Neural Networks

Lecture 7: Advanced Recurrent Neural Architectures

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

Revisited: **Recurrent** Artificial Neural Networks ...are everywhere.

End-to-End Speech Recognition

Motion Tracking and Learning

Dialogue / Text Summarisation

Hi!

Hello, what can I help you with today?

I'd like to book a table for six people.

I'm on it!

Where should it be?

In Hamburg.

Ok, let me look into some options.

Actually I would prefer for four.

Sure, is there anything else to update?

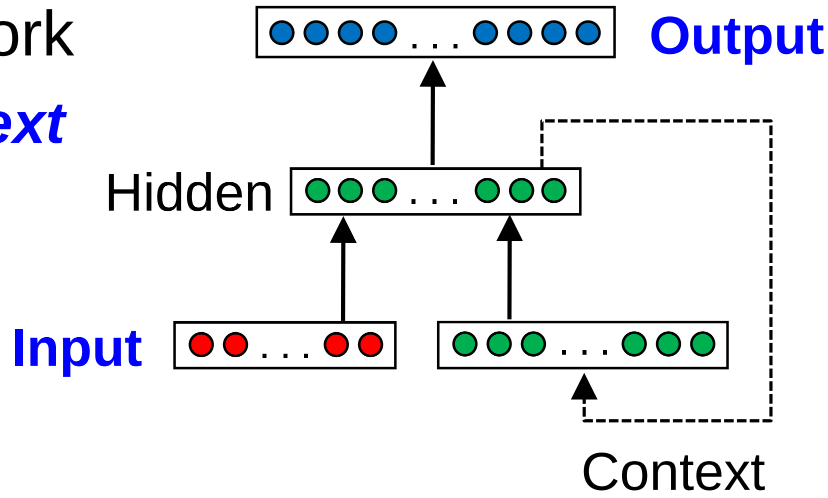
No.

How about the Gröninger Brauerei?

Revision: Recurrent Neural Networks

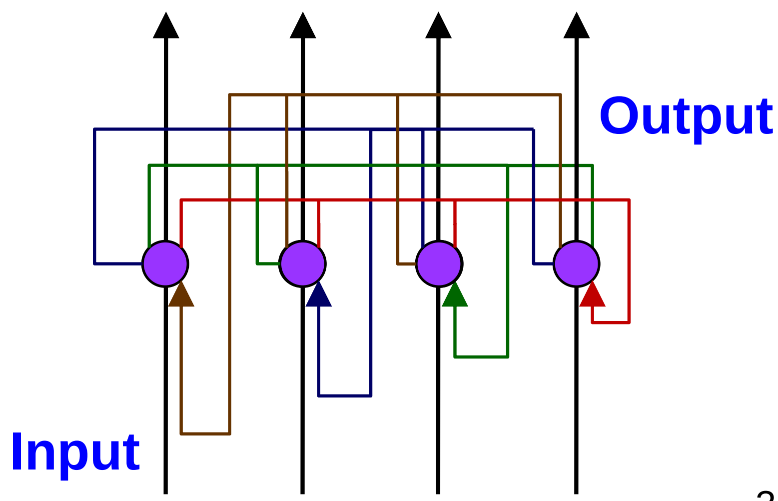
- Simple Recurrent Neural Network
  - Previous activation adds ***context*** to the current activation

  Examples:
  - Elman Network
  - Jordan Network



- Fully Connected Neural Network
  - Often called ***auto-associator***

  Examples:
  - Hopfield Network (binary)
  - Boltzmann machine (stochastic)



RNN Theory: Universal Dynamics Approximator

- RNN can have ***continuous*** activation function (sigmoidal) and ***continuous*** weight spaces
- RNN can approximate any finite sequence within Euclidean space $\mathbb{R}^n$ (arbitrary but finite n)
- Proof sketch:
  - Model dynamical system as a superset of differential equations
  - Match superset by set of differential equations possible with RNN using sigmoidal function
  - Find RNN over n output units, m hidden units and certain ***initial state*** of network

⇨ RNN are ***Super-Turing*** (can solve NP-complete problems)

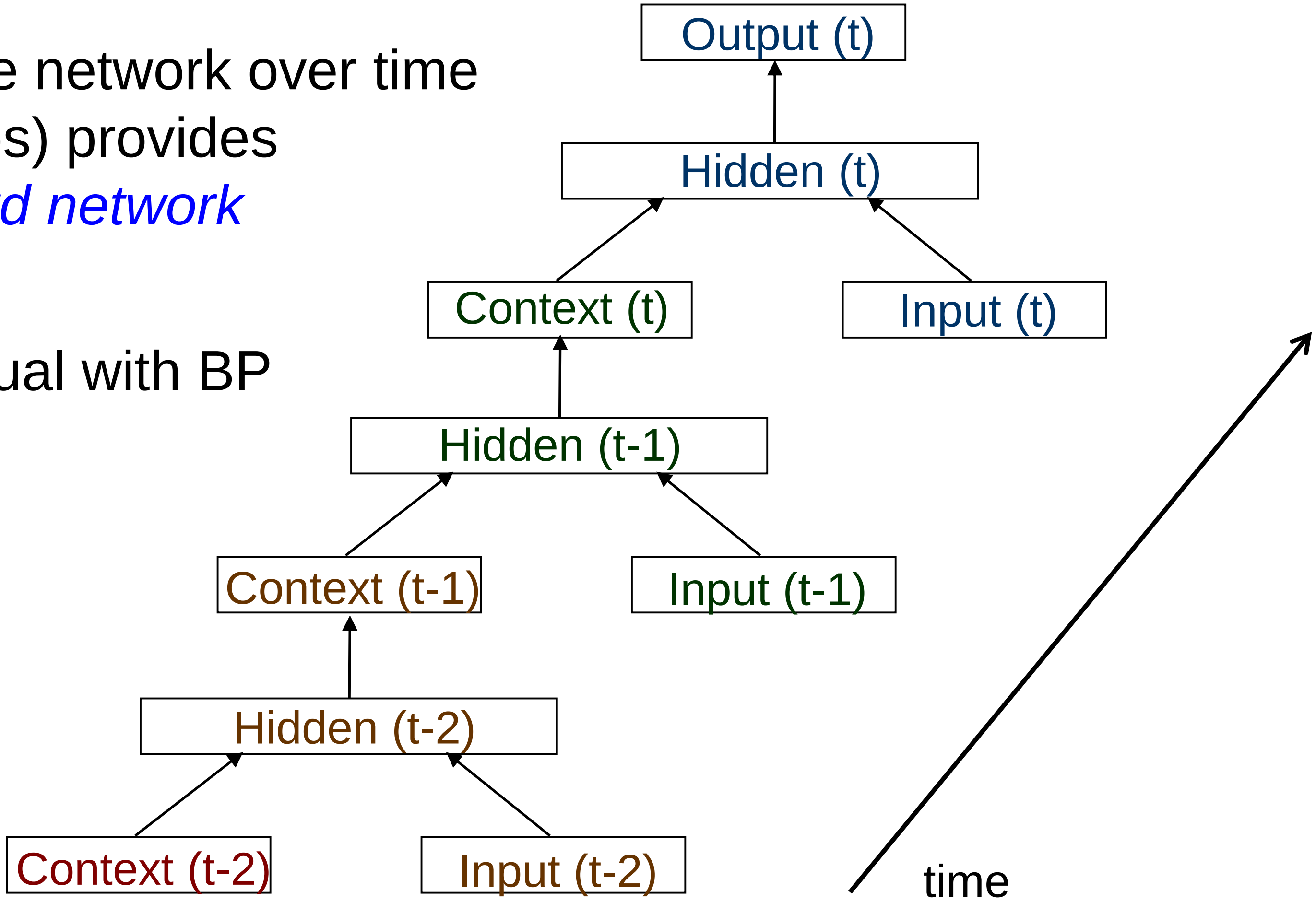
- Open issue: We can verify but not derive an RNN that solves an NP-complete problem

[Siegelmann 1995]

Revision: Learning with Backpropagation Through Time (BPTT)

- Unfolding the network over time (here: 3 steps) provides a *feedforward network*

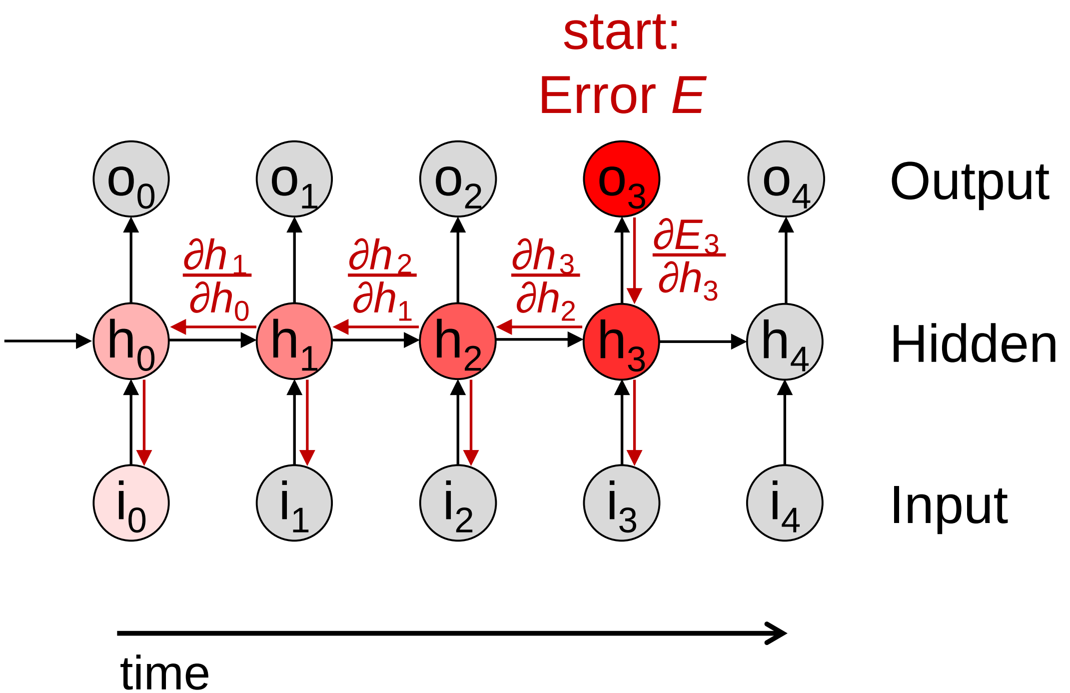
⇨ Train like usual with BP



- Important: employ a ***differentiable threshold*** function

Revision: The vanishing or exploding gradient problem

- Multiplication and nonlinearities at each step amplify the signal:
  - If the weights are small, the gradients shrink exponentially
  - If the weights are big, the gradients grow exponentially
- RNN trained on long sequences: influence of past inputs diminishes
  - usually for length >10

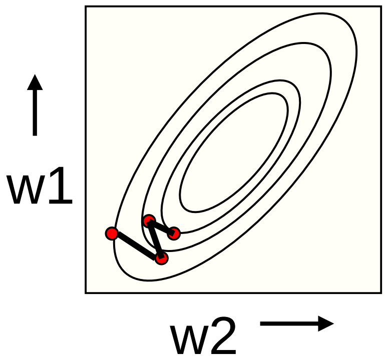


[Hochreiter 91]

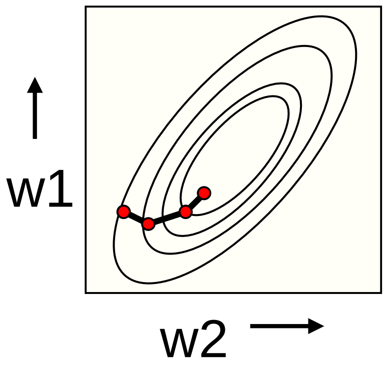
Back-propagation for Recurrent Neural Networks (cont.)

