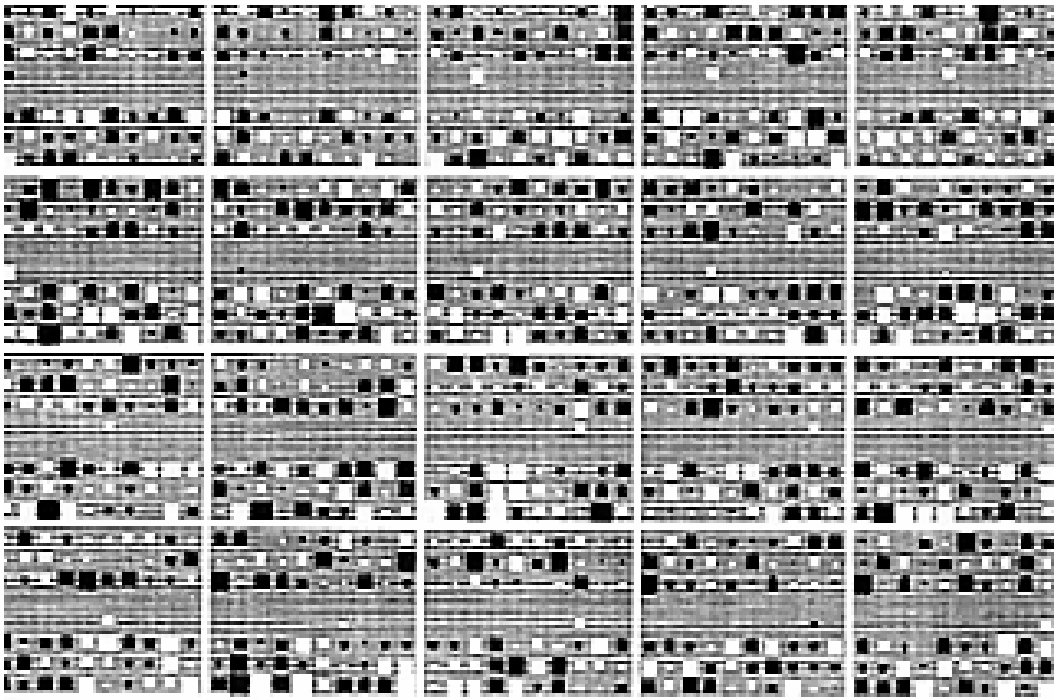
# Neural Nets and Symbolic Reasoning

## Distributed representations and auto-associative memory

# Outline

- Local and distributed representations
- Coarse Coding
- RAAM
- Syntactic transformations on distributed representations
- General conclusions

# 1 Local and distributed representations



# The difference

(A) Local representation: Each neuron represents a single concept and each concept is represented by a single neuron

(B) Distributed representation: Concepts are represented by patterns of activity over a collection of neurons.

- Each concept (e.g., an entity, token, or value) is represented by a pattern of neural activity in which more than one neuron is active.
- Each neuron participates in the representation of more than one concept.

In a distributed representation one cannot interpret the meaning of activity on a single neuron in isolation: the meaning of activity on any particular neuron is dependent on the activity in other neurons.

# Advantages of distributed representation

1. Representational efficiency: distributed representations form a more efficient code than localist representations, provided that only a few concepts are to be represented at once.

   - A localist representation using *n* neurons can represent just *n* different entities.

   - A distributed representation using *n* binary neurons can represent up to $2^n$ different entities

2. Mapping efficiency: a micro-feature-based distributed representation often allows a simple mapping (that uses few connections or weights) to solve a task. The point is that the relevant features may be encoded as single units (or small groups of units) for solving the task. Example: classify color shapes as to whether or not they are yellow.

# Advantages of distributed representation, cont.

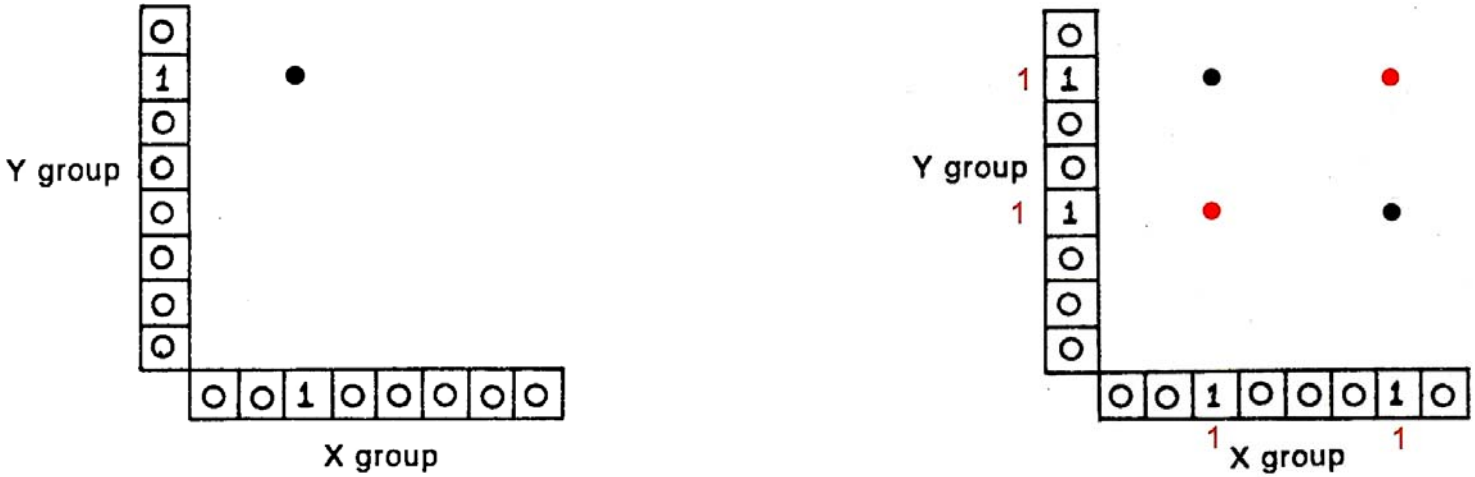
3. Continuity (in the mathematical sense): representing concepts in continuous vector spaces allows powerful gradient-based learning techniques such as backpropagation to be applied to many problems, including ones that might otherwise be seen as discrete symbolic problems.

