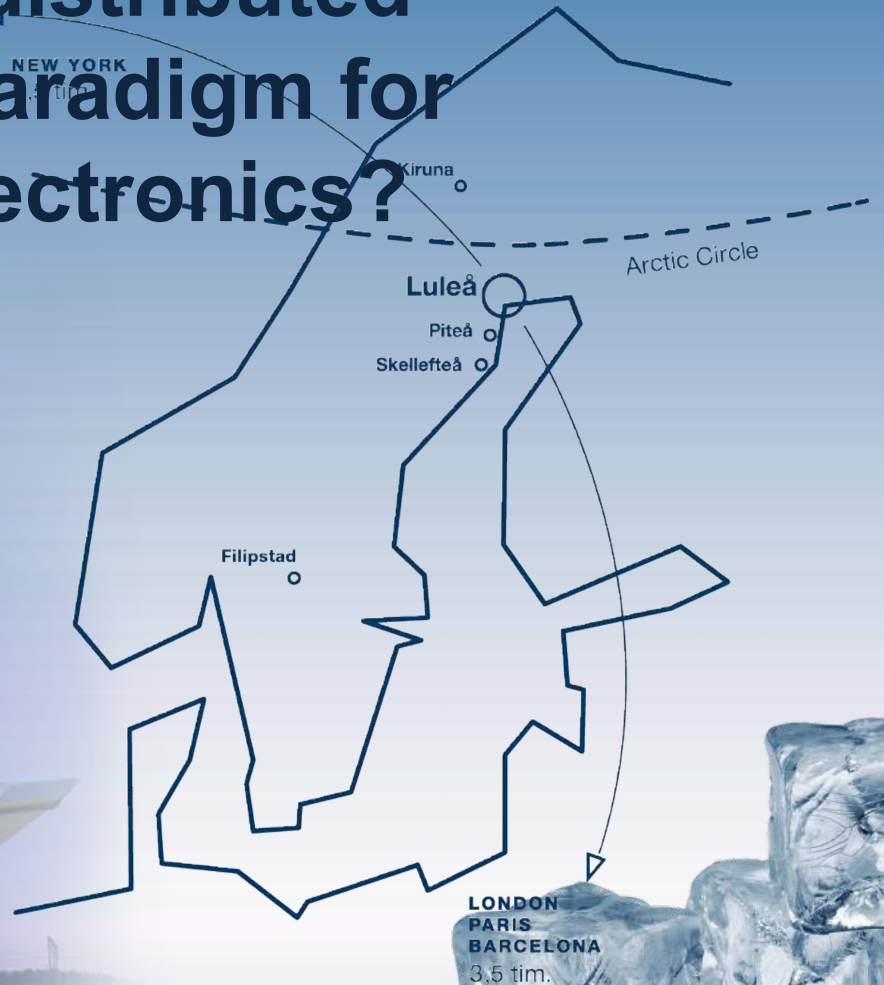
Hyperdimensional distributed representations: a paradigm for future intelligent electronics?

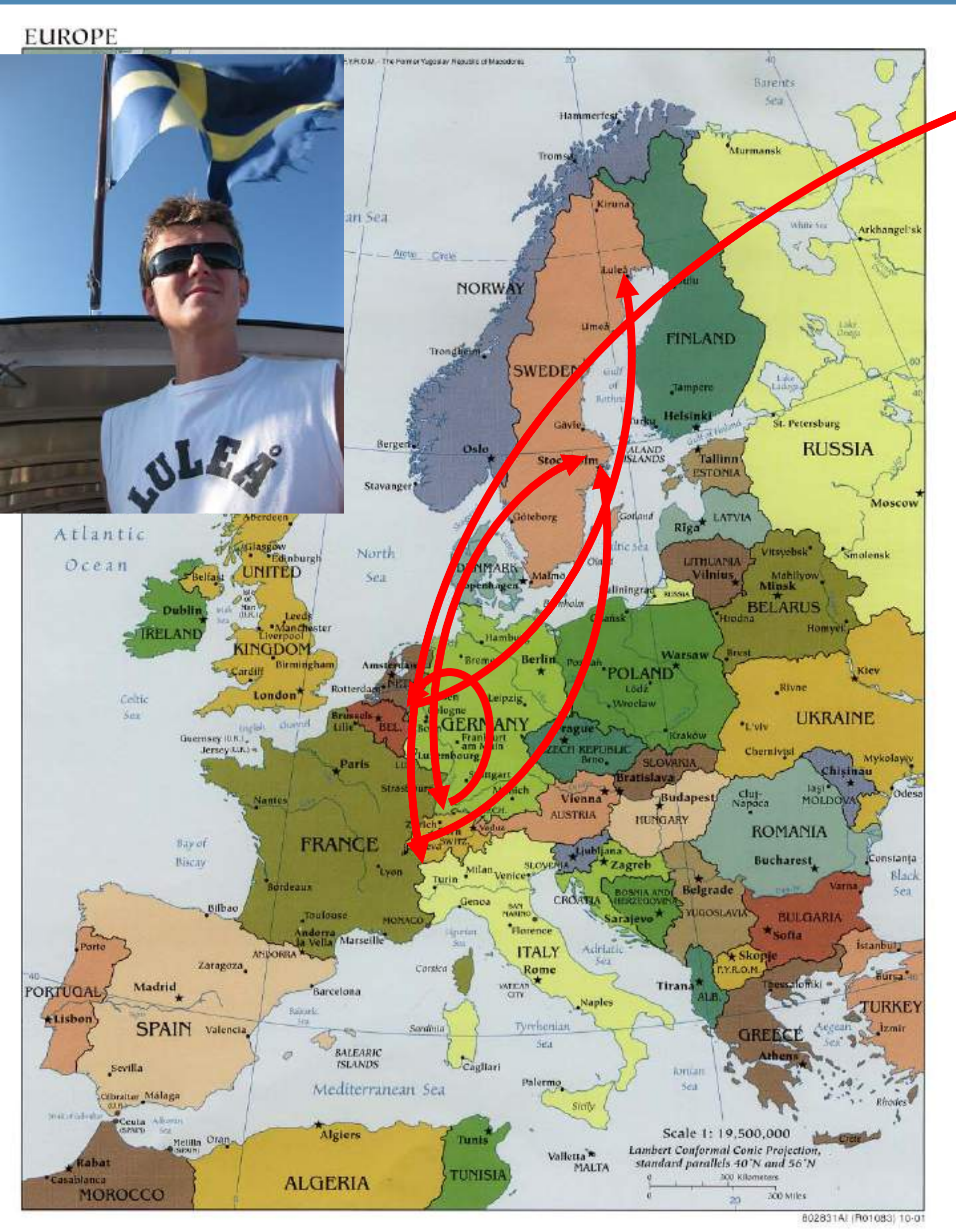


Professor Evgeny Osipov,
Department CSESE
Luleå University of Technology
Evgeny.Osipov@LTU.SE

LULEÅ UNIVERSITY OF TECHNOLOGY

Who is Evgeny Osipov

Krasnoyarsk



1999 – 2003 : Kungliga Tekniska Högskolan (Prof. Dr. Gunnar Karlsson)

- PhD student
- Teaching Assistant

2004 -2005: University of Basel (Prof. Dr. Christian Tschudin)

- Researcher (Wissenschaftler)

2005 SICS, Swedish Institue of Computer Science: (Dr. Bengt Ahlgren)

- Visiting researcher

2006: RWTH Aachen University (Prof. Dr. Petri Mähönen)

- Postdoctoral fellow
- Project Leader

2007 - present: Luleå Tekniska Universitet

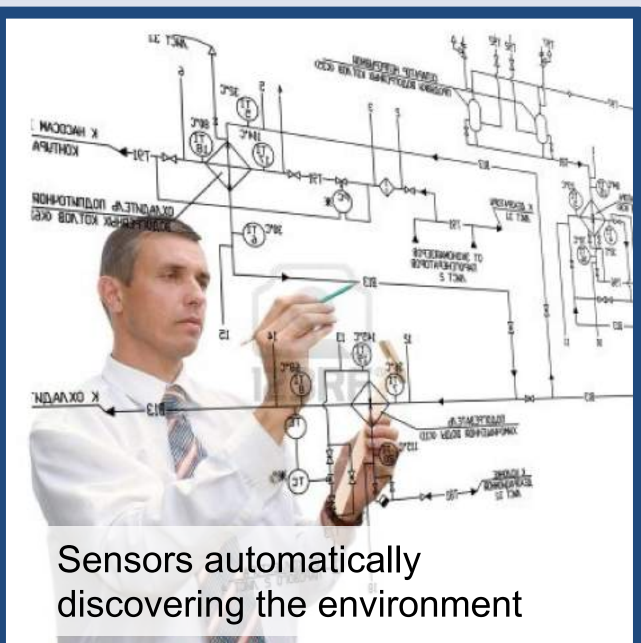
- Universitetslektor, Docent, Professor

This talk is about

- Alternative way of computing
  - belongs to a class of approximate computing
- A specific mathematical framework, called Hyperdimensional Computing and also Vector Symbolic Architectures
  - computations are done in high-dimensional space
  - allows similarity based reasoning
- Properties of high-dimensional spaces
- Transformation from the low-dimensional to high-dimensional spaces
- Use-cases
  - Intelligent transportation systems
  - Fault classification in complex systems
  - Processing of strings of symbols

Technology: Narrow down the scope


Mini robots discovering unknown environment


Sensors automatically discovering the environment

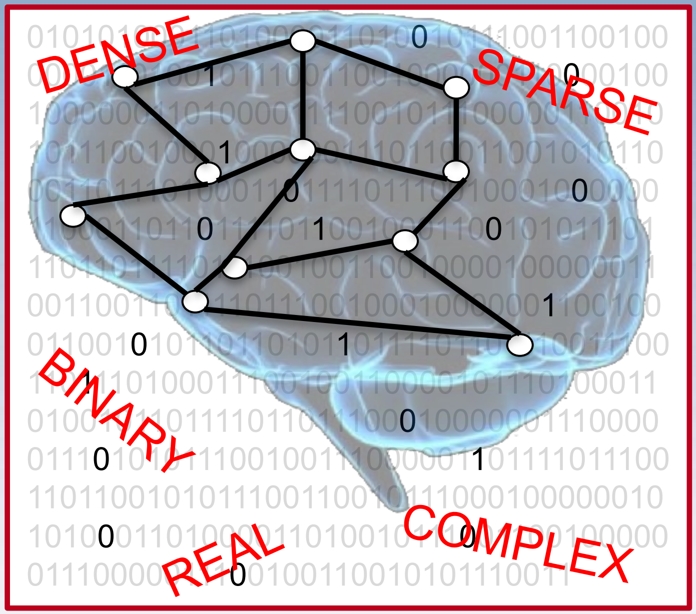
- Consider an artificial learning and prediction system
- With extremely limited possibilities of pre-training, unique individual experience
- The goal is to develop a predictive algorithm that scales up to potentially large (a priory unknown) representational spaces using finite computational resources
- The algorithm should learn on the fly as more and more data is observed from individual episodic instances
  - One shot (incremental learning)
  - Efficient similarity based reasoning

HD computing: Narrowing down the scope

- Cognitive Architectures = artificial mind through understanding and modeling mental processes of biological organisms
- Algorithms and mathematical models inspired by selected brain features for solving a particular set of problems
- HD computing is a **mathematical model** belonging to a class **of approximate computing** for similarity based reasoning and analogical mappings

HD is

- A mathematical (bio-inspired) framework for building a (potentially) general cognitive system with many positive qualities we associate with brains:
  - Robust and noise-tolerant
  - Learns from data/example, learns by analogy
  - Can learn fast: "One-shot" learning
  - Integrates signals from disparate senses
  - Allows simple algorithms that scale to large problems efficiently
  - Allows high degree of parallelism
  - Is implementable on extremely low power electronics



HIGH DIMENSIONALITY
hundreds/thousands of bits and more

RANDOMNESS

SIMILARITY BASED COMPUTING

object	Random bit-string	object	Random bit-string
name	01011100010100001110	Pat	11010000000011000100
sex	01010101110001011100	male	10000011100010001110
age	01111011010011101000	66	10101001101100001111

Biological inspiration:Some intuition would not hurt☺

РСАРПЕДЕЛЕНЫНЕ ПЕРСДТАЛВЕНИЯ И АСОСЦАИТИВАНЯ ПМАЯТЬ

ЧТИТАЬ ТКАИМ ОРБЗАОМ

ТРДНУО ТЛЬОКО ПВЕРЫЕ НСЕКЛОКЬО СКЕНУД

ЧЛОЧЕВЕЕС4КЯ П4ЯМТЬ 4СЦИСО4ИВТН4!

LULEÅ
UNIVERSITY
OF TECHNOLOGY

The mathematical theory of HD computing

- The theory dates to the 1990s and is referred to variously as Holographic Reduced Representation (Plate), Binary Spatter Code (Kanerva), and Vector-Symbolic Architecture (Gayler & Levy) also Semantic Pointer Architecture - SPAUN (Elliasmith)
- The underlying math dates to the late 1800s and early 1900s and is referred to as abstract algebra.