- Idea: ***unfold network*** in time and use a backprop variant:
  - **B**ack-**p**ropagation **t**hrough **t**ime
  - **R**eal-**t**ime **r**ecurrent **l**earning
  - **B**royden–**F**letcher–**G**oldfarb–**S**hanno method
  - **N**atural **G**radient **D**escent
  - **C**onjugate **G**radient **D**escent
  - ...

First-order vs second-order gradient descent



- Also important: Online vs. Batch learning
  - Online: More exploration; more dynamic
  - Batch: Faster convergence to a minimum; steepest descent ≠ global minimum
  - Good Idea: ***Mini batches***

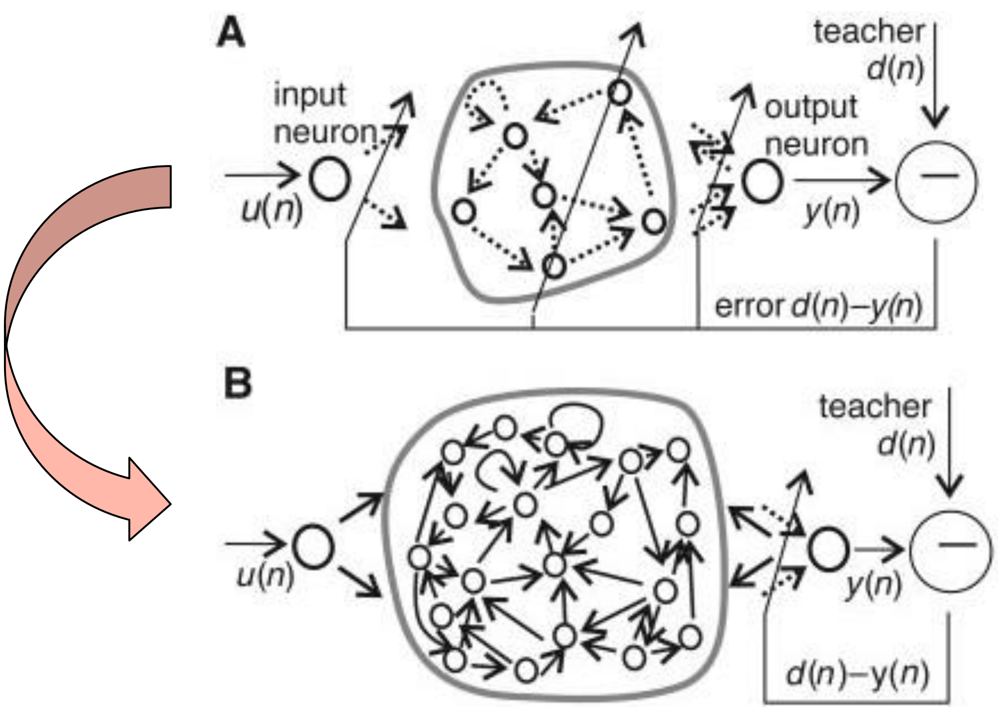


Advanced RNN: Constrained SRN!

- Recurrent Neural Networks are ***efficiently applicable*** to
  - Sequence prediction, classification, generation
  - Applications: Handwriting and speech recognition, text summarisation and keywords spotting, attentive vision ...
- Advantages – so far:
  - Bio-inspired method to problem solving ***using*** some ***context*** ... that is still ***deterministic*** and can be analysed
- Issues:
  - Time leaks or disturbations in the sequences are destructive
  - ***Training*** methods are ***uninformed*** or ***slow***
- Today's solution: **constrain the SRN** towards easier training or easier capturing the task's characteristics

1. RNN constraint: No training in hidden layers

- Revision: Echo State Network
  - Do not unfold (e.g. ***just the output layer***)
  - ***Randomize*** untrained connections (input & hidden layers)
  - Use ***linear*** methods for training (e.g. Linear regression)



- ***Echo States Networks*** (H. Jaeger 2001)
  - Average firing-rate neurons (leaky or not)
- ***Liquid State Machines*** (W. Maass 2002)
  - ***Spiking*** neurons

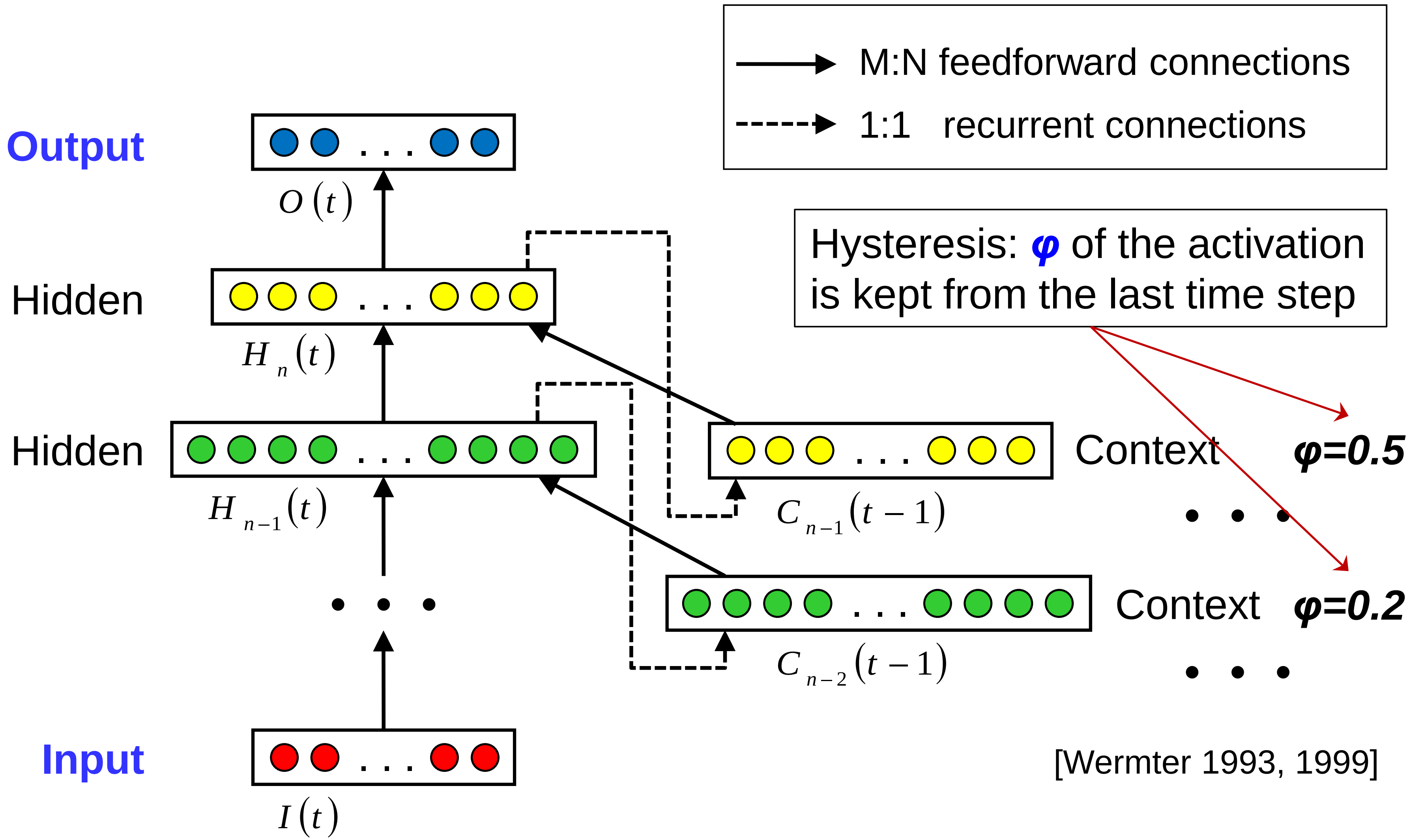
2. RNN constraint: Multiple context layers

- Characteristics
  - Arbitrary number of hidden & context layers
  - Every context layer memorises a ***different*** degree of ***dynamics***

  Example:
  - Recurrent Plausibility Networks

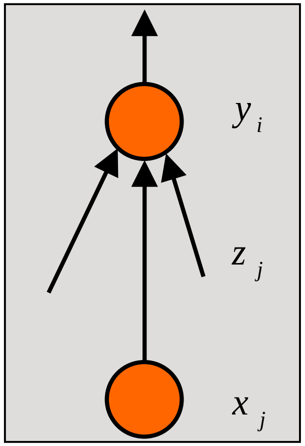
- Promises
  - Architecture reflects short-context and larger-context memory
  - Very ***robust*** against noise
  - Can be trained with backprop

Recurrent Plausibility Networks (RPN)



[Wermter 1993, 1999]

RPN: Activation and learning



- Units of context layers perform ***time-averaging***

$$C_{n,i}(t) = (1-\varphi_n)H_{n,i}(t-1) + \varphi_n C_{n,i}(t-1)$$

hysteresis value

- Learning: Employ ***back propagation in RPN***:

$$\Delta w_{ij}(t) = \begin{cases} \left(d_j(t) - y_j(t)\right) \cdot f'\left(z_j(t)\right) \cdot y_i(t) & \text{if } i \in H(t),\ j \in O(t) \\ \left(\sum_k \delta_k(t) \cdot w_{jk}\right) \cdot f'\left(z_j(t)\right) \cdot y_i^*(t) & \text{otherwise} \end{cases}$$

with $$y_i^*(t) = \begin{cases} y_i(t) & \text{if } i \in I(t) \\ y_i(t-1) & \text{if } i \in C(t) \\ \dots & \dots \\ y_i(t-t_l) & \text{if } i \in C(t-t_l) \end{cases}$$

for an arbitrary number l of recurrent (horizontal) context layers

FFN

SRN

RPN

$z_j(t) = \sum_l \sum_i w_{ij} y_i(t-l)$, for $l \in (0, \dots, t_l)$, t_l is maximal time step

RPN experiment (Arevian 2007)

