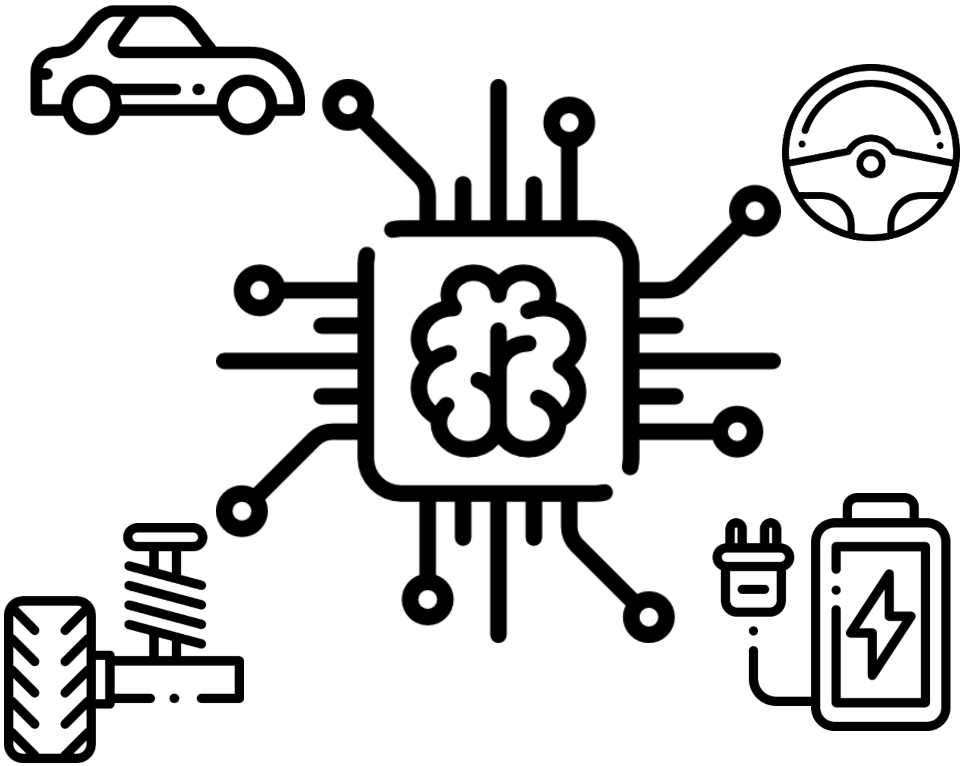
Institute of Automotive Technology
School of Engineering and Design
Technical University of Munich

Artificial Intelligence in Automotive Technology

Maximilian Geißlinger / Fabian Netzler

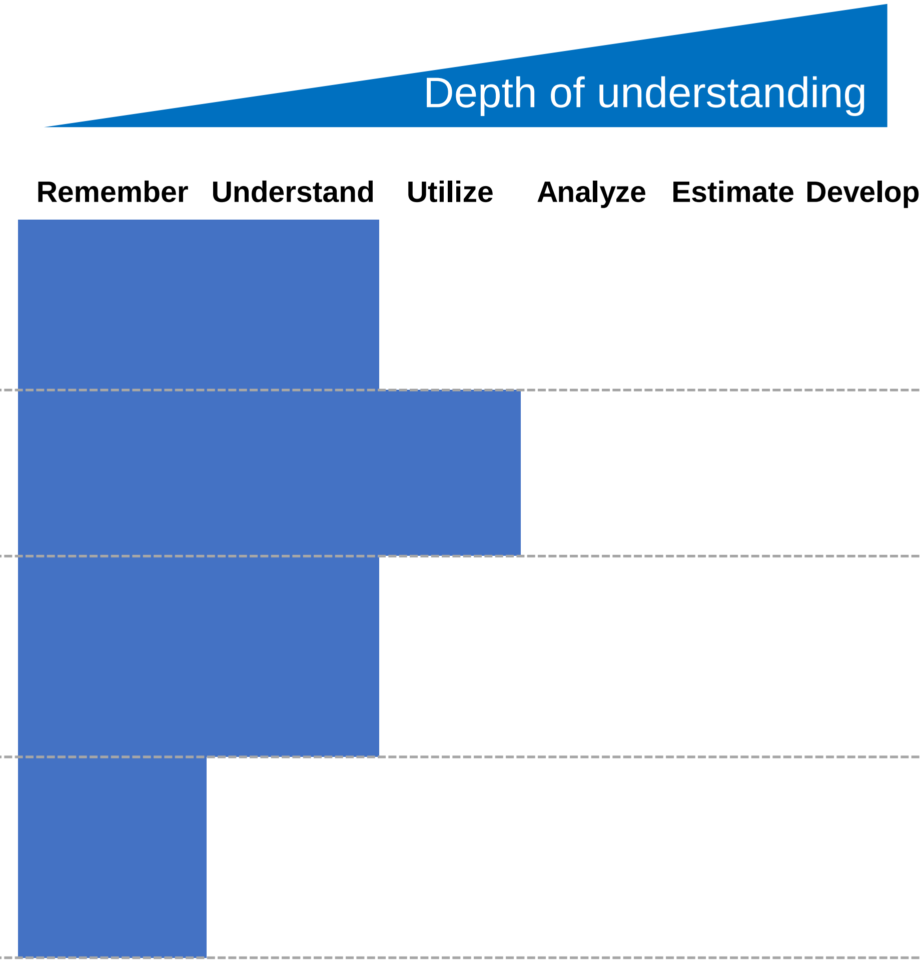
Prof. Dr.-Ing. Markus Lienkamp



Lecture Overview

<table>
<tr><th colspan="2">Lecture 16:15-17:45 | Practice 17:45-18:30</th></tr>
<tr><td>1 Introduction: Artificial Intelligence</td><td>20.10.2022 – Maximilian Geißlinger</td></tr>
<tr><td>2 Perception</td><td>27.10.2022 – Sebastian Huber</td></tr>
<tr><td>3 Supervised Learning: Regression</td><td>03.11.2022 – Fabian Netzler</td></tr>
<tr><td>4 Supervised Learning: Classification</td><td>10.11.2022 – Andreas Schimpe</td></tr>
<tr><td>5 Unsupervised Learning: Clustering</td><td>17.11.2022 – Andreas Schimpe</td></tr>
<tr><td>6 Introduction: Artificial Neural Networks</td><td>24.11.2022 – Lennart Adenaw</td></tr>
<tr><td>7 Deep Neural Networks</td><td>08.12.2022 – Domagoj Majstorovic</td></tr>
<tr><td>8 Convolutional Neural Networks</td><td>15.12.2022 – Domagoj Majstorovic</td></tr>
<tr><td>9 Knowledge Graphs</td><td>12.01.2023 – Fabian Netzler</td></tr>
<tr><td>10 Recurrent Neural Networks</td><td>19.01.2023 – Matthias Rowold</td></tr>
<tr><td>11 Reinforcement Learning</td><td>26.01.2023 – Levent Ögretmen</td></tr>
<tr><td>12 AI-Development</td><td>02.02.2023 – Maximilian Geißlinger</td></tr>
<tr><td>13 Guest Lecture</td><td>09.02.2023 – to be announced</td></tr>
</table>

Objectives of Lecture 10



Clarify when Recurrent Neural Networks (RNN) are preferred over ‘static’ Neural Networks.

Show the basic methods for training a RNN.

Clarify problems that might occur when training a RNN for long sequences and how to avoid / mitigate them.

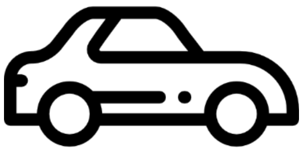
Give a short overview of different architectures for RNNs and their respective use cases.

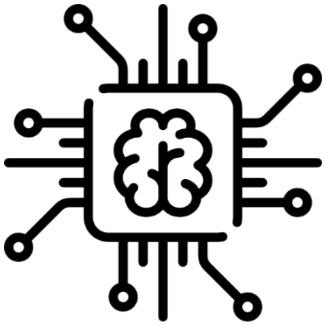
Recurrent Neural Networks

Maximilian Geißlinger / Fabian Netzler / Prof. Dr. Markus Lienkamp
(Matthias Rowold, M. Sc.)

Agenda

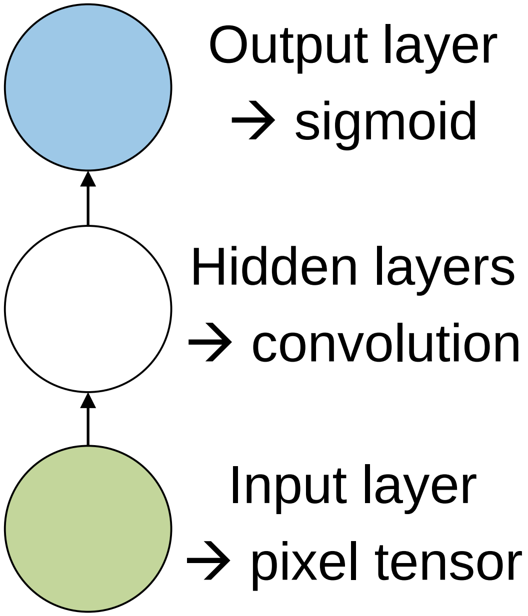
1. **Introduction**
   1. Motivating example
   2. The “hidden state”
   3. Connection to dynamical systems
2. Training RNNs
   1. Backpropagation through time
   2. Methods
3. Vanishing and exploding gradients
4. Advanced RNN structures
5. Examples of RNNs in automotive applications





1.1 Introduction – Motivating Example

Will he score?



1.1 Introduction – Motivating Example

Will he score?

1.1 Introduction – Motivating Example

He misses.

