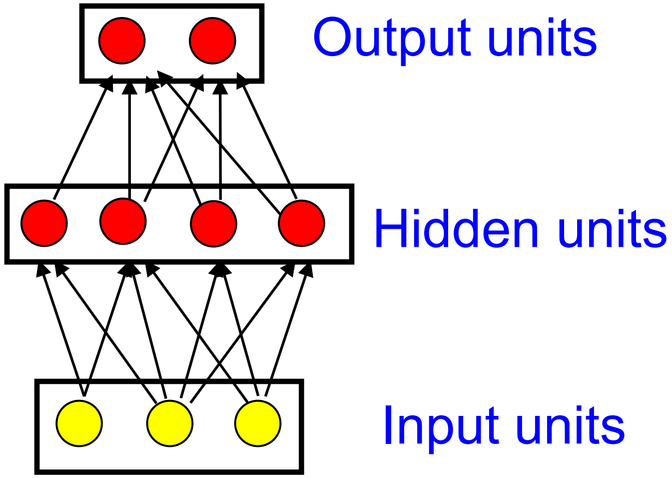
Knowledge Processing with Neural Networks

Lecture 5: Sequences in Supervised Neural Architectures

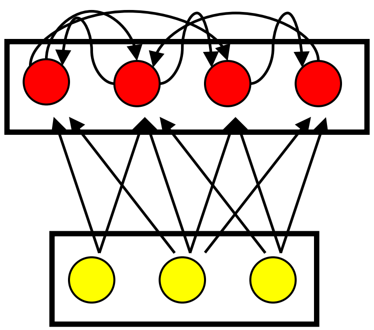
http://www.informatik.uni-hamburg.de/WTM/

Revision: Types of connectivity

- Feedforward networks
  - These compute a series of transformations.
  - Typically, the first layer is the input and the last layer is the output.

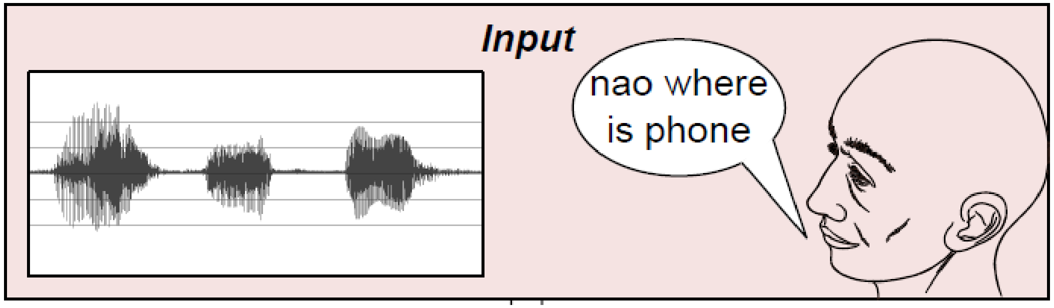


- Recurrent networks
  - These have directed cycles in their connection graph. They can have complicated dynamics.
  - More biologically realistic.



Sequences in neural networks

- Sequences are everywhere, in vision, in speech, in text, in condition monitoring, in movement…
- Neural networks need to represent sequential knowledge
- ***Spatial*** or ***temporal*** approaches
- How to ***represent*** sequences?

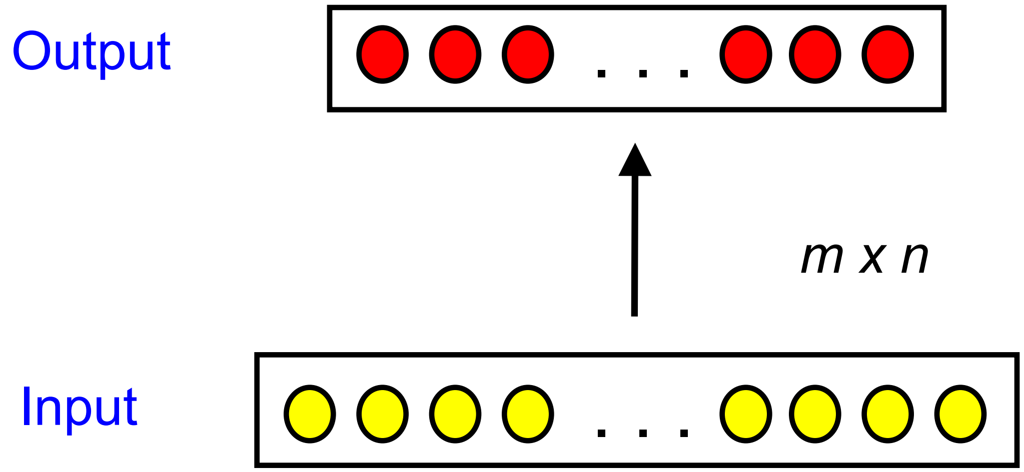


Fixed sequences

- In some cases we know the number of inputs or output components, e.g. cases
- Example case role assignment e.g. "break:
  - The boy broke the window
  - The rock broke the window
  - The window broke
  - The boy broke the window with the rock
  - The boy broke the window with the curtain
- First NP can be: Agent (1,4,5), Instrument (2), Patient (3)
  PP can be: Instrument (4), Modifier (5)

Fixed sequences (cont)

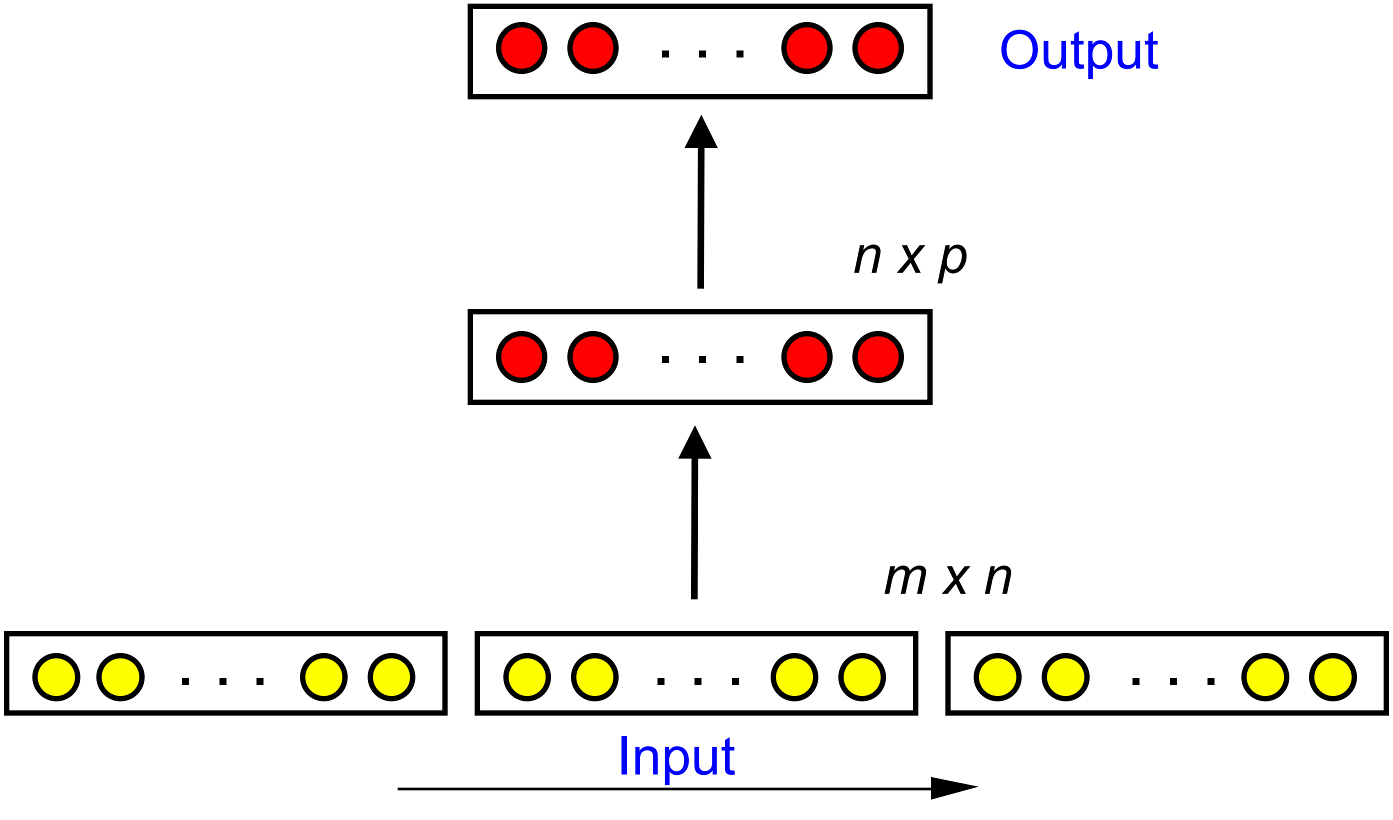
- Feedforward network for restricted input and output (fixed sequences)



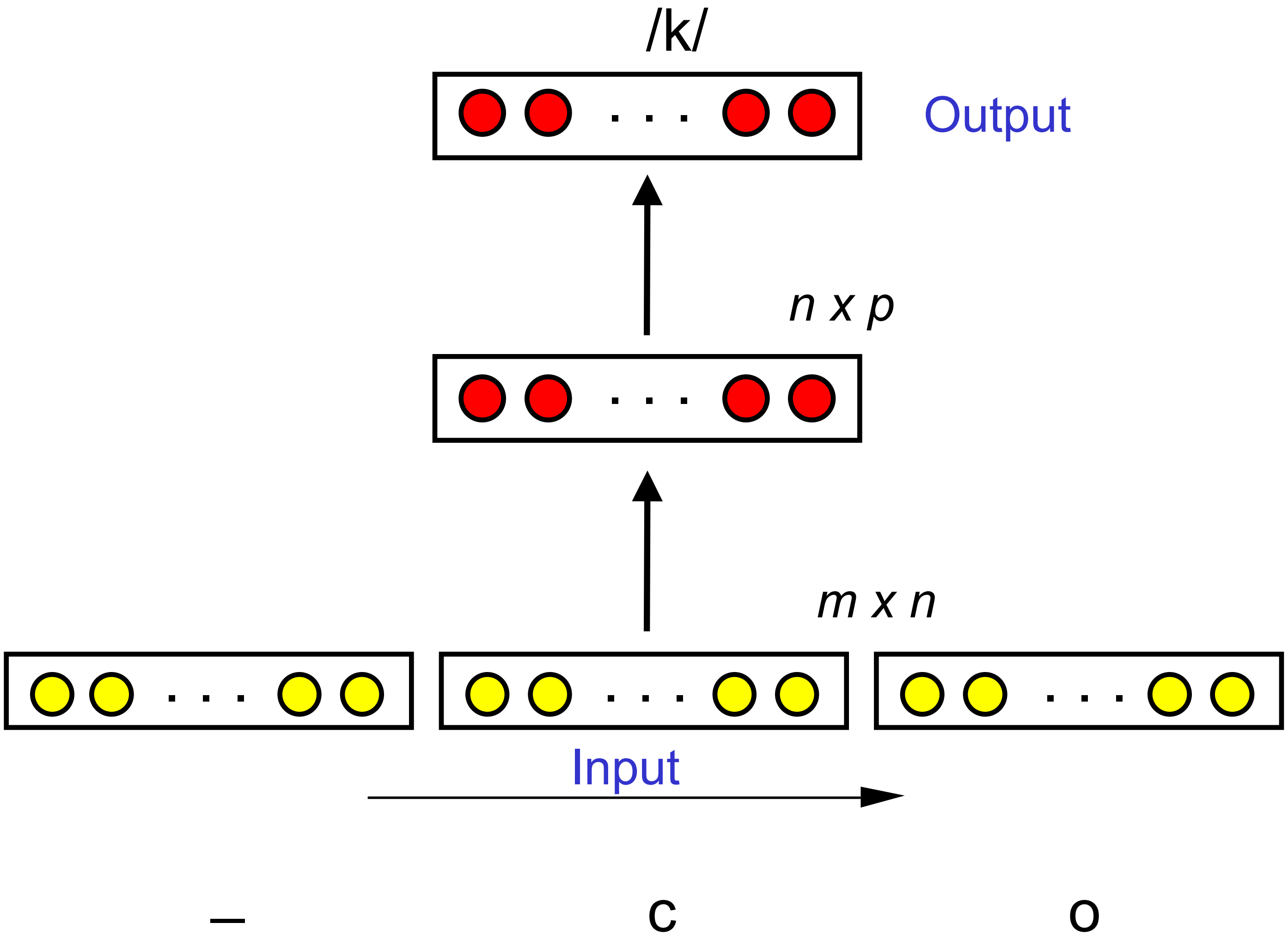
Sliding windows: space for time

- Trading space for time
- Instead of presenting whole sequence only limited part presented
- For instance in NETtalk, a window of seven letters moved over text
- Task was to produce the central phoneme
- Disadvantage of fixed context
- Same applies to Time Delay Networks (TDNN)