- Classification on the Reuters-21578 Corpus
  - **Task**: determine a ***category*** of a news title
  - Dataset of 21578 news with 118 categories
  - *Example*:

```
<REUTERS TOPICS=''YES'' LEWISSPLIT=''TRAIN''
CGISPLIT=''TRAINING-SET'' OLDID=''12981'' NEWID=''798''>
<DATE> 2-MAR-1987 16:51:43 42</DATE>
<TOPICS><D>livestock</D><D>hog</D></TOPICS>
<TITLE>AMERICAN PORK CONGRESS KICKS OFF TOMORROW</TITLE>
<DATELINE> CHICAGO, March 2 - </DATELINE><BODY>The American Pork
Congress kicks off tomorrow, March 3, in Indianapolis with 160
                         ...
trade show, in conjunction with the congress, will feature
the latest in technology in all areas of the industry, the NPPC
added. Reuter
\&\#3;</BODY></TEXT></REUTERS>
```

- Experiment:
  - Train on a sub-set (1,040 titles)
  - Test on 9,663 titles

RPN experiment: Classification results & noise

Method	Mean performance - 50 networks (%)		
	Recall	Precision	F_1 Measure
Randomised	92.72	92.12	92.42
Original Corpus	92.59	91.73	92.16
Reversed	92.26	91.39	91.83
Noise Factor 2	92.39	91.63	92.01
Noise Factor 4	91.28	90.37	90.82
Noise Factor 6	86.40	85.63	86.01

$$precision = \frac{tp}{tp + fp}$$

$$recall = \frac{tp}{tp + fn}$$

$$F_{score} = \frac{(1 + N^2) \cdot pre \cdot rec}{pre + (N^2 \cdot rec)}$$

F_1 Measure : F_{score} with $N = 1$

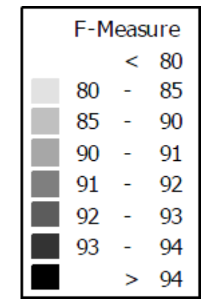
Noise: Introduce ***stop-words*** at ***random*** ⇨ Increase length of titles from e.g. 8 words to 16 (x2), 32 (x4) or 48 (x6) words

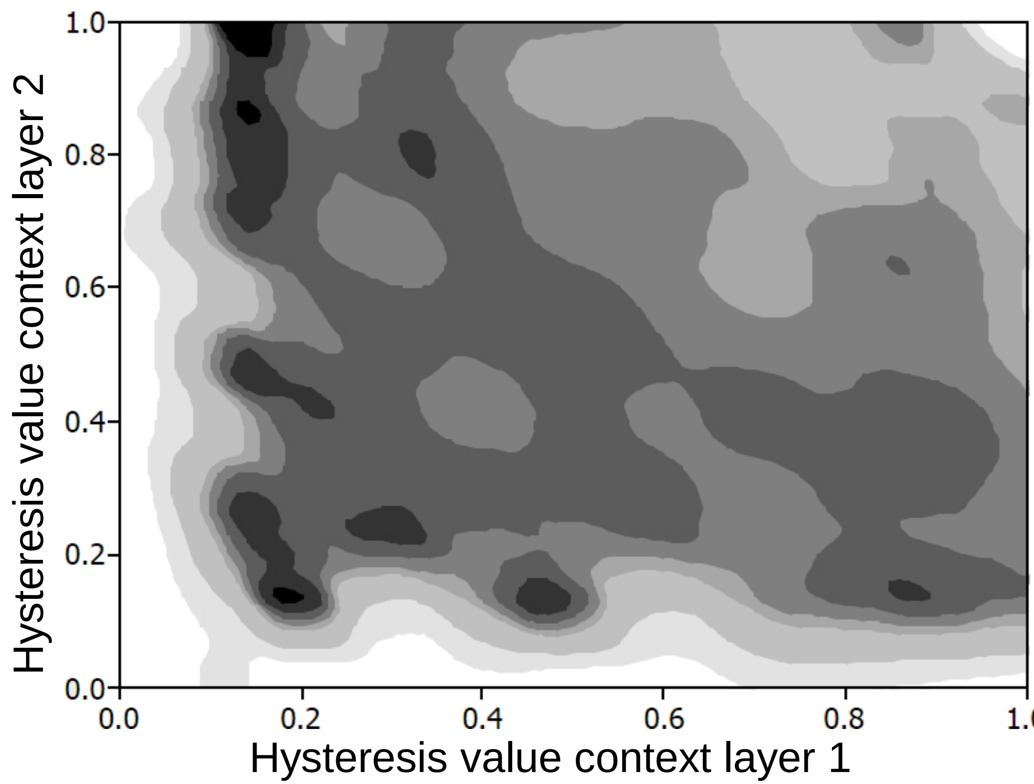
Adding ***noise*** leads to ***graceful degradation***!

How to determine Hysteresis parameters?

- Depending on the problem!
  - Good choice: ***smaller*** values for ***first context*** layers, ***higher*** values for ***second context*** layers
- On Reuters corpus on average:
  - $\varphi(C_1) = 0.2$
  - $\varphi(C_2) = 0.7$

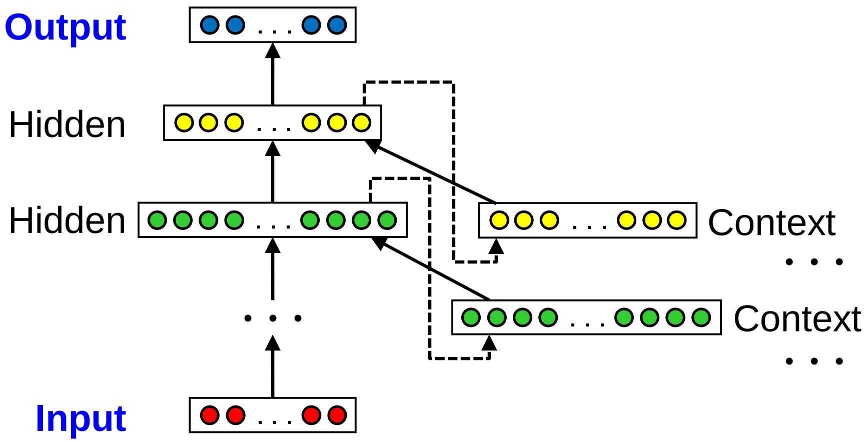
Hysteresis plots for sequential dataset with average title length of 8 words.





RPN experiment summary

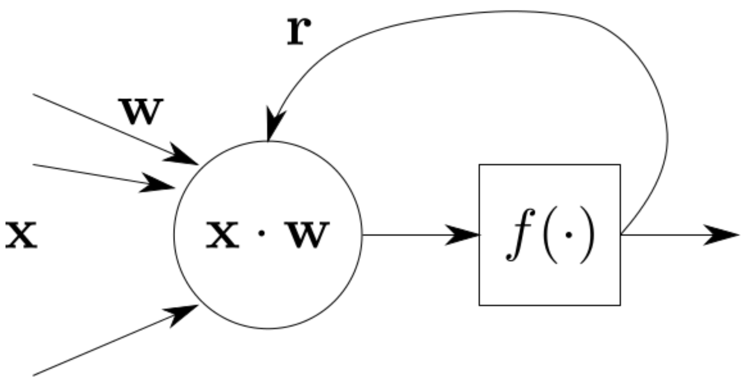
- Advantages
  - RPN can better ***capture*** the ***important context*** no matter whether relevant words occur in the beginning or the end of a sequence
    ⇨ Local context / local word order is less important
  - Noise robustness ⇨ good classification results for potentially disturbed sequences
- Disadvantages
  - Still a ***complex*** network: many Parameters
- Hysteresis values can tune the reach-out of the context units

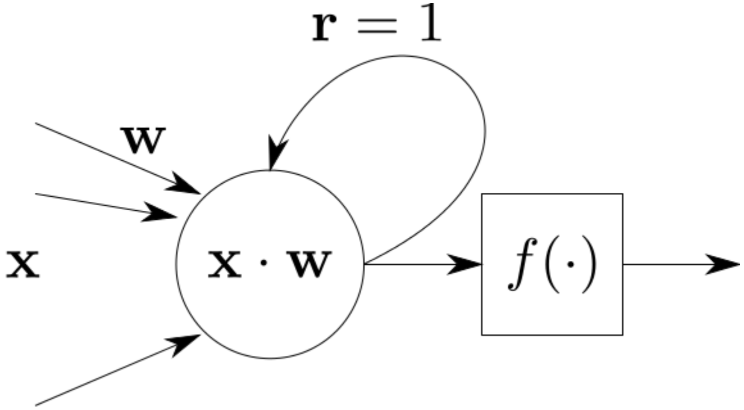


3. RNN constraint: Avoid error multiplication

- Characteristics
  - Architectural solution to vanishing gradient problem
  - Place recurrent connection before nonlinearity

  Example:
  - Long Short-Term Memory





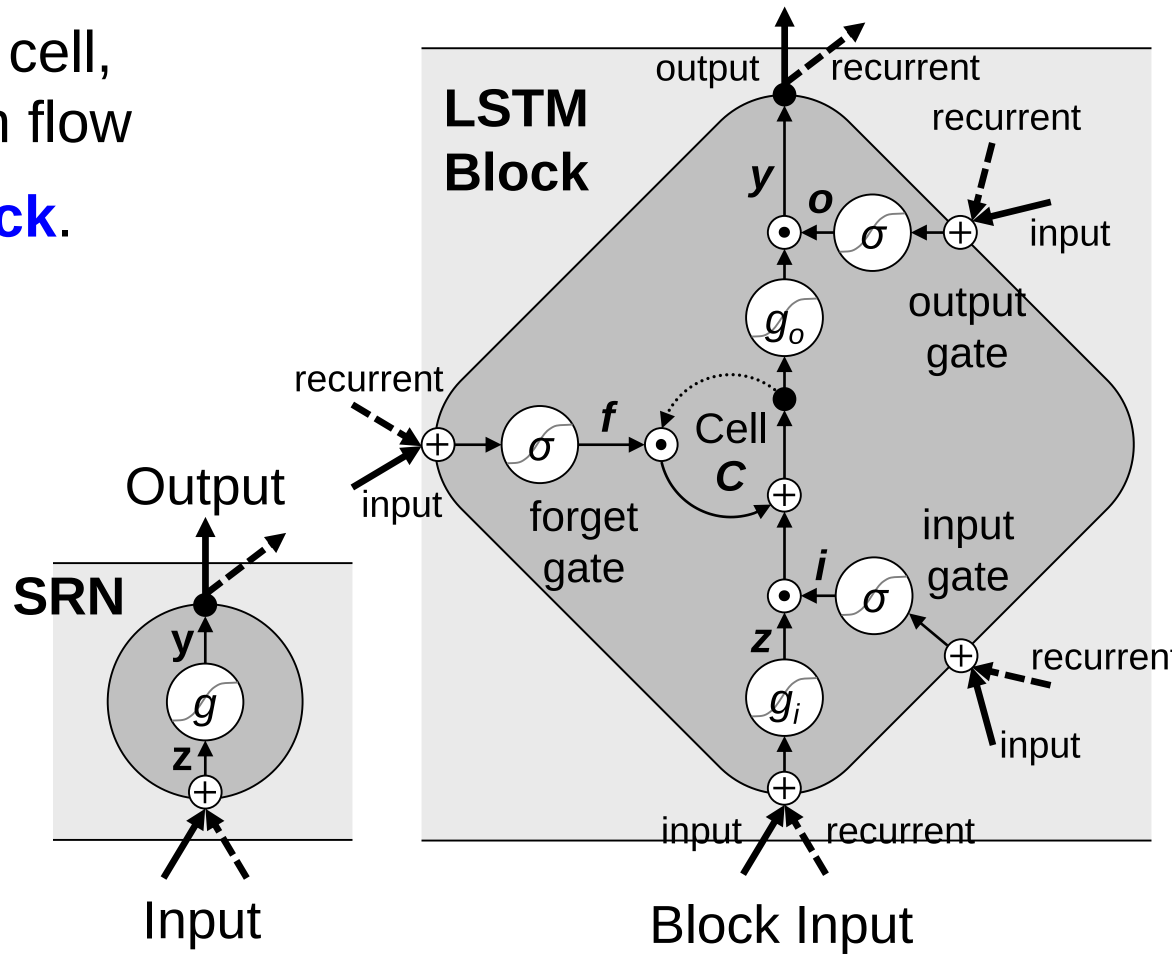
- Promises
  - Gradients do not vanish or explode

Long Short-Term Memory block

- Linear cell + constant “Error Carousel”
- Gates surround the cell, blocking information flow

⇨ called **memory block**.