- **Kanerva, P. "Hyperdimensional Computing: An Introduction to Computing in Distributed Representation with High-Dimensional Random Vectors." Journal of Cognitive Computation, 2009.**

Bio-inspired: Similarity as the central factor of cognitive systems -> Vector Symbolic Architectures

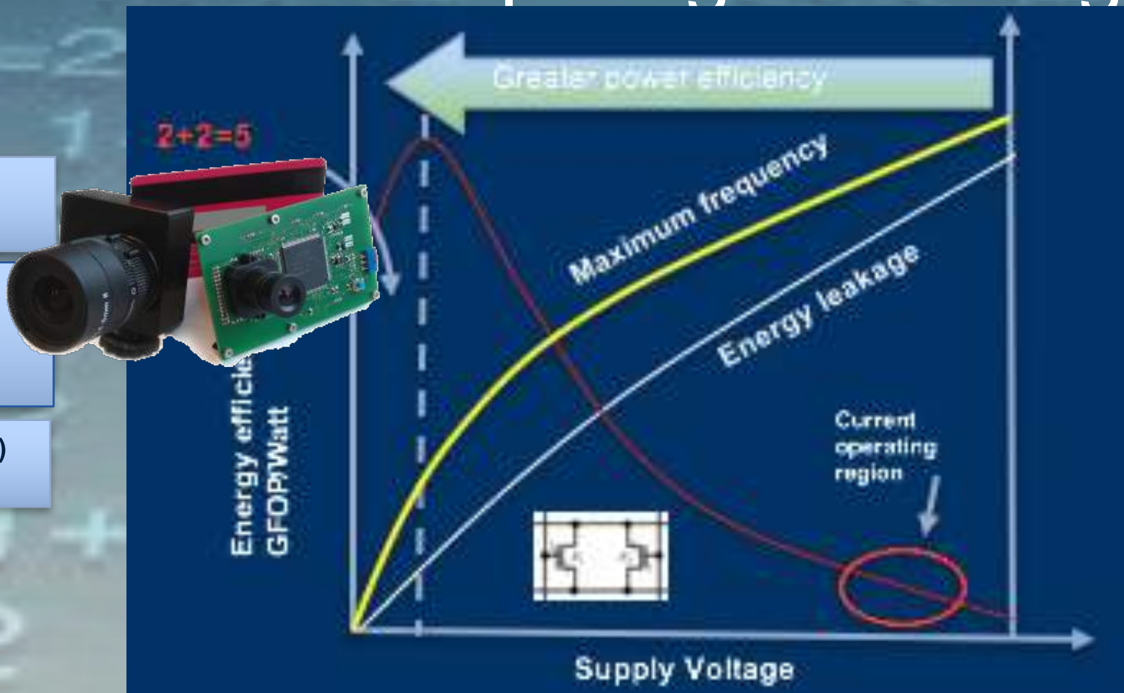
- Large diversity of symbols/concepts are represented by unrelated (orthogonal) vectors
- There are functions, which transform (initially) unrelated symbols into places with desired proximity: **Vector-Symbolic Architectures (Gayler, Levy), Semantic Pointer Architecture (Eliasmith)**
- There is a distance metric which reflects (semantic) similarity between point in space

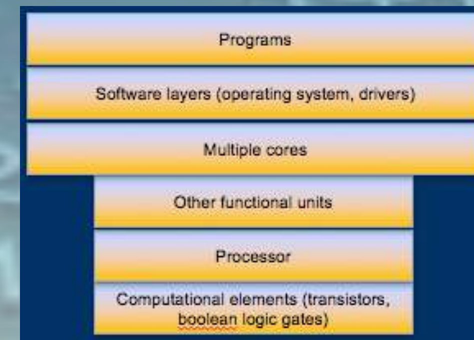
HD and HW: What's wrong with the current Computing technology

inference

Approximate Computational Models

New (imprecise) devices

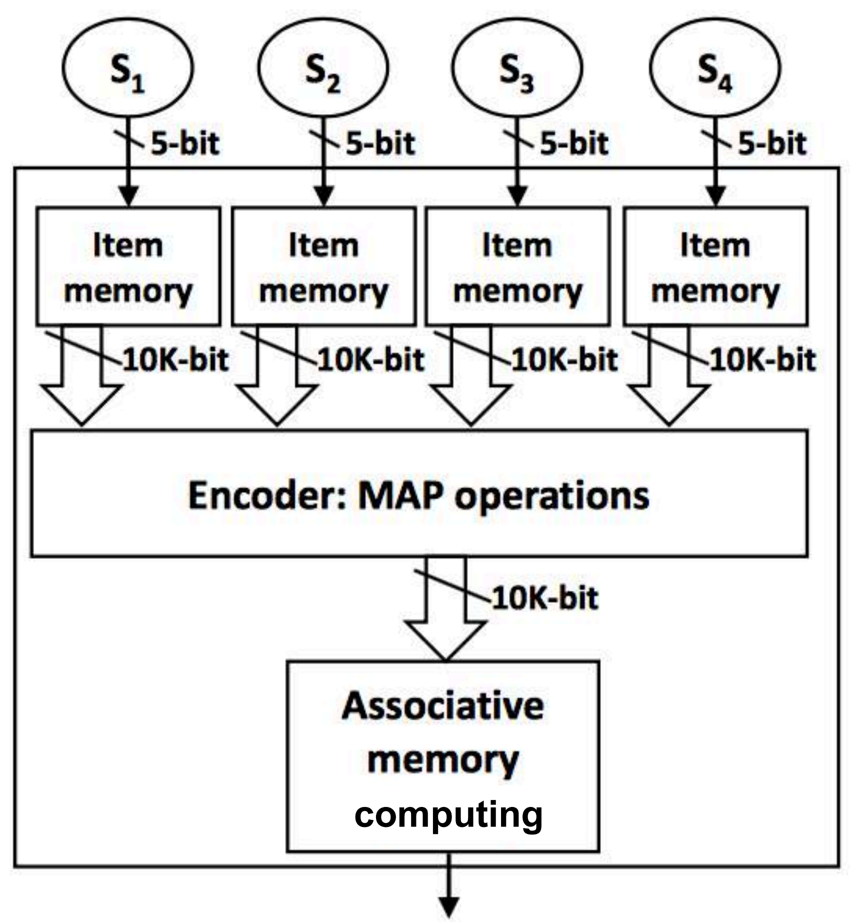




- Subthreshold computing:
  - Approximate computing
  - Neuro-inspired computing

NSF Enigma project. UC Berkeley, U. Stanford

HD computer: an architecture



Output (Classification/Prediction/result)

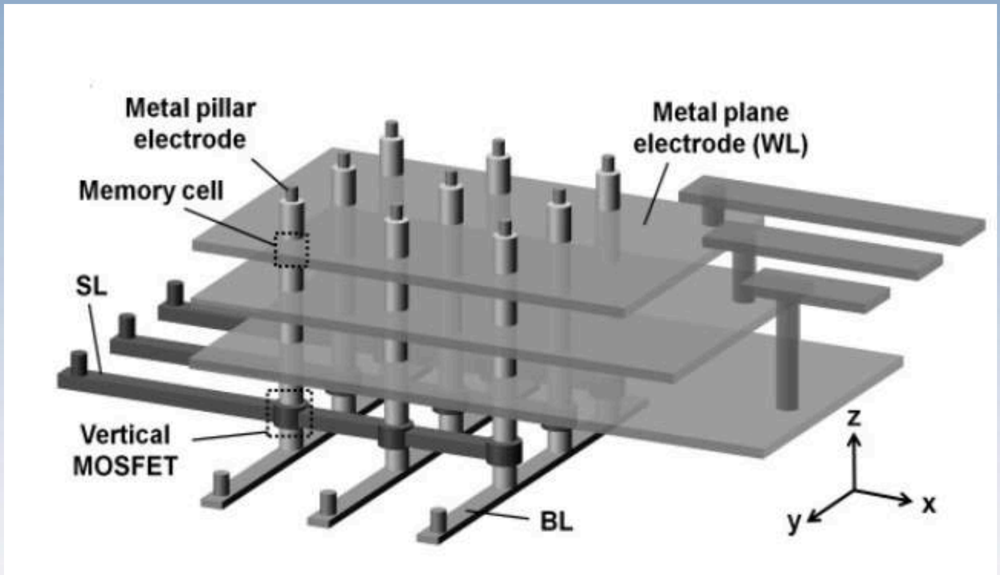
A fully general system with many positive qualities :

- Integrates signals from disparate senses
- Learns from data/example, learns by analogy
- Can learn fast: "One-shot" learning
- Allows simple algorithms that scale to large problems efficiently
- Robust and noise-tolerant
- Allows high degree of parallelism
- AND…

- This is an architecture of an HD computing device

HD computer: computing using Associative Memory

- HD computer is a memory-centric machine
  - Frequent patterns can be stored and retrieved from a table using a set of keys
- Used in a broad spectrum of applications including query processing, search engines, text and image processing, pattern recognition and data mining
- Implemented on memory and operation efficient hardware architectures

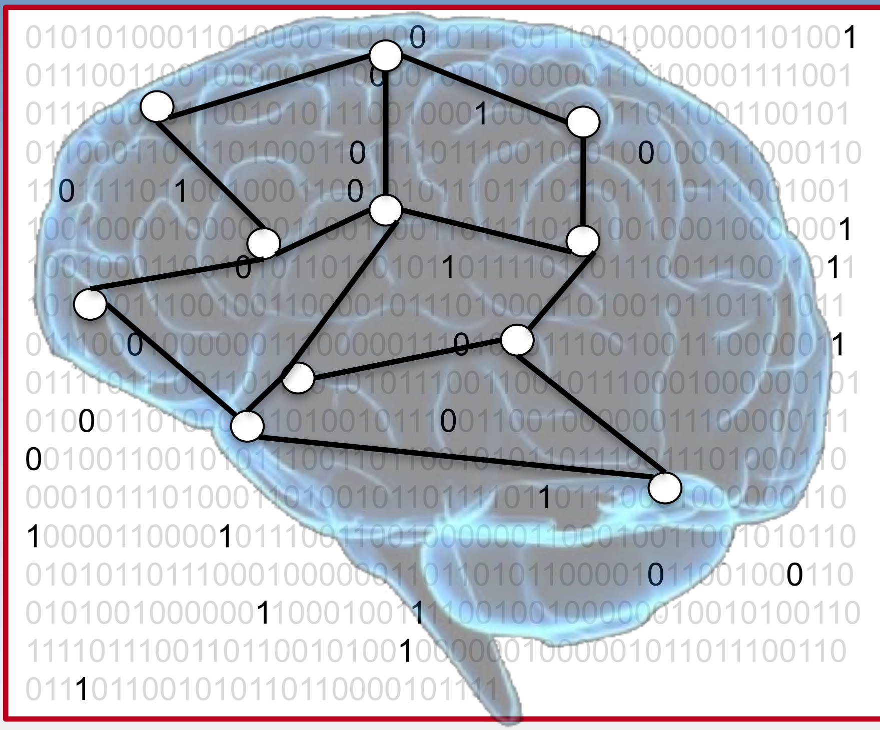


Schematic of a 3D Resistive Memory Array

47% more power efficient in classification tasks with marginal loss of accuracy (approx. 1%)

H. Li *et al.*, "Hyperdimensional computing with 3D VRRAM in-memory kernels: Device-architecture co-design for energy-efficient, error-resilient language recognition," *2016 IEEE International Electron Devices Meeting (IEDM)*, San Francisco, CA, 2016, pp. 16.1.1-16.1.4.

Hyper-dimensional computing



Computations:

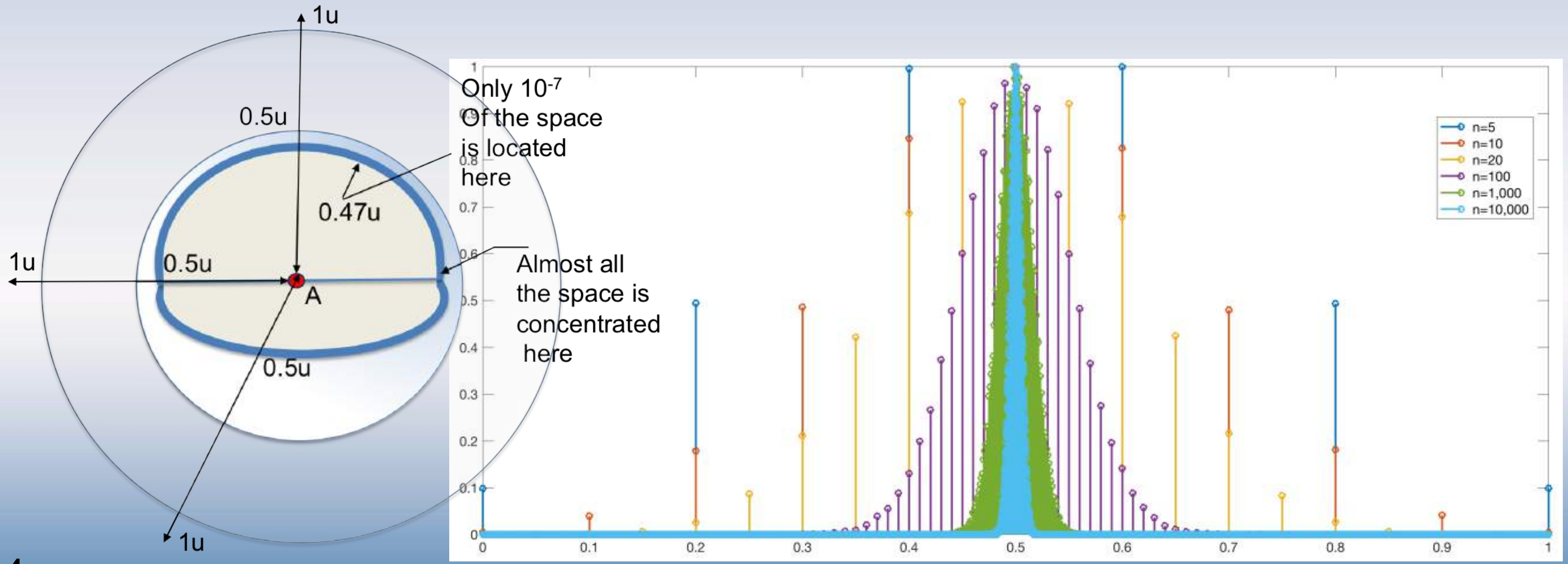
- Distributed representations
- Rich ambiguous representation:
  - (>1000 dimentional space)
- Simple operations
- Similarity-based reasoning
- Extremely robust to errors
- Complex functionality

object	Random bit-string	object	Random bit-string
name	01011100010100001110	Pat	11010000000011000100
sex	01010101110001011100	male	10000011100010001110
age	01111011010011101000	66	10101001101100001111

Each bit does not matter per se only the entire sequence make sense
Very robust to errors
Suitable for implementation on on imprecise hardware

