Constant-Time Probabilistic Learning & Inference in Hierarchical Sparse Distributed Representations

Rod Rinkus
Neurithmic Systems

Neuro-Inspired Computational Elements Workshop
Sandia Labs, Albuquerque
Feb. 25, 2013

Massive Increase in Computational Power is Needed

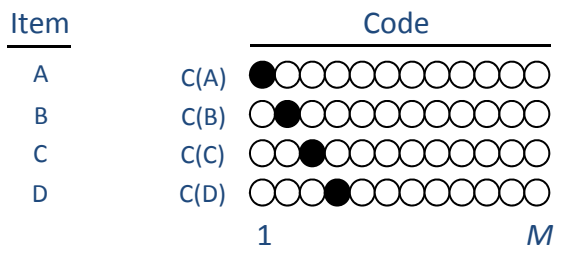
- Moore's Law is running out: Hitting the physical limit of packing conventional computing elements on chips.
- Problem size is growing at an exponential rate
  - e.g., inference over tera/peta-byte databases.
- $\rightarrow$ *computational power* must also scale *exponentially*.
- Two possible sources for computational power increase:
  - Hardware
  - Software
- New hardware technologies , e.g., memristors, graphenes, spintronics, 3D packing, etc., may bring several orders of mag. Improvement.

Massive Computing Power Increase via Software

- The greatest gains in computing power will result from a *software* change.
  - By changing from *localist* to *sparse distributed representations* (SDR).
- Currently, virtually all data is represented using localist representations.
- Reasons for SDR's increased power
  1. SDR allows a population of *M* representational units to represent an *>>M* items (hypotheses).
  2. SDR allows the likelihoods of *ALL* stored hypotheses to be computed with:
     - a single pass over the model's components (weights, units).
     - ***Not***, with a single pass over the stored hypotheses.
- Thus, SDR admits a *constant time* algorithm for computing the MLH, i.e., finding the closest matching item in memory.
- This in turn allows constant time storage (learning) of new items.
- I'll describe an algorithm that has both these properties.

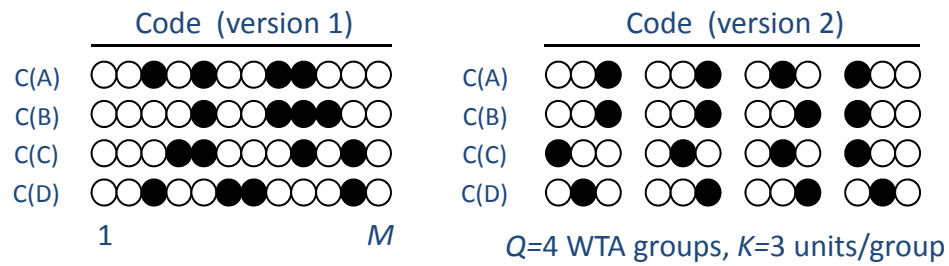
Crucial Advantages of SDR over Localism

Localist Representation

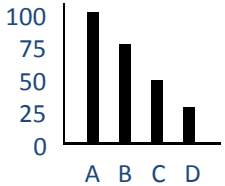


- Can only represent *M* items
- CANNOT represent similarities by code intersections.
- Activating a *localist* code of any single input leaves all other stored codes completely *inactive*, regardless of how similar or relevant their represented inputs may be.

SDR



- Can represent $\gg M$ items. The # of unique codes is ${}_{M}C_{Q}$ (Version 1) & K^{Q} (Version 2).
- CAN represent similarity directly by size of code intersections.
- Define the *activation strength* of a code as the fraction of its units that are active.
- If C(A) becomes active, C(B), C(C), and C(D) also *simultaneously* become *partially active* in proportion to their intersections with C(A), *without any extra computational cycles.*

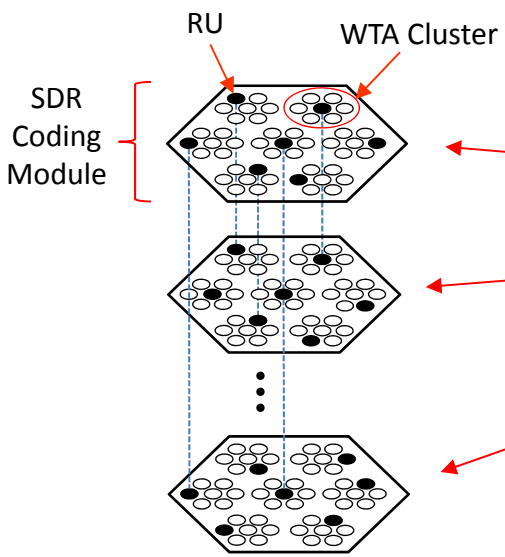


- ***If*** code intersection size ~ input similarity, then when hypothesis A is active, ALL stored hypotheses are simultaneously active in descending order of similarity (likelihood, relevance).

Terminology Clarification: Two senses of "Sparse Representation" in Literature

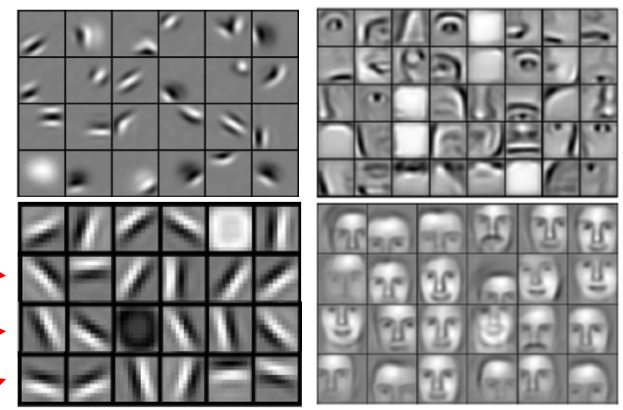
Sparse Distributed Representation

- Every represented item, even at the lowest level:
  - is represented by a set of units.
  - Those sets can overlap.