1.1 Introduction – Motivating Example

Output layer (e.g. sigmoid)

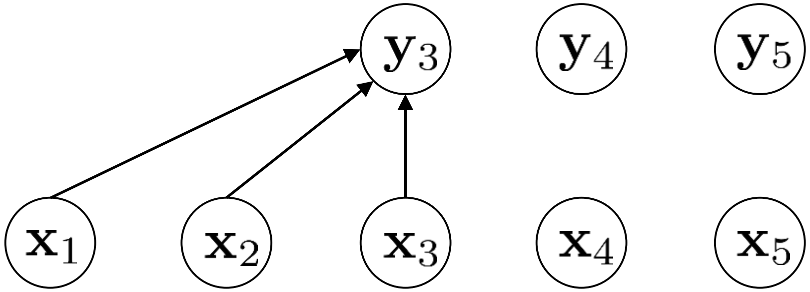
Hidden layers
(e.g. convolutional and pooling)

Input layer

Problem: size

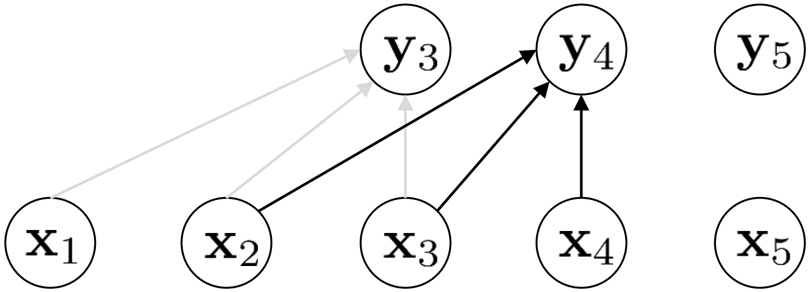
1.2 Introduction – Hidden State

NN with a large input layer for sequences:



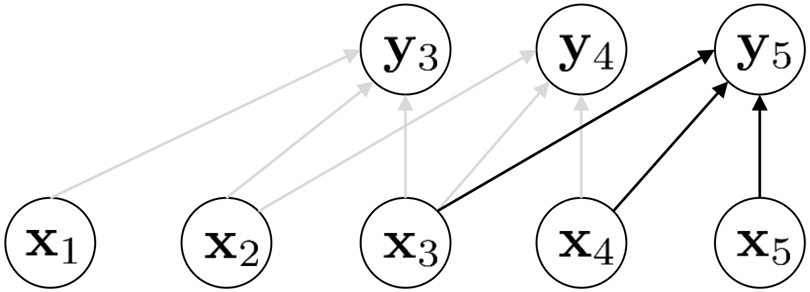
1.2 Introduction – Hidden State

NN with a large input layer for sequences:



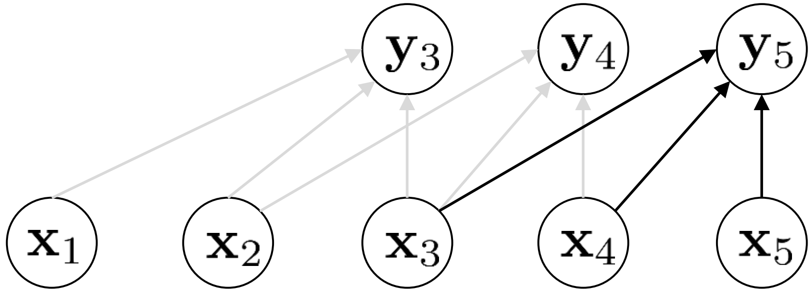
1.2 Introduction – Hidden State

NN with a large input layer for sequences:

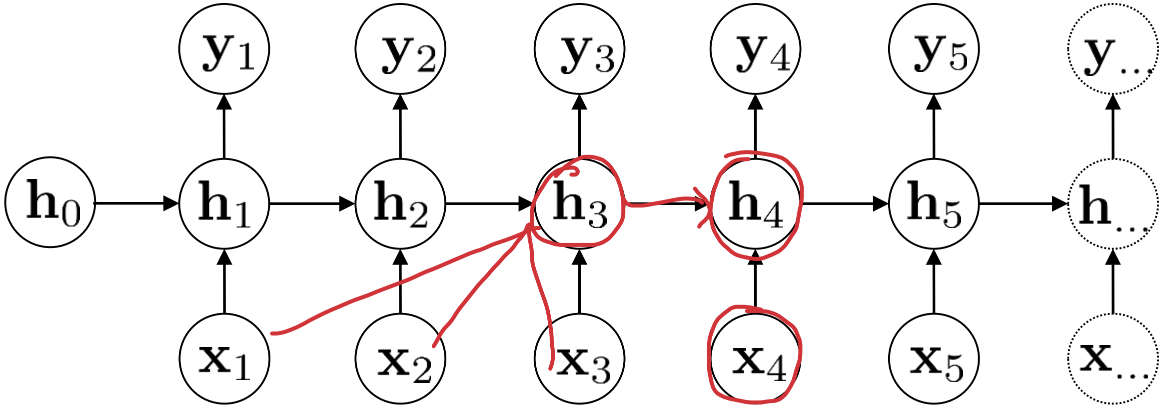


1.2 Introduction – Hidden State

NN with a large input layer for sequences:

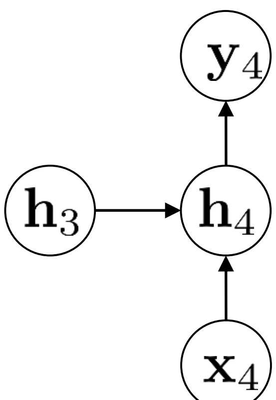


Hidden state summarizes the past sequence:



1.2 Introduction – Hidden State

Back to our example:



$\mathbf{y}_4$ Predicted outcome: hit or miss

$\mathbf{h}_4$ Hidden state at time $t = 4$:

$\mathbf{x}_4$ Input at time $t = 4$: pixel values

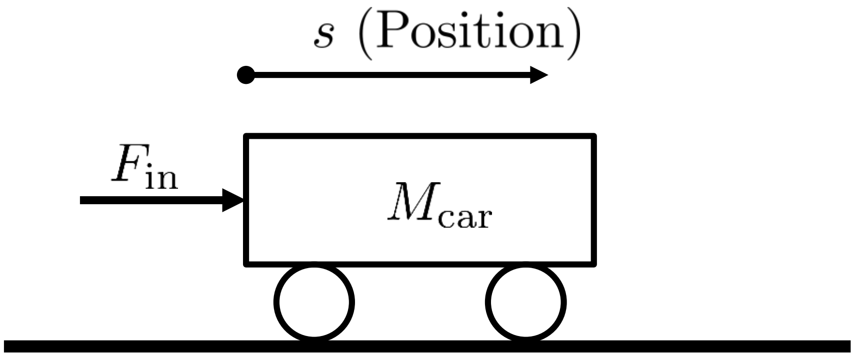
$\mathbf{h}_3$ Hidden state at time $t = 3$:

1. Introduction

Use cases of RNNs

- Speech recognition and generation
- Music recognition and generation
- Translation
- Image capturing
- Video capturing
- Prediction of movement of other traffic participants
- **Modeling dynamics of physical systems**
- …

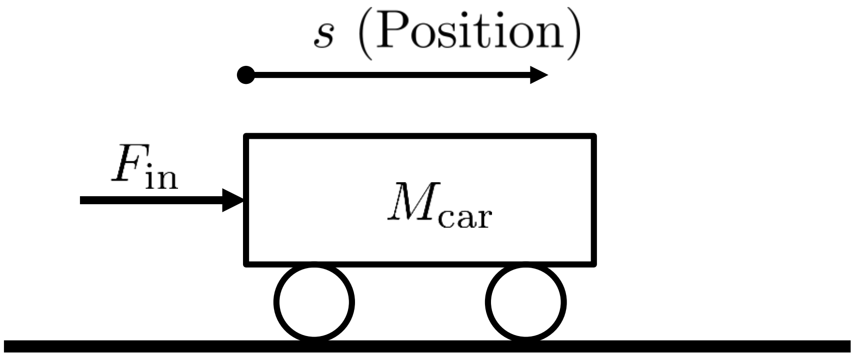
1.3 Connection to Dynamical Systems



Input $u = F_{\text{in}}$

Output $y = s$

1.3 Connection to Dynamical Systems



Input $u = F_{\text{in}}$

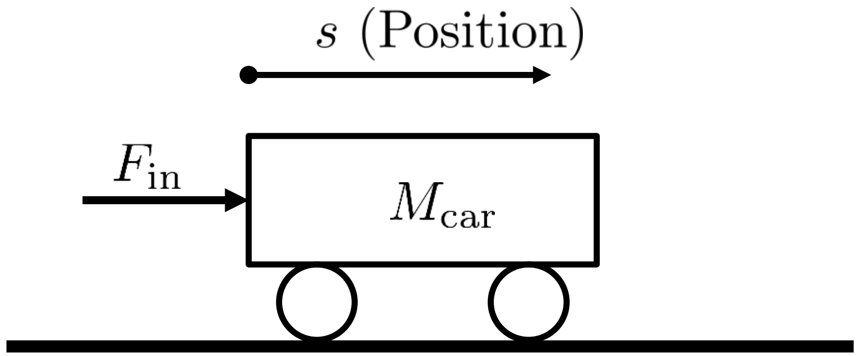
Output $y = s$

Model equations:

$$v = \dot{s}$$

$$M_{\text{car}}\dot{v} = F_{\text{in}}$$

1.3 Connection to Dynamical Systems



Input $u = F_{\text{in}}$

Output $y = s$

Model equations:

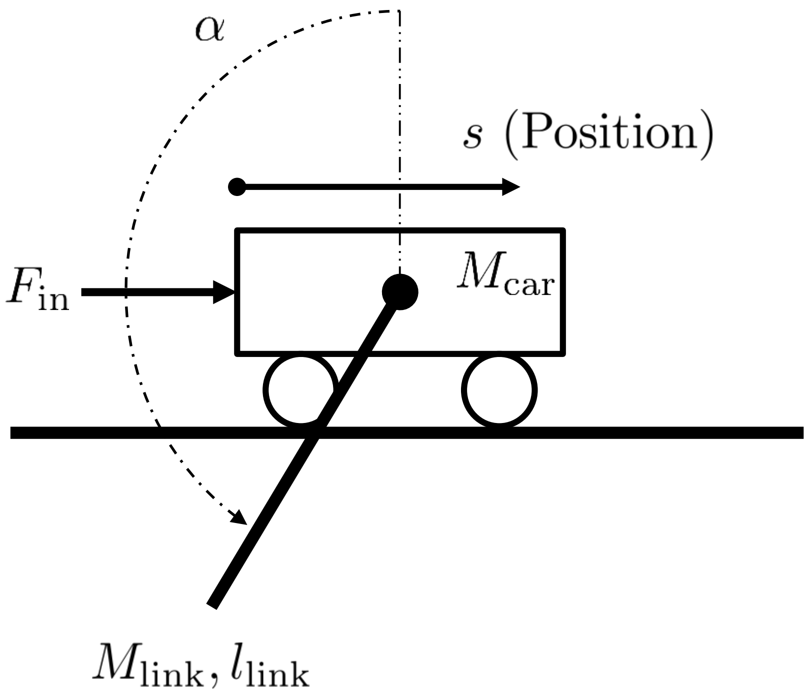
$$v = \dot{s}$$
$$M_{\text{car}}\dot{v} = F_{\text{in}}$$

Linear state-space model:

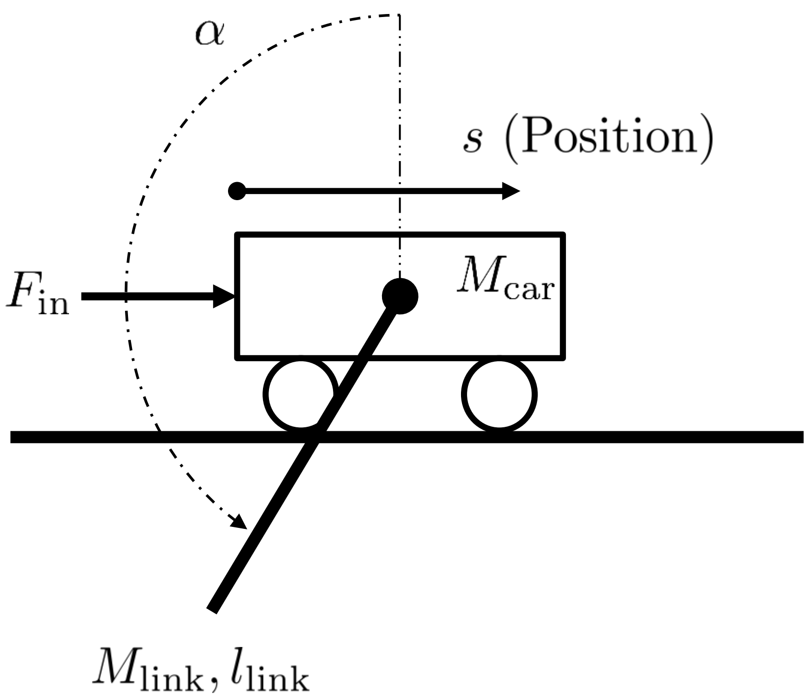
$$\begin{bmatrix} \dot{s} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} s \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{M_{\text{car}}} \end{bmatrix} F_{\text{in}}$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} s \\ v \end{bmatrix}$$

1.3 Connection to Dynamical Systems



1.3 Connection to Dynamical Systems



Non-linear state-space model:

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

$$y = s$$

Model equations:

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

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

1.3 Connection to Dynamical Systems

Continuous time system

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

Discretization:

- Euler's Method
- Runge-Kutta
- …

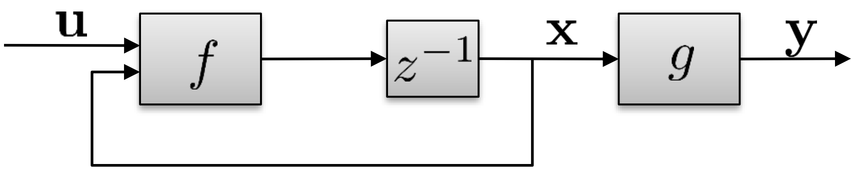
Discrete time system

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

1.3 Connection to Dynamical Systems

Engineering / control theory:

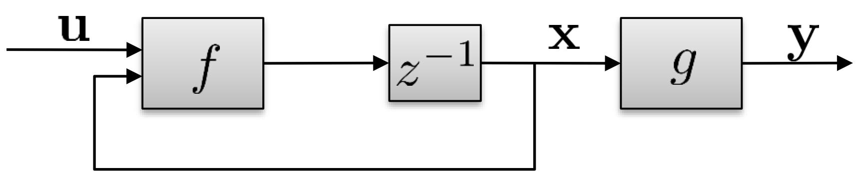
$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$$
$$\mathbf{y}_t = g(\boldsymbol{x}_t)$$



1.3 Connection to Dynamical Systems

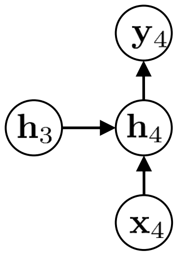
Engineering / control theory:

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



Machine learning:

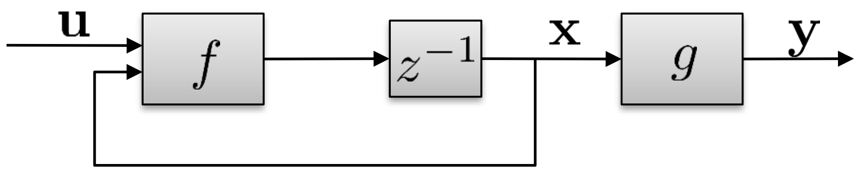
$$\boldsymbol{h}_t = f(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t)$$
$$\mathbf{y}_t = g(\boldsymbol{h}_t)$$



1.3 Connection to Dynamical Systems

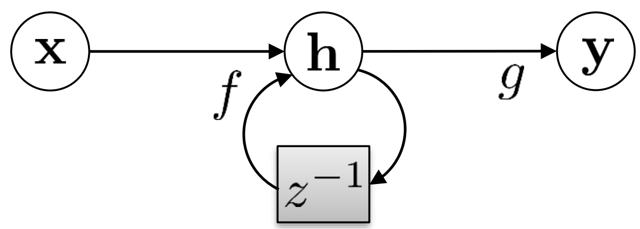
Engineering / control theory:

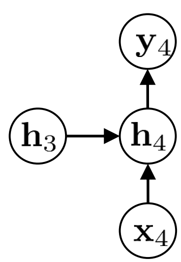
$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$$
$$\mathbf{y}_t = g(\boldsymbol{x}_t)$$



Machine learning:

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

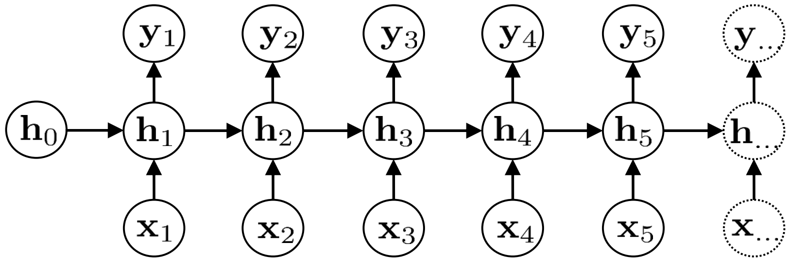




1. Introduction: Wrap Up

- Often one observation does not contain the required information to make an accurate prediction.
- The information is hidden in a **sequence of data.**
- Taking a whole sequence as an input for a NN requires too many parameters.

→ Share parameters and add a memory (hidden state) to capture important features of the past:



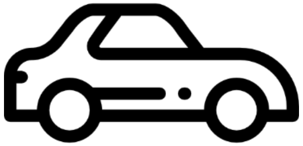
- A RNN can be interpreted as a **state-space model** with free parameters that we want to learn.

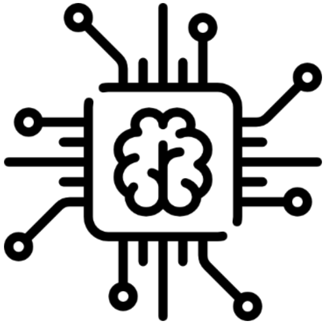
Recurrent Neural Networks

Maximilian Geißlinger / Fabian Netzler / Prof. Dr. Markus Lienkamp
(Matthias Rowold, M. Sc.)

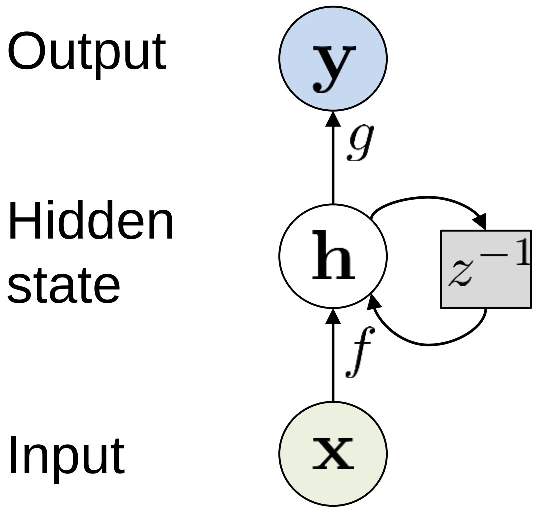
Agenda

1. Introduction
   1. Motivating example
   2. The “hidden state”
   3. Connection to dynamical systems
2. **Training RNNs**
   1. Backpropagation through time
   2. Methods
3. Vanishing and exploding gradients
4. Advanced RNN structures
5. Examples of RNNs in automotive applications





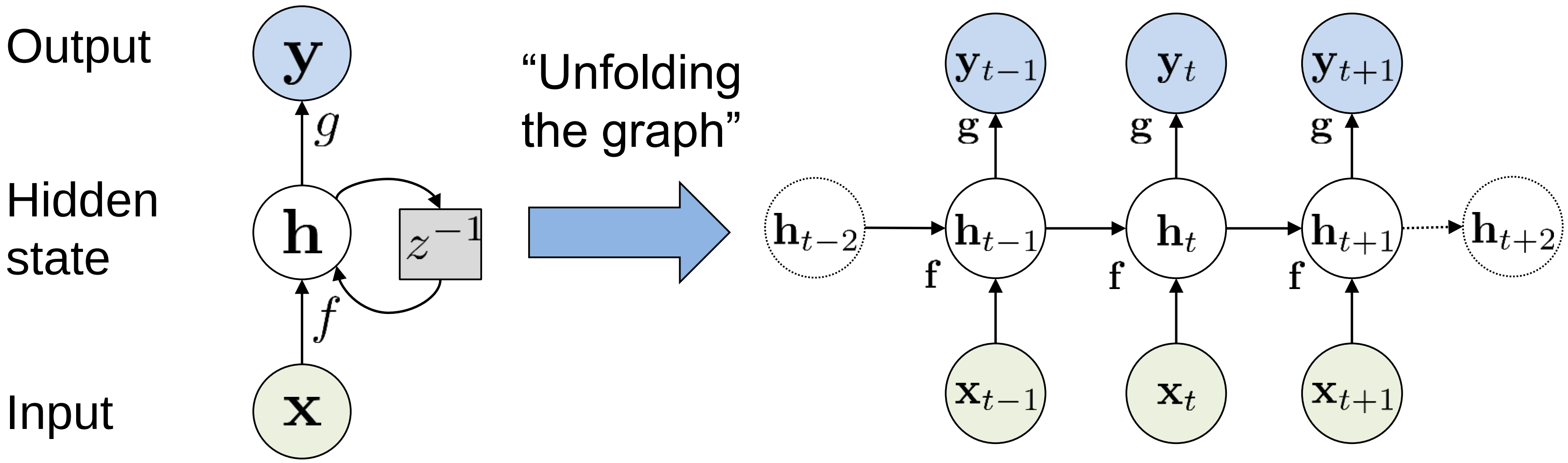
2 Training RNNs



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

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

2 Training RNNs



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

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

2 Training RNNs

A simple example:

$$\mathbf{h}_t = \text{relu}(a\mathbf{h}_{t-1} + b\mathbf{x}_t)$$

$$y_t = \sum_i h_{t,i}$$

$$\mathbf{x}_1 = [1, -1]^\top$$
$$\mathbf{x}_2 = [-1, -1]^\top$$
$$\mathbf{x}_3 = [-1, 1]^\top$$
$$\mathbf{h}_0 = [0, 0]^\top$$
$$a = b = 1$$

$$h_1 = \text{relu}\left(a\begin{bmatrix}0\\0\end{bmatrix} + b\begin{bmatrix}1\\-1\end{bmatrix}\right) = \begin{bmatrix}1\\0\end{bmatrix}$$

$$h_2 = \text{relu}\left(a\begin{bmatrix}1\\0\end{bmatrix} + b\begin{bmatrix}-1\\-1\end{bmatrix}\right) = \begin{bmatrix}0\\0\end{bmatrix}$$

$$h_3 = \text{relu}\left(a\begin{bmatrix}0\\0\end{bmatrix} + b\begin{bmatrix}-1\\1\end{bmatrix}\right) = \begin{bmatrix}0\\1\end{bmatrix}$$

