1

The primary question of interest was whether a network could learn to associate data produced by an RNG with the person who was attempting to influence that RNG, then recognize that person on the basis of new data the network had not "seen" before.

This network consisted of six input nodes, four output nodes, and twelve hidden nodes.⁴ The network "learning rate" was set to 0.2,⁵ and the network was trained for 750 passes.⁶ To avoid potentially biasing the network during the training process, the *order of the rows* in the training set was effectively randomized by numerically sorting the dataset according to the first data item in each row.⁷

The neural network data format was as follows:

----- INPUT ----- ----- OUTPUT -----						
<i>t</i> hi	Z hi	<i>t</i> lo	Z lo	<i>t</i> base	Z base	ID
-1.994	-0.803	0.884	-0.848	0.563	-1.116	0 1 0 0

where *t* and *Z* are series summary statistics as explained above, "hi", "lo", and "base" correspond to the three directions of intentional aim,⁸ and "ID" corresponds to an arbitrary but unique code used to identify the person who produced the data. Person one was simply coded as "1 0 0 0," person two as "0 1 0 0," and so on. The first six data items are inputs to the network and the ID code is the output that is to be learned.

The values at the network's output nodes represent network activation levels, which normally range continuously between 0 and 1. However, because the output ID codes in this context were defined as exactly 0 or exactly 1, a simple transform was used whereby network output activation values ≥ 0.5 were recoded as 1 and activation values < 0.5 were recoded as 0.

To maintain consistency with previous analyses (Radin, 1989), only PEAR series consisting of 50 or 100-trial runs were used in this first analysis (i.e., 1000-trial series were excluded). There were a total of 65 such series available. For simplicity, a training set of 40 series was created, composed of 10 series per person (i.e., 4 people x 10 series each), and random and scrambled datasets consisting of 15 series, composed of 5 series for each of three people. (That is, three people because one person in the database only had data for 10 series, thus no additional series were available to be added to the transfer dataset.)

Note that an unbalanced number of series per person in the transfer or scrambled datasets would not affect the transfer test results because the network had

already learned its associations from a completely balanced training dataset. If the training dataset were biased, such that it represented greater or fewer numbers of series for some people, the network would indeed capture that bias and any transfer test results based on what the network had learned would reflect that bias. But, since the training sets reported here were all symmetrically balanced (according to ID code, aim, and the other parameters), no such bias was permitted or possible in the transfer, random, or scrambled tests."

For the random dataset, new data were generated by simulating an entire PEAR experiment using a well-tested PRNG.¹⁰ After the experimental data was simulated, *t* and *Z* scores were calculated in the usual way, and a new dataset was created (i.e., the random dataset) which was identical to the transfer dataset in form, except that all data was produced pseudorandomly and without any form of human intervention.

For the scrambled dataset, ID codes were shifted right two positions, such that person 1's code was shifted from "1 0 0 0" to "0 0 1 0," person 2 shifted to "0 0 0 1," and so on. In this way, a control dataset was formed using the identical data from the transfer dataset, but the identities of the people who produced the data were scrambled.

At this point I must make a slight diversion into the mechanics of neural networks. When a neural network is trained, it goes through a process roughly analogous to the process of casting a metal object. Metal is heated until it turns into a liquid, it is poured into a mold, and it is allowed to cool. Ultimately, the metal permanently takes on the shape of the mold. Heating is like the initial disorganized or randomized state in a neural network, cooling represents the organizing or training process, and the mold represents the nature of the problem that is cast into the network.

When metal cools into a mold, it does so wholistically, with all molecules of the metal settling into the mold simultaneously. The metal does not settle into its proper place sequentially, one molecule at a time. The process of training a neural network is similarly wholistic and non-algorithmic, and as a result the eventual state that a network "settles into," mathematically-speaking, is a *priori* unknown. This is because a neural network (like a liquid metal) provides a large computational space (many configurations of molecules) in which to solve its association problem (the final cast can be achieved by many different arrangements of molecules), and the final solution to the problem depends on the characteristics of the problem (the cast) as well as upon the initial conditions of the network (the heat of the metal). The initial state of these networks is intentionally randomized (i.e., the initial weights on the interconnecting links between nodes are random numbers), thus although the learning process is fully determined by the back-propagation learning equations, the final learned state for a given input-output problem is undetermined.

