# Robustness of Tensor Product Networks Using Distributed Representations

William H. Wilson
School of Computer Science and Engineering
The University of New South Wales

Graeme S. Halford
Department of Psychology
University of Queensland

## Abstract

*This paper describes experiments on on the robustness of tensor product networks using distributed representations, for simple analogical reasoning tasks and straightforward recall tasks. The networks retained satisfactory performance on analogical reasoning tasks when over half of the binding units in the network were destroyed. Performance when some binding units were damaged by causing them to generating random values was robust, though less impressive.*

## 1 Introduction

This paper describes experiments on the robustness of the tensor product network used for the STAR model [2, 3] of analogical problem solving. In the experiments, nodes in the network ("neurons") were damaged by causing them to produce either always zero output, or else random output, and then, in each case, the performance of the resulting network was assessed with various proportions of the neurons so damaged.

## 2 Tensor Product Networks in STAR

Tensor product networks have been used as one-shot learning systems for applications like the modelling of variable binding and for working memory structures in connectionist implementations of production system architectures [1, 5]. A tensor product network has a *rank*: the rank of the networks used in the experiments described in this paper was 3, and we will describe tensor product networks in terms of the rank 3 case.

A rank 3 tensor product network has 3 dimensions - of size *p*,*q*, and *r*, say. Its processing units include *pqr* binding units, and input/output units grouped into 3 vectors of length *p*, *q* and *r*. In the underlying application for the experiments described in this paper, one of the I/O vectors represented a *predicate*, and the other two represented a pair of *arguments*. The information represented in the tensor (i.e. in the binding unit structure) was thus relational information such as larger–than(mare, foal) (the analogical reasoning tasks that the STAR system undertook included proportional analogies such as *mother* is to *baby* as *mare* is to *what?*)

To store the fact larger–than(mare, foal) in the tensor, one computes the tensor product larger–than⊗mare⊗foal of vectors representing the concepts larger–than, mare, and foal and adds it to the values stored in the binding units: thus if l represents larger–than, m represents mare, and f represents foal, one would add $l_i * m_j * f_k$ to the value stored in binding unit $b_{ijk}$.

A simple mechanism allows retrieval ("unbinding") of the stored information - for details see [2, 3, 5].

Figure 1 shows the binding units and input/output vectors in an 8x8x8 tensor product network.

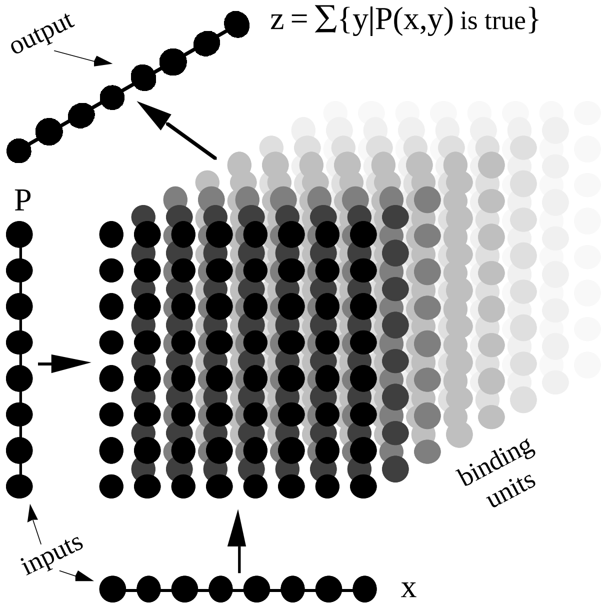

*Figure 1: finding out for which y P(x,y) holds using a tensor product network.*

## 3 Analogical Reasoning in STAR

The STAR model of human analogical reasoning [2, 3] uses tensor product networks based on orthonormal sets of representation vectors, with *exact unbinding* [5]. As an aim of the STAR model is to provide a distributed model of analogical reasoning, it is desirable for the representation vectors to have a high proportion of non-zero components. This can be achieved in a systematic way by using the rows of a Hadamard matrix, suitably normalised. (An *n*x*n* Hadamard matrix is a square matrix all of whose entries are ±1, such that $HH^T=nI_n$.)