Sparse Basis

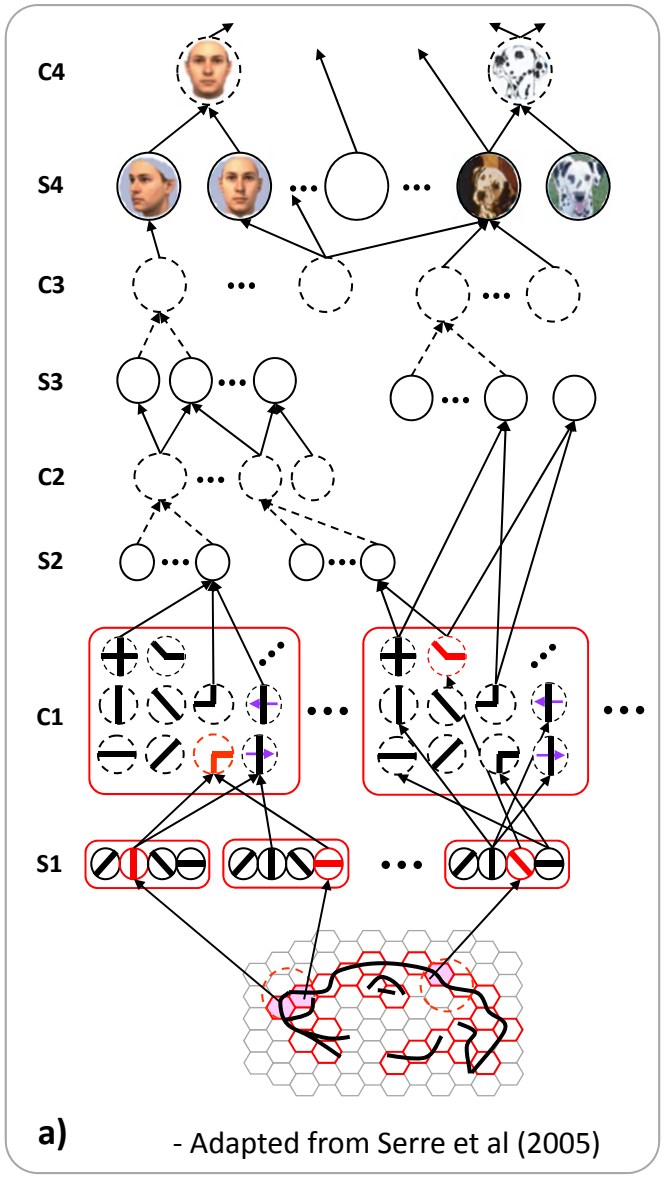
- The number of features needed to describe an input space is small compared to the number of all possible input patterns.
- Olshausen & Field (O&F) sense, "sparse overcomplete basis"