4. Soft capacity limits and graceful degradation: distributed representations typically have soft limits on how many concepts can be represented simultaneously before ghosting or interference becomes a serious problem. Also, the performance of neural networks using distributed representations tends to degrade gracefully in response to damage to the network or noise added to activations.

# Advantage of local representations

- Local representations are far simpler to understand, implement, interpret, and work with.
- If distributed representations do not provide significant advantages for a particular application, it may be more appropriate to use a local representation.
- This may be the case in many cognitive modeling applications. The local/distributed distinction is not intrinsic to a network. It relates to an observer-dependent property. It's your decision to give a localist or distributed interpretation of the network's units.

# Problem: Representation of dots in 2D




(a) (b)

(a) A simple way of using two groups of binary units to encode the position of a point in a two dimensional space. The active units in the X and Y groups represent the x- and y-coordinates.

(b) However, when two points must be encoded, it is impossible to tell which x-coordinate goes with which y-coordinate (binding problem).

One solution to the binding problem is the use of local representations: one unit for each possible combination of X and Y values. This can be realized by the outer product: $(\mathbf{x} \otimes \mathbf{y})_{ij} = \mathbf{x_i} \cdot \mathbf{y_j}$

$$\begin{matrix}0\\1\\0\\0\\0\\0\\0\end{matrix} \otimes \begin{matrix}0 & 0 & 1 & 0 & 0 & 0 & 0 & 0\end{matrix} = \begin{matrix}0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\end{matrix}$$

For representing two dots the sum of the two outer products has to be calculated: $(\mathbf{x}_1 \otimes \mathbf{y}_1) \oplus (\mathbf{x}_2 \otimes \mathbf{y}_2)$

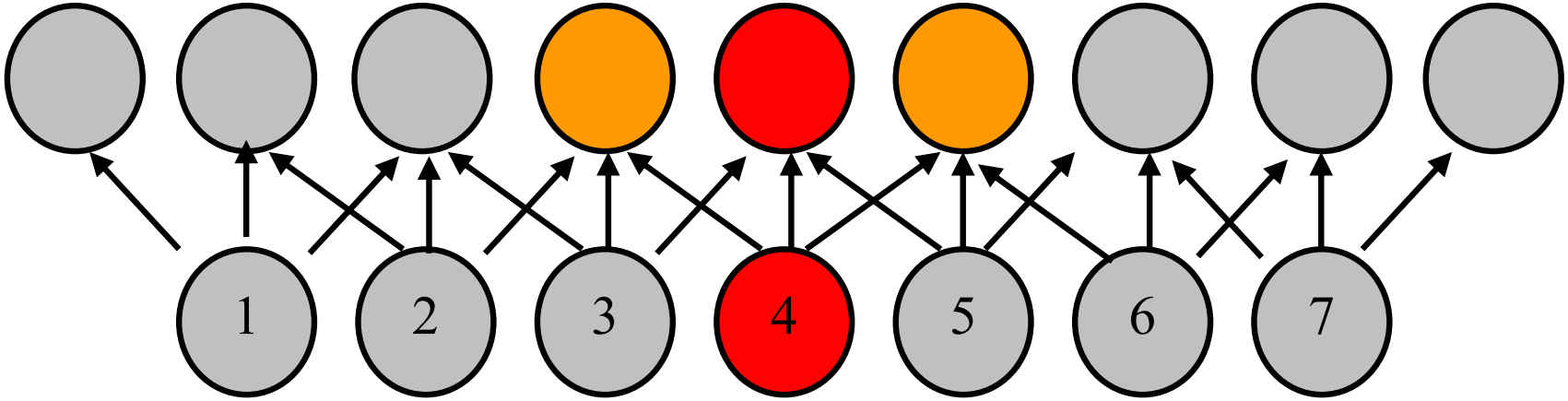
$$\begin{matrix}0\\1\\0\\0\\0\\0\\0\\0\end{matrix} \otimes \begin{matrix}0&0&1&0&0&0&0&0\end{matrix} \oplus \begin{matrix}0\\0\\0\\0\\1\\0\\0\\0\end{matrix} \otimes \begin{matrix}0&0&0&0&0&0&1&0\end{matrix} = \begin{matrix}0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\end{matrix}$$

# Binding vs. accuracy

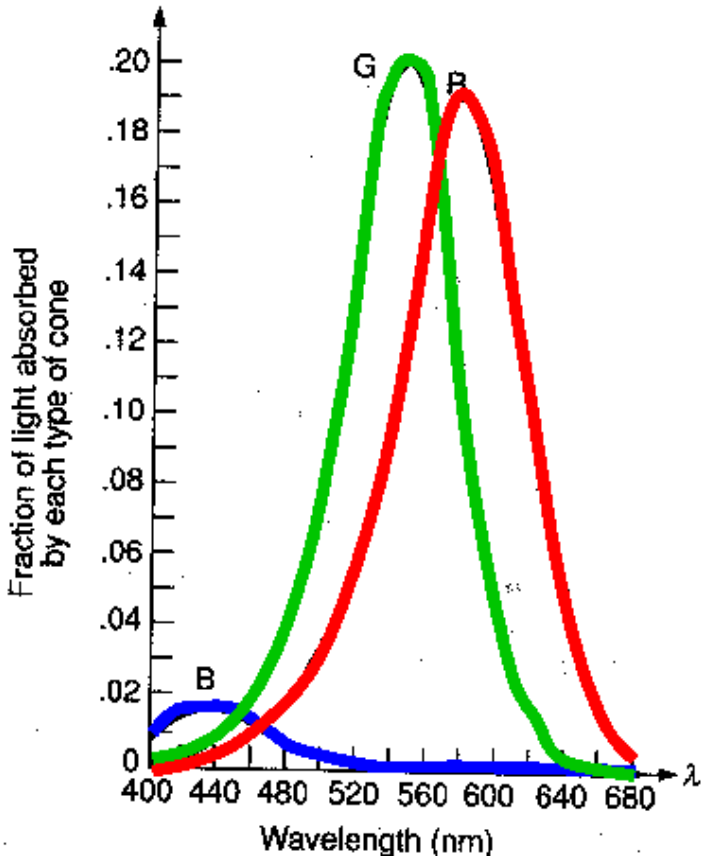
- This kind of local encoding is very expensive: N·N nodes
- It is very inefficient if only a very small fraction of the possible features are present at once. The information conveyed by a unit is very small ($-p \cdot \log p - (1-p) \cdot \log(1-p)$; 1 bit if p=1/2; about 0.1 if p=1/64)
- The accuracy is very low because the system may fail if only one unit fails.
- It would therefore be more efficient to use a *distributed* encoding in which a larger fraction of the units were active at any moment.