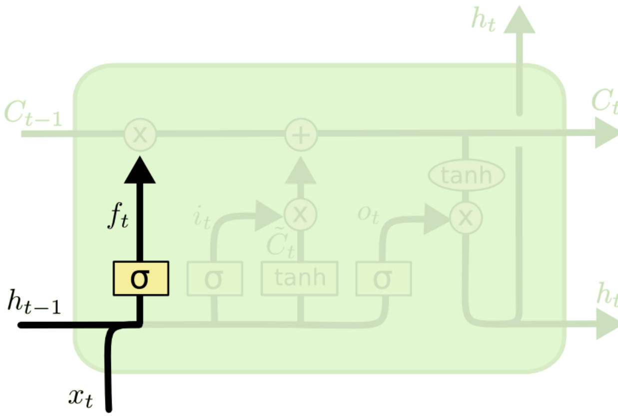
- unweighted connection
- weighted connection
- connection with time-lag
- branching point
- multiplication
- sum over all inputs
- σ gate activation function (sigmoid)
- g_i input activation function (usually tanh)
- g_o output activation function (usually tanh)



[Hochreiter 1997 / Greff 2015]

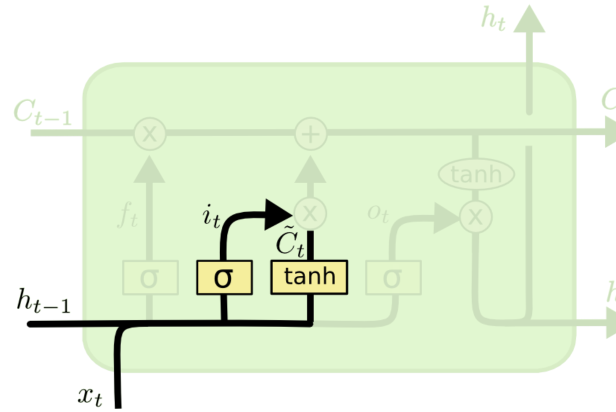
LSTM: Activation

forget gate



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

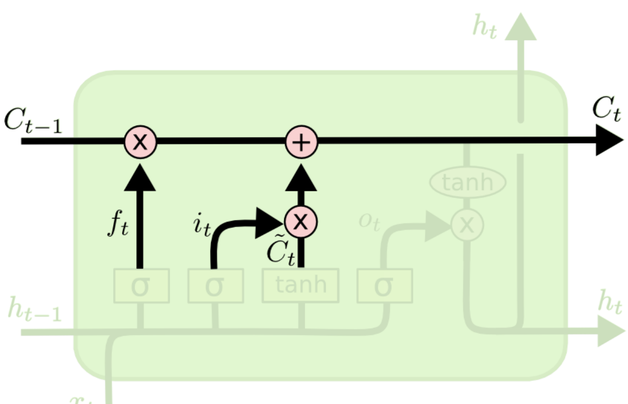
input cell state



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$$

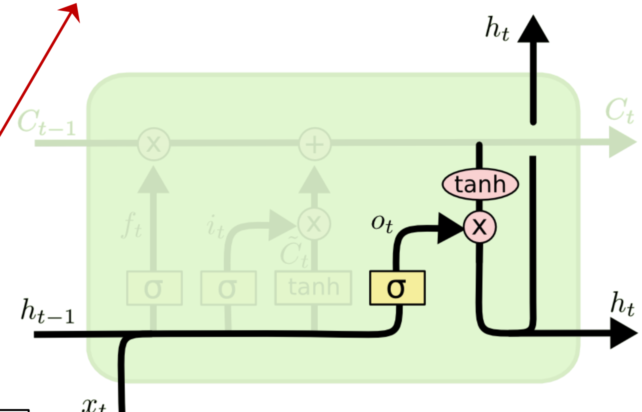
$$\tilde{C}_t = g_i\left(W_C \cdot [h_{t-1}, x_t] + b_C\right)$$

update cell state



$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$$

output



$$o_t = \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right)$$

$$h_t = y = o_t \cdot g_o(C_t)$$

watch notation!
$\tilde{C}_t \approx z$ from our previous formulas

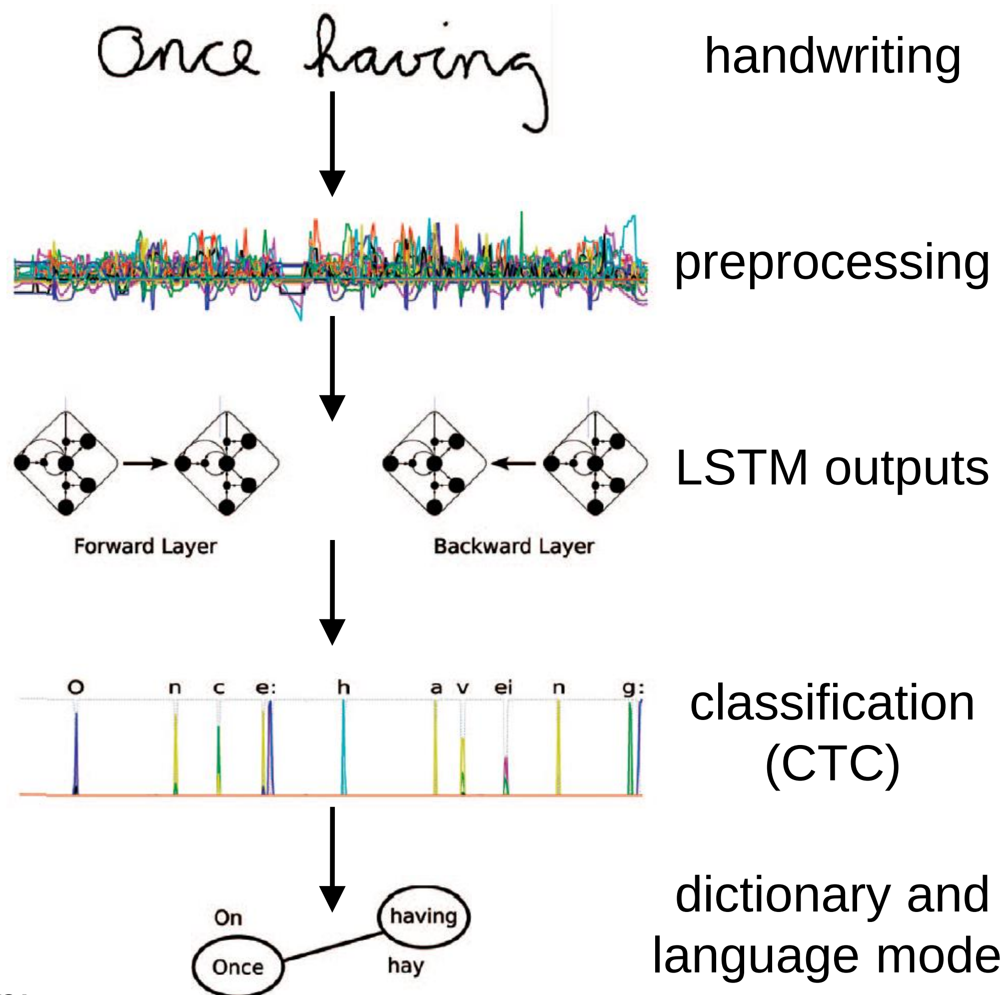
[https://colah.github.io/]

LSTM experiment (Graves 2008)

- Trained ***handwriting strokes*** and tested for recognition

Outcome:

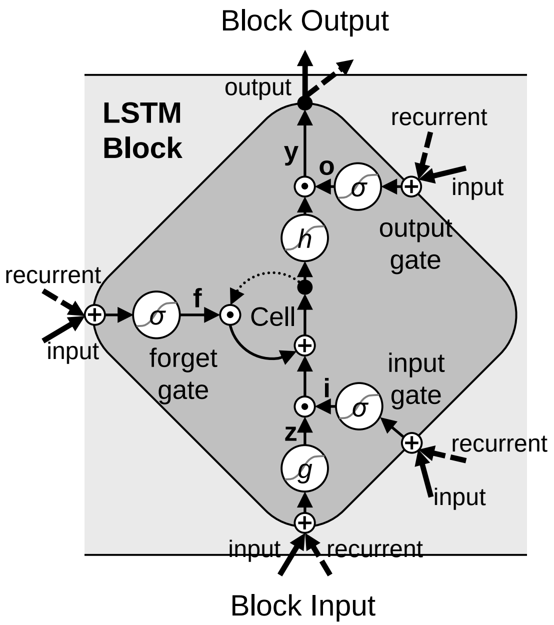
- LSTM learns long range contextual information
  - Successful on ***unsegmented*** data
  - Out-performed all HMM/GMM approaches at that time



[check interactive activation-visualisation: https://distill.pub/2016/handwriting/]

[Graves et al. 2008]

LSTM analysis and summary



- Advantages:
  - Capture ***long-term dependencies***
  - Currently most popular unit model, started industrial applications of RNNs

- Disadvantages:
  - Like SRN: sensitive to initialization
  - ***Not*** biologically ***plausible***
  - ***Complexity***, number of parameters, "ad hoc" design choices
- Gated Recurrent Units (GRU) are less complex and have been shown to behave very similar to LSTM
- Recently suggested simpler models have been shown to surpass LSTMs on specific domains

4. RNN constraint: Multiplicity of time

- Characteristics
  - Multiple context layers with ***different timescales***
    - ⇨Approximates different delays of spikes
  - *Context controlling* nodes that bias the sequence

  Example:
  - Multiple Timescale Recurrent Neural Network
  - Also related to
    - RPN with hysteresis concept
    - Various recent networks with leakage concept

- Promises
  - Self organising of different aspects of the sequences
  - Hierarchy of dynamics can emerge