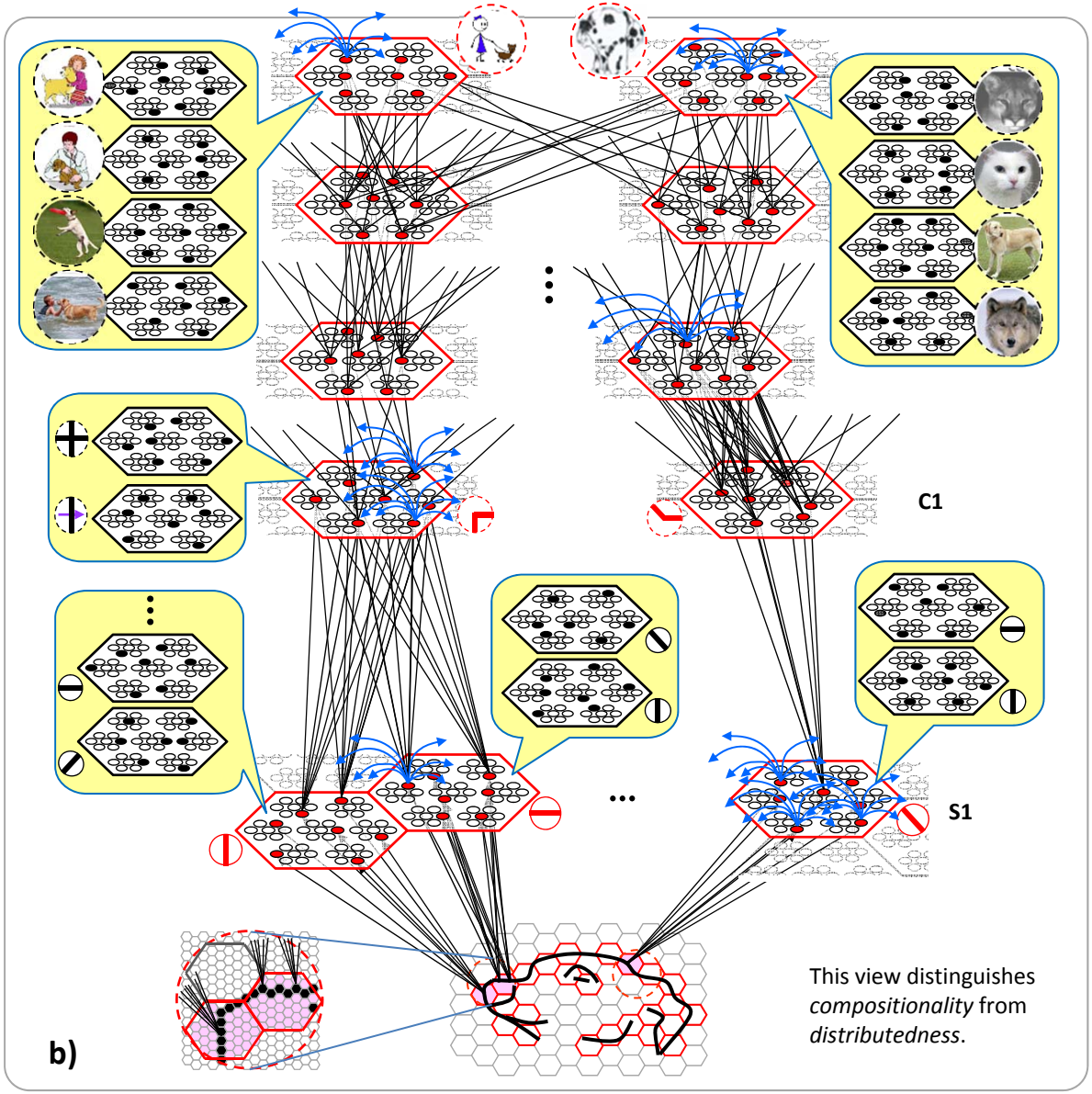


- From Lee, Grosse et al (2011)

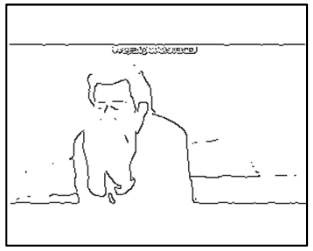
- All papers that I've seen that describe learning of sparse bases use localist representations; one unit per feature.

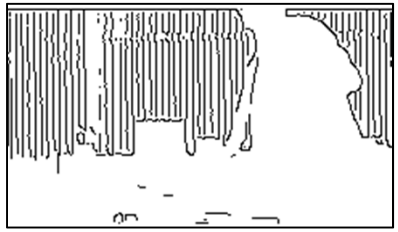
Localist vs. SDR-based Hierarchical Representations

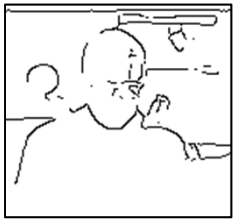




Application: Understanding Video







[redacted]

[redacted]