➔ Coarse coding

# 2 Coarse Coding

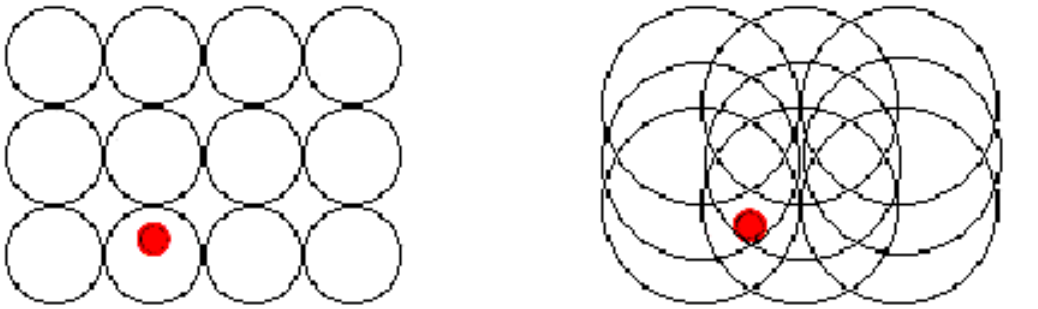


- Coarse coding requires that a property be encoded by a set of detectors
- Usually the detectors will have overlapping sensitivities
- Many examples of this type of coding are found in the human visual system
- Good example: Colour detection



# Two coding schemes for detecting spatial locations

Note how it is possible to obtain fine spatial resolution by combining the responses of poor spatial detectors