Hadamard matrices of size *n*x*n* are known to exist for a wide range of *n* divisible by 4 (including all *n* of the form $2^m$) [4]. For example, for *n*=8, there is the matrix:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \end{bmatrix}$$

The mechanism used by STAR, then, is to associate with each concept that is to be represented a representation vector which is a member of an orthonormal basis for the representation space and which has no zero components. Facts involving the concepts of interest, whether relevant to the reasoning problem or not, are stored in the binding units.

In these networks, the I/O vectors can serve both as input or output vectors at any time. Thus, to outline the STAR method, to solve *mother* is to *baby* as *mare* is to *what?* the system presents (representations of) mother and baby as inputs to the argument vector in the tensor, and the output at the predicate vector is the sum of the vectors representing predicates relating mother to baby: e.g. loves, feeds, larger-than.

This unanalysed vector is then presented as an input to the tensor product network (again on the predicate "axis") along with the representation of mare, on the first argument axis. The output on the second argument axis is a linear combination of concepts, and, as described in [2, 3], the basis vector with the highest coefficient in this linear combination is the answer to the proportional analogy problem. This is illustrated in Figure 2 in simplified form.

## 4 The Experiments

The networks used for simulating analogical reasoning were damaged in two ways:

(a) by "killing" binding units in the tensor product network -that is, setting their remembered value to zero, so that their contribution to any output vector would be zero under all circumstances;

(b) by "randomizing" binding units, so that their value on each occasion one was called for in order to contribute to computing an output vector was a small uniform random value (a different one on each occasion).

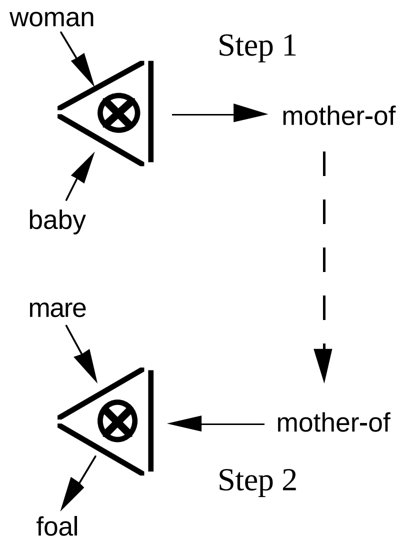

*Figure 2: Simplified analogical reasoning with a tensor product network - the triangular symbols represent a rank 3 tensor product.*

The effect of randomization damage, of course, depends on the range [–ρ,ρ] of numbers from which the uniform random value is drawn.

When binding units in the tensor product were killed in the experiments, the unit to be killed was chosen in a uniform random way: if the unit so selected happened to be already "dead", then another unit was chosen in the same way. The same procedure was used in choosing a unit to randomize.

The methods used to test the effect of damaging the network were to kill or randomize a certain percentage of the units and then to compare the performance of the network (a) on standard analogical reasoning tasks, and (b) on straightforward recall tasks ("is larger-than(mare, rabbit) true") with the known performance of an intact network.

In the case of an analogical reasoning task, the criterion for success is that the pattern representing the correct answer should have a coefficient significantly larger than the next largest coefficient.
In the case of a recall task, the measure of recall of a known fact in an intact network is 1.0, computed from the formula:

$$\mathit{recall}(P(\mathrm{arg1},\mathrm{arg2})) = \sum_{ijk} b_{ijk} P_i \mathrm{arg1}_j \mathrm{arg2}_k$$

and the degree of performance degradation of a damaged network was measured by a triple ($r_f$, $r_n$, $r_m$) where $r_f$ is the proportion of known facts for which the recall measure is greater than some cutoff C, $r_n$ is the proportion of a sample of expressible non-facts for which the recall measure is greater than the cutoff C, and $r_m$ is the sum of the recall measures for the known facts. By “known facts”, we mean the facts which the original undamaged network knew. The choice of the cutoff C is a trade-off, if C is large, the network will be deemed to forget known facts quickly, and if C is small, it will seem to “recall” non-facts when the proportion of damaged units is small. All of $r_f$, $r_n$, $r_m$ are normalised to lie between 0 (nothing recalled) and 1 (everything recalled).