The fact that there can be many solutions to the input-output problem suggests that it may be difficult to know when (or indeed, if) a network has achieved an optimal solution to any given problem. If we were to train a network once, then judge the amount of information learned in a transfer test

TABLE 1
Transfer test results using actual data, random data, and scrambled data.

	Transfer to	mean	sd	N	series	%	t-test (df)
Analysis 1	Actual data	3.82	1.63	100	15	25.5	
	Random data	1.63	1.29	100	15	10.9	10.48 (99)
	Scrambled data	1.13	1.05	100	15	7.5	13.80 (99)

Mean and *sd* are the average number and standard deviation of correctly identified outputs out of the total number of *series* in the test dataset, after N repetitions of the training-transfer process. Percent (%) indicates the mean percentage of correctly identified items in the transfer dataset. *T-tests* compare the difference between means of actual vs. random and actual vs. scrambled datasets.

using actual, and then randomized data, we would not know if those results were characteristic of the "true" performance of the network. Thus, to help estimate this generalized performance, the training process was repeated from scratch 100 times in this analysis (and 50 times in succeeding analyses), each time using newly randomized initial conditions. The means of the distributions of transfer test results using actual and control data were then compared to see how well the network had performed.

Results, shown in Table 1, suggest that the network learned to associate performance "signatures" with individual identities, confirming the findings reported in Radin (1989). The table shows that when testing the network with actual data (this was called the "transfer" test), the mean number of correctly identified ID's was 3.82 out of 15 possible, or 25.5% correct on average. The same measure with random data was 10.9%, and with scrambled data, 7.5%. A t-test of the differences among these means shows that the network identified significantly more IDs using actual data than control data. (A more complete discussion is deferred to the Discussion section after Analysis 4.)

Analysis 2

This analysis examined whether a network could learn to associate data from a single intentional aim and the identity of the person who produced the data, with the direction of intentional aim plus the type of RNG that produced that data. The data format was as follows:

----- INPUT -----		OUTPUT -----		
t	Z	ID code	Aim	RNG type
-2.372	0.006	0 0 0 1	0 1 0	0

where t, Z and the ID code are as described above, "Aim" corresponds to the direction of mental aim (high = "1 0 0", low = "0 0 1", and control = "0 1 0"), and "RNG type" corresponds to the type of random number generator used to produced the data (pseudorandom = "0", truly random = "1").

TABLE 2
Transfer test results for Analysis 2 using actual and random data.

	Transfer to	mean	sd	N	series	%	t-test (df)
Analysis 2	Actual data	9.44	2.74	50	48	16.6	
	Random data	6.68	2.44	50	48	11.7	5.27 (49)

The training dataset consisted of 72 such data items, corresponding to 24 series, six from each of four people (i.e., 24 series at 3 aims per series = 72), and the transfer and random datasets consisted of 48 new data items, corresponding to 16 series of data, four from each of four people. The network learning rate was set at 0.1, with 15 hidden nodes, and training continued for 3500 passes. A scrambled dataset was not employed in this or in succeeding analyses because all previous transfer tests using control data had shown no significant differences between random and scrambled datasets.

Results, shown in Table 2, indicate that the network was able to learn to associate individuals' RNG performance and identity with their intentional aim and the type of RNG they used. (Again, further discussion is postponed until after the description of Analysis 4.)

Analysis 3

The purpose of this analysis was identical to that of Analysis 2, except that *only* data produced in 1000-trial series were employed. A total of 54 such series were used in the training dataset and 48 series used in the transfer and random datasets. The network learning rate was set at 0.2, with 15 hidden nodes, and training continued for 2500 passes. The data format was identical to that used in Analysis 2:

----- INPUT -----	----- OUTPUT -----			
<i>t</i>	<i>Z</i>	ID code	Aim	RNG type
-1.899	-0.221	0 0 1 0	0 1 0	1

Results in Table 3 show that the network was able to learn to associate individuals' RNG performance and identity with their intentional aim and the type of RNG they used.

TABLE 3
Transfer test results for Analysis 3 using actual and random data.

