Fall detections in humanoid walk patterns using Reservoir based control architectures

5th National Conference on "Control Architecture of Robots"

Rahul Kanoi & Cédric Hartland

rahulkanoi2006@vit.ac.in - cedric.hartland@efrei.fr

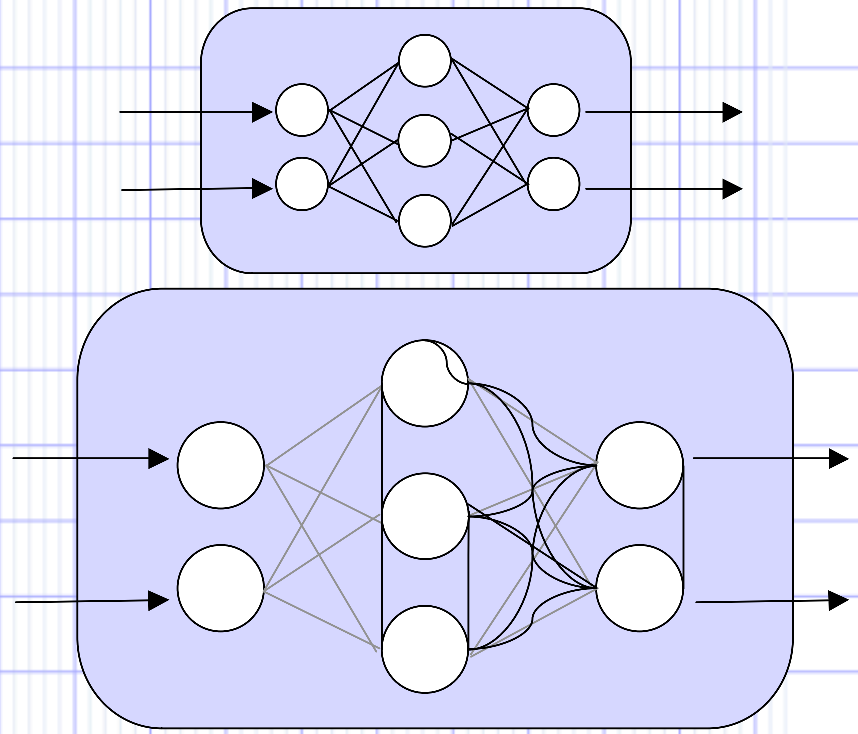
Mai 2010

Summary

- Neural networks for robotic control
  - Recurrent neural networks
  - From robotic controller to middleware
- Experiments
  - Experimental setup
  - Parameters
  - Results
- Discussion
- Conclusion