Link to BAI: From Spiking to Rate-Coding

- ***Firing-rate models***: approximation of continuous models
  - Approximation of activation update
  - Derivation of output firing rate over time t:

$$\boxed{\tau_i} \frac{\mathrm{d}\, y_{t,i}}{\mathrm{d}\, t} = - y_{t,i} + f\left(\sum_{j=1}^{N} w_{ij} \cdot x_{t,j}\right)$$

time constant how rapidly the firing rate approaches its steady state value

current activity

threshold function

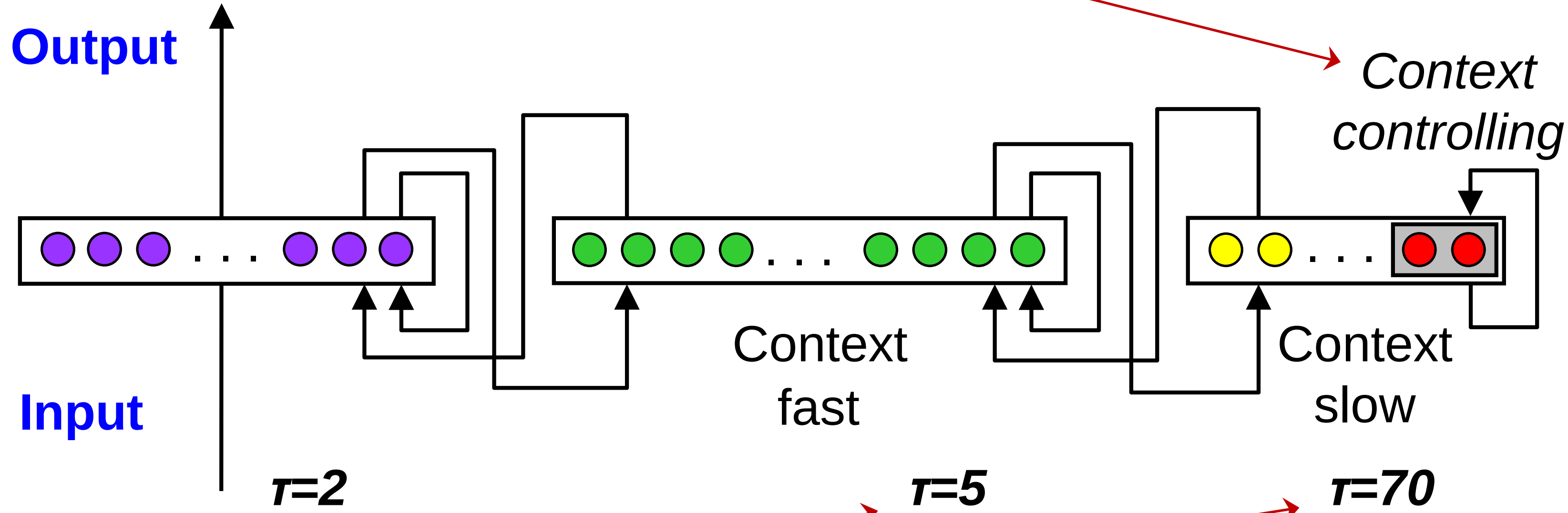
inputs *(could be recurrent as well)*

  - Models exponential dynamics of activation at neuron's membrane
  - Leaky integrator model: RC-circuit equivalent

⇨ Firing-Rate approach allows to model large networks

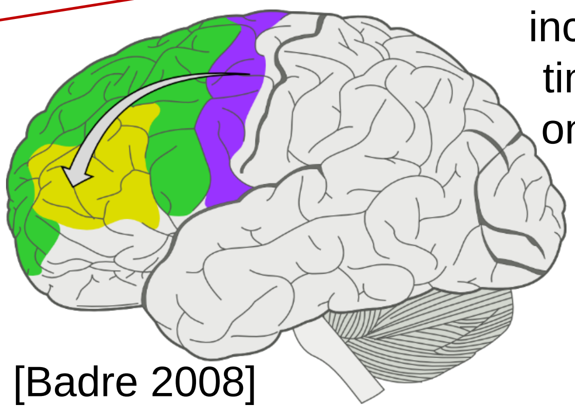
Multiple Timescale Recurrent Neural Network

Context controlling nodes (time step **0**)
can determine the whole sequence ⇨ ***bias***



[Yamashita 2008]

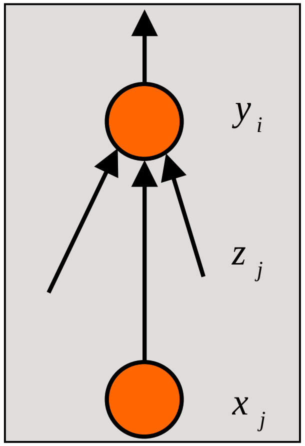
Timescale:
1/τ of the recurrent activation is taken into account while (1-1/τ) is kept from the last timestep



increasing timescale on rostro-caudal-axis

[Badre 2008]

MTRNN: Activation and learning



- Activation value of the *i*th neuron at step t:

summed input

$$z_{t,i} = \begin{cases} 0 & \text{if } t = 0 \\ \dfrac{1}{\tau_i} \sum_{j \in I_{\text{all}}} w_{ij} \cdot x_{t,j} + \left(1 - \dfrac{1}{\tau_i}\right) z_{t-1,i} & \text{otherwise} \end{cases}$$

time constant τ

output activity $$y_{t,i} = f\left(z_{t,i}, b_i\right)$$

- Backprop Learning:

error derivative

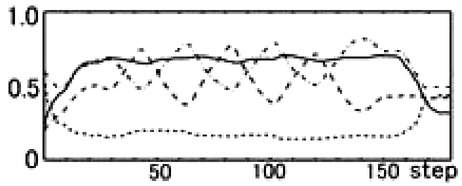
$$\frac{\partial E}{\partial z_{t,i}} = \begin{cases} \left(1 - \dfrac{1}{\tau_i}\right) \dfrac{\partial E}{\partial z_{t+1,i}} + \left(y_{t,i} - y^*_{t,i}\right) & \text{if } i = I_{\text{IO}} \\ \left(1 - \dfrac{1}{\tau_i}\right) \dfrac{\partial E}{\partial z_{t+1,i}} + \sum_{k \in I_{\text{all}}} \dfrac{w_{k,i}}{\tau_k} \dfrac{\partial E}{\partial z_{t+1,k}} f'(z_{t,i}) & \text{otherwise} \end{cases}$$

$$\Delta w_{ij} = \frac{1}{\tau_i} \cdot \sum_t x_{t,j} \frac{\partial E}{\partial z_{t,i}}$$

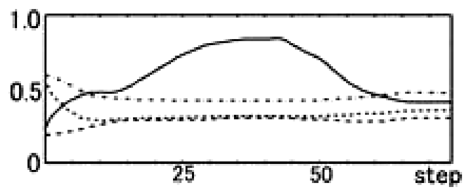
MTRNN experiment A (Yamashita 2008)

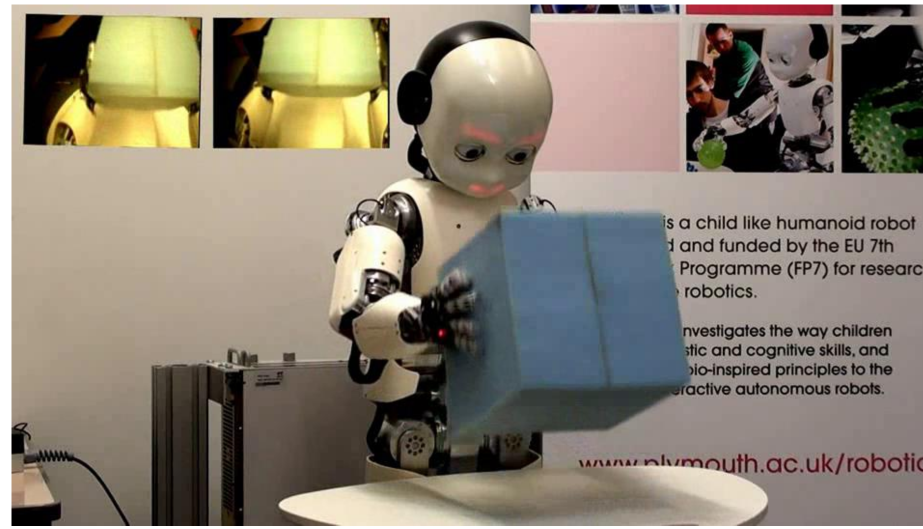
- Trained ***motor sequences***
  - Learn ***primitives***:

backward-forward



touch

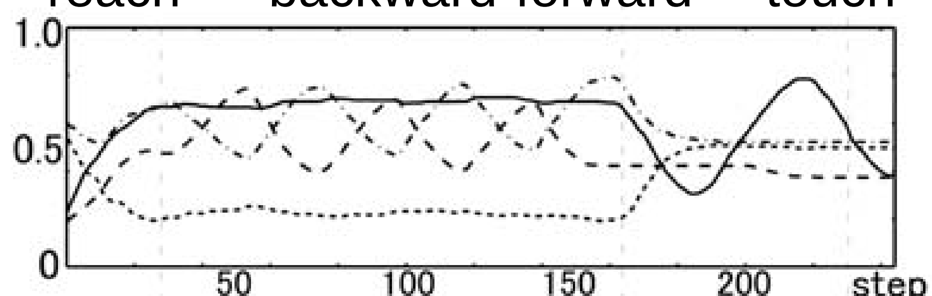




Reproduced with the ICub: [http://www.italkproject.com]