TEMECOR Algorithm: Constant Time Probabilistic Inference

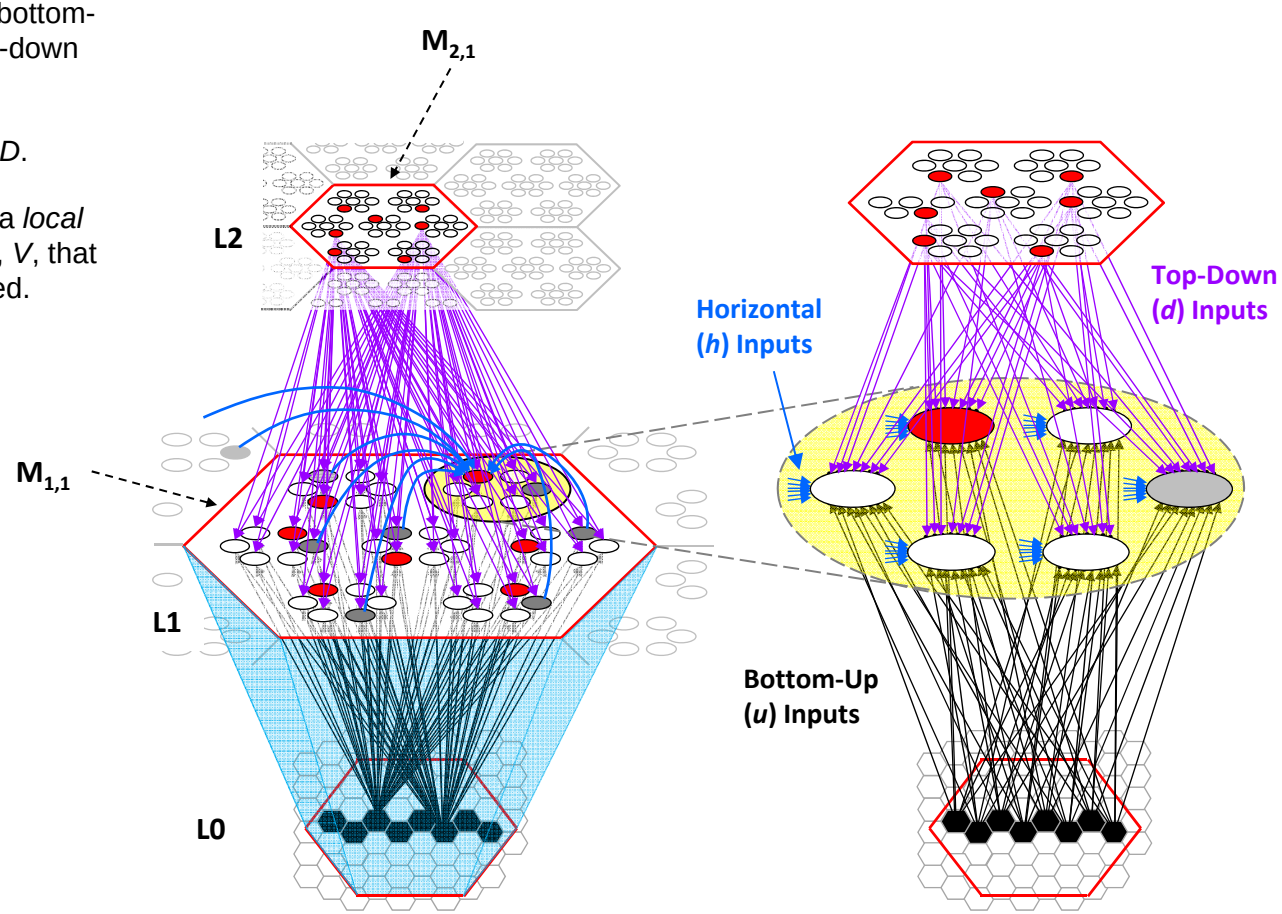
Steps 1 - 2.5

1. Individual neurons sum their bottom-up (*u*), horizontal (*h*), and top-down (*d*) inputs.

2. Normalize them to *U*, *H*, and *D*.

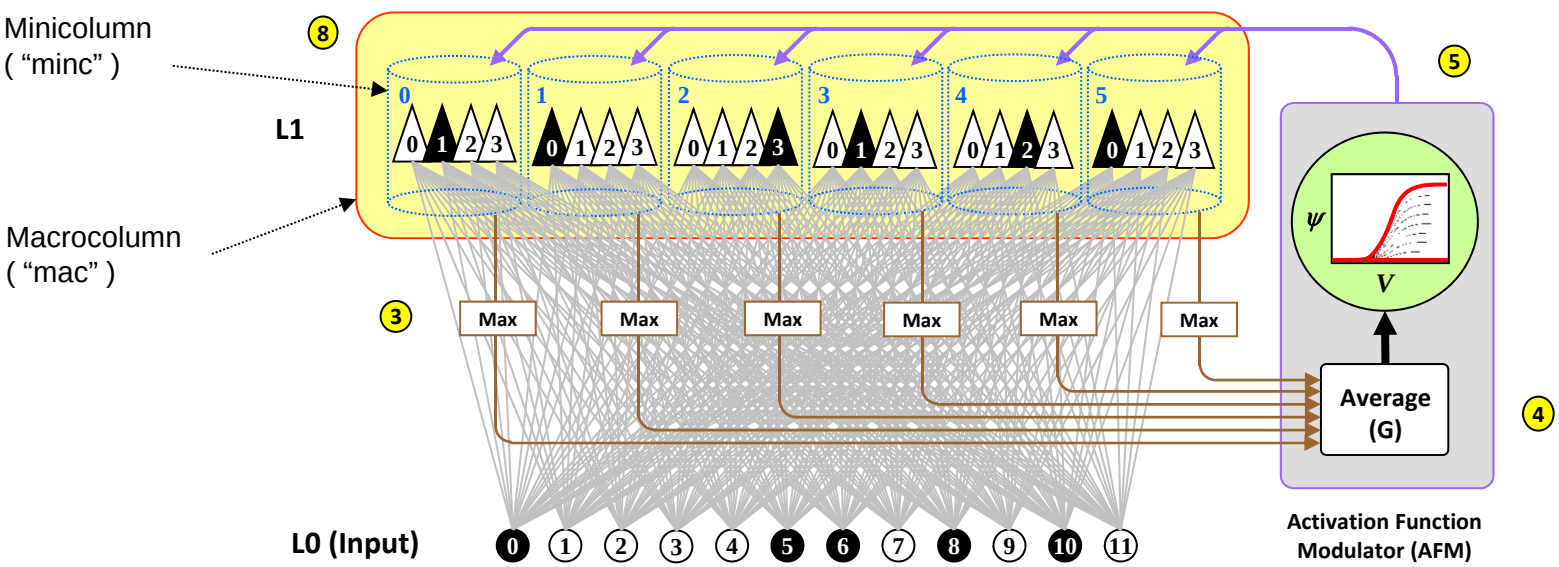
2.5. Multiply *U*, *H* and *D*, yielding a *local* degree of support (evidence), *V*, that the neuron should be activated.

TEMECOR: Temporal Episodic Memory via Combinatorial Representations (1996)