Similarity measure

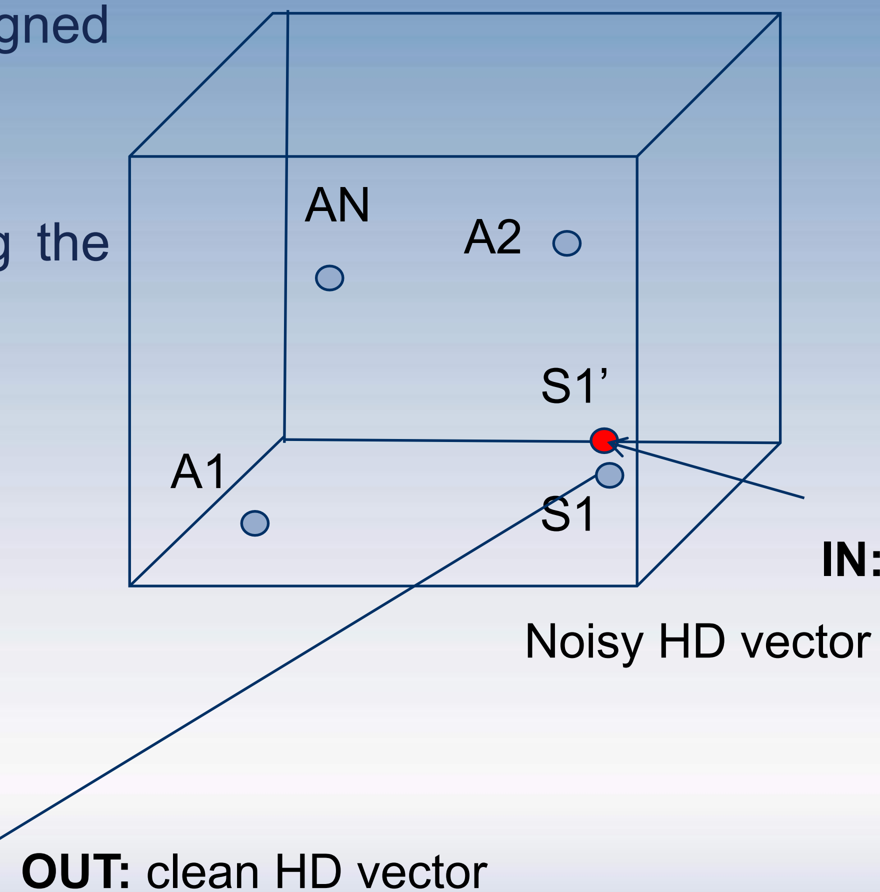
- Measures the similarity between two vectors **x** and **y**, giving a real number
- ‘No significant similarity’ between two randomly generated vectors
- Normalized Hamming distance for binary vectors



most of the space is concentrated on a thin surface with radius 0.5u±0.03

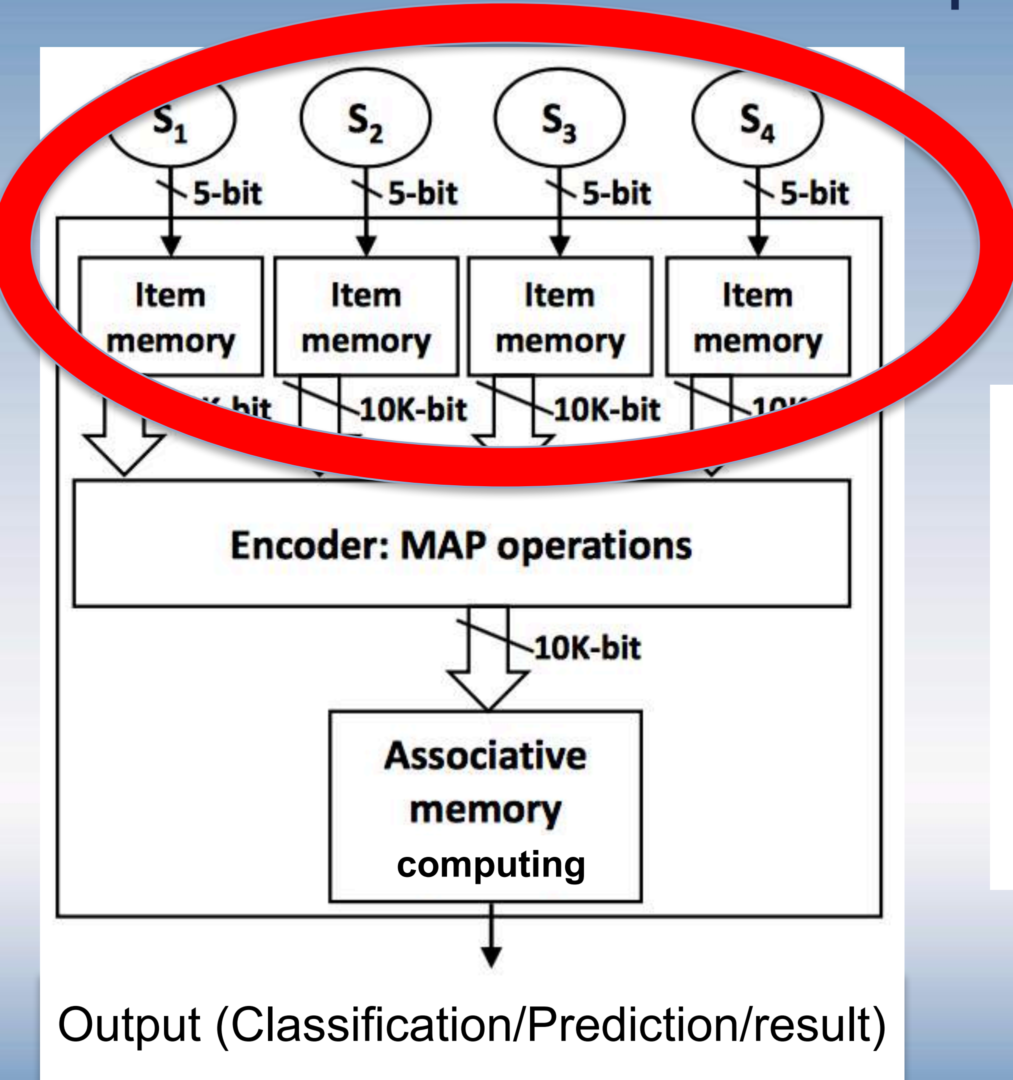
Item-memory: storage of meaningful HD-vectors

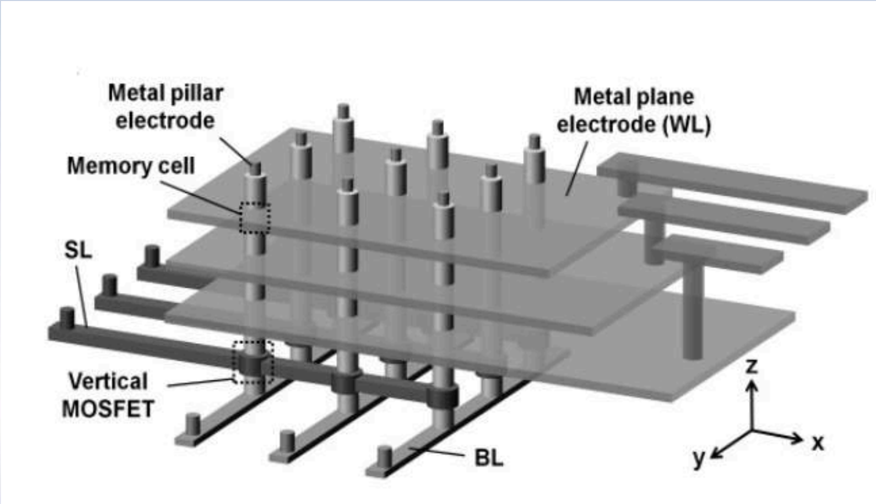
- Denoising principle
- A set of random HD vectors with assigned meaning
- Also called clean-up memory
- Does nearest-neighbor search among the set of stored meaningful HD-vectors.



HD-vector	Meaning
01110010101 (A1)	OBJECT1
10101010011 (A2)	OBJECT2
…	
10111011011 (AN)	OBJECT3
01010101110 (S1)	…

A very quick cross-reference back: HD computer



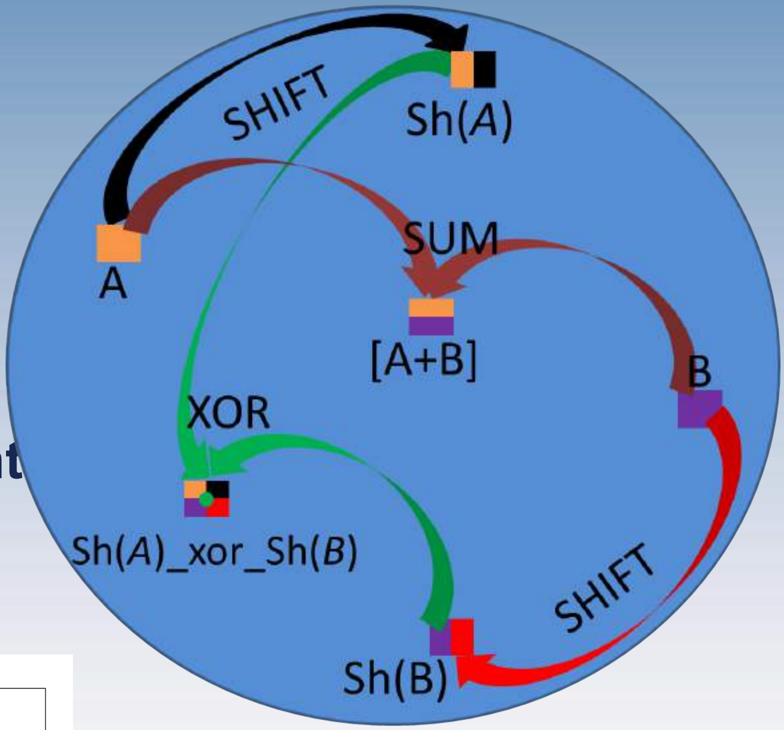


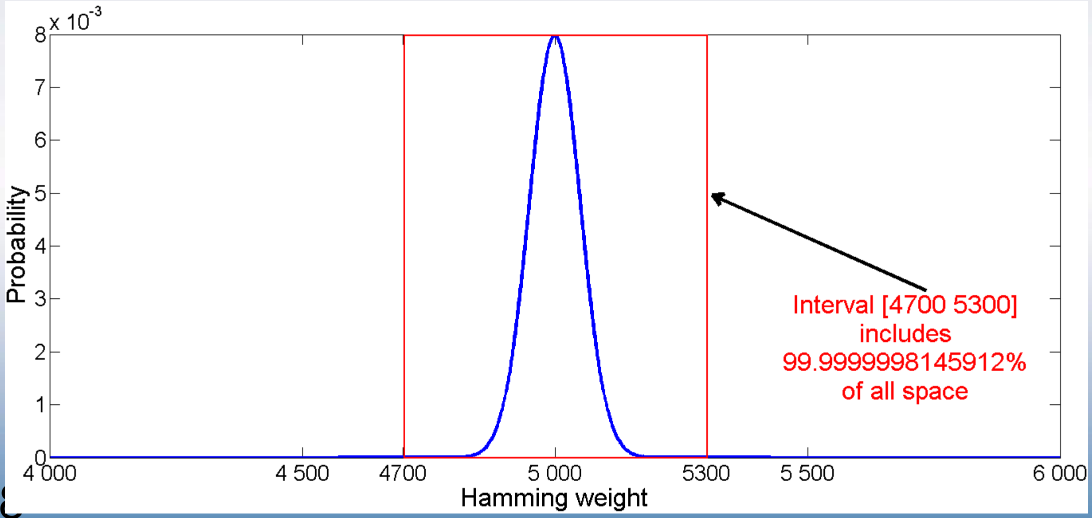
Key properties of HD algebra

- Resembles ordinary computing on Booleans and numbers: Addition (bundling), Multiplication (binding), Permutation (ordering):
  - Addition commutes: A + B = B + A
  - Addition, multiplication and permutation are invertible
  - Multiplication distributes over addition
  - Permutation distributes over both addition and multiplication
  - The output of addition is similar to the inputs
  - The outputs of multiplication and permutation are dissimilar to the inputs
  - Multiplication and permutation preserve similarity

Properties some more details

- **Multiplication (used for Binding):** XOR
- Δ_H(operand,result)≈0.5
- **Addition (used for Bundling):** Majority sum
- Δ_H(operand,result)<0.5
- **Permutation (application dependent usage)**: Cyclic shift
- Δ_H(vector, shifted vector)≈0.5



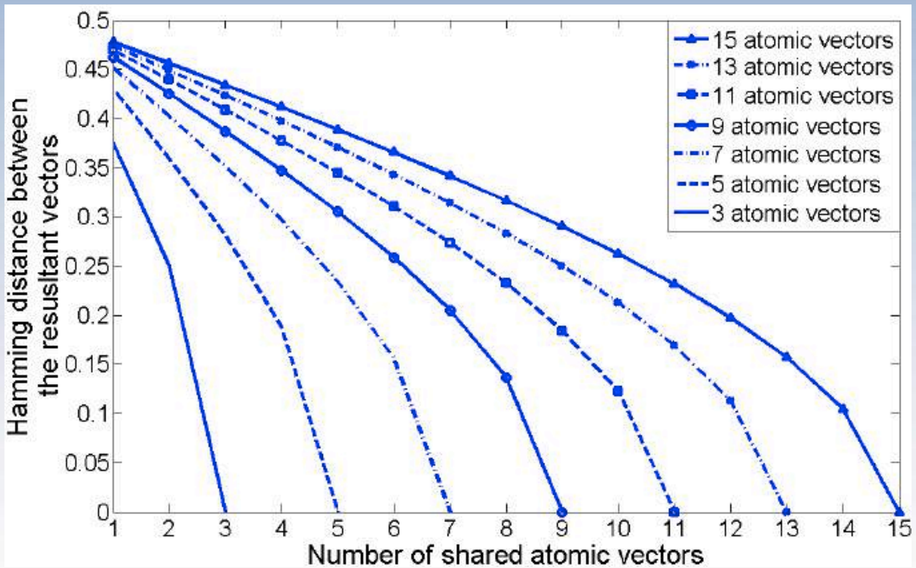


Majority SUM

1 0 0 1 1 0 1
1 1 0 0 1 0 1
0 0 0 1 1 1 0
----
2 1 0 2 3 1 2

1 0 0 1 1 0 1

Treshold = Number of inputs/2



Hamming distance changes in discrete steps depending on the number of summed vectors

The cognitive staff: Analogical mapping (preliminaries)

- Using HD vectors it is possible to create a distributed representation of information
- Let us represent each element as a random hypervector (for example 10 000 dimensional denote as "*_hv")

Example

Record: AU
Country = Australia
Currency = Dollar

Country-> **country_hv**

Currency-> **currency_hv**

Australia -> **Australia_hv**

Dollar -> **Dollar_hv**

Record -> **AU_hv**

The cognitive staff: Analogical mapping (preliminaries)

- Using HD vectors it is possible to create a distributed representation of information

Example

Record: AU
Country = Australia
Currency = Dollar

2. Build the relationship field = value (binding):

$$\texttt{country\_hv} \otimes \texttt{Australia\_hv}$$
$$\texttt{currency\_hv} \otimes \texttt{Dollar\_hv}$$

3. Build the resultant vector (bundling):

$$\texttt{AU\_hv} = [(\texttt{country\_hv} \otimes \texttt{Australia\_hv}) + (\texttt{currency\_hv} \otimes \texttt{Dollar\_hv})]$$

Probing (decoding) example

AU_hv =[(country_hv ⊗ Australia_hv) + (currency_hv ⊗ Dollar_hv)]

What is the name of the country?