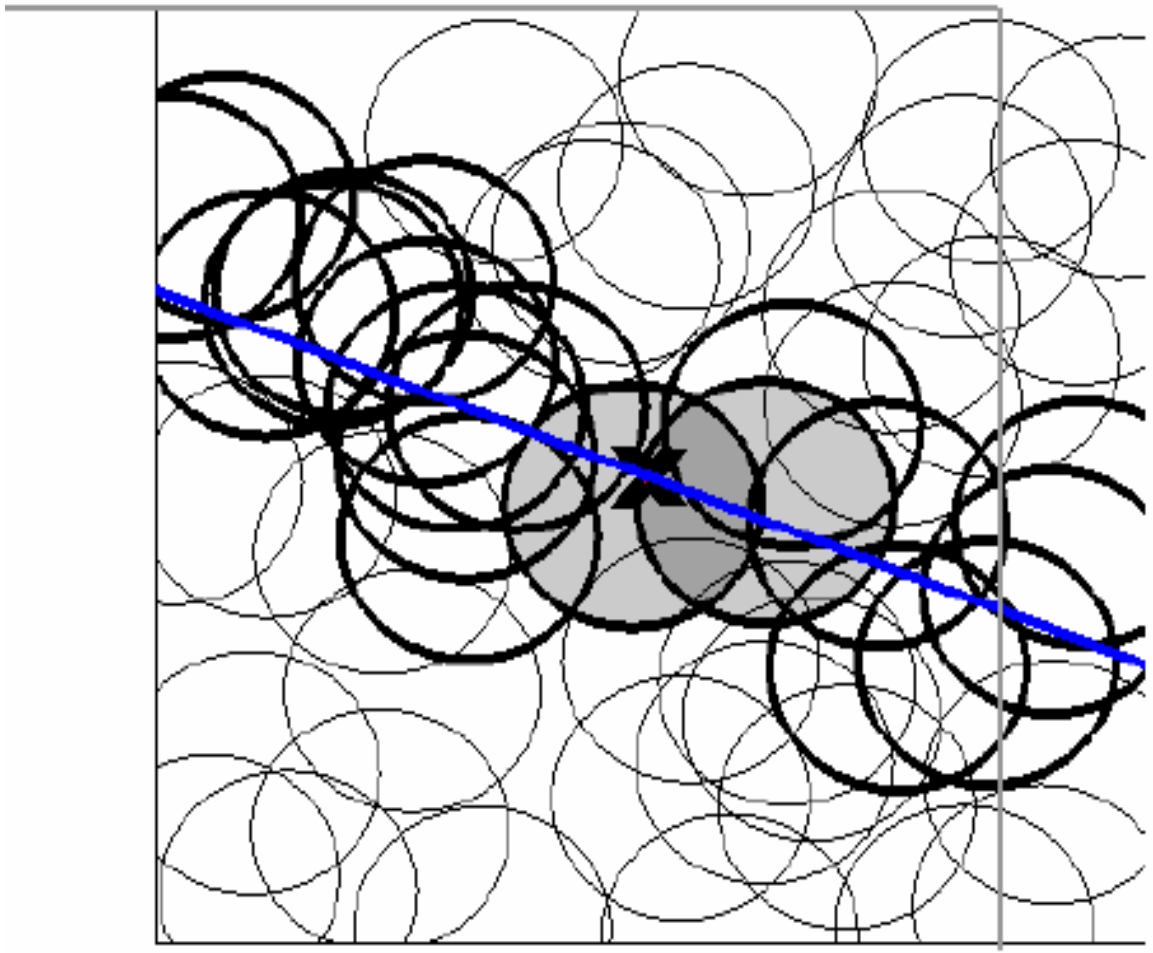
**Local**
**(Fine coding)**

**Distributed**
**(Coarse coding)**

**Larger receptive fields provide better resolution**
**(when whole pattern is considered)**

# Coarse coding of 2D positions

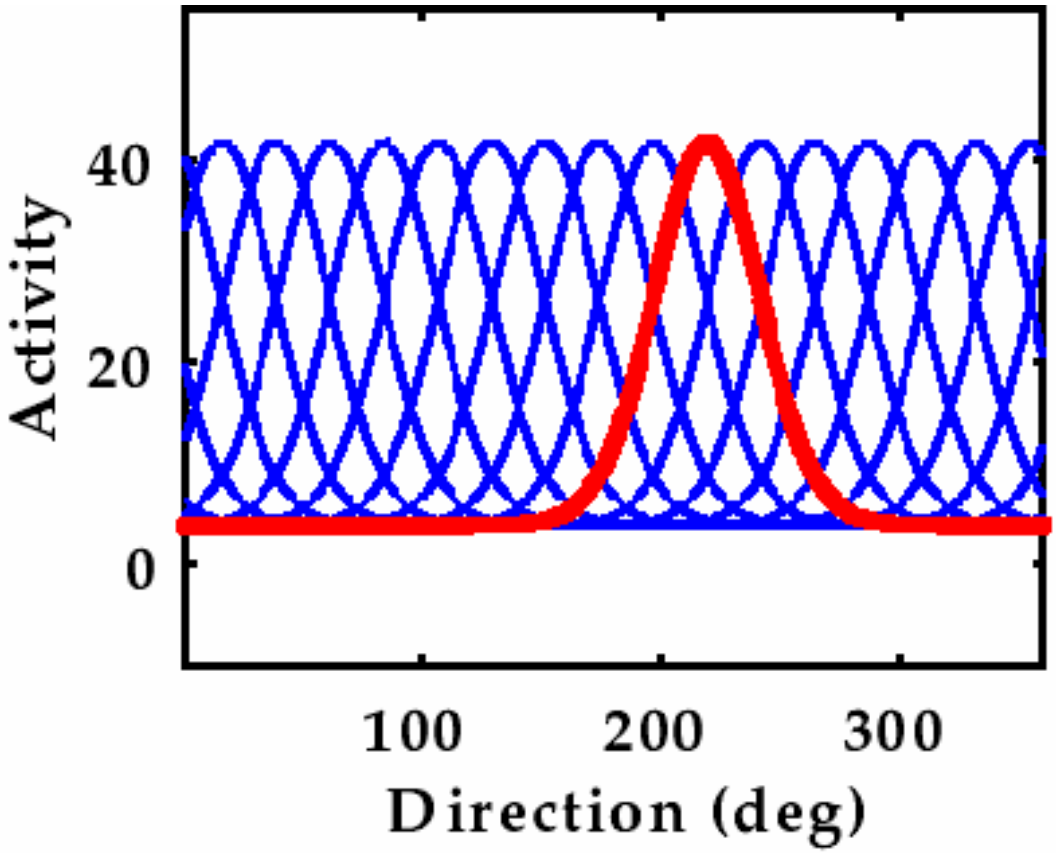
Each neuron is represented by a circle showing its receptive field. Neurons that respond to the point X have dark grey receptive fields. Neurons that respond to some point along the solid line have bold receptive fields.



