Intelligent Robotics

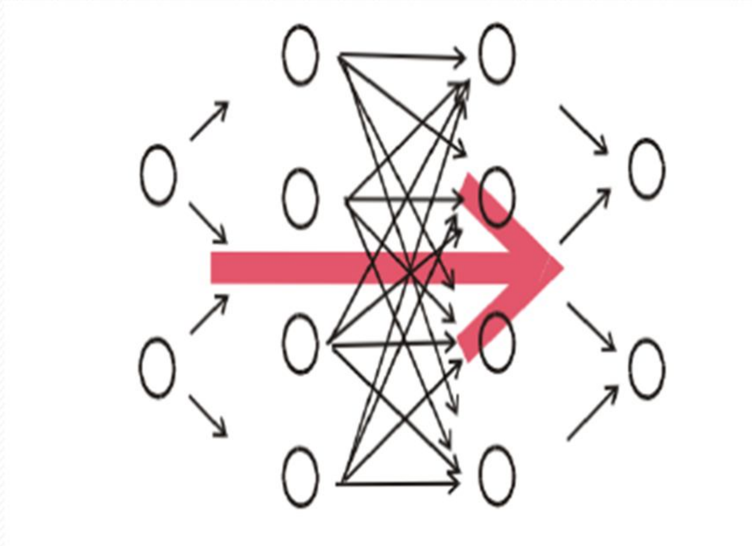
RNN for Object Classification

Presented by:-
Surender Kumar
Matr.Nr. 6519753
3kumar@informatik.uni-hamburg.de

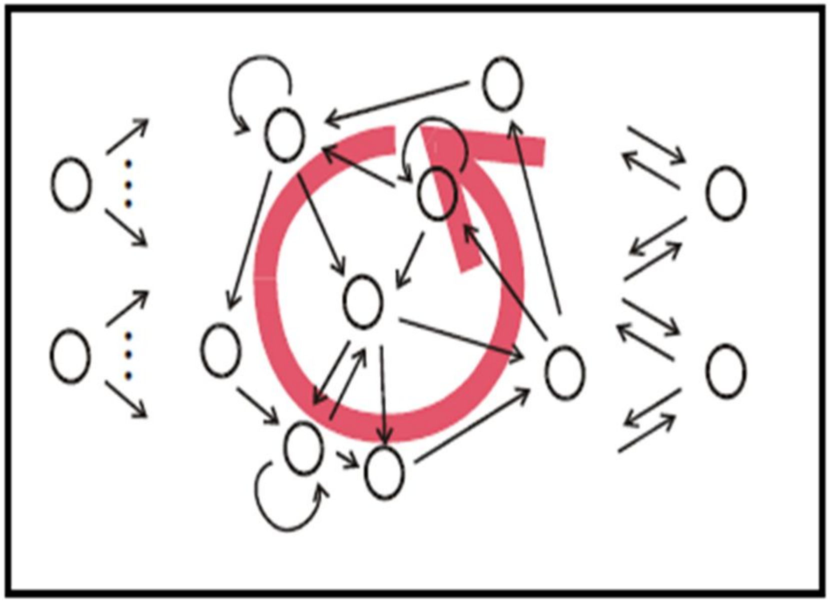
INTRODUCTION

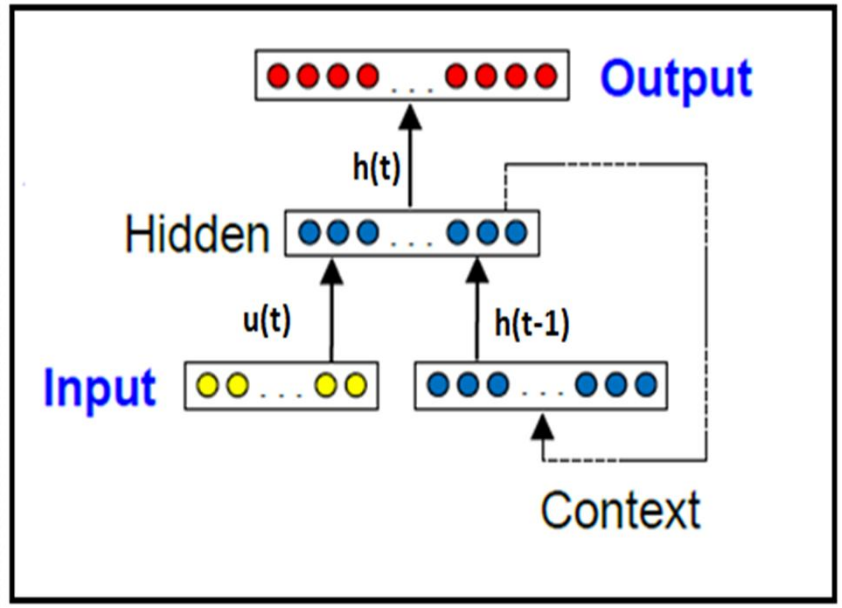
- **Mainly 2 types of neural network**
  - **Feed Forward Neural Network**
  - **Recurrent Neural Network**

- **Feed Forward Neural Network:-**
  - activation is "piped" through the network from input units to output units (from left to right)
  - No cycle and the layers are clear
  - Not capable of processing time series
  - Back propagation is used for training.