	Transfer to	mean	sd	N	series	%	t-test (df)
Analysis 3	Actual data	8.02	2.06	50	48	16.7	
	Random data	6.12	2.34	50	48	12.8	4.27 (49)

Analysis 4

This analysis examined the possibility that if a network were trained with additional information beyond input data and the ID code, it may further enhance a network's ability to identify the type of intentional aim used to produce that data. To test this hypothesis, a data format with eight inputs and three outputs was used, as follows:

INPUT						OUTPUT	
<i>t</i>	<i>Z</i>	ID code	RNG type	Instruction			
1.936	-0.944	00 10	1	0	001		

where the additional input items are "RNG type," as described previously, and a new item called "Instruction." Instruction is either volitional (value = 0), which is when the person gets to select if they want to aim high, low, or baseline on a given button press, or instructed (value = 1), which is when an RNG is used to assign the direction of aim for the subject.

A total of 72 series were used in the training set (data excluding 1000-trial series) and 117 series in the transfer and random datasets. The network learning rate was set at 0.2, with 10 hidden nodes, and training continued for 2000 passes. Results in Table 4 indicate that the network was able to learn this association, and the actual vs. random t-score of 5.62 was larger, but not significantly so, than the similar actual vs. random t-score of 4.27 obtained in Analysis 3.

Discussion

The results shown in Tables 1 - 4 indicate that the data in the PEAR RNG experiments show forms of internal structure that are unexpected by chance and are consistent with the hypothesis that person-unique patterns are somehow impressed into the data. The differences between means of correctly identified outputs given actual data vs. control data (both random and scrambled) are statistically unambiguous. Even the absolute magnitude of some of the differences are non-trivial (e.g., 10.9% knowledge transferred with random data vs. 25.5% knowledge transferred with actual data, as seen in Analysis 1). But what does this all mean?

In discussing the results of the above four analyses with colleagues," three questions were repeatedly raised: First, What do the percentage transfer rates in Tables 1 - 4 actually mean? In Analysis 1, for example, shouldn't one expect

TABLE 4
Transfer test results for Analysis 4 using actual and random data.

	Transfer to	mean	sd	N	series	%	t-test (df)
Analysis 4	Actual data	28.16	4.89	50	117	24.1	
	Random data	22.92	4.32	50	117	19.6	5.62 (49)

by chance to see a 25% transfer rate since there are only four possible output nodes in that network? Likewise, shouldn't one also expect 25% transfer rates by chance in Analyses 2 and 3, and a 33% transfer rate by chance in Analysis 4? Why do none of the transfer results using random data show these expected percentages? These questions are addressed in Analyses 5 and 6, below.

The second question asked what the network was learning that allowed it to discriminate between actual and random data, and how can we begin to understand what was learned given that the information is captured in a computational space of many more than three dimensions (perhaps ten to fifteen "dimensions," depending on the number of hidden nodes used in the network). Analysis 7 addressed this question.

The third question was, Does the network learn to identify only unusual data items, such as data due to one "outlier" individual, or does it learn something about the entire body of data, somehow capturing the gestalt of the dataset?

Results of Analysis 5 - 8

Analysis 5

As a reminder, the data presented to the neural network in Analysis 1 used the following format:

INPUT		OUTPUT														
<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 0 10px;">thi</td> <td style="padding: 0 10px;">Zhi</td> <td style="padding: 0 10px;">tlo</td> <td style="padding: 0 10px;">Zlo</td> <td style="padding: 0 10px;">t base</td> <td style="padding: 0 10px;">Zbase</td> <td style="padding: 0 10px;">ID</td> </tr> <tr> <td style="padding: 0 10px;">-1.994</td> <td style="padding: 0 10px;">-0.803</td> <td style="padding: 0 10px;">0.884</td> <td style="padding: 0 10px;">-0.848</td> <td style="padding: 0 10px;">0.563</td> <td style="padding: 0 10px;">-1.116</td> <td style="padding: 0 10px;">0 1 0 0</td> </tr> </table>	thi	Zhi	tlo	Zlo	t base	Zbase	ID	-1.994	-0.803	0.884	-0.848	0.563	-1.116	0 1 0 0		
thi	Zhi	tlo	Zlo	t base	Zbase	ID										
-1.994	-0.803	0.884	-0.848	0.563	-1.116	0 1 0 0										

Thus, if random (i.e., unpatterned) data were input, it seems reasonable that we should expect the observed output to correctly match the desired output only 1 in 4 times or 25%. Yet Table 1 shows that the actual data transferred at 25% and random data transferred at only 11%, implying that either the network did not learn anything and somehow the random dataset gained "negative knowledge."