Sliding windows for seqentiality (e.g. NetTalk)



Demonstration of NetTalk

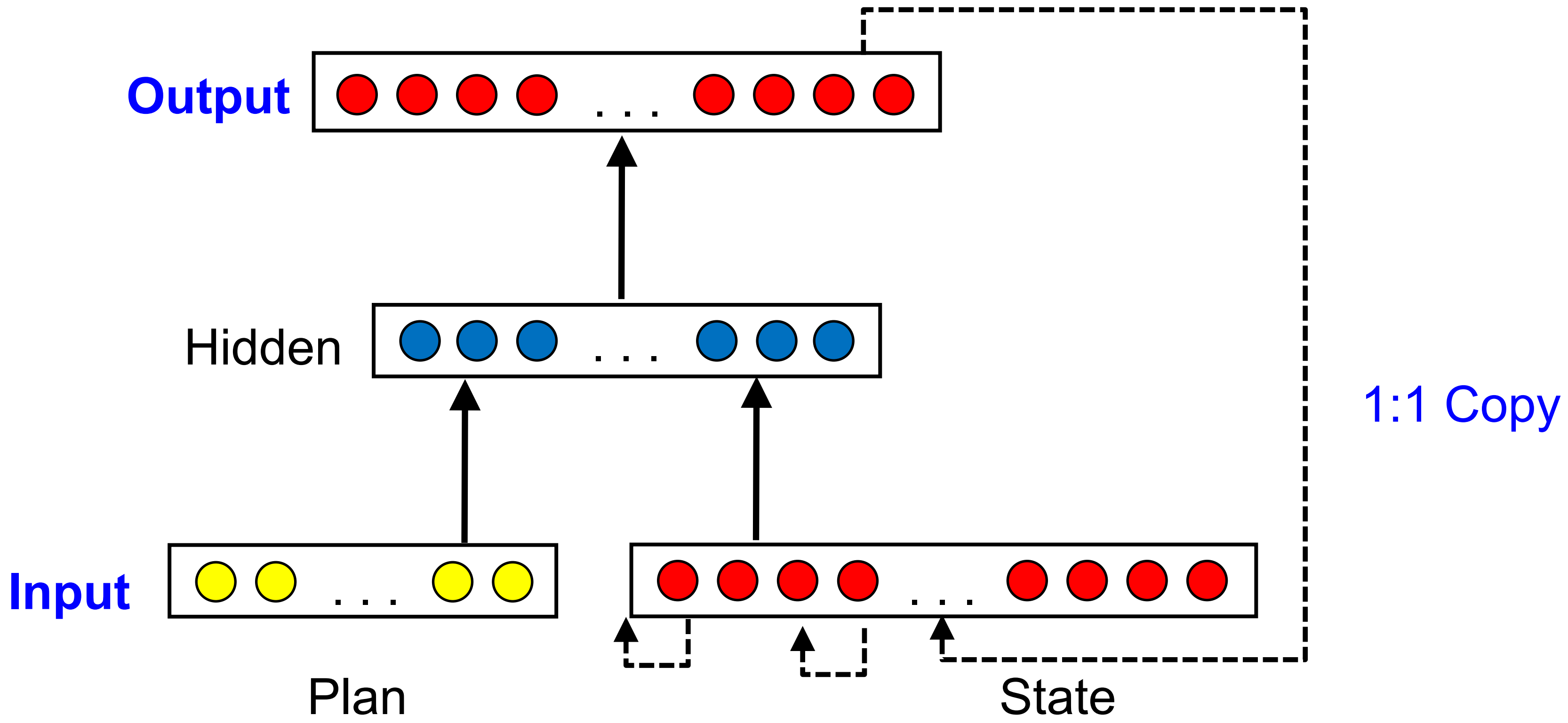


[http://cnl.salk.edu/Media/nettalk.mp3]

Jordan network

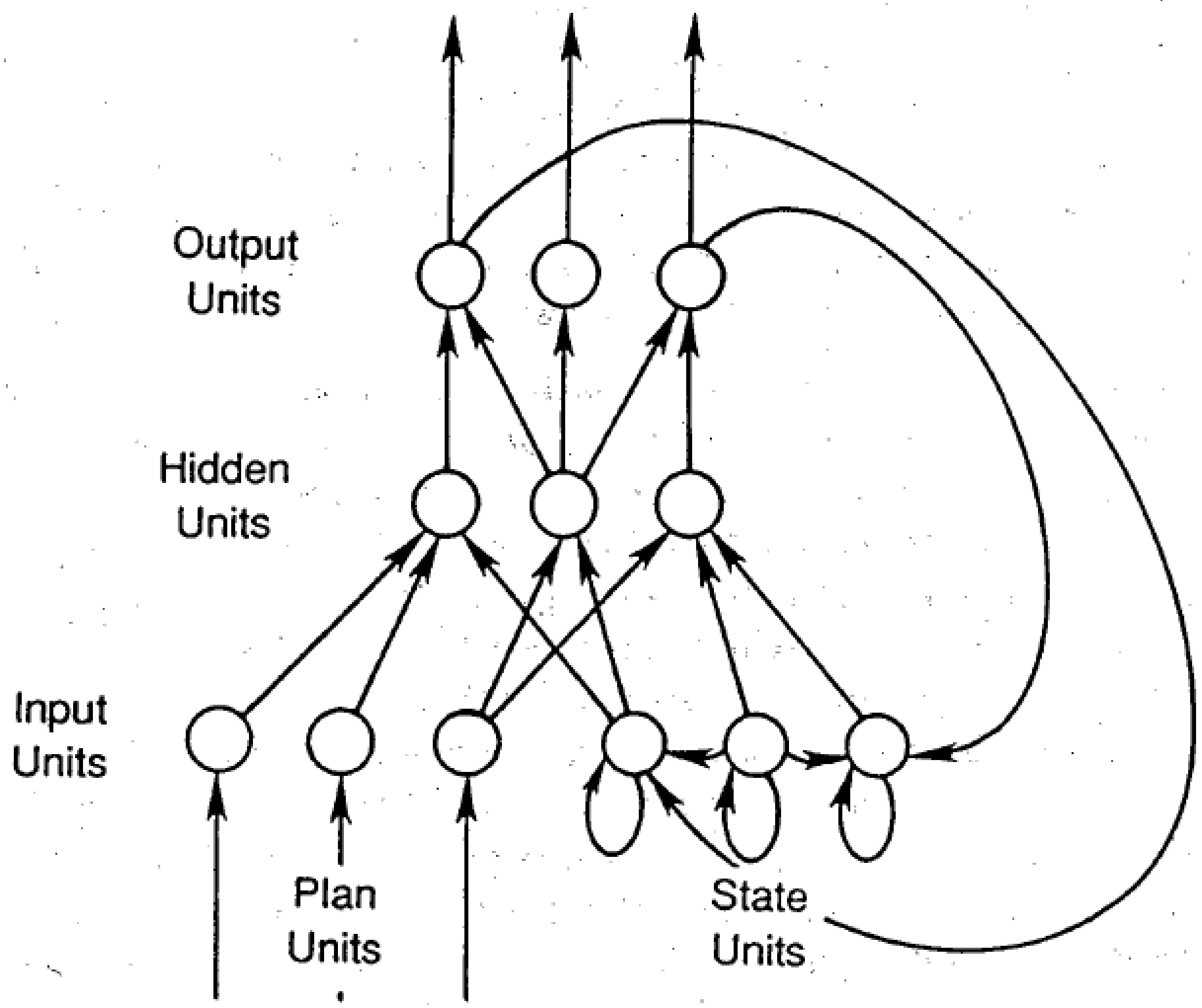
- Jordan-Network for action planning
- Plan-Units receive initial Input as a Plan
- Output-Units receive desired action
- Sequential feedback with 1:1 copied State-Units
- Advantage: Sequential knowledge not limited
- Disadvantage: Direct feedback of action values with possible error feedback into the networks

Jordan network



- Activations are copied from output layer to state layer on a one-for-one basis, with fixed weight of 1.0.
- Straight lines represent trainable connections.

Jordan network (with some more details)



Simple recurrent network (SRN)

- Problem with Time

  [0 1 1 1 0 0 0 0 0]

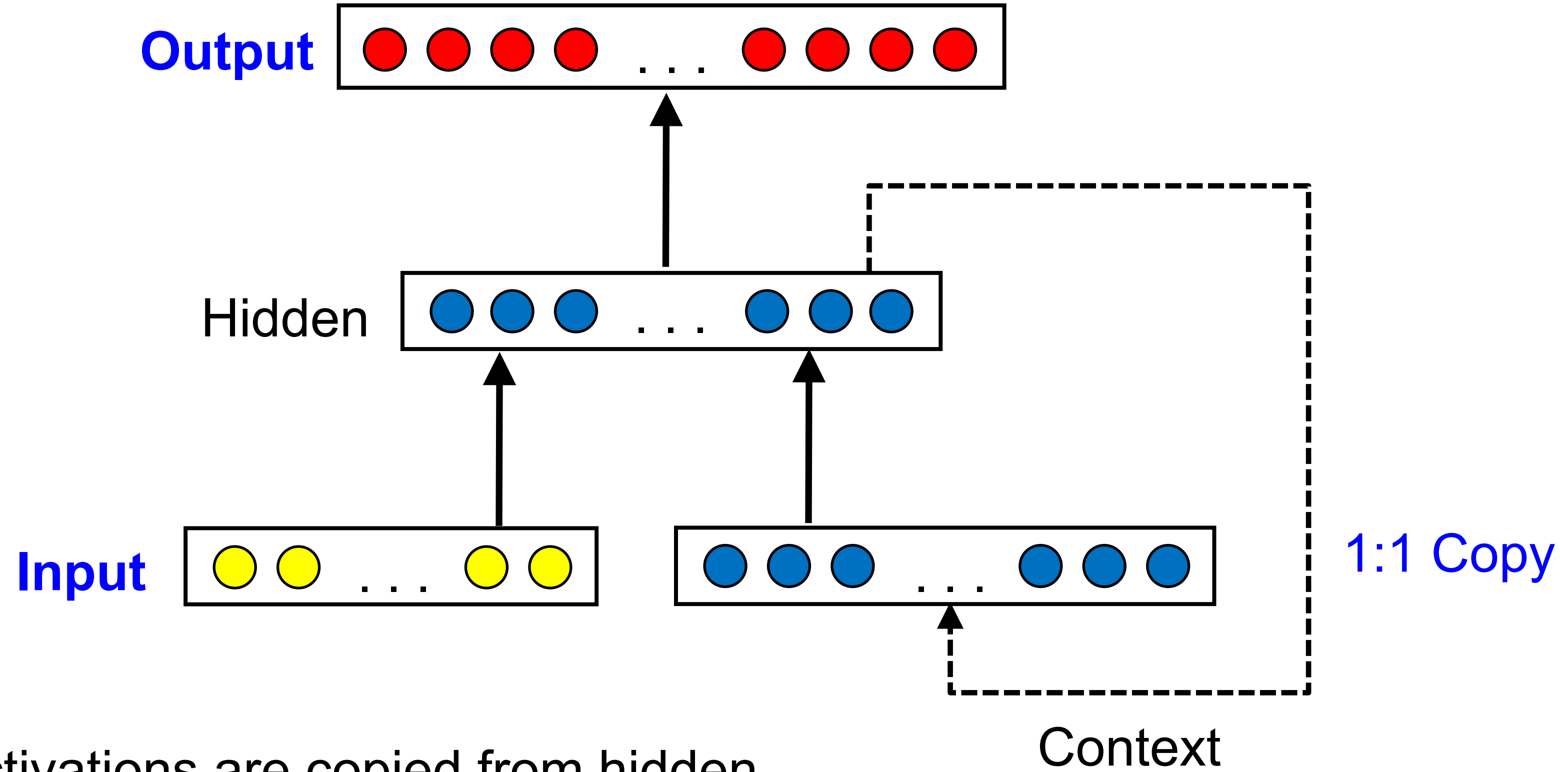
  [0 0 0 1 1 1 0 0 0]