$y_1 = 1$

$y_2 = 0$

$y_3 = 1$

2 Training RNNs

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

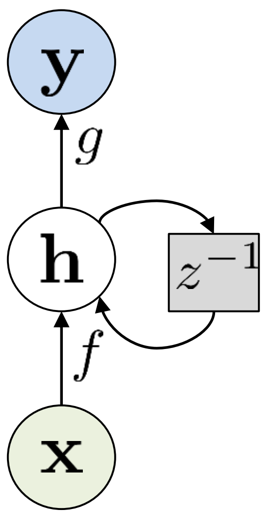
$$\mathbf{y}_t = g(\boldsymbol{h}_t)$$

Evaluate/simulate RNN (Inference):

Require: h, input data x_i for $i = 0, \dots, T$

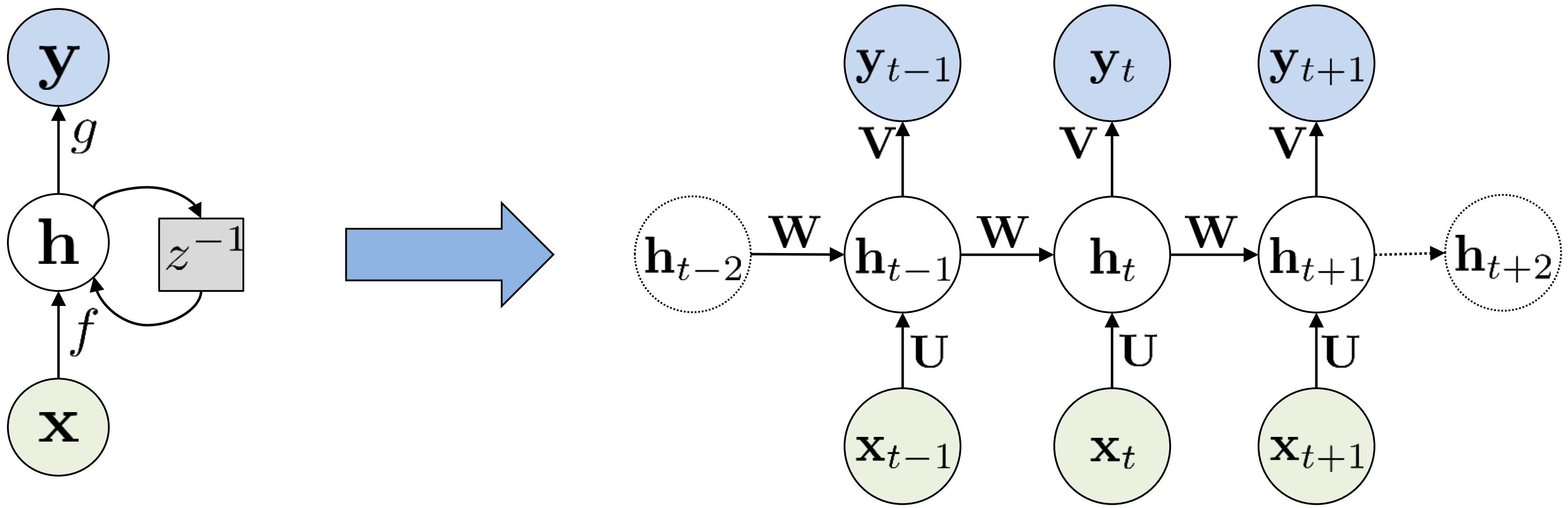
for $t = 0 : T$ do:

$h_t = f(h_{t-1}, x_t)$



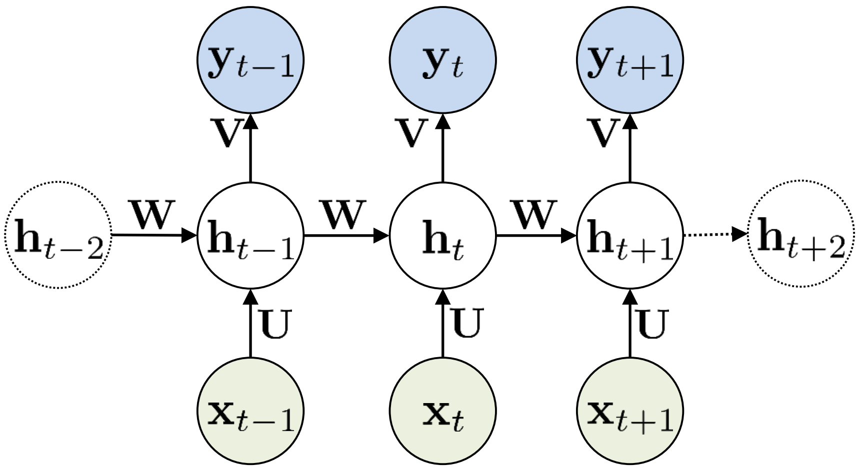
2 Training RNNs

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



2 Training RNNs

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



Given:
Input sequences

$$\mathbf{X}_i = \begin{bmatrix} \mathbf{x}_{1,i} & \dots & \mathbf{x}_{T,i} \end{bmatrix}$$

True output sequences (labels)

$$\hat{\mathbf{Y}}_i = \begin{bmatrix} \hat{\mathbf{y}}_{1,i} & \dots & \hat{\mathbf{y}}_{T,i} \end{bmatrix}$$

Supervised learning

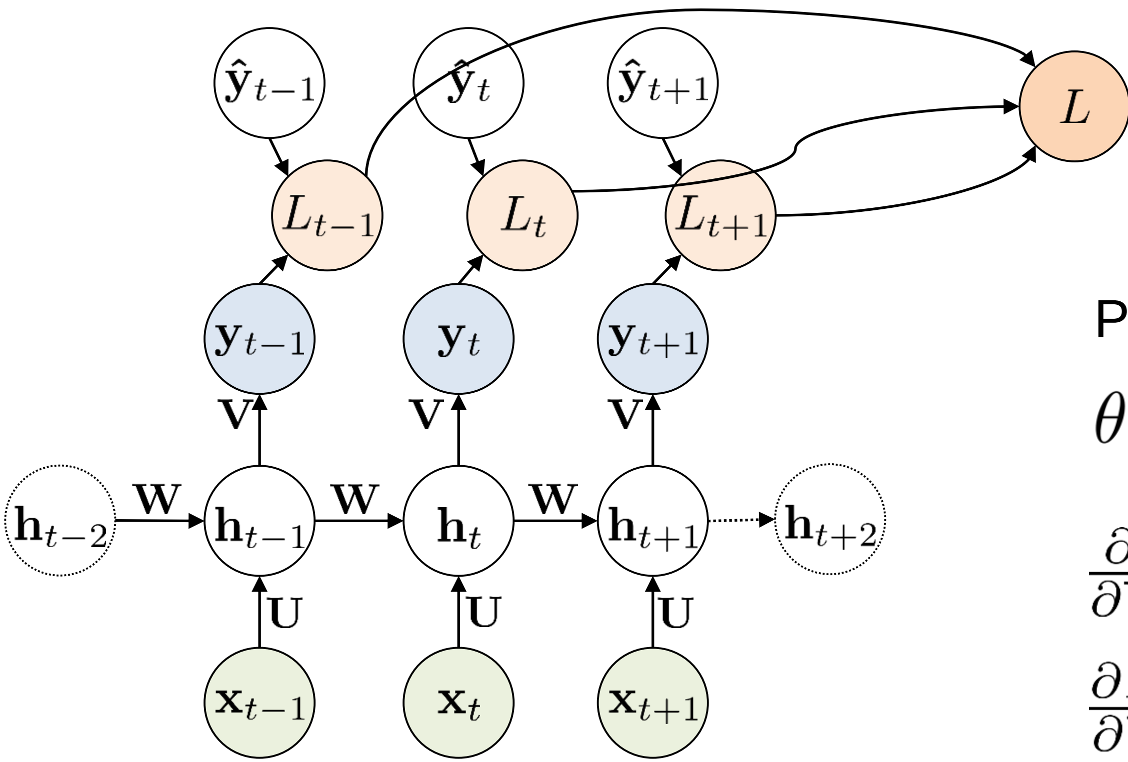
Goal:
Good parameters

$$\mathbf{W}, \mathbf{U}, \mathbf{b}, \mathbf{V}, \mathbf{v}$$

such that $\mathbf{Y}_i$ is similar to $\hat{\mathbf{Y}}_i$ for a given $\mathbf{X}_i$.

2 Training RNNs

Loss function as a sum over time-steps:



$$L = \sum_t L_t(\mathbf{y}_t, \hat{\mathbf{y}}_t)$$

Parameter update:

$$\theta = \theta - \alpha \frac{\partial L}{\partial \theta}$$

$$\frac{\partial L}{\partial \mathbf{W}} \quad \frac{\partial L}{\partial \mathbf{V}} \quad \frac{\partial L}{\partial \mathbf{U}}$$

$$\frac{\partial L}{\partial \mathbf{b}} \quad \frac{\partial L}{\partial \mathbf{v}}$$

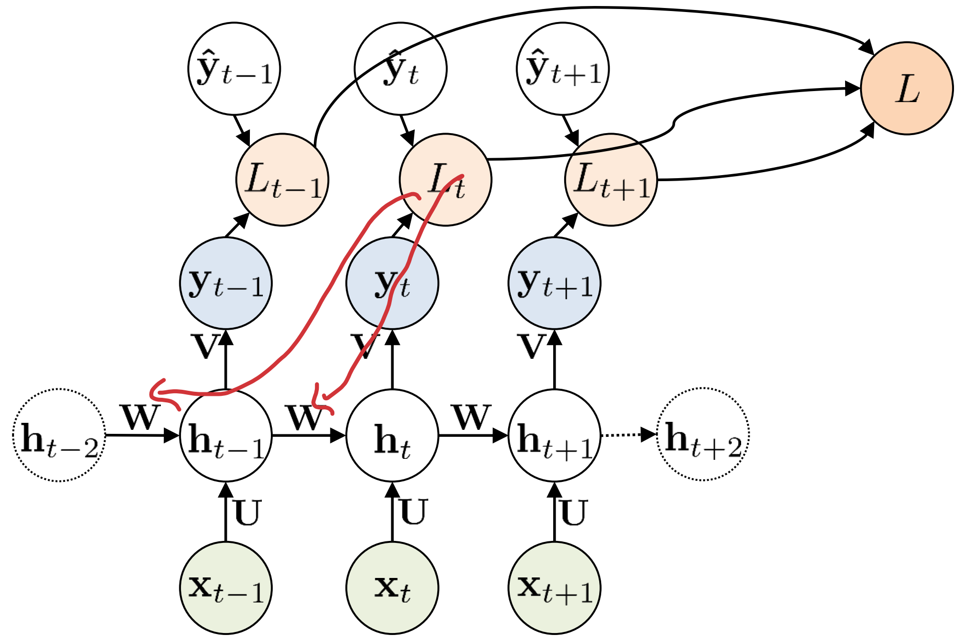
2.1 Training RNNs: Backpropagation Through Time

$$L = \sum_t L_t(\mathbf{y}_t, \hat{\mathbf{y}}_t)$$

$$\underline{\frac{\partial L_t}{\partial \mathbf{W}}} = \frac{\partial L_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{W}} +$$