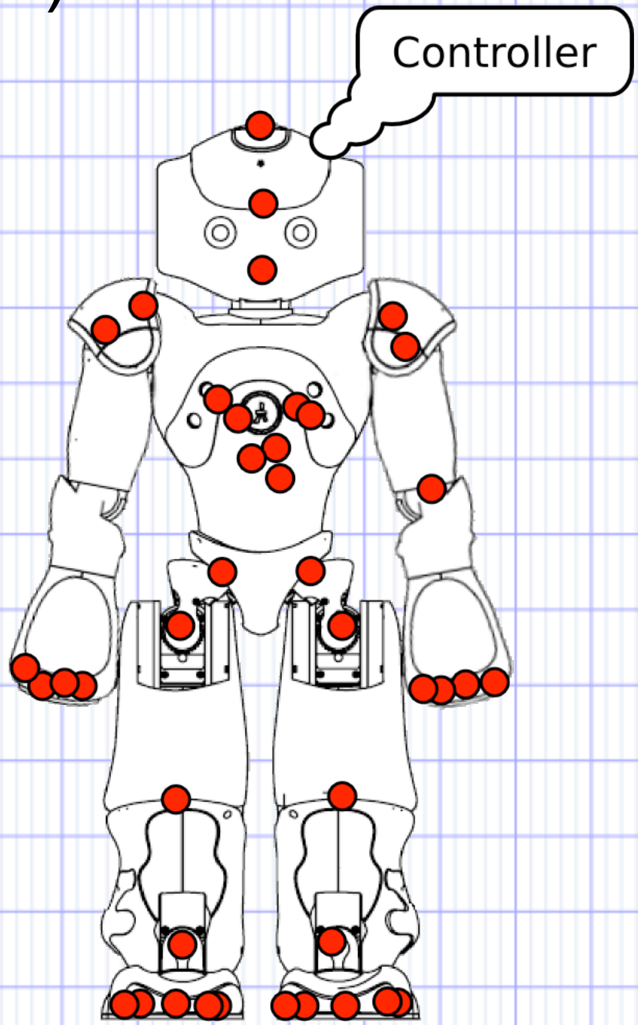
Recurrent Neural Networks RNN

- Neural networks & robotic control
  - Feedforward topologies → reactive control
  - Recurrent topologies → beyond reactive memory, states
- Flexible/adaptive software
  - Connection weights parametric
  - Topology non parametric
  - Recurrences dynamic system
- … but
  - Exponential memory loss
  - Harder to train non linear



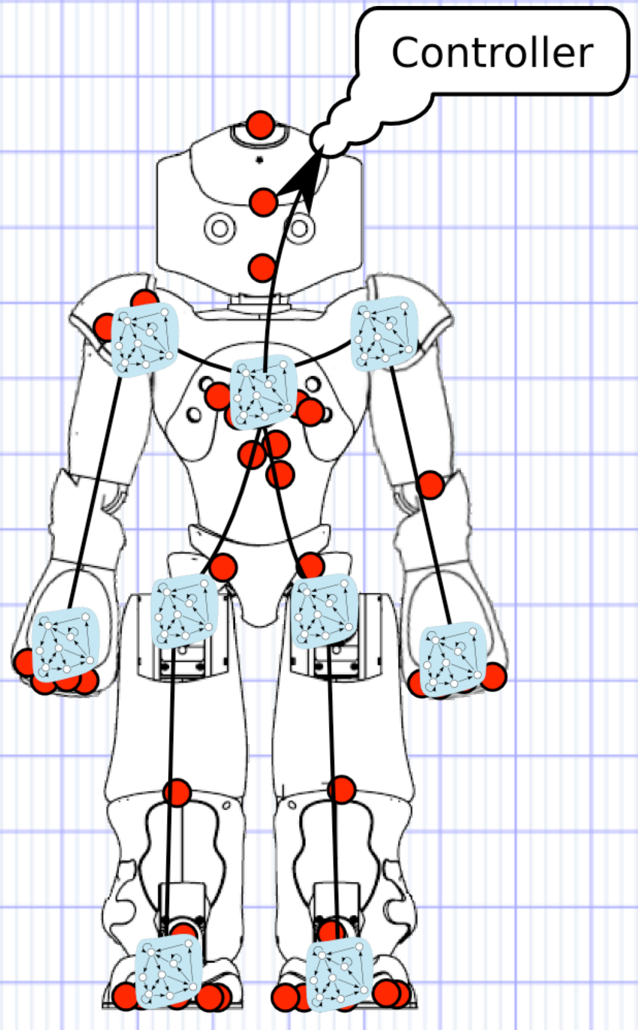
From controller.........

