

Computational Intelligence

Winter Term 2025/26

Prof. Dr. Günter Rudolph

Computational Intelligence

Fakultät für Informatik

TU Dortmund

- Recurrent Neural Networks
 - Excursion: Nonlinear Dynamics
 - Recurrent Models
 - Training

S state space with states $s \in S$

$s^{(t)}$ is a state $\in S$ at time $t \in \mathbb{N}_0$

Θ parameter space with parameters $\theta \in \Theta$

$f : S \times \Theta \rightarrow S$ transition function

→ dynamical system $s^{(t+1)} = f(s^{(t)}, \theta)$ (*) **recurrence relation**

$$s^{(t)} = f^t(s^{(0)}, \theta) = \underbrace{f \circ \dots \circ f}_{t \text{ times}}(s^{(0)}, \theta) = \underbrace{f_\theta(f_\theta(\dots f_\theta(s^{(0)})))}_{t \text{ times}}; \quad f_\theta(s) = f(s, \theta)$$

D: s^* is called **stationary point / fixed point / steady state of (*)** if $s^* = f(s^*)$

D: stationary point s^* is **locally asymptotical stable (l.a.s.)** if

$$\exists \varepsilon > 0 : \forall s^{(0)} \in B_\varepsilon(s^*) : \lim_{t \rightarrow \infty} s^{(t)} = s^*$$

T: Let f be differentiable. Then s is l.a.s. if $|f'(s)| < 1$, and unstable if $|f'(s)| > 1$.

Remark: D: $s \in S$ is **recurrent** if $\forall \varepsilon > 0 : \exists t > 0 : f^t(s) \in B_\varepsilon(s)$ infinitely often (i.o.)

examples

- linear case: $f(x) = a x + b \quad a, b \in \mathbb{R}$

fixed points: $x = f(x) = a x + b \Rightarrow x = \frac{b}{1-a} \quad \text{if } a \neq 1$

stability: $f'(x) = a \Rightarrow |f'(x^*)| = |a| < 1$ l.a.s., $|a| > 1$ unstable
- nonlinear case: $f(x) = r x (1 - x) \quad r \in (0, 4] \quad x \in (0, 1) \quad \text{logistic map}$

fixed points: $x = f(x) = r x (1 - x) \Rightarrow x = 0 \text{ or } x = 1 - \frac{1}{r} = \frac{r-1}{r}$

stability: $f'(x) = r - 2r x$

$|f'(0)| = r < 1 \Rightarrow$ l.a.s. also for $r = 1$ since $x < f(x)$ for $x < \frac{1}{2}$

$|f'(\frac{r-1}{r})| = |2 - r| < 1 \Leftrightarrow 1 < r < 3$ l.a.s.

$r \in [3, 1 + \sqrt{6})$ oscillation between 2 values

$r \in [1 + \sqrt{6}, 3.54 \dots)$ oscillation between 4 values

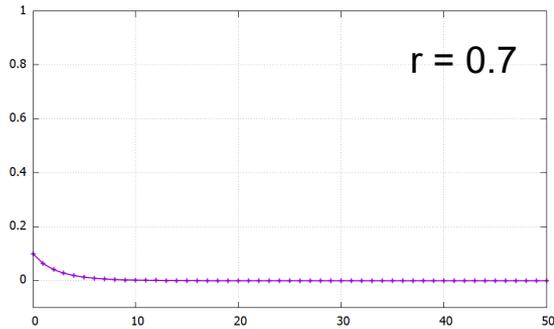
\vdots 8, 16, 32, ...