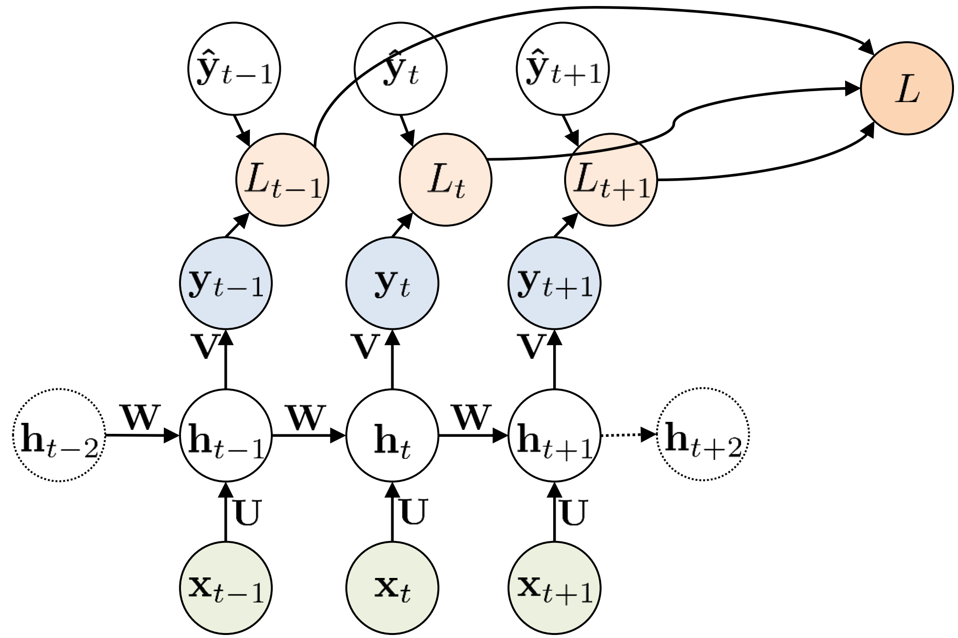
$$\frac{\partial L_t}{\partial h_t} \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial W} +$$

$$(\ldots)$$



2.1 Training RNNs: Backpropagation Through Time

$$L = \sum_t L_t(\mathbf{y}_t, \hat{\mathbf{y}}_t)$$

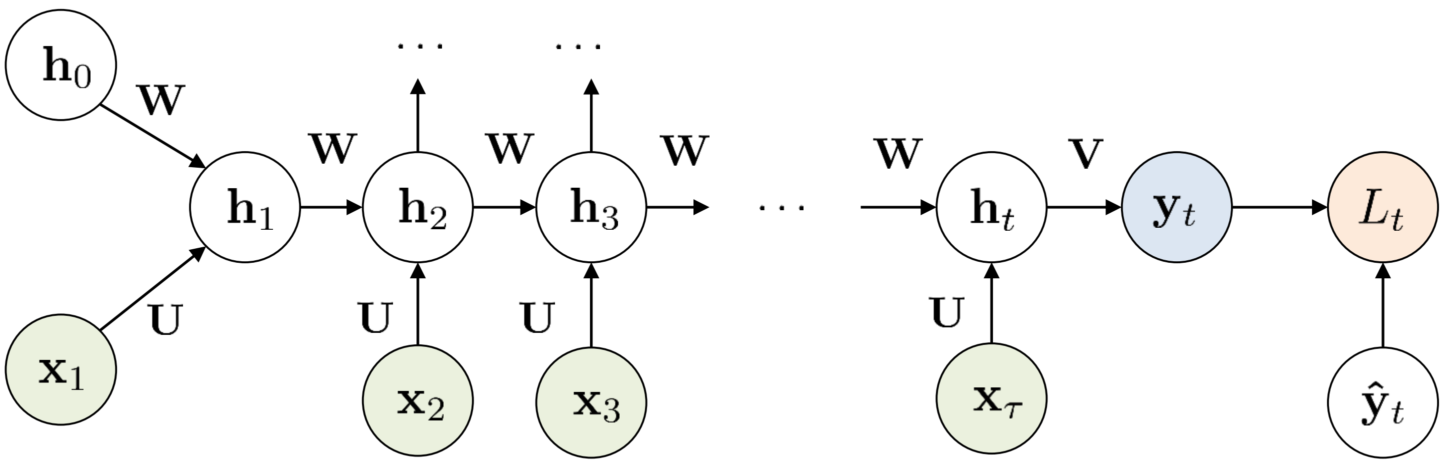


$$\begin{aligned}\frac{\partial L_t}{\partial \mathbf{W}} = & \frac{\partial L_t}{\partial \mathbf{h}_t}\frac{\partial \mathbf{h}_t}{\partial \mathbf{W}} + \\ & \frac{\partial L_t}{\partial \mathbf{h}_t}\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}}\frac{\partial \mathbf{h}_{t-1}}{\partial \mathbf{W}} + \\ & \frac{\partial L_t}{\partial \mathbf{h}_t}\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}}\frac{\partial \mathbf{h}_{t-1}}{\partial \mathbf{h}_{t-2}}\frac{\partial \mathbf{h}_{t-2}}{\partial \mathbf{W}} + \dots\end{aligned}$$

$$\frac{\partial L_t}{\partial \mathbf{W}} = \sum_{k=1}^{t} \frac{\partial L_t}{\partial \mathbf{h}_t}\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k}\frac{\partial \mathbf{h}_k}{\partial \mathbf{W}} \quad \text{with} \quad \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} = \prod_{i=k+1}^{t} \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}}$$

2.1 Training RNNs: Backpropagation Through Time

A RNN is similar to a very deep NN with as many layers as time steps.
The weight matrix $\mathbf{W}$ is the same for each layer!



$$\frac{\partial L_t}{\partial \mathbf{W}} = \sum_{k=1}^{t} \frac{\partial L_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \frac{\partial \mathbf{h}_k}{\partial \mathbf{W}} \quad \text{with} \quad \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} = \prod_{i=k+1}^{t} \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}}$$

Maximilian Geißlinger / Fabian Netzler / Prof. Dr. Markus Lienkamp
(Matthias Rowold, M. Sc.)

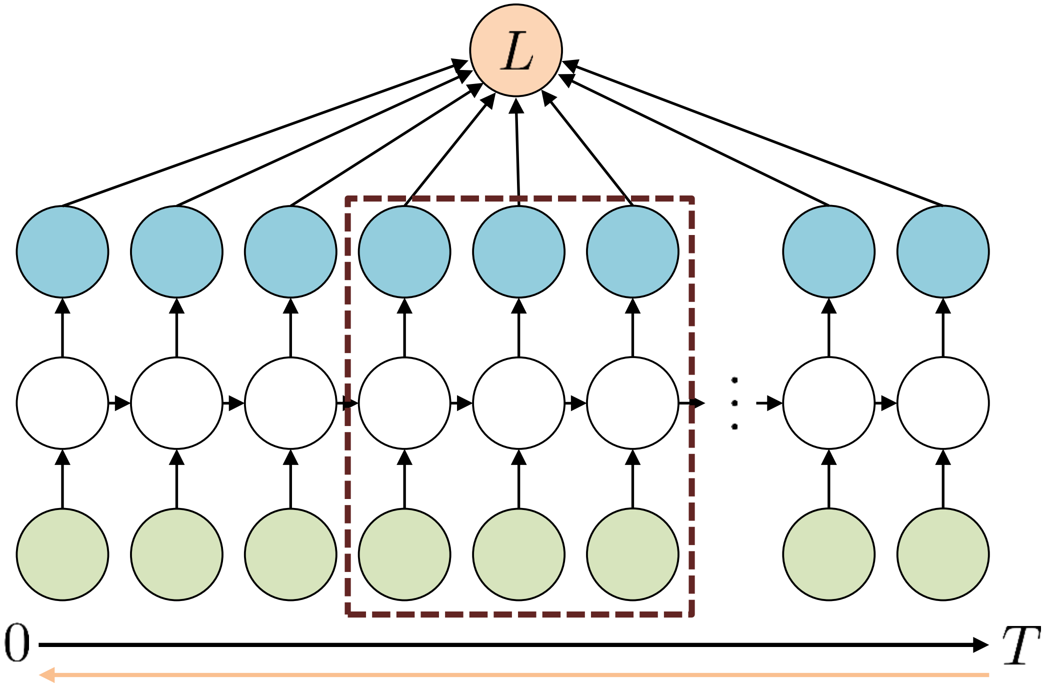
Agenda

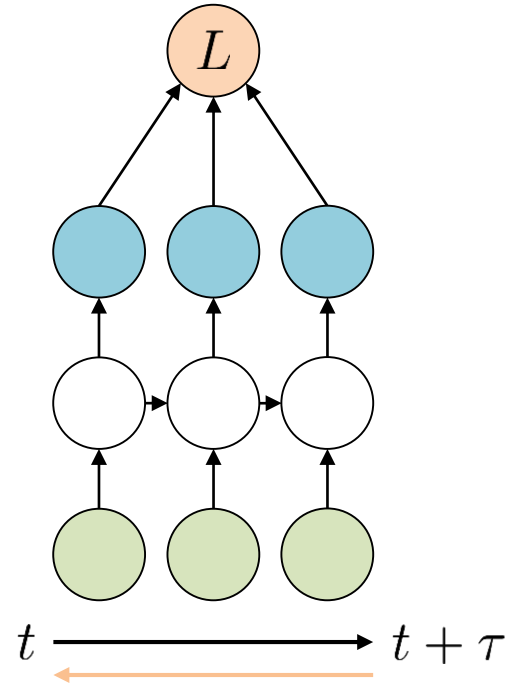
1. Introduction
   1. Motivating example
   2. The “hidden state”
   3. Connection to dynamical systems
2. Training RNNs
   1. Backpropagation through time
   2. **Methods**
3. Vanishing and exploding gradients
4. Advanced RNN structures
5. Examples of RNNs in automotive applications

2.2 Training RNNs: Methods

Truncated backpropagation through time

Backpropagation applied on the unfolded graph of a chunk of the whole sequence.



