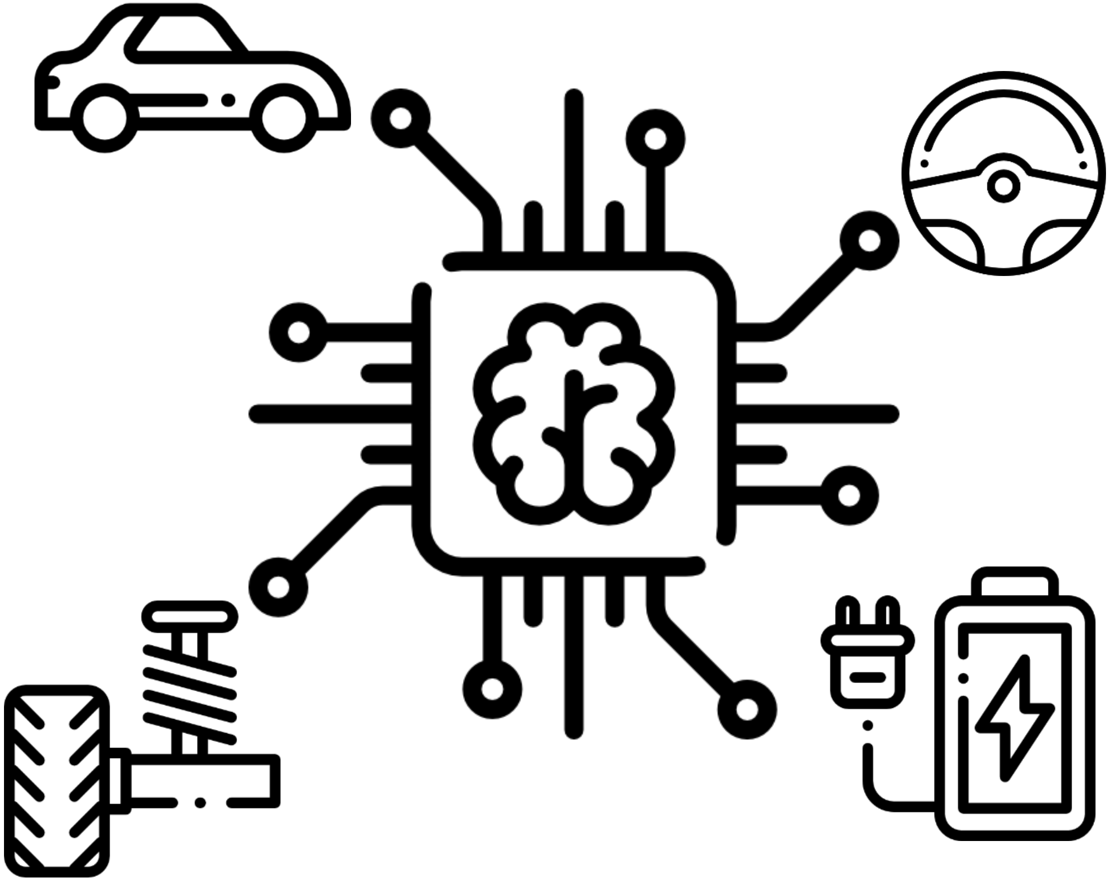
Lehrstuhl für Fahrzeugtechnik
Fakultät für Maschinenwesen
Technische Universität München

Artificial Intelligence in Automotive Technology

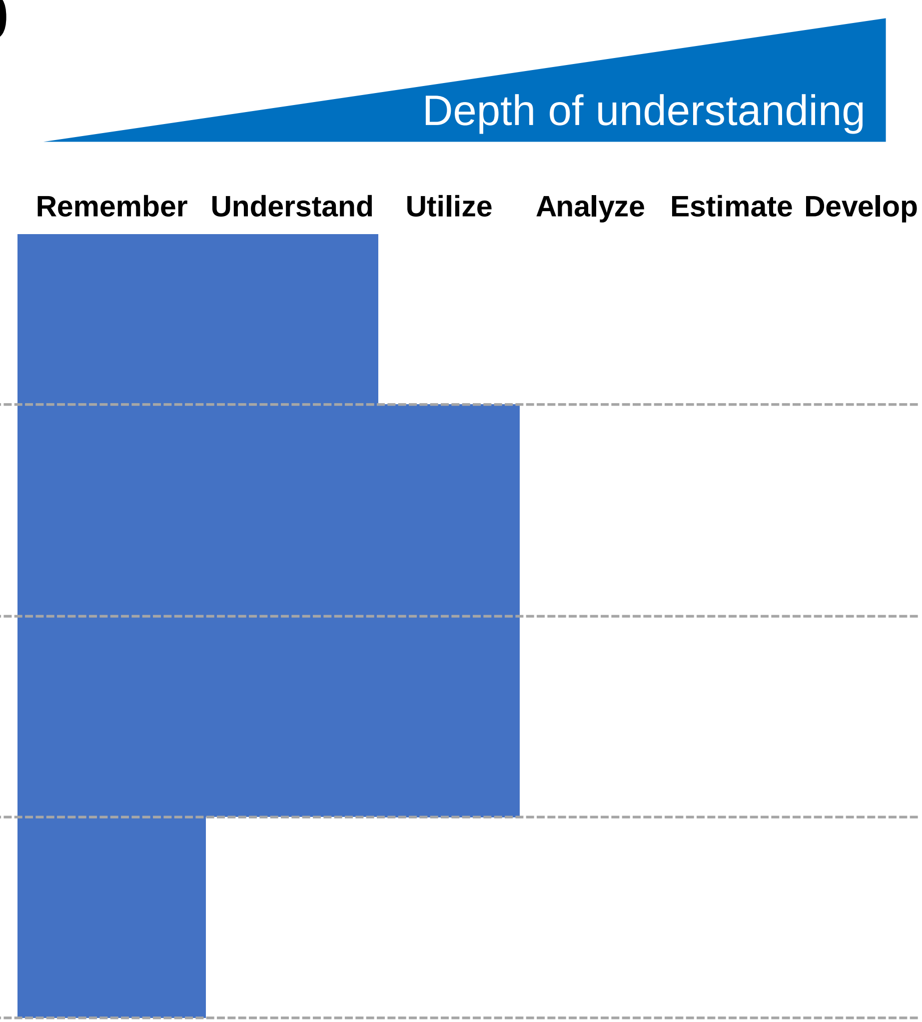
Johannes Betz / Prof. Dr.-Ing. Markus Lienkamp / Prof. Dr.-Ing. Boris Lohmann



Lecture Overview

Overall Introduction for the Lecture 18.10.2018 – Betz Johannes	**6 Pathfinding: From British Museum to A*** 29.11.2018 – Lennart Adenaw	**11 Reinforcement Learning** 17.01.2019 – Christian Dengler
1 Introduction: Artificial Intelligence 18.10.2018 – Betz Johannes	**P6:** 29.11.2018 – Lennart Adenaw	**P11** 17.01.2019 – Christian Dengler
P1: 18.10.2018 – Betz Johannes	**7 Introduction: Artificial Neural Networks** 06.12.2018 – Lennart Adenaw	**12 AI-Development** 24.01.2019 – Johannes Betz
2 Perception 25.10.2018 – Betz Johannes	**P7** 06.12.2018 – Lennart Adenaw	**P12** 24.01.2019 – Johannes Betz
P2: 25.10.2018 – Betz Johannes	**8 Deep Neural Networks** 13.12.2018 – Jean-Michael Georg	**13 Free Discussion** 31.01.2019 – Betz/Adenaw
3 Supervised Learning: Regression 08.11.2018 – Alexander Wischnewski	**P8** 13.12.2018 – Jean-Michael Georg	
P3: 08.11.2018 – Alexander Wischnewski	**9 Convolutional Neural Networks** 20.12.2018 – Jean-Michael Georg	
4 Supervised Learning: Classification 15.11.2018 – Jan-Cedric Mertens	**P9** 20.12.2018 – Jean-Michael Georg	
P4: 15.11.2018 – Jan-Cedric Mertens	**10 Recurrent Neural Networks** 10.01.2019 – Christian Dengler	
5 Unsupervised Learning: Clustering 22.11.2018 – Jan-Cedric Mertens	**P10** 10.01.2019 – Christian Dengler	

Objectives of the lecture 10



1. Make clear when to use Recurrent Neural Networks (RNN) instead of static Neural Networks.

2. Show the basic methods to train a RNN.

3. Clarify problems that might occur when training a RNN over long sequences, and how they can be avoided/reduced.

4. Give a short overview of different architectures for RNNs with use cases.

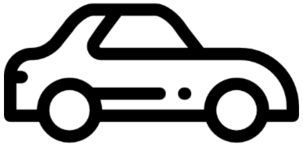
Recurrent Neural Networks

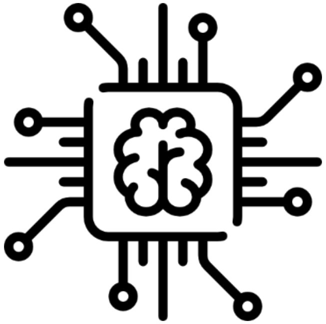
Johannes Betz / Prof. Dr. Markus Lienkamp / Prof. Dr. Boris Lohmann

(Christian Dengler, M. Sc.)

Agenda

1. Sequential Data and Use Cases
2. Simple RNN and Backpropagation through Time
3. Challenge of long Term Dependencies
4. Advanced RNN Structures
5. Recurrent Neural Networks for Automobiles



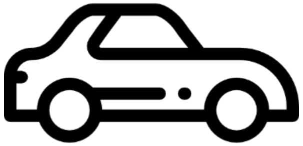


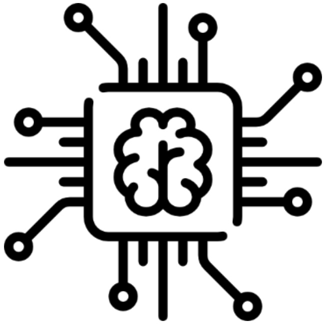
Recurrent Neural Networks

Johannes Betz / Prof. Dr. Markus Lienkamp / Prof. Dr. Boris Lohmann
(Christian Dengler, M. Sc.)

Agenda

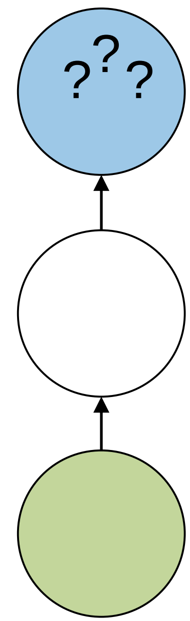
1. **Sequential Data and Use Cases**
2. Simple RNN and Backpropagation through Time
3. Challenge of long Term Dependencies
4. Advanced RNN Structures
5. Recurrent Neural Networks for Automobiles





1. Sequential Data and Use Cases

Sequential Data



Output layer
→Sigmoid

Hidden layers
→Conv.

Input layer
→Pixel Tensor

Will he score?

1. Sequential Data and Use Cases

Sequential Data

Will he score?

1. Sequential Data and Use Cases

Sequential Data

Will he score?

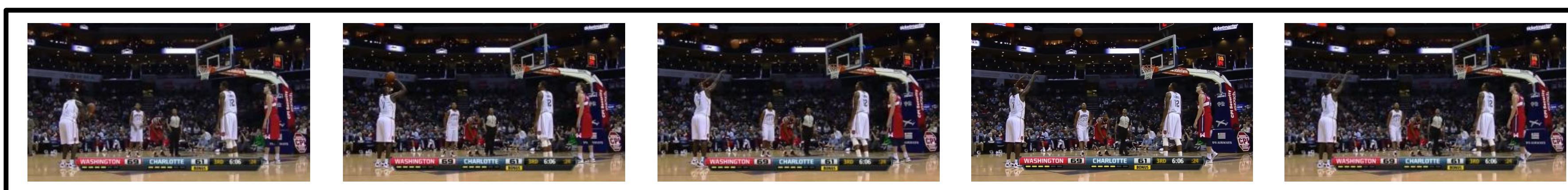
1. Sequential Data and Use Cases

Sequential Data

Can I pass, should I break?

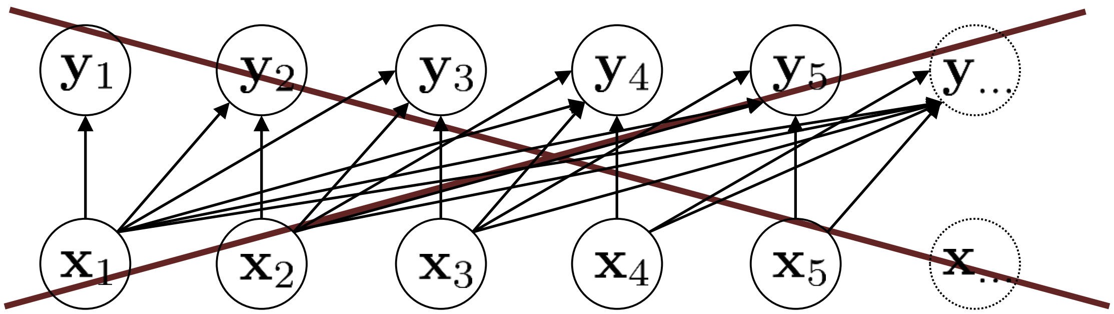
1. Sequential Data and Use Cases

Sequential Data

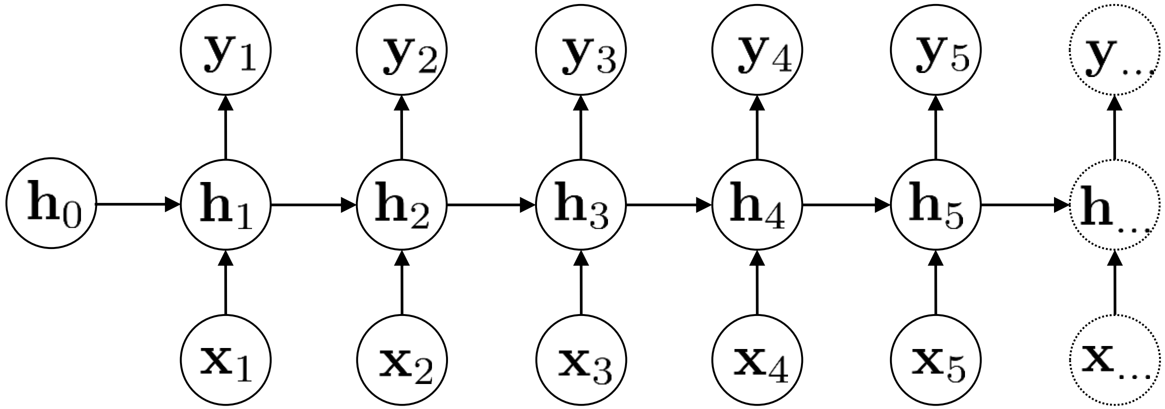


1. Sequential Data and Use Cases

Fully connected model:



Single time-step update:



1. Sequential Data and Use Cases

Use cases

- Speech recognition and generation
- Music recognition and generation
- Translation
- Image Capturing
- Video Capturing
- **Modeling dynamics of physical systems**
- …

1. Sequential Data and Use Cases

Image Classification and Generation [12]

1. Sequential Data and Use Cases

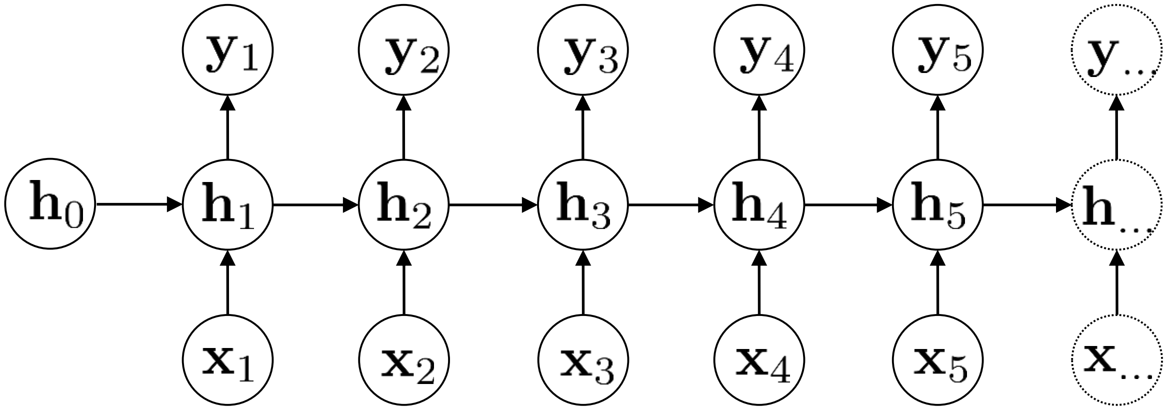
Image Classification and Generation [12]

1. Sequential Data and Use Cases

Wrap up

- Often one observation does not contain the required information.
- The information is often hidden in sequences of data, but just taking a whole sequence as input will require too many parameters

→Share parameters and add a memory to capture the important features of the past.