The answer to this apparent problem is actually quite simple. As mentioned above, the continuous-valued activation levels at the network's output nodes were explicitly transformed into either a 0 or a 1 bit by coding activation output values ≥ 0.5 as 1 and activation output values < 0.5 as 0. This means that each output node was coded as either exactly 0 or exactly 1. Since the network had four output nodes, it produced 16 possible output codes (0000, 0001, 0010, etc.) of which only 4 were valid ID codes (i.e., 0001, 0010, 0100, and 1000). We would therefore expect the chance transfer rate for random data to be about 1 in 16. However, this is not precisely the case because during the training process the network is only presented with 4 valid ID codes as possible outputs, so it becomes biased to produce only these four outputs.

The result is that the true chance transfer rate lies somewhere in the range of 1 in 16 to 1 in 4, or 6.25% to 25%. Unfortunately, the theoretically expected

chance transfer rate is unknown. To further compound the problem, if the network really does learn something that can be transferred to actual data, then the scrambled dataset should result in a transfer rate that is *less* than that observed with random data. This is because the data in the "scrambled" dataset are the same as in the actual dataset, but the ID codes are explicitly mismatched by being shifted. Because we postulate that actual data tend to produce correct ID codes as output, if we attempt to match correct IDs with scrambled IDs, we should explicitly produce too many *mismatches*. Table 1 shows that the transfer rates do indeed fall in the expected rank order in Analysis 1, with transfer percentages in the order: actual > random > scrambled. Unfortunately, interpreting what the magnitudes of these percentages mean is problematic.

To help understand whether these results are in line with what one would expect by chance, I changed the output coding transform to define the one node containing the *maximum* output activation as 1, and the other three nodes as 0.¹² For the network used in Analysis 1, there are only four possible outputs, thus the random expected transfer rate using the maximum coding scheme is exactly 1 in 4, or 25%. The scrambled transfer rate should be less than 25%, and the actual transfer rate should be greater than 25%. The same network parameters were used as previously described in Analysis 1.

Table 5 shows the results, which conform to the above expectations. The 95% confidence interval for the random transfer rate includes the expected 25%, the scrambled transfer rate is well below random, and the actual transfer rate is well above random. (t-tests were deemed unnecessary because the magnitude of the differences are self-evident.)

Analysis 6

To replicate results found in Analysis 5, we combined the datasets used in Analyses 2 and 3, and tested a network with the following input/output configuration:

----- INPUT -----			----- OUTPUT -----	
<i>t</i>	<i>Z</i>	ID code	Aim	
-2.372	0.006	0 0 0 1	0	1 0

This is the same input as in Analyses 2 and 3, but the output is just three levels of aim (high, baseline, and low). The training set consisted of 126 items; the transfer and random datasets consisted of 105 items. The random transfer rate using the maximum coding scheme is expected to be exactly 1 in 3 or 33%.

Results shown in Table 6 again conform to expectation. The random transfer rate 95% confidence interval includes 33%, as expected, but the transfer rate using actual data exceeds 33%. (A scrambled dataset was not tested in Analysis 6.)

TABLE 5
Transfer test results for Analysis 5 using the maximum coding scheme.

	Transfer to	mean	sd	N	series	%	95% confidence interval
Analysis 5	Actual data	5.72	1.39	50	15	38.1	35.6 - 40.7
	Random data	3.66	1.64	50	15	24.4	21.4 - 27.4
	Scrambled data	2.20	1.41	50	15	14.7	12.1 - 17.3
	Chance	3.00				25.0	13.80 (99)

Mean is the number of correctly identified outputs, *sd* is the standard deviation of the mean, *N* is the number times the neural network simulation was repeated, *series* is the number of data items used, *percent* (%) is the percentage of data items correctly identified (i.e., mean / series), and *95% confidence* is the 95% confidence interval for the percentage.