(R)NN as a controller



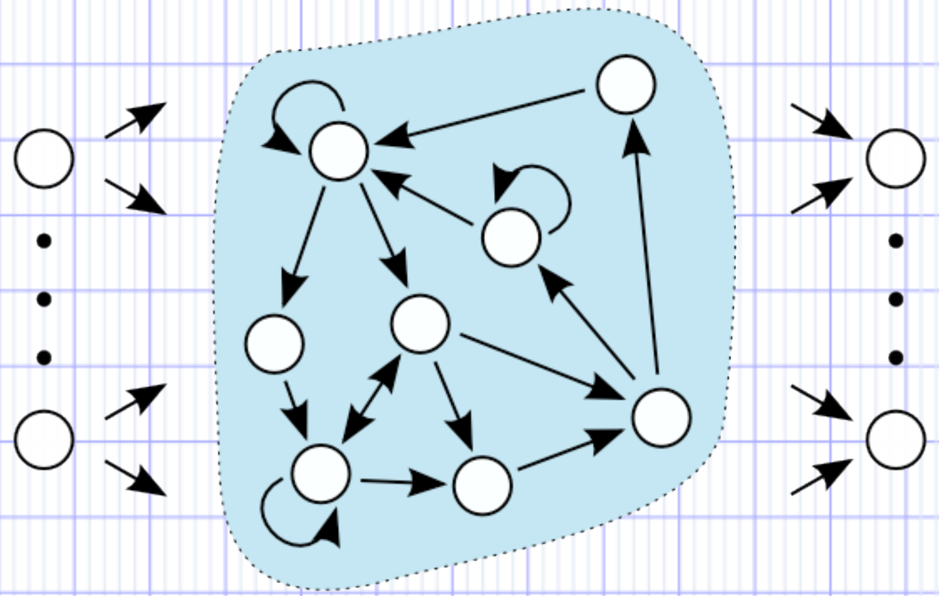
.........to Middleware

RNN as meta-sensors



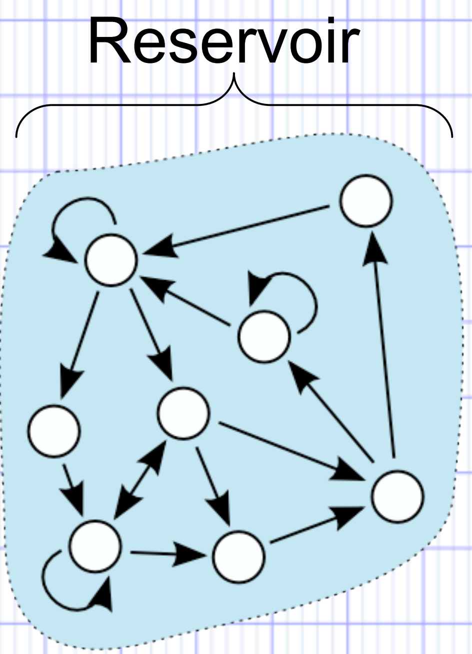
Reservoir computing [Jaeger, 01][Maas, 02]

- General idea
  - Temporal dynamics
  - Fine memory tuning
- Advantages
  - Few parameters
  - Easier training
  - Modularity
- Two main paradigms
  - Liquid State Machines [Maas, 02]
  - Echo State Networks [Jaeger, 01]



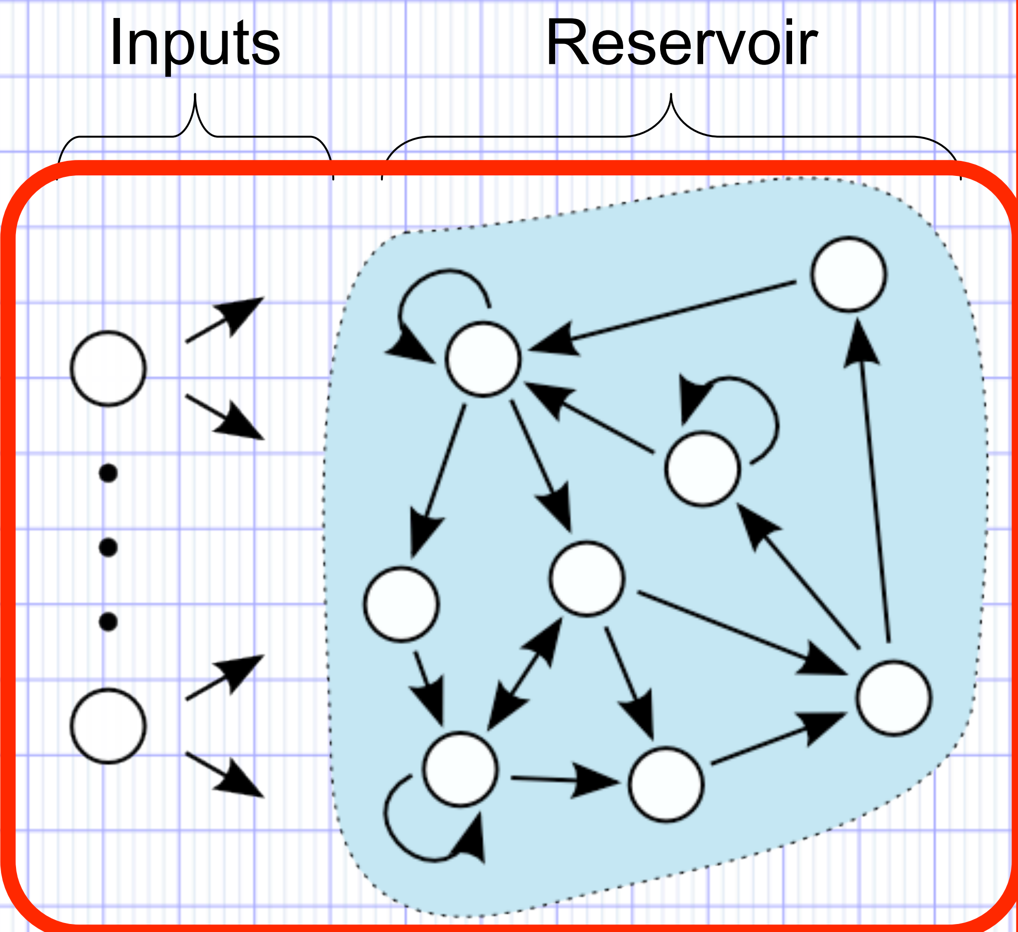
Echo State Network (ESN) [Jaeger 01]