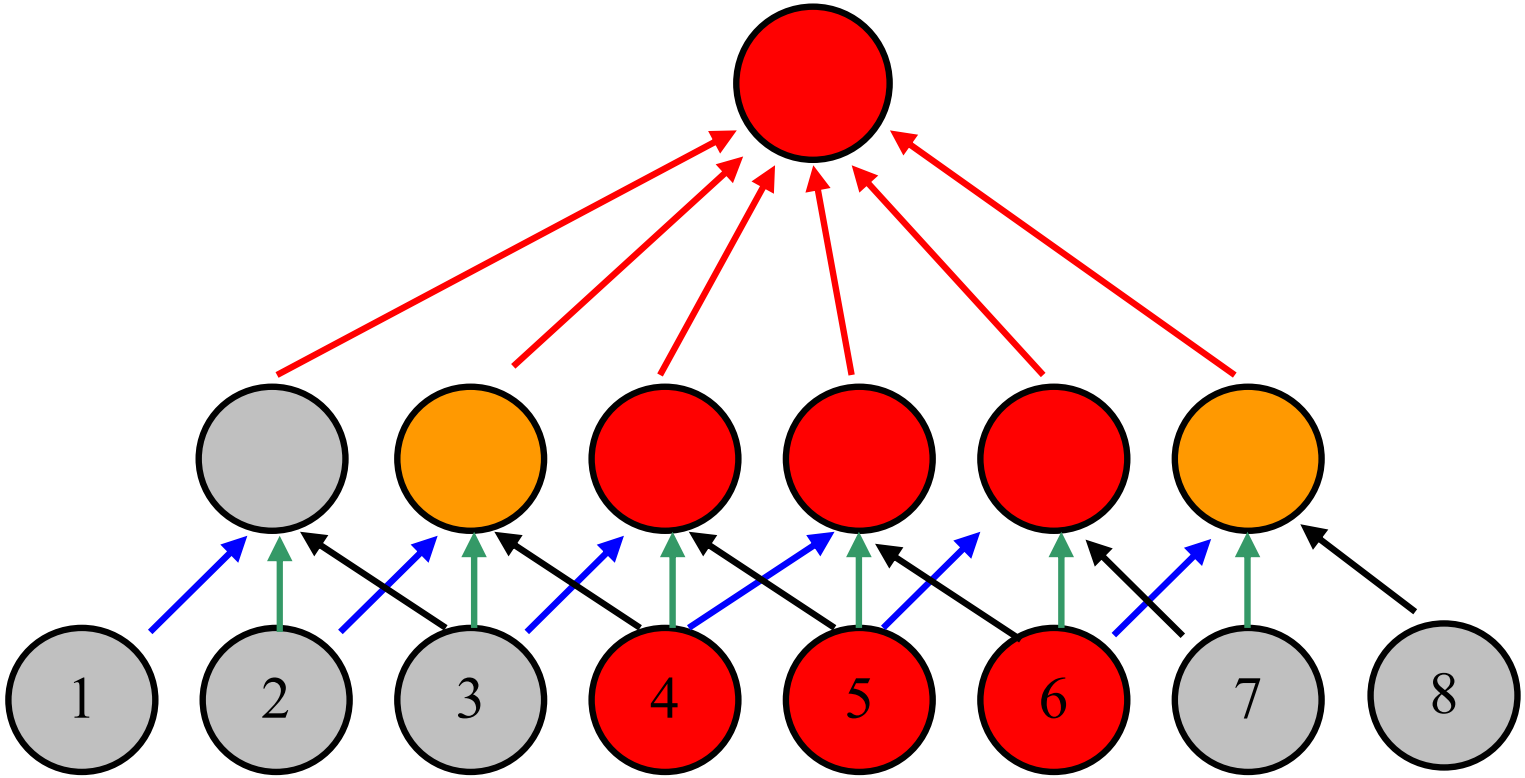
# Coding direction of visual motion

Direction of visual motion is believed to be encoded in the medial temporal (MT) visual area by the responses of a large number of cells with bell-shaped tuning to direction, as illustrated in the Figure (Maunsell & Van Essen, 1983).

In this illustration, 16 idealized tuning curves are shown corresponding to 16 direction-tuned neurons (including neuron 10)



# Coarse coding and translation invariance

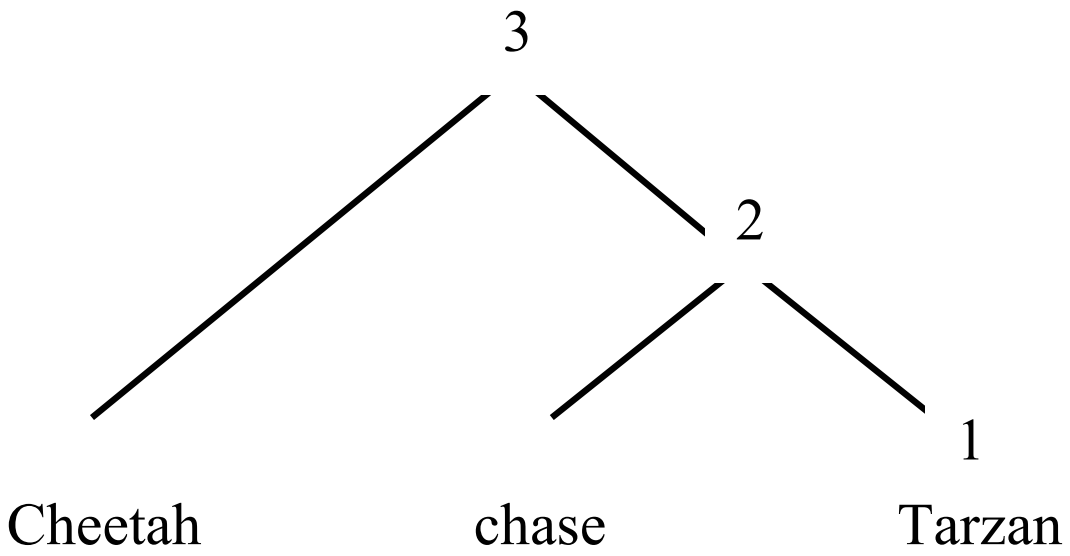


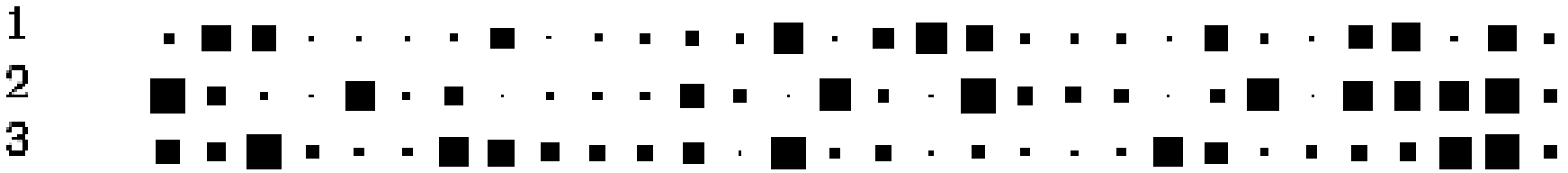

Build useful structure into the net: (1) receptive field spanning three neurons (2) grouping of links with identical values

# Idea of the solution

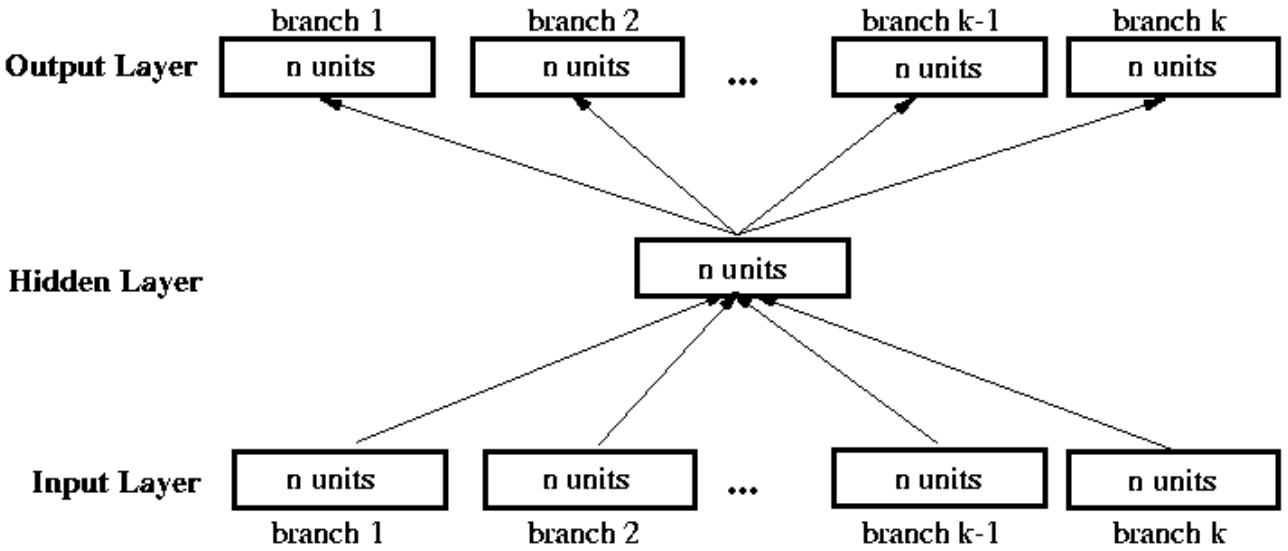
- The guiding idea in solving the translation invariance problem is to build as much useful (innate?) structure into the network as possible
- The idea of receptive fields: the hidden units are connected to a limited number of neurons on the first level
- The spatial relation between the hidden units and the corresponding first level units are reflected in similar weights
- Try to understand the solution based on these basic architectural assumptions by examining the actual weights and drawing the network! (tlearn exercise 5)