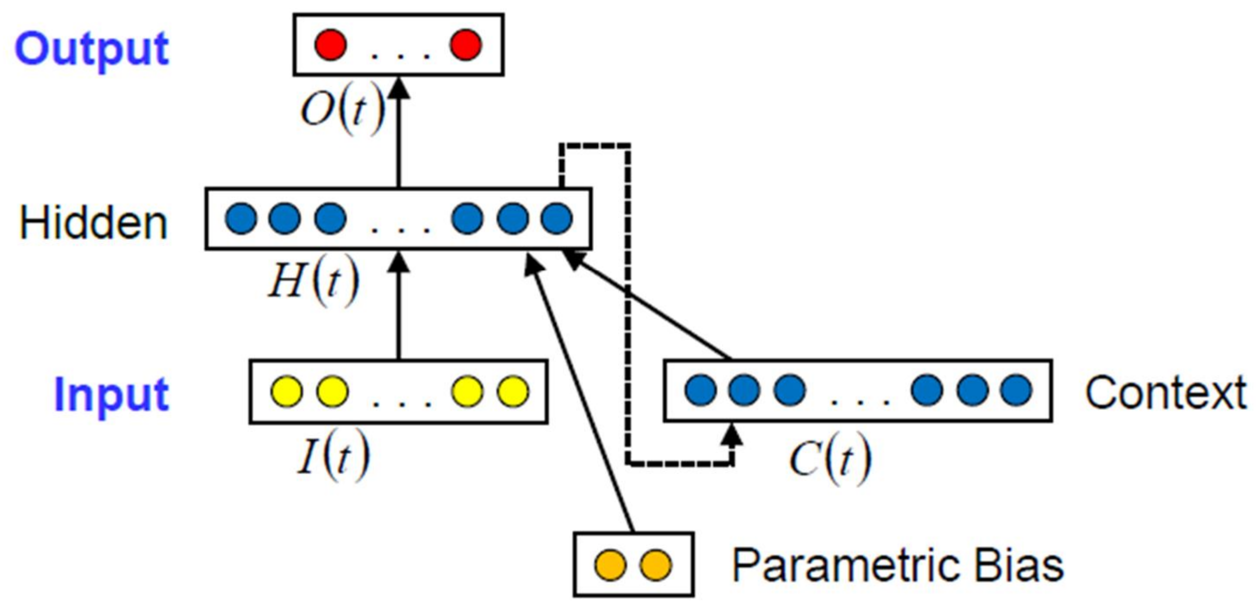
Elman type RNN





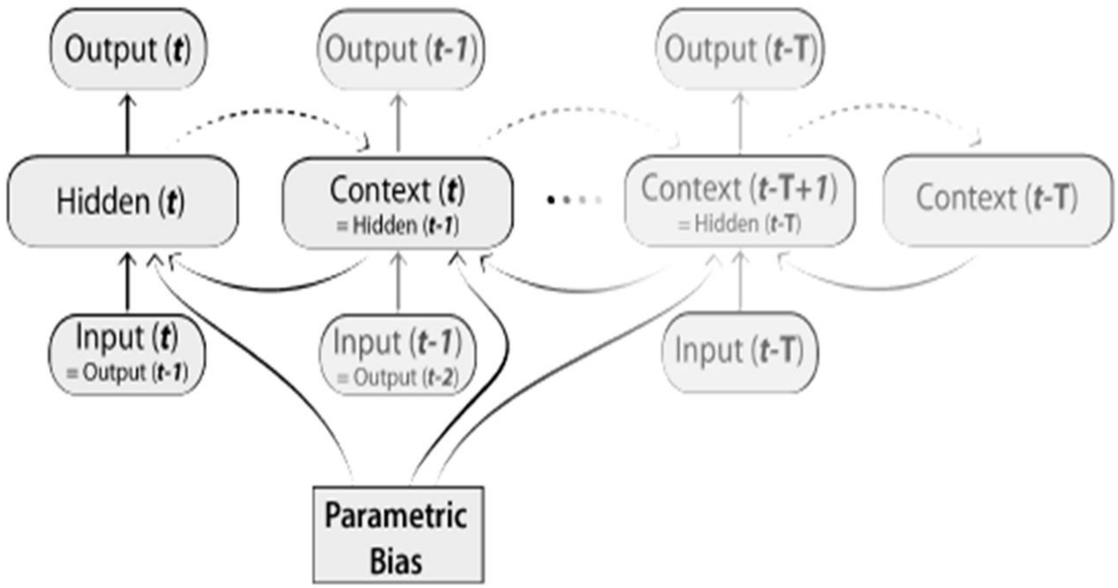
- Elman Type network models non-linear dynamical system
- Total input to hidden layer:
  - u(t) : current input at time step t.
  - h(t-1) : activation of hidden layer at time step t-1.
- Context unit activations represent the internal state of the network

RNNPB Overview



- Elman Type network model the non-linear dynamical system.
- PB vector acts as the bifurcation parameters of nonlinear dynamical systems.
- Network generates ***nonlinear mappings*** between the **parametric *bias*** and corresponding ***sequences***
- RNNPB can encode the several no. of dynamical patterns.

RNNPB unfolded over time



- The same ***bias*** is influencing the activation in every time step, but is ***self-organised*** by back-propagation for every sequence.

RNNPB Features

- **Record Patterns:-**

- PB Vector Maps Spatio-temporal Patterns.
- PB Vector for each pattern is self-determined in unsupervised way.
- Therefore, similar sequences are clustered together and distinguishable sequence are located further apart.

- **Reconstruct Patterns :-**

- Once PB Vector is learned, it can be used for the generation of the stored patterns
- **HOW** ?
  - Network is operated in closed loop.
  - The PB values are 'clamped' to a previously learned value.
  - *Forward Pass*: Network starts with initial input I(0).

    output at any time t serves as an input at time t+1

RNNPB Features

- **Recognize Patterns :-**

- Patterns are recognized by corresponding PB value.

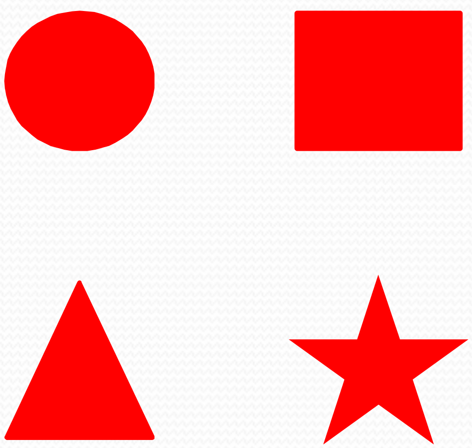
- **HOW** ?
  - Observed pattern is fed to the network.
  - No updates made in connection weights.
  - Only PB values are accumulated with constant learning rate.
  - PB Vector obtained is compared to the PB vector obtained during the training.

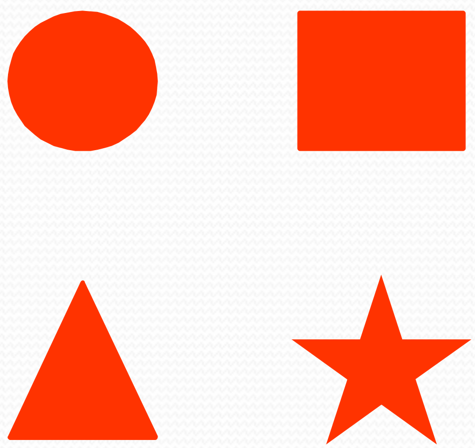
Experiment Scenario (Kleesiek 2011)

- **Aim**:
  - NAO robot has to distinguish among 8 toy bricks.