1. NAME
   = country_hv ⊗ **AU_hv**
   = country_hv ⊗ [(country_hv ⊗ Australia_hv) +
   (currency_hv ⊗ Dollar_hv)]
   $= \text{Australia\_hv} + \underbrace{\text{country\_hv} \otimes \text{currency\_hv} \otimes \text{Dollar\_hv}}_{\text{not valid vector; act as a noise}}$
2. NAME to Item-memory returns **Australia_hv ->**
3. Name of the country is **Australia**

Analogical mapping in HD computing:
What is the Dollar of Mexico?

- Allows to infer the literal meaning from figurative expressions

AU:
Country = Australia
Currency = Dollar

AU_hv =[(country_hv ⊗ Australia_hv) + (currency_hv ⊗ Dollar_hv)]

MX:
Country = Mexico
Currency = Peso

MX_hv =[(country_hv ⊗ Mexico_hv) + (currency_hv ⊗ Peso_hv)]

P. Kanerva, Hyperdimensional Computing: An Introduction to Computing in Distributed Representation with High-Dimensional Random Vectors, 2009

What is the Dollar of Mexico? -2

- Literal interpretation of Dollar of Mexico produces nonsense:

```
Dollar_hv ⊗ MX_hv
    = Dollar_hv ⊗ [(country_hv ⊗ Mexico_hv) + (currency_hv ⊗
      Peso_hv)]
    = [(Dollar_hv ⊗ country_hv ⊗ Mexico_hv) + (Dollar_hv ⊗
      currency_hv ⊗ Peso_hv)]
    = NOISE + NOISE
```

P. Kanerva, Hyperdimensional Computing: An Introduction to Computing in Distributed Representation with High-Dimensional Random Vectors, 2009

What is the Dollar of Mexico? -3

- Implicit query gives the answer. What in Mexico corresponds to dollar in Australia?

$\texttt{Dollar\_hv} \otimes (\texttt{MX\_hv} \otimes \texttt{AU\_hv})$

$= \texttt{Dollar\_hv} \otimes (\texttt{Australia\_hv} \otimes \texttt{Mexico\_hv} + \texttt{Dollar\_hv} \otimes \texttt{Peso\_hv} + \text{noise})$

$= \texttt{Dollar\_hv} \otimes \texttt{Australia\_hv} \otimes \texttt{Mexico\_hv} + \texttt{Dollar\_hv} \otimes \texttt{Dollar\_hv} \otimes \texttt{Peso\_hv} + \texttt{Dollar\_hv} \otimes \texttt{noise}$

$= \text{noise} + \text{noise} + \texttt{Peso\_hv} + \text{noise}$

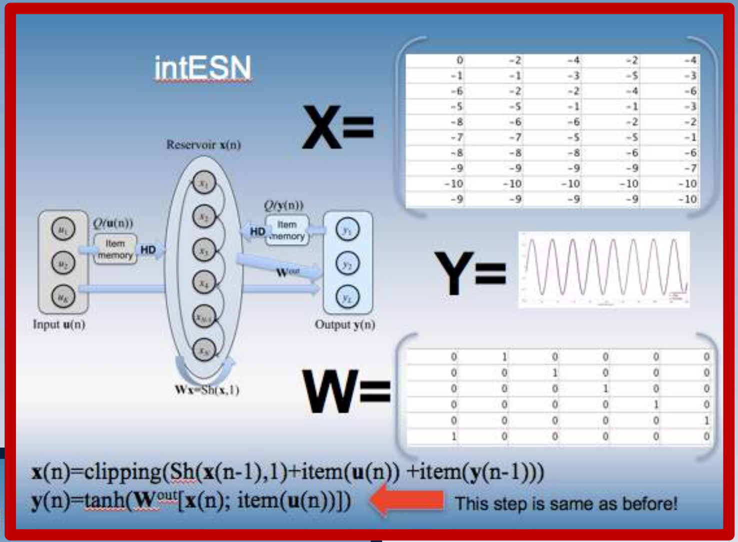
$= \texttt{Peso\_hv} + \text{noise}$

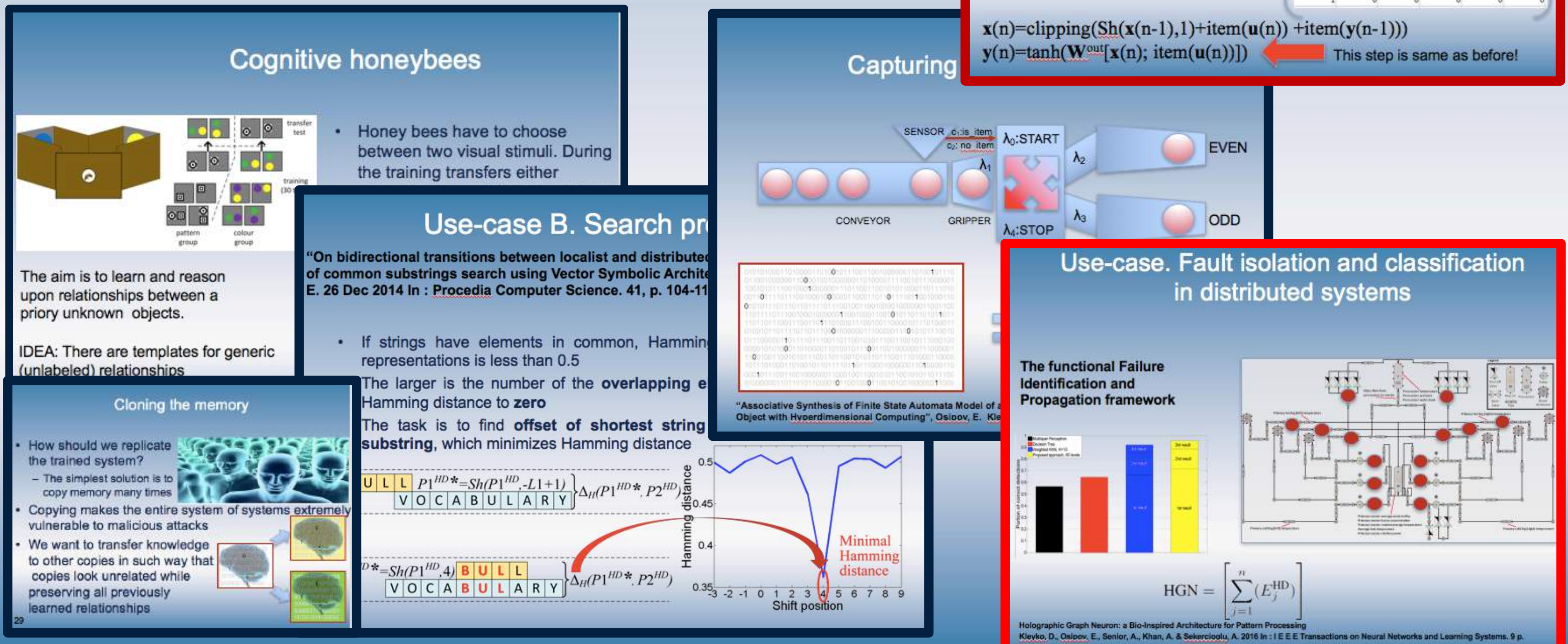
$\approx \texttt{Peso\_hv}$

P. Kanerva, Hyperdimensional Computing: An Introduction to Computing in Distributed Representation with High-Dimensional Random Vectors, 2009

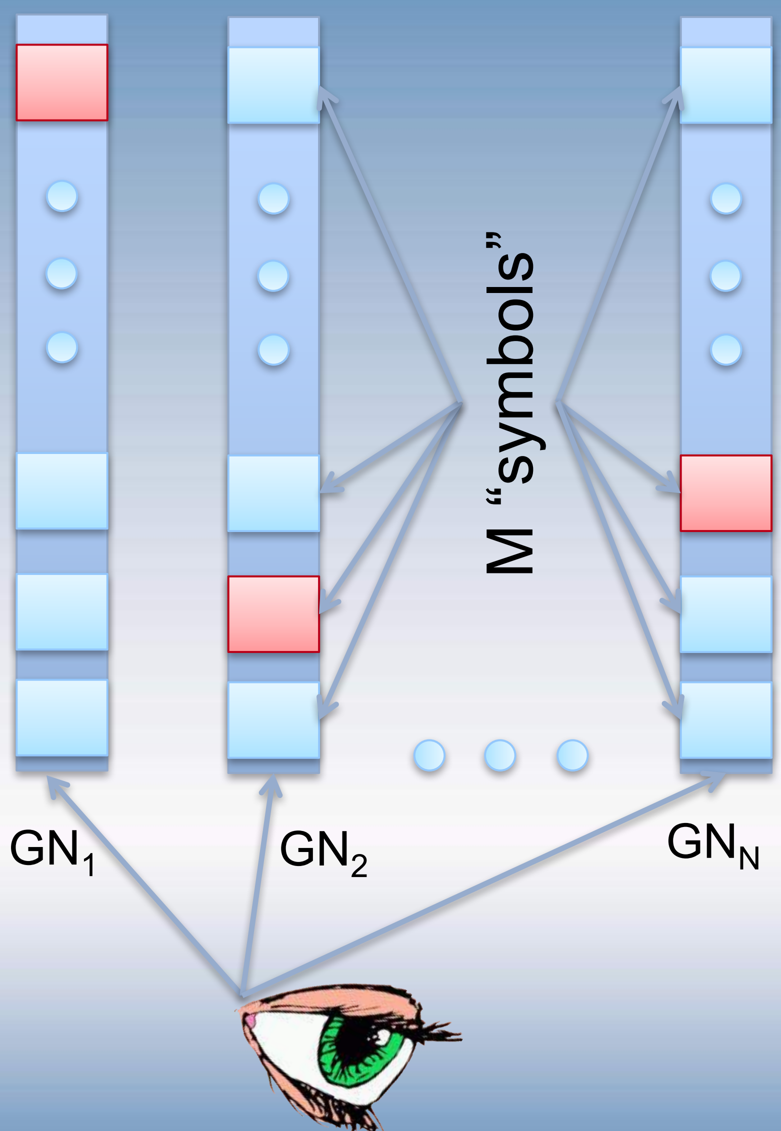
Our selected results

- Encoding features
- Longest substring search
- Modeling learning in honey bees
- Synthesis of finite state automata
- Capturing dynamics: binarization of Echo State Networks





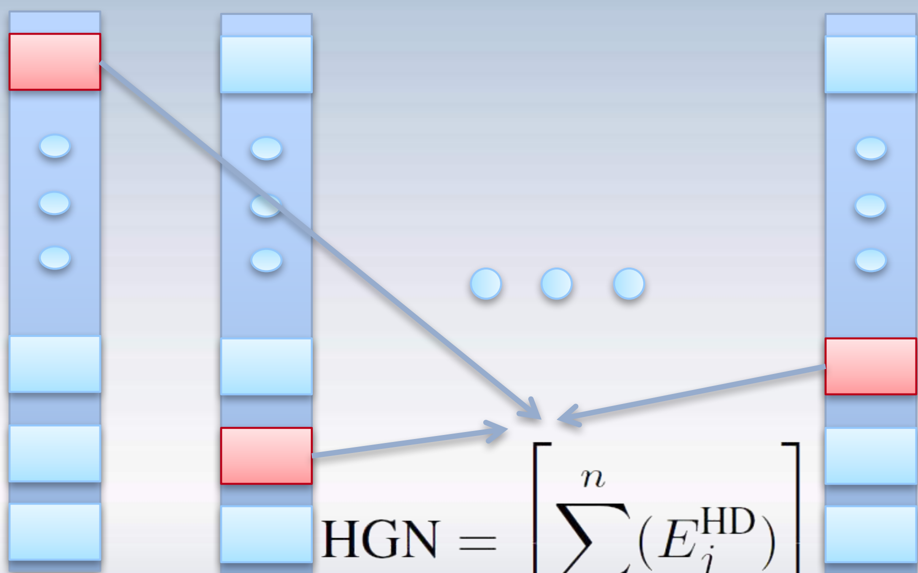
Mapping of features: Graph Neuron



- **Graph Neuron** is an approach for memorizing of patterns of **generic sensor stimuli** for later template matching.
- Neuron = set of discrete values
  - Value of one pixel in an image
  - Value of one sensor
  - Character in on a certain position in a word/sentence
- Pattern = activated values in all neuron

Orthogonal mapping: Holographic Graph Neuron

Generate a set of dissimilar initial vectors for each column (IV_j)