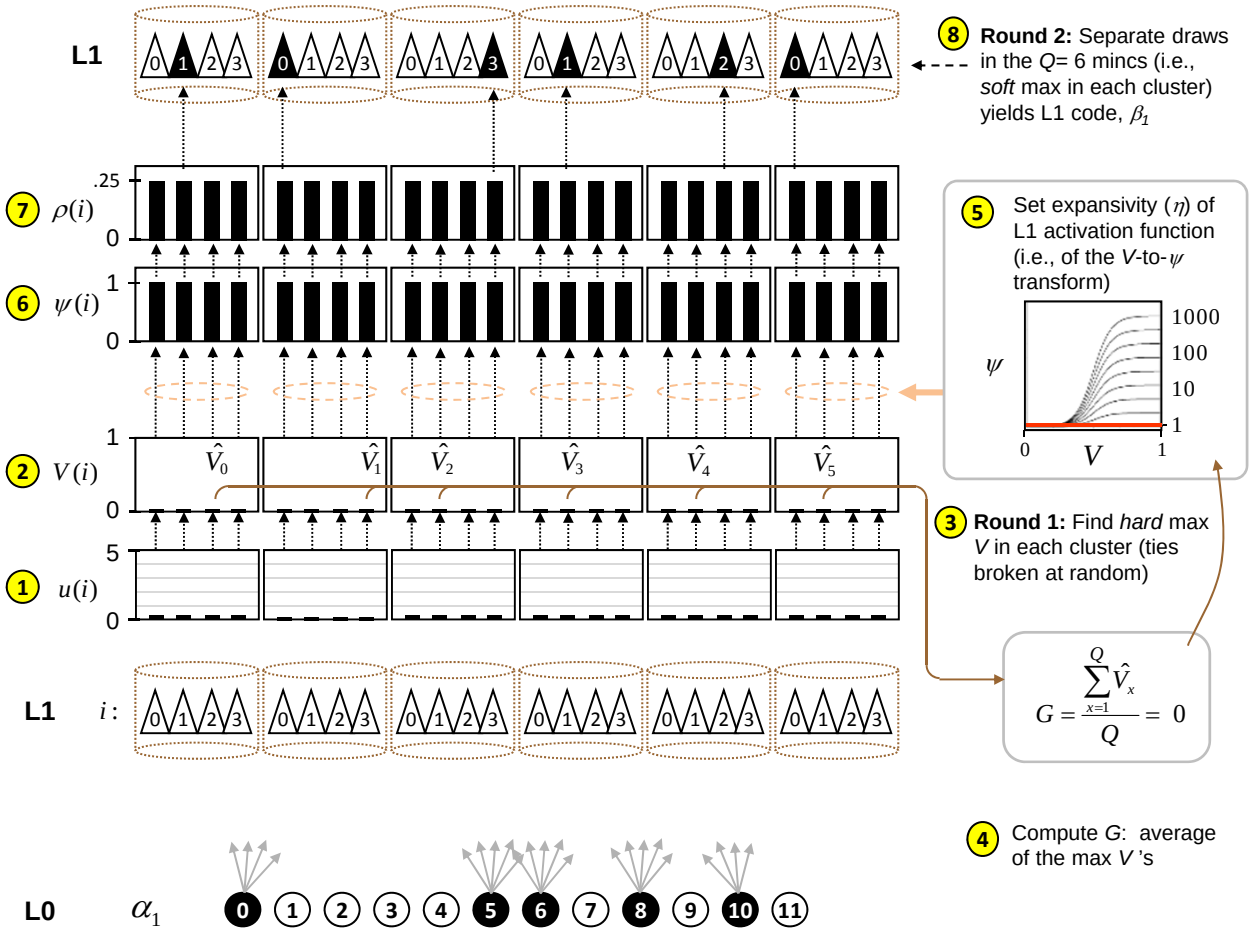


TEMECOR Algorithm – Steps 3-8

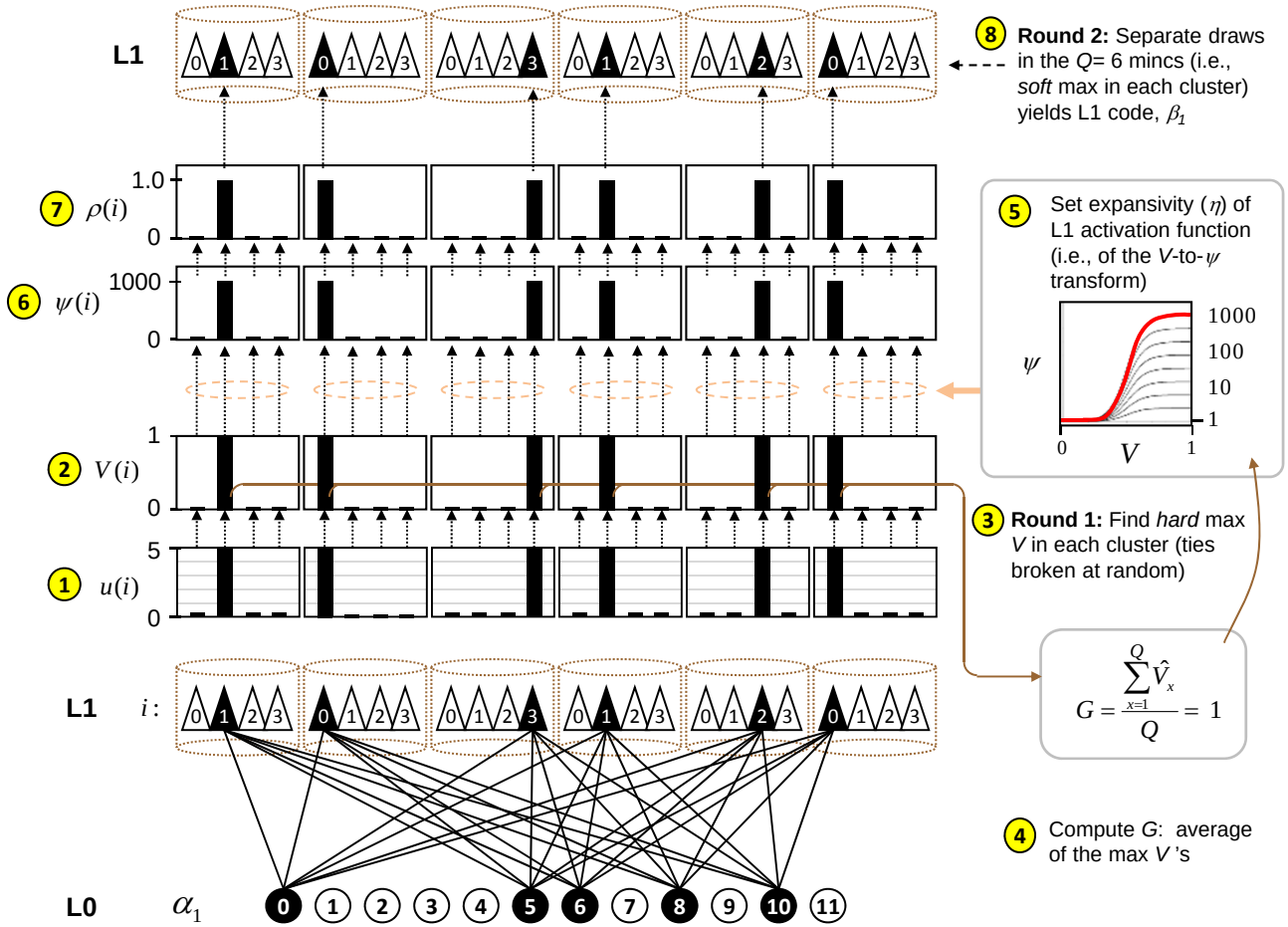
3. Neuron with max *V* in each minc wins (1st round of competition, hard max)
4. Compute the average, *G*, of the max *V*'s over all *Q* mincs in a mac.
   - *G* is a measure of the *familiarity* of total input to the mac.
5. Modulate the units' I/O functions so that:
   - as *G* goes to 1, the I/O function becomes an increasingly expansive nonlinearity
   - as *G* goes to 0, it becomes maximally compressive, i.e. the constant function.
6. Push the *V* 's through modulated I/O function, yielding a *relative* prob. Vector (ψ) in each cluster.
7. Normalize ψ to a *full* probability measure, ρ.
8. Final winner in each minc chosen *as draw* from ρ distribution (2nd round, soft max)