- Two vectors appear to be instances of the ***same*** basic ***pattern***, but displaced in space
- Relative temporal structure should be ***preserved*** in the face of absolute temporal displacements
- Related to variances in vision

Simple recurrent network (SRN)

- Problem of using output-input recurrent connections in Jordan Network
- Motivation for using ***internal state*** information
- Copy the internal hidden layer for next input
- Paper: Elman J., Finding Structure in Time, Cognitive Science 14, 1990

Simple recurrent network (SRN)



- Activations are copied from hidden layer to context layer on a one-for-one basis, with fixed weight of 1.0.
- Straight lines represent trainable connections.

Example Prediction

Input: w_1 w_2 w_3 .... w_n

Output: w_2 w_3 w_4 .... w_{n+1}

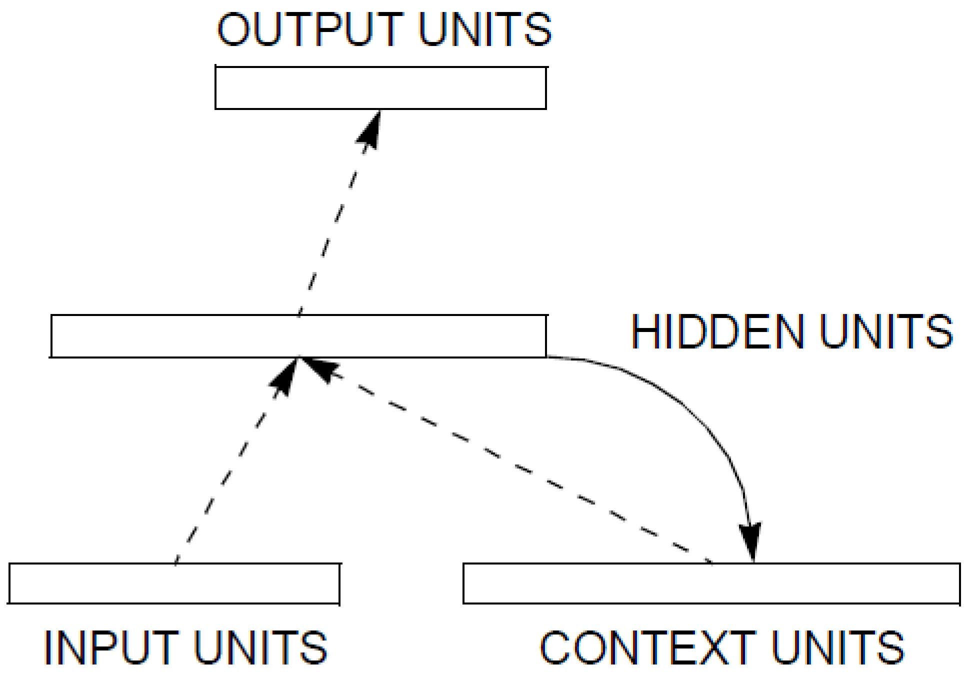
SRN as a predictor of letter sequences

- Multi-bit inputs of sequences
- 3 consonants (**b**, **d**, **g**) combined in random order to obtain 1000-letter sequence. Then each consonant replaced using rules
  - b->ba
  - d->dii
  - g->guuu
- **dbgbddg**... into **diibaguuubadiidiiguuu**

Vector definitions of alphabet

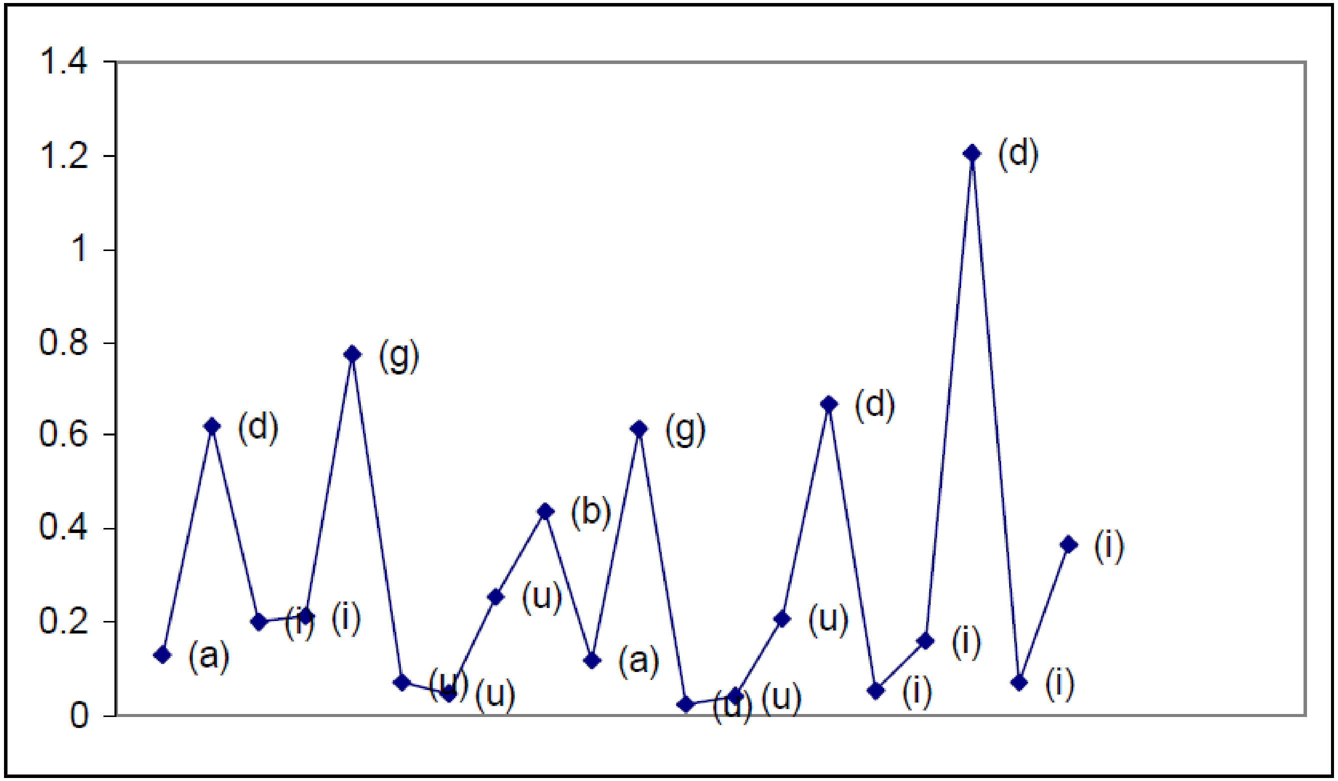
	Consonant	Vowel	Interrupted	High	Back	Voiced
b	[1	0	1	0	0	1]
d	[1	0	1	1	0	1]
g	[1	0	1	0	1	1]
a	[0	1	0	0	1	1]
i	[0	1	0	1	0	1]
u	[0	1	0	1	1	1]

SRN for letter sequences



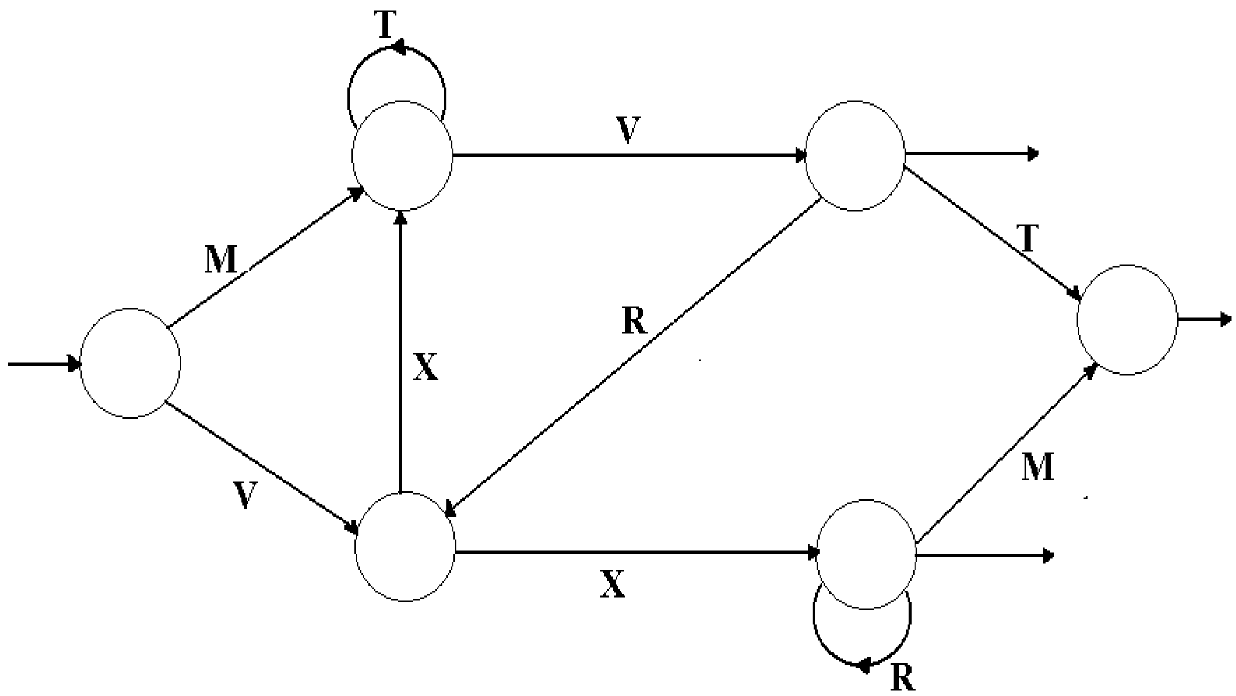
6 input units, 20 hidden units,
6 output units, and 20 context units

Root mean squared error in letter prediction task



- Labels indicate the correct output prediction at each point in time.
- Once network has consonant as input, it can predict following vowel.