- **Toy Bricks**:-

Heavy toy bricks ***Light toy bricks***





Data Acquisition

- A time series contain s 14 sensor values for each modality.
- In each single trial:-

Action 1: the toy brick is rotated by 45.8 degree back-and-forth

Repetition: 2 times

Data: after action, raw image of the lower camera of the Nao robot is captured

Action 2: lifting the toy up-and-down(altering pitch of the shoulder joint by 11.5^{o})

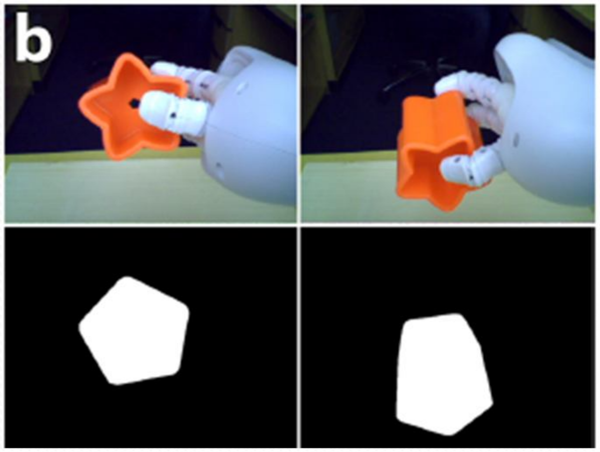
Repetition: 3 times

Data: for entire movement interval ,electric current of the shoulder pitch servo motor is recorded constantly (sampling frequency 10 hz).

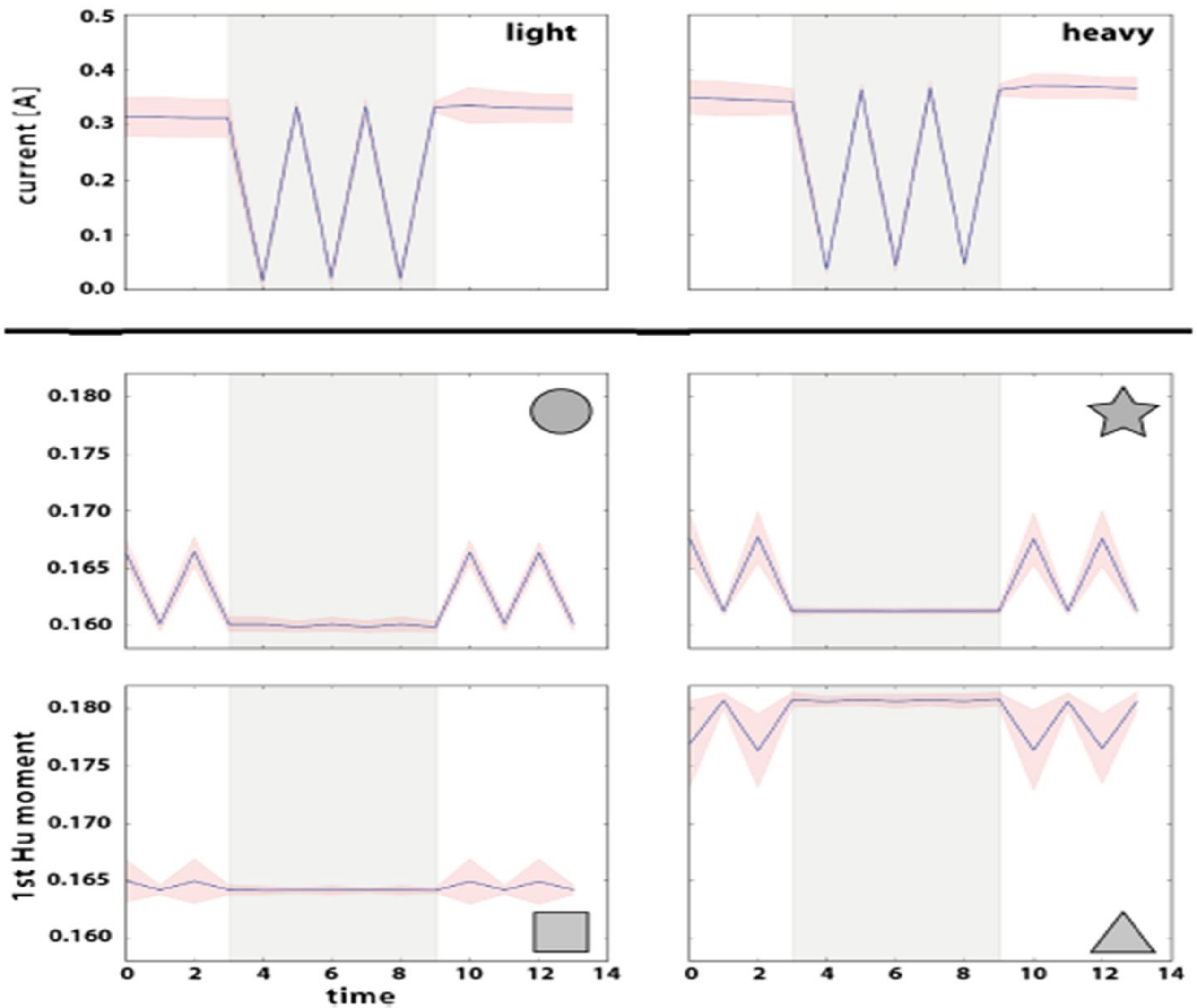
- In the similar way, 10 single time series are recorded for every toy brick i.e. for all shape and both weight categories.
- Thus, in total we will have 80 bi-modal time series.