$r > 3.56995 \dots$ deterministic chaos

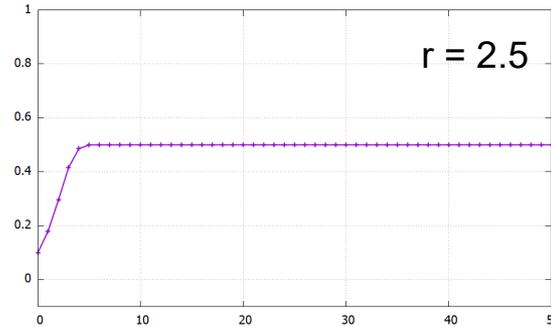
→ predicting a nonlinear dynamic system may be impossible!

logistic map

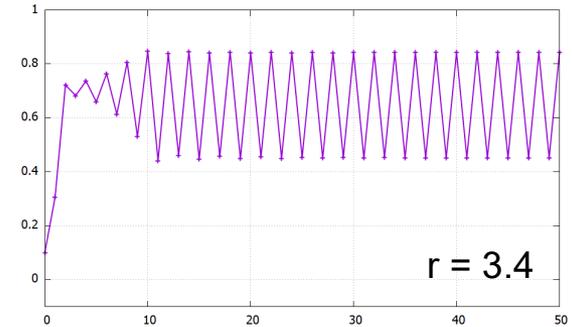
starting at $x = 0.1$



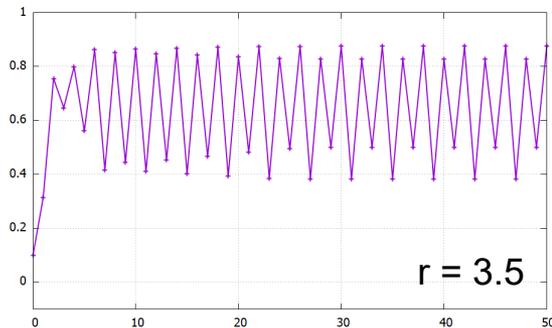
stable fixed point at $x = 0$



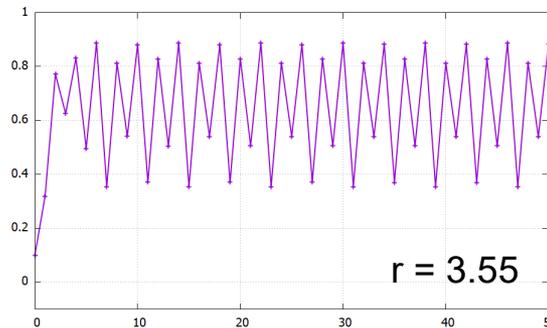
stable fixed point at $x = 0.5$



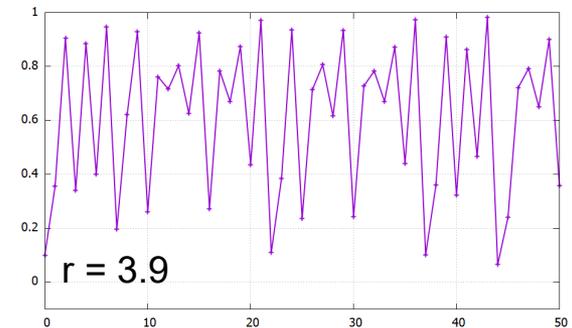
periodic orbit of size 2



periodic orbit of size 4



periodic orbit of size 8



deterministic chaos

extensions

- dynamical system with inputs

$$s^{(t)} = f(s^{(t-1)}, x^{(t)}; \theta)$$

↑
input at time $t \in \mathbb{N}$