# 3 Pollack's recursive auto-associative memory (RAAM)

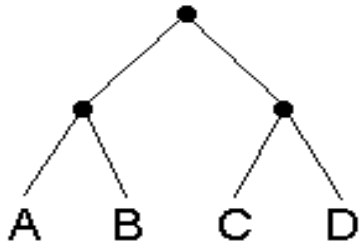


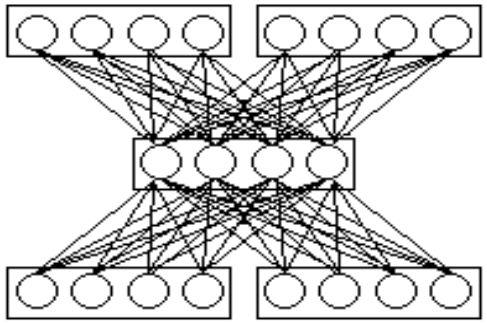
- Technique for encoding recursive data structures (lists and trees) as distributed numerical vectors, e.g. [0.33, -0.87, 0.13, -0.43]
- Uses feedforward network architecture, backpropagation to encode arbitrary-depth trees with fixed branching factor k into distributed representations of length n
- Network trained by auto-association

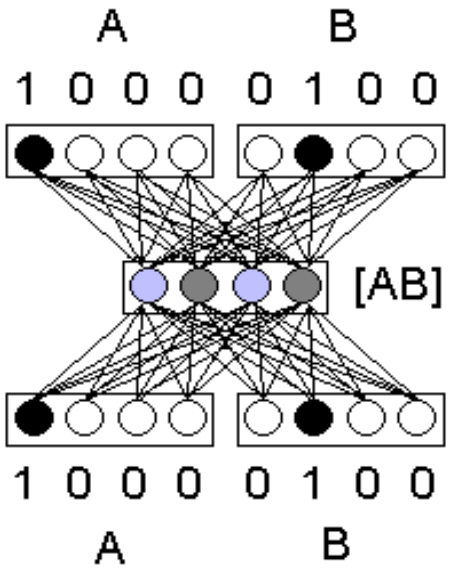


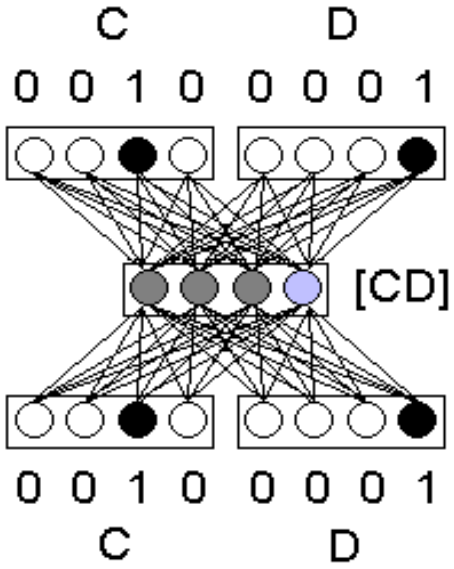
# Example

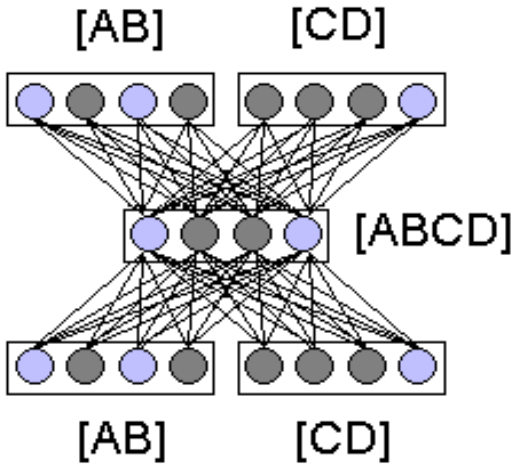


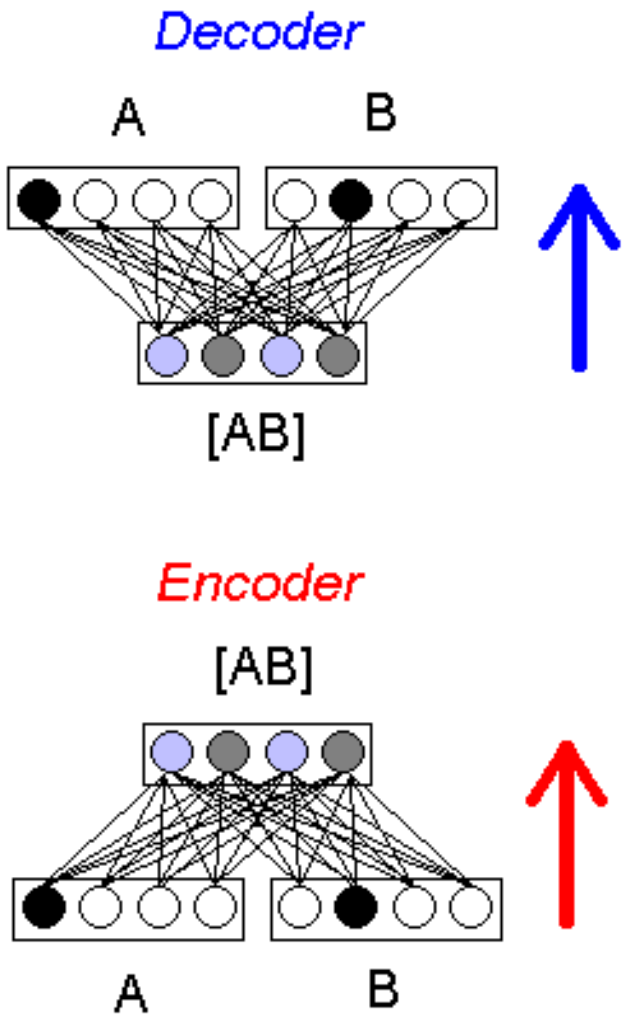

A → 1000
B → 0100
C → 0010
D → 0001








- Auto-association makes it possible to use the network both for encoding and decoding
- Recursive encoding:

  A B → [A B]  C D → [C D]

  [A B]  [C D] → [A B C D]
- Recursive decoding

  [A B C D] → [A B]  [C D]

  [A B] → A B  [C D] → C D

- Binary RAAM realizes a function $raam_2$: $A^n \times A^n \rightarrow A^n$ , where $A^n$ designates the activation vectors of length n.
- Whereas in symbolic systems *concatenative composition* plays the main role in constructing *structured symbolic representations*, in PDP the function $raam_k(x_1, \ldots, x_k)$ plays a similar role (*functional composition*)
- cf. Tim van Gelder (1990): Compositionality: A connectionist variation on a classical theme. *Cognitive Science* 14.
- Encoding lists in PDP
  abcd: $raam_2$(d, $raam_2$(c, $raam_2$(b, $raam_2$(a, nil))))
  the atoms *a*, *b*, *c*, *d* are normally encoded as *local representations*.

# Importance of RAAM

- Following the publication of (Pollack 1990), RAAM gained widespread popularity as a model of linguistic structure
- Some researchers (Blank, Meeden, and Marshall 1991) found it an attractive way of "closing the gap" between the symbolic and sub-symbolic paradigms in cognitive science
- Others (Van Gelder 1990) saw in RAAM a direct and simple refutation of the traditional cognitive scientists' counterattack against connectionism