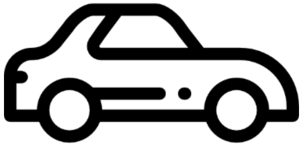


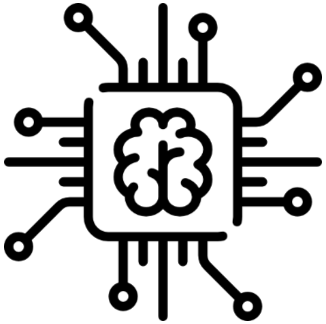
Recurrent Neural Networks

Johannes Betz / Prof. Dr. Markus Lienkamp / Prof. Dr. Boris Lohmann
(Christian Dengler, M. Sc.)

Agenda

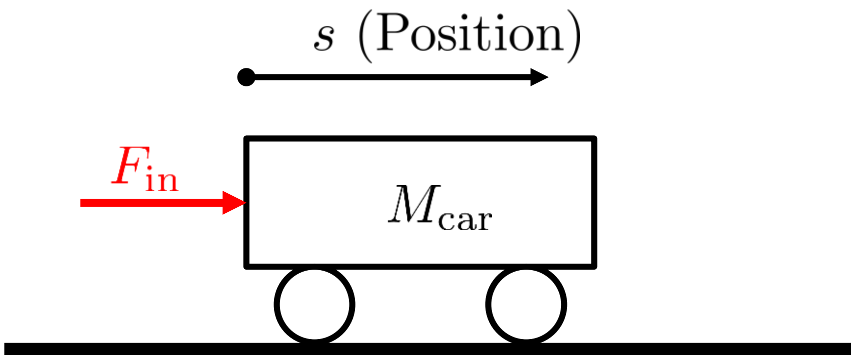
1. Sequential Data and Use Cases
2. **Simple RNN and Backpropagation through Time**
3. Challenge of long Term Dependencies
4. Advanced RNN Structures
5. Recurrent Neural Networks for Automobiles





2. Simple RNN and backpropagation through time

Dynamical Systems in Engineering



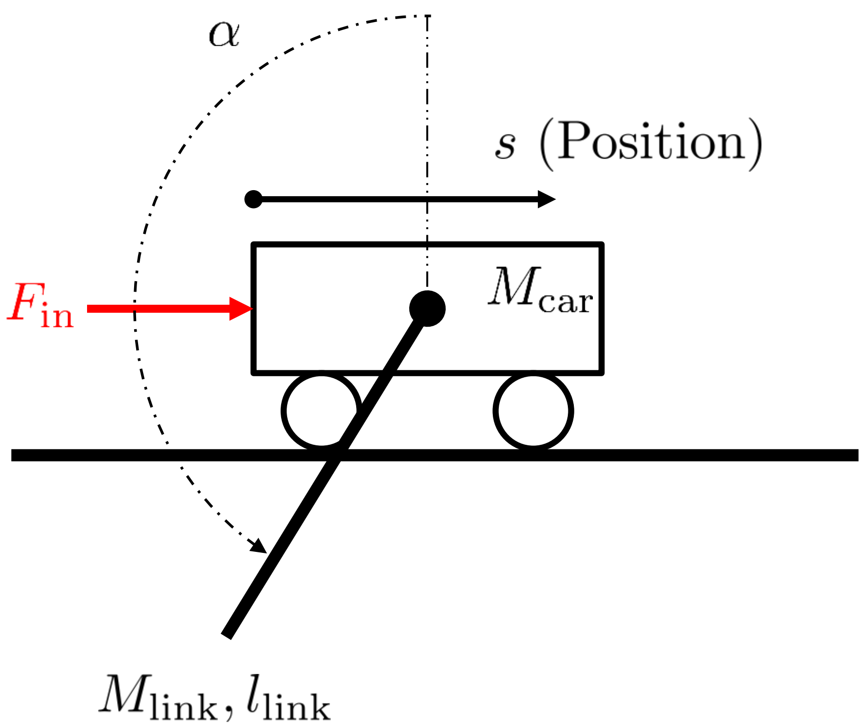
Input $u = F_{\text{in}}$,
output $y = s$.

Model equations

linear state-space representation

2. Simple RNN and backpropagation through time

Dynamical Systems in Engineering



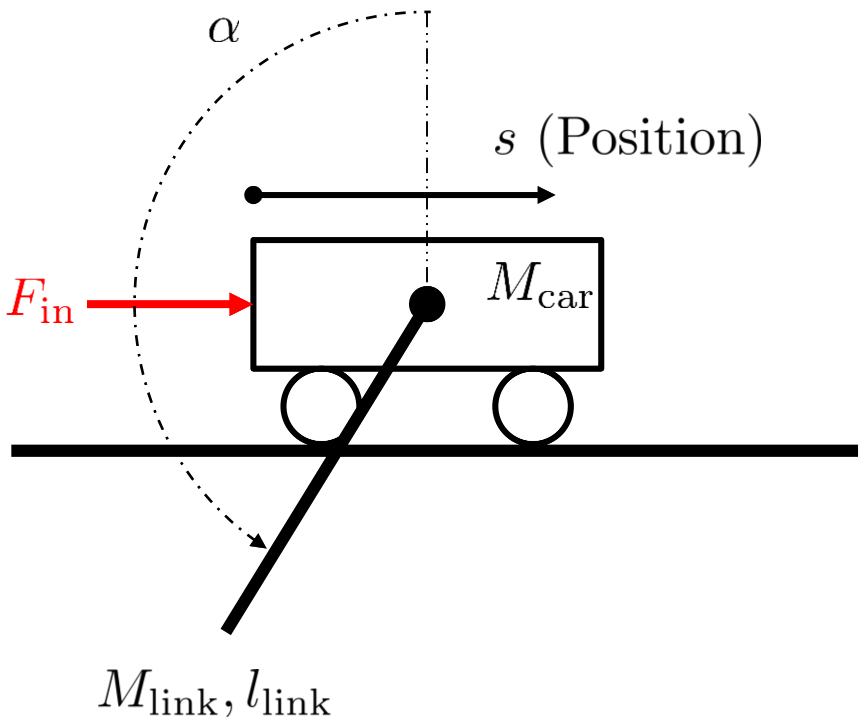
Model equations [16]:

$$(M_{\text{cart}} + M_{\text{link}})\ddot{s} - M_{\text{link}} l_{\text{link}} \ddot{\alpha} \sin(\alpha) = F$$

$$l_{\text{link}} \ddot{\alpha} - g \sin(\alpha) = \ddot{s} \cos(\alpha)$$

2. Simple RNN and backpropagation through time

Dynamical Systems in Engineering



Nonlinear state-space model:

$$\begin{bmatrix} \dot{s} \\ \ddot{s} \\ \dot{\alpha} \\ \ddot{\alpha} \end{bmatrix} = \begin{bmatrix} \dot{s} \\ f_1(\dot{s}, \alpha, \dot{\alpha}, F) \\ \dot{\alpha} \\ f_2(\dot{s}, \alpha, \dot{\alpha}, F) \end{bmatrix}$$

$$y = s$$

2. Simple RNN and backpropagation through time

Dynamical Systems in Engineering

Continuous-time system:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$$
$$\mathbf{y} = \mathbf{g}(\mathbf{x})$$

Forward Euler

Runge-Kutta 4

…

Discrete-time system:

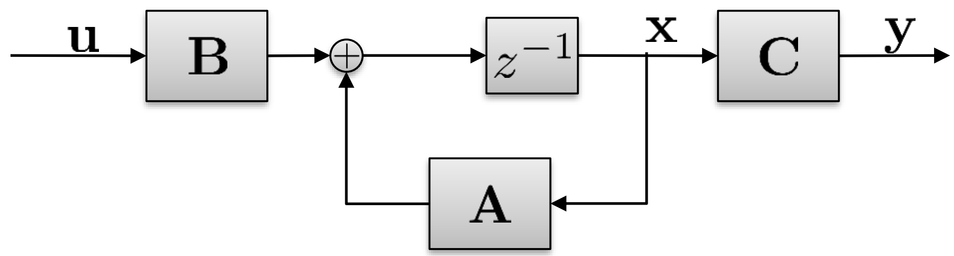
$$\mathbf{x}_{t+1} = \tilde{\mathbf{f}}(\mathbf{x}_t, \mathbf{u}_t)$$
$$\mathbf{y}_t = \mathbf{g}(\mathbf{x}_t)$$

2. Simple RNN and backpropagation through time

Dynamical Systems in Engineering

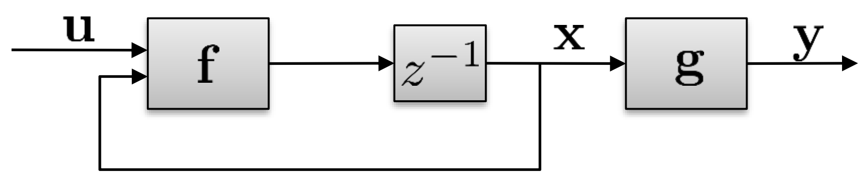
Linear Case

$$\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{B}\mathbf{u}_t$$
$$\mathbf{y}_t = \mathbf{C}\mathbf{x}_t$$



Nonlinear Case

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t)$$
$$\mathbf{y}_t = \mathbf{g}(\mathbf{x}_t)$$

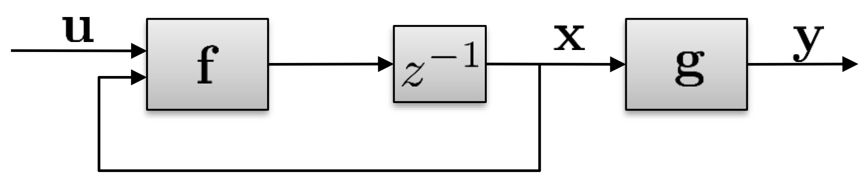


2. Simple RNN and backpropagation through time

Notation

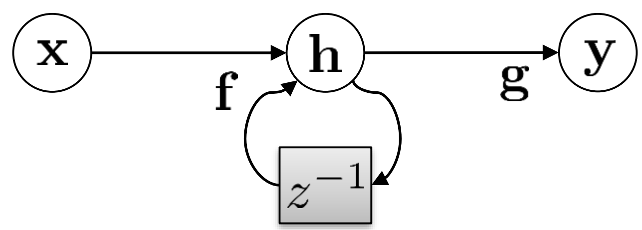
Engineering, e.g. Control Theory

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t)$$
$$\mathbf{y}_t = \mathbf{g}(\mathbf{x}_t)$$



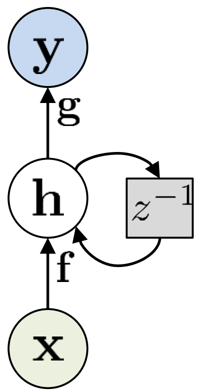
Machine Learning

$$\mathbf{h}_t = \mathbf{f}(\mathbf{h}_{t-1}, \mathbf{x}_t)$$
$$\mathbf{y}_t = \mathbf{g}(\mathbf{h}_t)$$



2. Simple RNN and backpropagation through time

Notation



Symbol	Meaning
$\mathbf{x}$	Input
$\mathbf{h}$	Hidden state
$\mathbf{y}$	Output
$\hat{\mathbf{y}}$	Observation/Data

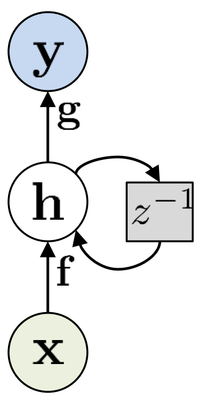
2. Simple RNN and backpropagation through time

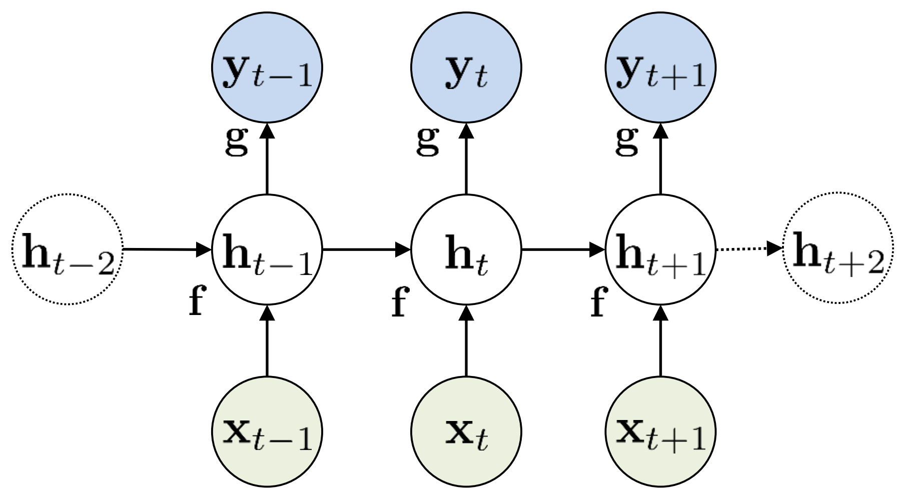
Unfolding the graph

Dynamical system:

$$\mathbf{h}_t = \mathbf{f}(\mathbf{h}_{t-1}, \mathbf{x}_t)$$

$$\mathbf{y}_t = \mathbf{g}(\mathbf{h}_t)$$





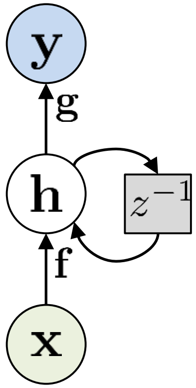
2. Simple RNN and backpropagation through time

Unfolding the graph

Dynamical system:

$$\mathbf{h}_t = \mathbf{f}(\mathbf{h}_{t-1}, \mathbf{x}_t)$$
$$\mathbf{y}_t = \mathbf{g}(\mathbf{h}_t)$$

Evalute/Simulate RNN (Inference) :

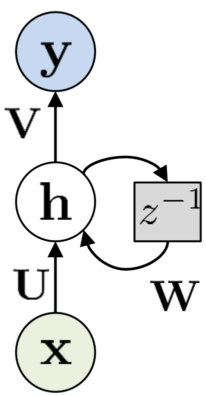


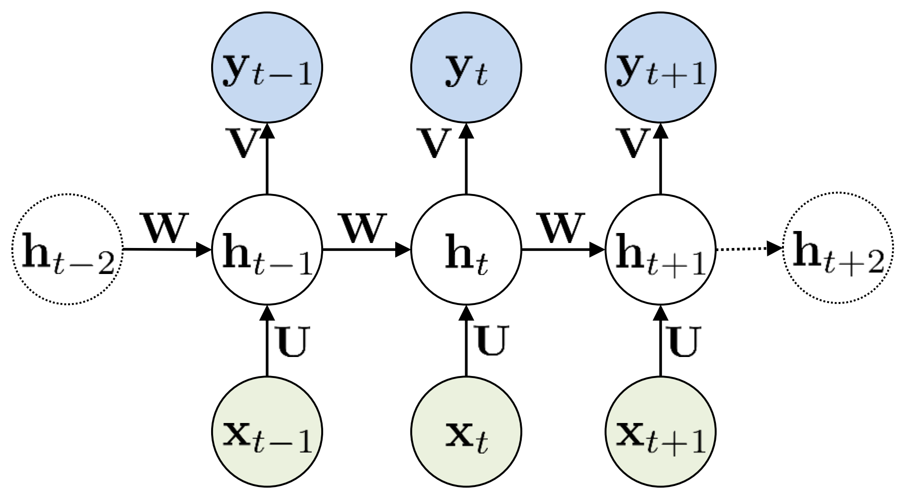
2. Simple RNN and backpropagation through time

Unfolding the graph

Dynamical system:

$$\mathbf{h}_t = \tanh(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}\mathbf{x}_t + \mathbf{b})$$
$$\mathbf{y}_t = \mathbf{V}\mathbf{h}_t + \mathbf{v}$$