Extension for SRN learning Reber grammar



- Finite state grammar for generating sequence stimuli in artificial grammar learning
- Example of implicit learning: people learn task but not rules

SRN for Reber grammar: Decide whether sequences are accepted or rejected

1. VXTTTV	1. Y
2. MVRTR	2. N
3. MVRXRM	3. Y
4. MTVT	4. Y
5. MTRVRX	5. N
6. VXRM	6. Y
7. VRVXV	7. N
8. MXRRM	8. N

[Reber 1967: “implicit learning”]

SRN for learning lexical classes from word order

- Order of words is constraint
- Can a network learn ***structure*** from order?
- Sentence generator based on categories of lexical items
- Each word represented by random 31 bit vector
- Each word represented by a different bit which is on if word present
- 27,354 word vectors in the 10,000 sentences were concatenated

Categories of lexical items

Category	Examples
NOUN-HUM	man, woman
NOUN-ANIM	cat, mouse
NOUN-INANIM	book, rock
NOUN-AGRESS	dragon, monster
NOUN-FRAG	glass, plate
NOUN-FOOD	cookie, sandwich
VERB-INTRAN	think, sleep
VERB-TRAN	see, chase
VERB-AGPA	move, break
VERB-PERCEPT	smell, see
VERB-DESTROY	break, smash
VERB-EA	eat

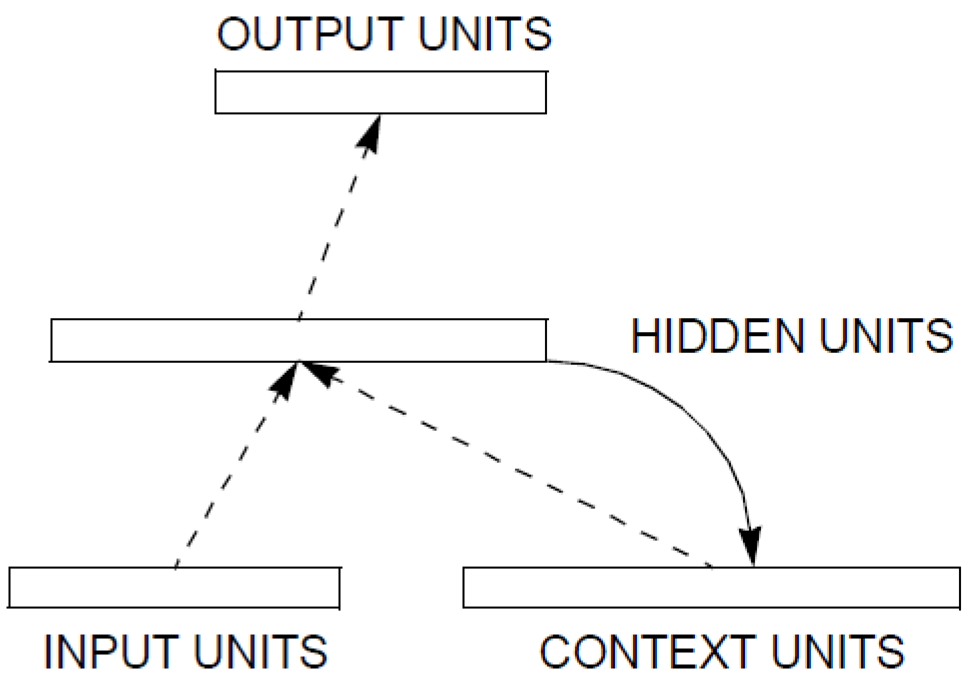
Templates for sentence generator

WORD 1	WORDS	WORD 3
NOUN-HUM	VERB-EAT	NOUN-FOOD
NOUN-HUM	VERB-PERCEPT	NOUN-INANIM
NOUN-HUM	VERB-DESTROY	NOUN-FRAG
NOUN-HUM	VERB-INTRAN	
NOUN-HUM	VERB-TRAN	NOUN-HUM
NOUN-HUM	VERB-AGPAT	NOUN-INANIM
NOUN-HUM	VERB-AGPAT	
NOUN-ANIM	VERB-EAT	NOUN-FOOD
NOUN-ANIM	VERB-TRAN	NOUN-ANIM
NOUN-ANIM	VERB-AGPAT	NOUN-INANIM
NOUN-ANIM	VERB-AGPAT	
NOUN-INANIM	VERB-AGPAT	
NOUN-AGRESS	VERB-DESTORY	NOUN-FRAG
NOUN-AGRESS	VERB-EAT	NOUN-HUM
NOUN-AGRESS	VERB-EAT	NOUN-ANIM
NOUN-AGRESS	VERB-EAT	NOUN-FOOD

Learning successive words

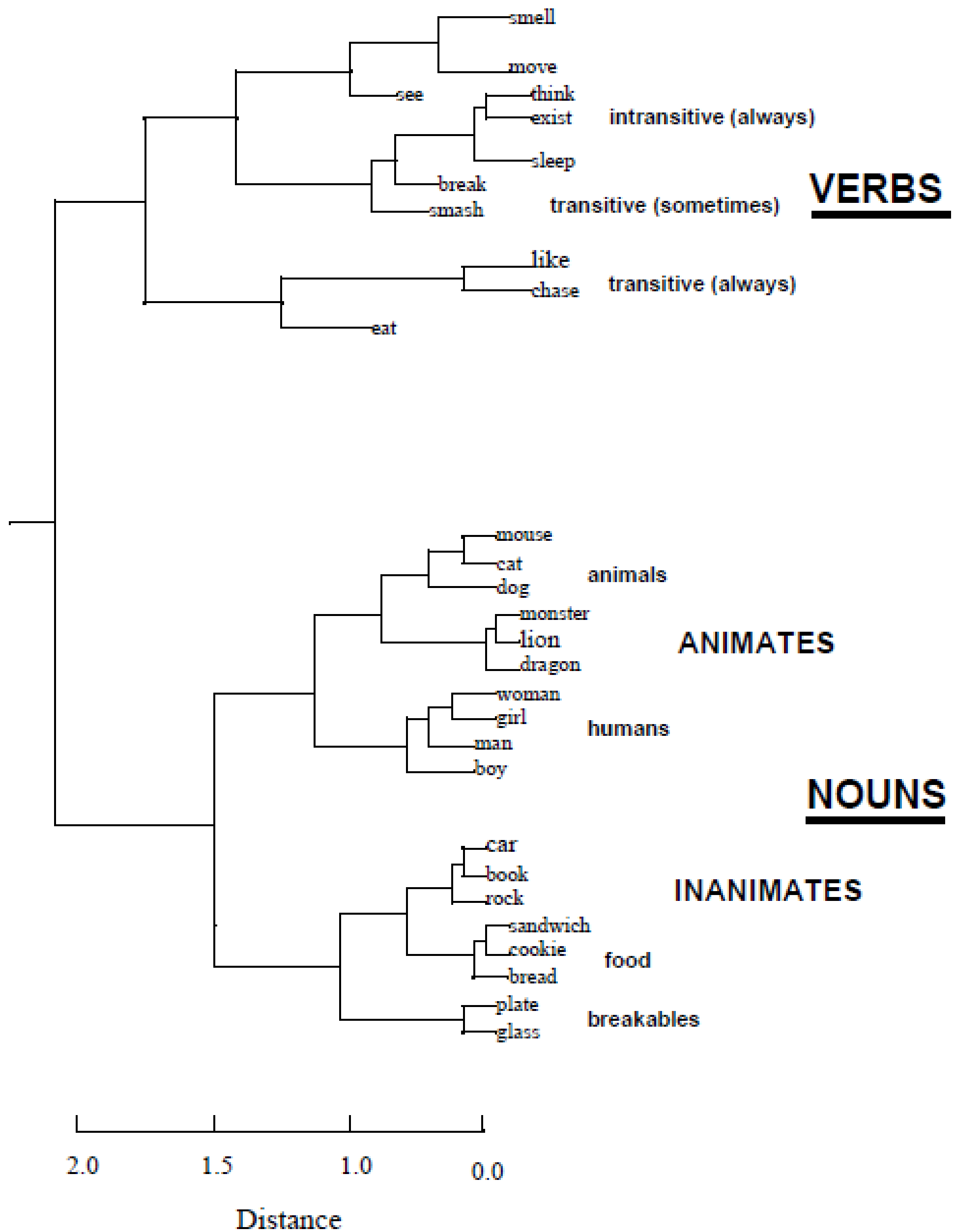
INPUT		OUTPUT	
0000000000000000000000000000010	(woman)	0000000000000000000000000010000	(smash)
0000000000000000000000000010000	(smash)	0000000000000000000001000000000	(plate)
0000000000000000000001000000000	(plate)	0000010000000000000000000000000	(cat)
0000010000000000000000000000000	(cat)	0000000000000000000100000000000	(move)
0000000000000000000100000000000	(move)	0000000000000000100000000000000	(man)
0000000000000000100000000000000	(man)	0001000000000000000000000000000	(break)
0001000000000000000000000000000	(break)	0000100000000000000000000000000	(car)
0000100000000000000000000000000	(car)	0100000000000000000000000000000	(boy)
0100000000000000000000000000000	(boy)	0000000000000000000100000000000	(move)
0000000000000000000100000000000	(move)	0000000000001000000000000000000	(girl)
0000000000001000000000000000000	(girl)	0000000000100000000000000000000	(eat)
0000000000100000000000000000000	(eat)	0010000000000000000000000000000	(bread)
0010000000000000000000000000000	(bread)	0000000010000000000000000000000	(dog)
0000000010000000000000000000000	(dog)	0000000000000000000100000000000	(move)
0000000000000000000100000000000	(move)	0000000000000000001000000000000	(mouse)
0000000000000000001000000000000	(mouse)	0000000000000000001000000000000	(mouse)
0000000000000000001000000000000	(mouse)	0000000000000000000100000000000	(move)
0000000000000000000100000000000	(move)	1000000000000000000000000000000	(book)
1000000000000000000000000000000	(book)	0000000000000001000000000000000	(lion

SRN for learning lexical classes from word order



31 input units, 150 hidden units,
31 output units, and 150 context units

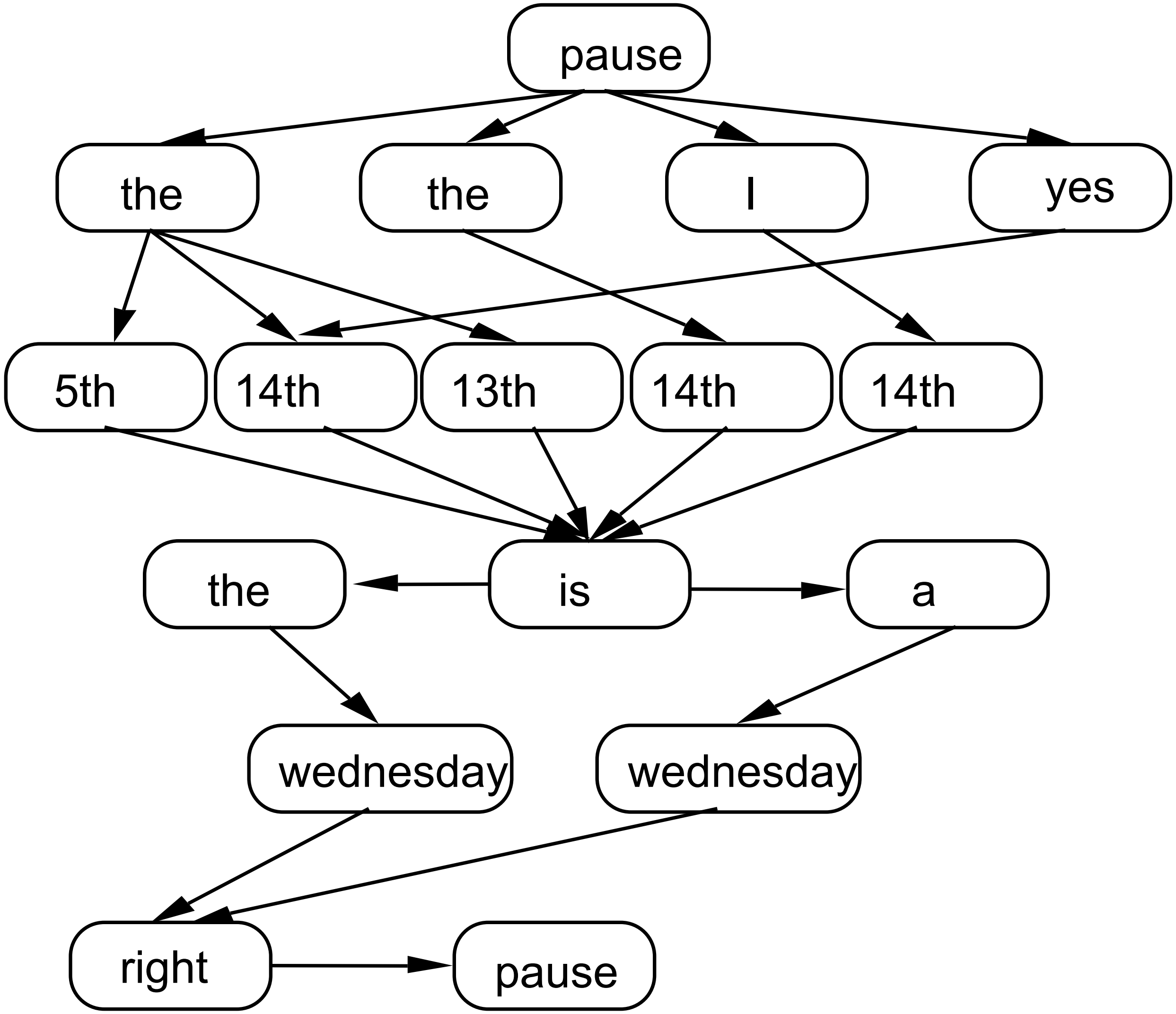
Hierarchical cluster analysis of hidden layers