Data Processing

- **Proprioceptive processing**:
  - Mean values for the time interval b/w movements.
- **Visual Processing**:
  - Done using OpenCV.
  - Raw image converted to binary using color threshold.
  - Convex hull is computed
  - Contour belonging to toy bricks is extracted
  - Calculate first Hu moment.
  - Scale visual measurement in interval [-0.5, 0.5]



Training Data

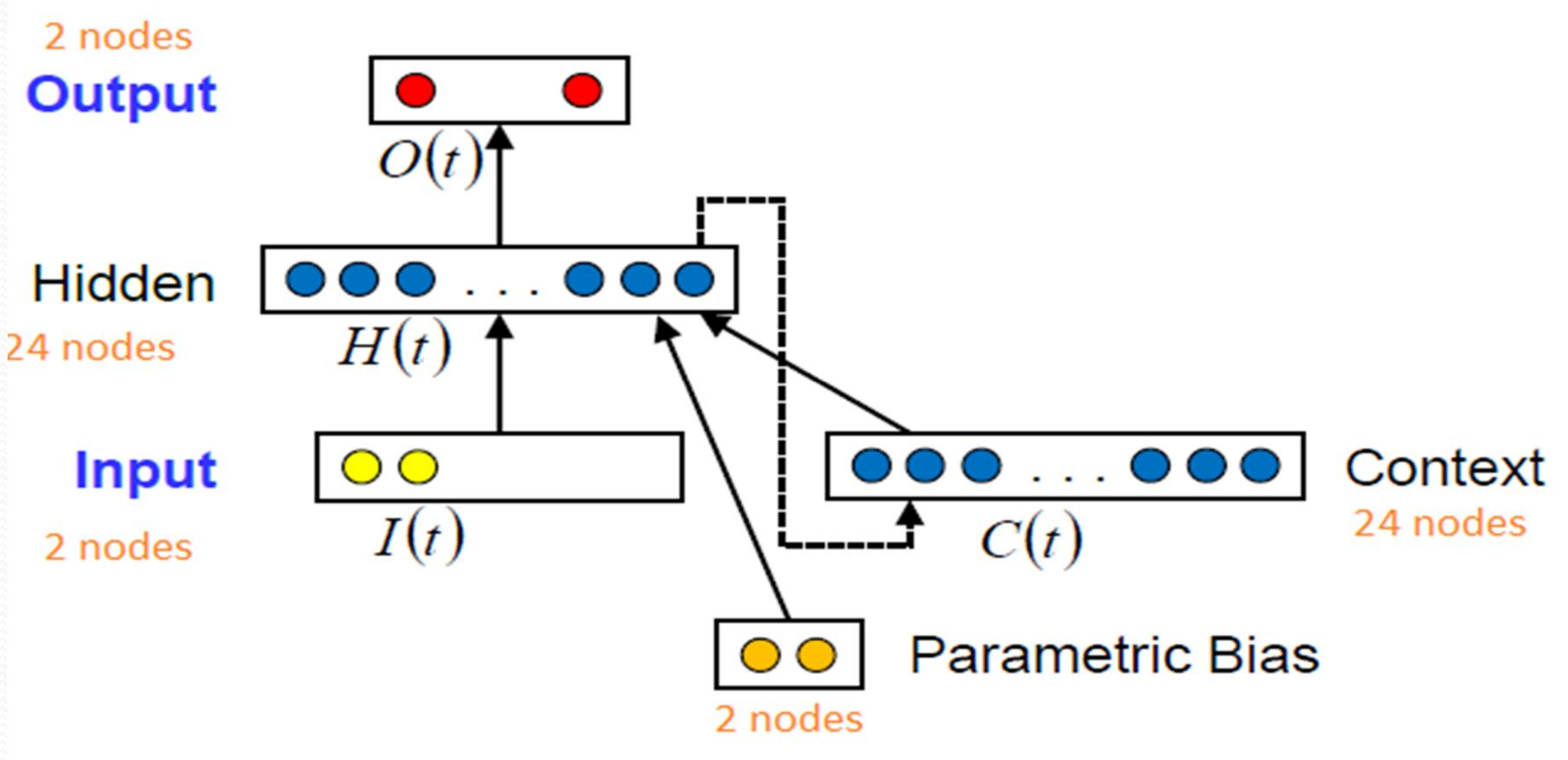


Proprioceptive Prototype

Visual Prototype

Network Parameters