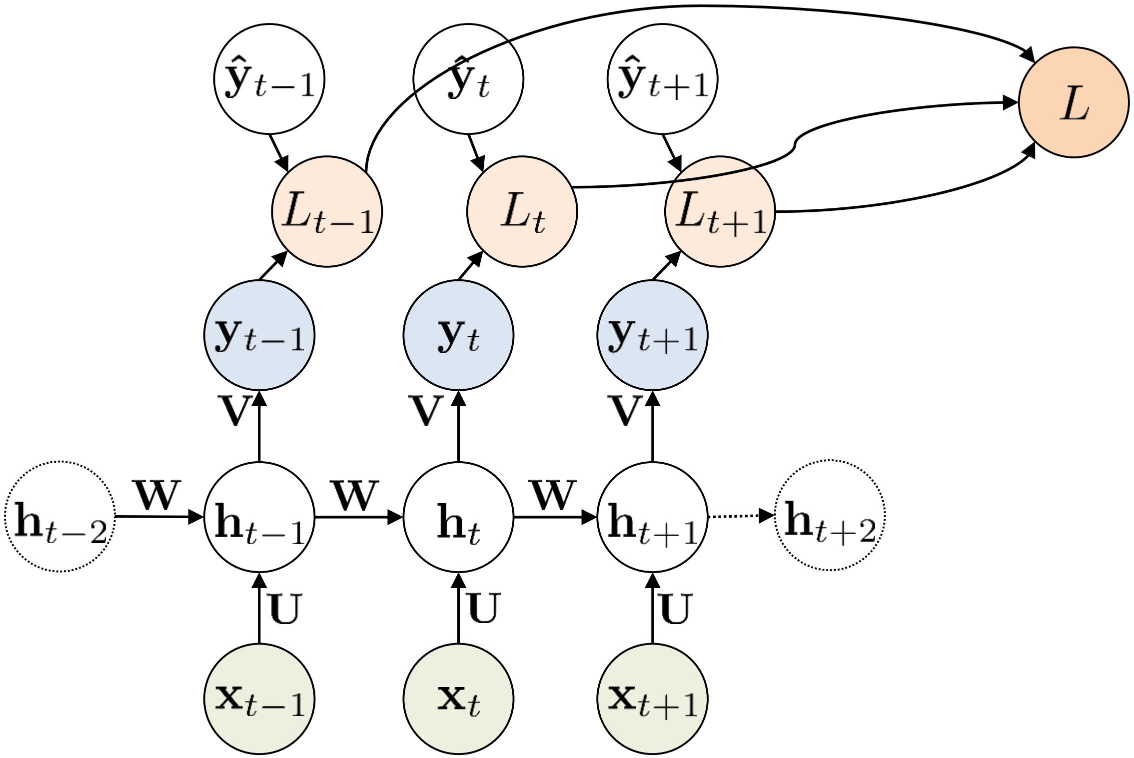




2. Simple RNN and backpropagation through time

Optimization setting

Loss function as a sum over timesteps: $L = \sum_t L_t = \sum_t L(\mathbf{y}_t, \hat{\mathbf{y}}_t)$



2. Simple RNN and backpropagation through time

Loss Function

As for static Neural Networks: minimize the cross-entropy between the generating distribution and our model P.

$$L_t = -\log P\left(o_t = y_t | x_t, x_{t-1}, \ldots, x_1, h_0\right)$$

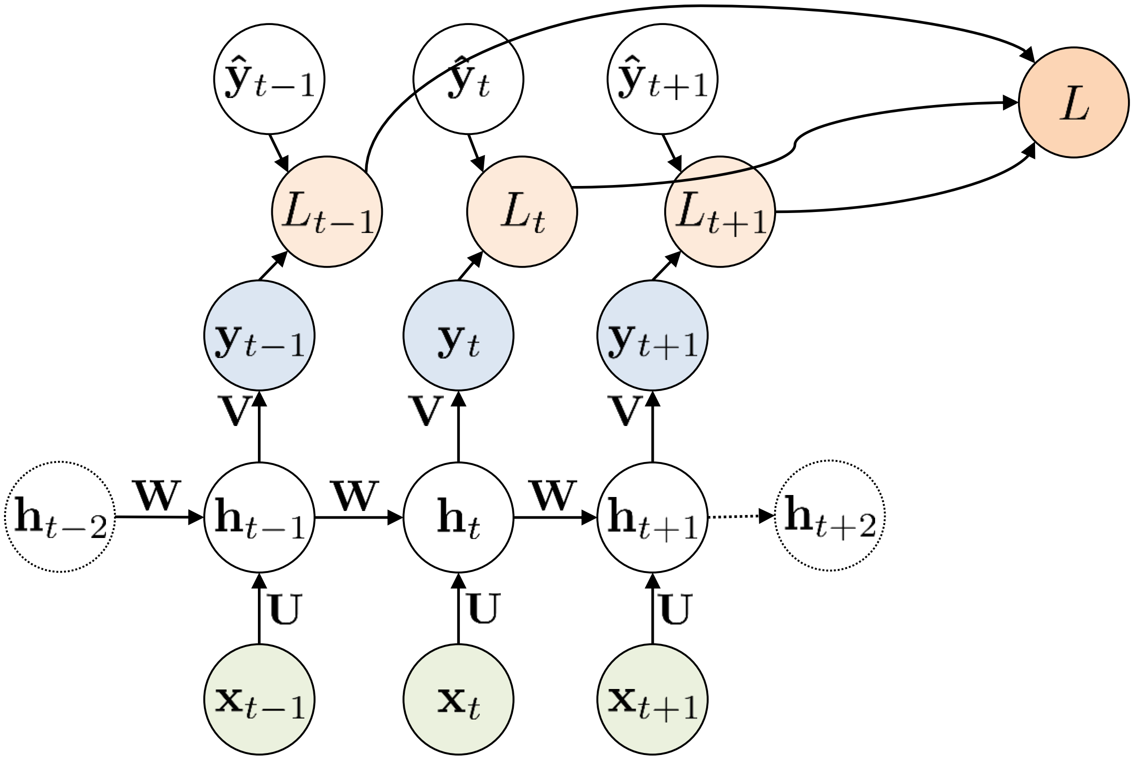
E.g. for a normal distribution $\mathcal{N}\left(\mu(x_t, x_{t-1}, \ldots, x_1, h_0), 1^2\right)$ we restore the quadratic loss function

$$L_t = \frac{1}{2}\left(y_t - \mu(x_t, \ldots)\right)^2$$

2. Simple RNN and backpropagation through time

Backpropagation through time (BPTT)

Backpropagation, applied to the unfolded graph of a sequence.

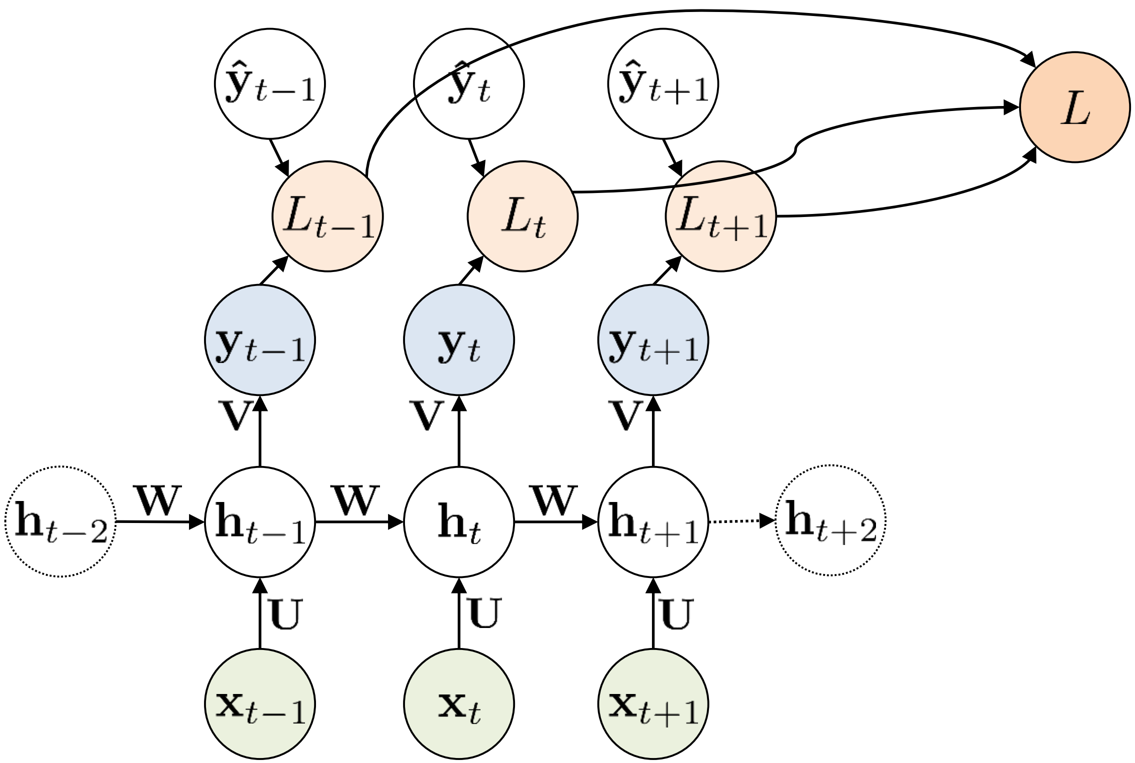


2. Simple RNN and backpropagation through time

Backpropagation through time (BPTT)

Backpropagation, applied to the unfolded graph of the sequence.

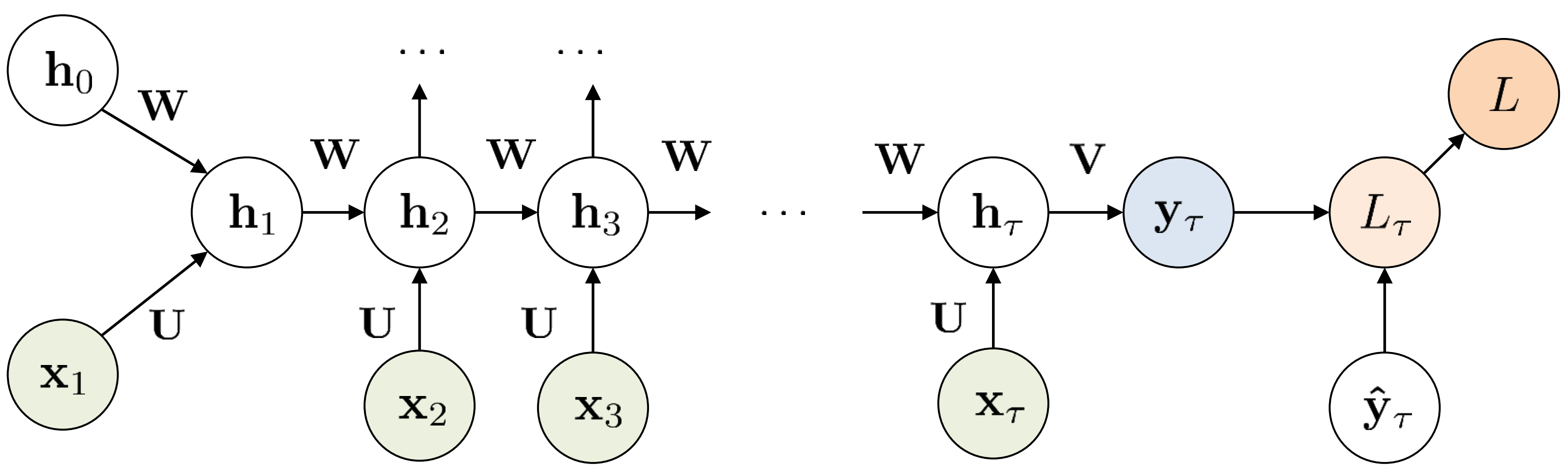
$$\frac{\partial}{\partial W^T} L = \frac{\partial L}{\partial h^{\tau,T}} \frac{\partial h^{\tau}}{\partial W^T} + \frac{\partial L}{\partial h^{\tau-1,T}} \frac{\partial h^{\tau-1}}{\partial W^T} + \frac{\partial L}{\partial h^{\tau-2,T}} \frac{\partial h^{\tau-2}}{\partial W^T} + \ldots$$



2. Simple RNN and backpropagation through time

Backpropagation through time (BPTT)

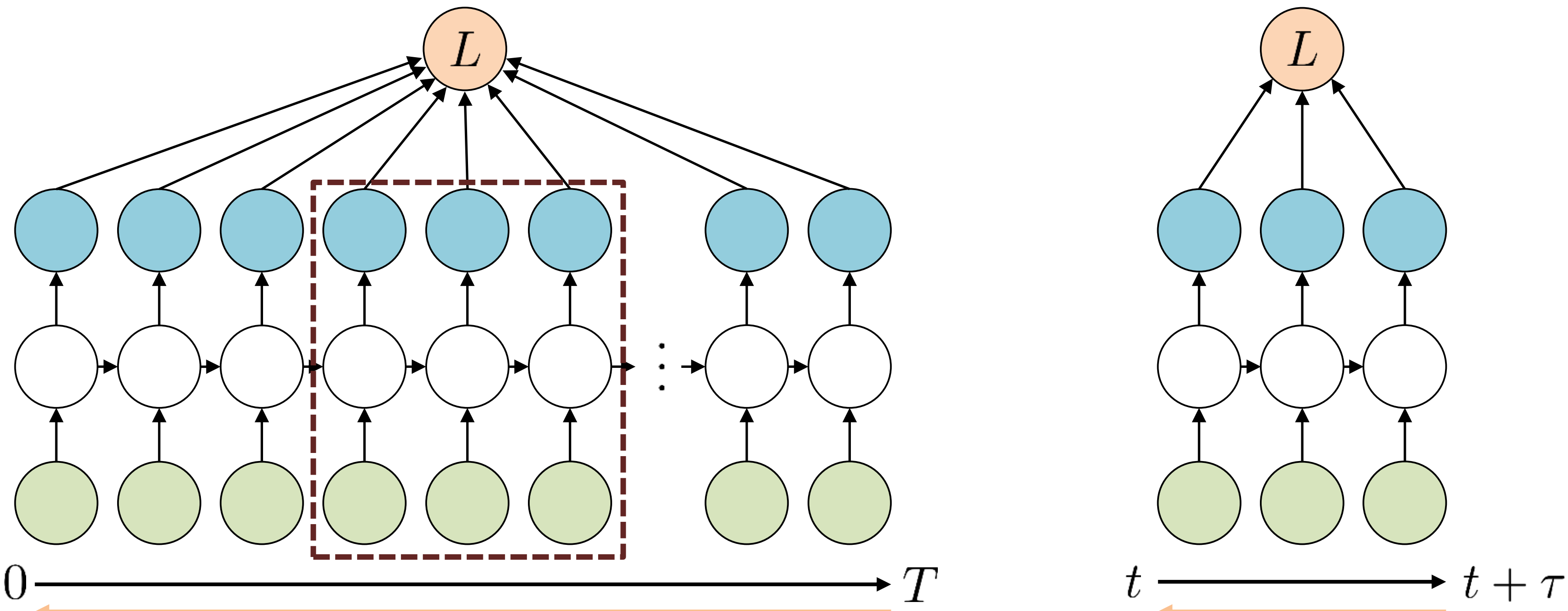
Similar to a deep neural network, with the number of layers increasing with the number of timesteps (deep computation graph).