SRN dealing with “noisy” sequences

- Spoken language is very noisy
- “Incorrect” grammatical constructions
- Interjections and pauses (eh, ah)
- Word repairs, phrase repairs (in - on the table)
- False starts
- Example: I would like eh to call a meeting - pause - a meeting on Wednesday no Thursday at four
- Speech recognizers are even worse

Simplified word graph from speech recognizer

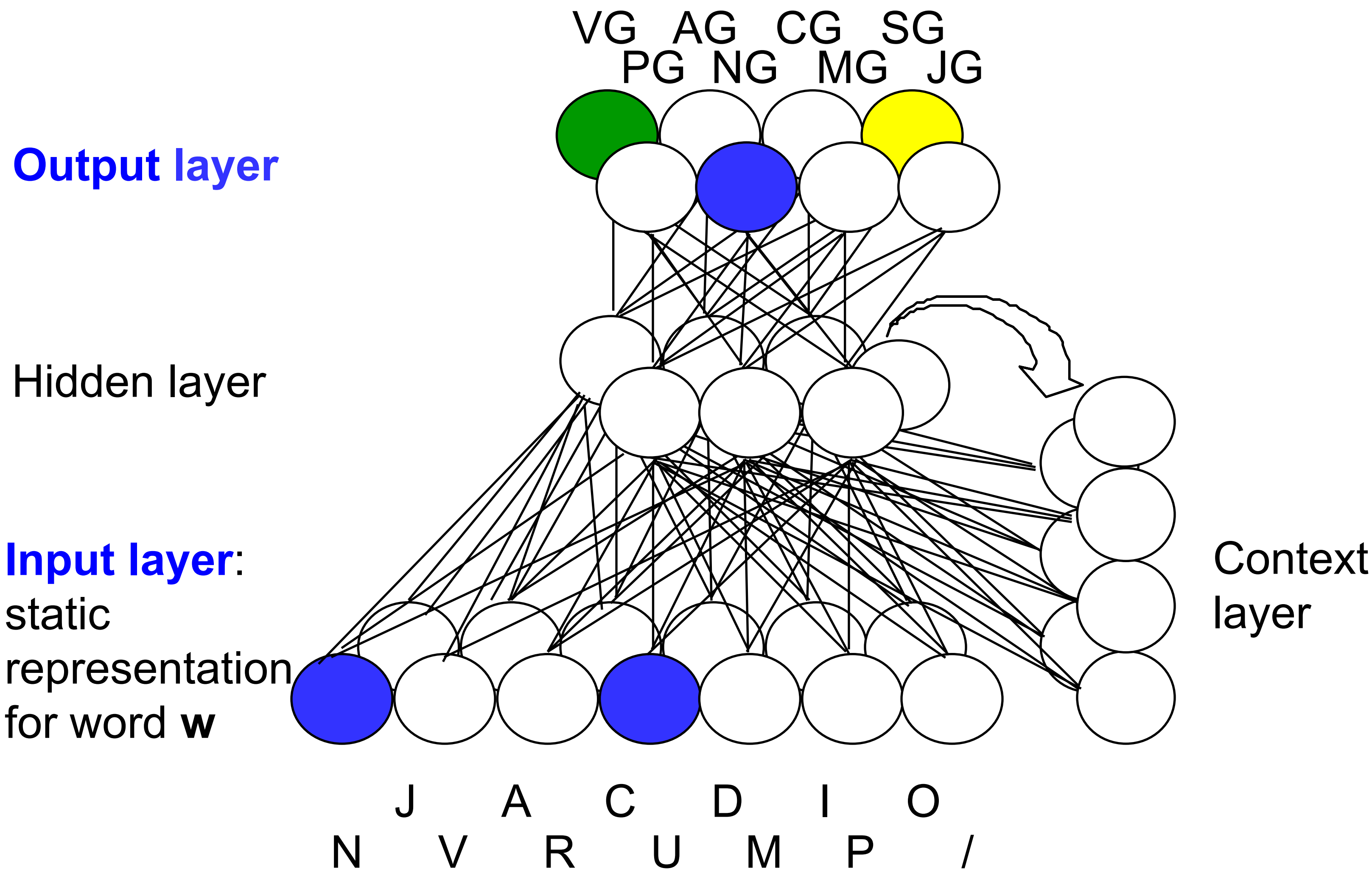


Dealing with incorrect sequences in recurrent networks

- Basic syntactic categories
  - noun (N), verb (V), preposition (R), pronoun (U), numeral (M), participle (P), pause (/), adjective (J), adverb (A), conjunction (C), determiner (D), interjection (I), other (O)
- Abstract syntactic categories
  - noun group (NG), verb group (VG), adverbial group (AG), prepositional group (PG), conjunction group (CG), modus group (MG), special group (SG), interjection group (IG)
- Goal: Learning a robust analysis at the level of phrasal syntactic categories
- **Example**:

I would suggest eh a meeting on Friday
U V V I D VN R N
NG VG IG NG PG

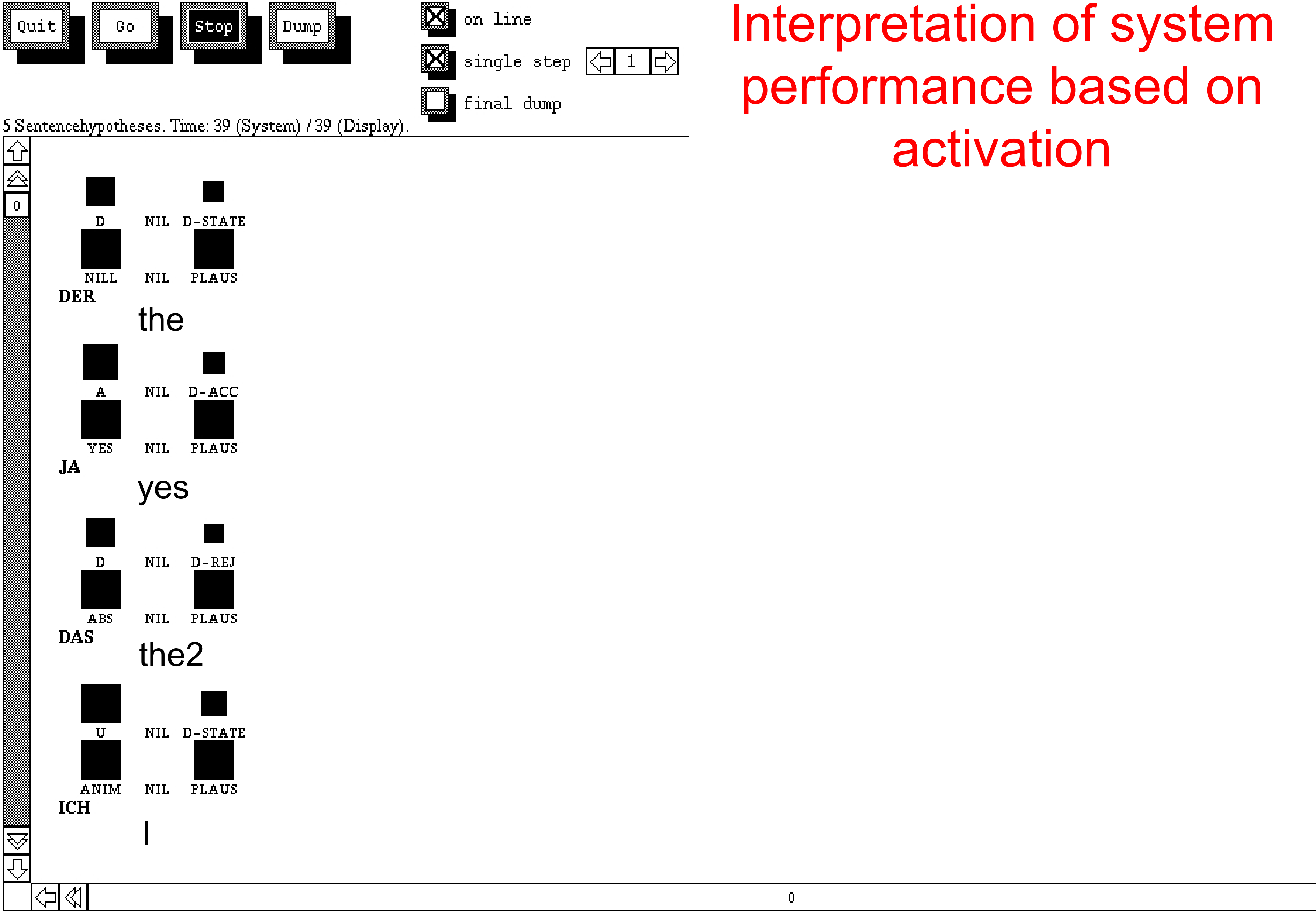
Dealing with incorrect sequences in recurrent networks