Analysis 7

Now that there is some reason to believe that the network transfers knowledge to actual data but not to random data, the question naturally arises as to what exactly has the network learned? Unfortunately, visualizing this learning is difficult because the network has captured its knowledge in an abstract multidimensional space. In Analysis 5, for example, the network has effectively taken a 6 dimensional input, filtered it through a 10 dimensional (hidden layer) space, and transformed that into a 3 dimensional output.

To help visualize what was learned, I used a trained network from Analysis 6, fixed the ID code for each of the four individuals in turn, applied *t* and *Z* values ranging from -2 to +2 to the input nodes (in increments of 0.2), and then examined the value of the output nodes. Because the maximum coding scheme was used, there were only three possible neural network outputs: high aim, baseline, and low aim. A three-dimensional graph was thus created, where the *x* and *y* dimensions were *t* and *Z*, as described above, and the *z* dimension was the network output coded as 1 (high aim), 0 (baseline), and -1 (low aim). These three-dimensional graphs are displayed as contour maps in Figure 2.

These graphs illustrate how the network partitioned its computational space to create the equivalent of an input-output associative map. The maps also reveal that the input variance measure (the ordinates in Figure 2) helped the network discriminate among the different individuals and different intentional aims. This is interesting because the usual measure of interest in RNG experi-

TABLE 6
Transfer test results for Analysis 6.

	Transfer to	mean	sd	N	series	%	95% confidence interval
Analysis 6	Actual data	38.22	3.75	50	105	36.4	35.4 - 37.4
	Random data	35.60	5.10	50	105	33.9	32.5 - 35.3
	Chance	35.00				33.3	

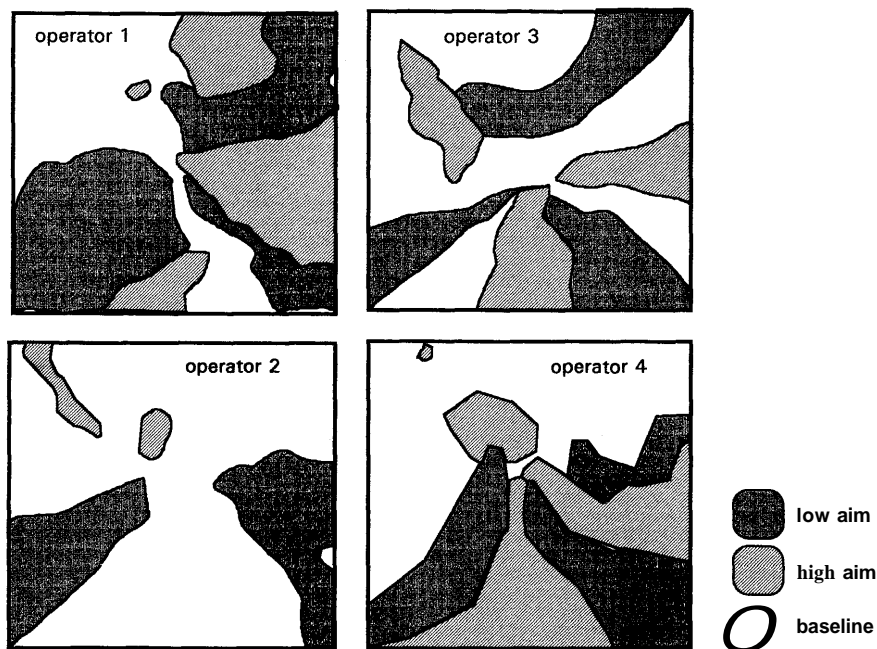


Fig. 2. Visualization showing how the neural network partitioned its computational space to solve the input/output association problem for the four operators. The x and y axes correspond to t and Z scores, as described in the test, and the three different shades correspond to the low aim, high aim, and baseline conditions.

ments is to shift the mean, not the variance. In general then, what the network has learned is represented by the associative maps shown in Figure 2. We see by these graphs that the internal representation generated by the neural network can be quite complex, even for a very simple, six-input system. Networks with many more inputs are correspondingly much more difficult to visualize.

Analysis 8

The remaining question is whether the network in fact learned individual signatures for each person in the dataset, or whether the results were perhaps due to the network focusing in on one individual's deviant data.