2. Simple RNN and backpropagation through time

Truncated Backpropagation through time

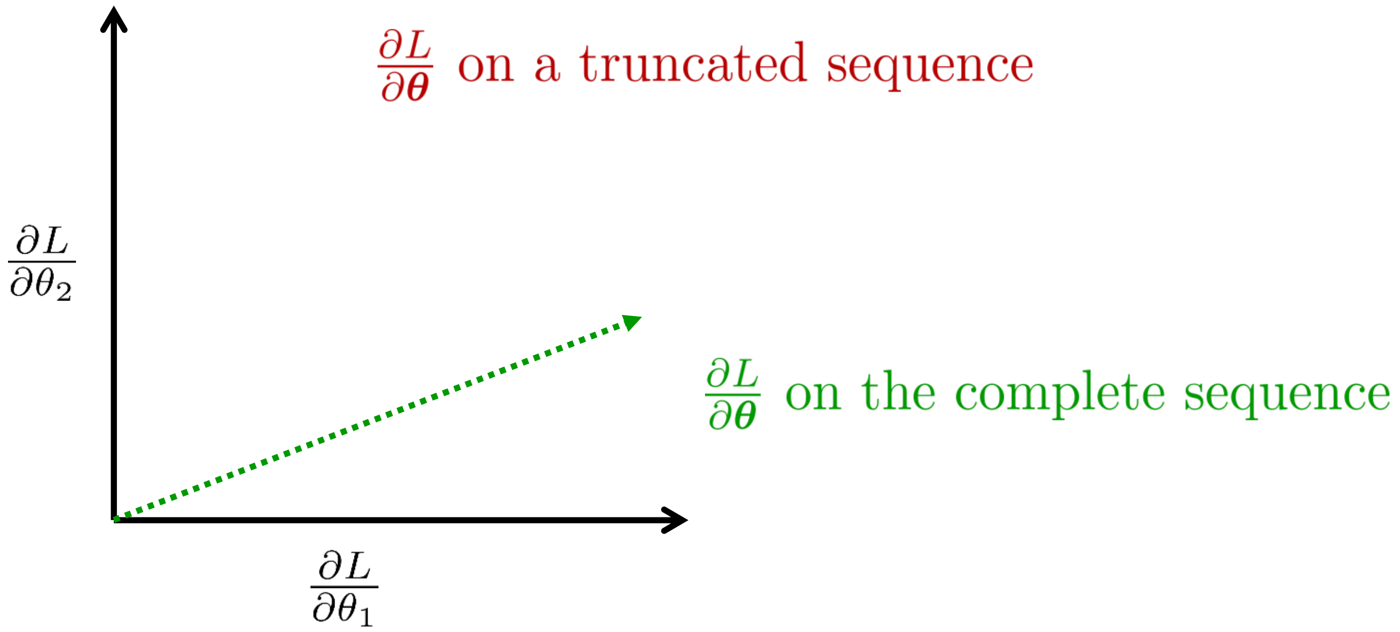
Backpropagation, applied to the unfolded graph of a chunk of the sequence of length τ. Similar to minibatches in supervised learning.



2. Simple RNN and backpropagation through time

Truncated Backpropagation through time

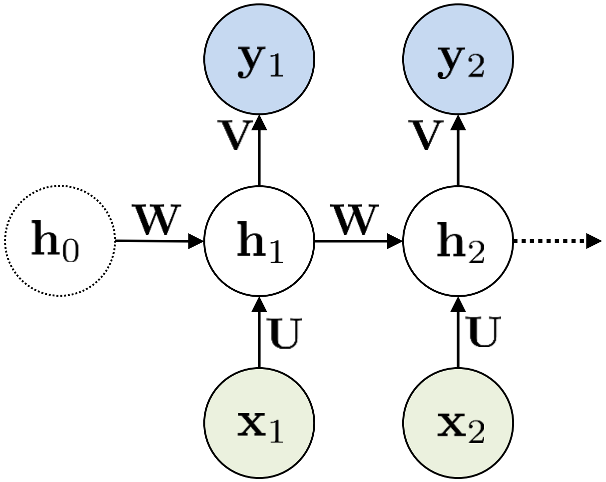
Truncated Backpropagation through time is biased! Unbiased versions e.g. in [1].



2. Simple RNN and backpropagation through time

State initialization

How can we chose h_0 ?

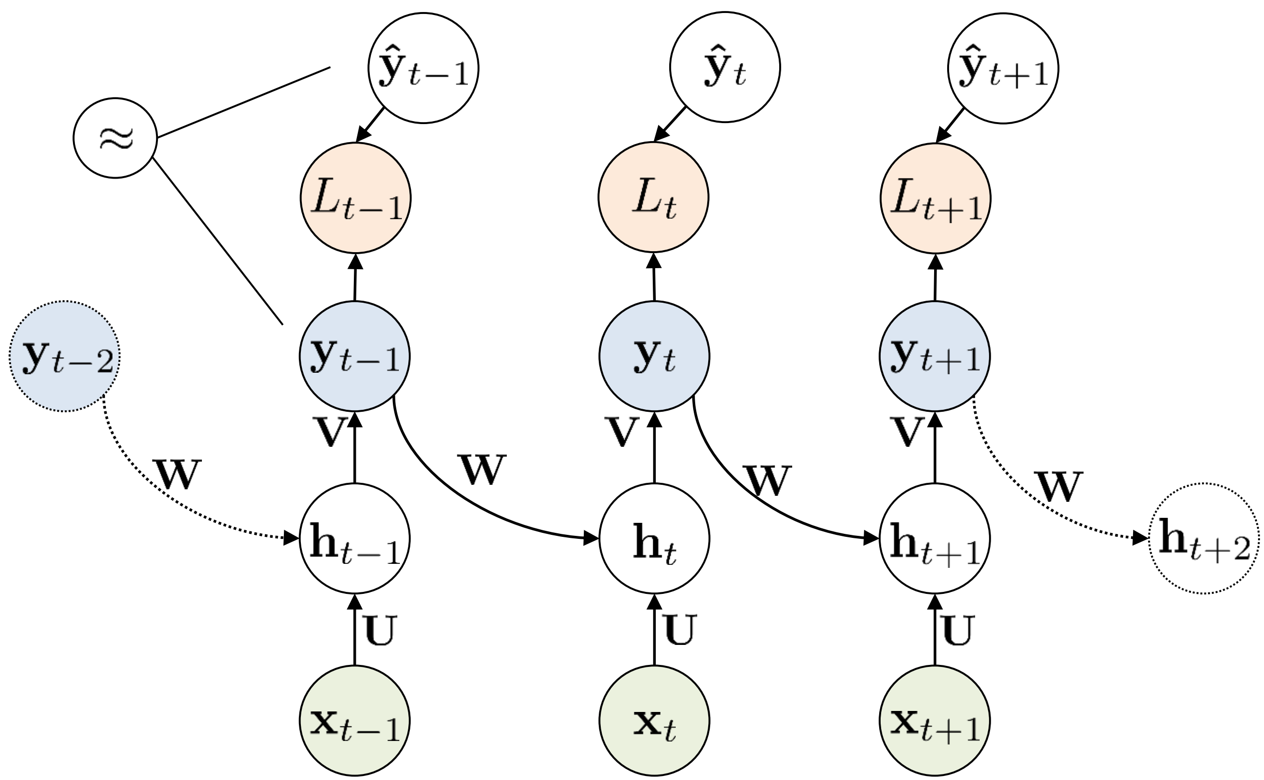


- Initialize as zero [2]
- Noisy zero mean [5]
- Treat as parameter to learn [6]
- Initialize using a second Neural Network [3]

2. Simple RNN and backpropagation through time

Feedback of the output

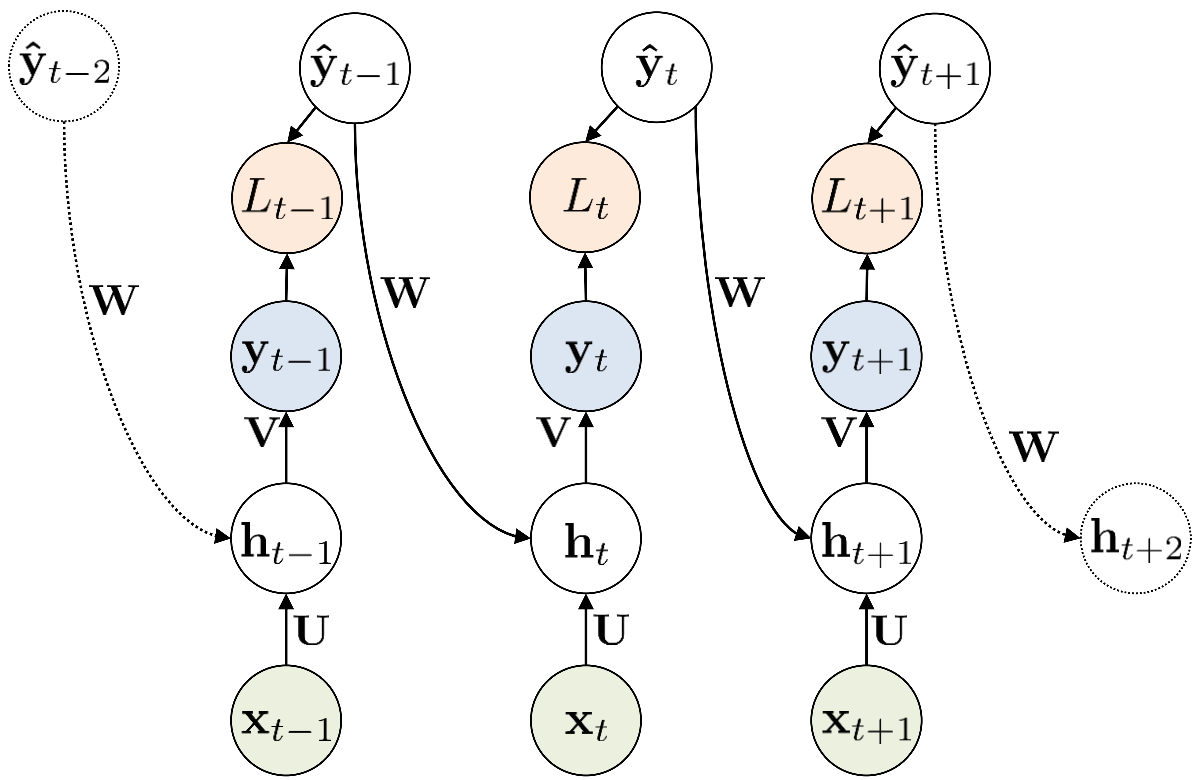
Feedback of the **output**. We can split the computation graph by using the target values from the data → supervised learning



2. Simple RNN and backpropagation through time

Teacher forcing

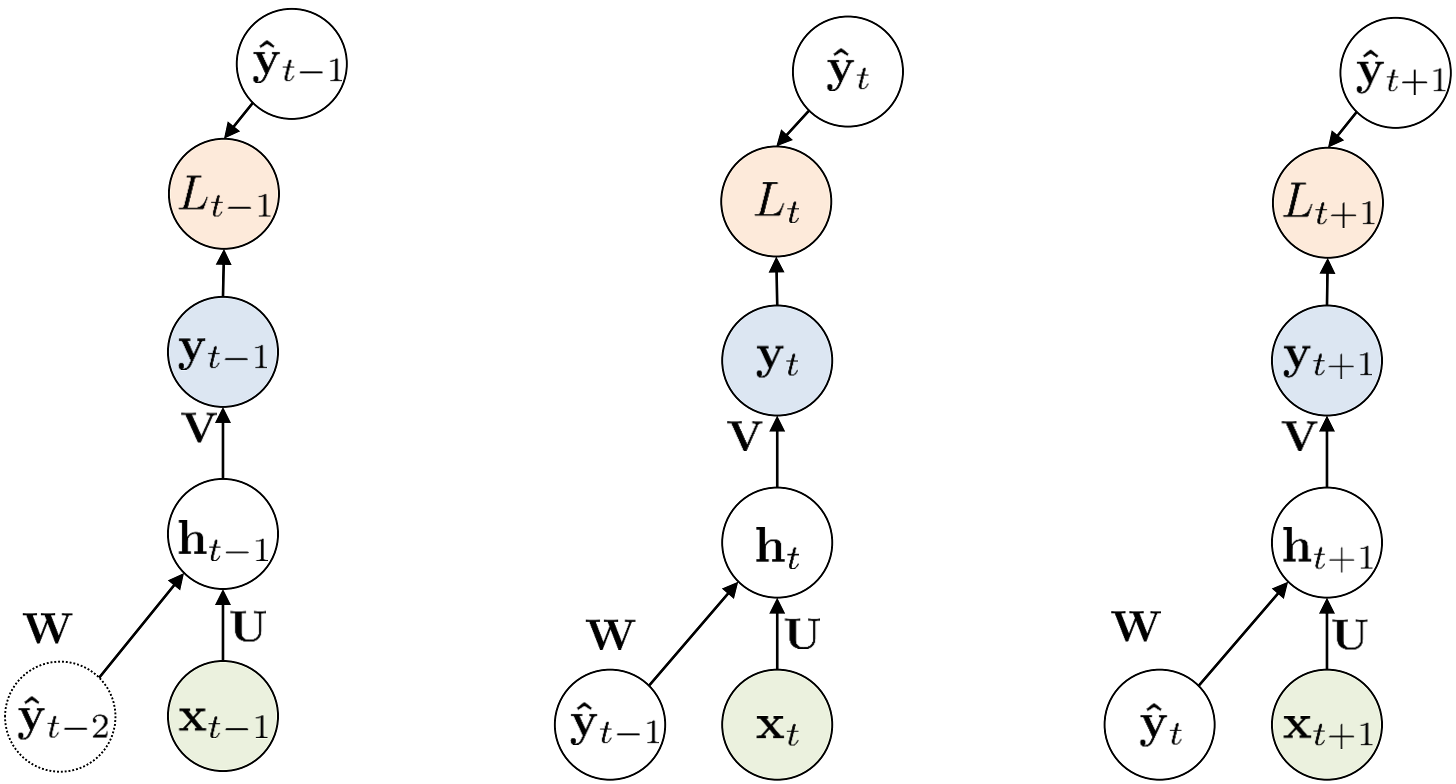
Feedback of the **output**. We can split the computation graph by using the target values from the data → supervised learning



2. Simple RNN and backpropagation through time

Teacher forcing

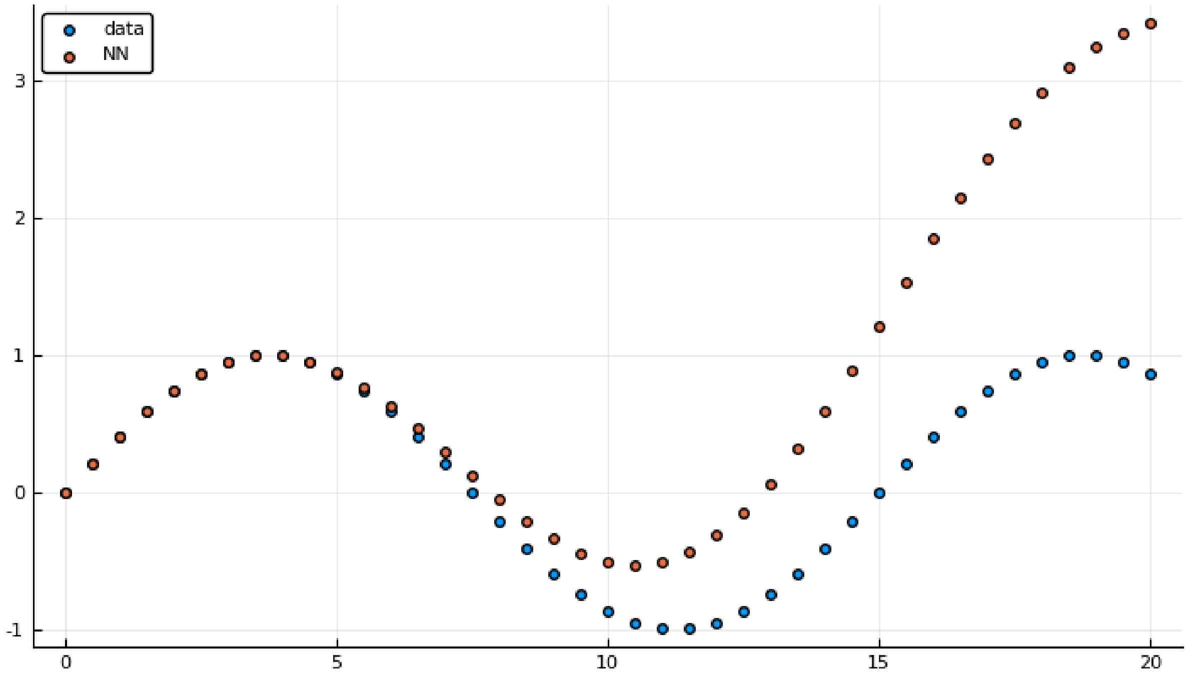
Feedback of the **output**. We can split the computation graph by using the target values from the data → supervised learning



2. Simple RNN and backpropagation through time

Teacher forcing

Problem with accumulating errors when sampling longer sequences.