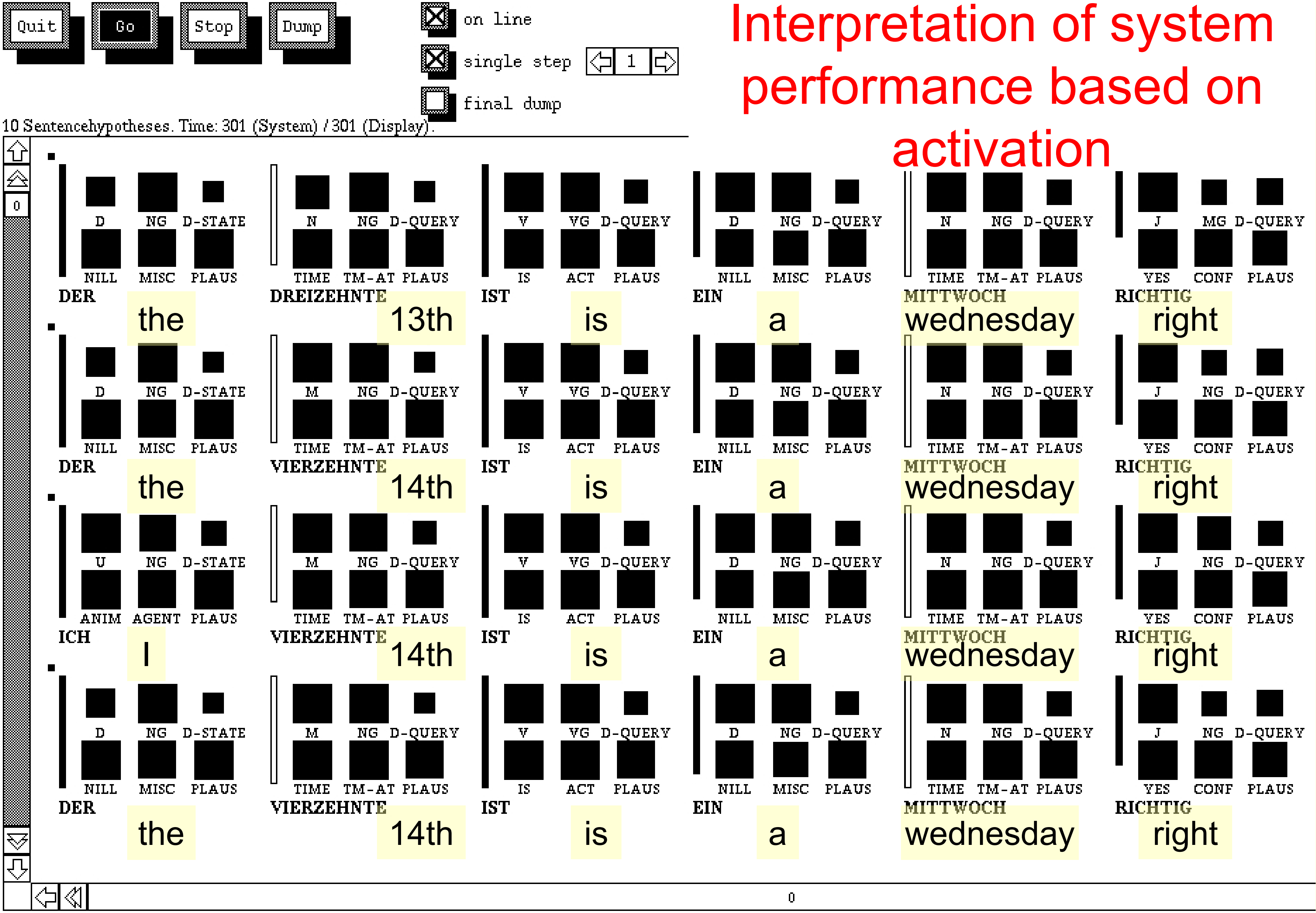
Interpretation of system performance based on activation of networks

- Building larger architectures based on simple recurrent networks
- Detailed interpretation of learning capabilities
- Detailed interpretation of performance
- Example systems: SCREEN (Wermter, Weber)

Interpretation of system performance based on activation



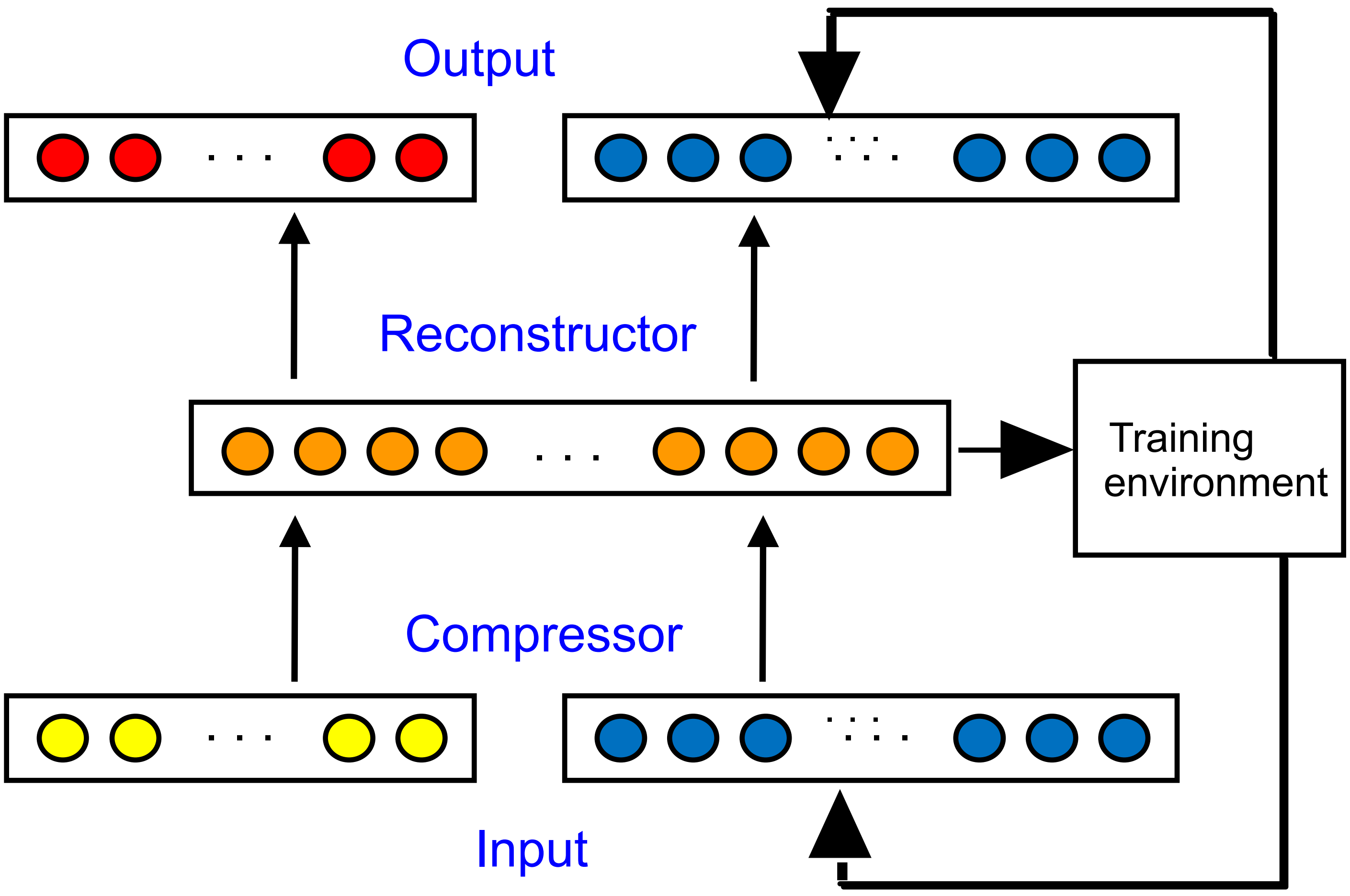
Interpretation of system performance based on activation



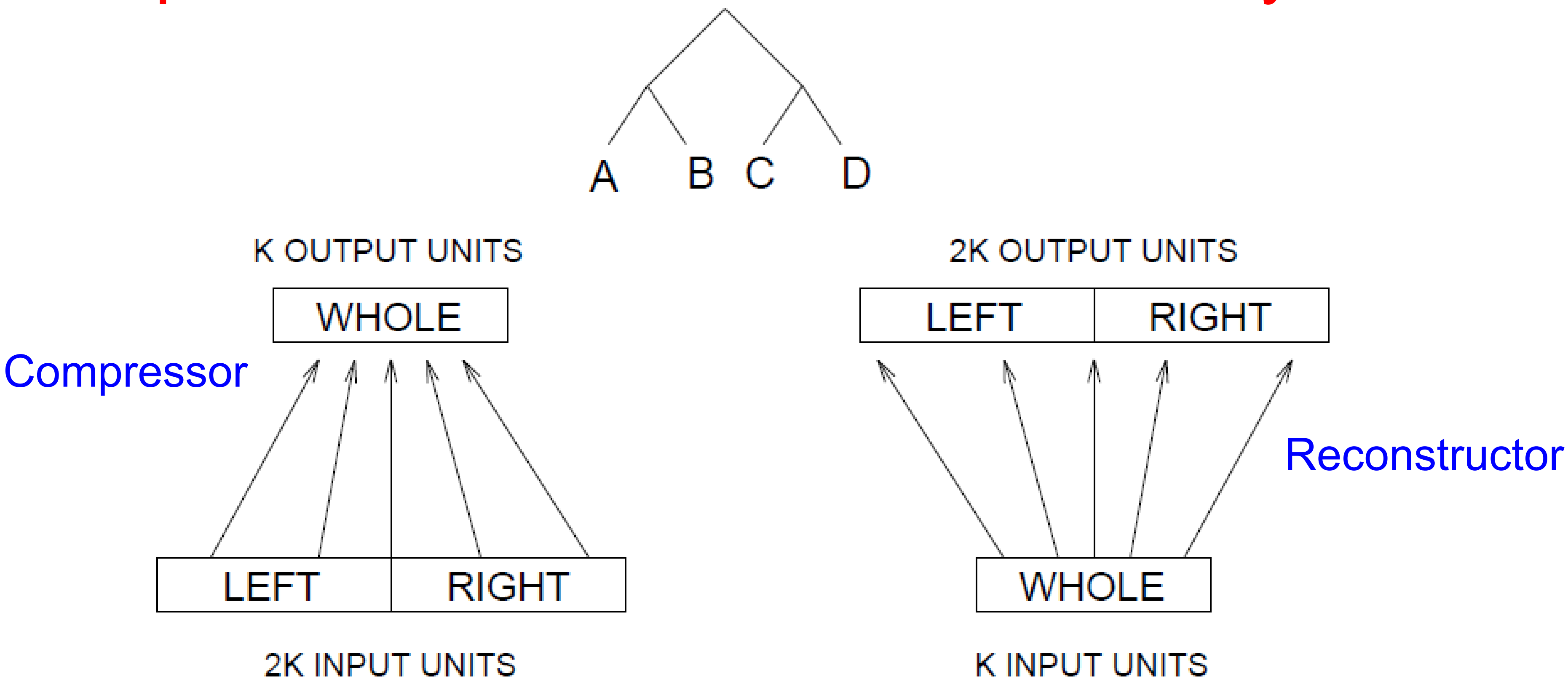
RAAM: Recursive Autoassociative Memory

- Feedforward network with memory
- Compressor maps input to internal reduced representation
- Reconstructor decodes the internal representation into output
- Can represent tree-like structures:
- (det (adj noun)) trained as
  - (adj noun)-> R(adj noun) -> (adj* noun*)
  - (det R(adj noun)) -> R(det adj noun) -> (det* R(adj noun)*)

RAAM (Pollack)
Recursive autoassociative memory



RAAM: Recursive autoassociative memory
Compressor and Reconstructor of binary trees



input pattern		*hidden pattern*		*output pattern*
$(A \quad B)$	$\rightarrow$	$R_1(t)$	$\rightarrow$	$(A'(t) \quad B'(t))$
$(C \quad D)$	$\rightarrow$	$R_2(t)$	$\rightarrow$	$(C'(t) \quad D'(t))$
$(R_1(t) \quad R_2(t))$	$\rightarrow$	$R_3(t)$	$\rightarrow$	$(R_1(t)' \quad R_2(t)')$