The results of these experiments were compared with the results of experiments using a tensor product network which did not use distributed representations. Without loss of generality, one can use the standard basis of the underlying vector spaces for the representation vectors in this case.

## 5 Results

### 5.1 Killing units: effect on analogy

The analogy is regarded as solved provided the recognition score for the correct answer to the analogy problem is significantly greater than the next highest score. This proved to be the case for the distributed representation until around 80% of the binding units had been killed.

Figure 3 illustrates a typical pattern of degradation. With the non-distributed representation, performance varied widely from run to run, depending on when critical binding units happened to be randomly deleted. There is some robustness built into the analogy algorithm, in that more than one fact may tend to suggest that a particular concept is the solution to the analogy problem (mare:foal is analogous to woman:baby in more than one respect), so that in some runs a large number of units can be destroyed before the analogy problem becomes unsolvable. However, in the worst case, damaging less than 10% of the neurons could make a particular analogy problem unsolvable. In the runs observed in this experiment, in fact, the standard analogy problem did not become unsolvable until 40% of the binding units were killed.

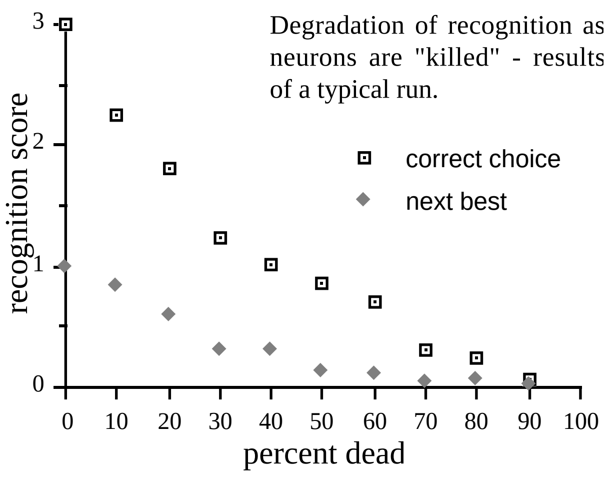

Figure 3: Effect of killing units on analogy-solving, using a distributed representation.

### 5.2 Killing units: effect on recall

With the recall cutoff C set to 0.4, all known facts could often be recalled until more than 30% of units hand been destroyed. A typical run is shown in Figure 4. With a non-distributed representation, again, results were variable: the first 10% of units killed could include the 20 binding units that encode the 20 facts. Killing units in this representation *cannot* introduce spurious “facts”.

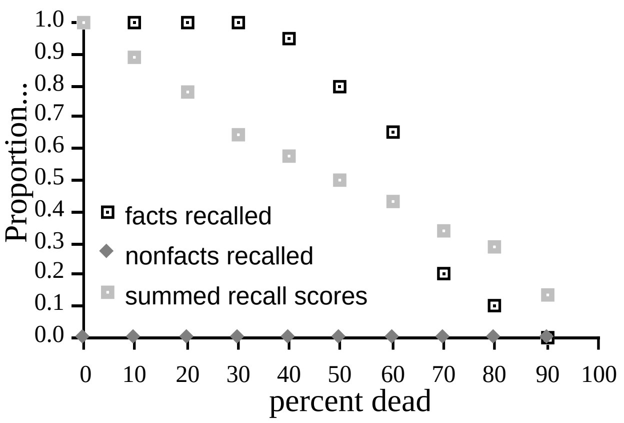

Figure 4: Effect of killing units on recall, using a distributed representation.

### 5.3 Randomizing units: effect on analogy

When binding units were randomized, in the sense described above, the network remained robust from the point of view of analogical tasks, although, at a “noise volume” setting of $\rho = 0.1$, not quite as robust as when units were killed instead. An example of the degradation of performance as binding units are randomized is shown in Figure 5. With a non-distributed representation, performance is uneven and unreliable, depending on whether critical binding units/facts are randomized early.