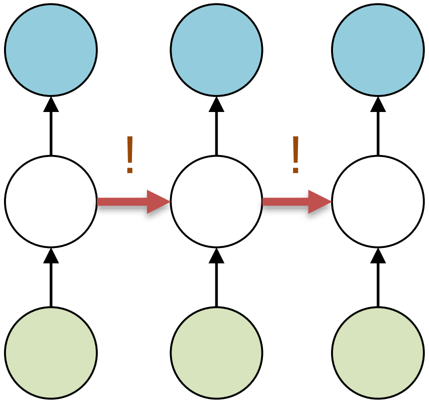


2. Simple RNN and backpropagation through time

Regularization using Dropout

Problematic when used on shared weights

- Regularize non-shared parameter only
- Sample one dropout pattern for the shared parameters and reuse it for the timesteps in the minibatch



2. Simple RNN and backpropagation through time

Wrap up

- A RNN is just a **state-space model with free parameters** that we want to learn.
- Gradient descend using **backpropagation as usual**, but:
  - Computation graph deep (need to store and compute a lot)
    -> use **truncated sequences**
  - But truncated sequences lead to **bias** in the gradient
- If we use **output as input** again, we can use **teacher forcing**
  -> no need to backpropagate over sequences

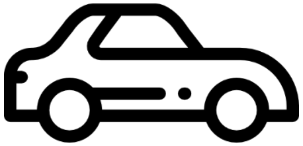
Recurrent Neural Networks

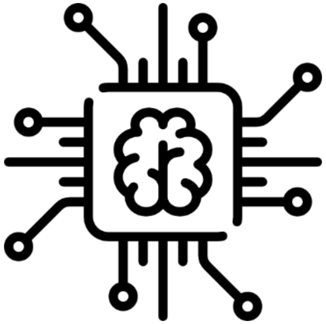
Johannes Betz / Prof. Dr. Markus Lienkamp / Prof. Dr. Boris Lohmann

(Christian Dengler, M. Sc.)

Agenda

1. Sequential Data and Use Cases
2. Simple RNN and Backpropagation through Time
3. **<u>Challenge of long Term Dependencies</u>**
4. Advanced RNN Structures
5. Recurrent Neural Networks for Automobiles





3. Challenge of long term dependencies

Vanishing or exploding Gradients

Suppose we have a recurrent function

$$h_{t+1} =$$

$$h_{t+k} =$$

$$\frac{\partial h_{t+k}}{\partial h_t} =$$

For long sequences

$$\lim_{k \to \infty} \frac{\partial h_{t+k}}{\partial h_t} =$$

3. Challenge of long term dependencies

Vanishing or exploding Gradients

$$\mathbf{h}_{t+1} = \tanh(\mathbf{W}\mathbf{h}_t + \mathbf{U}\mathbf{x}_t + \mathbf{b})$$
$$\mathbf{y}_t = \mathbf{V}\mathbf{h}_t + \mathbf{v}$$

For backpropagation we need

$$\frac{\partial L_{t+k}}{\partial \mathbf{h}_t^T} = \frac{\partial L_{t+k}}{\partial \mathbf{h}_{t+k}^T} \frac{\partial \mathbf{h}_{t+k}}{\partial \mathbf{h}_t^T}$$

$$\frac{\partial \mathbf{h}_{t+1}}{\partial \mathbf{h}_t^T} = \operatorname{diag}(1 - \mathbf{h}_{t+1} \odot \mathbf{h}_{t+1})\mathbf{W}$$

3. Challenge of long term dependencies

Vanishing or exploding Gradients

For matrices, we need to look at the spectral norm = largest singular value

$$\gamma = \max \left\| \mathrm{diag}(1 - \tanh(\dots)^2) \right\|$$

One can show that

$$\left\| \frac{\partial \mathbf{h}_{t+1}}{\partial \mathbf{h}_t^T} \right\| \leq \|\gamma \mathbf{W}\|$$

$$\left\| \frac{\partial \mathbf{h}_{t+k}}{\partial \mathbf{h}_t^T} \right\| \leq \|\gamma \mathbf{W}\|^k$$

3. Challenge of long term dependencies

Vanishing or exploding Gradients

We want

$$\left\| \frac{\partial \mathbf{h}_{t+k}}{\partial \mathbf{h}_t^T} \right\| \leq \|\gamma \mathbf{W}\|^k \approx 1$$

We know γ, so initializing $\mathbf{W}$ using a good distribution helps at the beginning of the training.

3. Challenge of long term dependencies

Exploding gradient, Gradient clipping

If gradient is to big, use only the direction, not the magnitude.

$$\text{If } \|\mathbf{g}\| > \nu:$$
$$\mathbf{g}^* = \nu \cdot \frac{\mathbf{g}}{\|\mathbf{g}\|}$$

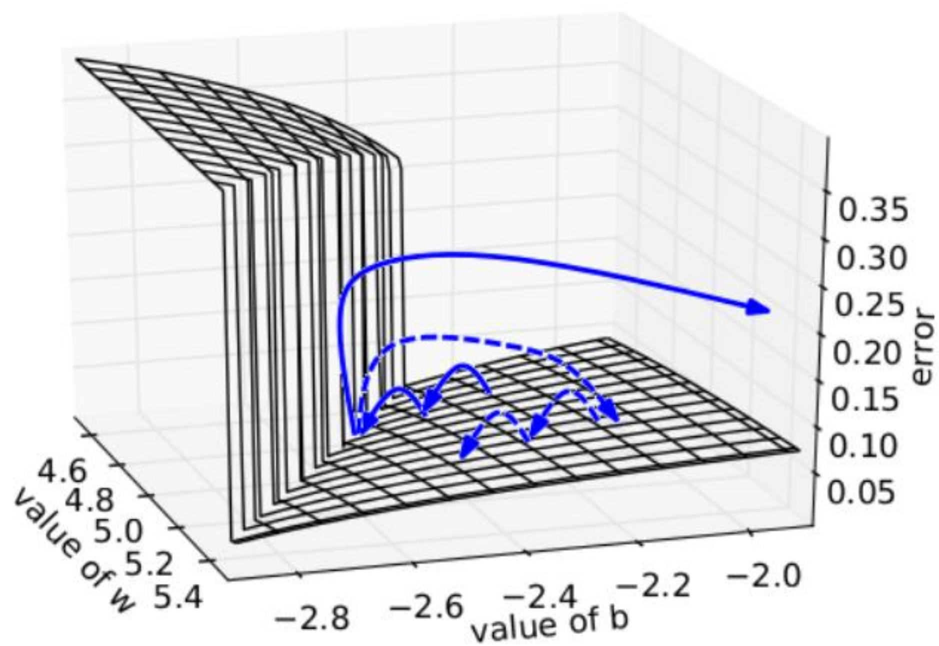
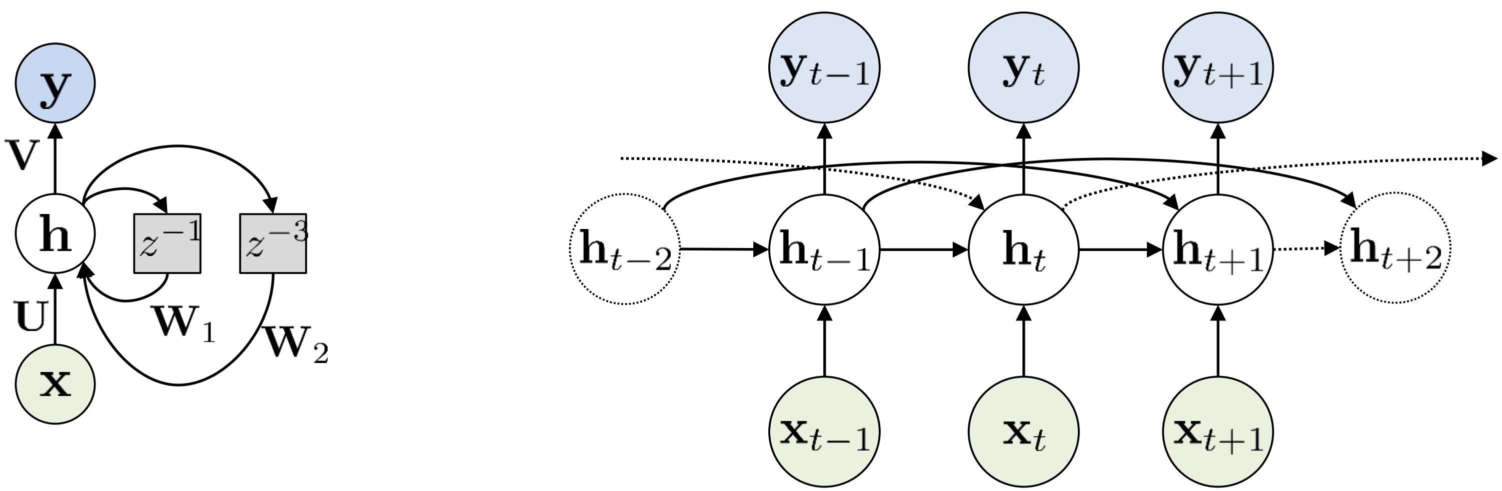


Image from [10]

3. Challenge of long term dependencies

Vanishing Gradient, Skip Connections

„Jump over" iterative gradients



Today: use **different parameterization**, e.g. gated network

3. Challenge of long term dependencies

Wrap up

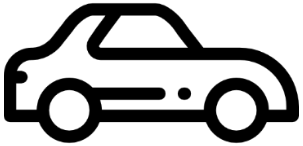
- Reusing the same weights can lead to **very big or very small gradients** that can slow down training.
- **Big gradients** should be **diminished in some way**, e.g. cut norm, cut single entires
- **Small gradients** can be tackled by using **inputs from multiple steps in the past**
- Good **initialization of the recurrent** weights can avoid small or big gradients at the beginning of the training.

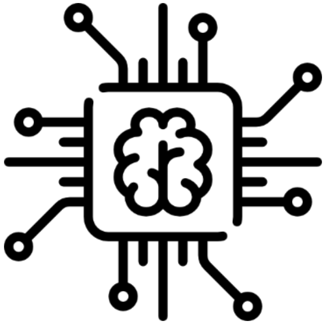
Recurrent Neural Networks

Johannes Betz / Prof. Dr. Markus Lienkamp / Prof. Dr. Boris Lohmann
(Christian Dengler, M. Sc.)

Agenda

1. Sequential Data and Use Cases
2. Simple RNN and Backpropagation through Time
3. Challenge of long Term Dependencies
4. **Advanced RNN Structures**
5. Recurrent Neural Networks for Automobiles

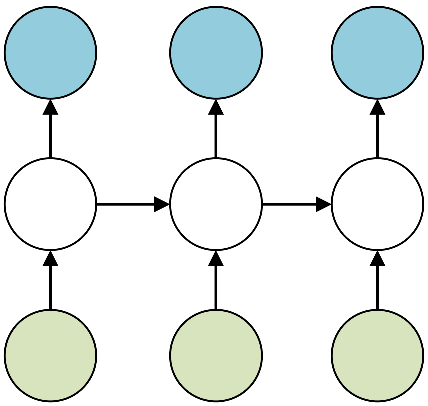




4. Advanced RNN structures

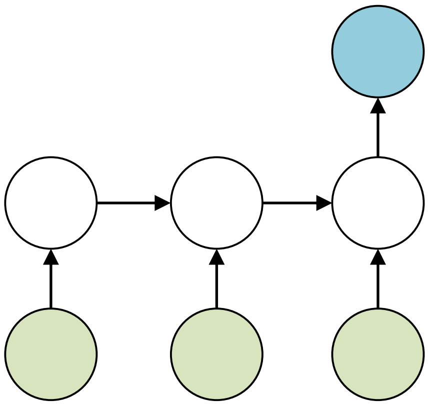
Input – Output relations

Many to many



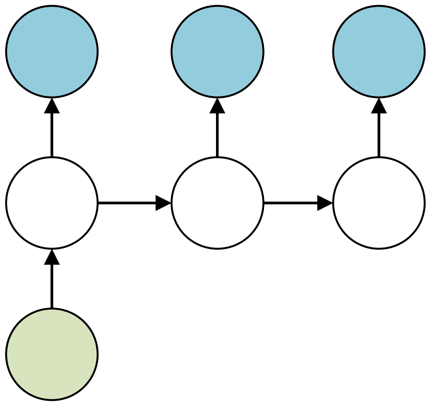
E.g. Approximate the dynamics of a physical system

Many to one



E.g. classify a video

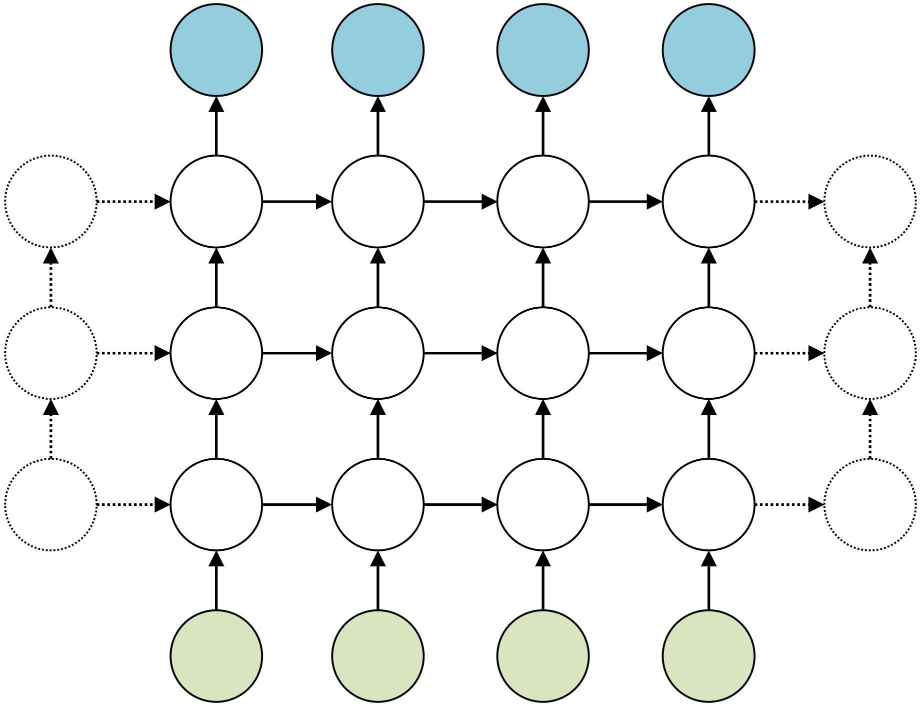
One to many



E.g. describe an image with a sentence

4. Advanced RNN structures

Multilayer RNN

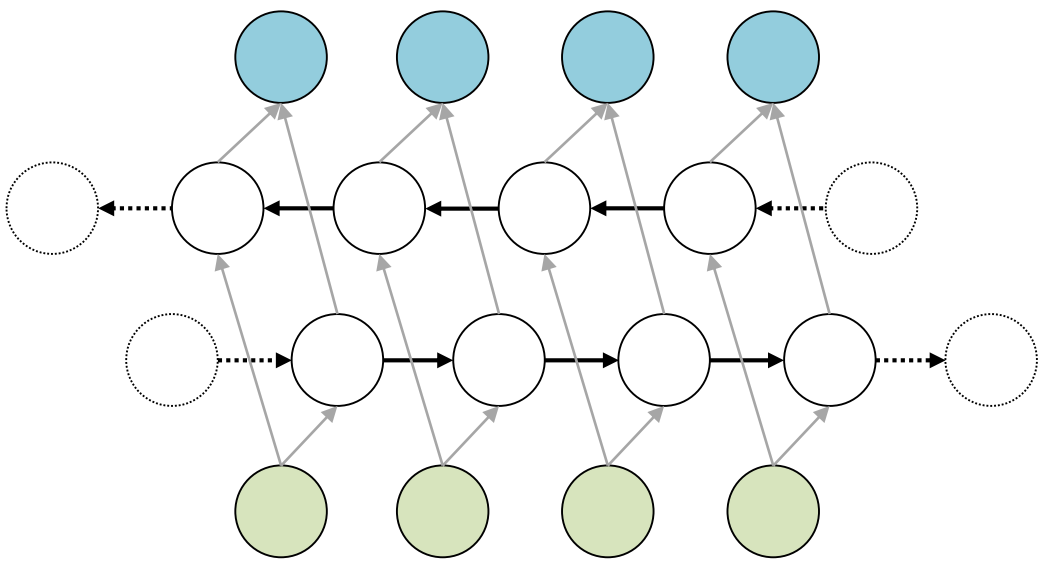


Short analysis of the influence of multiple layers in [11]

4. Advanced RNN structures

Bidirectional RNN [4]

Including future information can be helpful. E.g. handwriting recognition.