- Concept – Echo State Property
  - Hidden neurons : reservoir
  - Random connections based on given density
  - Stability achieved through Damping



Echo State Network (ESN) [Jaeger 01]

- Dynamical system approximator

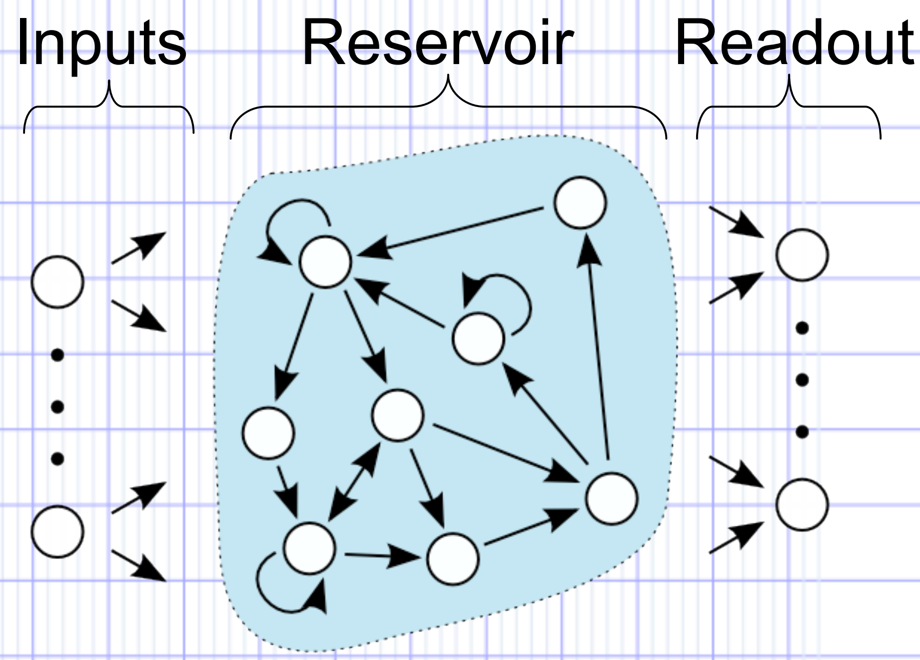


Randomly generated

Parameters :
- Reservoir size
- Reservoir density
- Connection damping

Echo State Network (ESN) [Jaeger 01]

- Reservoir
  - Input signal → randomly generated Dynamic system
  - Dynamics maps the input to a higher dimension
- Readout network trained to read the state of the reservoir and map to the desired output
  - Training only on the readout
  - Reservoir fixed