  - Learn ***combinations*** of primitives

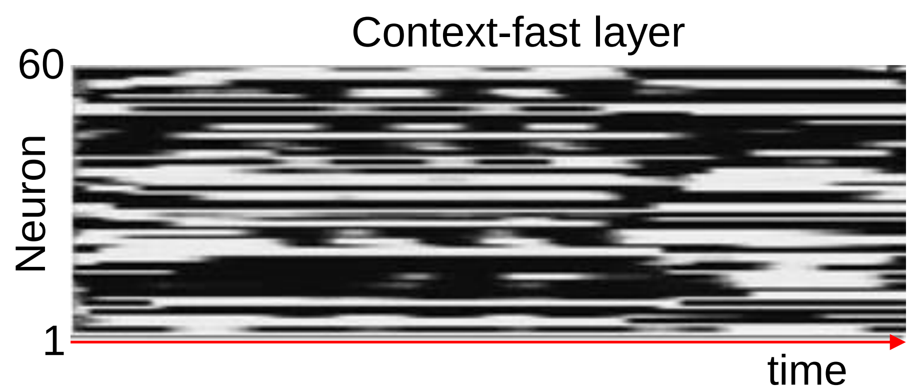
reach backward-forward touch



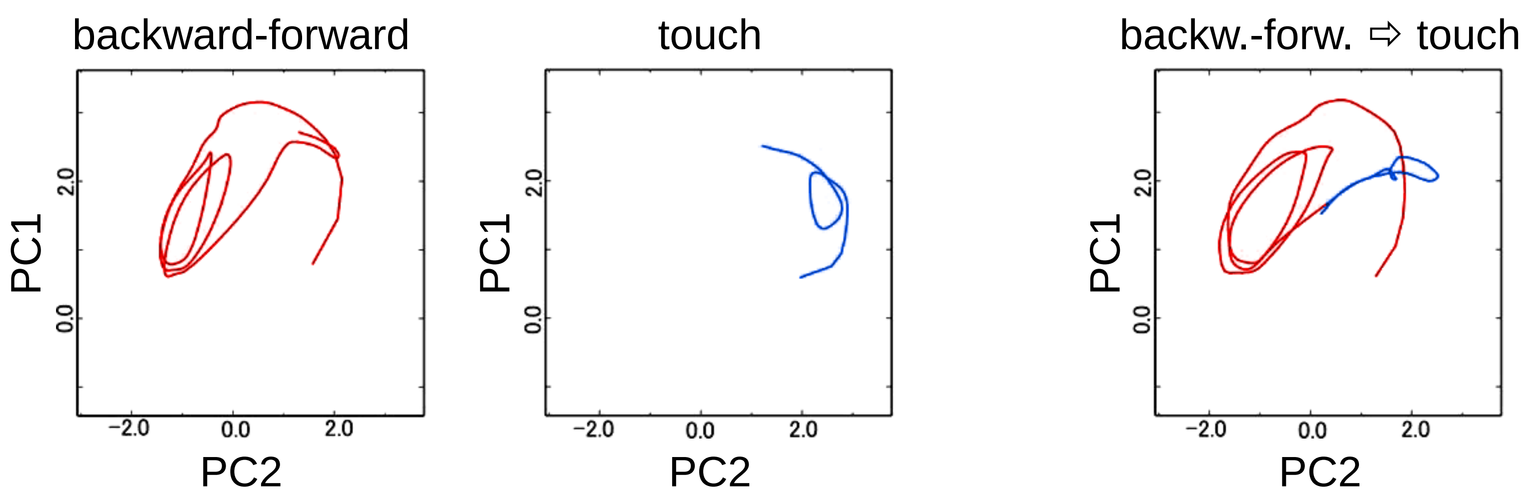
…and test for the ***emergence*** of a hierarchy

MTRNN experiment A: Analysis

- Question: How does the network ***self-organize***?
- Approach: Run a ***Principle Component Analysis*** on the neural activity

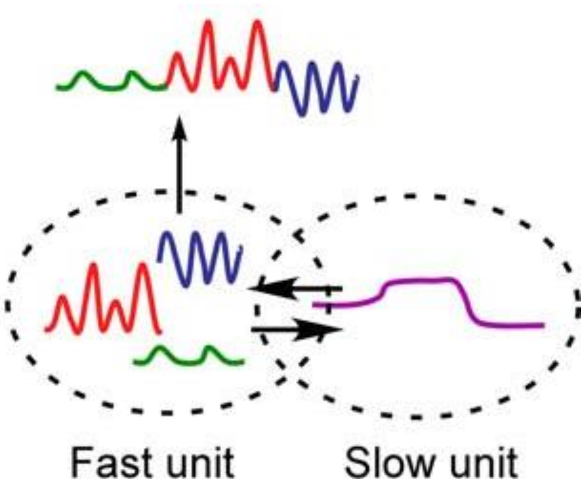


- Result: Emergence of a ***functional hierarchy***



MTRNN experiment B (Hinoshita 2011)

- Trained ***sentences*** and tested for emulation and recognition

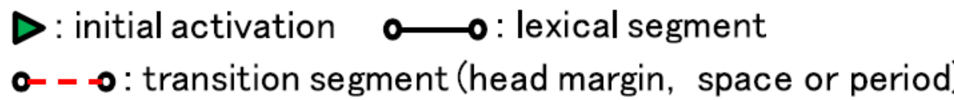


Interesting observation:

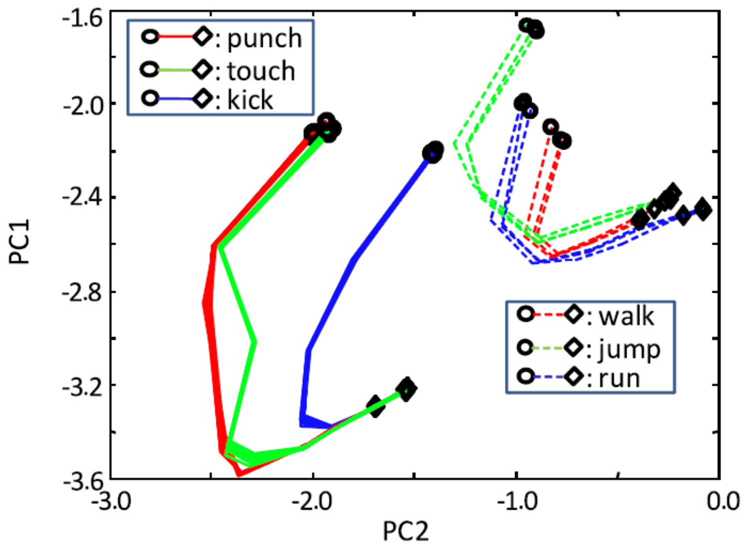
- Linguistic ***hierarchy*** emerges in the network:
  - Word representations in the Cf
  - Sentence representations in the Cs
- Linguistic structure can produce sentences from the ***inferred*** grammar.
  - Even if they where not learned explicitly!

MTRNN experiment B: Analysis

- MTRNN is ***decomposing*** the data



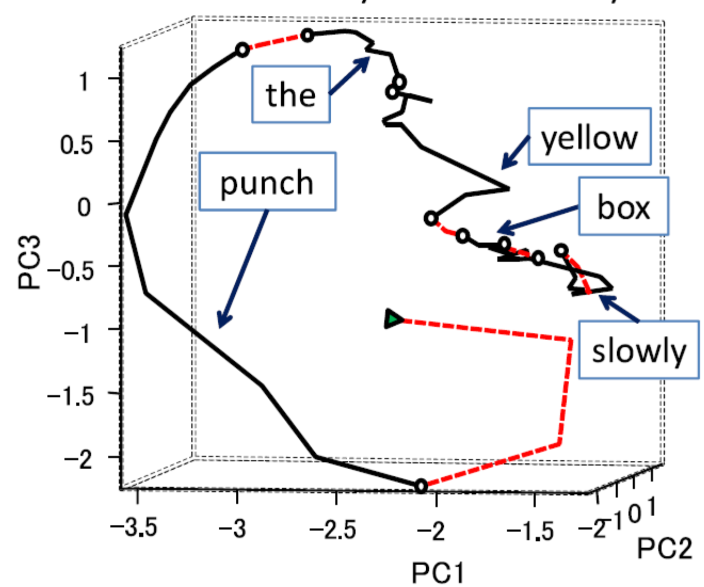
Activity in Cf layer



Activity of network over time is reduced to 2 or 3 principle components
⇨ shown as trajectory in 2D or 3D

Activity in Cs layer

Punch the yellow box slowly.

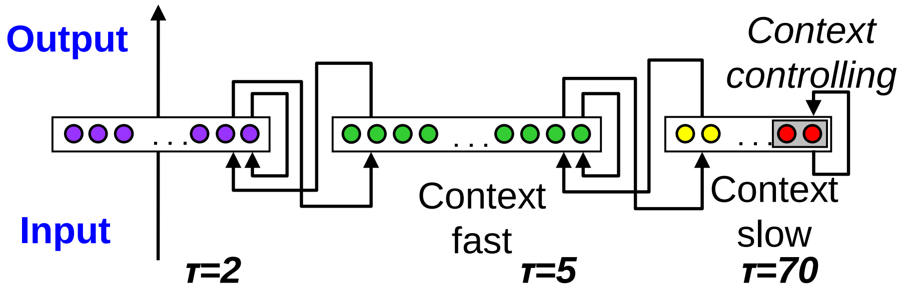


- Same words have nearly identical trajectories
- Words in the same categories have similar trajectories

- Sentences are represented as trajectory with segments for roles that can have different role fillers.

MTRNN experiments summary

- ***Deterministic*** recurrent neural network
- Can recognize, generate and correct sequences
- ***Self-organizing*** internal hierarchical ***structure***
- Uses fast and slow adapting context nodes
- Issue: BPTT difficult to calculate in real time
- Advantages:
  - Reproduces ***compositionality*** in the data



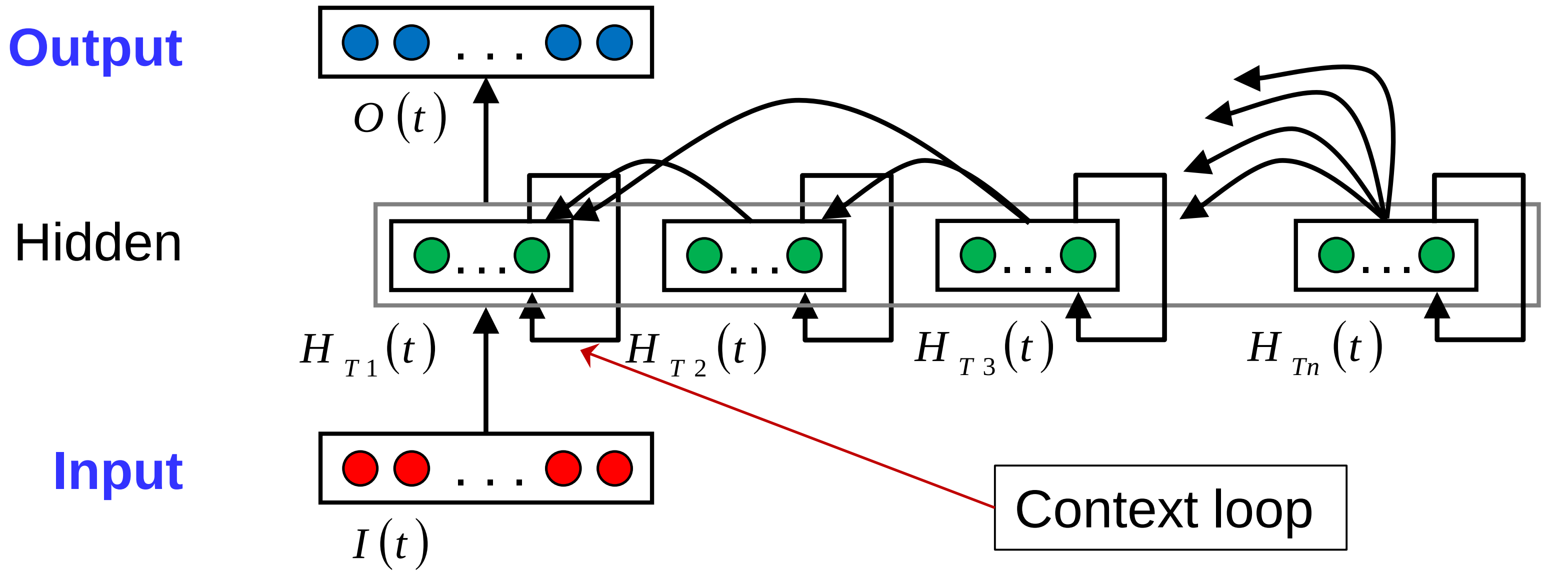
5. RNN constraint: Learn on different timescales