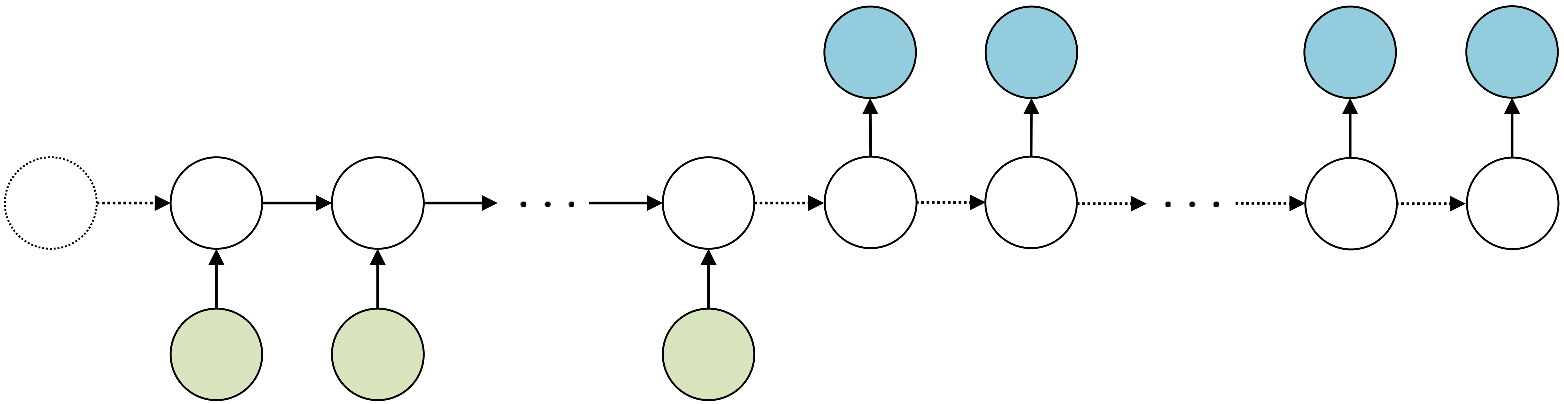


4. Advanced RNN structures

Sequence to sequence [7]

1) A sequence is encoded by a RNN with parameters $\mathbf{W}_1$
2) The information is decoded by RNN with parameters $\mathbf{W}_2$

E.g. translation into different languages.



4. Advanced RNN structures

Long short-term memory (LSTM) [8]

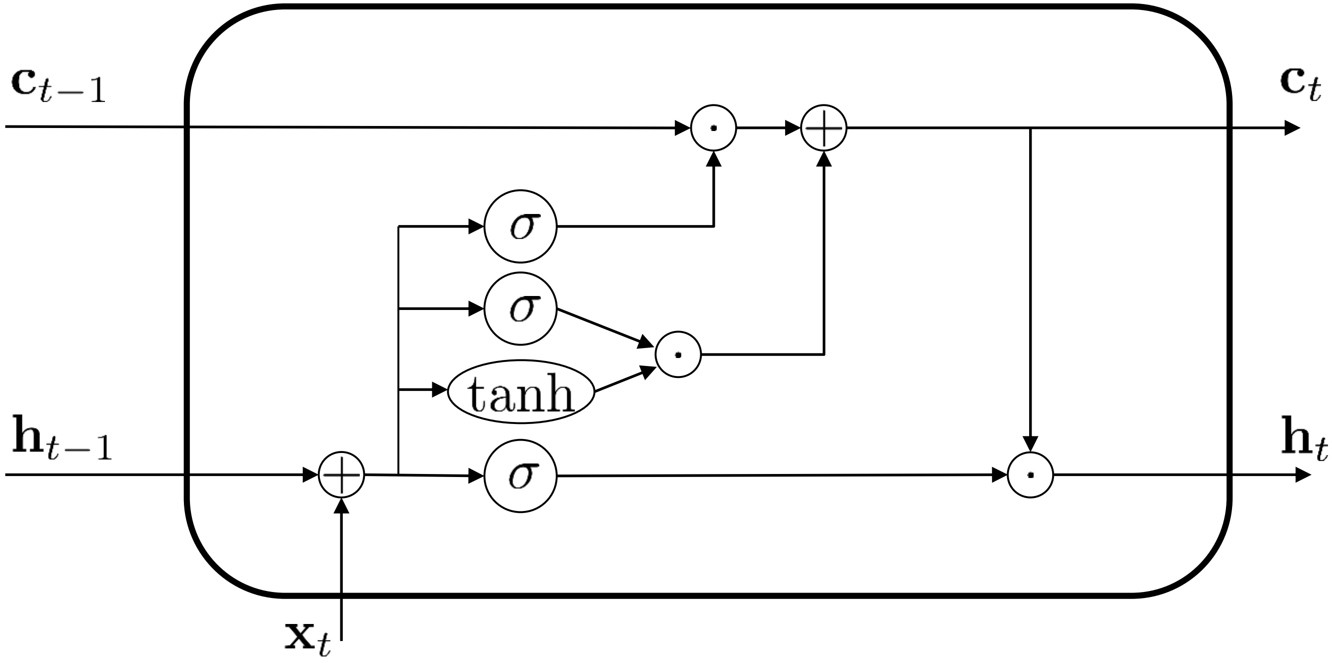
- Idea of not updating the whole hidden state each time
- Protect the state from being overwritten by useless information
- Be selective in
  - What to write (input gate) $\mathbf{i}_t = \sigma(\mathbf{W}_i \mathbf{h}_{t-1} + \mathbf{U}_i \mathbf{x}_t + \mathbf{b}_i)$
  - What to read (output gate) $\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{h}_{t-1} + \mathbf{U}_o \mathbf{x}_t + \mathbf{b}_o)$
  - What to forget (forget gate) $\mathbf{f}_t = \sigma(\mathbf{W}_f \mathbf{h}_{t-1} + \mathbf{U}_f \mathbf{x}_t + \mathbf{b}_f)$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{U}_h \mathbf{x}_t + \mathbf{b}_h)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot c_t$$