- dynamical system with inputs and outputs

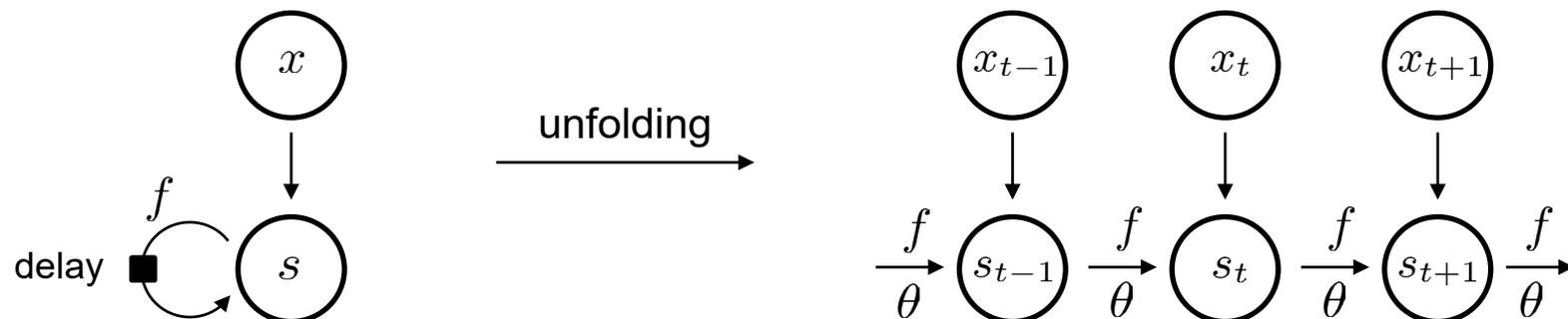
$$s^{(t)} = f(s^{(t-1)}, x^{(t)}; \theta_f)$$
$$o^{(t)} = g(s^{(t)}; \theta_g)$$

↑
output at time $t \in \mathbb{N}$

describes a
recurrent
neural network
(RNN)

unfolding

- finite input sequence
⇒ can unfold RNN completely to (deep) feed forward network
- infinite input sequence
⇒ can unfold RNN only finitely many steps into the past
⇒ assumption: behavior mainly depends on few inputs in the past (i.e., **no** long-term dependencies)



remark: parameters θ in unfolded network are shared otherwise with θ_t overfitting becomes very likely!

- Jordan network (1983)

