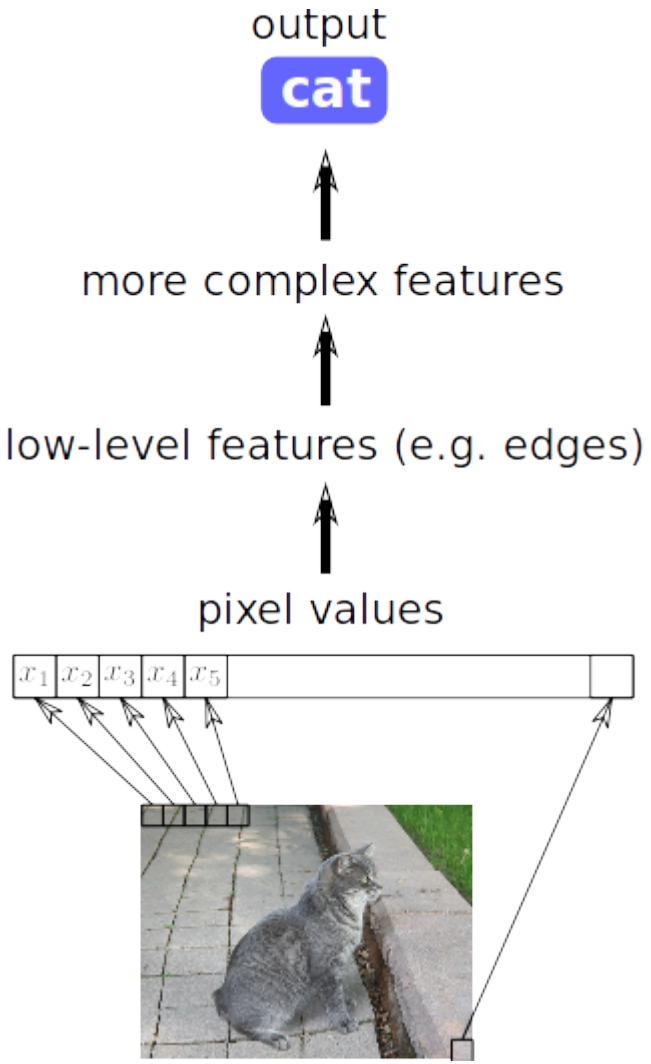
SoSe 2018: Deep Neural Networks

Lecture 8: Stateful DNNs

Machine Learning Group
Technische Universität Berlin

Recap: Feedforward Neural Networks



Example:

Prediction from a single static image.

Neural Networks for Time

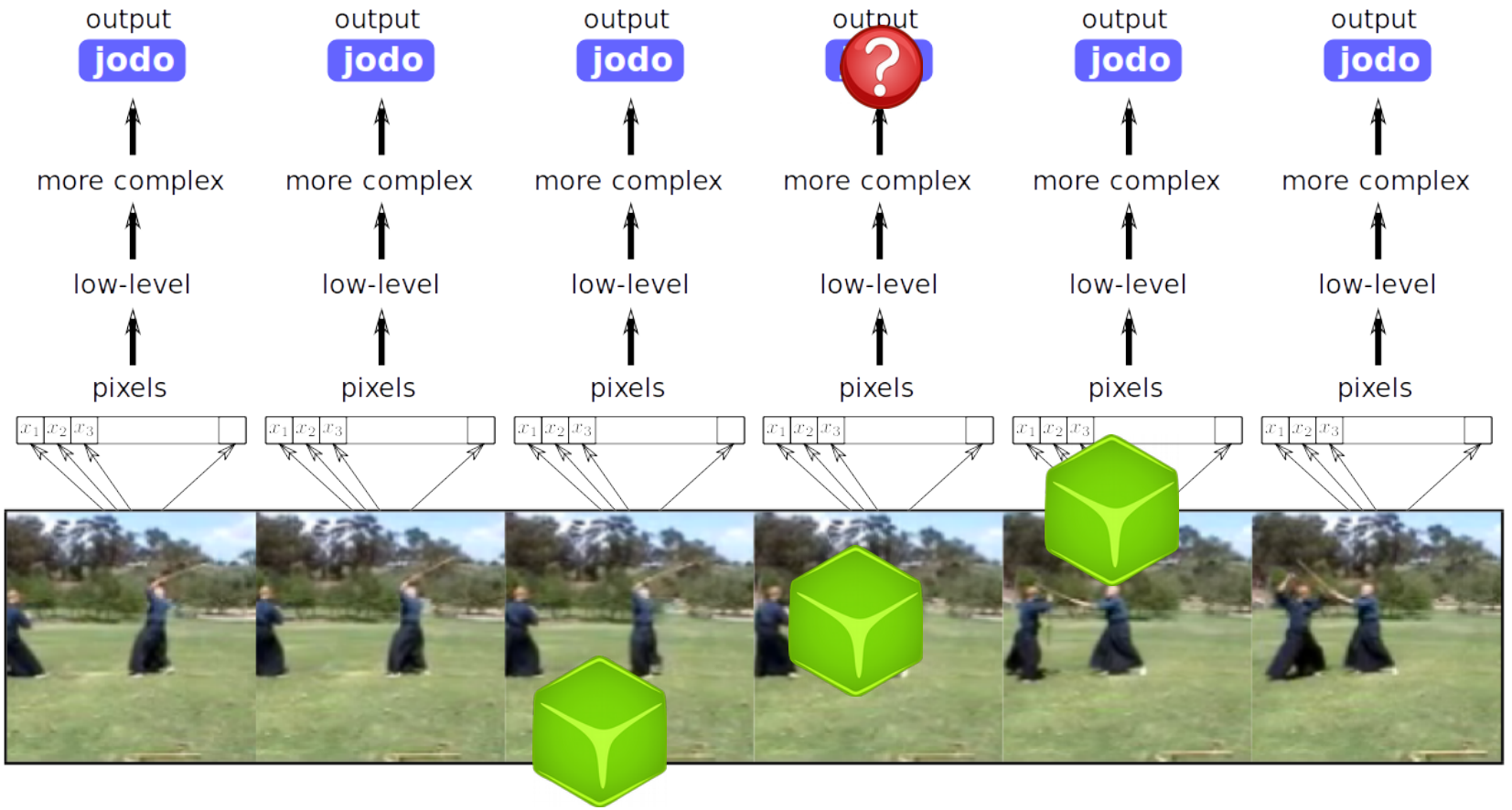
Many signals (e.g. videos, motion data, speech, text, ...) have a time component.

Neural Networks for Time

Basic approach, predict each time step individually.

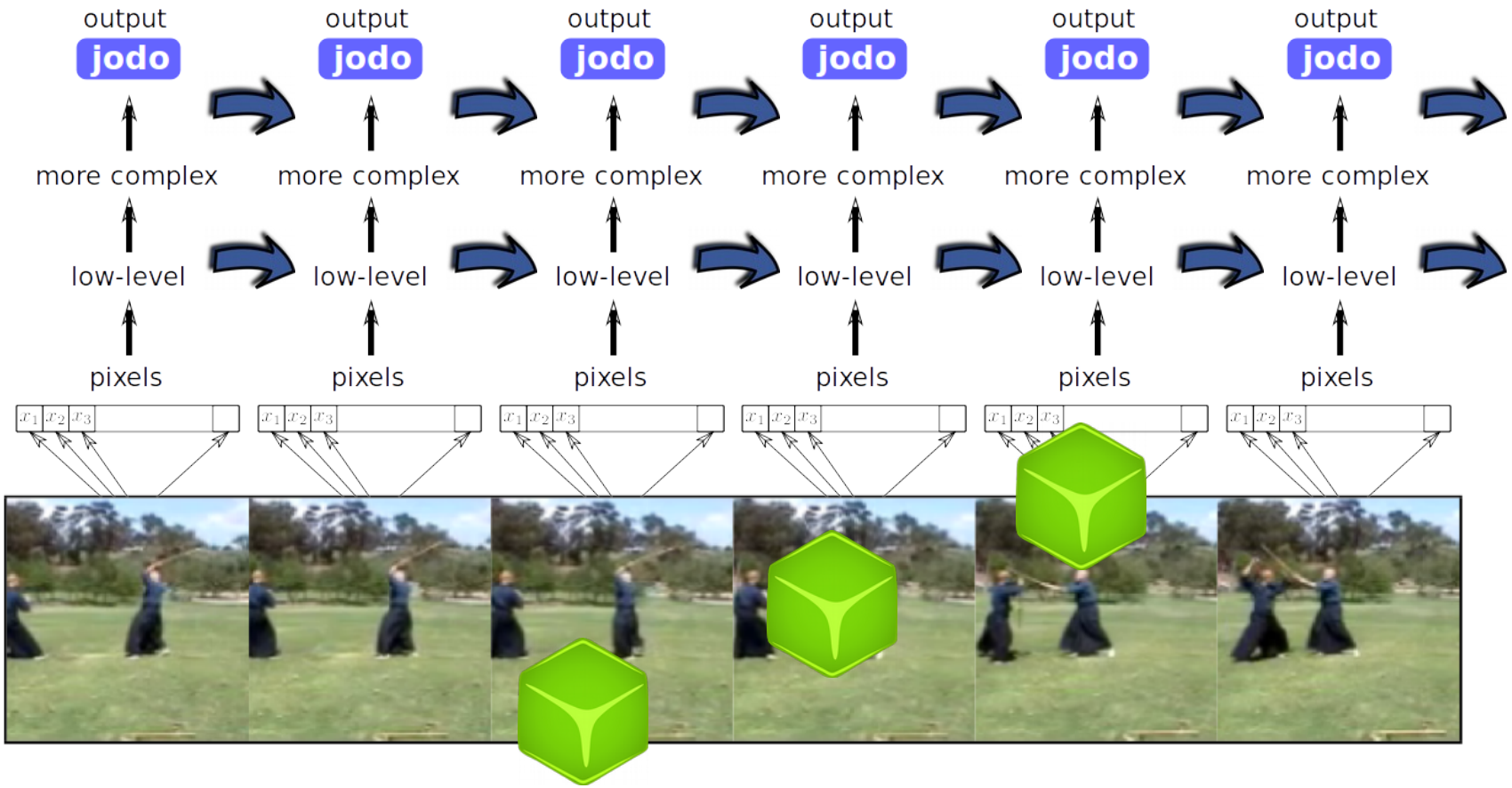
Neural Networks for Time

Basic approach not robust to e.g. occlusion.

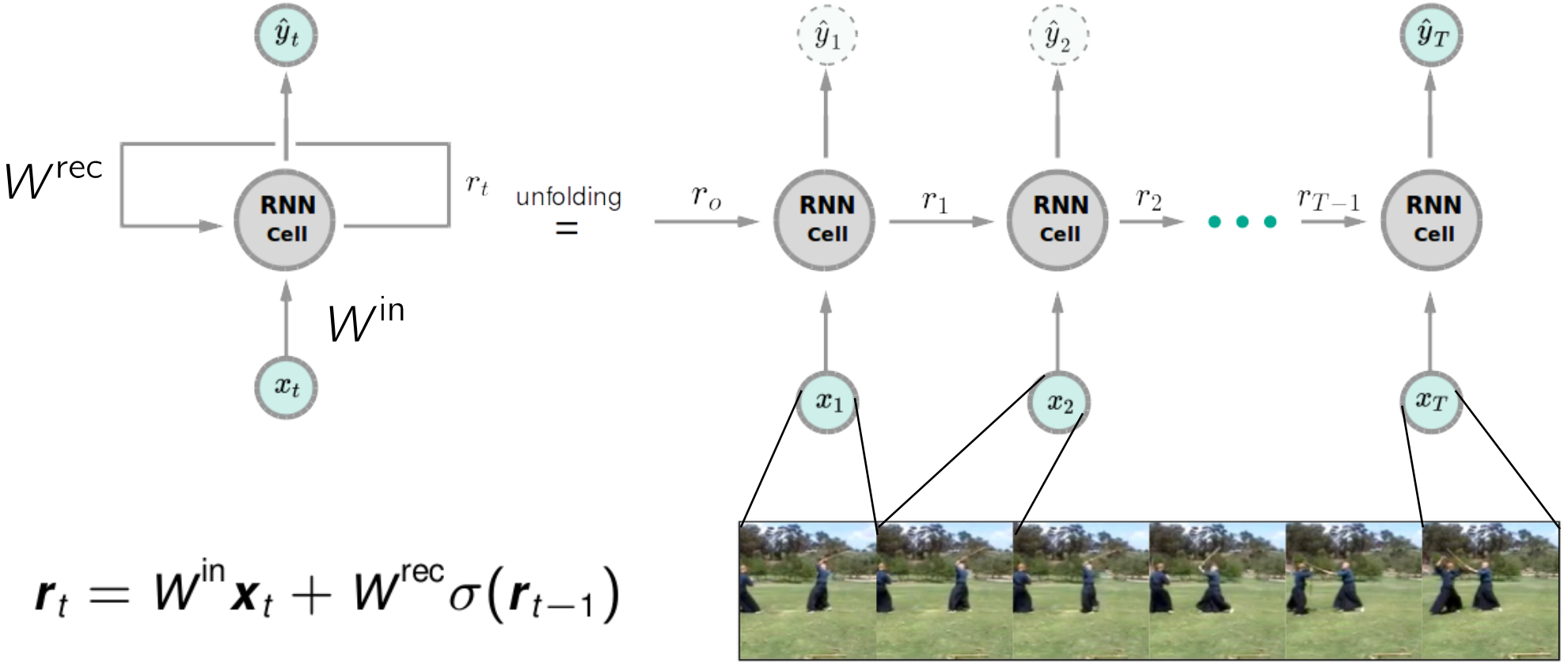


Neural Networks for Time

Idea: incorporate recurrent connections that maintain underlying dynamics.