Algorithm Example: Storing First Spatial Input



Algorithm Example: Presenting Training Input Exactly



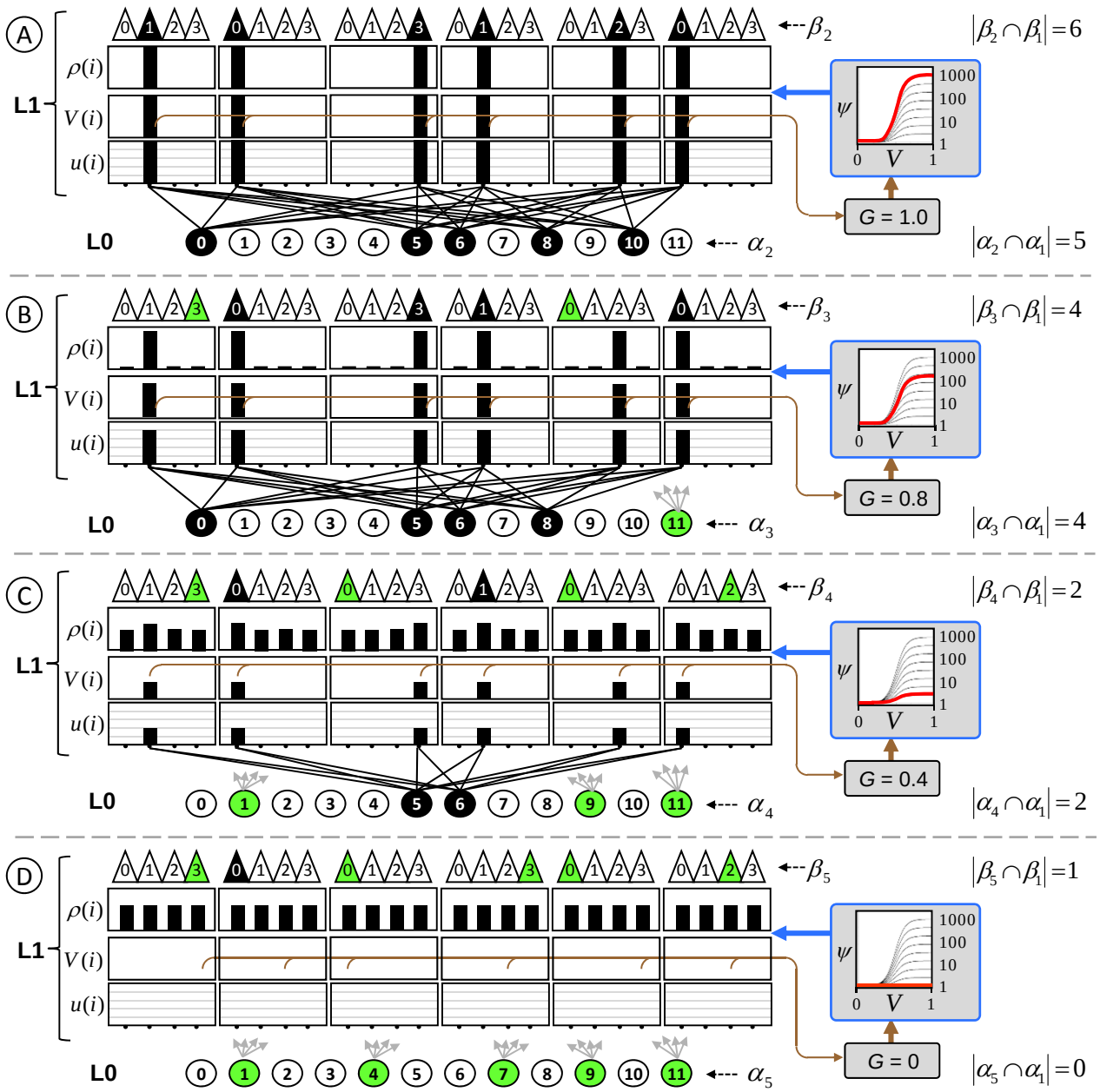
Algorithm Example Cont'd:

- Presenting exactly the same input again
- 100% overlap

- Presenting very similar input
- 80% overlap

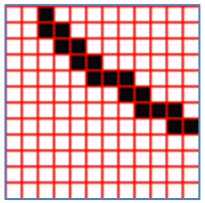
- Presenting very different input
- 40% overlap

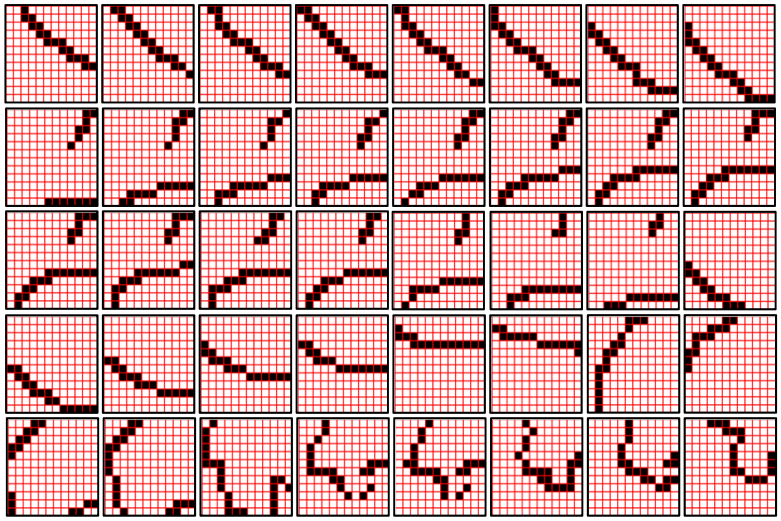
- Presenting entirely different input
- 0% overlap