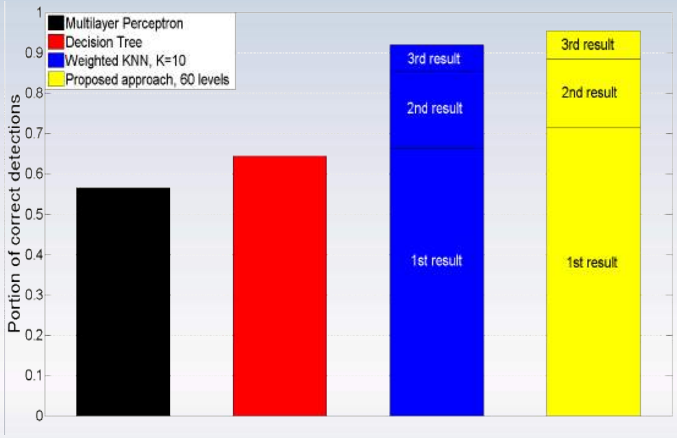
$$\mathrm{HGN} = \left[\sum_{j=1}^{n} \left(E_j^{\mathrm{HD}} \right) \right]$$

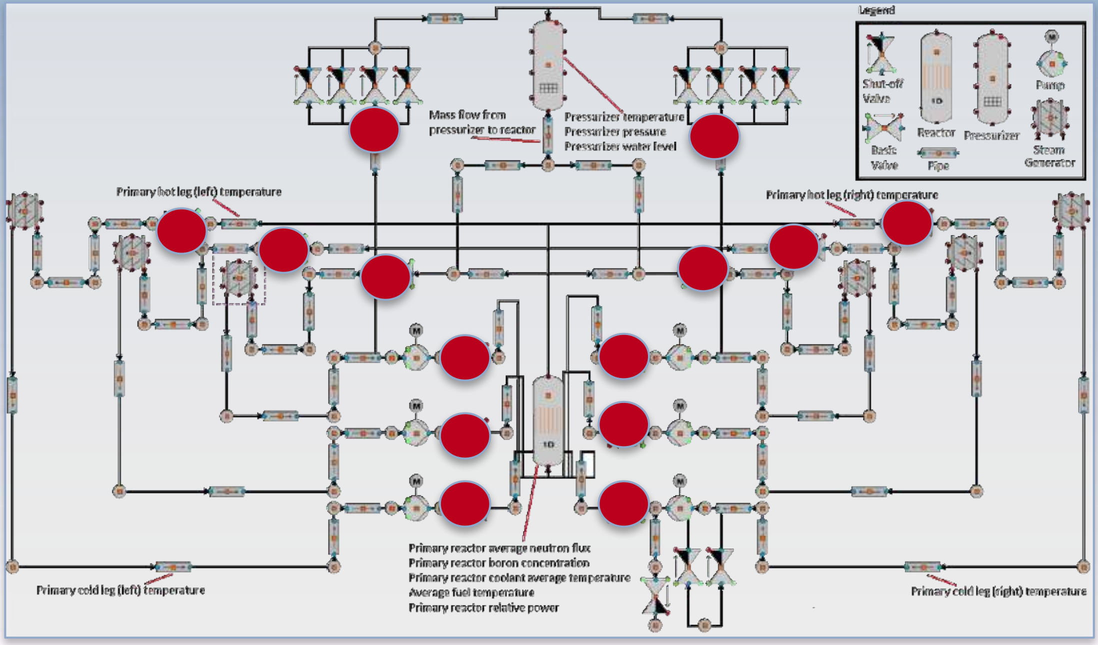
Generate dissimilar random vector for each Graph Neuron to encode indices of activated elements (E)

E^{HD}=SHIFT(IV_j, INDEX)

Use-case A. Fault isolation and classification in distributed systems

The functional Failure Identification and Propagation framework



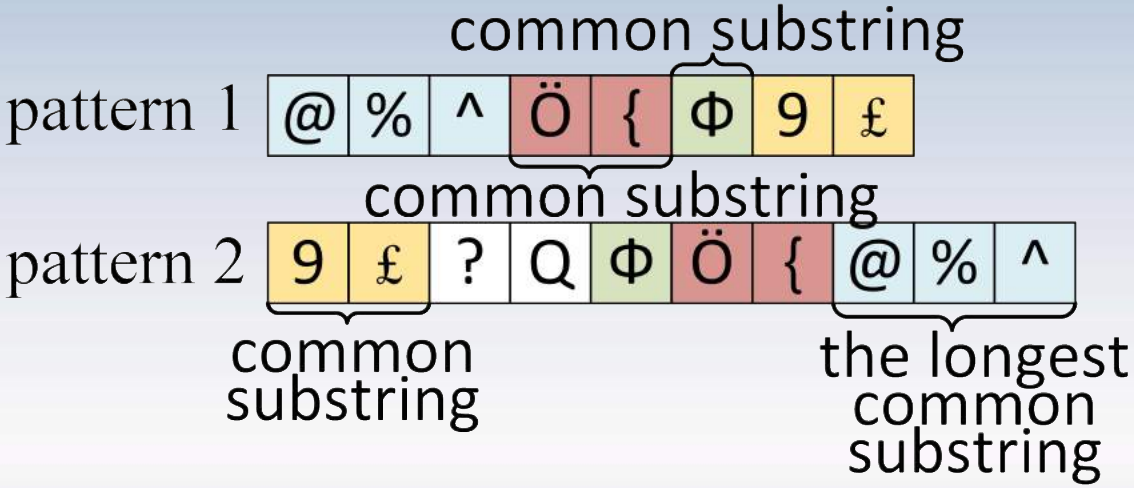


$$\mathrm{HGN} = \left[\sum_{j=1}^{n} (E_j^{\mathrm{HD}}) \right]$$

Holographic Graph Neuron: a Bio-Inspired Architecture for Pattern Processing
Kleyko, D., Osipov, E., Senior, A., Khan, A. & Sekercioglu, A. 2016 In : I E E E Transactions on Neural Networks and Learning Systems. 9 p.

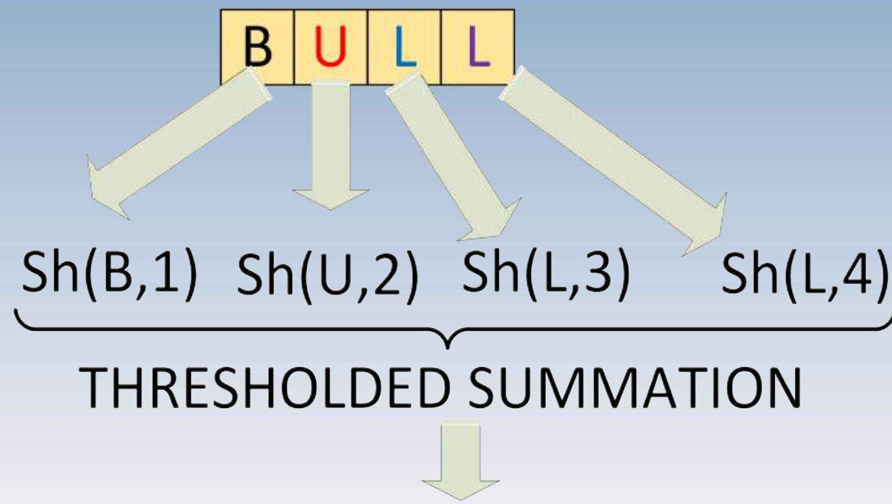
Use-case B. Common substrings search

- Given two strings of characters of different lengths, find the substring of the maximal length



Use-case B. Encoding of strings

- Generate dictionary $\mathbf{D}_{\mathbf{HD}}$ of random HD-vectors for each symbol.
- **Cyclically shift** the initial HD-vector for a given symbol ***S*** on the value of its position ***i***: $Sh(D_{HD}[S]; i)$
- **VSA representation** of the string is a **thresholded** sum of the distributed representations of shifted individual elements $Sh(D_{HD}[S]; i)$.



Distributed representation

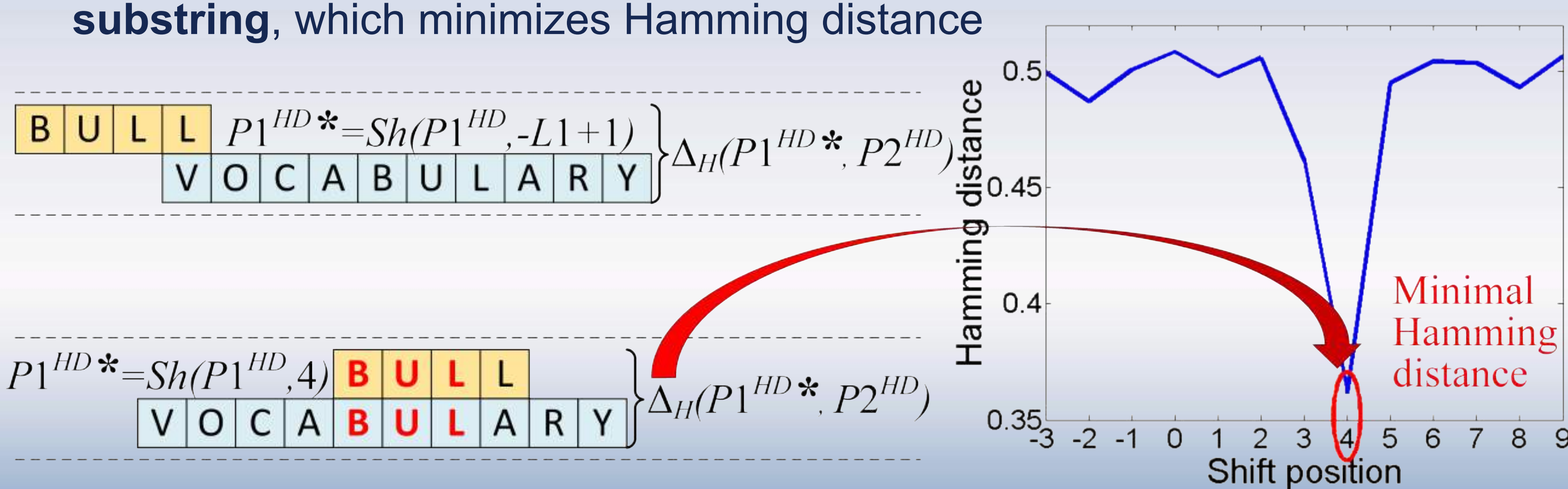
$P1^{HD}$ = [Sh(B; 1) + Sh(U; 2) + Sh(L; 3)+ Sh(L;4)]

101001111010 ... 100100

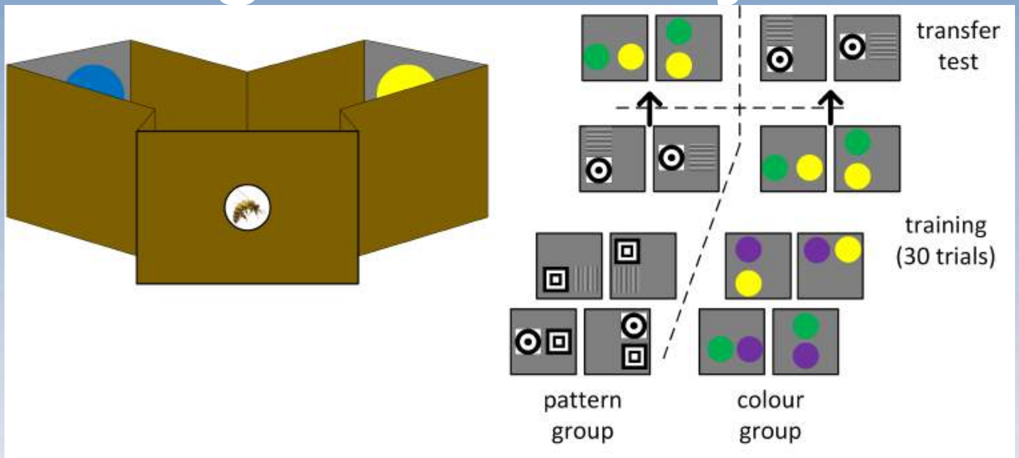
Use-case B. Search procedure

"On bidirectional transitions between localist and distributed representations: The case of common substrings search using Vector Symbolic Architecture", Kleyko, D. & Osipov, E. 26 Dec 2014 In : Procedia Computer Science. 41, p. 104-113.

- If strings have elements in common, Hamming distance between their representations is less than 0.5
- The larger is the number of the **overlapping elements** the closer is the Hamming distance to **zero**
- The task is to find **offset of shortest string relative to the longest substring**, which minimizes Hamming distance



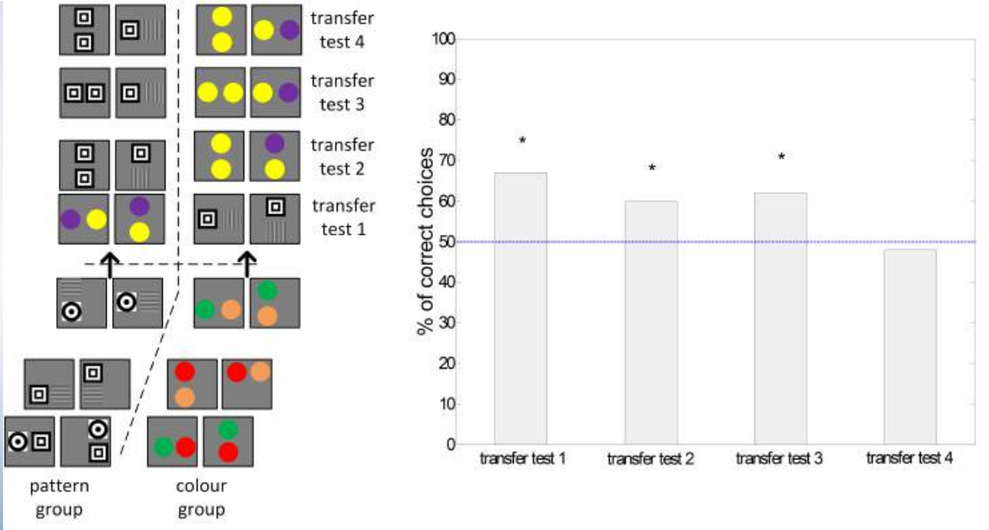
An inspiration from the nature: The cognitive honey bees



Honey bees have to choose between two visual stimuli. During the training trails either above-below or left-right relations are corresponding to the reward.

Opposite relation leads to quinine solution.

The aim is to learn relationships and reason upon between a priori unknown (unseen) objects.