RNN Basics



$$\boldsymbol{r}_t = W^{\text{in}}\boldsymbol{x}_t + W^{\text{rec}}\sigma(\boldsymbol{r}_{t-1})$$

RNN Basics (Forward Pass)

Assume that $\theta = \{W^{\text{in}}, W^{\text{rec}}\}$, $\boldsymbol{r}_0 = \mathbf{0}$, and the **forward pass** is constructed as follows:

$$\boldsymbol{r}_1 = W^{\text{in}}\boldsymbol{x}_1 + W^{\text{rec}}\sigma(\boldsymbol{r}_0)$$

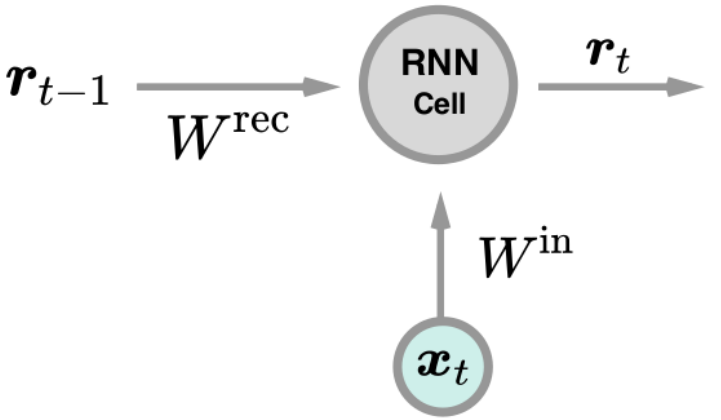
$$\vdots$$

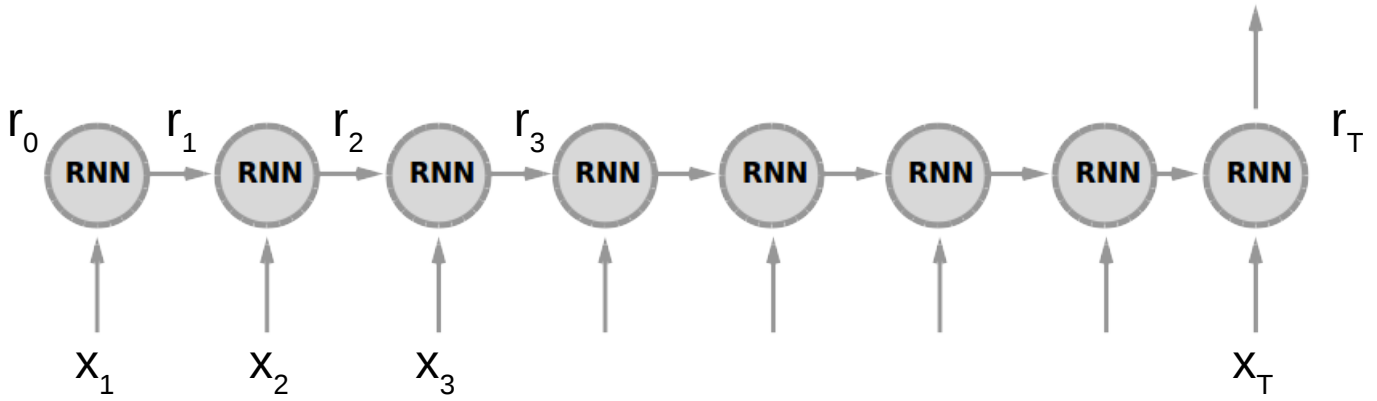
$$\boldsymbol{r}_t = W^{\text{in}}\boldsymbol{x}_t + W^{\text{rec}}\sigma(\boldsymbol{r}_{t-1})$$

$$\vdots$$

$$\boldsymbol{r}_T = W^{\text{in}}\boldsymbol{x}_T + W^{\text{rec}}\sigma(\boldsymbol{r}_{T-1})$$

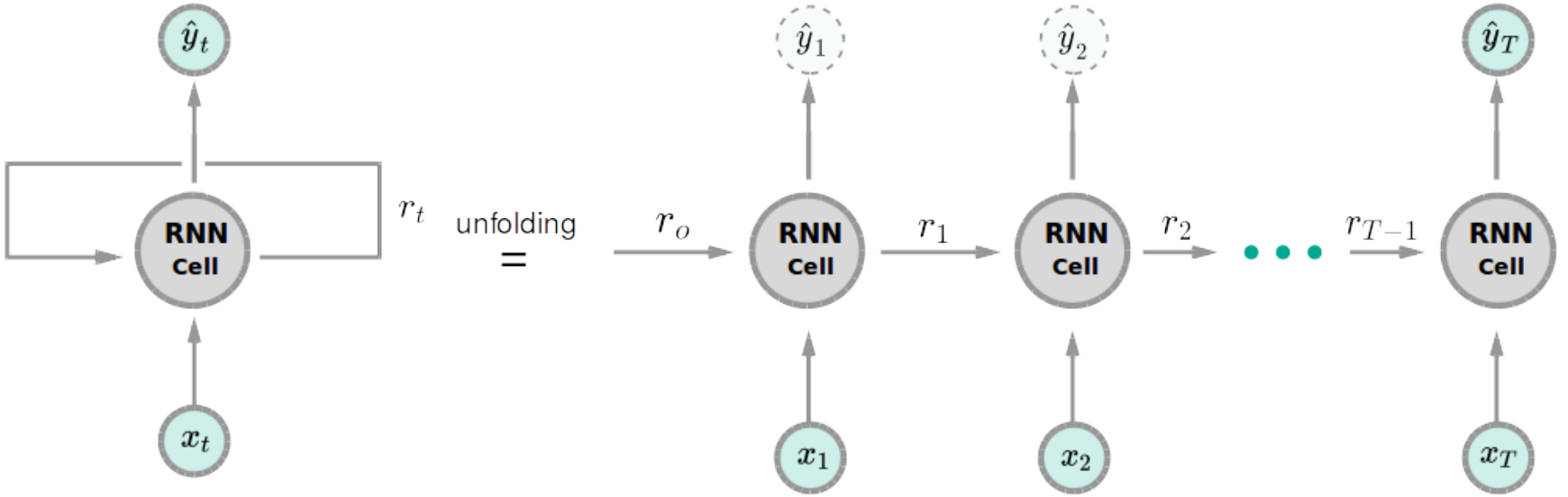
$$\hat{\boldsymbol{y}} = \boldsymbol{r}_T$$



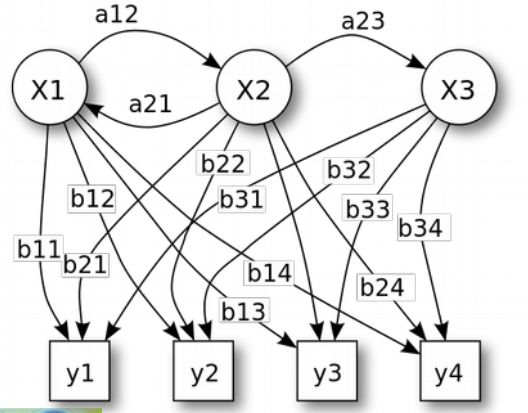


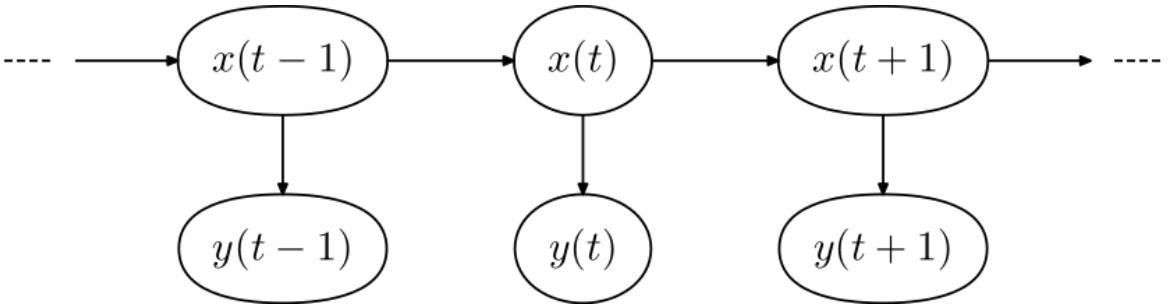
RNNs vs. Hidden Markov Models (HMMs)

RNNs



HMMs





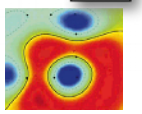
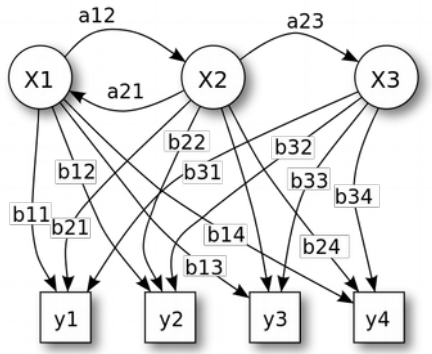


Image source wikipedia.org user: Tdunning

Recurrent Neural Networks vs. HMMs

HMMs

Image source
wikipedia.org
user: Tdunning



Discrete state space

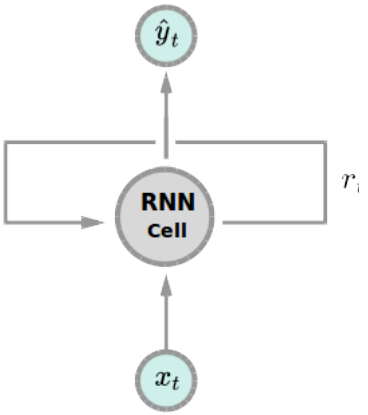
$$S = \{s_1, \ldots, s_h\}$$

Probabilistic transitions with transition model encoded in a matrix

$$A : h \times h$$

No inputs

RNNs



Continuous state space

$$S = \mathbb{R}^h$$

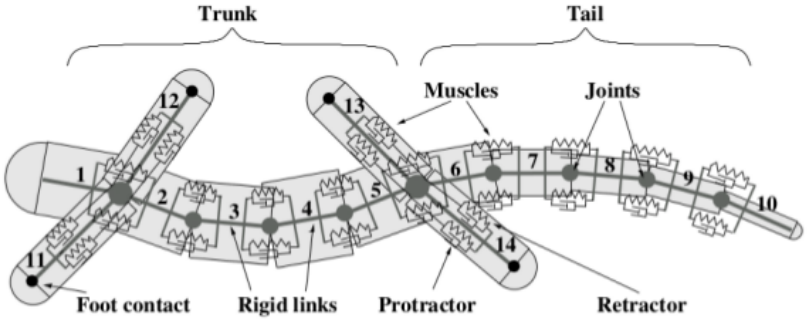
Deterministic transitions with transition model encoded by a parameterized function

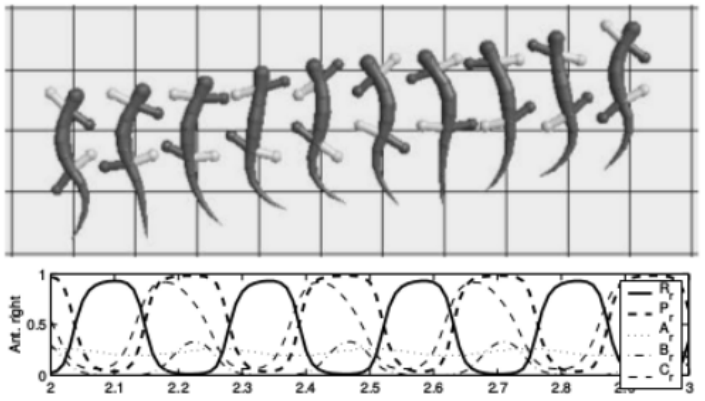
$$f : \mathbb{R}^h \to \mathbb{R}^h$$

Receives inputs

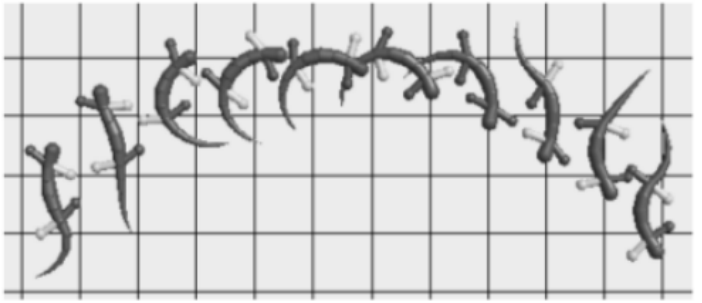
RNN Application: Salamander Motion

Experiments by Ijspeert et al. (2001):