- Characteristics
  - Constrain ***activation time steps***
  - Partition hidden layer H into separate ***modules***

  Example:
  - Clockwork Recurrent Neural Network

- Promises
  - ***Reduce*** vanishing gradient problem
  - ***Pick up*** different ***timings***, inherent in data
  - Use any arbitrary form of backprop
  - Low number of parameters

Clockwork Recurrent Neural Network



- Assign a clock period T_k to each module k
- For each time step t:
  - Apply activation function if module is active
  - Otherwise, keep module activations from previous time step

Clockwork RNN activation function

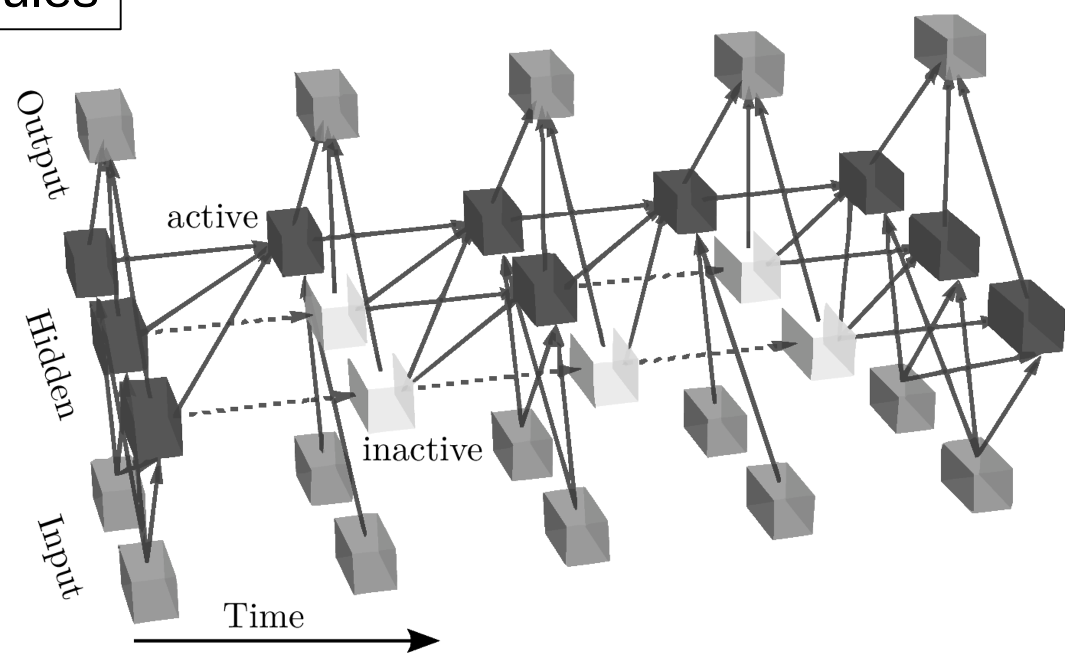
- Activation value of the neuron in *k*th module:

$$z_{t,k} = \begin{cases} f_h\left(x_t \cdot \mathbf{W}_{x,k} + \sum_{l=k}^{n} h_{t-1,l}\mathbf{W}_{l,k} \right) & \text{if } t \bmod T_k = 0 \\ h_{t-1,k} & \text{otherwise} \end{cases}$$

adjacent modules

Module k is only updated if t is multiple of clock period

Clockwork RNN unfolded in time

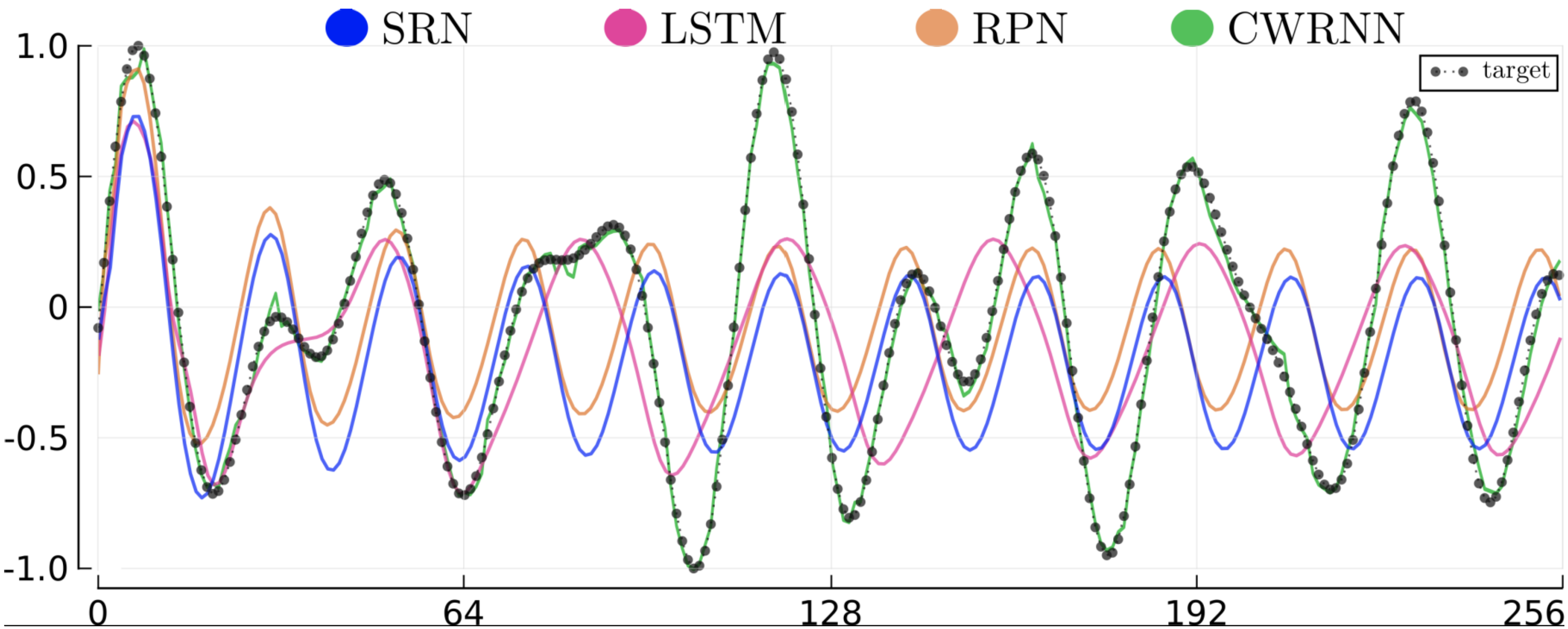


- Training:
  - Error propagated according to "activeness"

[Koutnik et al. 2014]

Analysis: Sequence generation

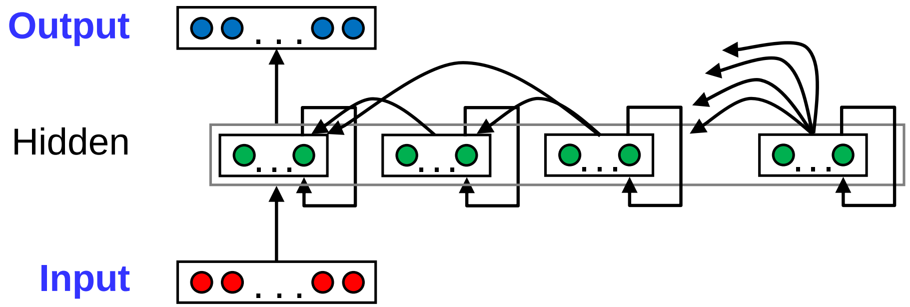
- Task: Learn to ***reproduce*** combination of sine waves



[Alpay et al. 2016]

Clockwork analysis and summary

- Advantages:
  - ***Emergence*** of different ***timescales*** in processing
  - Captures temporal dependencies more ***efficient*** than other networks
  - Store entire sequences in memory of clocked modules



- Disadvantages:
  - Good clock periods are dependent on data
  - So far no mechanism known to ***learn the clock*** periods
- Timescale effect is similar to MTRNNs but based on ***selective update*** instead of long term accumulation

6. RNN constraint: Memory augmentation

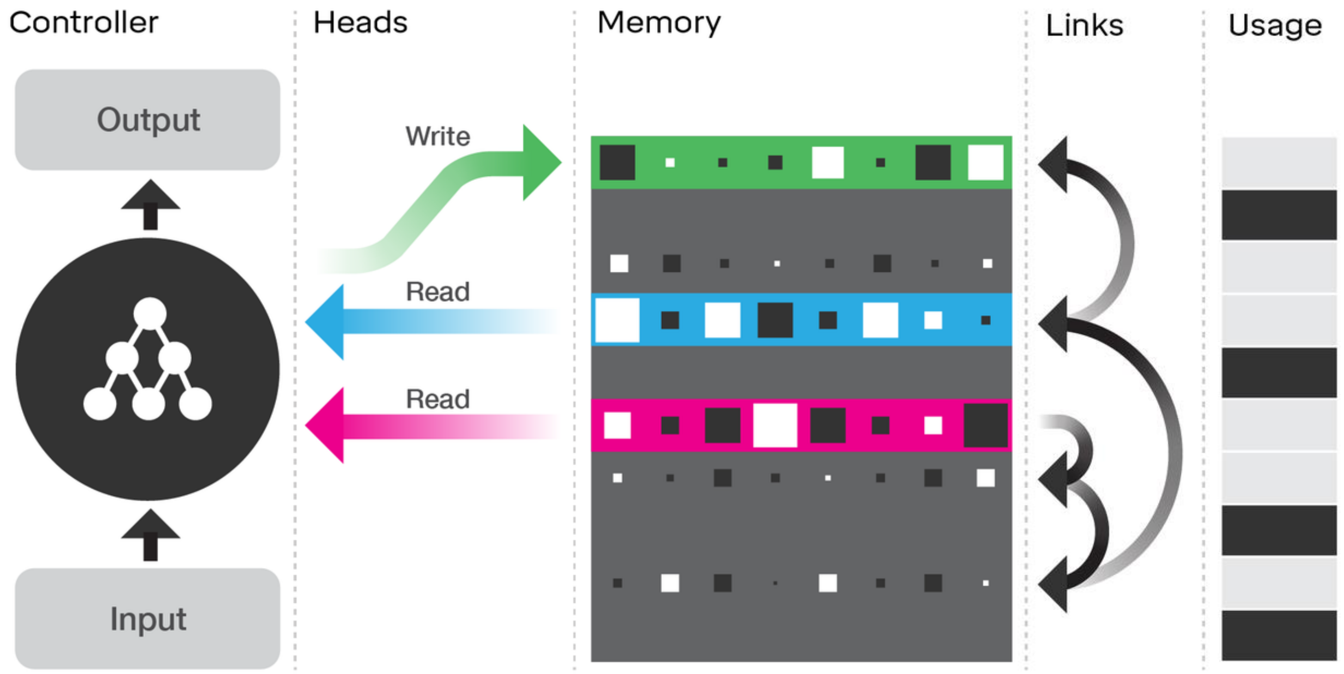
- Characteristics
  - Adds external memory bank for arrays of vectors
  - Reads and writes all memory via the attention mechanism