A. Avarguès-Weber, et al., Simultaneous mastering of two abstract concepts by the miniature brain of bees, 2012. PNAS.

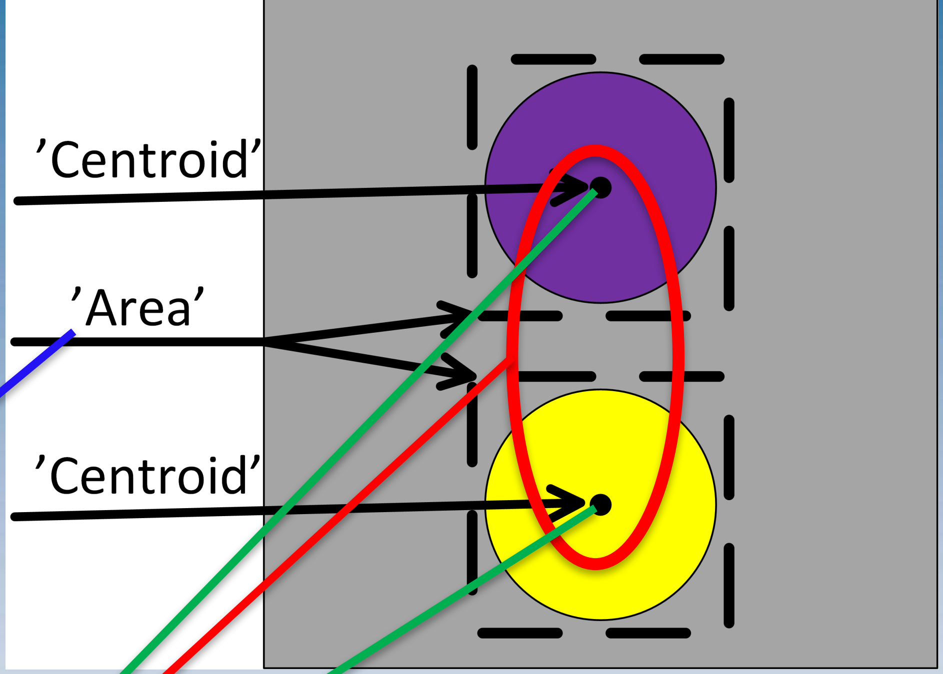
Approach Outline

1. Associative memory based architecture

2. Vector Symbolic Architectures for encoding episodic relationships

3. The essence: There are templates for generic (unlabeled) relationships
   - These are filled with instances of particular objects, their features and are given labels (above, to the right, etc) at runtime for further recall.

Visual stimuli



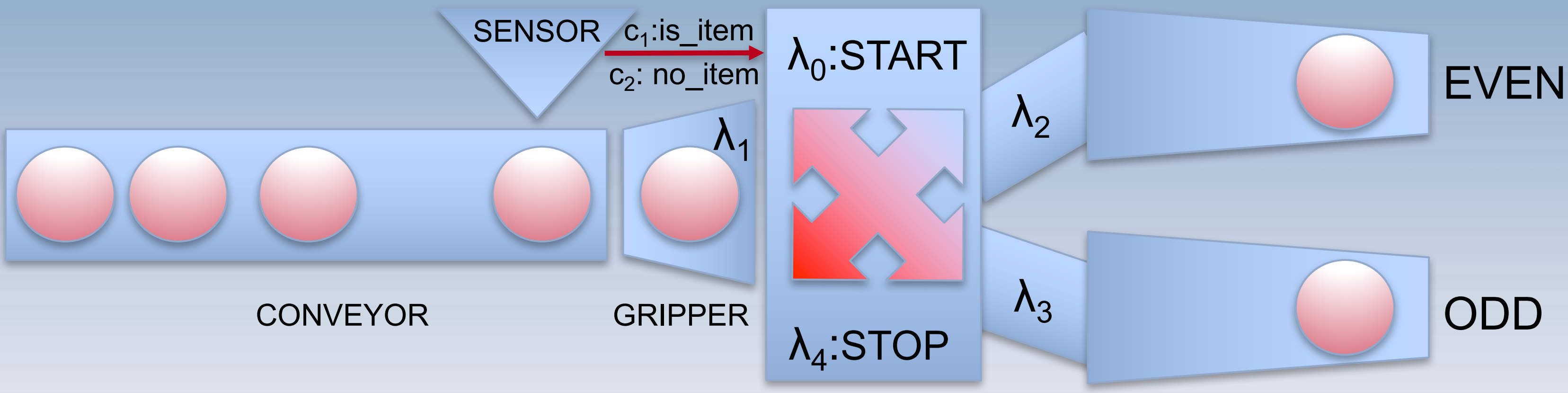
From the measured values:

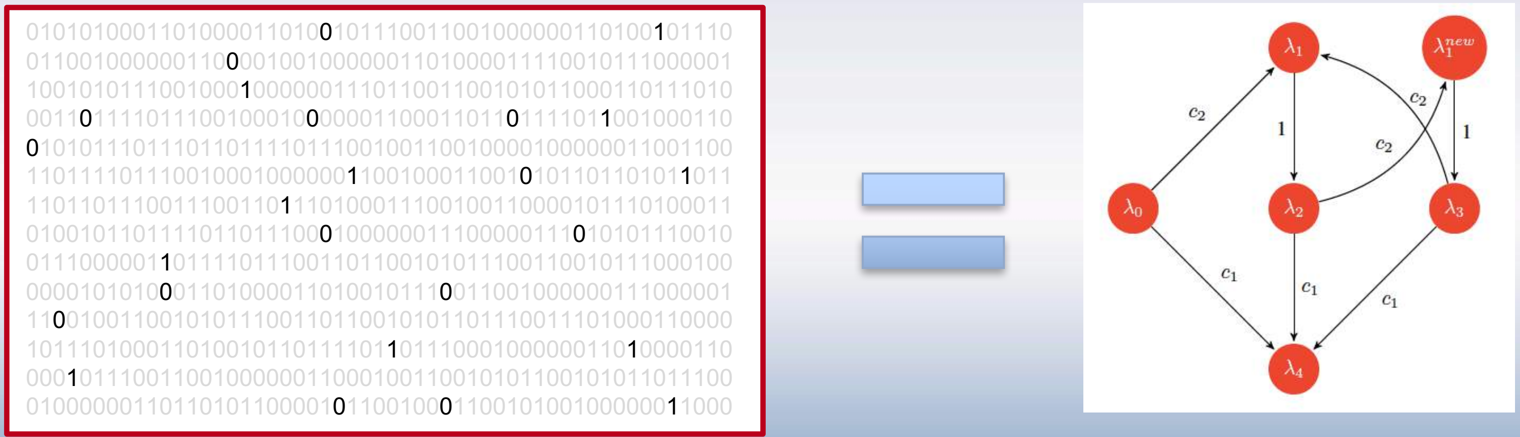
1. Spatial relation between figures
2. Similarity of shapes
3. Similarity of colors

VSA mapping structure:

[SPATIAL RELATION+ Placeholder_1⊗figure1+ Placeholder_2 ⊗figure2+ Similarity of shapes +Similarity of colors]

Hyperdimensional synthesis of automata model of a controlled object





"Associative Synthesis of Finite State Automata Model of a Controlled Object with Hyperdimensional Computing", Osipov, E. Kleyko, D. & Legalov, A. Submitted to IEEE IECON 2017.

Details HD synthesis

Given this

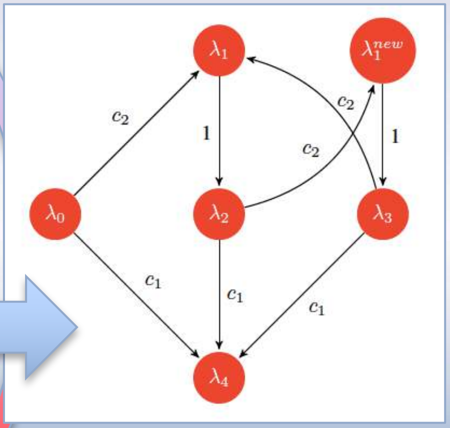
1) $T1 : \{\lambda_0 \xrightarrow[t_1]{c_1} \lambda_4\}$
2) $T2 : \{\lambda_0 \xrightarrow[t_2]{c_2} \lambda_1 \xrightarrow[t_3]{c_1} \lambda_2 \xrightarrow[t_4]{c_1} \lambda_4\}$
3) $T3 : \{\lambda_0 \xrightarrow[t_5]{c_2} \lambda_1 \xrightarrow[t_6]{c_2} \lambda_2 \xrightarrow[t_7]{c_2} \lambda_1 \xrightarrow[t_8]{c_1} \lambda_3 \xrightarrow[t_9]{c_1} \lambda_4\}$
4) $T4 : \{\lambda_0 \xrightarrow[t_{10}]{c_2} \lambda_1 \xrightarrow[t_{11}]{c_2} \lambda_2 \xrightarrow[t_{12}]{c_2} \lambda_1 \xrightarrow[t_{13}]{c_2} \lambda_3 \xrightarrow[t_{14}]{c_2} \lambda_1 \xrightarrow[t_{15}]{c_1} \lambda_2 \xrightarrow[t_{16}]{c_1} \lambda_4\}$

$$T_1 = \lambda_0^{<<} \oplus C_{0\to 4} \oplus \lambda_4 \text{ (created)}$$

t_i	Item Memory	Compositional memory		
		$\mathrm{STATE}_{cur}^{<<} \oplus \mathrm{STATE}_{next}$	$C_{cur\to next}$	$\mathrm{STATE}_{cur}^{<<} \oplus C_{cur\to next} \oplus \mathrm{STATE}_{next}$
t_0	EMPTY	EMPTY	EMPTY	EMPTY
t_1	$\lambda_0, c_1, \lambda_4$	$\lambda_0^{<<} \oplus \lambda_4$	$c_1 \notin C_{0\to4} : C_{0\to4} += c_1$	T_1 [illegible] $\lambda_0^{<<} \oplus C_{0\to4} \oplus \lambda_4$ (created)
t_2	c_2, λ_1	$\lambda_0^{<<} \oplus \lambda_1$	$c_2 \notin C_{0\to1} : C_{0\to1} += c_2$	T[illegible] $= \lambda_0^{<<} \oplus C_{0\to1} \oplus \lambda_1$ (created)
t_3	λ_2	$\lambda_1^{<<} \oplus \lambda_2$	$c_1 \notin C_{1\to2} : C_{1\to2} += c_1$	[illegible] $= \lambda_1^{<<} \oplus C_{1\to2} \oplus \lambda_2$ (created)
t_4		$\lambda_2^{<<} \oplus \lambda_4$	$c_1 \notin C_{2\to4} : C_{2\to4} += c_1$	[illegible]4 $= \lambda_2^{<<} \oplus C_{2\to4} \oplus \lambda_4$ (created)
t_5		$\lambda_0^{<<} \oplus \lambda_1$ (exists)	$c_2 \in C_{0\to1}$: (not updated)	(not updated)
t_6		$\lambda_1^{<<} \oplus \lambda_2$ (exists)	$c_2 \notin C_{1\to2} : C_{1\to2} += c_2$	[illegible]3 $= \lambda_1^{<<} \oplus C_{1\to1} \oplus \lambda_2$ (updated)
t_7		$\lambda_2^{<<} \oplus \lambda_1$	$c_2 \notin C_{2\to1} : C_{2\to1} += c_2$	$T_5 = \lambda_2^{<<} \oplus C_{2\to1} \oplus \lambda_1$ (created)
t_8	λ_3	$\lambda_1^{<<} \oplus \lambda_3$	$c_1 \notin C_{1\to3} : C_{1\to3} += c_1$	$T_6 = \lambda_1^{<<} \oplus C_{1\to3} \oplus \lambda_3$ (created)
t_9		$\lambda_3^{<<} \oplus \lambda_4$	$c_1 \notin C_{3\to4} : C_{3\to4} += c_1$	$T_7 = \lambda_3^{<<} \oplus C_{3\to4} \oplus \lambda_4$ (created)
t_{10}		$\lambda_0^{<<} \oplus \lambda_1$ (exists)	$c_2 \in C_{0\to1}$: (not updated)	(not updated)
t_{11}		$\lambda_1^{<<} \oplus \lambda_2$ (exists)	$c_2 \in C_{1\to2}$: (not updated)	(not updated)
t_{12}		$\lambda_2^{<<} \oplus \lambda_1$ (exists)	$c_2 \in C_{2\to1}$: (not updated)	(not updated)
t_{13}		$\lambda_1^{<<} \oplus \lambda_3$ (exists)	$c_2 \notin C_{1\to3} : C_{1\to3} += c_2$	T[illegible] $= \lambda_1^{<<} \oplus C_{1\to3} \oplus \lambda_3$ (updated)
t_{14}		$\lambda_3^{<<} \oplus \lambda_1$	$c_2 \notin C_{3\to1} : C_{3\to1} += c_2$	T_8 [illegible] $\lambda_3^{<<} \oplus C_{3\to1} \oplus \lambda_1$ (created)
t_{15}		$\lambda_1^{<<} \oplus \lambda_2$ (exists)	$c_1 \in C_{1\to2}$: (not updated)	(not updated)
t_{16}		$\lambda_2^{<<} \oplus \lambda_4$ (exists)	$c_1 \in C_{2\to4}$: (not updated)	(not updated)