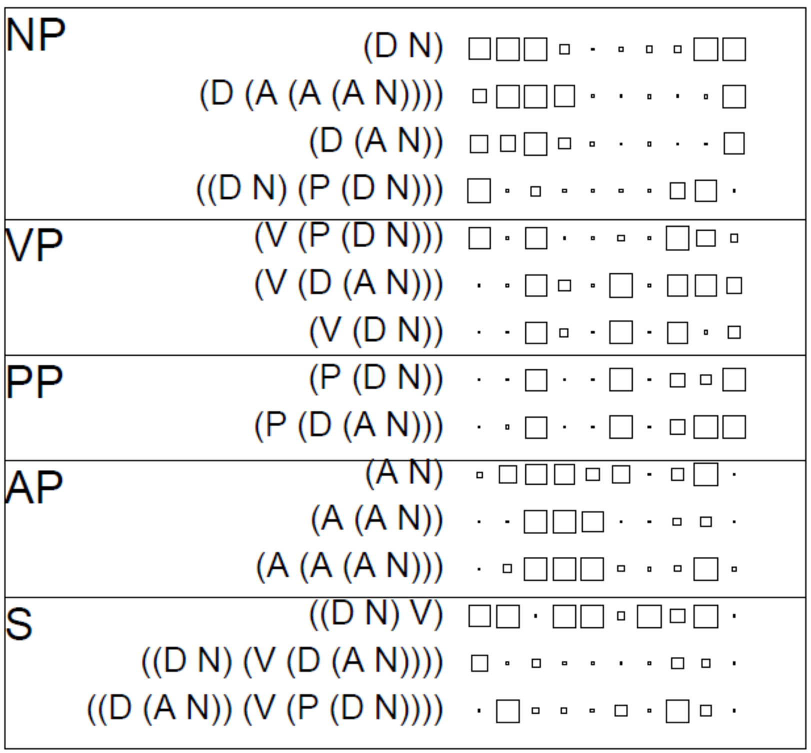
RAAM: Recursive autoassociative memory
Learning representations for syntactic trees

Grammar

S -> NP VP | NP V
NP -> D AP | D N | NP PP
PP -> P NP
VP -> V NP | V PP
AP -> A AP | A N

Sequences

(D (A (A (A N))))
((D N)(P (D N)))
(V (D N))
(P (D (A N)))
((D N) V)
((D N) (V (D (A N))))
((D (A N)) (V (P (D N))))

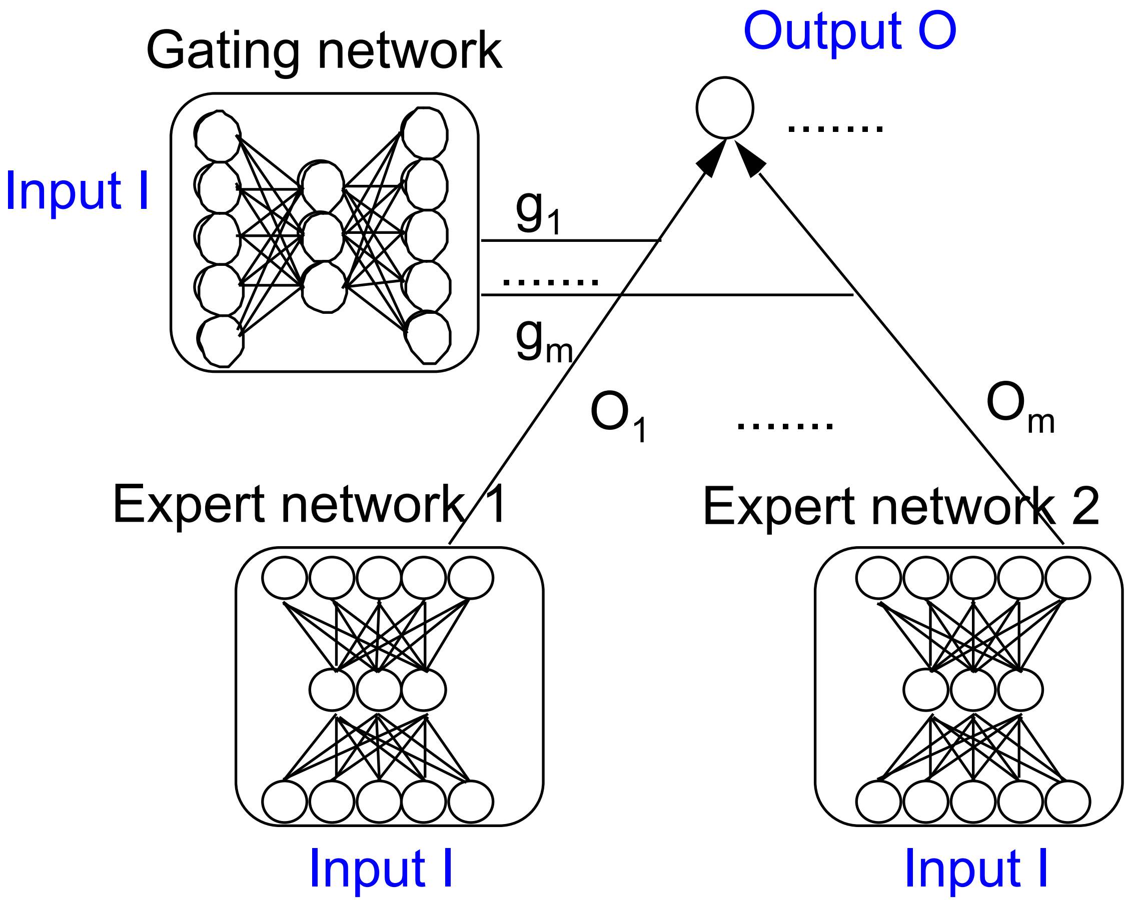


Learned Internal Representations

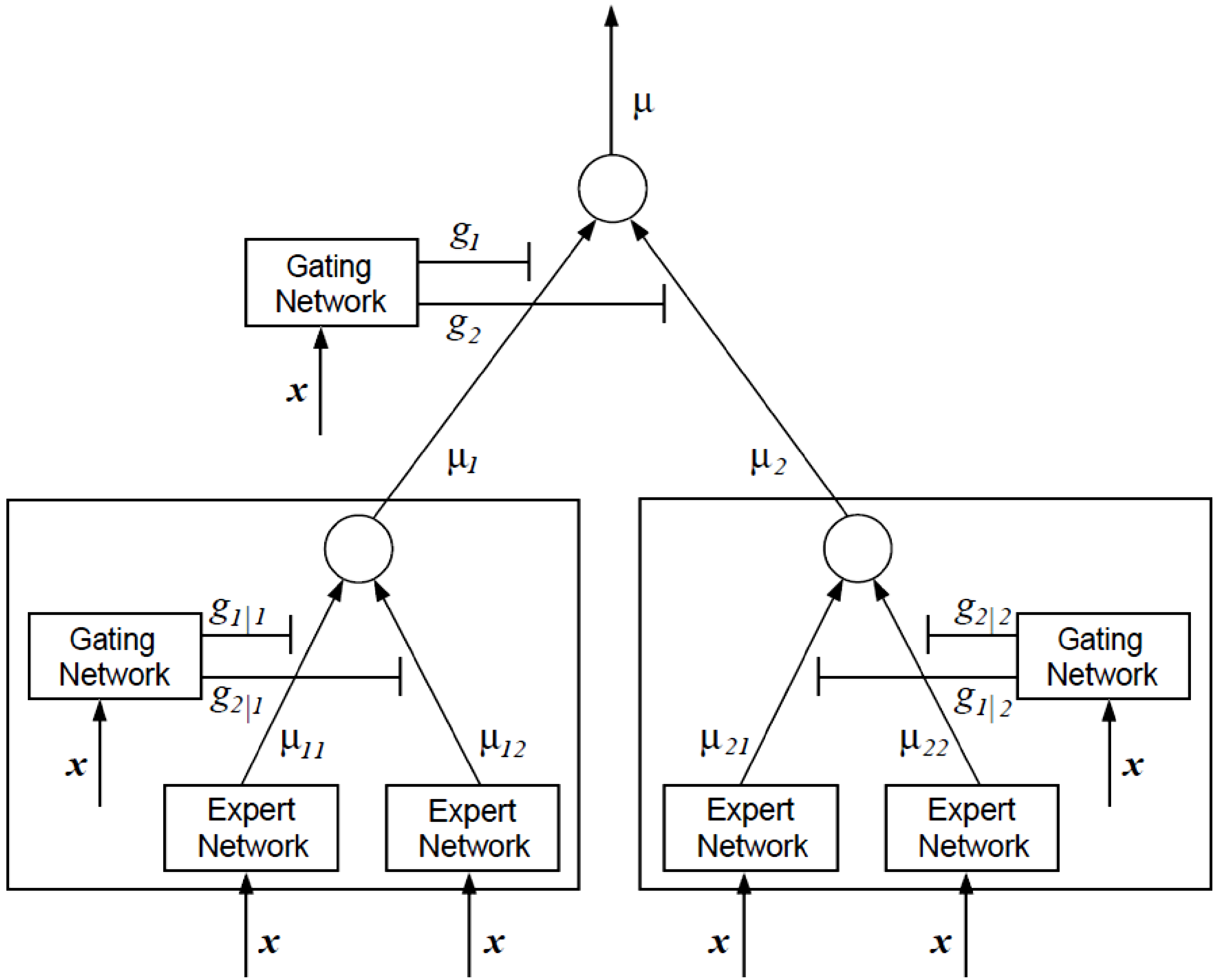
Modular mixture of experts model (Jacobs et al)

- Assumption: data for a complex system are generated by different processes
- Architecture is split into ***separate expert*** networks
- For each input each expert network computes a new output
- Selection of best expert output by gating network
- Related to sigma pi (***multiplicative***) connections

Gating: combining expert networks



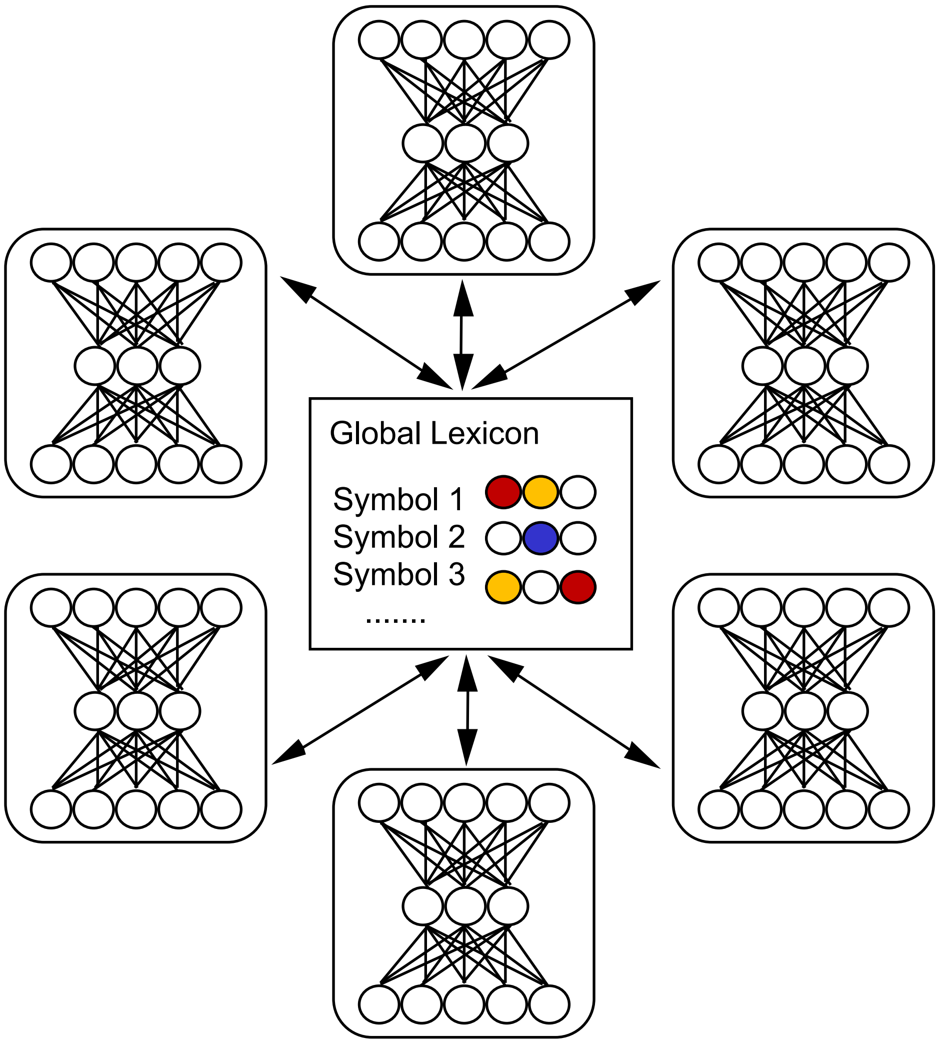
Hierarchical expert networks (Jordan, Jacobs)



Symbolic recirculation of connectionist lexicon representations

- Problem of automatic acquisition of feature representation
- Dynamic determination of input representations based on context from one or several connectionist modules
- Central connectionist lexicon
- Update of the lexicon representations based on the context
- Initial concept representations are changed over time by collecting individual changes of each subnetwork

Symbolic recirculation in connectionist structure architecture (Miikkulainen)



Using a validation set

- Divide the total dataset into three subsets:

- ***Training data*** is used for learning the parameters of the model.
- ***Validation data*** is not used for learning but is used for deciding what type of model best and when to stop training.
- ***Test data*** is used to get a final, unbiased estimate of how well the network works. Expect this estimate to be worse than on the training data.