- **Based in Empirical trials**:-

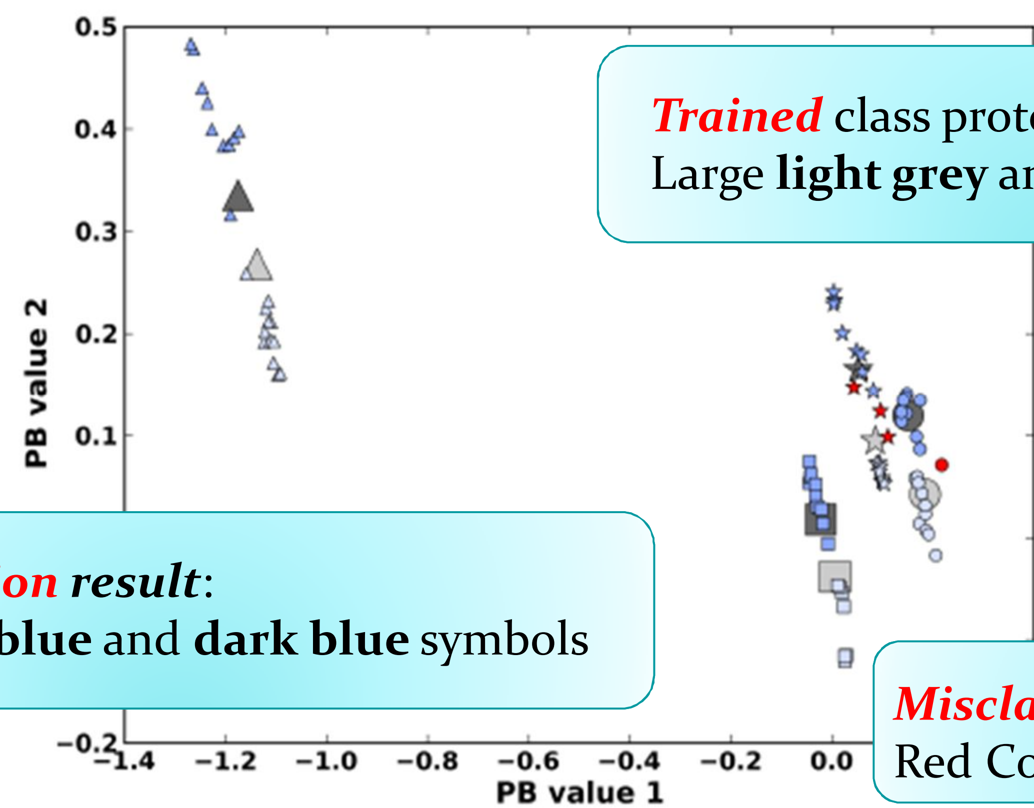


- **For recognition mode**:-
  - *Learning rate for PB values*: 0.1

Experiment 1: Using all objects for training

- **<u>Classification</u>**:-

- Light Colors :- Light weight object
- **Dark Colors** :- Heavy weight object



Trained class prototypes:
Large **light grey** and **dark grey** symbols

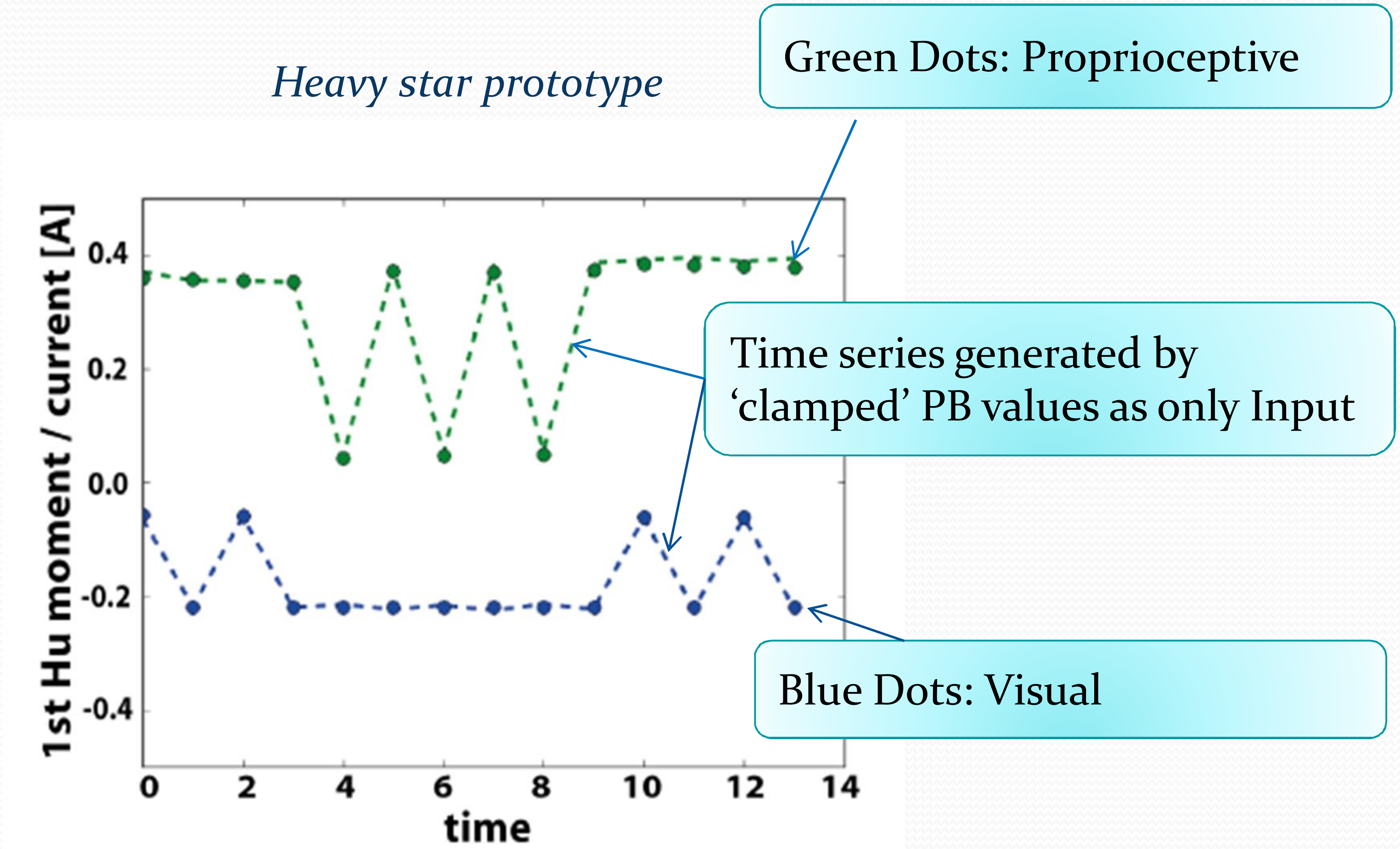
Classification result:
Small **light blue** and **dark blue** symbols