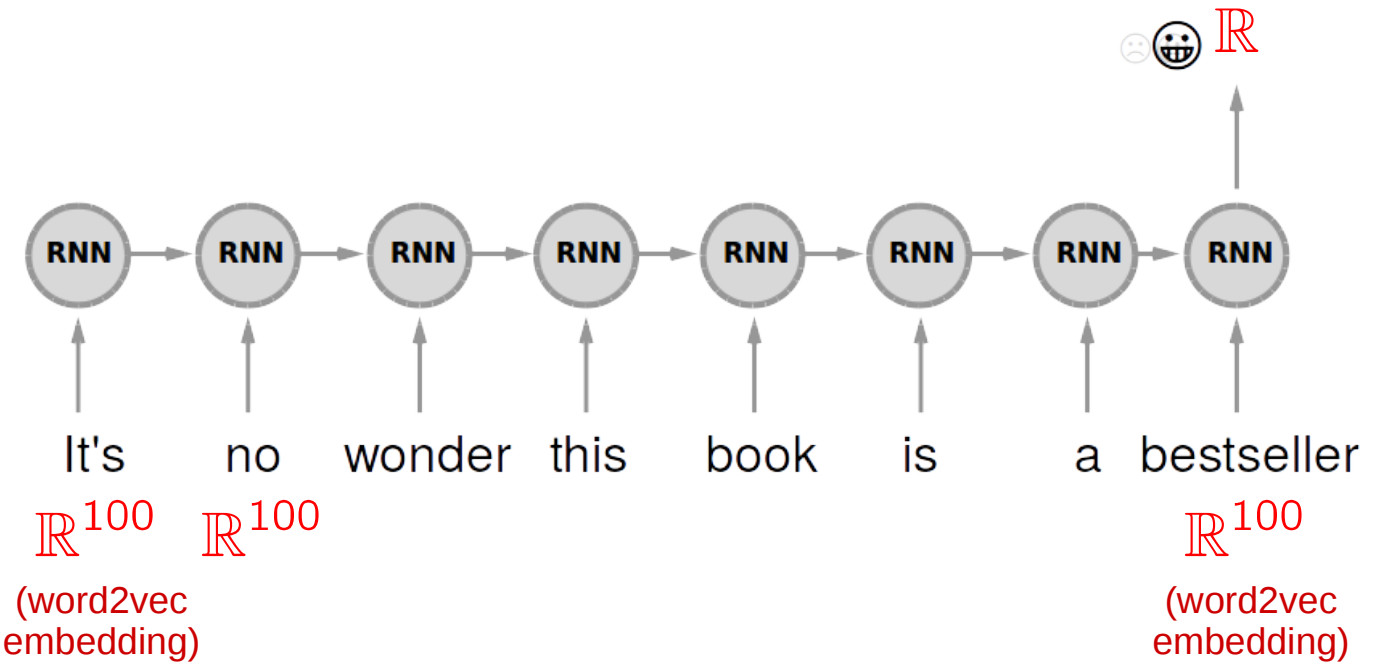
Free regime of the RNN



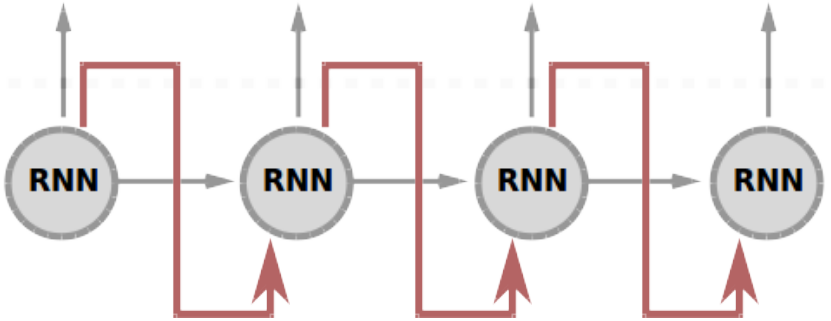
Some neurons of the RNN forced to take specific values.

Images from Ijspeert et al. 2001, A Connectionist Central Pattern Generator for the Aquatic and Terrestrial Gaits of a Simulated Salamander

RNN Application: Sentiment Analysis



RNN Application: Random Text Generation



RNN output can be given as input to condition generation on what has been produced at previous time steps.

Lemma 0.3. *Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.*

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

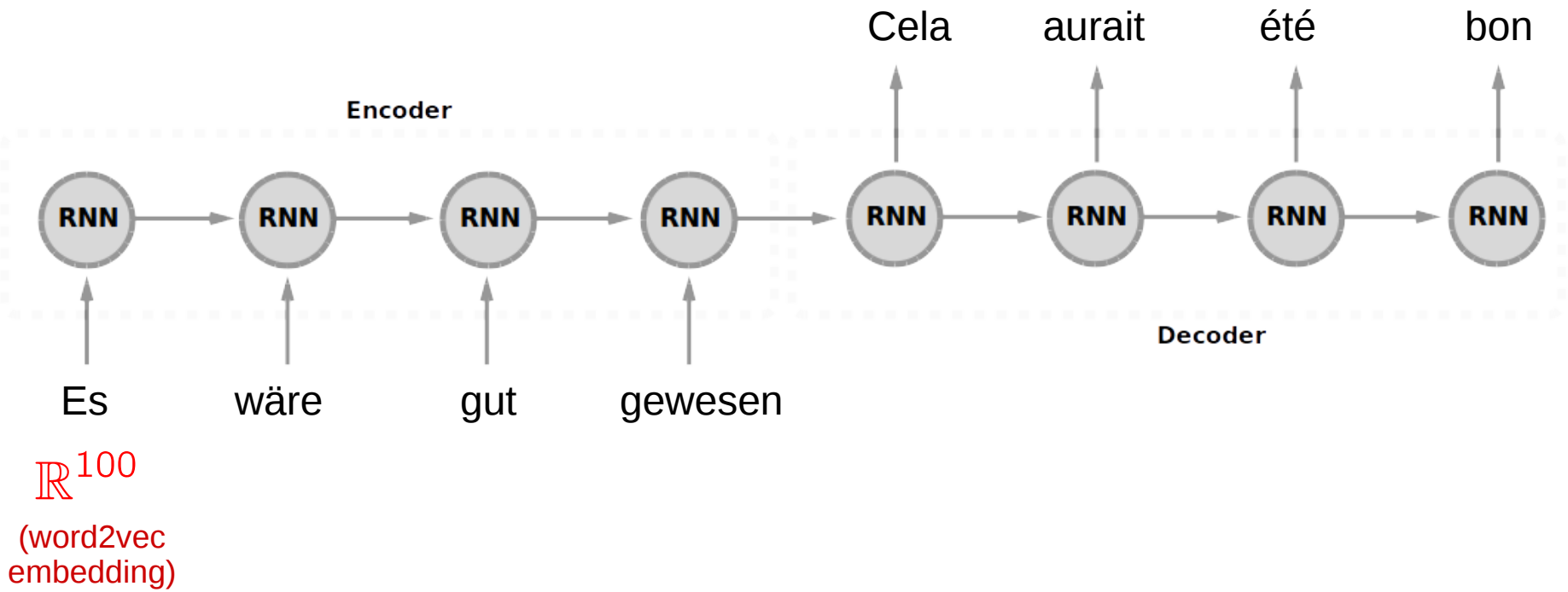
$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

be a morphism of algebraic spaces over S and Y.

Proof. Let X be a nonzero scheme of X. Let X be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

(1) $\mathcal{F}$ is an algebraic space over S.
(2) If X is an affine open covering.

RNN Application: Machine Translation

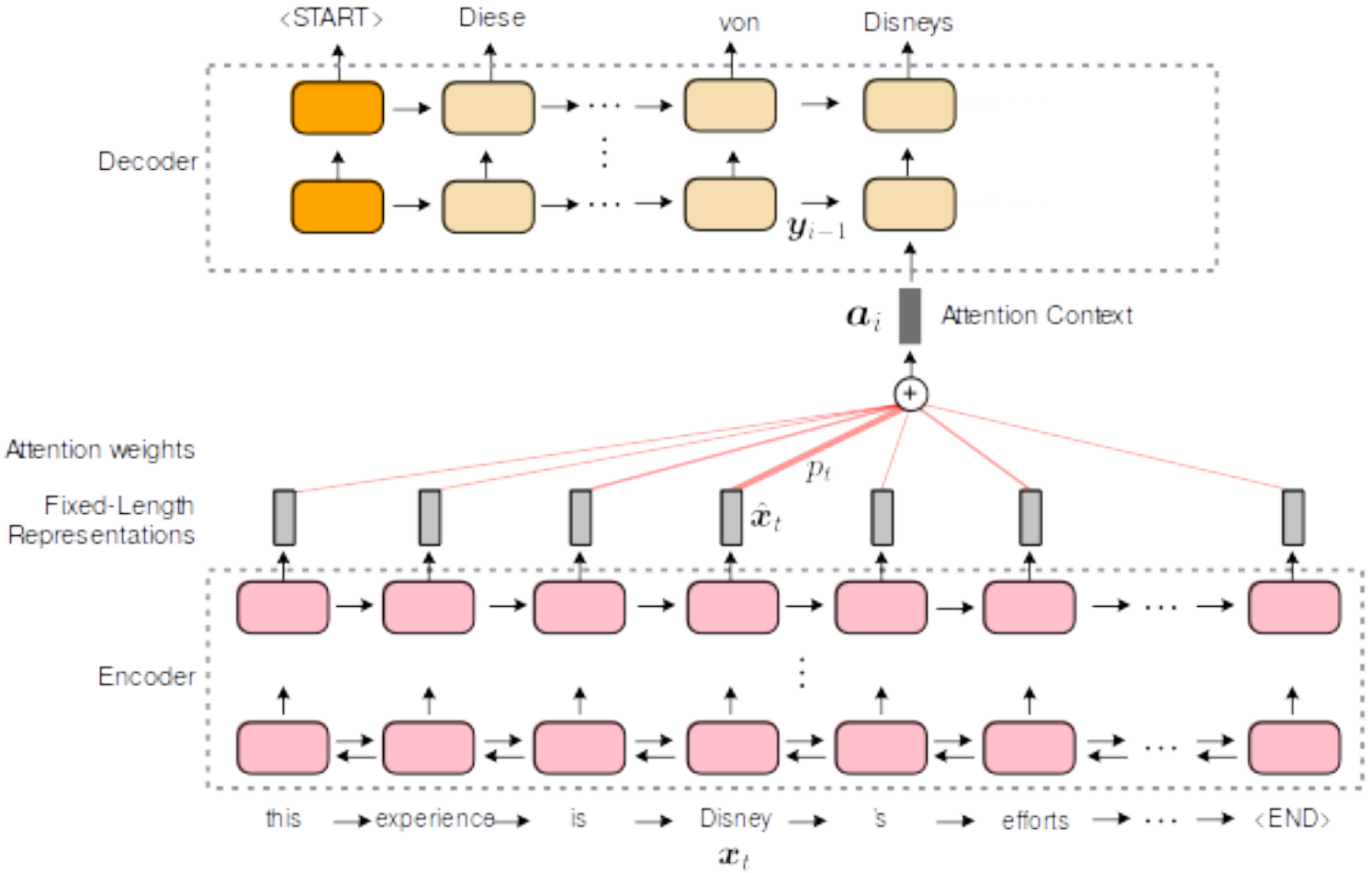


Notes:

- Need to finish reading the sentence to start the translation.
- Complete sentence meaning is being encoded in the RNN state.

Machine Translation

- Google's Neural Machine Translation System [WSC + 16].



Source: This experience is part of Disney's efforts to enlarge the audience.

Target: Diese Erfahrung ist Teil von Disneys Bemühungen, das Publikum zu vergrößern.

Machine Translation

- Google's Neural Machine Translation System [Wu'16].

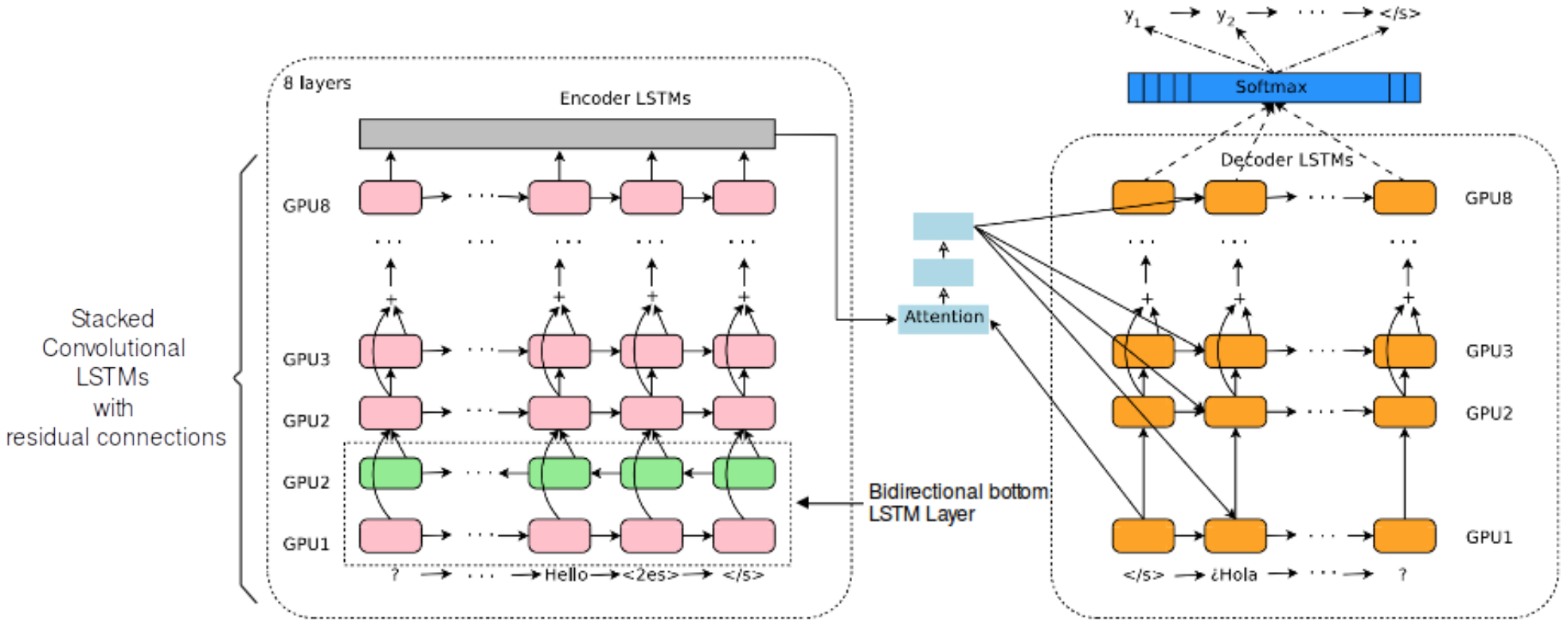


Image from Wu'16: Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation

RNNs for Image Segmentation

- An RNN for image segmentation [McIntosh'17].