This was tested using the same network and parameters as in Analysis 7, except that the training dataset was comprised of 24 series (6 series for each of the four individuals), and 19 series for the transfer datasets. The chi-squared values shown in Table 3 are formed from a contingency table with three columns (high, baseline, and low aim) and four rows (four individuals). The contingency table was used to test the hypothesis that there were no significant differences among the four individuals in terms of how well the network was able to correctly identify the three outputs (i.e., high, baseline, low).

The number of correctly identified outputs in each of the three output conditions is shown in italics in Table 7. For example, in the *Training* table, the total possible number of correctly identified outputs for each aim condition was $N = 300$, calculated as $N = [6 \text{ series per aim per person}] * 50 \text{ training-transfer repetitions}$. The different number of possible outputs per person in the *Transfer* and *Random* datasets shown in Table 7 is due to the different numbers of series available to be tested in those datasets. Recall that an unbalanced number of data items per individual in the *Transfer* (or *Random*) dataset does not bias the outcome because the network is already trained before it is presented with this new data. (Obviously, if a network were changed in any way *after* being presented with new data, it would still be in the training mode.)

The overall percentage transfer rates are shown in bold in Table 7. The network was trained until the output error began to asymptote to a constant value.¹³ This occurred at about 3000 passes. At this point, the *Trained* network correctly identified an overall average of 88.2% of the input-output associations. The non-significant chi-squared value obtained with the *Training* dataset indicates that the network learned to identify aim (high, baseline, low) to approximately the same degree of accuracy (89.3%, 86.3%, 89.1%, respectively) regardless of which individual produced the data. Person 2's data, independent of the aim condition, was correctly identified most often (91.0%).

The slight bias in favor of Person 2 in the *Training* dataset is more evident in the *Transfer* results. The *Transfer* dataset chi-squared value of $\chi^2 = 60.863$ reveals that row (person) and column (aim) figures are not distributed as expected by chance. It seems that this non-chance distribution is largely due to the fact that high aim was correctly identified more often than the other two aims (39.9% vs. 35.4% overall), and Person 2's data was correctly identified more often than the other three people (45.3% vs. 35.4% overall).

The *Random* dataset is also not distributed by chance, but the overall transfer rate for this dataset (33.1%) is almost exactly that as expected by chance (i.e., 33.3%). Note that the aim condition most often identified correctly in the random dataset is the baseline condition. This makes sense because random data *should* look like a baseline aim to the neural network provided it was trained properly *and* that the baseline condition in the actual data resembled "genuinely" random data. The higher success rate in identifying the baseline condition seems to be responsible for the significant chi-square score.

In summary, Analysis 8 indicates that the *Trained* network was not significantly biased in favor of identifying any particular person or aim condition. The *Transfer* test, however, suggested that something about Person 2's performance made that person easier to correctly identify, and it was also easier to identify high and low aims, independent of who produced the data, compared to the baseline aim, as expected if high and low aims actually do introduce consistent patterns into the random data. The *Random* test showed that the baseline data was easiest to identify, as expected if the baseline data in the experiment is genuinely random.

TABLE 7

Contingency tables for correctly identified aims in Training, Transfer, and Random datasets. (See text for explanation.)

<i>Training</i>	<i>High</i>	<i>Baseline</i>	<i>Low</i>	<i>Sum</i>	<i>Possible</i>	<i>Percent</i>
Person 1	274	251	254	779	900	86.6
Person 2	280	271	268	819	900	91.0
Person 3	262	276	275	813	900	90.3
Person 4	255	238	272	765	900	85.0
Sum	1071	1036	1069	3176	3600	88.2
Possible	1200	1200	1200			
Percent	89.3	86.3	89.1			
χ^2 (6 df)	3.474	p = .75				
<i>Transfer</i>	<i>High</i>	<i>Baseline</i>	<i>Low</i>	<i>Sum</i>	<i>Possible</i>	<i>Percent</i>
Person 1	64	80	65	209	600	34.8
Person 2	165	124	119	408	900	45.3
Person 3	29	56	48	133	450	29.6
Person 4	121	34	103	258	900	28.7
Sum	379	294	335	1008	2850	35.4
Possible	950	950	950			
Percent	39.9	30.9	35.3			
χ^2 (6 df)	60.863	p < .0001				
<i>Random</i>	<i>High</i>	<i>Baseline</i>	<i>Low</i>	<i>Sum</i>	<i>Possible</i>	<i>Percent</i>
Person 1	35	80	75	190	600	31.7
Person 2	54	152	89	295	900	32.8
Person 3	55	68	43	166	450	36.9
Person 4	111	85	97	293	900	32.6
Sum	255	385	304	944	2850	33.1
Possible	950	950	950			
Percent	26.8	40.5	32.0			
χ^2 (6 df)	52.598	p < .0001				