Misclassified Objects:
Red Color (4 out of 80) i.e 5%

Experiment 1: Using all objects for training

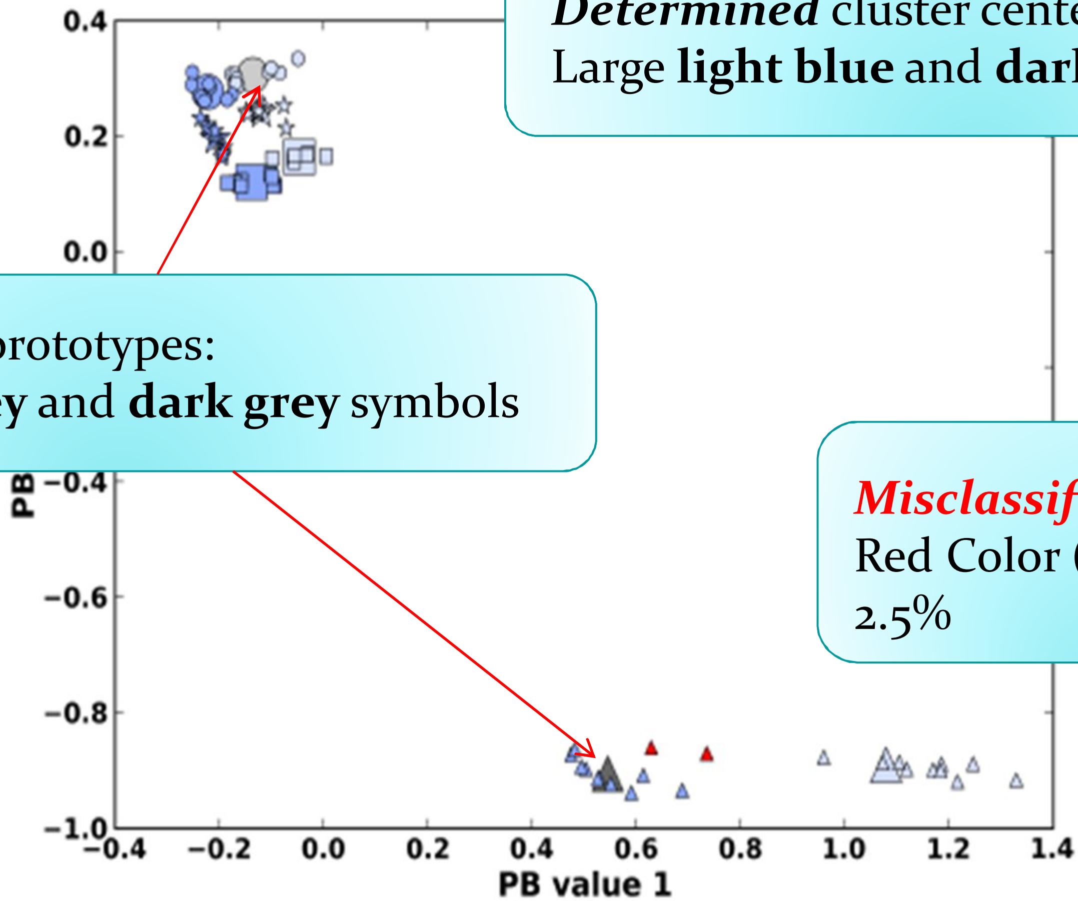
- **Retrieval and Generation**:-

Heavy star prototype



Experiment 2: Using 2 objects for training

- **Classification**:-



Determined cluster centers:
Large **light blue** and **dark blue** symbols

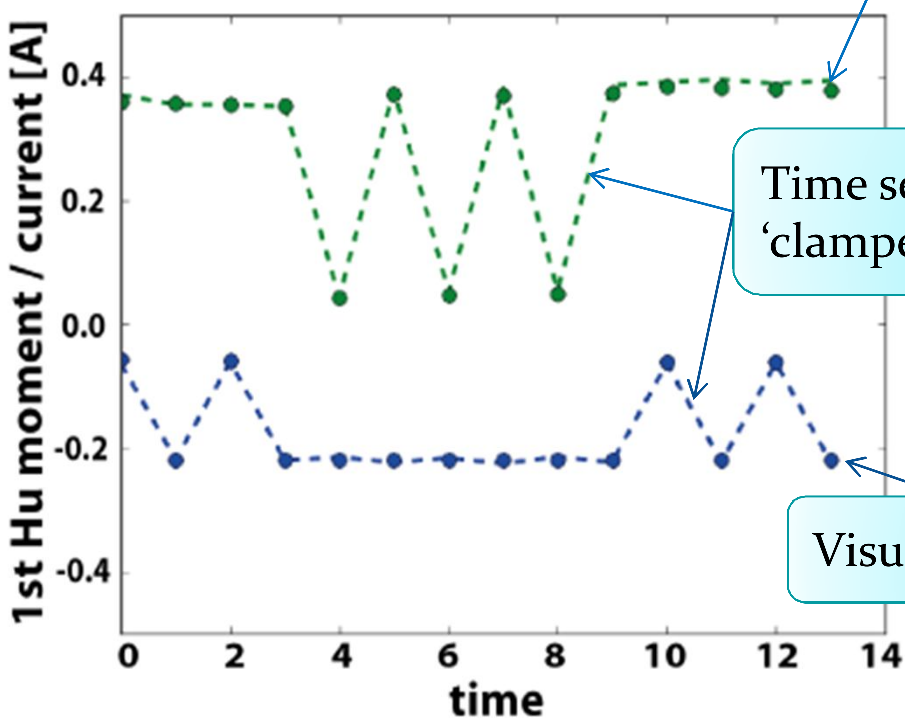
Trained class prototypes:
Large **light grey** and **dark grey** symbols

Misclassified Objects:
Red Color (2 out of 80)
2.5%

Experiment 1: Using all objects for training

- **<u>Retrieval and Generation</u>**:-

Heavy star prototype

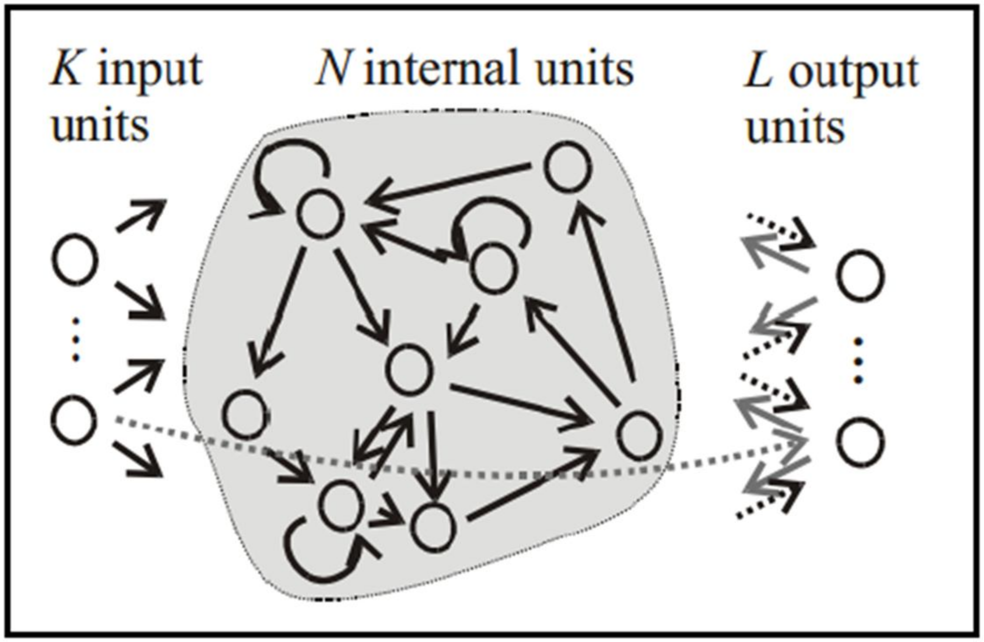


Proprioceptive sample points

Time series generated by 'clamped' PB values as only Input

Visual Sample Points

ESN (Herbert Jaeger in 2001)



- Hidden layer also known as reservoir :-
  - Non-linear expansion of the input
  - Act as a memory unit of input at the same time
  - The past states echoes in the network, even when no input.