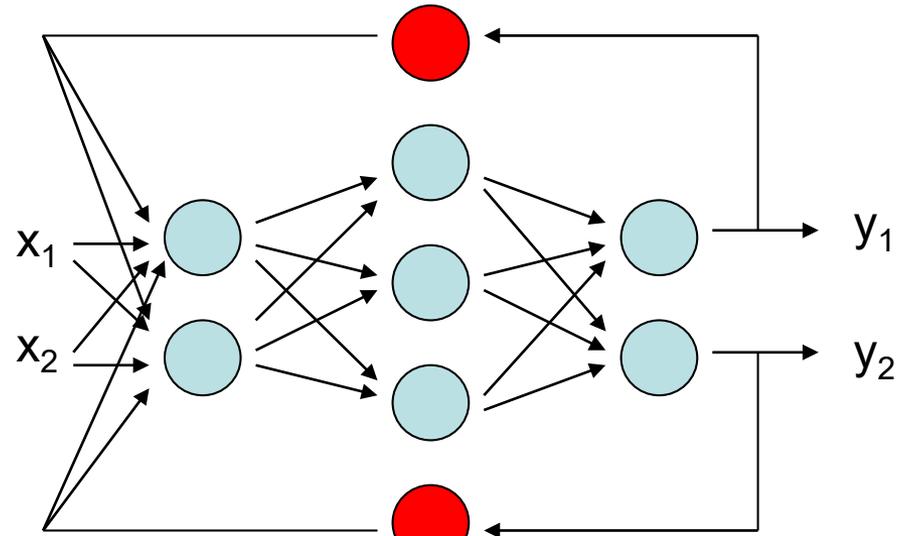
$$s_t = f(s_{t-1}, x_t; W, U, b)$$

$$= \sigma(Wx_t + U\hat{y}_{t-1} + b)$$

$$o_t = g(s_t; V, c)$$

$$= Vs_t + c$$

$$\hat{y}_t = a(o_t)$$

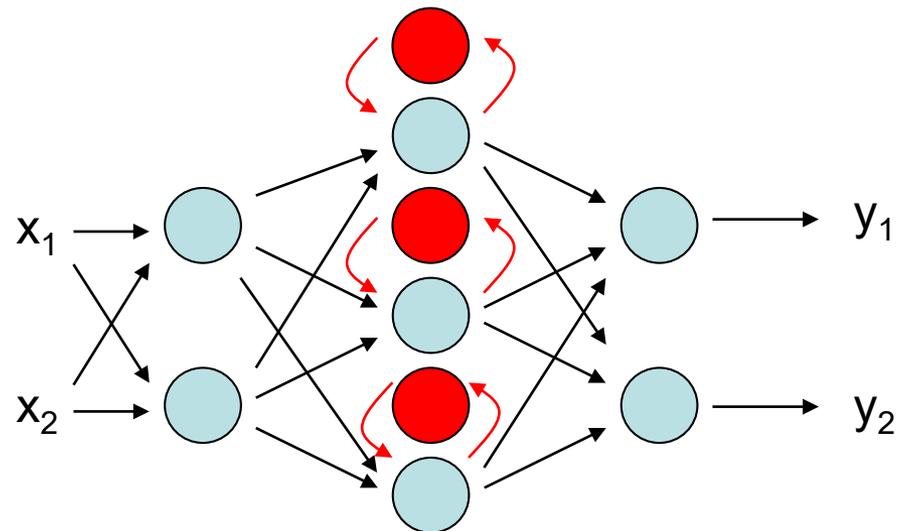


- Elman network (1990)