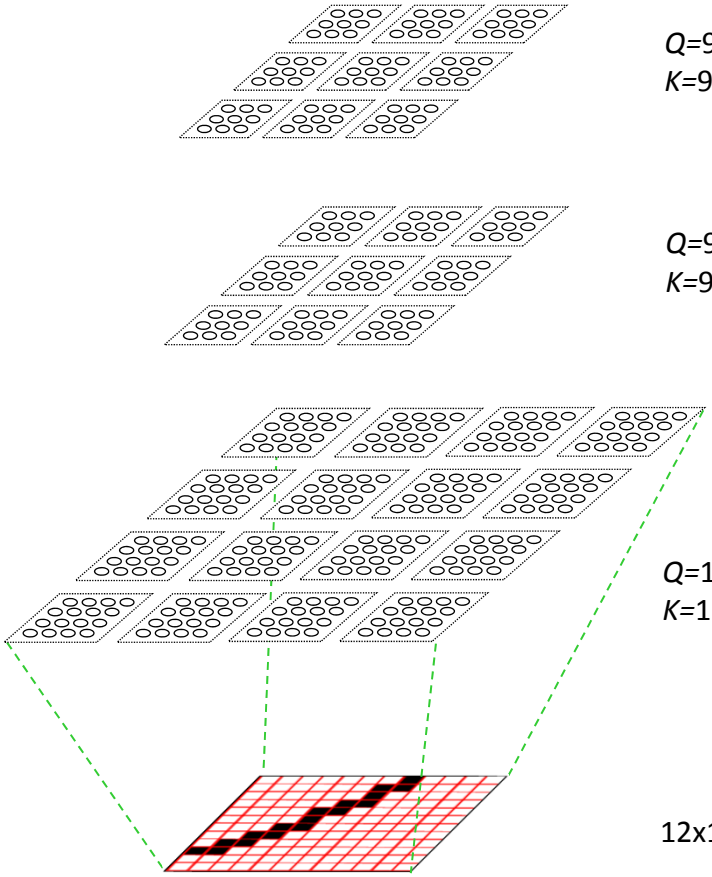


Simulation Results: Learning and Recognizing a 40-frame Sequence

40-frame 12x12 edge-filtered snippet from a natural video (some frames deleted)





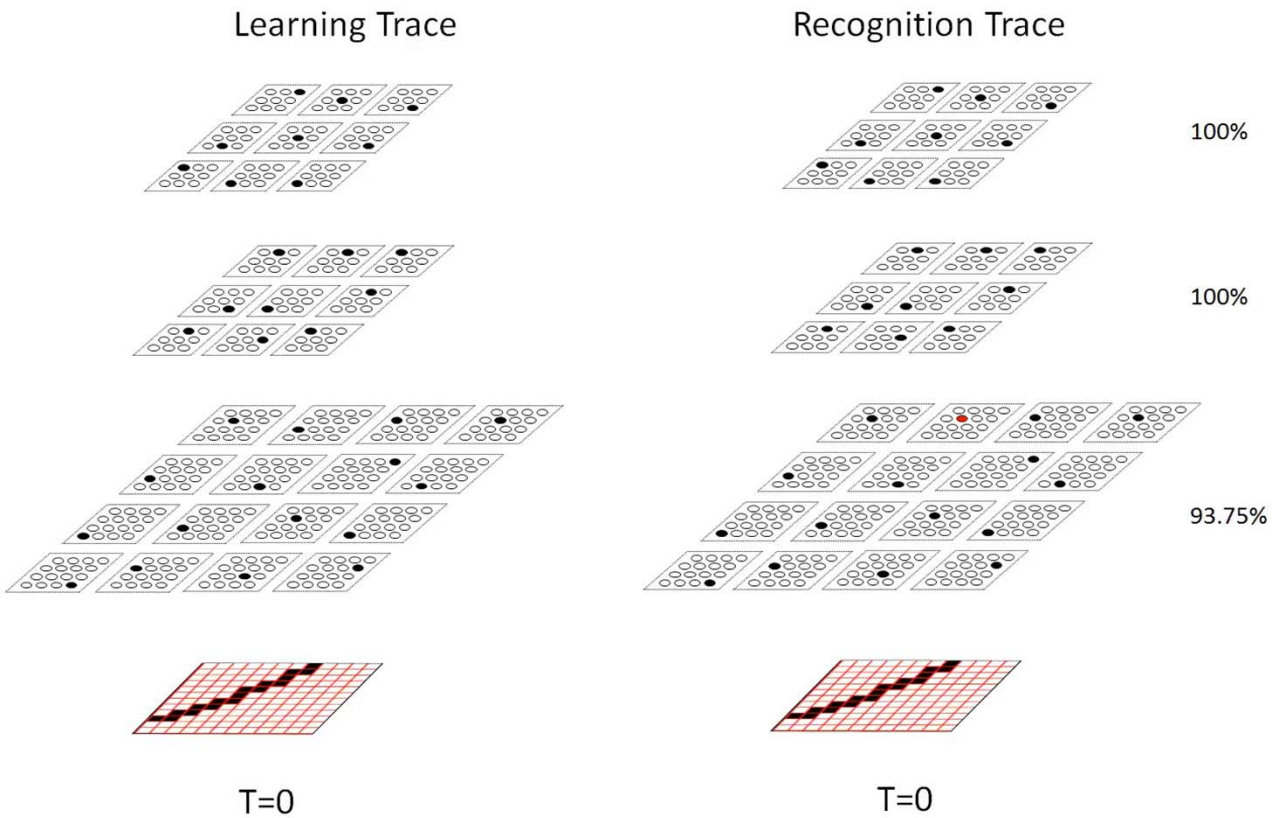


207,000 binary wts.