Training ESN

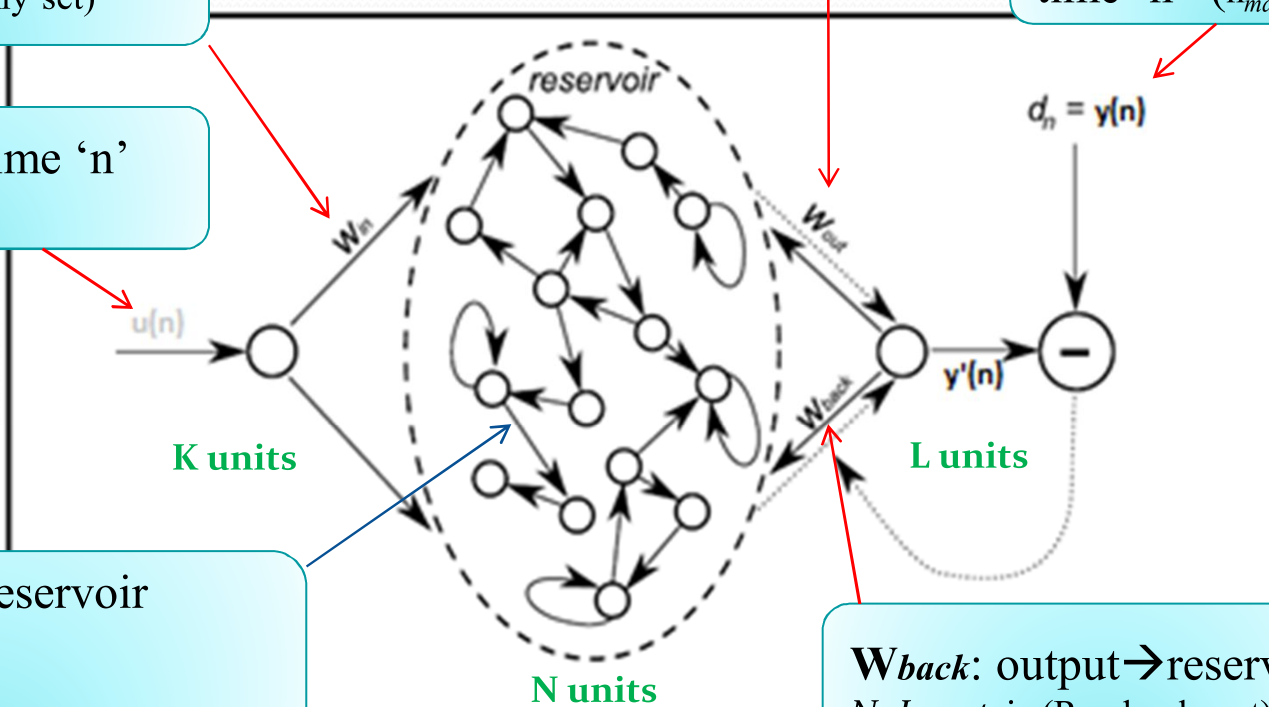
- **Training Data**= {(u(n),y(n)) } where 1< n < n_{max} (length of time series)
- **Network Output** = y'(n)

W*out*: reservoir→output
L×(K+N) matrix (will be learnt)

W*in*: Input →Reservoir
N×K matrix (Randomly set)

y(n): desired output at time 'n' ($n_{max} \times L$ *matrix*)

u(n): Input at time 'n'
$n_{max} \times K$ matrix



W*r*: reservoir→reservoir
N×N matrix
(Spectral radius < 1)
(Sparse connectivity)

W*back*: output→reservoir
N×L matrix (Randomly set)

Training ESN (2)

- **Output of the reservoir :**
  - Reservoir generates a sequence ***x(n)*** of N-dimensional reservoir states.
  - ***x(n)*** is the non-linear high dimensional expansion of the input signals.
  - Each component signal ***x'(n)*** contributed by reservoir unit is non-linear transform of driving input.

- **Formula used for calculation :-**
  - $\mathbf{x(n+1)} = \boldsymbol{f}\left(\mathbf{W_r . x(n) + W_{in} .u(n+1) + W_{back} . y(n)}\right)$ where,

    f is the activation function

Training ESN (4)

- **Output state:-**
  - concatenation of the reservoir and input states at time step 'n'. i.e. $\mathbf{z}(n)=[\mathbf{x}(n); \mathbf{u}(n)]$
  - State collection matrix:-
    - $\mathbf{S} = z(n)$ where **S** is of size $n_{max} \times (N+K)$
  - Teacher output collection matrix:-
    - $D = y(n)$ [row-wise] where **D is** of size $n_{max} \times L$.
  - $\mathbf{y}'(\mathbf{n}) = g(\mathbf{W}_{out}\, z(n))$ where g is output activation function

- **Learning of Output weights**:-
  - **W*out*** *:* linear regression weights of **y**(n) on the reservoir output of harvested extended states **y**'(n).
  - The weights **W*out***, should minimize the mean squared error between **y'(n)** and **y(n)** .
  - Let $\mathbf{R} = \mathbf{S}^T\mathbf{S}$ be the correlation matrix of the extended reservoir state and,
  - $\mathbf{P} = \mathbf{S}^T\mathbf{D}$ be the cross-correlation matrix of the states vs. the desired outputs.
  - Then, using Weiner-Hopf solution:

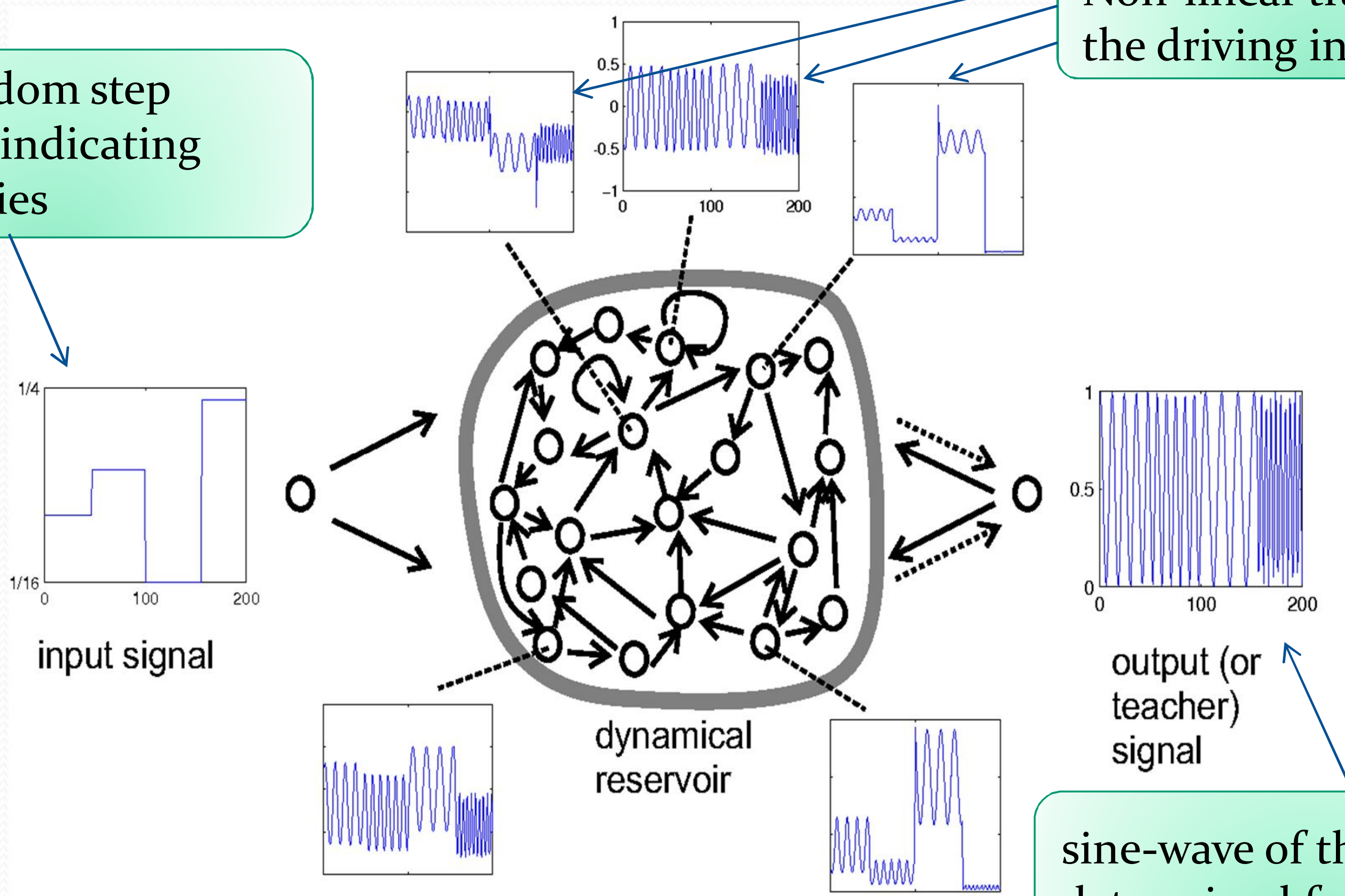
$$W_{out} = (\mathbf{R}^{-1}\mathbf{P})^T$$

$$W_{out} = (\mathbf{S}^T\mathbf{S})^{-1}\mathbf{S}^T\mathbf{D}$$

Application 1: Tunable frequency generator

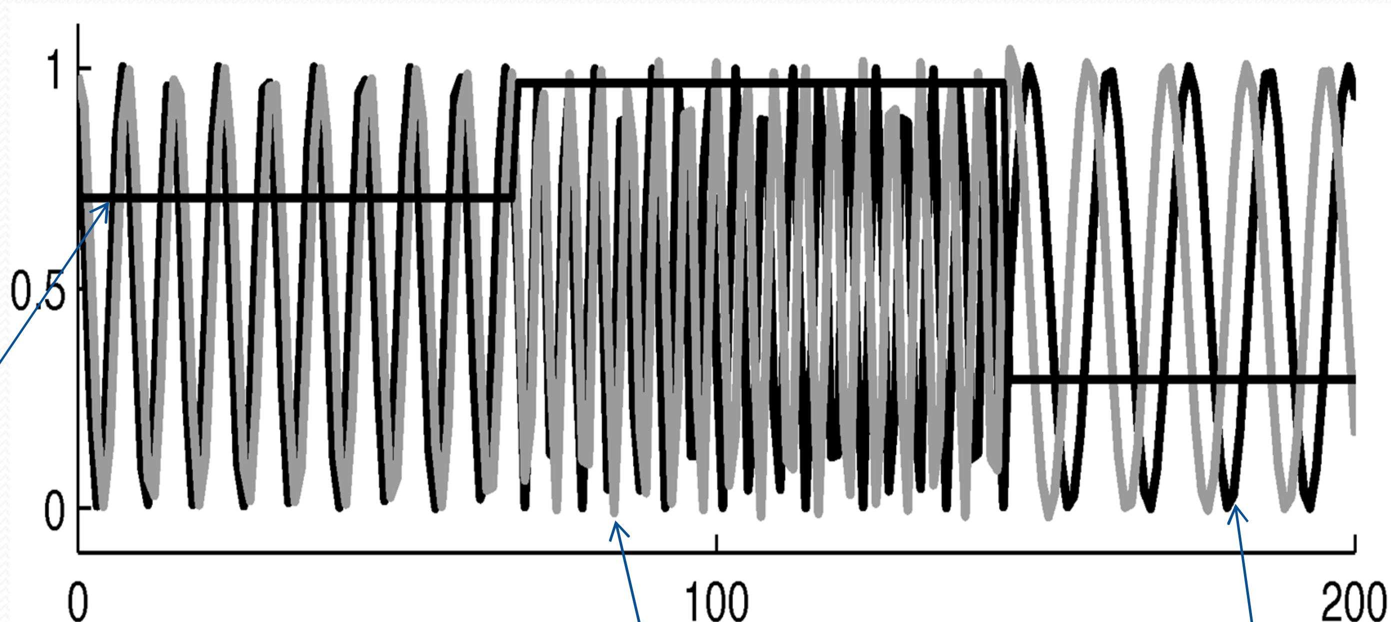
Non-linear transform of the driving input

Slow random step function indicating frequencies



sine-wave of the input determined frequency.

Application 1: Results



Slow random step function indicating frequencies

Desired sine wave of the input-determined frequency.

Actual sine wave of the input-determined frequency.

Thank You