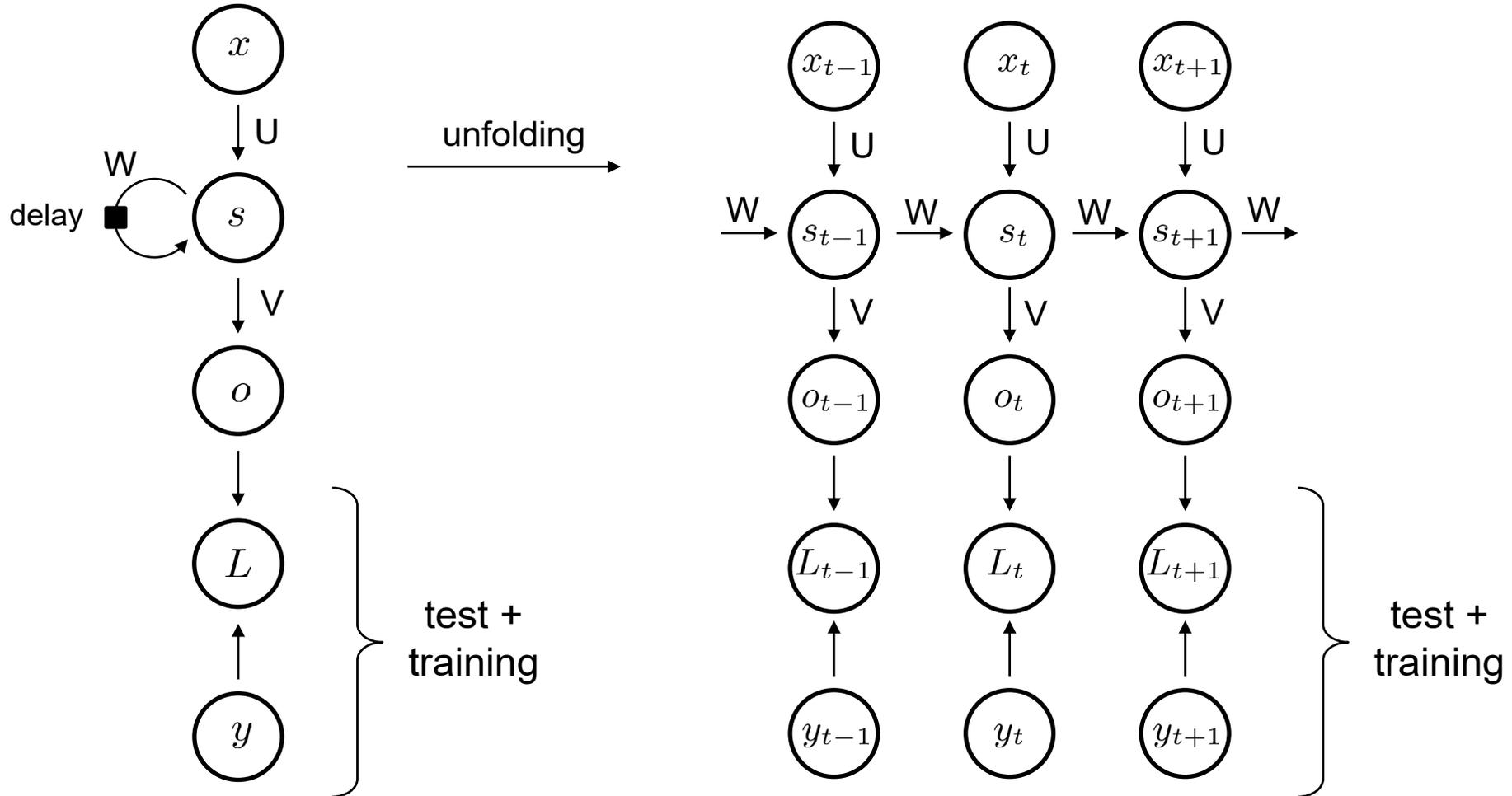
$$s_t = \sigma(Wx_t + Us_{t-1} + b)$$

$$o_t = Vs_t + c$$

$$\hat{y}_t = a(o_t)$$



test / training mode



loss per input $L(\hat{y}, y) = \|\hat{y} - y\|_2^2$ where $\hat{y} = \text{SOFTMAX}(o)$

training? → **backpropagation through time (BPTT)**

P.J. Werbos: Generalization of Backpropagation with Application to a Recurrent Gas Market Model.
Neural Networks 1(4):339-356, 1988.

- works on unfolded network for a finite input sequence $x^{(1)}, \dots, x^{(\tau)}$
- some adaption to BP necessary, since many parameters are shared
↑
reduces #params and overfitting
- “straightforward“ (but tedious + error-prone if done manually)
→ use method from your software library!
- in principle: gradient descent on loss function

LSTM network (1997f.)

LSTM = long short-term memory

so far: no long-term dependencies

now: “remember the important stuff and forget the rest“ [Cha18, p.89]