ESN system equations

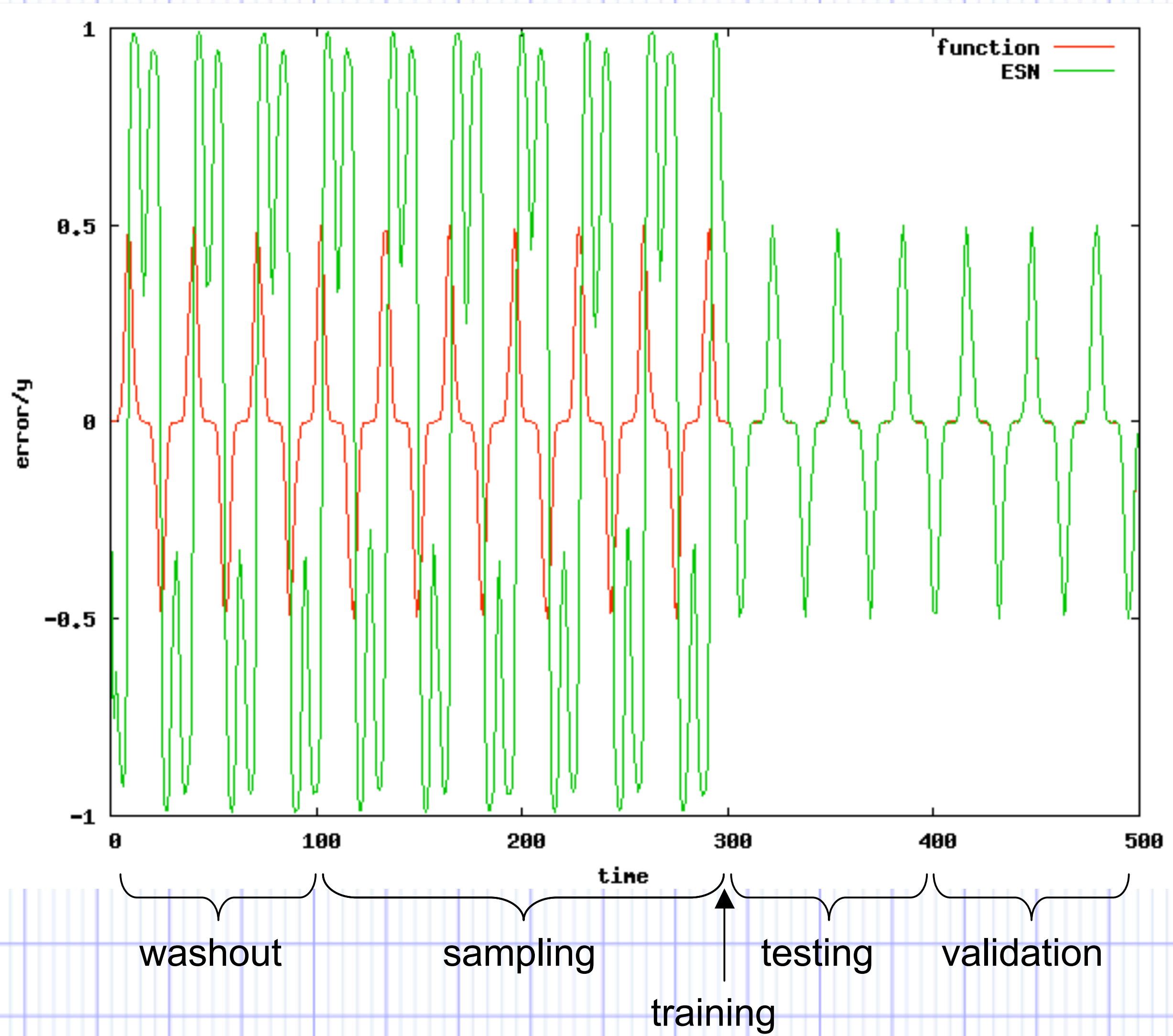
- input : $u(n)$
- Internal state : $x(n+1)=f(W^{in}u.(n+1)+W.x(n)+W^{back}.y(n))$
- Output : $y(n+1)=f^{out}(W^{out}(u(n+1),x(n+1),y(n)))$
- with :
  - f : activation functions
  - W^{in} : is the KxN input weight matrix
  - W : is the NxN reservoir weight matrix
  - W^{back} : is an optional LxN output feedback matrix
  - W^{out} : is the $Lx(K+N)$ input+reservoir to output matrix

ESN supervised learning

- We have
  - Input sequence *u(1), … u(T)*
  - Desired output sequence *d(1), … d(T)*

- Training algorithm :
  - We reset the Reservoir state washout
  - We feed u(1), … u(T) to the ESN :
    - We get state sequences x*(1), … x(T)*
  - We compute readout weights $\boldsymbol{W}^{out} = M^{-1}T$ linear regression