- Chalmers (1990) went as far as to show how traditional syntactic operations like transformations could be performed directly on RAAM representations

# 4 Syntactic transformations on distributed representations (Chalmers 1990)

- Demonstrating that the distributed representations formed by RAAM are well-suited for structure-sensitive operations (transformations). Not only is compositional structure *encoded* implicitly in a pattern of activation, but this implicit structure can be *utilized* by the familiar connectionist devices of feedforward/ backpropagation in a meaningful way.
- Demonstrating that connectionism offers the opportunity to operate on compositional representations holistically, without first proceeding through the step of extracting certain substructures.
- Demonstrating that connectionism is much more than a matter of implementation

# The procedure

1. Training the RAAM for encoding the syntactic structures of sentences such as *John loves Michael* or *Michael is loved by John*. (125 sentences were uses of active form and 125 of passive form)
2. Testing the RAAM
3. Training the transformation network
4. Testing the transformation network

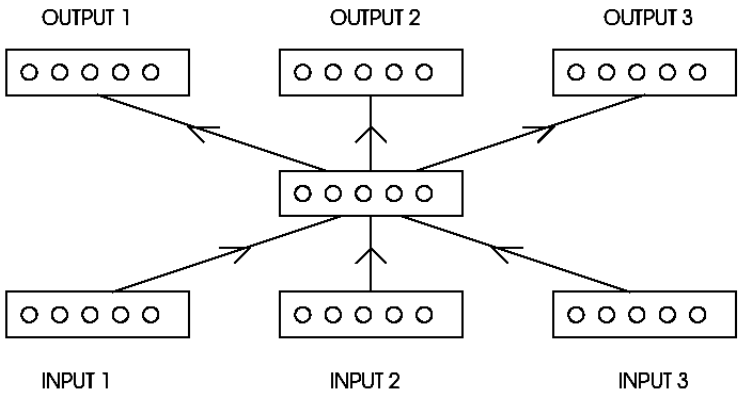
# Local representation of terminal elements

| WORD | REPRESENTATION | | |
|---|---|---|---|
| JOHN | 0 1 0 0 0 0 | 1 0 0 0 0 | 0 0 |
| MICHAEL | 0 1 0 0 0 0 | 0 1 0 0 0 | 0 0 |
| HELEN | 0 1 0 0 0 0 | 0 0 1 0 0 | 0 0 |
| DIANE | 0 1 0 0 0 0 | 0 0 0 1 0 | 0 0 |
| CHRIS | 0 1 0 0 0 0 | 0 0 0 0 1 | 0 0 |
| LOVE | 0 0 1 0 0 0 | 1 0 0 0 0 | 0 0 |
| HIT | 0 0 1 0 0 0 | 0 1 0 0 0 | 0 0 |
| BETRAY | 0 0 1 0 0 0 | 0 0 1 0 0 | 0 0 |
| KILL | 0 0 1 0 0 0 | 0 0 0 1 0 | 0 0 |
| HUG | 0 0 1 0 0 0 | 0 0 0 0 1 | 0 0 |
| IS | 0 0 0 0 1 0 | 1 0 0 0 0 | 0 0 |
| BY | 0 0 0 0 0 1 | 1 0 0 0 0 | 0 0 |
| NIL | 0 0 0 0 0 0 | 0 0 0 0 0 | 0 0 |

NIL is used to stay with Pollack's triadic format

```
Categorical   Particular   spare
Information   words        units
```