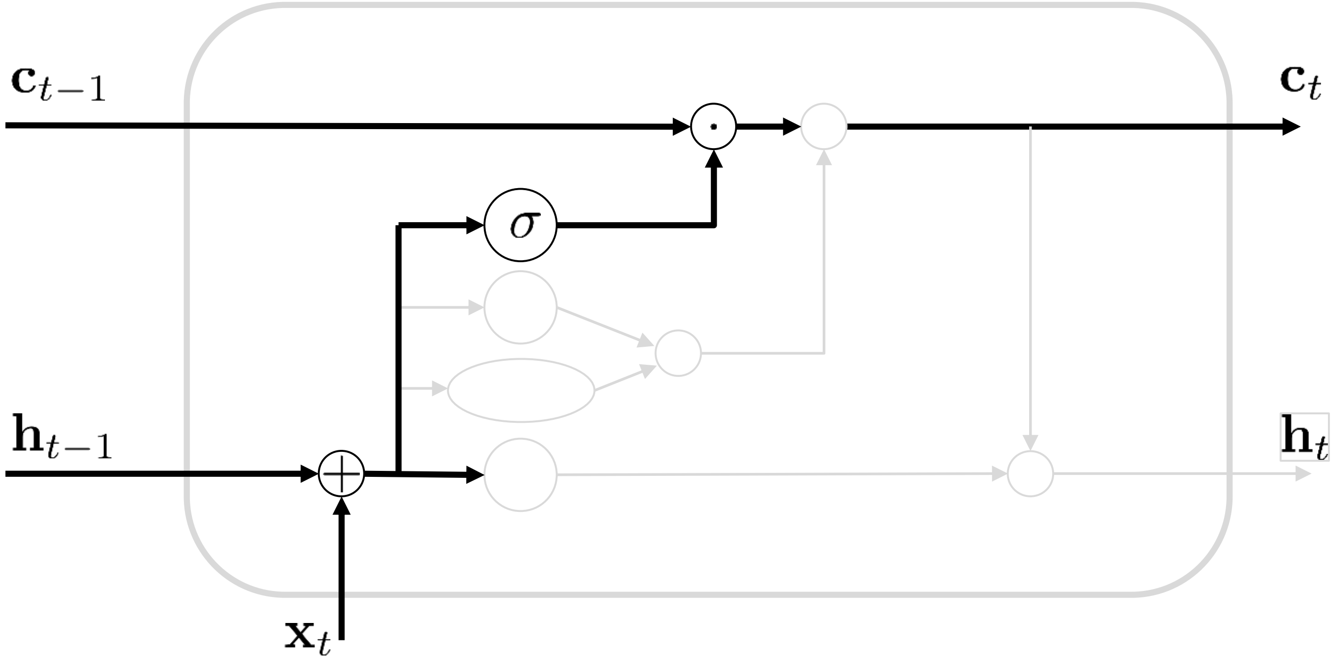
4. Advanced RNN structures

LSTM Structure



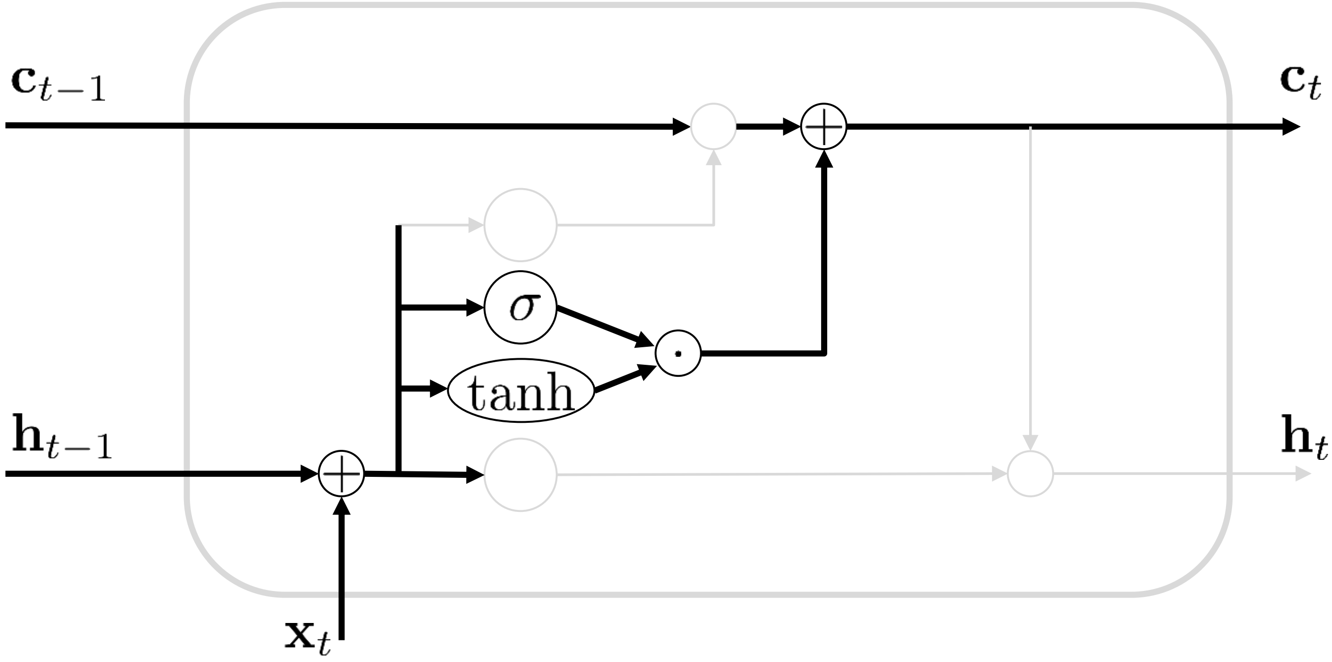
4. Advanced RNN structures

LSTM, forget



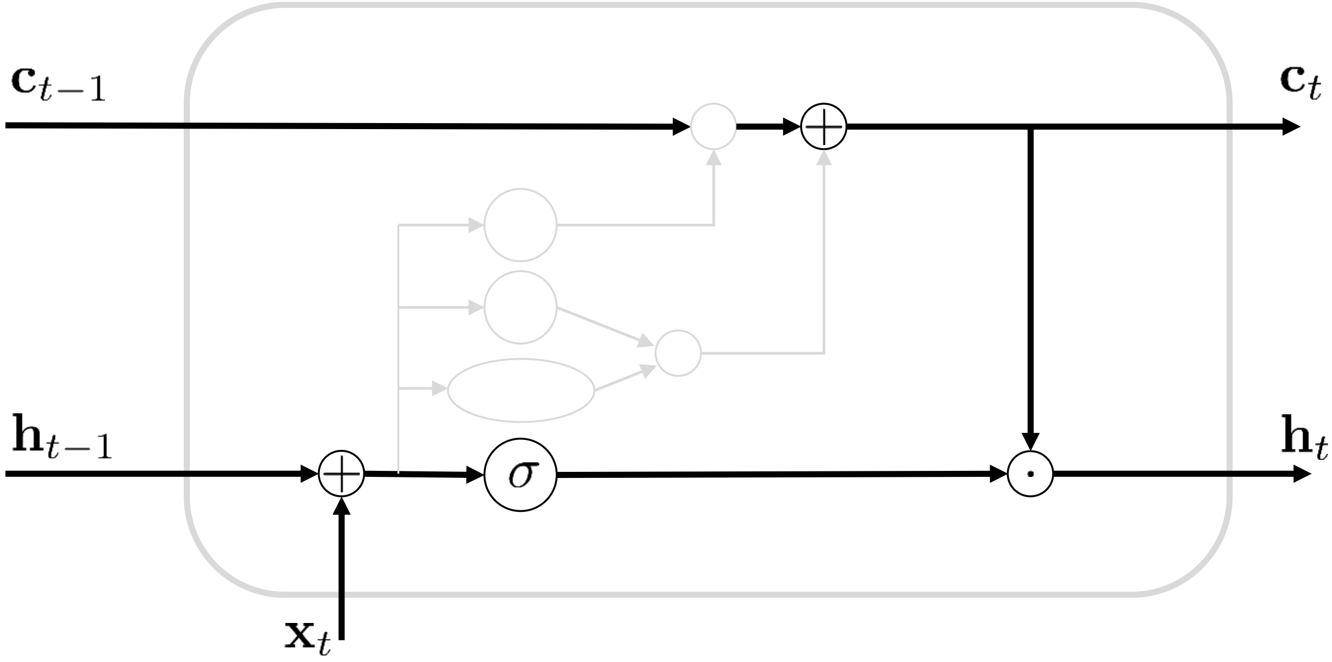
4. Advanced RNN structures

LSTM, input



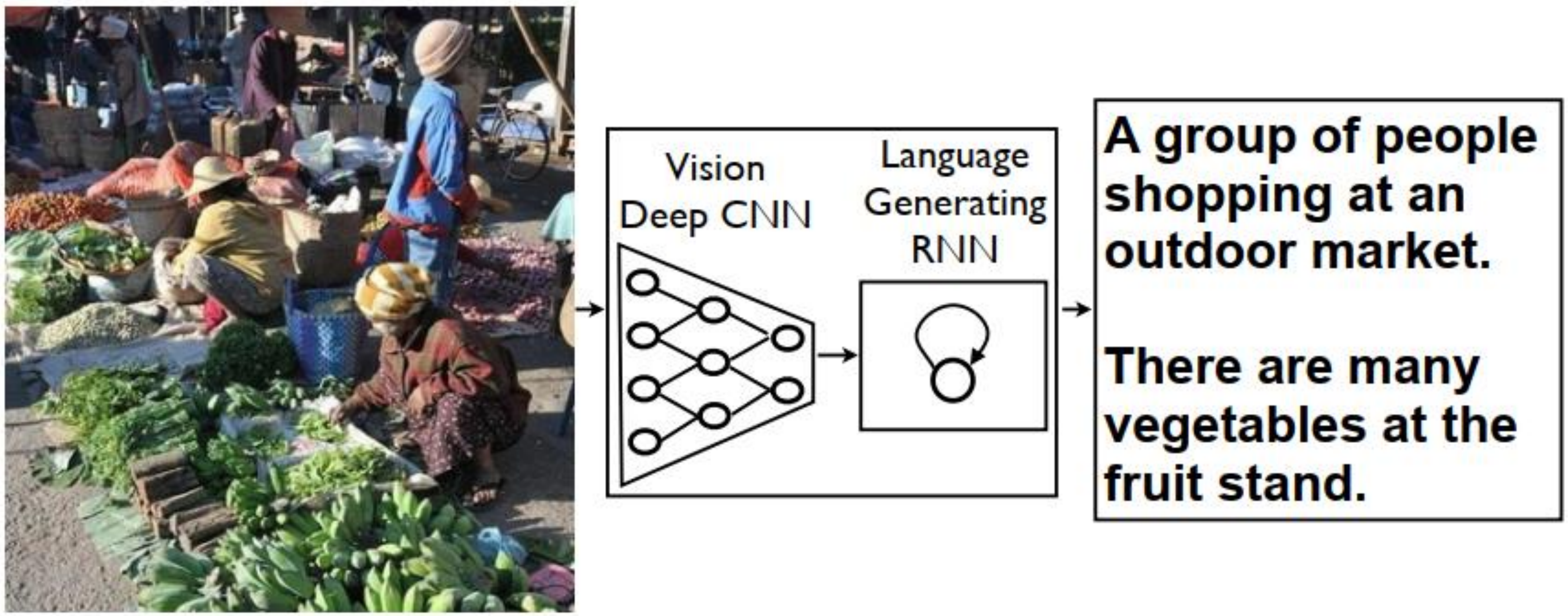
4. Advanced RNN structures

LSTM, output



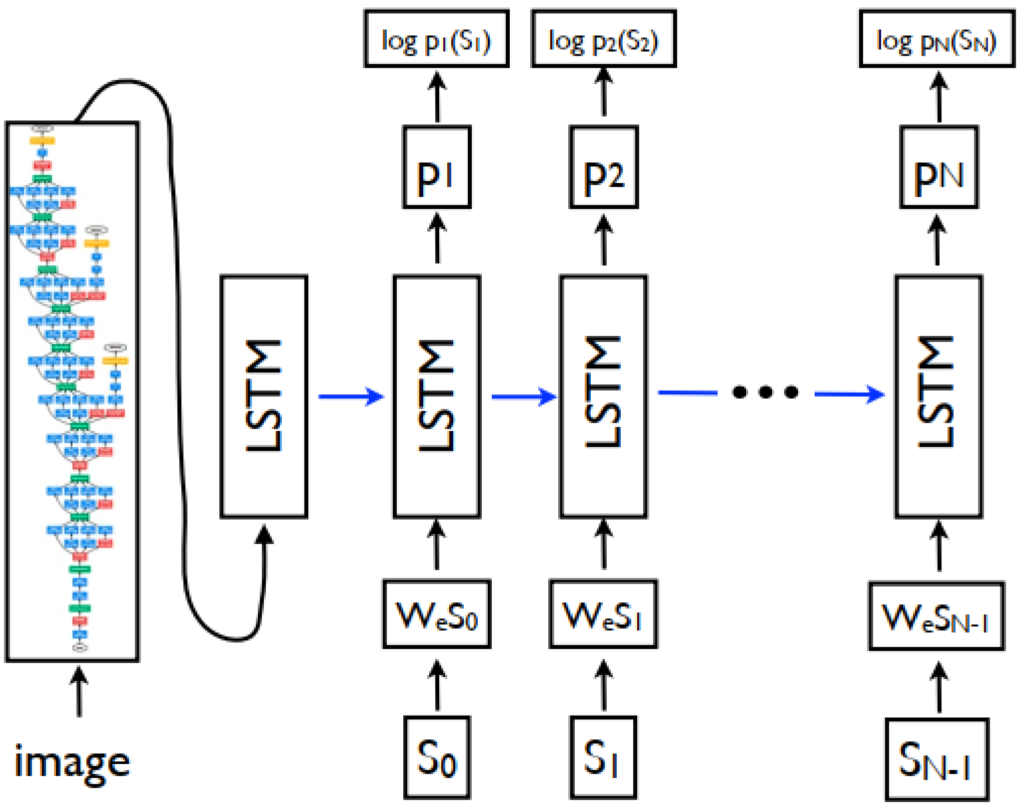
4. Advanced RNN structures

LSTM example in Vinyals et al. [9]



4. Advanced RNN structures

LSTM example in [9]



4. Advanced RNN structures

LSTM example in [9]

4. Advanced RNN structures

Gated Recurrent Unit (GRU)

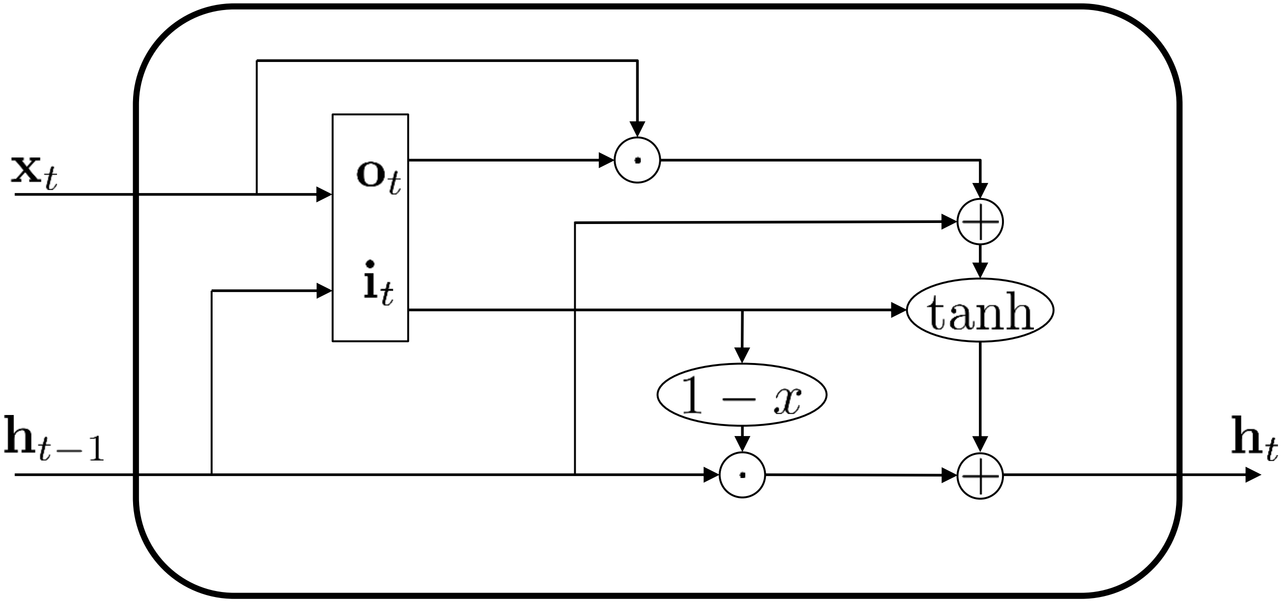
- Same idea using gates, here
  - Input/write $\mathbf{i}_t = \sigma(\mathbf{W}_i \mathbf{h}_{t-1} + \mathbf{U}_i \mathbf{x}_t + \mathbf{b}_i)$
  - Forget $\mathbf{f}_t = 1 - \mathbf{i}_t$
  - Output/read $\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{h}_{t-1} + \mathbf{U}_o \mathbf{x}_t + \mathbf{b}_o)$

$$\mathbf{h}_{t+1} = (1 - \mathbf{i}_t) \odot h_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{U}_h (\mathbf{o}_t \odot \mathbf{x}_t) + \mathbf{b}_h)$$

- Read first then write!

4. Advanced RNN structures

GRU Structure



4. Advanced RNN structures

Wrap up

- **Many structures and activations possible**: deep RNN, bidirectional RNN, sequence to sequence, LSTM-cells, GRU,…
- **Hard to know a priori** what will work best. Currently LSTM and GRU used a lot.

Wrap up

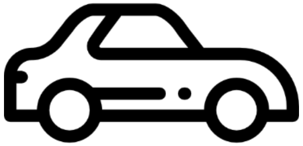
- Recurrent Neural Network = **Dynamical System**
- Common use cases: Wherever **data appears in sequences**, e.g. audio, video, text, language…
- **Unfolding in time** is simular to training deep neural network, but additional difficulties appear: deep computation graph, initial state, **biased gradient**, gradients small or large…
- **Different RNN structures** might be favorable for different situations: bidirection RNN, sequence to sequence, gated networks…
- No „one fits all“ solution, still topic of research.

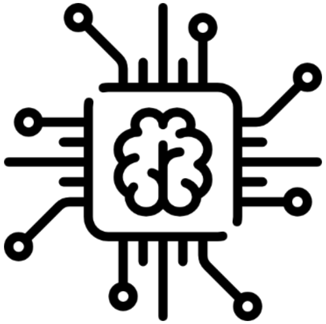
Recurrent Neural Networks

Johannes Betz / Prof. Dr. Markus Lienkamp / Prof. Dr. Boris Lohmann
(Christian Dengler, M. Sc.)

Agenda

1. Sequential Data and Use Cases
2. Simple RNN and Backpropagation through Time
3. Challenge of long Term Dependencies
4. Advanced RNN Structures
5. **<u>Recurrent Neural Networks for Automobiles</u>**





5. Recurrent Neural Networks in Automobility

Recurrent neural networks for driver activity anticipation via sensory-fusion architecture
Jain, A.; Singh, A.; Koppula, H. S.; Soh, S. & Saxena, A.
2016 IEEE International Conference on Robotics and Automation (ICRA), **2016**

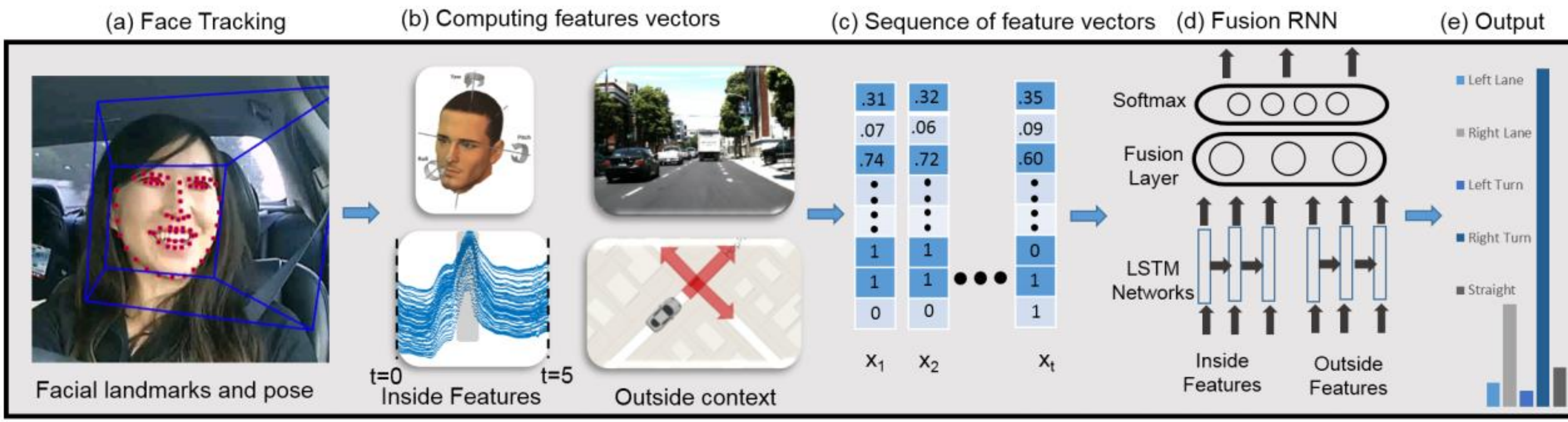
Maneuver anticipation

Predict the drivers action multiple seconds ahead.

→multiple sensors and LSTM‘s fused

5. Recurrent Neural Networks in Automobility

Recurrent neural networks for driver activity anticipation via sensory-fusion architecture



- **Multiple LSTM**, one for each sensor (camera, GPS, vehicle dynamics,…)
- Sensor fusion on hidden states as fully connected
- Loss function with increased loss in late predictions

5. Recurrent Neural Networks in Automobility

Recurrent neural networks for driver activity anticipation via sensory-fusion architecture

		Lane change			Turns		
Method		Pr (%)	Re (%)	Time-to-maneuver (s)	Pr (%)	Re (%)	Time-to-maneuver (s)
	Chance	33.3	33.3	-	33.3	33.3	-
	SVM [27]	73.7 ± 3.4	57.8 ± 2.8	2.40	64.7 ± 6.5	47.2 ± 7.6	2.40
	Random-Forest	71.2 ± 2.4	53.4 ± 3.2	3.00	68.6 ± 3.5	44.4 ± 3.5	1.20
	IOHMM [19]	81.6 ± 1.0	79.6 ± 1.9	3.98	77.6 ± 3.3	75.9 ± 2.5	4.42
	AIO-HMM [19]	83.8 ± 1.3	79.2 ± 2.9	3.80	80.8 ± 3.4	75.2 ± 2.4	4.16
Our Methods	S-RNN	85.4 ± 0.7	86.0 ± 1.4	3.53	75.2 ± 1.4	75.3 ± 2.1	3.68
	F-RNN-UL	**92.7** ± 2.1	84.4 ± 2.8	3.46	81.2 ± 3.5	78.6 ± 2.8	3.94
	F-RNN-EL	88.2 ± 1.4	**86.0** ± 0.7	3.42	**83.8** ± 2.1	**79.9** ± 3.5	3.78

- Comparison of different modifications, and also comparison with other works.

5. Recurrent Neural Networks in Automobility

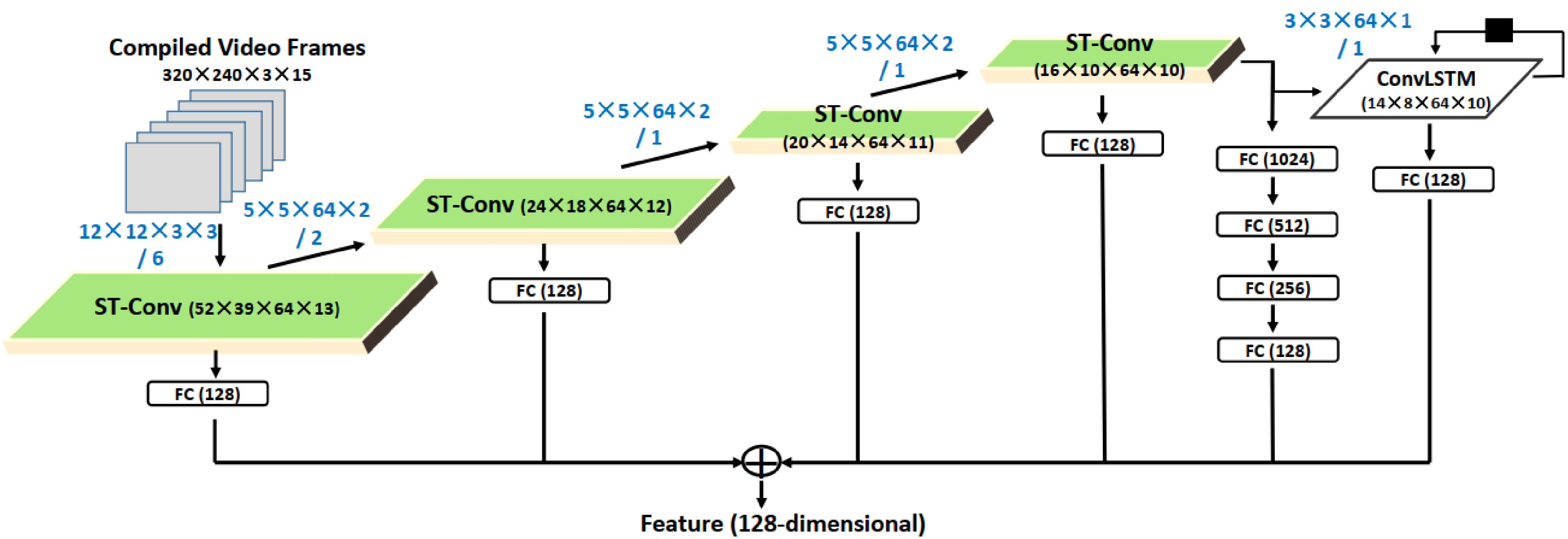
Deep steering: Learning end-to-end driving model from spatial and temporal visual cues
Chi, L. & Mu, Y.
arXiv preprint, **2017**

Only prediction in the paper, no driving ☹

5. Recurrent Neural Networks in Automobility

Deep steering: Learning end-to-end driving model from spatial and temporal visual cues