concept: two versions of the past

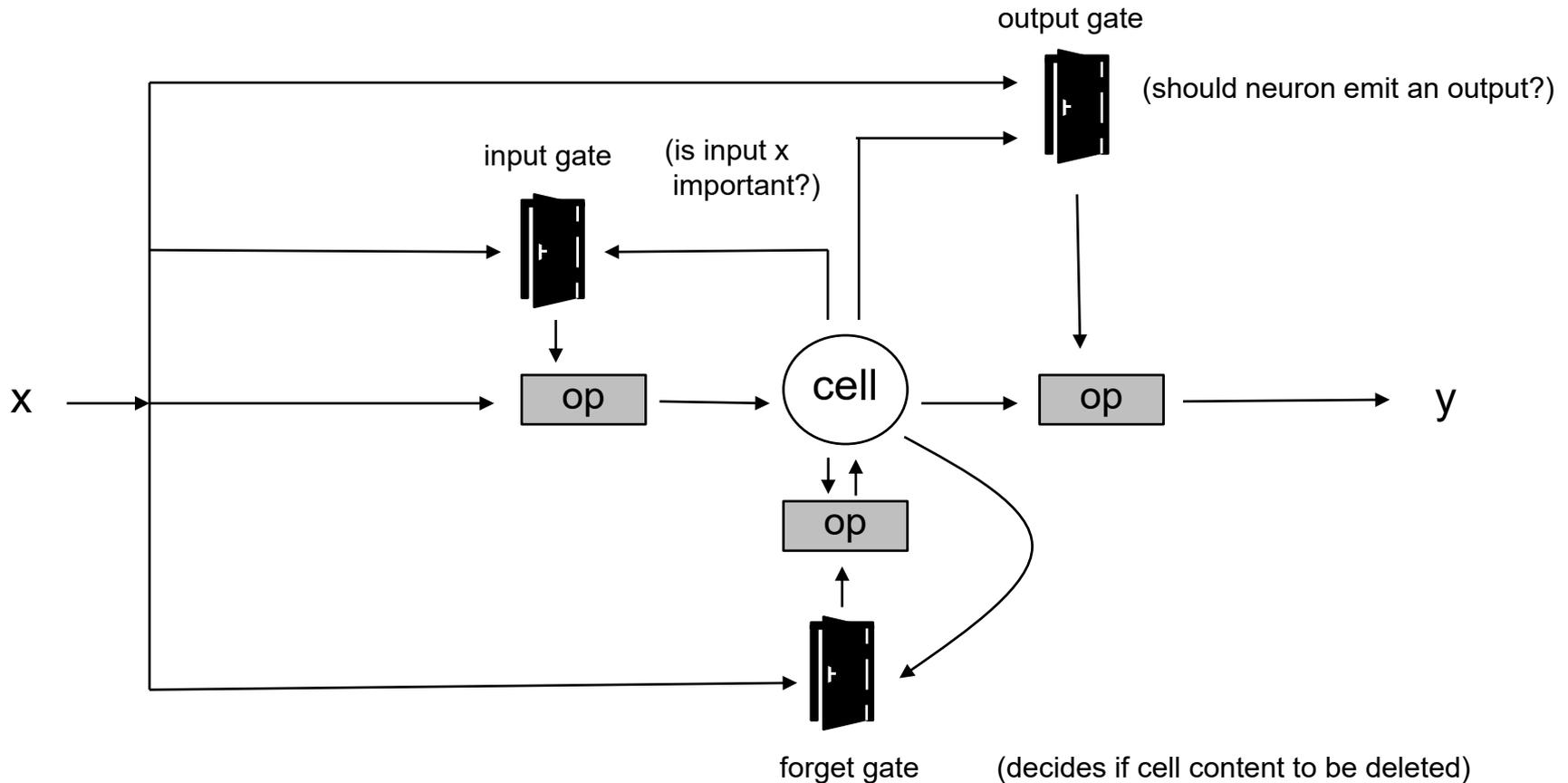
1. selective long-term memory
2. short term memory

historic/standard RNN
forget too quickly

- has the ability to learn long-term dependencies

LSTM Neuron [1997 f.]

LSTM = long short-term memory
cell content = memory



Gated Recurrent Unit (GRU) [2016]

“simplified“ LSTM neuron

- with input and forget gates
- with no output gate and context vector

- ⇒ leads to fewer parameters (compared to LSTM)
- ⇒ needs fewer training examples
- ⇒ possibly faster learning

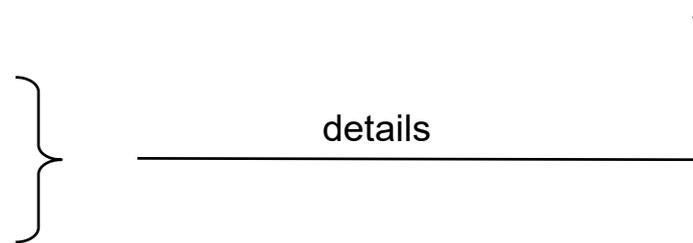
but: unclear if LSTM or GRU is better

Extended LSTM (xLSTM)

[2024]

<https://github.com/NX-AI/xlstm>

- based on LSTM
- different kind of gating
- matrix memory



→ initial performance results promising

<https://arxiv.org/abs/2405.04517>