# Training the RAAM

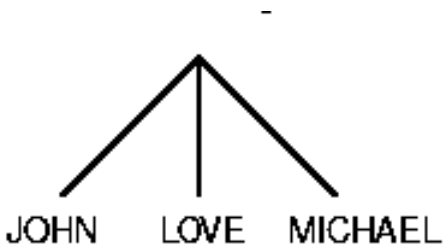


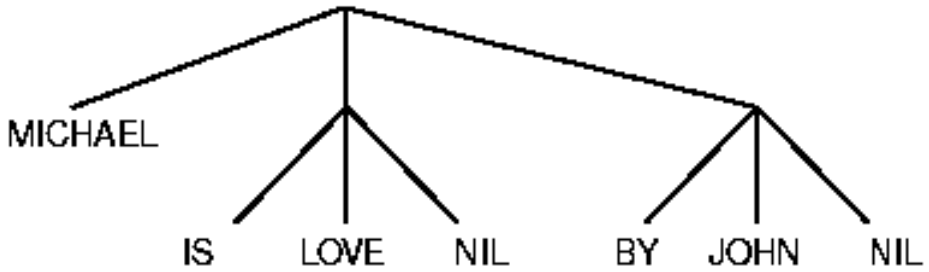

- The initial corpus of 80 sentences, 40 of each type, was used to train the RAAM (architecture: 39×13×39)
- There were 160 cycles of the network per epoch, one for each of the original sentences and three for each of the passivized sentences (corresponding to the three internal nodes of the tree)
- The initial learning rate was 0.1; this was lowered to 0.025 by the end of the training procedure. A momentum rate of 0.9 was used
- The network was trained for 6400 epochs, by which time all but 20 output units (out of 6240 in all) had an error of less than 0.05. The maximum error on any output unit was 0.12.

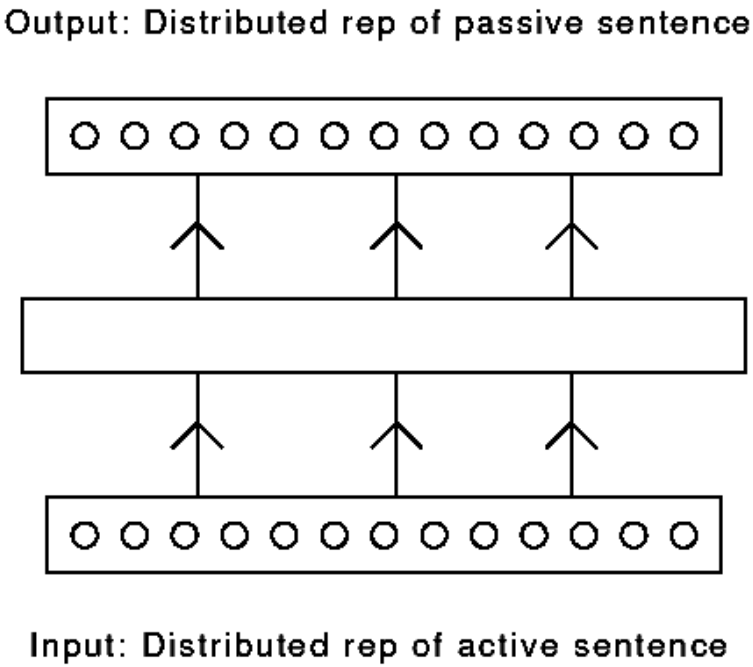
# Testing the RAAM

- Test for terminal elements: an vector x is deemed non-terminal iff
  - More than 2 units had an activation between 0.15 and 0.85, or
  - either of the two spare units had activation greater than 0.5, or
  - more than one categorical part unit had activation greater than 0.5
- As an initial test, the 80 sentences in the training corpus were recursively encoded, and then decoded using the above method. Unsurprisingly, all 80 were decoded back to the original sentence
- As a test of generalization, 80 new sentences from the testing corpus were encoded and decoded in the same fashion. All but 13 of these sentences decoded back to the original sentence. Of the 13 mistakes, only one was decoded to an incorrect sentence structure; the other 12 all had one incorrect word within a correct sentence structure.

# The transformation network




- Chalmers trained a feedforward transformation network using the first 80 compressed representations, appropriately paired as input and target patterns (13 to 29 hidden units)
- The learning = 0.1, momentum = 0.9. The network was trained for 1500 epochs, consisting of 40 cycles each. At the end of this time, only one output unit (out of 1560 per epoch) had an error greater than 0.05.

# Testing the transformation network

- As an initial test, the 40 “active” sentences from the training corpus were encoded using RAAM, and fed to the transformation network, yielding a new distributed representation. These representations were decoded using RAAM. All 40 of these decoded to the correct passive sentence.

- As a test of generalization, 40 new input/output pairs of sentences were used. Out of these 40, 26 decoded to the correct passivized sentence; of the remaining 14, one had incorrect sentence structure, and the other 13 had a single incorrect word.

# Conclusions

- The network has developed a high degree of sensitivity to the structure encoded implicitly in the distributed representations
- The representations support systematic processing. Explicit constituent structure is not needed for systematicity; implicit structure is enough
- It is impossible to describe the operation of this system, at any level of functional abstraction, as an implementation of a pure symbolic process. The compositional representations used here are not operated on reductionistically, by splitting into their constituent parts and then processing. Rather, the operations are direct and holistic.

# 5 General conclusions

- Distributed representations have many advantages
  - representational efficiency
  - use of gradient-based learning techniques
  - graceful degeneration
  - solving the problem of translation invariance
- Representations built by RAAM can have arbitrarily complex syntactic structure. The procedure RAAM works fairly well, however some limitations should be mentioned:
  - difficulties with the decision whether an output pattern corresponds to a terminal element or requires a further cycle of decompression (for RAAM networks with more than 5 levels)
  - Capacity limitations for autoassociation with more than 5 levels

- The recursion in the RAAM procedure needs an external controller; i.e. the procedure is not completely expressed within the framework of neural networks. The user/controller of the procedure determines what kind of structure is encoded.
- Representations have complex, distributed microstructure containing much more information than symbolic representations. This is helpful not only for describing processing but also for finding out interesting generalizations (cf. Chalmers 1990)
- Within RAAM, syntactic operations can be treated as holistic structure-sensitive operations. Syntactic transformations are handled very easily by pure symbolic systems. However, the real potential of holistic operations arises because of a special property of connectionist representations: they can carry their own content, or at least part of it (Chalmers 1990).