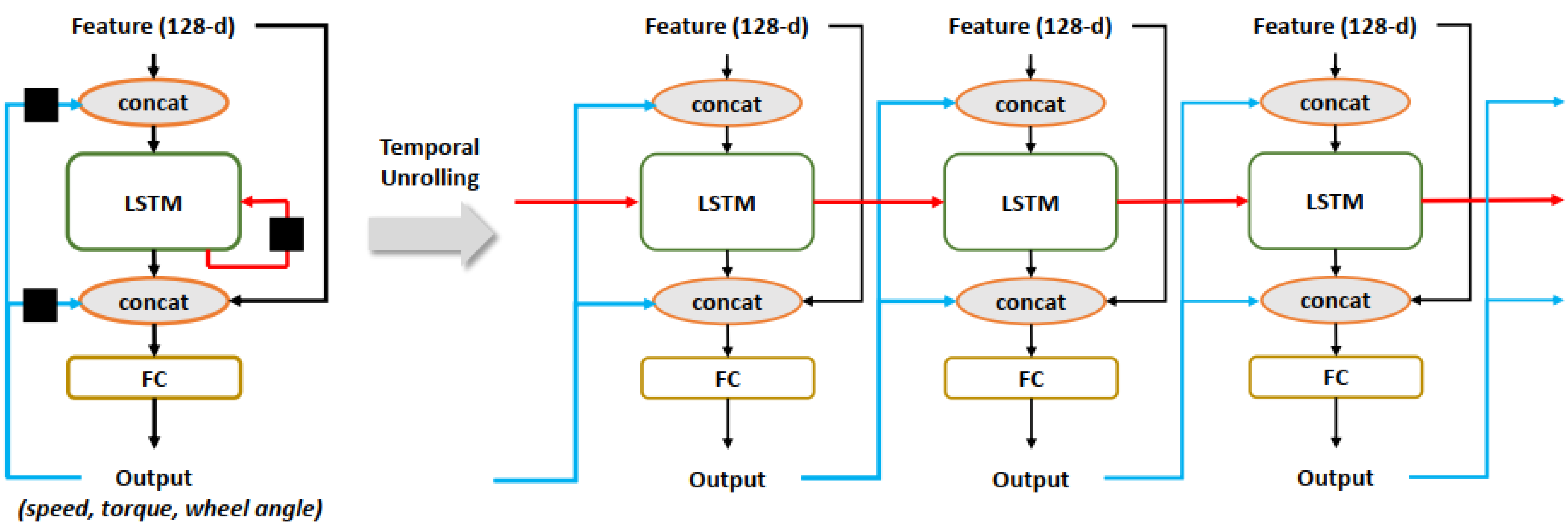
„Feature extraction sub-network“



5. Recurrent Neural Networks in Automobility

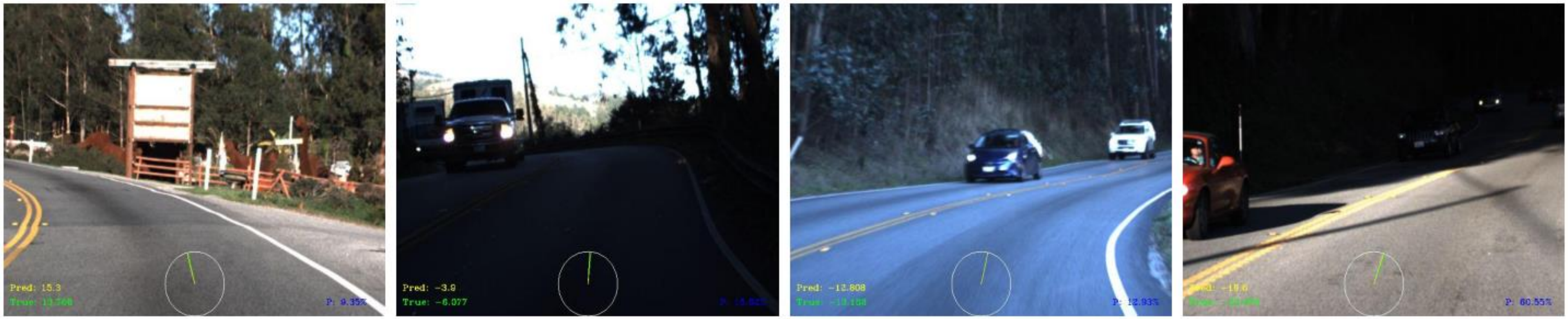
Deep steering: Learning end-to-end driving model from spatial and temporal visual cues

„Steering-prediction sub-network“



5. Recurrent Neural Networks in Automobility

Deep steering: Learning end-to-end driving model from spatial and temporal visual cues

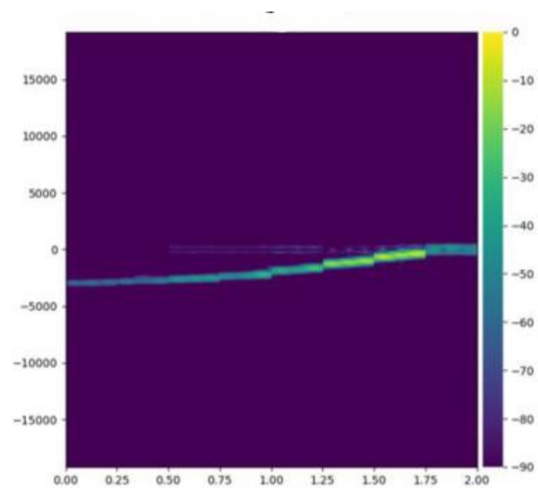


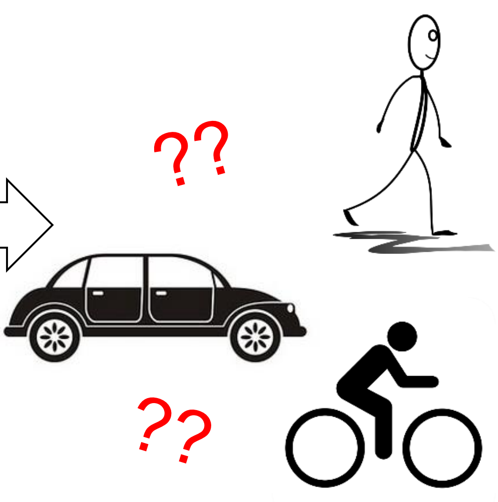
5. Recurrent Neural Networks in Automobility

Practical classification of different moving targets using automotive radar and deep neural networks

Angelov, A.; Robertson, A.; Murray-Smith, R. & Fioranelli, F.

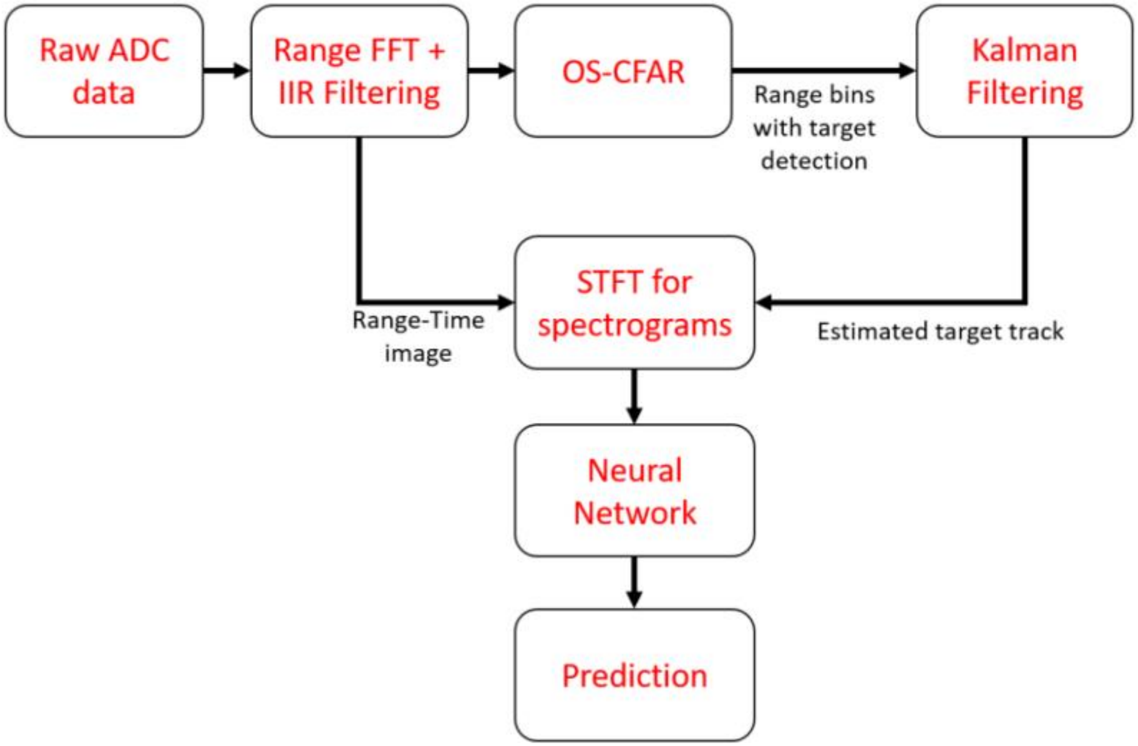
IET Radar, Sonar & Navigation, IET, **2018**





5. Recurrent Neural Networks in Automobility

Practical classification of different moving targets using automotive radar and deep neural networks



5. Recurrent Neural Networks in Automobility

Practical classification of different moving targets using automotive radar and deep neural networks

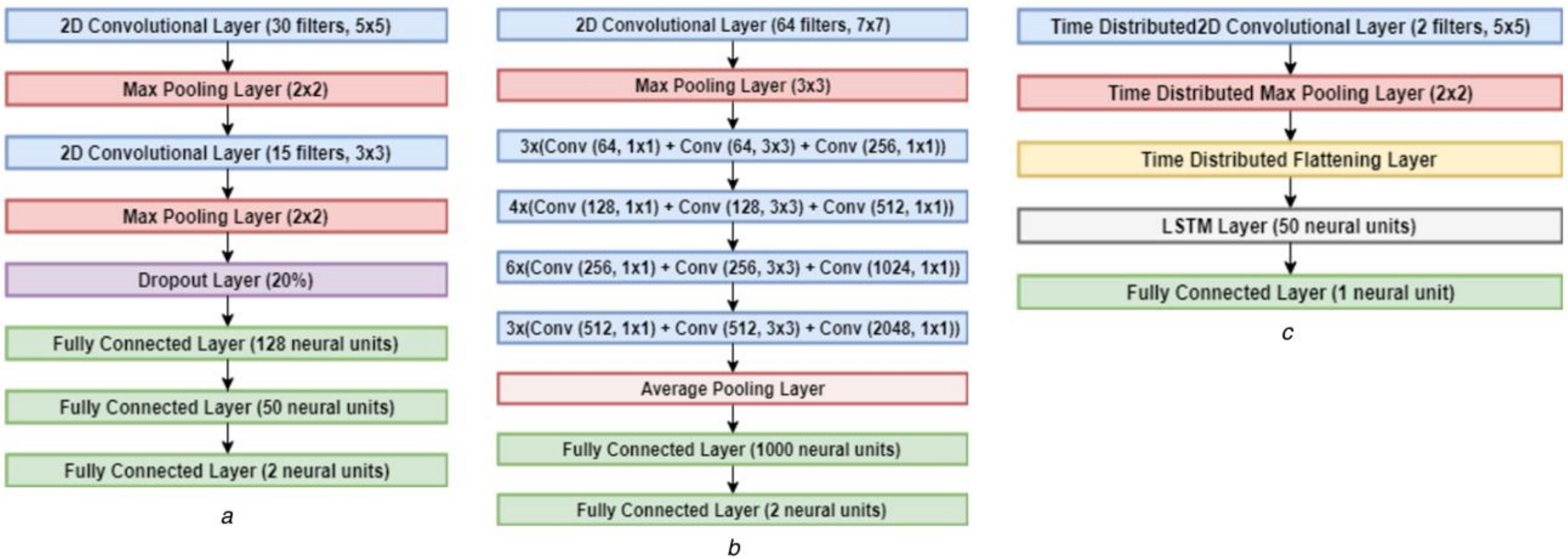


Fig. 3 *Representation of the different network architectures*
(a) Convolutional neural network similar to VGG type, ***(b)*** Convolutional residual network, ***(c)*** Combination of convolutional and recurrent LSTM network

5. Recurrent Neural Networks in Automobility

Practical classification of different moving targets using automotive radar and deep neural networks

Table 3 Test accuracy for two network architectures evaluated on three class problems

Evaluation/ network type	VGG-like CNN (2 s long datasets)	VGG-like CNN (0.5 s long datasets)	CNN-LSTM (2 s long datasets)	CNN-LSTM (0.5 s long datasets)
car-person-bicycle classification	79%	83%	93%	83%
car-person-2 people classification	81%	78%	80%	84%

Table 4 Test accuracy for three types of networks (VGG-like, CNN-LSTM, and VGG-LSTM) on all considered problems, with regularisation and batch normalisation

Evaluation/ network type	VGG-like CNN (2 s long datasets)	VGG-like CNN (0.5 s long datasets)	CNN-LSTM (2 s long datasets)	CNN-LSTM (0.5 s long datasets)
car-person-bicycle classification	78.6%	81.1%	50%	73.5%
car-person-2 people classification	77.8%	88.6%	44.4%	78.3%
all-4-classes-classification (VGG LSTM)	—	—	—	70%

Evaluation

Source: https://splinternews.com/12-years-of-steve-jobs-saying-one-more-thing-like-hes-c-1793850336

Evaluation

- In this lecture we are doing in regularly evaluation of each lecture
- We want **your** feedback for every **individual** lecture
- We evaluate the lecture each week
- We give feedback based on the evaluation the week after

Evaluation – Step by Step

1. Get out your smartphones
2. Open an app for QR-code Reading
3. Read the following QR-code on the right side
4. Open the website
5. Answer the questions
6. Send the evaluation

OR

1. Open the following website in your browser:
   https://evasys.zv.tum.de/evasys/online.php?p=AIAT-10
2. Answer the questions
3. Send the evaluation

A.1. Literature

[1] Tallec, C. & Ollivier, Y.: Unbiasing Truncated Backpropagation Through Time; *arXiv preprint arXiv:1705.08209,* **2017**

[2] Zhang, X.-Y.; Yin, F.; Zhang, Y.-M.; Liu, C.-L. & Bengio, Y.: Drawing and recognizing chinese characters with recurrent neural network; *IEEE transactions on pattern analysis and machine intelligence, IEEE,* **2018***, 40*, 849-862

[3] Mohajerin, N. & Waslander, S. L.: State initialization for recurrent neural network modeling of time-series data; *Neural Networks (IJCNN), 2017 International Joint Conference on,* **2017**, 2330-2337

[4] Schuster, M. & Paliwal, K. K.: Bidirectional recurrent neural networks; *IEEE Transactions on Signal Processing, IEEE,* **1997***, 45*, 2673-2681

[5] Zimmermann, H.-G.; Tietz, C. & Grothmann, R.: Forecasting with recurrent neural networks: 12 tricks; *Neural Networks: Tricks of the Trade, Springer,* **2012**, 687-707

[6] Geoffrey Hinton; Lecture 10, Recurrent neural networks; https://www.cs.toronto.edu/~hinton/csc2535/notes/lec10new.pdf , accessed 20.11.2018

A.1. Literature

[7] Sutskever, I.; Vinyals, O. & Le, Q. V.: Sequence to sequence learning with neural networks; *Advances in neural information processing systems,* **2014**, 3104-3112

[8] Hochreiter, S. & Schmidhuber, Jü.: Long short-term memory; *Neural computation, MIT Press,* **1997***, 9*, 1735-1780

[9] Vinyals, O.; Toshev, A.; Bengio, S. & Erhan, D.: Show and Tell: Lessons learned from the 2015 MSCOCO Image Captioning Challenge; *CoRR,* **2016***, abs/1609.06647*

[10] Pascanu, R.; Mikolov, T. & Bengio, Y.:On the difficulty of training recurrent neural networks; *International Conference on Machine Learning,* **2013**, 1310-1318

[11] Hermans, M. & Schrauwen, B.: Training and analysing deep recurrent neural networks; *Advances in neural information processing systems,* **2013**, 190-198

[12] Gregor, K.; Danihelka, I.; Graves, A. & Wierstra, D.: DRAW: A Recurrent Neural Network For Image Generation; *CoRR,* **2015***, abs/1502.04623*

[13] Jain, A.; Singh, A.; Koppula, H. S.; Soh, S. & Saxena, A.: Recurrent neural networks for driver activity anticipation via sensory-fusion architecture; *Robotics and Automation (ICRA), 2016 IEEE International Conference on,* **2016**, 3118-3125

A.1. Literature

[14] Chi, L. & Mu, Y.: Deep steering: Learning end-to-end driving model from spatial and temporal visual cues; *arXiv preprint arXiv:1708.03798,* **2017**

[15] Angelov, A.; Robertson, A.; Murray-Smith, R. & Fioranelli, F.: Practical classification of different moving targets using automotive radar and deep neural networks; *IET Radar, Sonar & Navigation, IET,* **2018***, 12*, 1082-1089

[16] Wikipedia: Inverted Pendulum; https://en.wikipedia.org/wiki/Inverted_pendulum