ESN supervised learning – example



Experimental setup

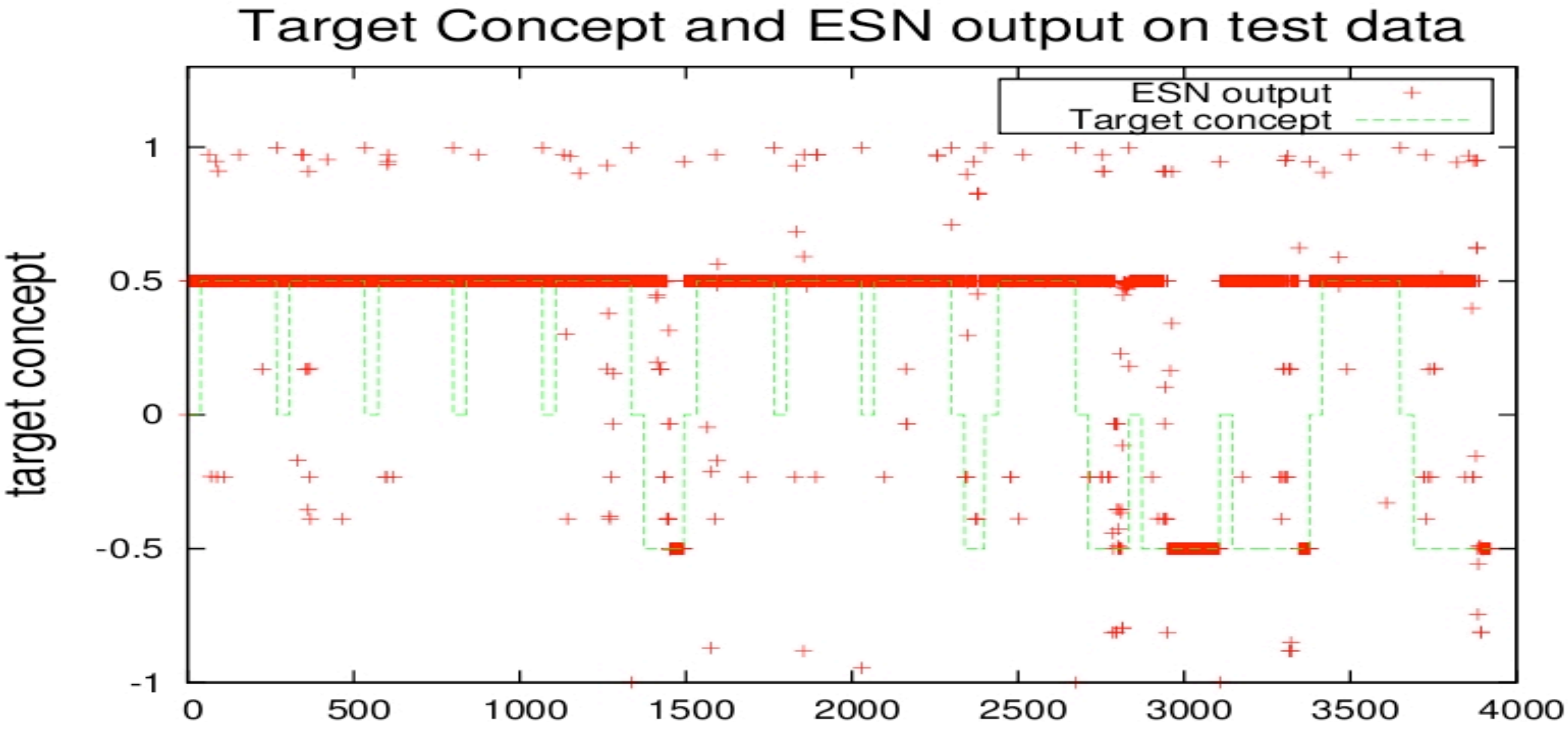
- Humanoid robot Aldebaran NAO
  - 15 relevant sensors as ESN data input
  - Output concept based on walk pattern
    - Stable (no fall occurs)
    - Unstable (the robot fall)
- Walk data recorded and labelled robot fall or not
  - ~ 4.000 lines of training data samples
  - ~ 3.000 lines of test data samples