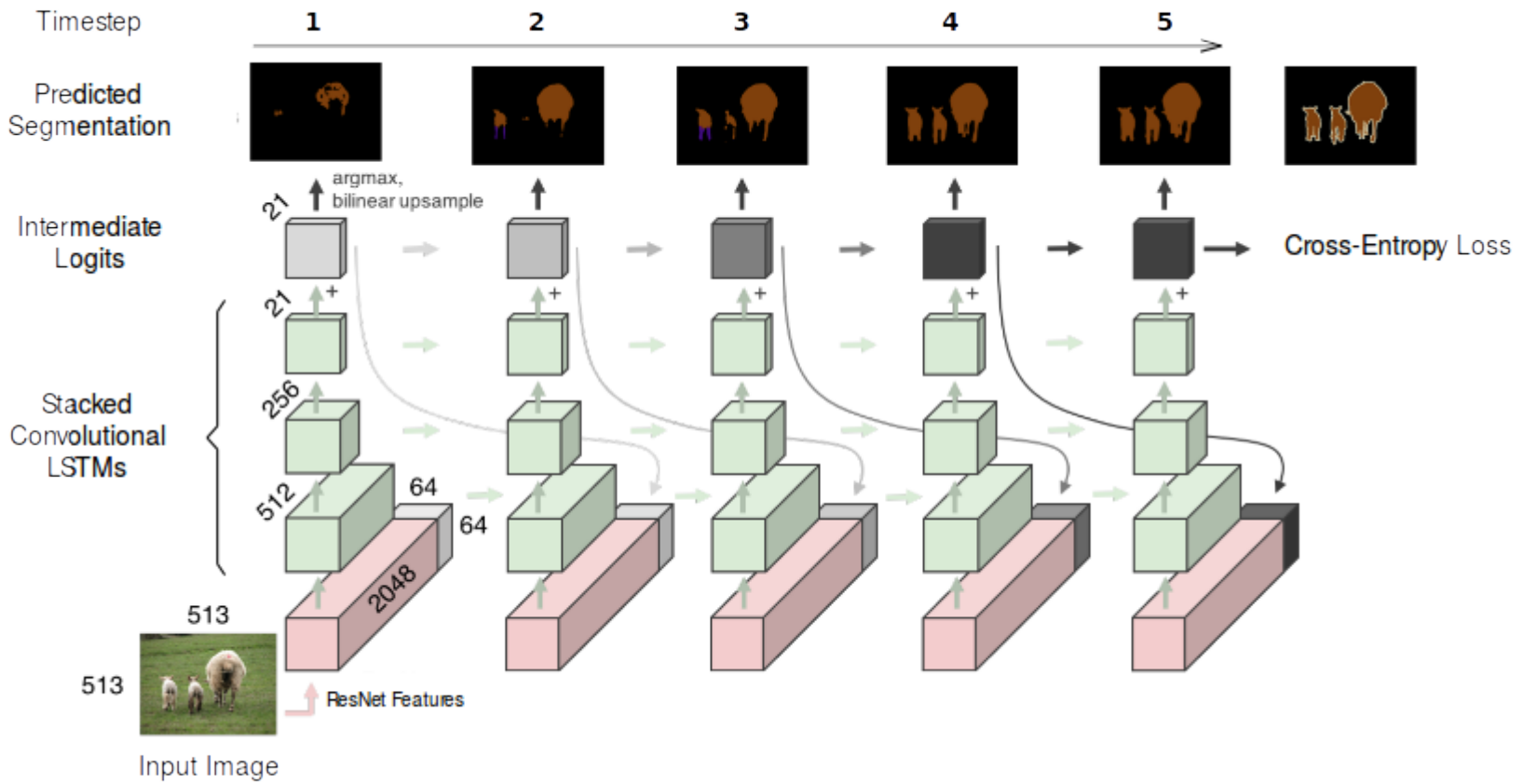


Image from McIntosh'17 Recurrent Segmentation for Variable Computational Budgets

RNN Basics Recap (Forward Pass)

Assume that $\theta = \{W^{\text{in}}, W^{\text{rec}}\}$, $\boldsymbol{r}_0 = \mathbf{0}$, and the **forward pass** is constructed as follows:

$$\boldsymbol{r}_1 = W^{\text{in}}\boldsymbol{x}_1 + W^{\text{rec}}\sigma(\boldsymbol{r}_0)$$

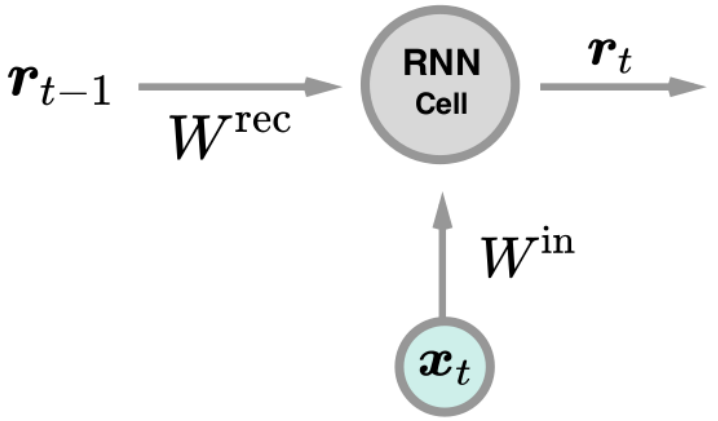
$$\vdots$$

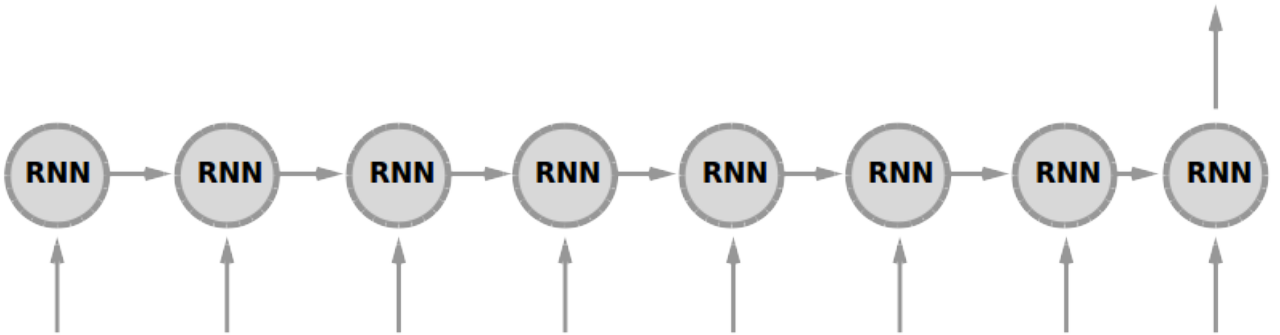
$$\boldsymbol{r}_t = W^{\text{in}}\boldsymbol{x}_t + W^{\text{rec}}\sigma(\boldsymbol{r}_{t-1})$$

$$\vdots$$

$$\boldsymbol{r}_T = W^{\text{in}}\boldsymbol{x}_T + W^{\text{rec}}\sigma(\boldsymbol{r}_{T-1})$$

$$\hat{\boldsymbol{y}} = \boldsymbol{r}_T$$





Extracting a RNN from a Dynamical System

Example: Linear dynamical system

$$\frac{\partial}{\partial t}\boldsymbol{r}(t) = A \cdot \boldsymbol{r}(t)$$

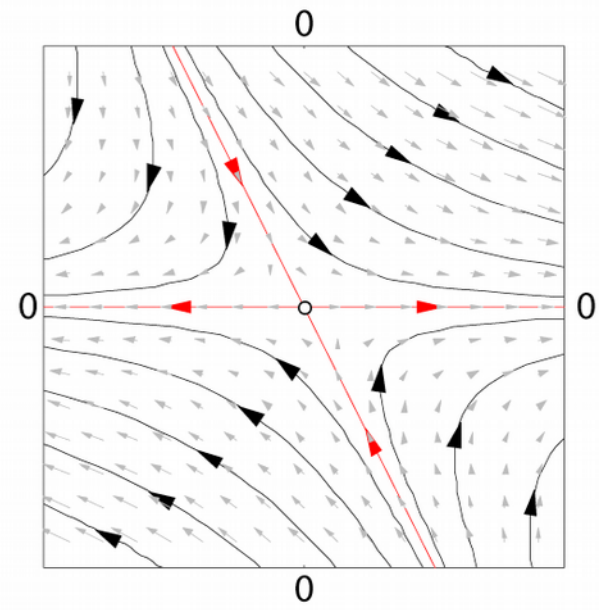


Image source: Wikipedia, user: XaosBits

Step 1: Euler discretization

$$\frac{\boldsymbol{r}(t_{n+1}) - \boldsymbol{r}(t_n)}{t_{n+1} - t_n} = A \cdot \boldsymbol{r}(t_n)$$

Step 2: Isolate the last time step

$$\boldsymbol{r}(t_{n+1}) = \boldsymbol{r}(t_n) + (t_{n+1} - t_n) \cdot A \cdot \boldsymbol{r}(t)$$

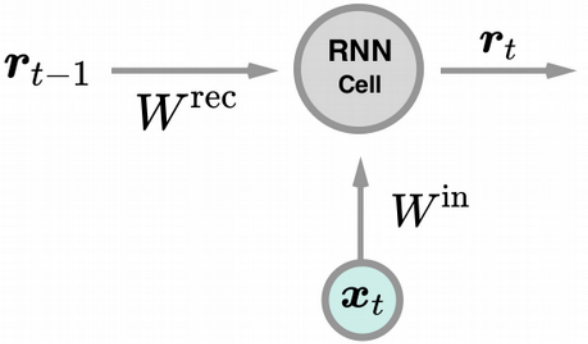
Step 3: Write in standard format

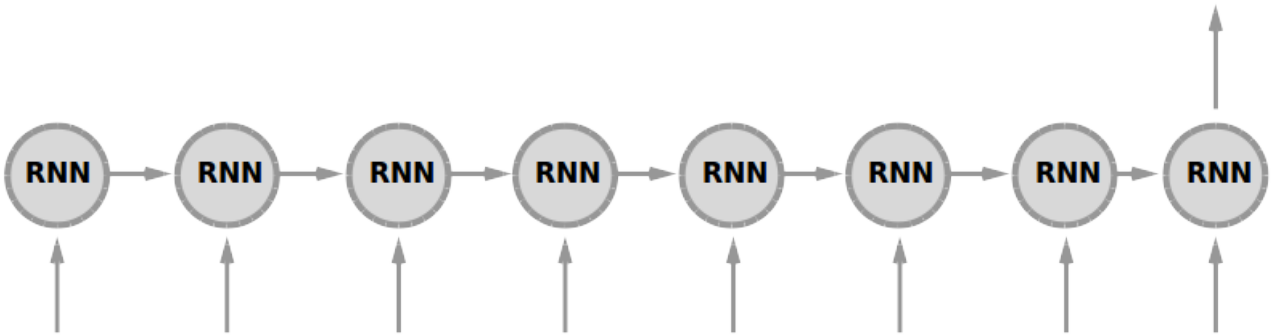
$$\boldsymbol{r}(t_{n+1}) = W_{\text{rec}} \cdot \boldsymbol{r}(t_n) \qquad W_{\text{rec}} = I + (t_{n+1} - t_n) \cdot A$$

Training the RNN: Computing the Gradient

$$
\begin{aligned}
\boldsymbol{r}_1 &= W^{\text{in}}\boldsymbol{x}_1 + W^{\text{rec}}\sigma(\boldsymbol{r}_0) \\
&\vdots \\
\boldsymbol{r}_t &= W^{\text{in}}\boldsymbol{x}_t + W^{\text{rec}}\sigma(\boldsymbol{r}_{t-1}) \\
&\vdots \\
\boldsymbol{r}_T &= W^{\text{in}}\boldsymbol{x}_T + W^{\text{rec}}\sigma(\boldsymbol{r}_{T-1}) \\
\hat{\boldsymbol{y}} &= \boldsymbol{r}_T
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial E}{\partial \theta} &= \sum_{k=1}^{T} \frac{\partial E}{\partial \boldsymbol{r}_T}\frac{\partial \boldsymbol{r}_T}{\partial \boldsymbol{r}_k}\frac{\partial^+ \boldsymbol{r}_k}{\partial \theta} \\
\frac{\partial \boldsymbol{r}_T}{\partial \boldsymbol{r}_k} &= \prod_{T \geq i > k} \frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{r}_{i-1}} \\
&= \prod_{T \geq i > k} \underbrace{(W^{\text{rec}})^T}_{!} \operatorname{diag}(\sigma'(\boldsymbol{r}_{i-1}))
\end{aligned}
$$



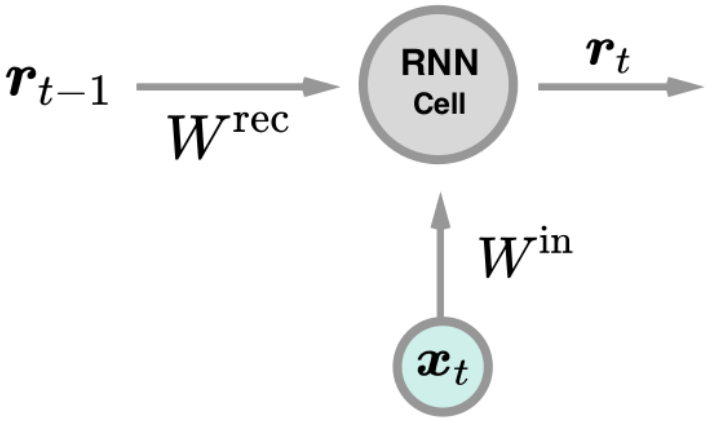


Optimizing RNNs: Eigenvalues View

$$\frac{\partial E}{\partial \theta} = \sum_{k=1}^{T} \frac{\partial E}{\partial \boldsymbol{r}_T} \frac{\partial \boldsymbol{r}_T}{\partial \boldsymbol{r}_k} \frac{\partial^+ \boldsymbol{r}_k}{\partial \theta}$$

$$\frac{\partial \boldsymbol{r}_T}{\partial \boldsymbol{r}_k} = \prod_{T \geq i > k} \frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{r}_{i-1}}$$

$$= \prod_{T \geq i > k} \underbrace{(W^{\text{rec}})^T}_{!} \operatorname{diag}(\sigma'(\boldsymbol{r}_{i-1}))$$



$\sigma' = 1$ assume the activations are linear

$$\prod_{T \geq i > k} \left(W^{\text{rec}}\right)^k = U \cdot \Lambda^{(T-k)} \cdot V$$

Optimizing RNNs: Eigenvalues View

$\sigma' = 1$ assume the activations are linear

$$\prod_{T \geq i > k} \left(W^{\text{rec}}\right)^k = \underbrace{U \cdot \Lambda^{(T-k)} \cdot V}_{\lambda_1, \ldots, \lambda_d}$$

Observations:

- Disbalance of gradient magnitude between different time steps $\frac{\lambda_1}{\lambda_1^{(T-k)}}$
- Bad conditioning at lower-time steps $\left(\frac{\lambda_1}{\lambda_d}\right)^{(T-k)}$

Optimizing RNNs: Eigenvalues View

Solution:

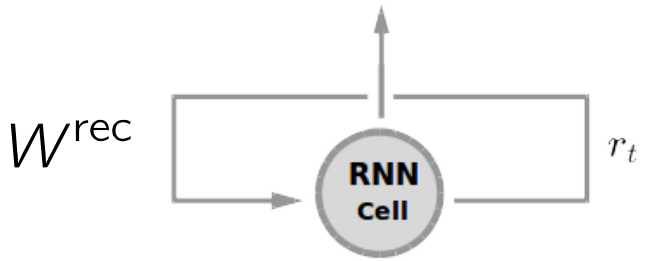
$$\frac{\lambda_1}{\lambda_1^{(T-k)}} \quad \rightarrow \quad \text{Normalize weights to make first eigenvalue a bit smaller than 1.}$$

$$\left(\frac{\lambda_1}{\lambda_d}\right)^{(T-k)} \quad \rightarrow \quad \text{Add a LOT of momentum.}$$

Note: eigenvalues view is only exact for linear models, what about nonlinear models?