Some Pointers to Neural Network Simulators

- Useful for demonstration and analysis of the networks
- Simulating activities range from single neurons to large neural network models
- Basic mechanisms are built in
- Some have functions for visualisation:
  - Physical architecture of the network
  - Weights and neuron activities
  - Training process

Theano

- Python lib to create, optimize & evaluate mathematical expressions
- Uses symbolic data representation
- Compile mathematical expressions into executable functions
  - Automatically build symbolic graphs for compute gradients
  - Can optimize mathematical functions automatically
- Works with GPU(CUDA) and CPU
- Used for Machine Learning
  - http://deeplearning.net/tutorial/contents.html
- Documentation
  - http://deeplearning.net/software/theano/theano.pdf

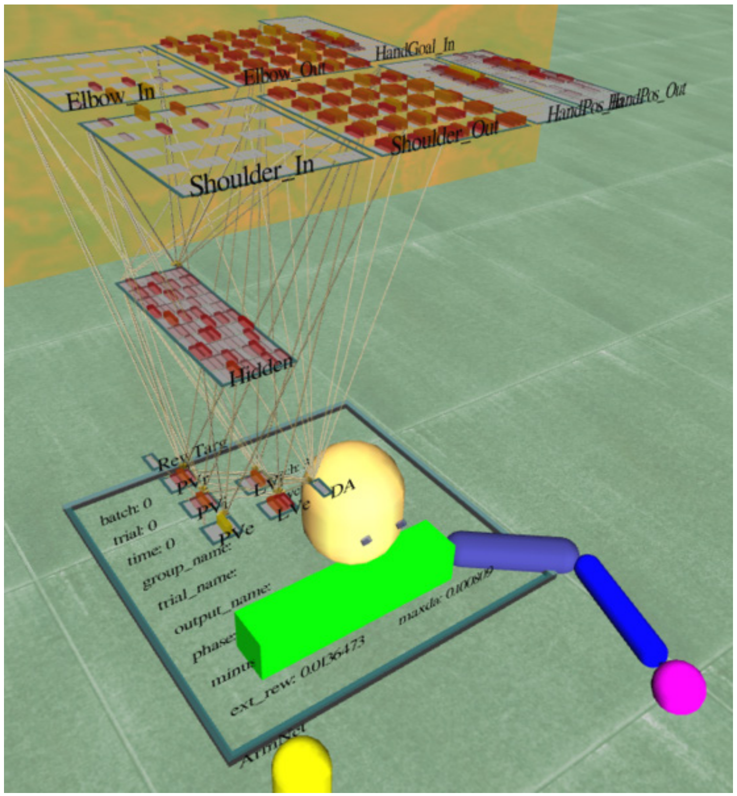
```python
import theano

# declare symbolic variable
a = theano.tensor.vector("a")

# build symbolic expression
b = a + a**10

# compile function
f = theano.function([a], b)

# prints `array([0,2,1026])`
print f([0,1,2])
```

ANN Simulator - Independent Software (Emergent, SNNS...)

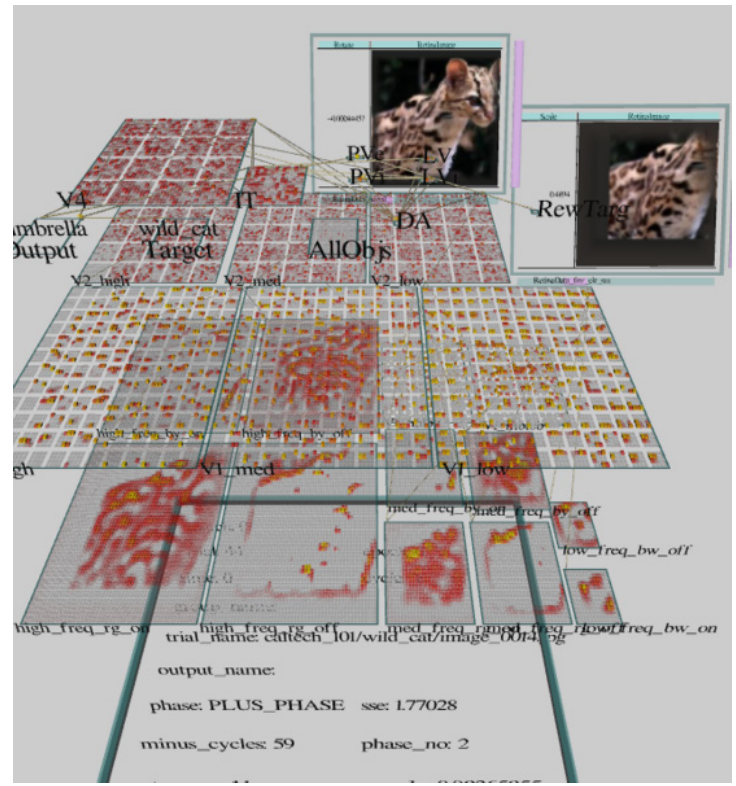
- Create models with established back-prop, self-organizing and other modules
- Parameters/training sets can be easily adjusted for simulations
- Emergent supports modelling of rigid body physics with NN applications (neural robotics etc.)
- Advantages:
  - Good for demonstration; fast development
  - Test of NN models used for specific applications with sampled data
- Limitations:
  - Algorithms need to be modified
  - Effort to develop tailored novel neural network models

ANN Simulator - Independent Software
Emergent – sophisticated GUI

- http://grey.colorado.edu/emergent/index.php/Main_Page



Robotics simulations
with rigid body physics



Advanced image processing in a
model of the human visual system

Programming Tool with NN Support (Pybrain)

- OOP Library in Python
  - Easy to implement own architectures with basic components (units, connections, etc.)
- Limited support for extension of own algorithms or components. (e.g. RNNPB).
- **Example**: training a NN for the XOR problem

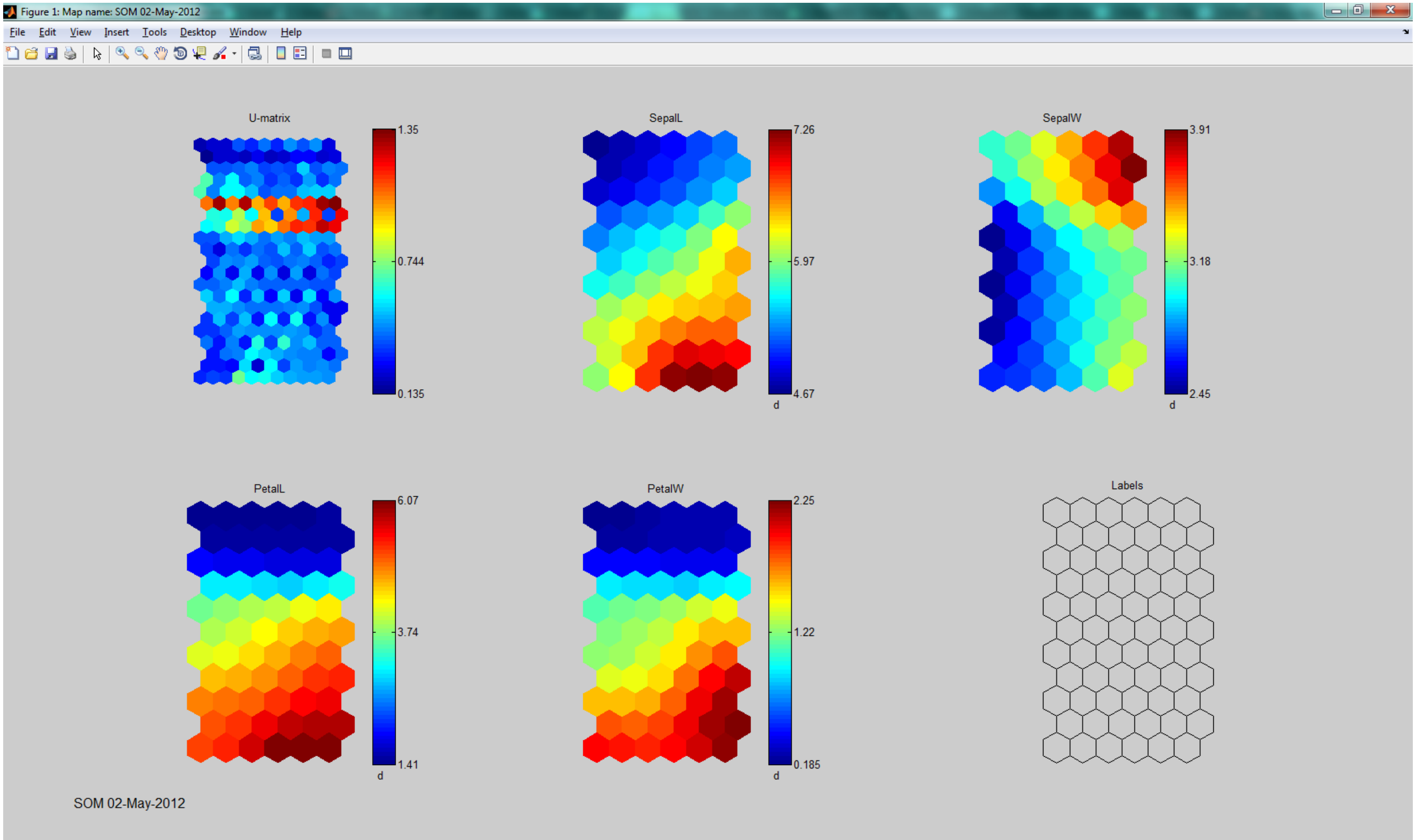
```
net = buildNetwork(2,2,1)
ds = SupervisedDataSet(2, 1)
ds.addSample((0, 0), (0,))
 ds.addSample((0, 1), (1,))
 ds.addSample((1, 0), (1,))
ds.addSample((1, 1), (0,))
trainer = BackpropTrainer(net,ds)
trainer.trainEpochs(500)
```

[http://pybrain.org]

Matlab (Artificial) Neural Network toolbox

- A toolbox integrated into Matlab for implementing, visualizing, and simulating neural networks
- Advantages:
  - Easily works with other Matlab toolboxes (e.g. Image Pre-processing)
  - Fast experiments: compare neural networks methods and other Matlab built-in methods
  - Visualization for the network
- Disadvantages:
  - Not easy to modify existing modules and examine/reuse the internal values

SOM Toolbox for Matlab



Summary and Further Reading

- Elman J. Finding Structure in Time. Cognitive Science 14, 179-211, 1990

- Chapter 2 on “Networks with Adaptive State Transitions” in Kolen and Kremer: A field guide to dynamical neural networks, 2001

- Optional research papers at Knowledge Technology website http://www.informatik.uni-hamburg.de/WTM/