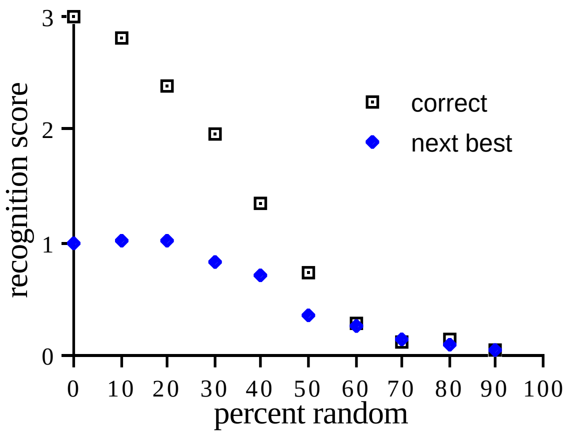

Figure 5: Effect of randomizing units on analogy-solving, using a distributed representation.

### 5.4 Randomizing units: effect on recall

With the non-distributed representation, the effect of randomizing a binding unit depends on the relationship between the volume setting ρ and the recall cutoff C. If ρ < C, then a randomized unit will never be recognized, so randomized "non-fact" units never cause problems, and randomized "fact" units are essentially forgotten, just as when units are killed. If ρ ≥ C, then a randomized unit *may* cause a fact to be recognized if the random value generated happens to be greater than C. In the experiments 0.1 = ρ < C =0.4, so randomized units were essentially killed for purposes of the non-distributed representations. (The random values could still add up to a quantity large enough to affect an analogical inference, of course).

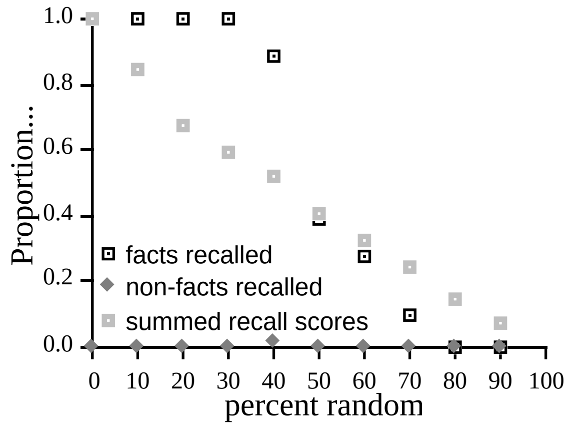

Figure 6: Effect of randomizing units on recall, using a distributed representation.

With a distributed representation, as shown in Figure 6, recall remains good until about 40 % of the binding units are randomized (with ρ = 0.1 and C =0.4), and then decays fairly rapidly. It should be noted that the recall performance (and indeed the analogical reasoning performance) at a particular level of randomization is *not* a constant, as the randomized units produce different outputs each time they are used. The points plotted in Figure 6 are an average of 5 recall attempts.

## 6 Conclusions

Tensor product networks using orthonormal distributed representations for the activations projected into the network along the input axes are relatively robust to destruction of individual neurons. They are less robust to randomization of neurons: i.e. changing neuron output so that the neuron emits a small random output no matter what input it gets, but still perform reasonably well under these conditions.

## References

[1] C.P. Dolan and P. Smolensky, Tensor product production system: A modular architecture and representation, *Connection Science* **1** (1989) 53-68.

[2] G.S. Halford, J. Wiles, M.S. Humphreys, & W.H. Wilson, Parallel distributed processing approaches to creative reasoning: Tensor models of memory and analogy, in *AI and Creativity* ed.T Dartnall, S.Kim, and F. Sudweeks, AAAI Technical Report (1993)

[3] G.S. Halford, W.H. Wilson, J. Guo, J. Wiles, and J.E.M. Stewart (in press), Connectionist implications for processing capacity limitations in analogies, in K.J. Holyoak and J. Barnden (editors) *Advances in Connectionist and Neural Computation Theory: Volume 2: Analogical Connections*, Norwood, NJ: Ablex.

[4] J. Seberry & M. Yamada, Hadamard matrices, sequences, and block designs, pp. 431-560 in *Contemporary Design Theory: A Collection of Surveys*, ed. J.H. Dinitz and D.R. Stinson, John Wiley, 1992

[5] P. Smolensky, Tensor product variable binding and the representation of symbolic structures in connectionist systems, *Artificial Intelligence* **46** (1990) 159-216