Discussion

At this point, some readers may wonder why these results are important. It is essential to realize that the effect under consideration here is something that current theory cannot adequately explain: a person-specific correlation between human conscious intention and the output of a machine that is not physically connected to the experimental participants in any (known) way. The present analyses show that the interaction of intention with an RNG produces person-specific patterns in what should be (under the null hypothesis) purely random data.

Summary

The results of the above analyses may be summarized as follows:

- (a) Repeated presentation of RNG data produced by individuals mentally interacting with the RNG apparently improves a network's ability to identify these same individuals when the same trained network is presented with new data. In an earlier paper (Radin, 1989) I reported that presenting a network with a *single* half-series per person produced a 1.1% transfer rate above chance (i.e., 4.6% using actual data minus 3.5% using random data). In that same paper I showed that presenting *three* half-series from each of nine people produced a 4.3% transfer rate. The present study showed that presentation of *ten* full series from each of four individuals produced a 14.6% transfer rate;
- (b) A network can apparently learn to associate individuals' performance with the type of mental aim they employed and the type of RNG used to generate the data;
- (c) A network can learn the association described above, even under an experimental protocol where an entire series is generated with a single button press per aim condition;
- (d) A network's ability to determine intentional aim might be further enhanced by presenting multiple sources of information (i.e., many variables, not just repeated examples of the same variable) about an individual's performance. For example, compare the percentage of transfer obtained when using actual data in Analysis 4 vs. Analyses 2 and 3: 24.1% vs. 16.6% vs. 16.7%, respectively;
- (e) After a successful training process, a network is not necessarily biased to identify one person's data over another, but when a trained network is used to identify patterns in new data, some peoples' data and some mental aim conditions may be easier to identify than others.

Implications

A methodology which can detect person-specific mental intentions in data generated by a remote machine suggests the possibility of developing an array of novel human-machine interaction applications. An example of one applica-

tion might be called a mental lock, or mock. Mock would be an RNG – neural network system, which, upon sensing the presence of a person (using, say, conventional proximity detection circuitry), would initiate the generation of a sequence of random bits, then match the resulting data pattern against previously learned patterns. If a suitable match is found, the mental lock has recognized the identity of the proximal individual on the basis of his or her unique mental patterns which had been "impressed onto" the RNG data stream, and the mental lock would open. It may be that patterns associated with human consciousness are as unique, and therefore as useful for identification or security purposes, as retinal patterns and fingerprints.

Another application might be called a mind-controlled robot, or robomind. This would consist of a two-stage neural network system where the output of the first network was robotic control signals, and the input to that network was the output of a second network which was designed to detect mental influences on RNGs. It might be possible to train a two-stage (or perhaps a multi-stage) network to perform useful work under a combination of conventional and remote intentional control. One obvious use for robomind would be deep-space or deep-ocean exploration, where some degree of control over a remote, autonomous robot may be desired, but where it may be difficult or impossible to systematically or reliably employ normal telemetric control signals because of electromagnetic interference, attenuation of the signal, or possibly even relativistically-prohibitive distances (assuming that the mental intention effect is space-time independent). A less exotic use might be a robomind-wheelchair that a quadriplegic could control by intentional thought.

Conclusion

This study confirms previously reported results suggesting that an artificial neural network can learn to associate machine-generated random data with individuals who somehow "mentally impress" patterns onto that data. The present study also found that properly configured artificial neural networks can learn to associate data with specific mental intentions, demonstrating the feasibility of developing a new form of novel human-machine interaction technologies.