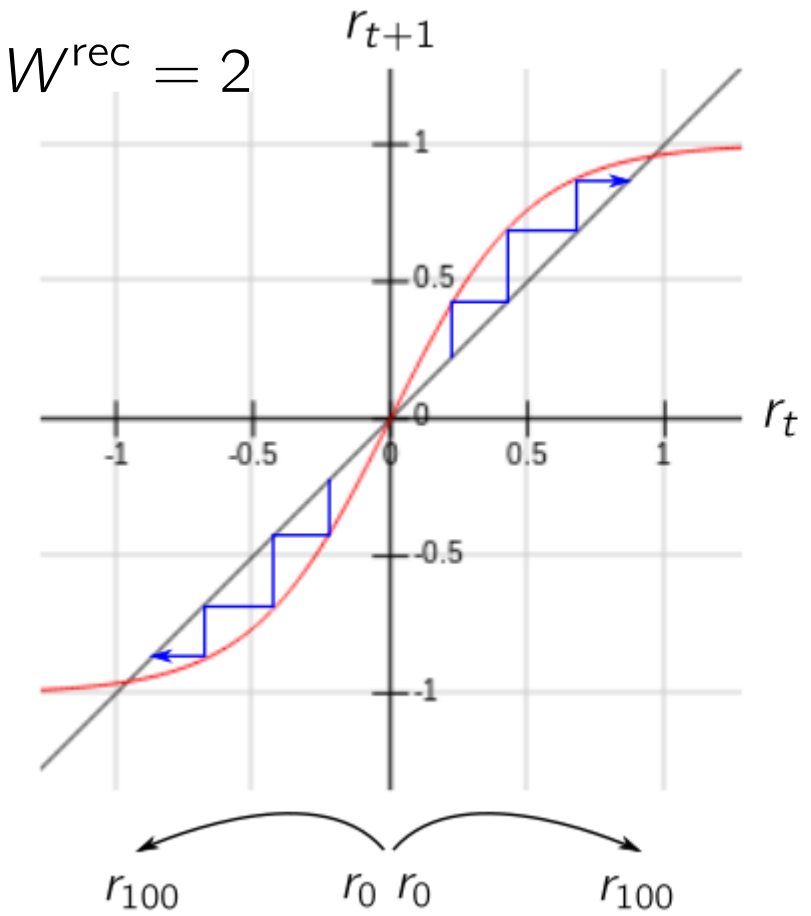
Optimizing RNNs: Dynamical System View

Most RNNs are nonlinear, let's take a different view at the problem:



$$r_t = \tanh(W^{\text{rec}} r_{t-1})$$

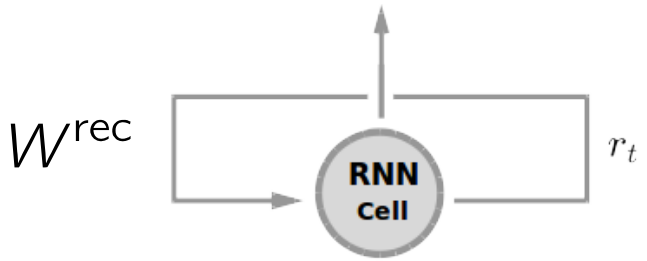
$$\frac{\partial r_{100}}{\partial r_0} = \lim_{r_0' \to r_0} \frac{r_{100}' - r_{100}}{r_0' - r_0}$$



birfucation at 0

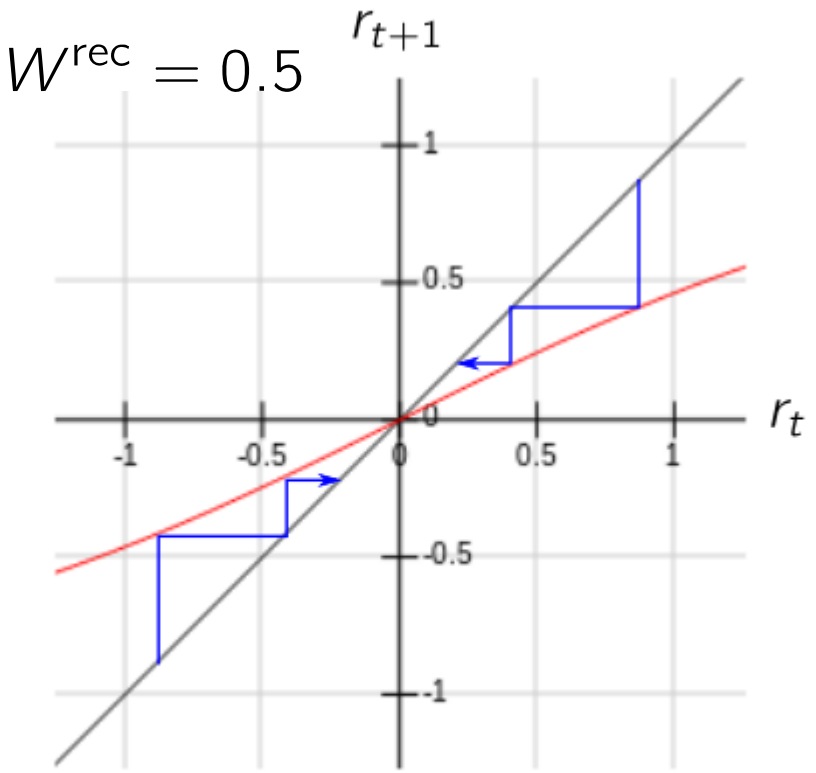
Optimizing RNNs: Dynamical System View

Small weights will reduce the number of bifurcations



$$r_t = \tanh(W^{\text{rec}} r_{t-1})$$

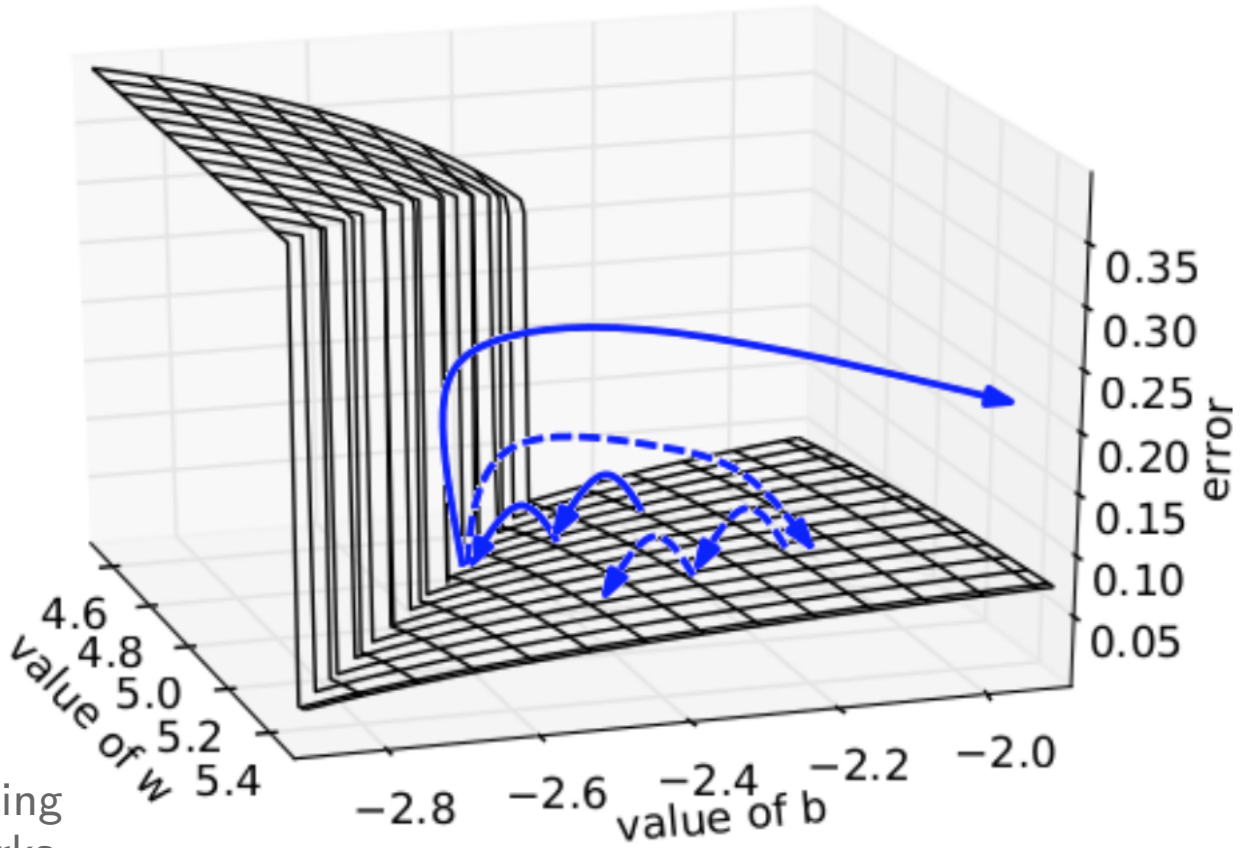
$$\frac{\partial r_{100}}{\partial r_0} = \lim_{r_0' \to r_0} \frac{r_{100}' - r_{100}}{r_0' - r_0}$$



no bifurcation

Optimizing RNNs: Dynamical System View

Bifurcations create cliffs in the error function. As a result, gradient descent will not work.

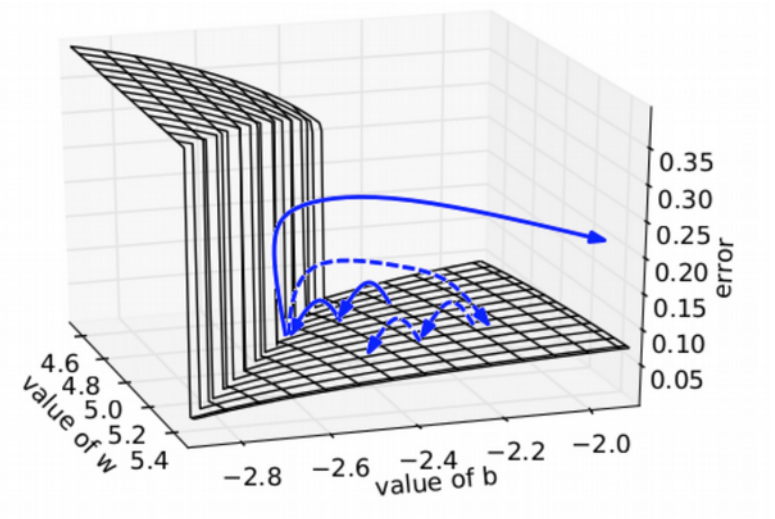


Source: Pascanu'13
On the difficulty of training Recurrent Neural Networks

Optimizing RNNs: Dynamical System View

Trick: gradient clipping (Mikolov'12, Pascanu'13):

$\epsilon \leftarrow threshold$;
$\hat{g} \leftarrow \nabla_\theta J$;
if $\|\hat{g}\| \geq threshold$ **then**
  $\hat{g} \leftarrow \frac{\epsilon}{\|\hat{g}\|}\hat{g}$;
end



Ignore the magnitude of the gradient, just look at the direction.

Related idea: RProp (Riedmiller'92)

The Difficulty of Training RNNs

RNNs have an instable gradient due to the recurrent nature of the model. Some tricks exists, but they do not solve the problem completely.