Parameters

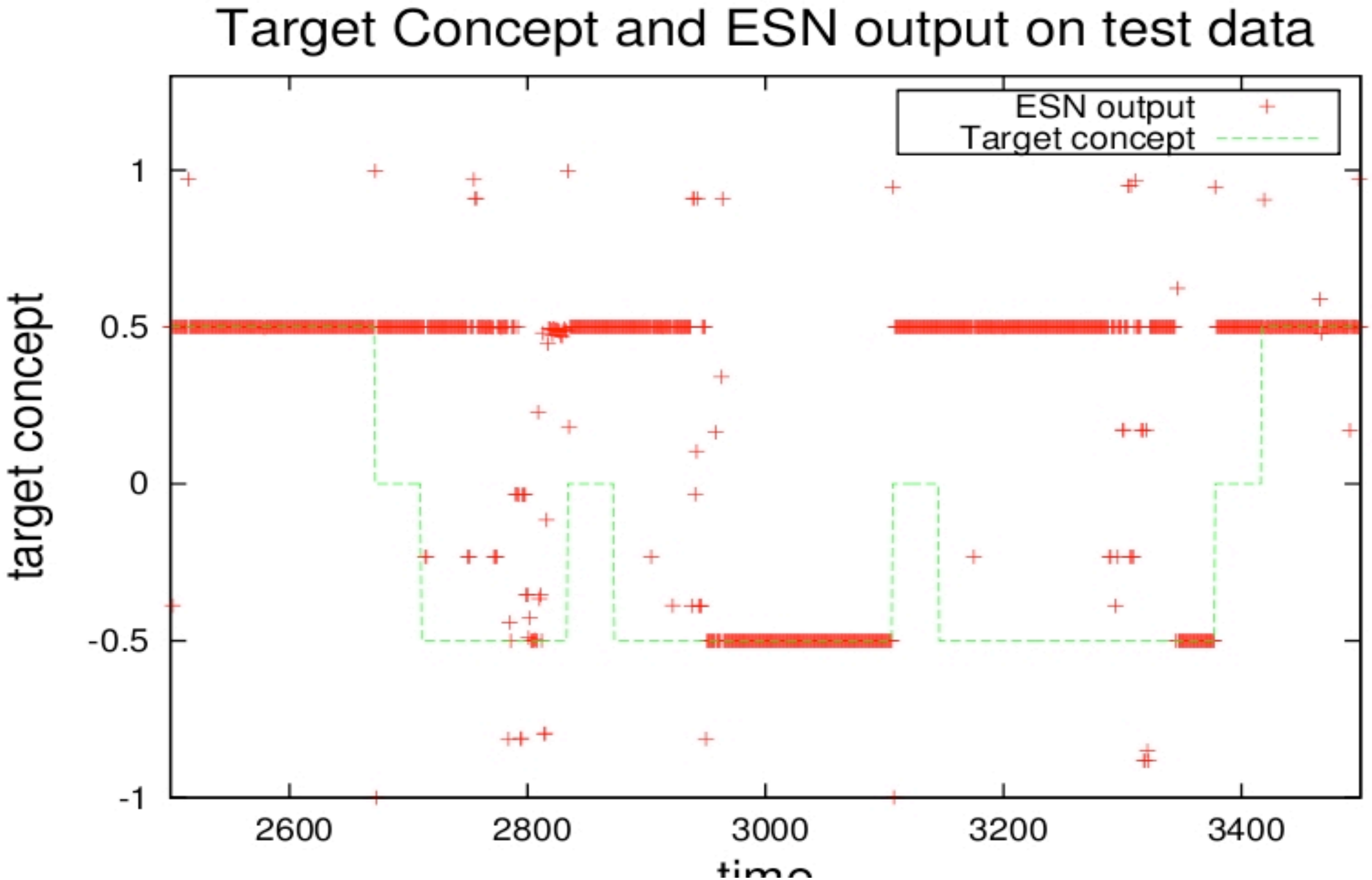
- ESN parameters
  - (inputs,reservoir, outputs) = (15, 100, 1)
  - Damping = 0.9
  - Connection density = 0.1
  - Activation function = Hyperbolic Tangent
- Evolution Parameters
  - Covariance Matrix Adaptation (CMA-ES) [Hansen 05] meta-optimisation involving no parameters
  - Fitness $F = (1/N) \sum [y(i) - d(i)]^2$

Results

- Over 16 walk samples, up to 4000 points plotted.
- 0 value indicates a no movement stable state of the robot, +0.5 is a stable walk and -0.5 indicates instability leading to fall.



3 Different Test Fall Patterns



Discussion

- For stable samples ESN provides negative output at very few points.
- For the fall samples, ESN does not immediately classify a data as ”Fall”.
- The output value does not jump directly from +0.5 to -0.5. Enables to predict a fall in advance and we have almost half a second to initiate an action
- Unstable points in stable pattern and stable walk before a fall proves the concurrency with practical observation.

Conclusion

- Reservoir Computing as meta-sensors
  - Not used for control but as middleware between sensors and control architecture
  - Echo State Networks based approach
  - First validation over Fall detection
    - Able to predict fall on short term
    - Able to detect unstable walks
- Perspectives
  - Compare to Liquid State Machines and NEAT
  - Train to predict on longer terms