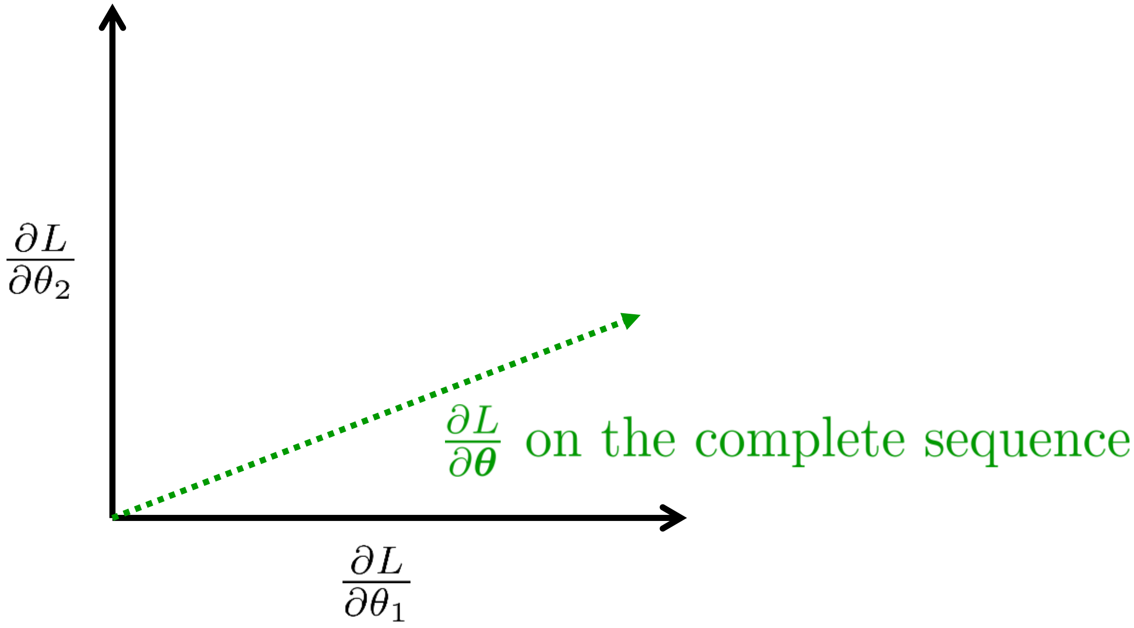


2.2 Training RNNs: Methods

Truncated backpropagation through time

Truncated backpropagation through time is biased!

Unbiased versions e.g. in [1].



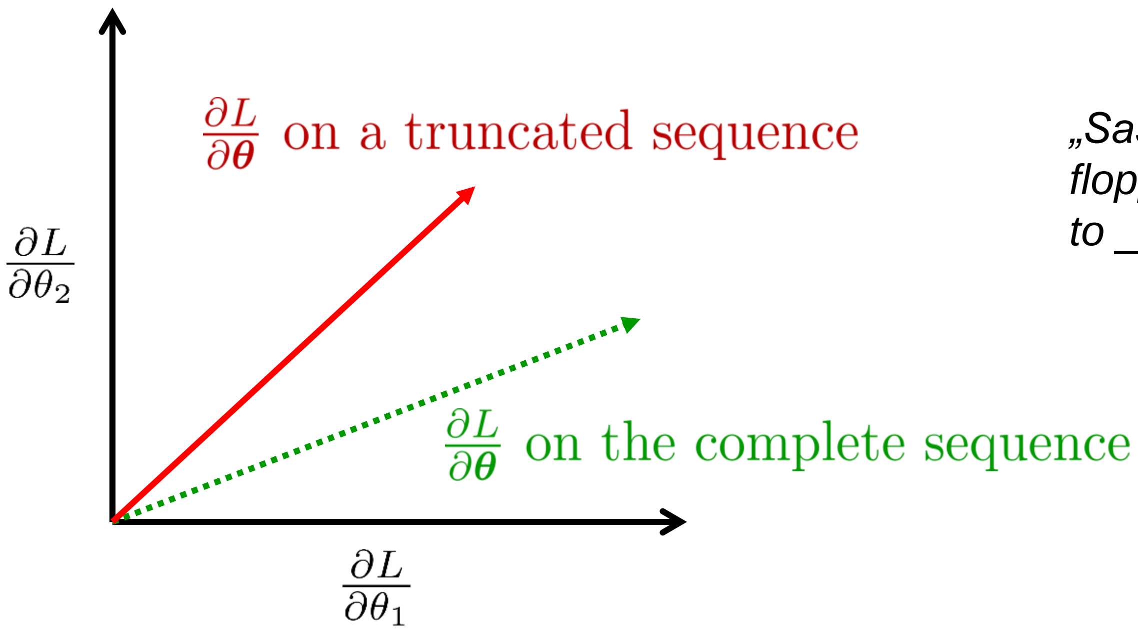
2.2 Training RNNs: Methods

Truncated backpropagation through time

Truncated backpropagation through time is biased.

Unbiased versions e.g. in [1].

*„The **summer** is ___“*

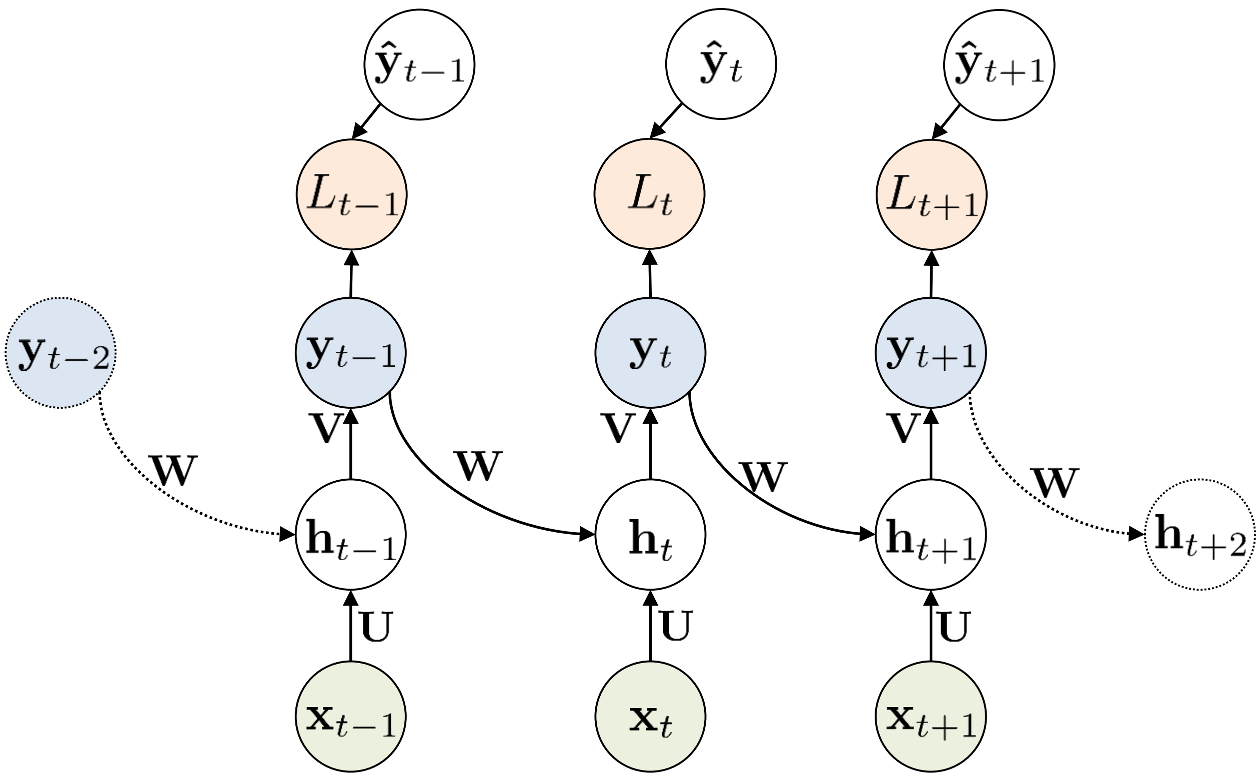


*„Saskia **grabbed a book**, … she flopped on the couch and began to ___“*

2.2 Training RNNs: Methods

Teacher forcing

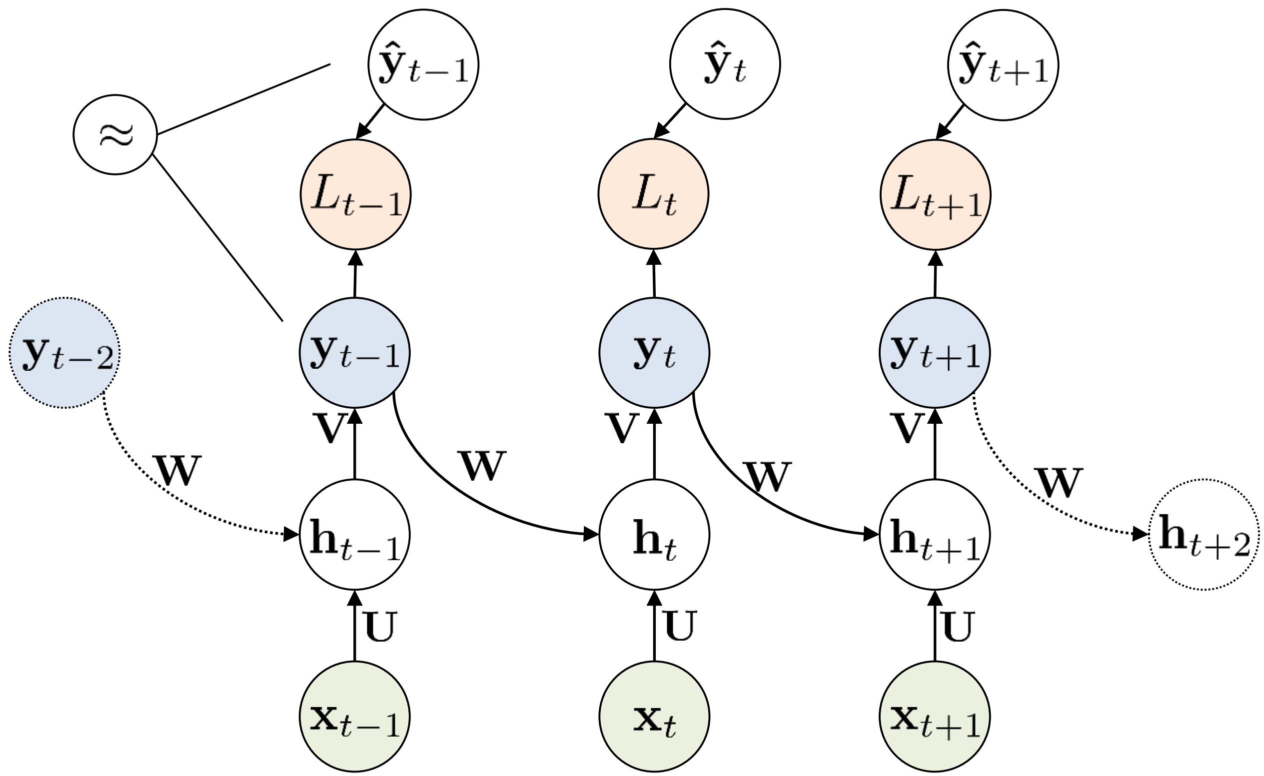
RNNs with a feedback of the output.