In the following, we consider two alternative approaches:

1. **Echo state networks** (only train the top layer)

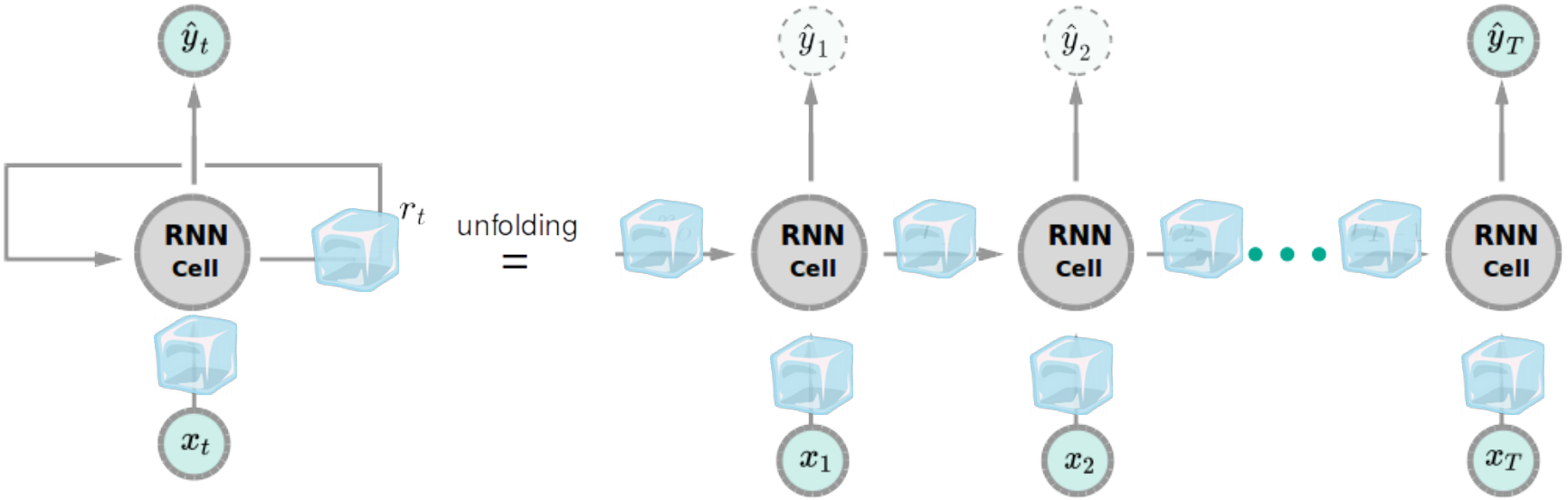
2. **LSTM** (better RNN architecture)

Echo State Networks (ESNs) [Jaeger'01]

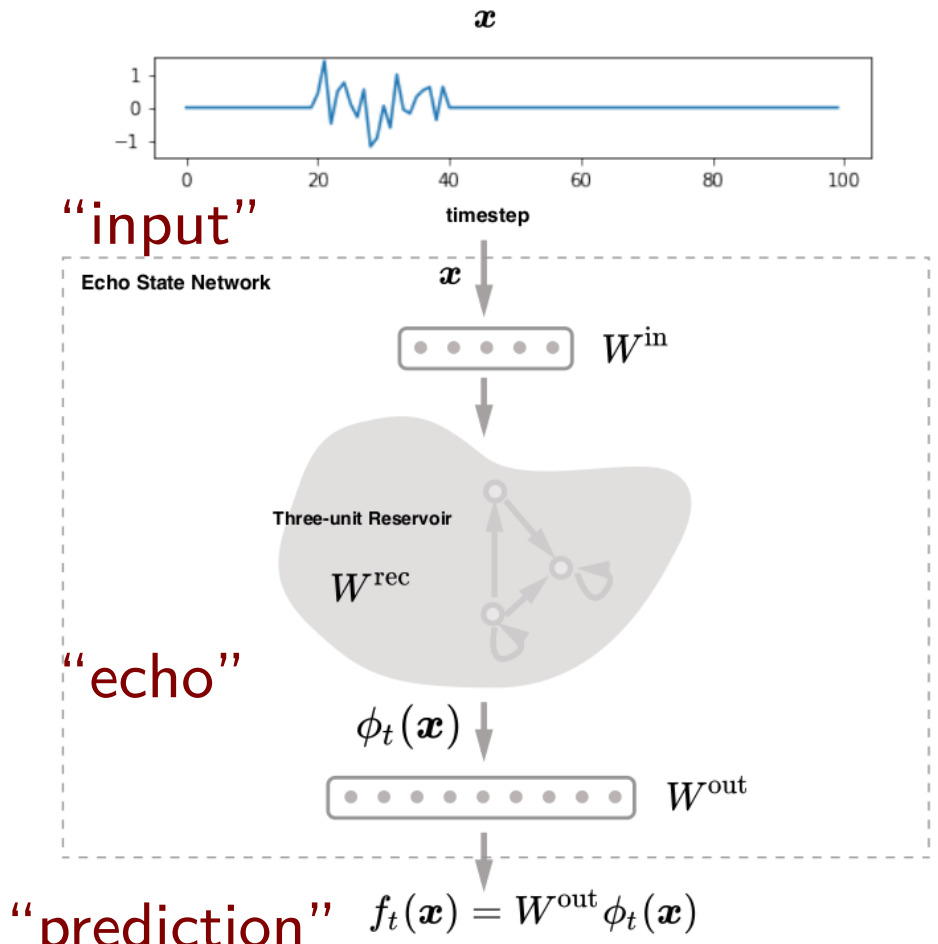
Special type of RNN where recurrent weights are initialized at random and kept fixed throughout training.

Only the output weights W^{out} are adapted.

Adapting the output weights is a simple convex problem, and can be easily optimized.



Echo State Networks (ESNs)



The name “echo state network” reflects the intuition that the network performs a time-dependent expansion (“echo”) of the input time series.

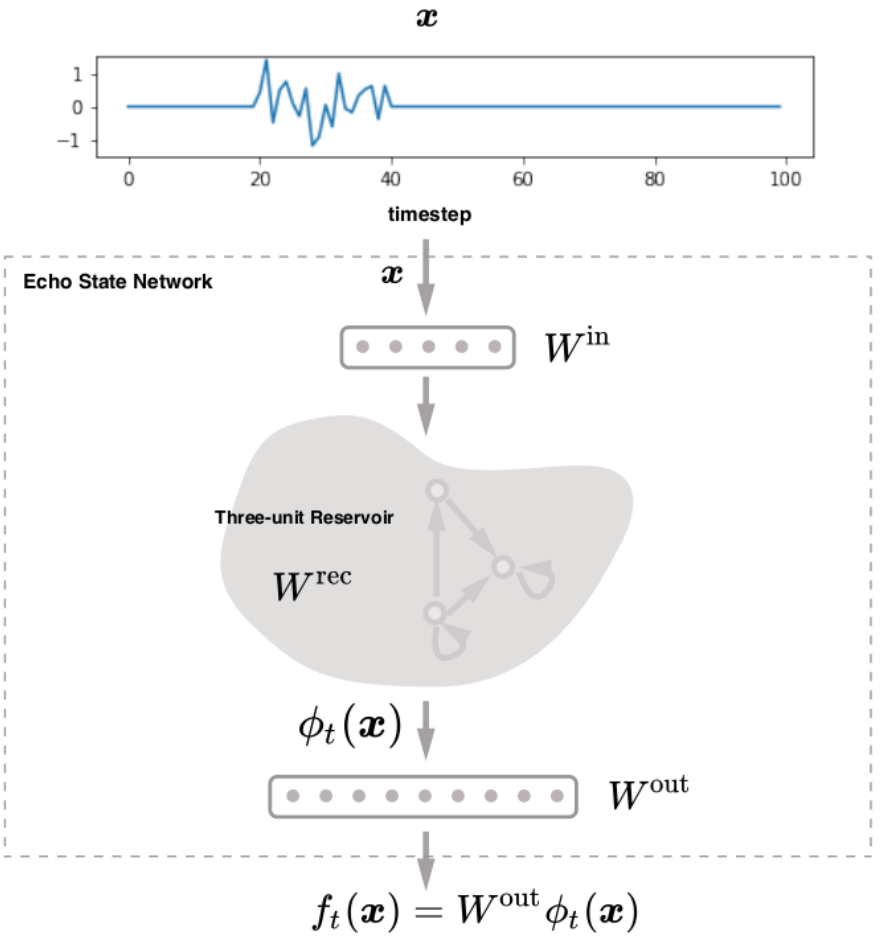
Ideally, the echo representation has produced features that are more suitable for predicting the input time series (e.g. its class).

Echo State Networks (ESNs)

Equations:

$$\tilde{\boldsymbol{h}}_t = \tanh(W^{\text{in}}\boldsymbol{x}_t + W^{\text{rec}}\boldsymbol{h}_{t-1}),$$
$$\boldsymbol{h}_t = (1-\alpha)\boldsymbol{h}_{t-1} + \alpha\tilde{\boldsymbol{h}}_t,$$
$$\hat{\boldsymbol{y}}_t = W^{\text{out}}\text{concat}(\boldsymbol{x}_t, \boldsymbol{h}_t),$$

where $W^{\text{rec}} \in \mathbb{R}^{M\times M}$ with M is the number of reservoir units, and $\alpha \in (0, 1]$ is the leaking rate.



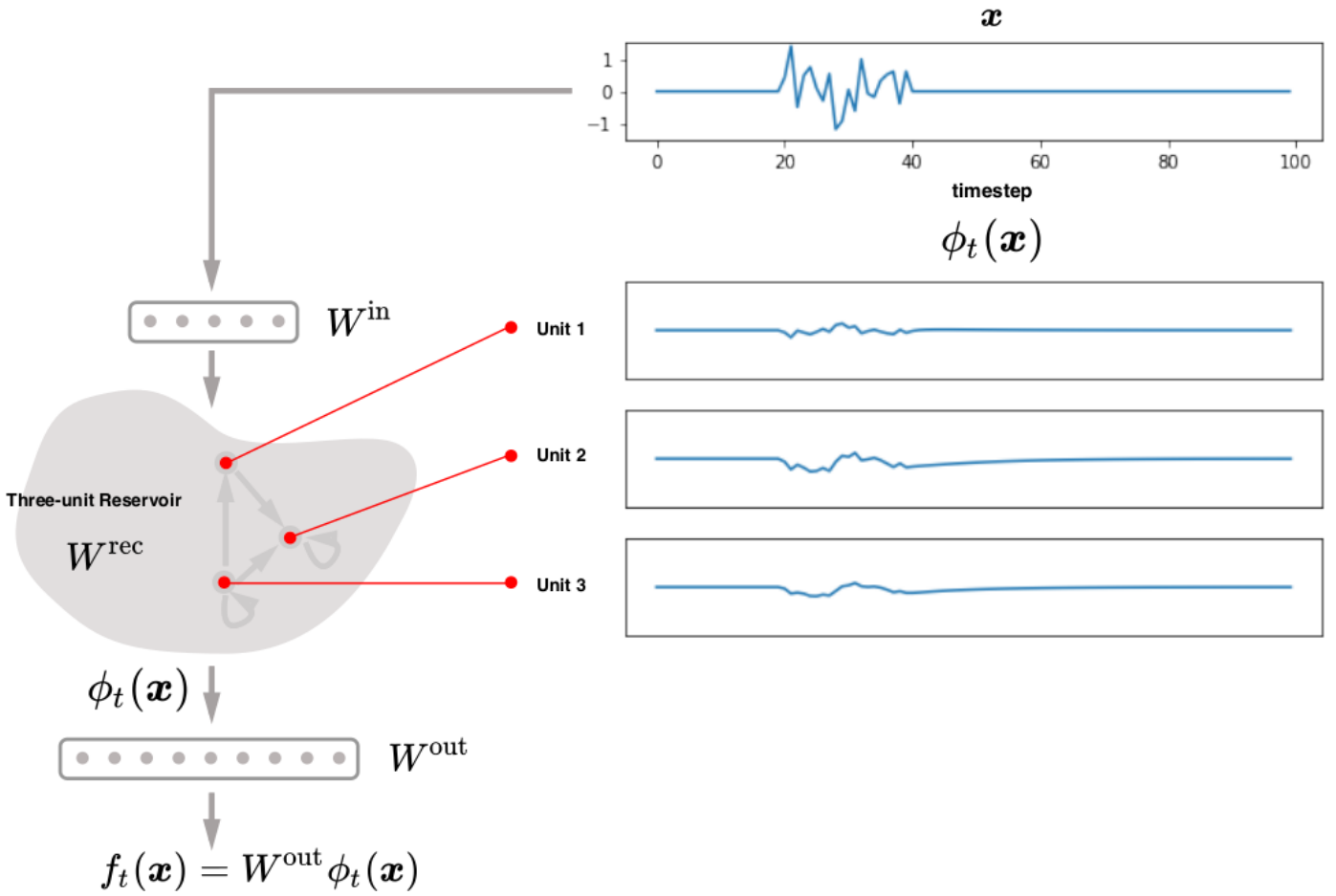
ESN: A Physical Analogy

input time signal

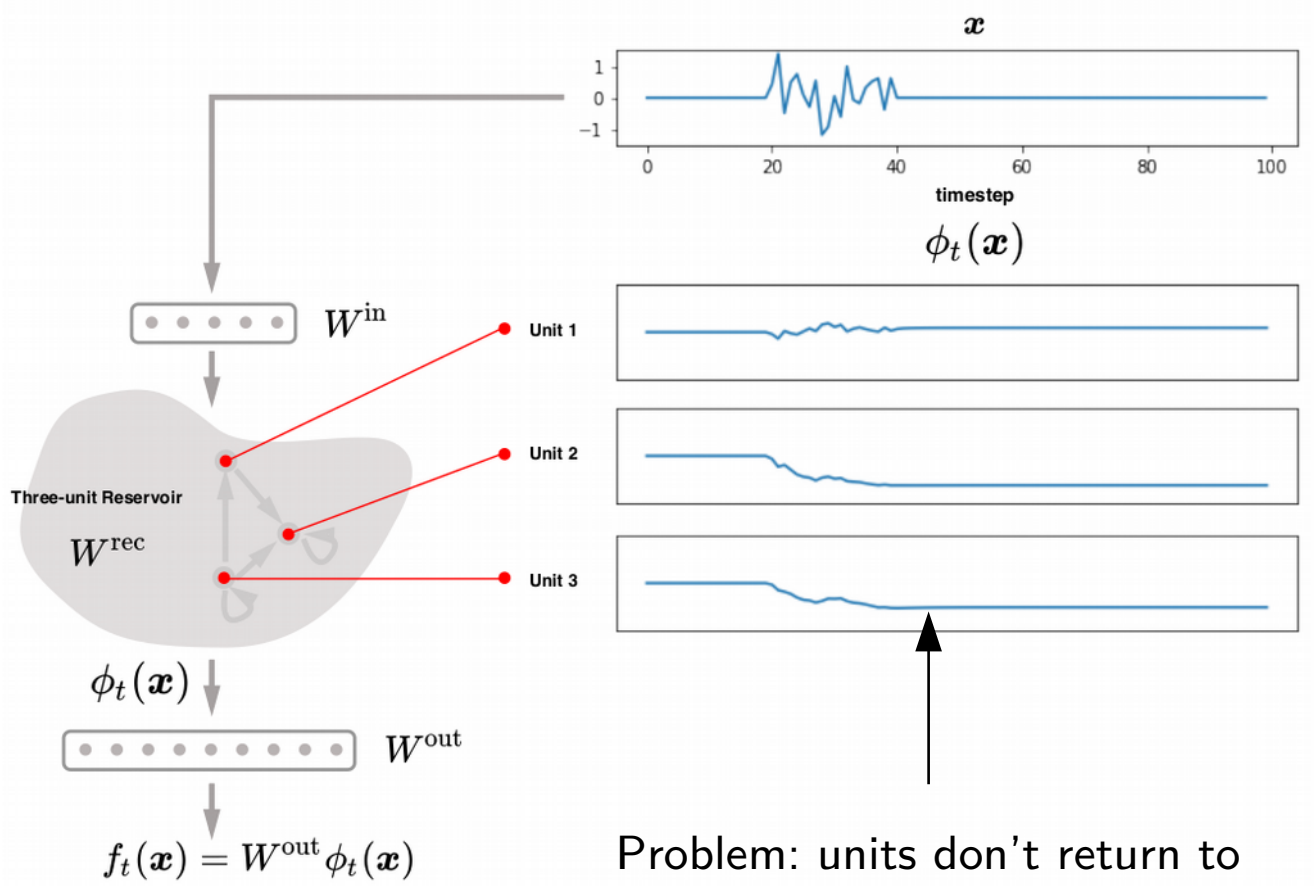
Image of ripples at a fixed time step: ripples