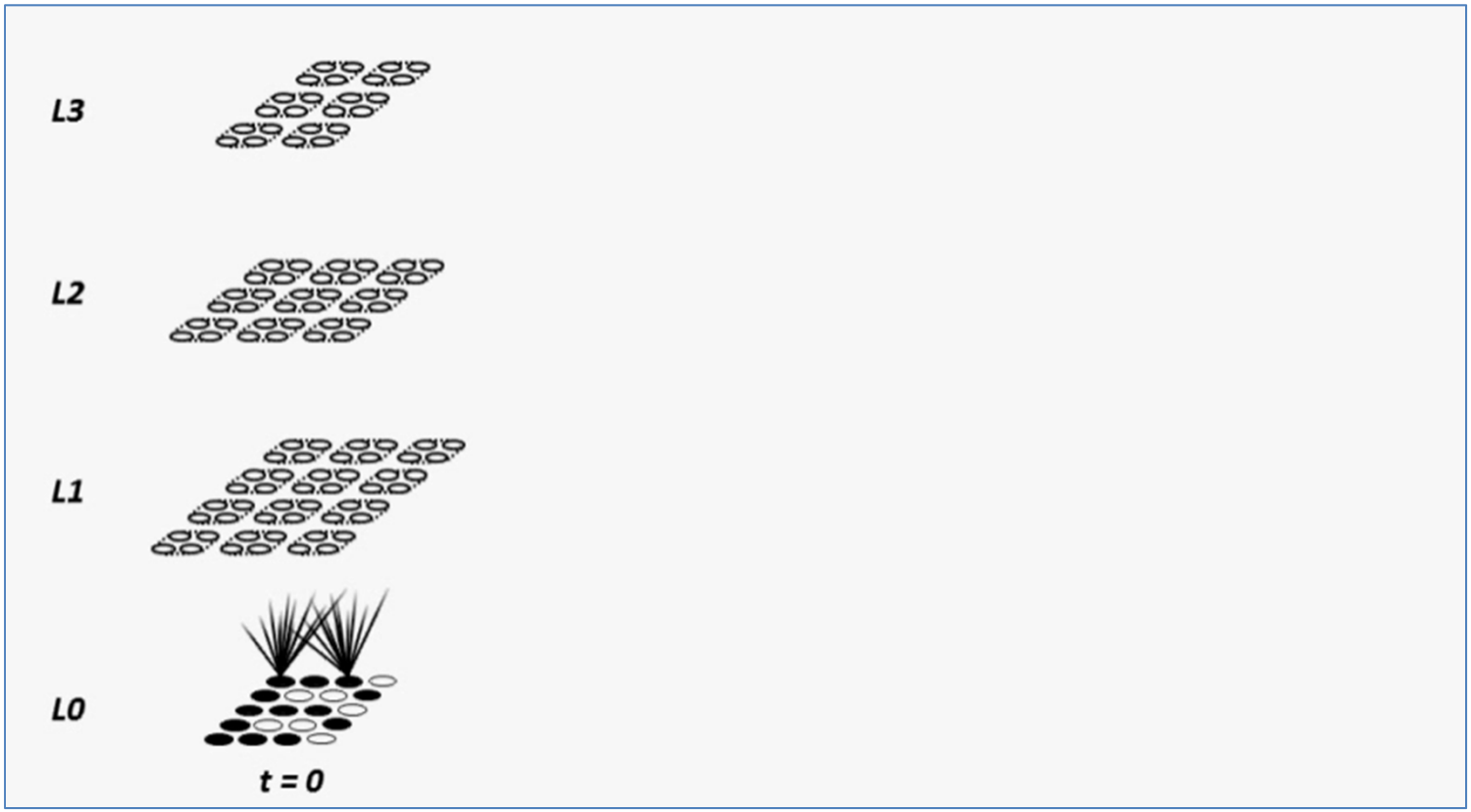
Simulation Results: Learning and Recognizing a 40-frame Sequence

- The sequence is presented once.
- A hierarchical spatiotemporal trace formed on the fly.
- When sequence is presented a 2nd time, the exact same memory trace transpires throughout all levels and time steps.

- In other words, the model finds the closest matching hypothesis at each moment.
- The L1 mac has stored 40 codes. Each of these codes represents a particular spatiotemporal hypothesis, i.e., a particular "moment"
- On each of the 40 frames, the combined U, H, and D signals to L1 conspire to reactivate the one correct code.
- But the algorithm does not iterate over stored codes.
- A few errors—single unit errors—are made during the recognition trial. However, they are corrected on the following time steps.
- This means that when mapped to hardware, the system is fault tolerant.



Formation of a Hierarchical Spatiotemporal SDR Memory Trace



- One mac per level
- Different numbers of clusters per mac at each level
- 4x5 pixel aperture

Conclusions

- Massive increase in computing power needed
- For general probabilistic inference.
- TEMECOR's probabilistic inference algorithm does constant time storage (learning) AND constant time best-match retrieval.
- Of spatiotemporal patterns (sequences)
- Primarily by using SDR instead of localist codes.
- It can mapped to the new hardware technologies.
- Corrects errors through time, fault-tolerant