2.2 Training RNNs: Methods

Teacher forcing

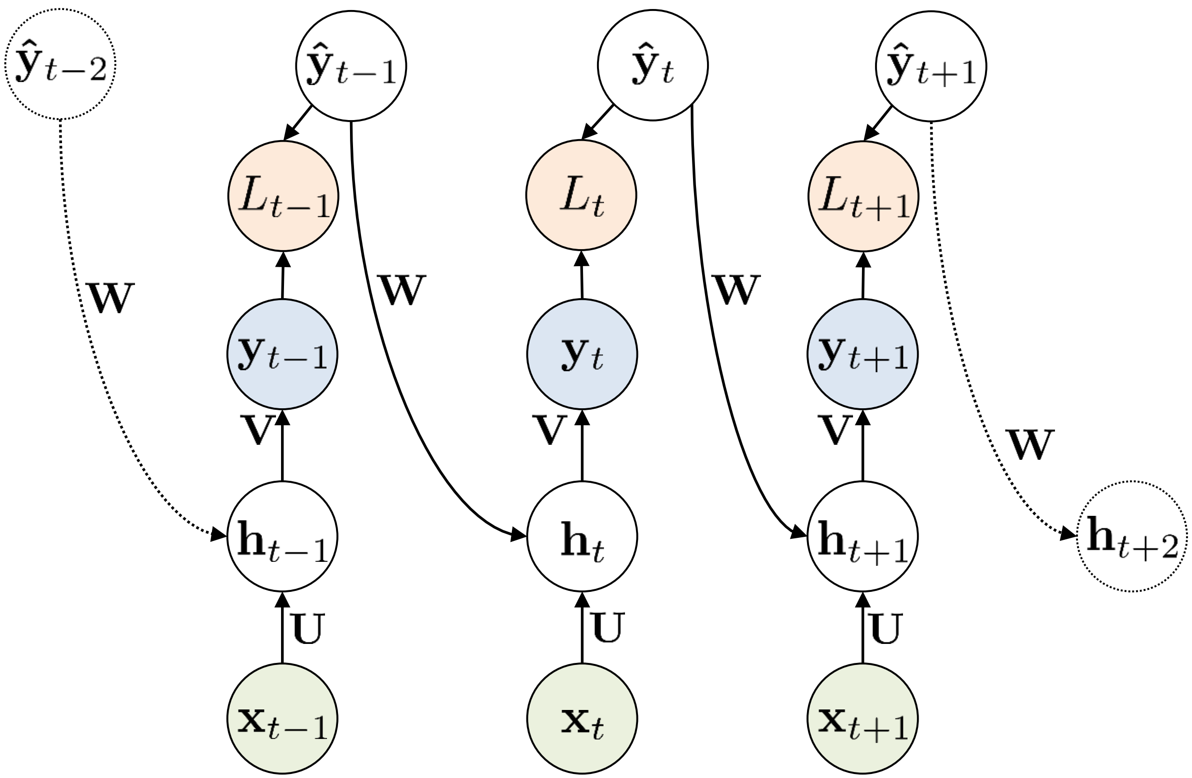
RNNs with a feedback of the output.



2.2 Training RNNs: Methods

Teacher forcing

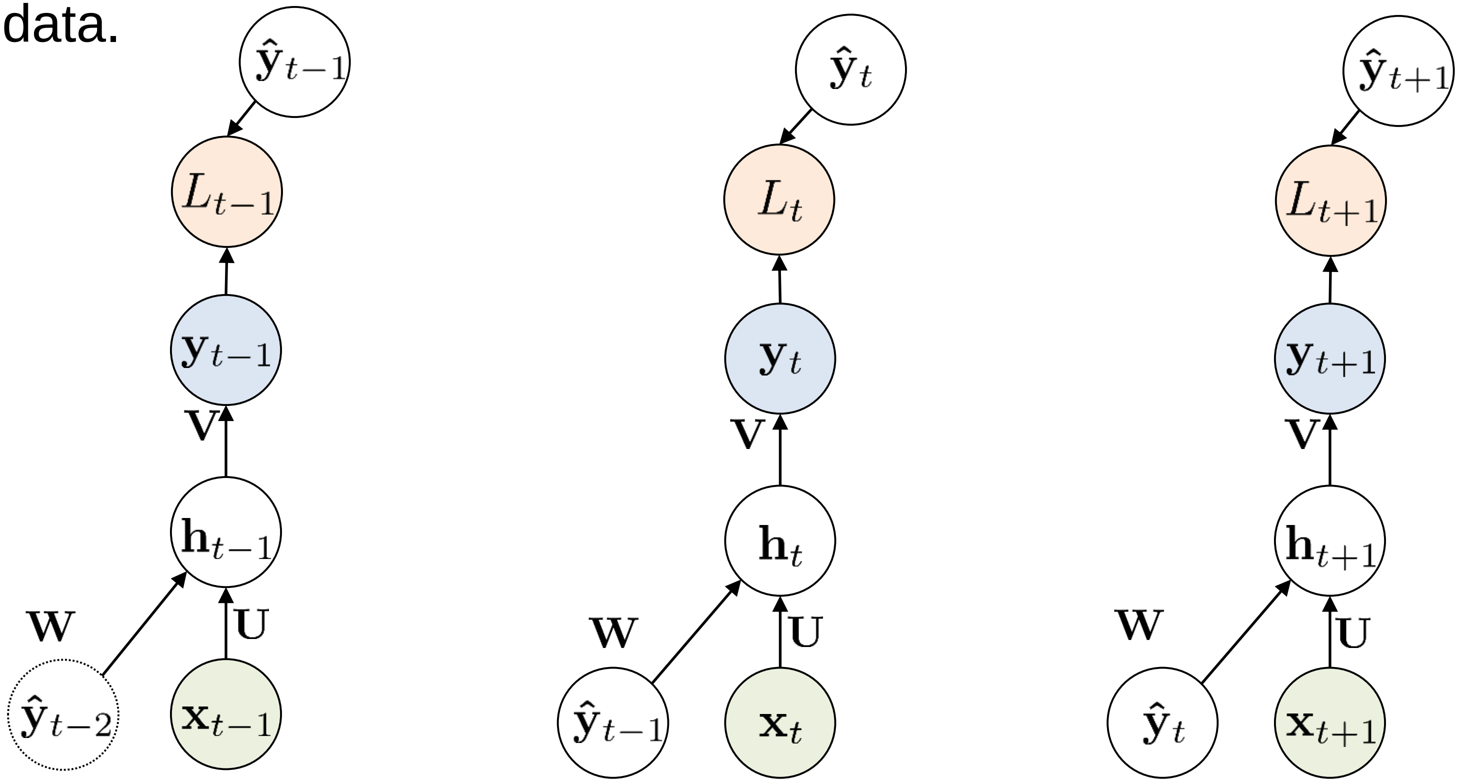
RNNs with a feedback of the output.



2.2 Learning Methods

Teacher forcing

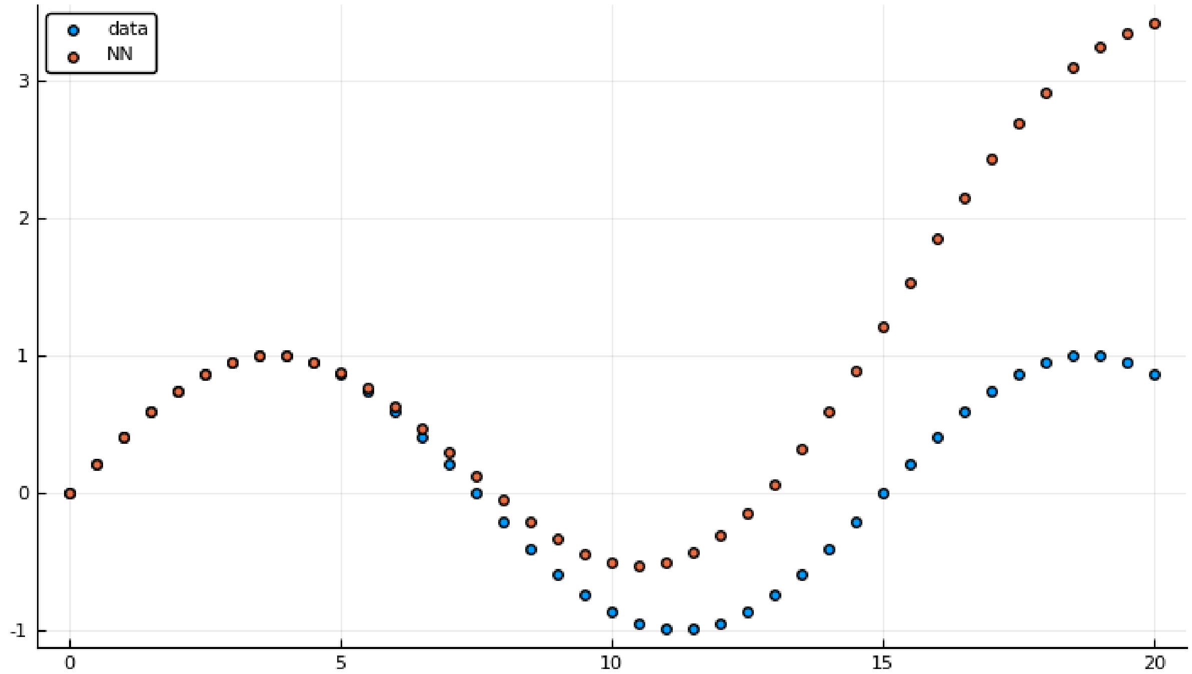
RNNs with a feedback of the output.
We can split the computation graph by using the target values from the data.



2.2 Training RNNs: Methods

Teacher forcing

Problem with accumulating errors when predicting longer sequences.