image classifier

ESN at Work: Example 1



ESN at Work: Example 2



Problem: units don't return to the initial state ($\rightarrow$ lack the Echo State Property).

Echo State Property (ESP)

Goal of ESP: make sure that the solution is not dependent on the initial conditions.

Conditions for ESP: For linear ESN, ESP is guaranteed when highest eigenvalue of the recurrent weights matrix (spectral radius) is smaller than 1. For nonlinear ESN, there is no formal guarantee, but the condition still works empirically.

Analogy to Kernels: ESP ensures the feature map is L2-integrable, and therefore enables the construction of a kernel:

$$k_{\mathrm{ESN}}(x, x') = \left\langle \phi_{\mathrm{ESN}}(x), \phi_{\mathrm{ESN}}(x') \right\rangle$$

Insight: An echo state network is a kernel (a smart one).

ESNs: How to Choose the Parameters?

Equations:

Set it large to create abrupt nonlinearities

Set it such that spectral radius is <1.

$$\tilde{\boldsymbol{h}}_t = \tanh\left(W^{\text{in}}\boldsymbol{x}_t + W^{\text{rec}}\boldsymbol{h}_{t-1}\right),$$
$$\boldsymbol{h}_t = (1-\alpha)\boldsymbol{h}_{t-1} + \alpha\tilde{\boldsymbol{h}}_t,$$
$$\hat{\boldsymbol{y}}_t = W^{\text{out}}\text{concat}(\boldsymbol{x}_t, \boldsymbol{h}_t),$$

where $W^{\text{rec}} \in \mathbb{R}^{M\times M}$ with M is the number of reservoir units, and $\alpha \in (0, 1]$ is the leaking rate.

As large as you can afford (typically M > 10000).

Set it close to zero if the time series evolves slowly.

Cf. tutorial paper: M. Lukosevicius: A Practical Guide to Applying Echo State Networks. Neural Networks: Tricks of the Trade (2nd ed.) 2012: 659-686

Advantages and Disadvantages of ESNs

Advantages:

Easy to train (convex)

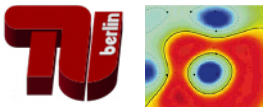
Work well for low-dimensional input

Embedded in the kernel framework

Disadvantages:

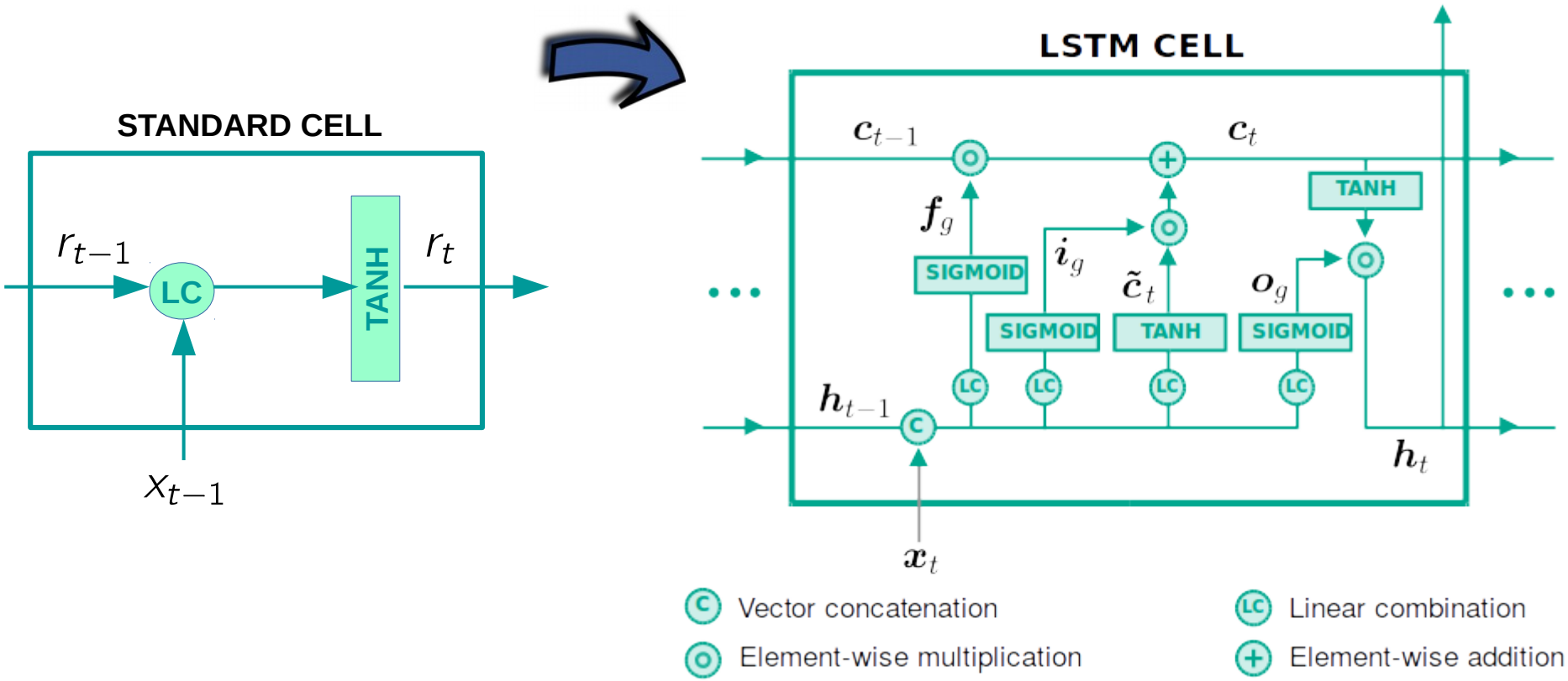
Can be tricky to design a reservoir with good properties

Require a big reservoir to solve complex problems



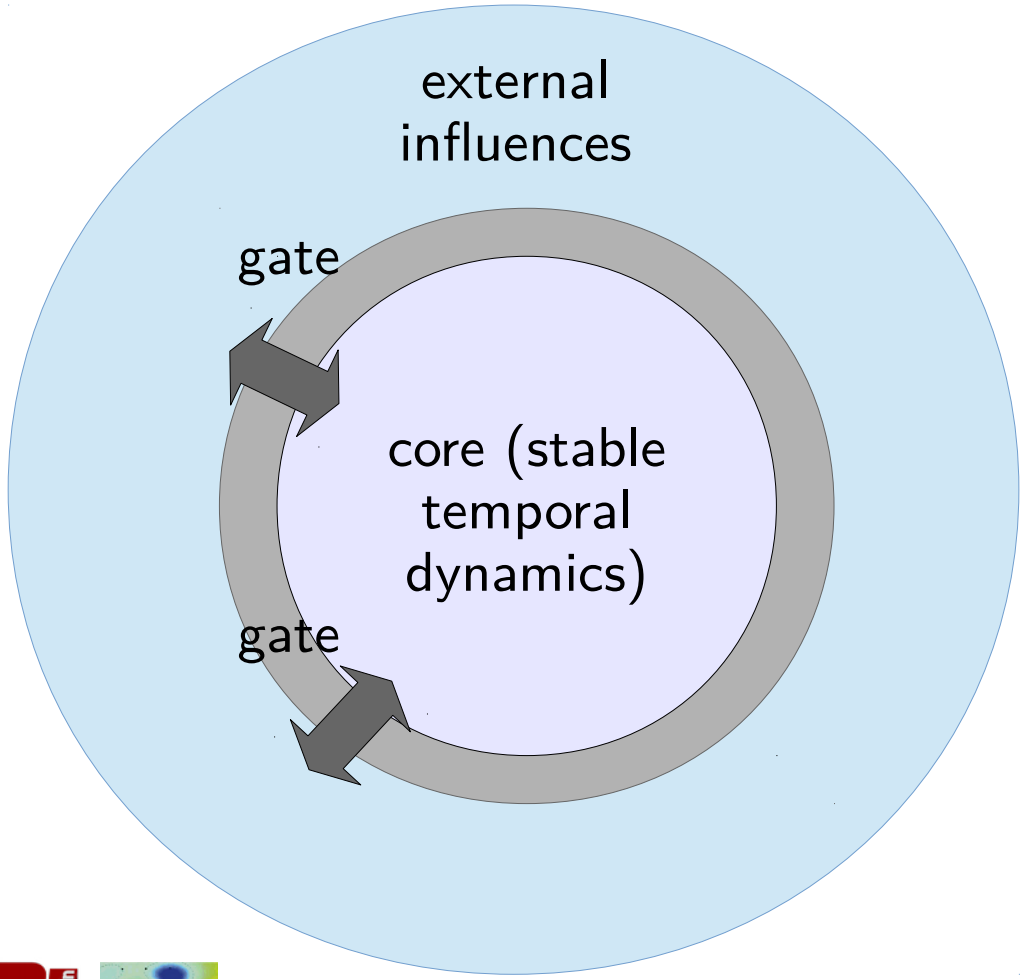
LSTMs [Hochreiter&Schmidhuber'97]

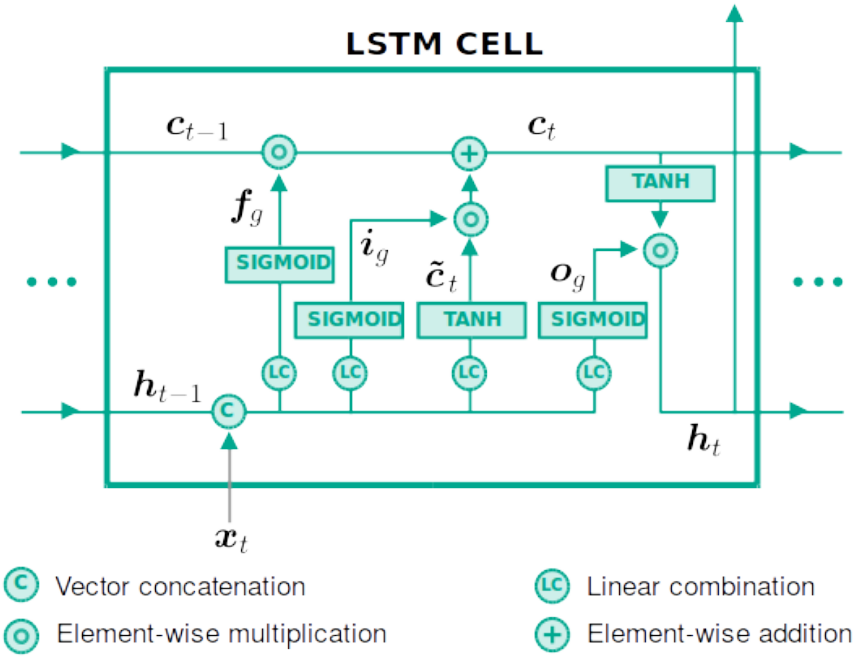
LSTM = RNN with long-short term memory



LSTMs

Key idea: protect the internal dynamics (core) from external influences (input and output feedback) through gates.