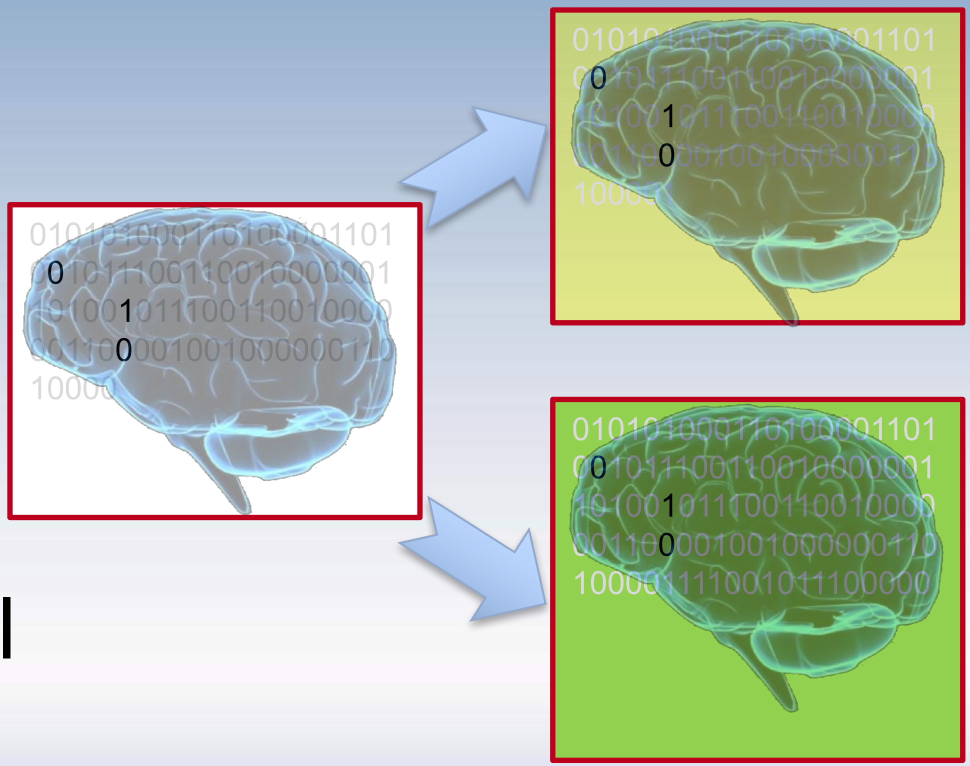
Majority sum

Σ

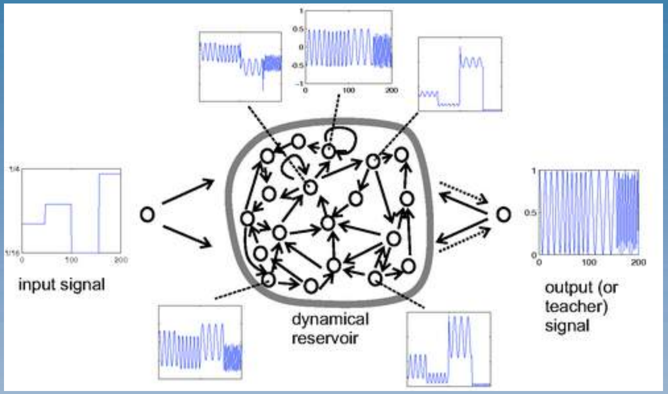


A foretaste of future developments

- A fully distributed modeling of automation processes down to the nodes
- A hyperdimensional execution environment

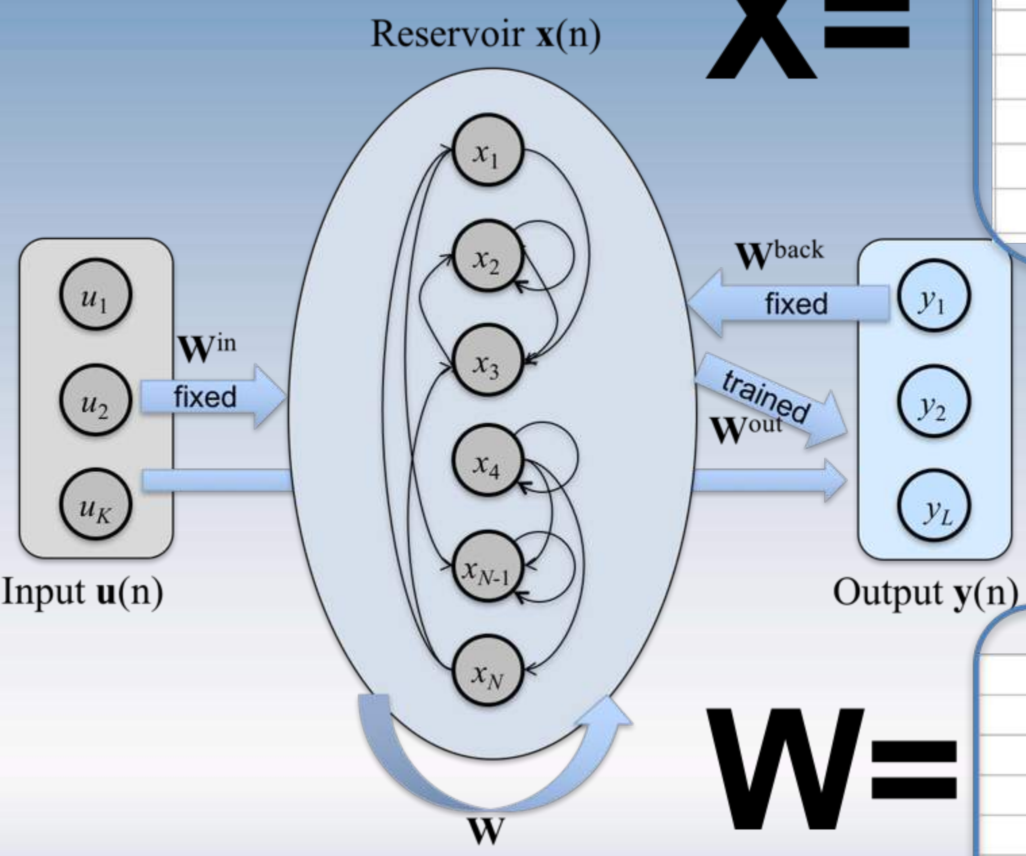


Integer Echo State Networks: Hyperdimensional reservoir computing



- Recurrent Neural Networks (RNNs) have been a prominent concept within Artificial Intelligence and Neural Networks.
- RNNs are good for temporal tasks because they are able to memorize historic inputs.
  - speech recognition
- Hard to train
- Reservoir Computing is an alternative to RNNs training.
  - a more accurate model of how recurrent topologies in the brain work, as opposed to feed-forward models
- RC = A dynamical system operating at the “edge of chaos” + a high-dimensional projection of the input + nonlinearity + a fading memory.
- Training = learning only connections to the last readout layer while keeping the other connections to be fixed.

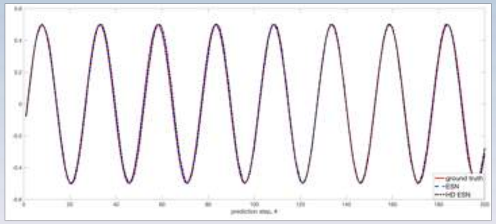
Echo State Networks (ESN)



X=

−0.1203	−0.1753	−0.2492	−0.1953	−0.2895
−0.2553	−0.2576	−0.2578	−0.3114	−0.2241
−0.0977	−0.2840	−0.3910	−0.4023	−0.4288
−0.0873	−0.0512	−0.0860	0.0240	0.0974
0.0679	0.1130	0.3028	0.3411	0.3518
0.0112	0.0287	0.0118	−0.0166	−0.0590
−0.0557	0.0018	−0.0369	−0.0221	0.0180
0.2321	0.4240	0.4554	0.4658	0.4944
−0.1397	0.0964	−0.0387	−0.0046	−0.1063
0.1740	0.2162	0.1590	0.1610	0.1791

Y=



W=

0	0	0	0	0.9754	0
0	−0.6574	0.5498	0	0	0
0	0.0803	0	0	0	0
0	0	0	−0.4422	−0.4987	0.2573
0.1385	0	0	0	−0.9750	0.0731
0	0	−0.1994	0	0.6850	0

$$\mathbf{x}(n)=\tanh(\mathbf{W}\mathbf{x}(n-1)+\mathbf{W}^{in}\mathbf{u}(n)+\mathbf{W}^{back}\mathbf{y}(n-1))$$
$$\mathbf{y}(n)=\tanh(\mathbf{W}^{out}[\mathbf{x}(n);\mathbf{u}(n)])$$

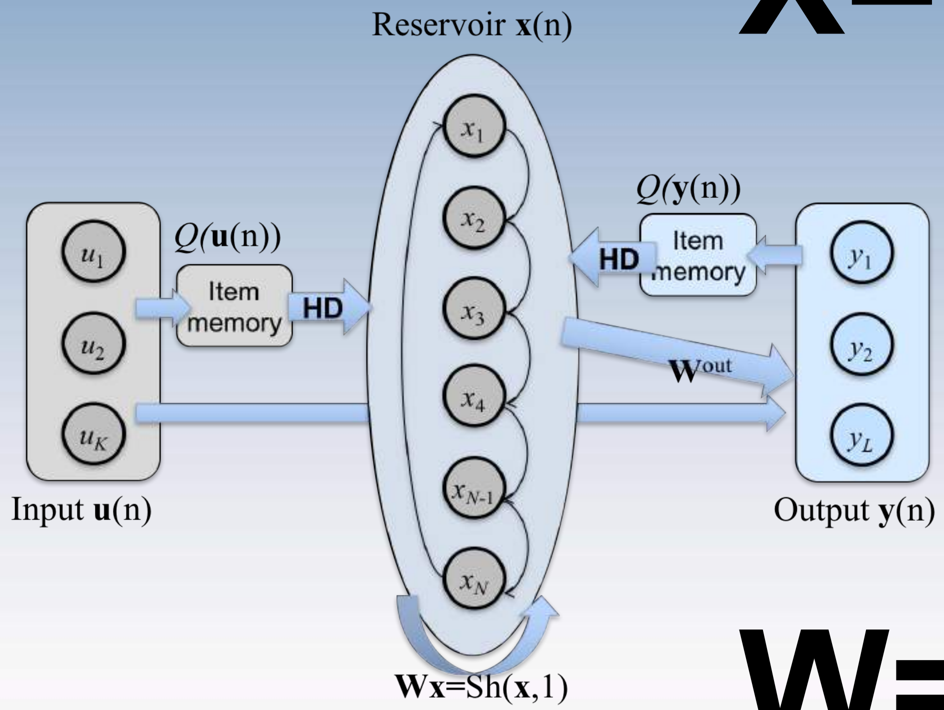
Achieving "binarization"

- **We want** to convert major operations on reservoir to the domain of (less than one-byte) integers

$$\mathbf{x}(n)=\tanh(\mathbf{W}\mathbf{x}(n-1)+\mathbf{W}^{in}\mathbf{u}(n)+\mathbf{W}^{back}\mathbf{y}(n-1))$$

- We want Inputs to reservoir $\mathbf{W}^{in}\mathbf{u}(n)$ and $\mathbf{W}^{back}\mathbf{y}(n-1)$ **u(n)** have integer values
- The result of projection of the reservoir on itself $\mathbf{W}\mathbf{x}(n-1)$ should have integer values
- Finally after nonlinear function tanh() the result should have integer values

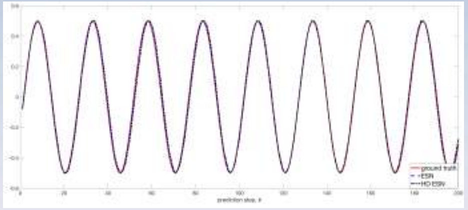
intESN



X=

0	−2	−4	−2	−4
−1	−1	−3	−5	−3
−6	−2	−2	−4	−6
−5	−5	−1	−1	−3
−8	−6	−6	−2	−2
−7	−7	−5	−5	−1
−8	−8	−8	−6	−6
−9	−9	−9	−9	−7
−10	−10	−10	−10	−10
−9	−9	−9	−9	−10

Y=



W=

0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	0	0	0	1	0
0	0	0	0	0	1
1	0	0	0	0	0

$$\mathbf{x}(n)=\mathrm{clipping}(\mathrm{Sh}(\mathbf{x}(n-1),1)+\mathrm{item}(\mathbf{u}(n))+\mathrm{item}(\mathbf{y}(n-1)))$$

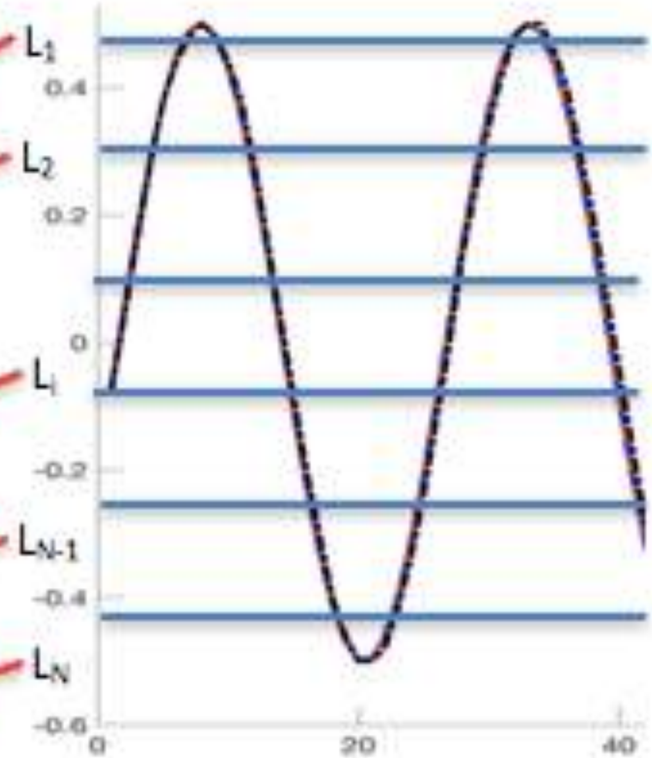
$$\mathbf{y}(n)=\tanh(\mathbf{W}^{out}[\mathbf{x}(n);\ \mathrm{item}(\mathbf{u}(n))])$$

This step is same as before!