References

- Berger, R. E. (1988). In search of "psychic signatures" in random data. *Research in Parapsychology* 1987, Metuchen, NJ: Scarecrow Press.
- Dunne, B. J. & Jahn, R. G. (1992). Experiments in remote human/machine interaction. *Journal of Scientific Exploration*, 6, 311-332.
- Jones, W. P. & Hoskins, J. (1987, October). Back-propagation: A generalized delta learning rule. *Byte*, 155-162.
- McConnell, R. A. (1989). Psychokinetic data structure. *Research in Parapsychology* 1988, Metuchen, NJ: Scarecrow Press.
- Nelson, R. D. & Dobyns, Y. H. (1991). Analysis of variance of random event generator experiments: Operator intention, secondary parameters, database structure. (Technical Note PEAR 91004). 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. (1989). Identifying "signatures" in anomalous human-machine interaction data with an artificial neural network. *Journal of Scientific Exploration*, 3, 185-200.
- Radin, D. I. & Nelson, R. D. (1989). Evidence for consciousness-related anomalies in random physical systems. *Foundations of Physics*, 19, 1499-1514.
- Ripley, B. D. (1992). Statistical aspects of neural networks. *Proceedings of SemStat (Seminaire Européen de Statistique)*, Sandbjerg, Denmark, 25-30 April 1992.
- 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 and J. L. McClelland (Eds.), *Parallel distributed processing, Explorations in the microstructures of cognition, Volume 1: Foundations*. Cambridge, MA: MIT Press, pp. 318-362.
- Shriver, B. D. (1988). Artificial neural systems. *Computer*, 21, 3, 8-10.

Footnotes

¹ I thank Robert Jahn, Brenda Dunne, and Roger Nelson for providing a copy of this database.

² A standard UNIX System V™ 48-bit multiplicative congruential PRNG called drand48 was used to provide pseudorandom numbers in this study. This generator has passed extensive first- and higher-order randomness tests (Radin, 1985; Roberts, 1982).

³ I am indebted to Joseph Lubin for his technical assistance with these simulations and for many valuable discussions.

⁴ The number of hidden nodes in this (and succeeding analyses) was empirically determined to provide the best associative fit to the data. In general, the smaller the number of hidden nodes, the better the network is able to generalize what it learns.

⁵ The network learning rate establishes a tradeoff between how quickly the network learns vs. the precision with which it learns. Set too high, the network may initially learn quickly, but then never converge on an acceptable answer to the input/output problem. Set too low, it may take a very long time to converge on an answer. The learning rate typically ranges from 0 to 1; it was determined in this and the succeeding analyses empirically rather than by algorithmic means.

⁶ One "pass" is one training sweep through the network, during which the associative links among the nodes are slightly adjusted to correct for the differences between the obtained and the desired input/output associations. Networks must be trained with hundreds or thousands of such passes before the difference or error between desired and observed associations are acceptably low.

⁷ The original data was provided in an orderly fashion: data from subject 1, series 1, aim high, low, and baseline, then subject 1, series 2, aim high, low, and baseline, etc. However, a too-regular input order can bias a network to

learn "too much too fast" about the first few data items, to the detriment of being able to learn about later data items. To overcome this bias, the subject/data order was scrambled using the sorting method mentioned above. Of course, order of items *within* a row of data was untouched.

⁸ Excepting the baseline condition, of course, which had no explicit direction for intentional aim.

⁹ Actually, even if the training dataset *had* been unsymmetrically biased, useful information could still be gained about whether the network had learned the desired associations by comparing results of the transfer test against those of similarly generated control tests.

¹⁰ One might argue that the random dataset is simply a variation of the single button-press PEAR experiment. The difference is that here an entire *set* of *series* is created with a single button press. Of course, if one accepts the PEAR evidence suggesting that a single button-press experiment can produce significant departures from chance expectation, then our fundamental assumption about the *meaning* of a control dataset — as a condition outside the realm of experimental influence — is immediately suspect.

¹¹ I thank Joseph Lubin, Robert Jahn, Roger Nelson, Brenda Dunne, Alain Kornhauser, Angela Thompson, York Dobyns, Paul Rake and John Bradish for their valuable comments.

¹² Hereafter this is referred to as the "maximum coding scheme."

¹³ Strictly speaking, the error did not asymptote to a constant, but was exponentially approaching zero. The process was stopped after there were no changes in the error term in the third decimal place.