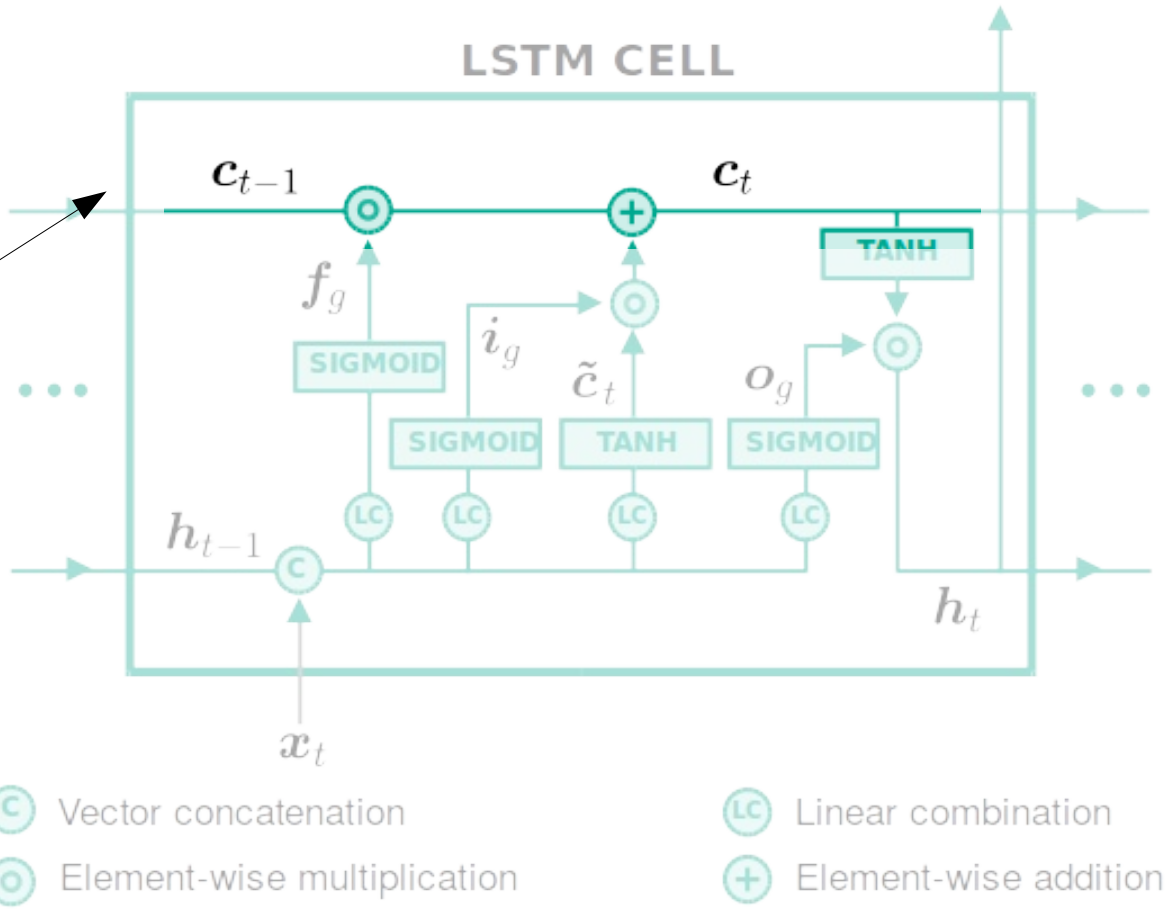
LSTMs

Key idea: protect the internal dynamics (core) from external influences (input and output feedback) through gates.

Internal dynamics

Stays stable over time (only small increments).

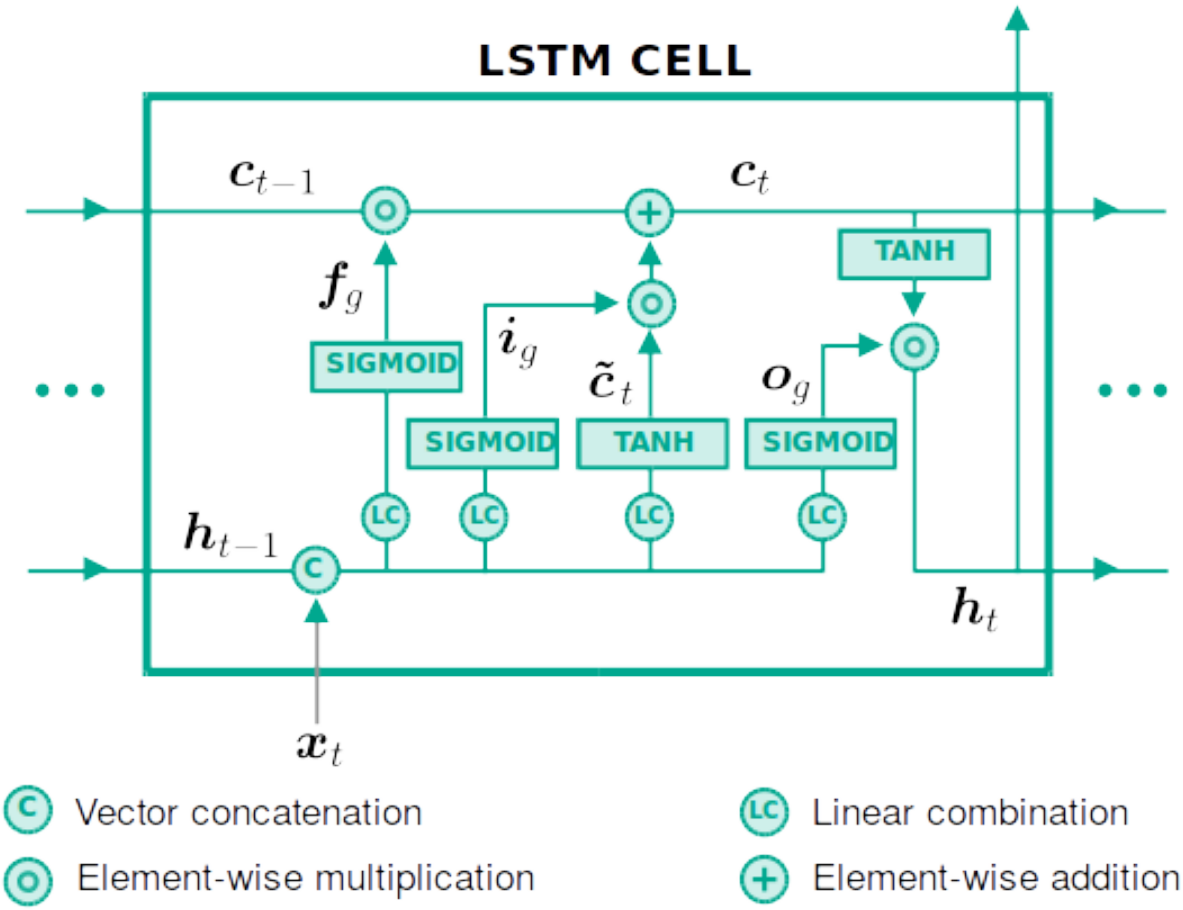
No weight matrices and squashing nonlinearities on that path.



LSTMs

Key idea: protect the internal dynamics (core) from external influences (input and output feedback) through gates.

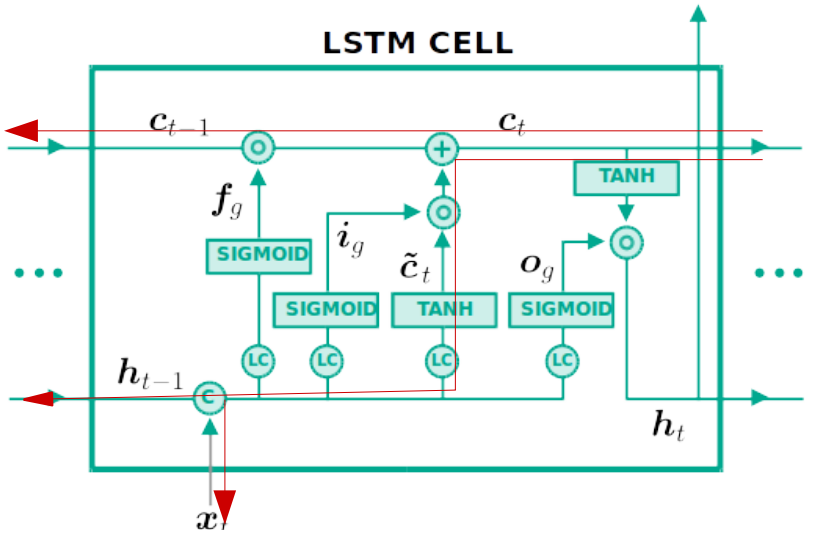
- Employ 3 gates:
  - Input gate $\boldsymbol{i}_g$
  - Forget gate $\boldsymbol{f}_g$
  - Output gate $\boldsymbol{o}_g$



Explaining LSTMs

LSTM predicts well, but can we explain its predictions?

Yes (cf. Arras'17, Rieger & Chormai'18), special rules for propagating backward in that cell.



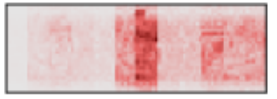
Explaining majority:

Explaining negative sentiment in movie reviews:

1. do n't waste your money .
2. neither funny nor suspenseful nor particularly well-drawn .
3. it 's not horrible , just horribly mediocre .
4. ... too slow , too boring , and occasionally annoying .
5. it 's neither as romantic nor as thrilling as it should be .
6. the master of disaster - it 's a piece of dreck disguised as cc

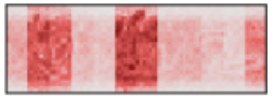
Shallow

Deep

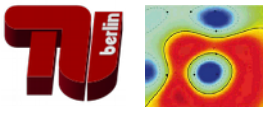


ConvDeep

R-LSTM



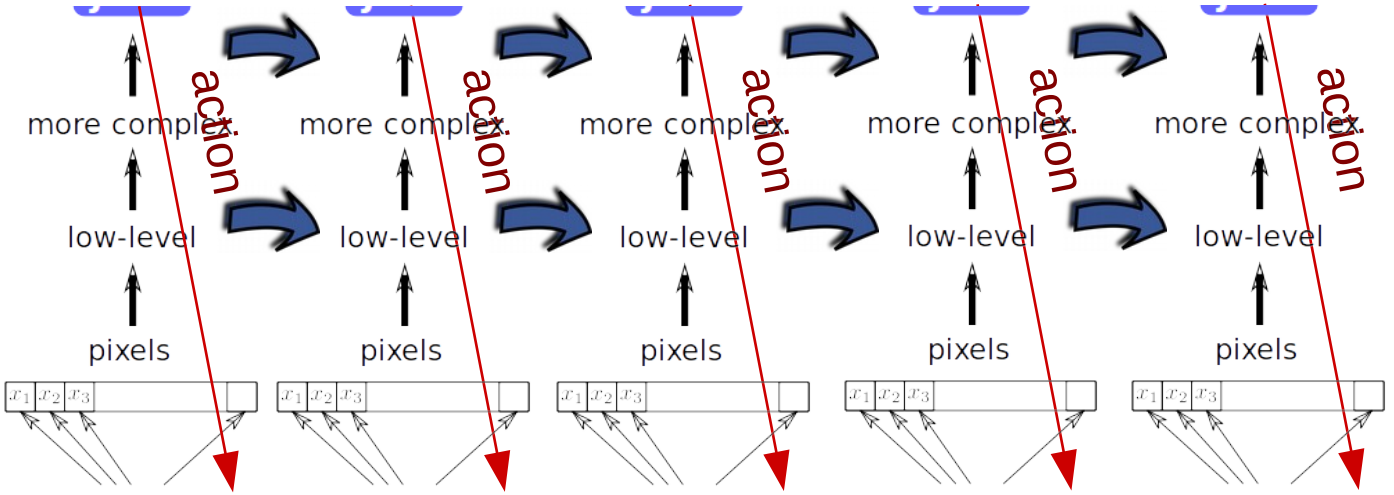
ConvR-LSTM



From Perception to Action

So far, we have built a state representation of the environment.

What about taking actions in an environment?



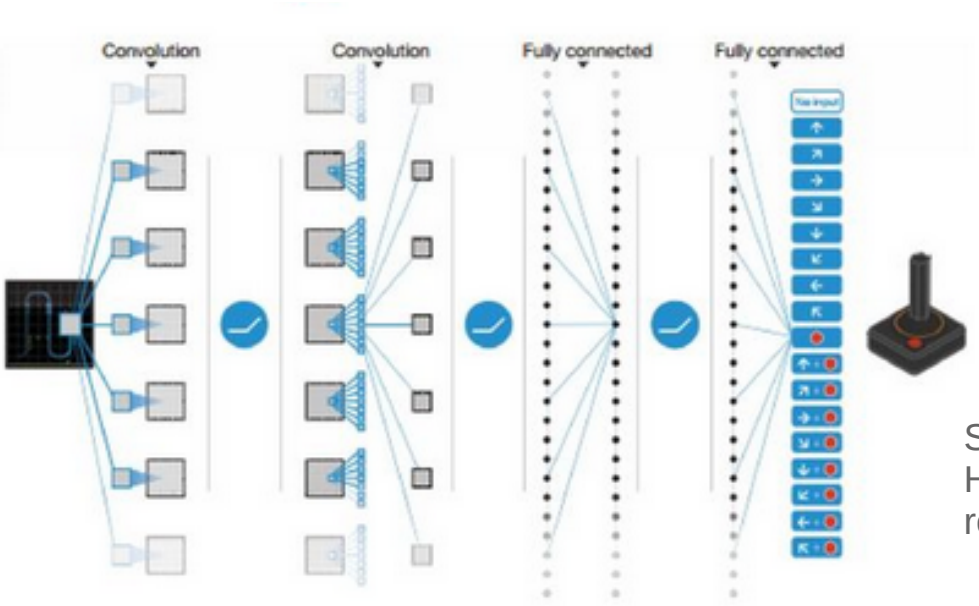
Neural Fitted Q Iteration

$$y_i = \mathbb{E}_{s' \sim \mathcal{E}} \left[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) | s, a \right]$$

better Q values

$$L_i(\theta_i) = \mathbb{E}_{s,a \sim \rho(\cdot)} \left[\left(y_i - Q(s, a; \theta_i) \right)^2 \right]$$

Deep Q Network



Source: Mnih'15
Human-level control through deep reinforcement learning

Summary

A large number of input data have a time component. Special models (e.g. recurrent neural networks) are needed for this.

Standard recurrent neural networks are hard to train (high curvature within and between layers, discontinuities in the error function).

Two approaches have been presented to make RNNs more applicable:

1. **Echo state networks** (train only the output layer)
2. **LSTMs** (adapt the structure to decouple the internal dynamics from the external influences)

RNNs is only one aspect. Reinforcement learning brings control capabilities. RNNs and RL work well in combination.