Discretization - projection – item memory

ITEM MEMORY

Item	BDDR vector
L_1	-1 +1 +1 +1 -1 +1 – -1 -1 … +1
L_2	+1 -1 -1 +1 +1 -1 – -1 +1 … -1
…	…
L_i	-1 -1 -1 +1 +1 +1 – -1 +1 … +1
…	…
L_{N-1}	-1 +1 +1 -1 -1 +1 – -1 -1 … +1
L_N	-1 -1 -1 +1 -1 +1 – +1 -1 … -1



Cyclic shift as a recurrent connection

- Cyclic shift is used instead of matrix-vector multiplication

$\mathbf{x}$ =

0	0	1	1	0	0	0	1	1	1

Sh($\mathbf{x}$,1) =

1	0	0	1	1	0	0	0	1	1

- Cyclic shift can be represented as a recurrent connection matrix **W**
- This matrix is extremely sparse
- The result of matrix-vector multiplication is the same as for cyclic shift

W =

0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	1	0

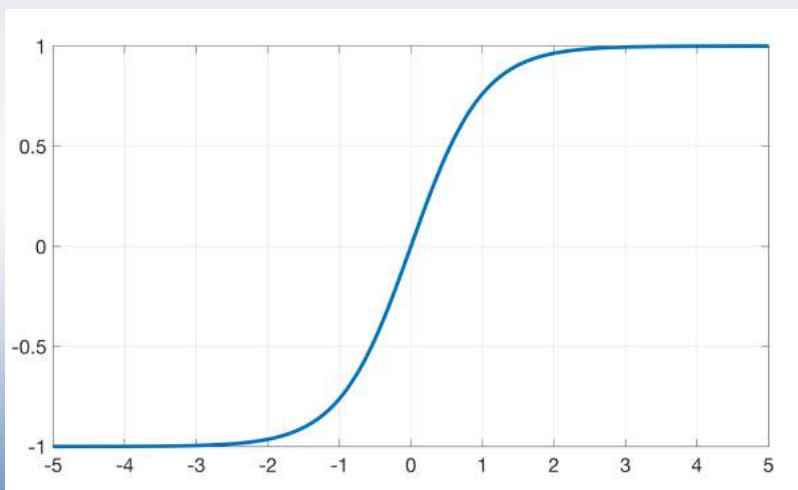
Wx =

1	0	0	1	1	0	0	0	1	1

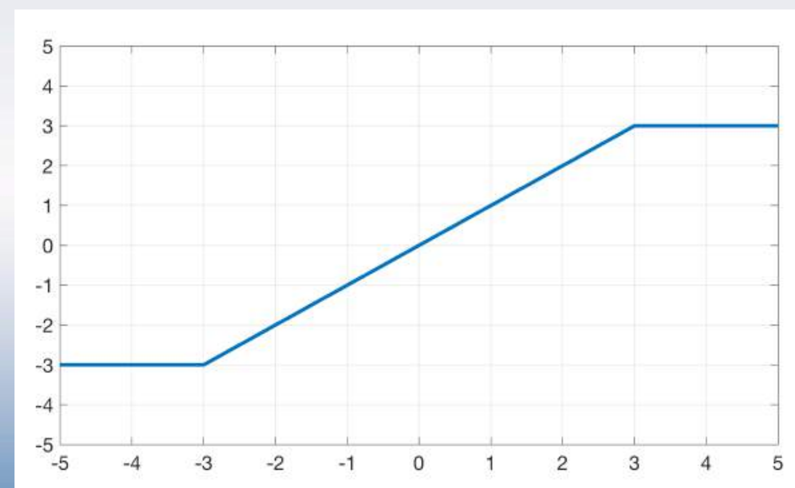
Clipping as a nonlinearity function

- One of the key operations in ESN is a nonlinear function tanh(x) applied on the reservoir at every step.
- Two key functions of tanh(x):
  - Keep values of the reservoir in the restricted range
  - Introduce nonlinearity
- We swap tanh(x) with the clipping function which clips all the values above certain threshold k

tanh(**x**)

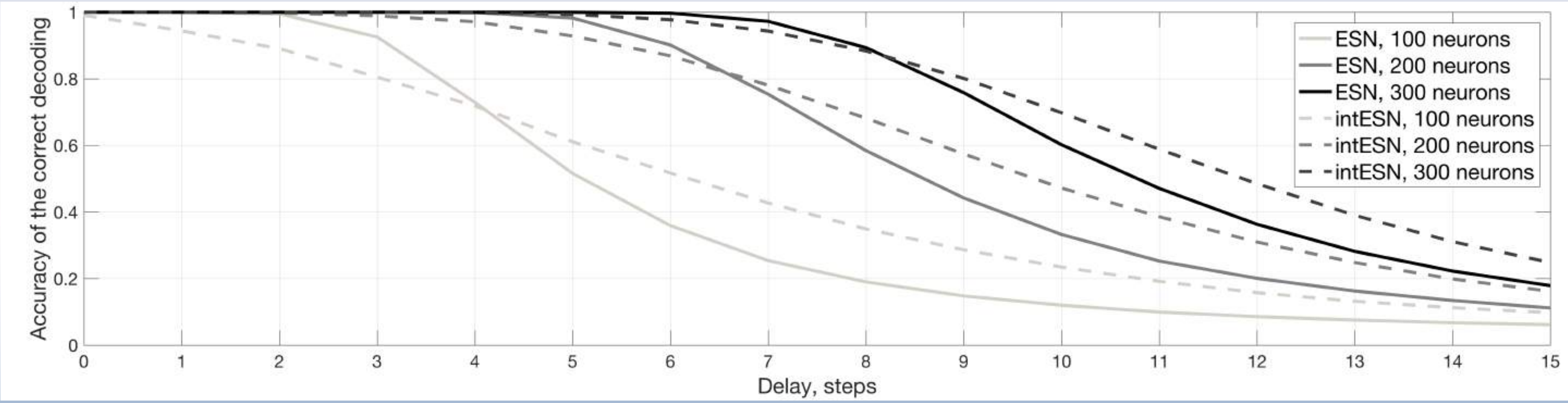


clipping(**x**), k=3



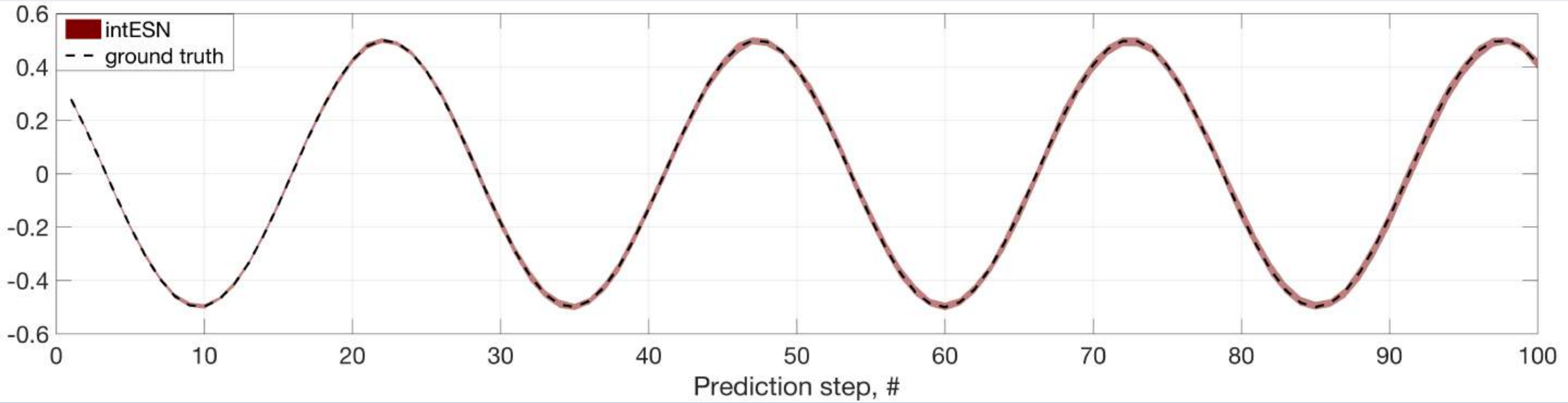
Comparison of short-term memory

- Sequence recall task. Network continuously stores a sequence of tokens.
- One token is presented as input each time step
- Dictionary contains 27 unique tokens
- At the recall stage, the network uses the content of its reservoir to retrieve the token stored *d* steps ago, where *d* denotes delay



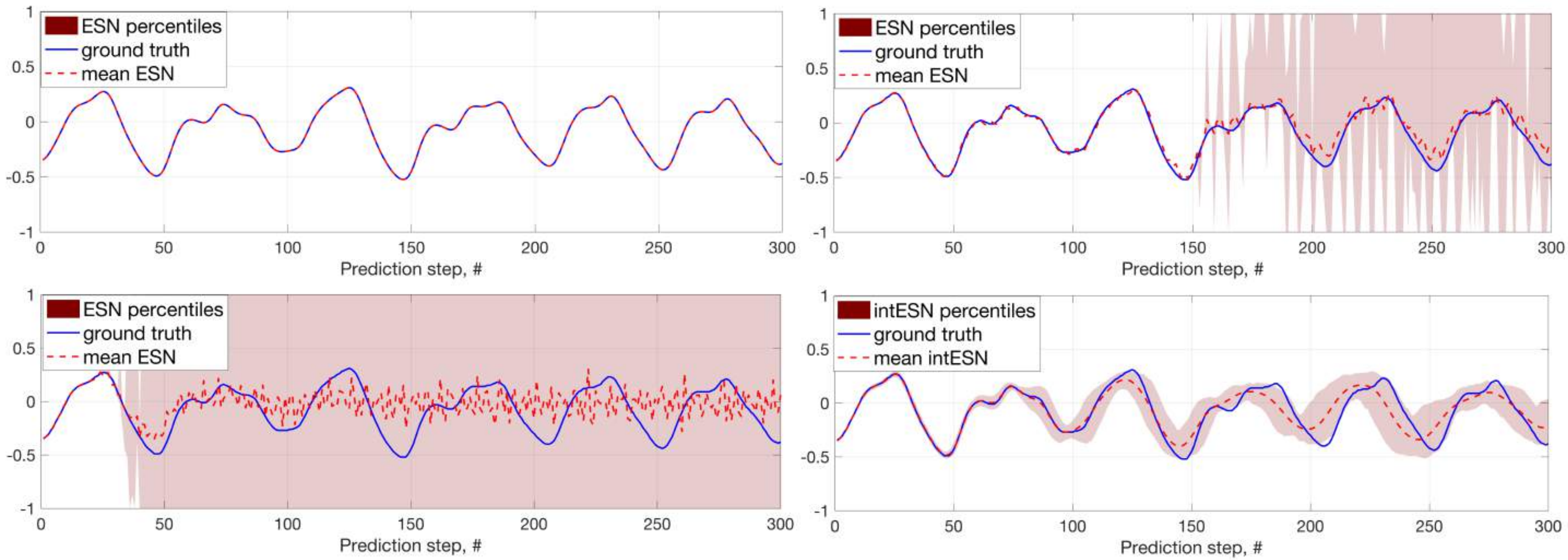
Sinusoidal generator

- An example of a learning simple dynamic system with the constant cyclic behavior. Function form $y(n) = 0.5 \sin(n/4)$
- The network projected the activations of the output layer back to the reservoir
- The output layer had only one neuron
- In the operating phase, the network acted as the generator of the signal feeding its previous prediction (at time n-1) back to the reservoir.



Mackey-Glass Prediction

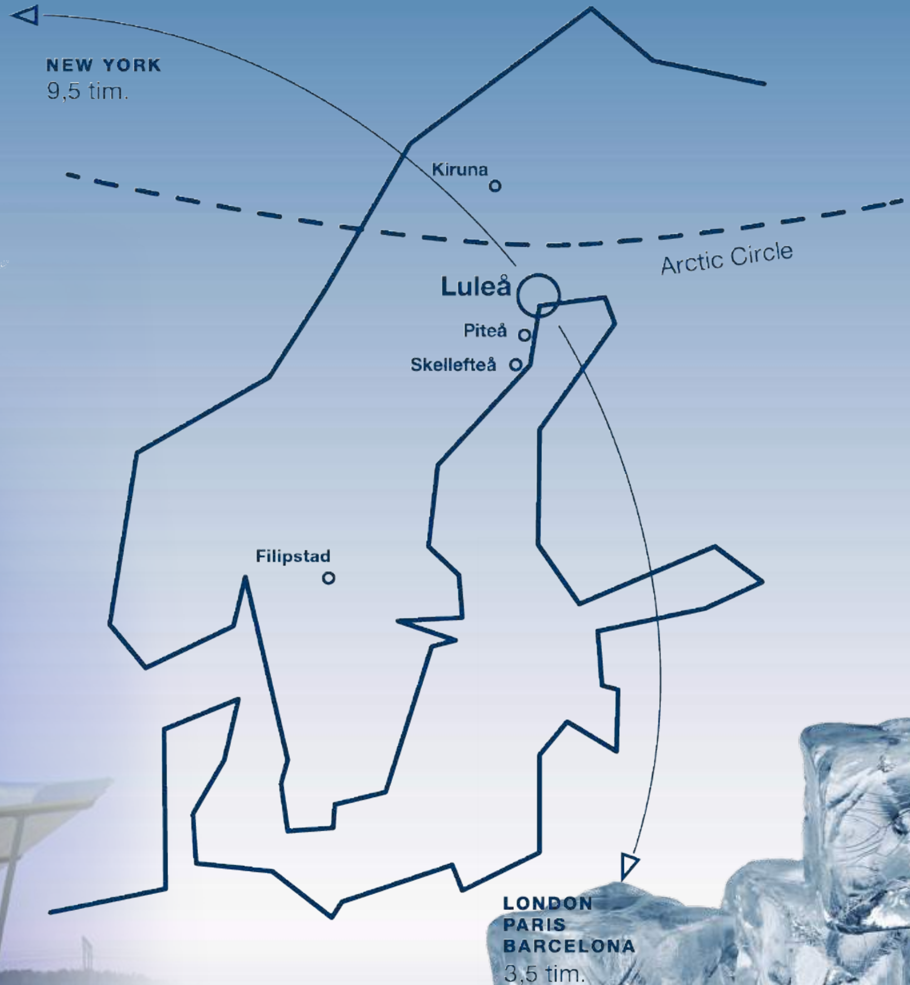
- A Mackey-Glass series is generated by the nonlinear time delay differential equation
- Panels show different cases related to the quantization of the data



Typical performance

- Due to the integer approximation the accuracy of intESN is to a certain degree lower than that of traditional ESN.
- Still very good
- Extremely attractive for:
  - Size, weight, and power constrained devices
  - General area of approximate computing (where errors and approximations are acceptable as long as the outcomes have a well-defined statistical behavior.

Questions – comments – reflections- ideas



Evgeny Osipov, Denis Kleyko
Department CSESE
Luleå University of Technology

LULEÅ UNIVERSITY OF TECHNOLOGY