  Example:
  - Neural Turing Machine, Differentiable Neural Computer

- Promises
  - Joins Turing Machine with Super-Turing complexity
  - ***Circumvent*** vanishing gradient problem
  - Transfer: Can learn algorithms

Neural Turing Machine / Differentiable Neural Computer

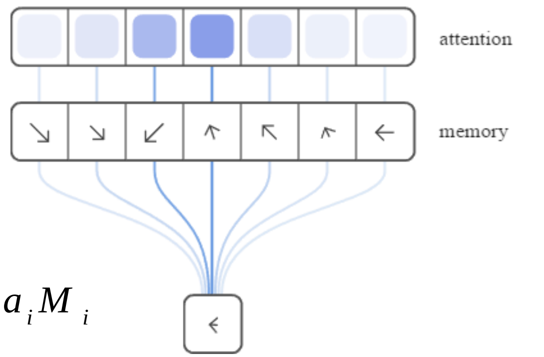
- DNC can make use of trained memory content
  - Focus attention ***content-based*** and ***location-based***
  - Effectively ***loop over activation*** patterns



[Graves et al. 2014/2016]

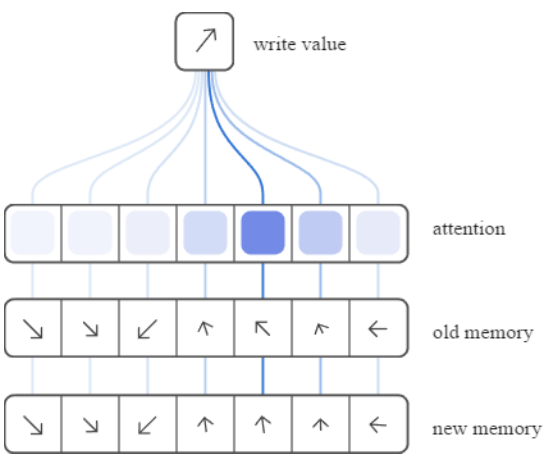
NTM/DNC: Memory usage

reading

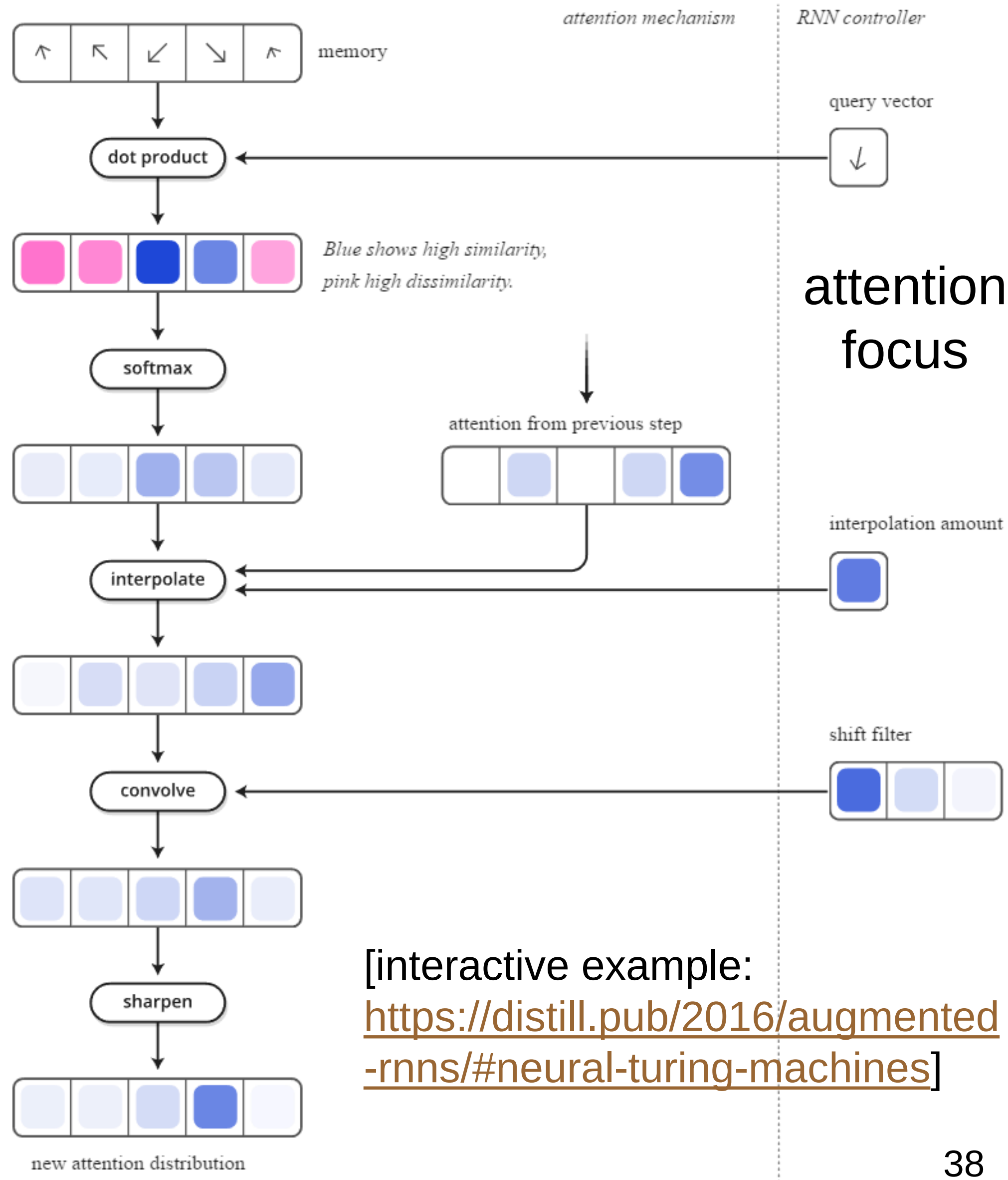


$$r \leftarrow \sum_i a_i M_i$$

writing



$$M_i \leftarrow a_i w + (1 - a)_i M_i$$



attention focus

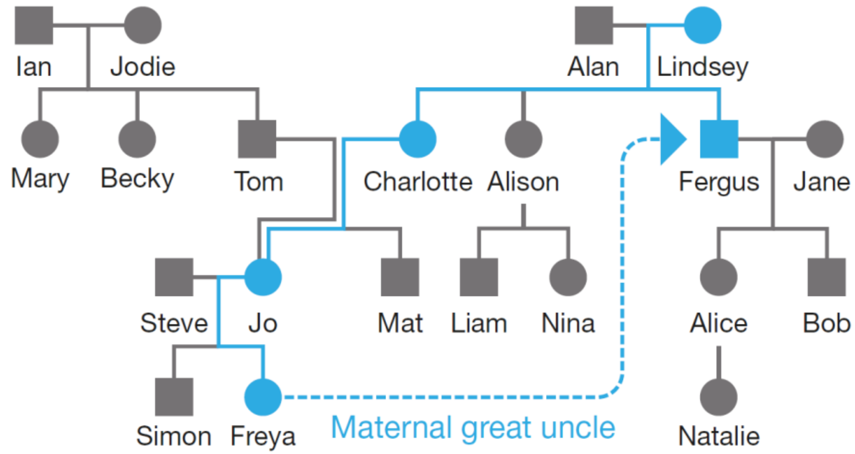
[interactive example: https://distill.pub/2016/augmented-rnns/#neural-turing-machines]

DNC experiments (Graves et al. 2016)

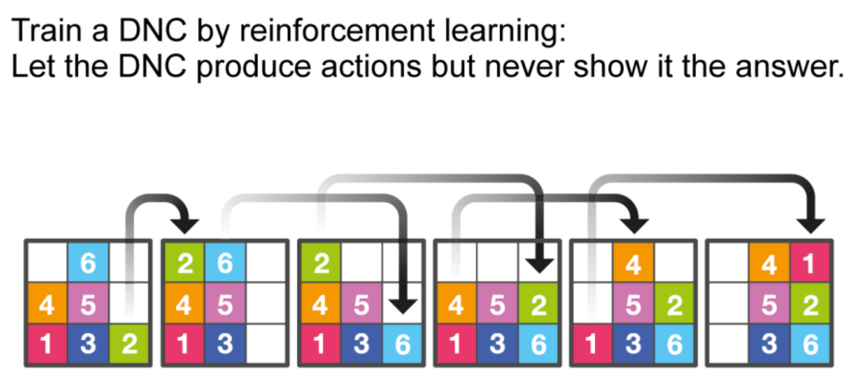
- Trained on
  - graph tasks for ***generalisation***
  - game tasks for ***action generation***

Outcome:

- Can learn algorithms
  - ***Transfer*** representations to other situations
  - Out-performed LSTMs on difficult sequence tasks



Family tree task



Block puzzle task

[Graves et al. 2016 / https://deepmind.com]

DNC analysis and summary

- Advantages:
  - Accessing all ***memory*** at once keeps the network ***differentiable***
  - For toy problems (so far): ***transfer learning***!
- Disadvantages:
  - Sequence learning tasks still "***easy***"
  - Brand new research!
    Not sure if promises are kept

- Alternative mechanisms to access memory are just getting researched – ***stay tuned***!
  - Neural Random Access Machines (Kurach et al.)
  - Stack-Augmented Recurrent Nets (Joulin et al.)

  …

Summary

- Recurrent Neural Networks with different constraints are ***efficient neural methods*** for various tasks and
  - Can reduce ***side effects*** of gradient descent methods
    ⇨ reduce vanishing/exploding gradient problem
  - Can make use of ***specific context*** information
  - Can ***approximate*** key ***patterns*** of time-series/sequences
    ⇨ find and use timescales in the data
- Offer high degree of ***noise robustness*** – even to significant disturbations in the sequences
- Allow ***general neural architectures*** to be developed

Further reading

- Wermter, S., Panchev, C. and Arevian, G., Hybrid Neural Plausibility Networks for News Agents. In AAAI/IAAI, pp. 93-98, 1999.
- Arevian, G. Recurrent Neural Networks for Robust Real-World Text Classification. *IEEE/WIC/ACM International Conference on Web Intelligence WI07,* pp. 326-329, 2007
- Greff, K., et al., LSTM: A search space odyssey. *arXiv preprint arXiv:1503.04069*, 2015.
- Yamashita, Y., & Tani, J. Emergence of functional hierarchy in a multiple timescale neural network model: a humanoid robot experiment. *PLoS Comput Biol*, *4*(11), 2008.
- Koutník, J., et al., A Clockwork RNN, *Proceedings of the 31st International Conference on Machine Learning, ICML*. Beijing, China, 2014.
- Graves, Alex, et al. Hybrid computing using a neural network with dynamic external memory. *Nature* 538(7626), 2016.