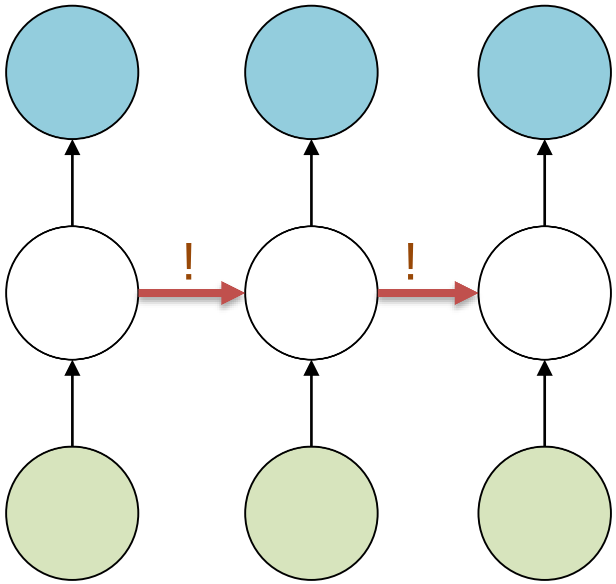


2.2 Training RNNs: Methods

Regularization using dropout

Regularization is problematic when used on recurrent weights.

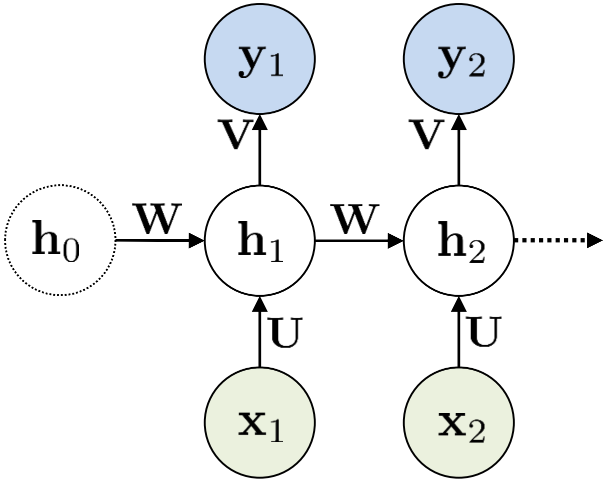
Apply dropouts on non-recurrent weights only [6].



2.2 Training RNNs: Methods

State initialization

How do we choose $\mathbf{h}_0$?



- Initialize as zero [2]
- Noisy zero mean [3]
- Treat it as a parameter that is to be learned [4]
- Initialize using a second neural network [5]

2. Training RNNs: Wrap Up

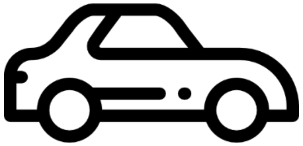
- Backpropagation applied on the unfolded graph of a RNN is called **backpropagation through time.**
- Training a RNN is like training a very deep neural network.
- Training with long sequences requires much computational effort and can be problematic.
  - Sequences can be truncated to mitigate the problem of too long sequences. However, the gradients can be biased.
  - Teacher forcing decouples the time-steps for the gradient calculation but is limited to a certain structure of RNNs.
- Dropouts can be applied only on non-recurrent weights.

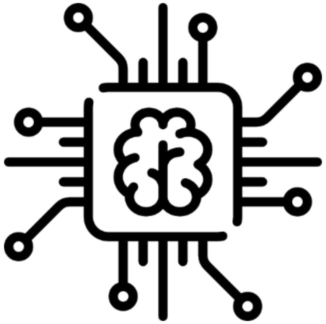
Recurrent Neural Networks

Maximilian Geißlinger / Fabian Netzler / Prof. Dr. Markus Lienkamp
(Matthias Rowold, M. Sc.)

Agenda

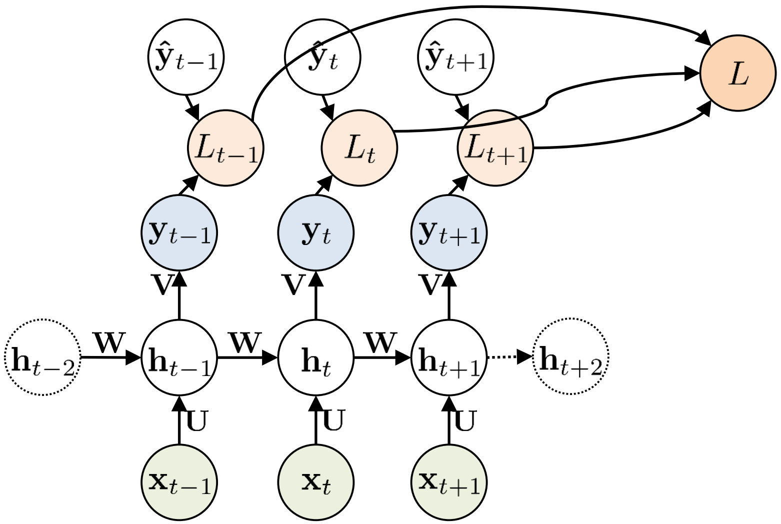
1. Introduction
   1. Motivating example
   2. The “hidden state”
   3. Connection to dynamical systems
2. Training RNNs
   1. Backpropagation through time
   2. Methods
3. **Vanishing and exploding gradients**
4. Advanced RNN structures
5. Examples of RNNs in automotive applications





3.0 Vanishing and Exploding Gradients

Recap of BPTT:



$$L = \sum_{t=1}^{T} L_t = \sum_{t=1}^{T} L(\mathbf{y}_t, \hat{\mathbf{y}}_t)$$

Parameter update:

$$\theta = \theta - \alpha \frac{\partial L}{\partial \theta}$$

$$\frac{\partial L_t}{\partial \mathbf{W}} = \sum_{k=1}^{t} \frac{\partial L_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \frac{\partial \mathbf{h}_k}{\partial \mathbf{W}} \quad \text{with} \quad \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} = \prod_{i=k+1}^{t} \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}}$$

3.0 Vanishing and Exploding Gradients

Suppose we have a recurrent function:

$$h_{t+1} = a \cdot h_t$$

$$h_{t+k} = a^k \cdot h_t$$

$$\frac{\partial h_{t+k}}{\partial h_t} = a^k = \prod_{i=t+1}^{t+k} \frac{\partial h_i}{\partial h_{i-1}}$$

For long sequences:

$$\lim_{k \to \infty} \frac{\partial h_{t+k}}{\partial h_t} = \begin{cases} 1 & \text{if } a = 1 \\ \text{l.m.e.} & \text{if } a = -1 \\ \infty & \text{if } a > 1 \\ -\infty & \text{if } a < -1 \\ 0 & \text{if } |a| < 0 \end{cases}$$

3.0 Vanishing and Exploding Gradients

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

$$\mathbf{y}_t = \mathbf{V}\mathbf{h}_t + \mathbf{v}$$

$$\frac{\partial L_t}{\partial \mathbf{W}} = \sum_{k=1}^{t} \frac{\partial L_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \frac{\partial \mathbf{h}_k}{\partial \mathbf{W}} \quad \text{with} \quad \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} = \prod_{i=k+1}^{t} \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}}$$

3.0 Vanishing and Exploding Gradients

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

$$\mathbf{y}_t = \mathbf{V}\mathbf{h}_t + \mathbf{v}$$

$$\frac{\partial L_t}{\partial \mathbf{W}} = \sum_{k=1}^{t} \frac{\partial L_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \frac{\partial \mathbf{h}_k}{\partial \mathbf{W}} \quad \text{with} \quad \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} = \prod_{i=k+1}^{t} \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}}$$

$$\frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} = \operatorname{diag}(\mathbf{1} - \mathbf{h}_{i-1} \odot \mathbf{h}_{i-1})\mathbf{W}$$

$$\left\| \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} \right\| \leq \|\operatorname{diag}(\mathbf{1} - \mathbf{h}_{i-1} \odot \mathbf{h}_{i-1})\| \, \|\mathbf{W}\|$$

3.0 Vanishing and Exploding Gradients

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

$$\mathbf{y}_t = \mathbf{V}\mathbf{h}_t + \mathbf{v}$$

$$\frac{\partial L_t}{\partial \mathbf{W}} = \sum_{k=1}^{t} \frac{\partial L_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \frac{\partial \mathbf{h}_k}{\partial \mathbf{W}} \quad \text{with} \quad \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} = \prod_{i=k+1}^{t} \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}}$$

$$\frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} = \operatorname{diag}(\mathbf{1} - \mathbf{h}_{i-1} \odot \mathbf{h}_{i-1})\mathbf{W}$$

$$\left\| \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} \right\| \leq \|\operatorname{diag}(\mathbf{1} - \mathbf{h}_{i-1} \odot \mathbf{h}_{i-1})\| \, \|\mathbf{W}\|$$

3.0 Vanishing and Exploding Gradients

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

$$\mathbf{y}_t = \mathbf{V}\mathbf{h}_t + \mathbf{v}$$

$$\frac{\partial L_t}{\partial \mathbf{W}} = \sum_{k=1}^{t} \frac{\partial L_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \frac{\partial \mathbf{h}_k}{\partial \mathbf{W}} \quad \text{with} \quad \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} = \prod_{i=k+1}^{t} \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}}$$

$$\frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} = \mathrm{diag}(\mathbf{1} - \mathbf{h}_{i-1} \odot \mathbf{h}_{i-1})\mathbf{W}$$

$$\left\| \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} \right\| \leq \left\| \mathrm{diag}(\mathbf{1} - \mathbf{h}_{i-1} \odot \mathbf{h}_{i-1}) \right\| \left\| \mathbf{W} \right\| \ \leq 1 \left\| \mathbf{W} \right\|$$

$$\left\| \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \right\| = \left\| \prod_{i=k+1}^{t} \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} \right\| \leq \left\| \mathbf{W} \right\|^{t-k}$$

[7], [8]

3.0 Vanishing and Exploding Gradients

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

We want: $$\left\| \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \right\| = \left\| \prod_{i=k+1}^{t} \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} \right\| \leq \|\mathbf{W}\|^{t-k} \approx 1$$

Initializing $\mathbf{W}$ with a good distribution helps at the beginning of the training:

3.0 Vanishing and Exploding Gradients

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

We want: $$\left\| \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \right\| = \left\| \prod_{i=k+1}^{t} \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} \right\| \leq \|\mathbf{W}\|^{t-k} \approx 1$$

Initializing $\mathbf{W}$ with a good distribution helps at the beginning of the training:

Xavier initialization [9]:

$$\mathbf{W} \in \mathbb{R}^{n \times n} \qquad W_{i,j} \sim \mathcal{N}\left(0, \frac{1}{n}\right)$$

3.0 Vanishing and Exploding Gradients

Gradient clipping

If the gradient is too large, rescale it.
→ Use the direction but not the magnitude.

$$
\begin{aligned}
&\mathbf{g} \leftarrow \frac{\partial L}{\partial \theta} \\
&\textbf{if } \|\mathbf{g}\| \geq \nu \textbf{ then} \\
&\qquad \mathbf{g} \leftarrow \frac{\nu}{\|\mathbf{g}\|}\mathbf{g} \\
&\textbf{end if}
\end{aligned}
$$

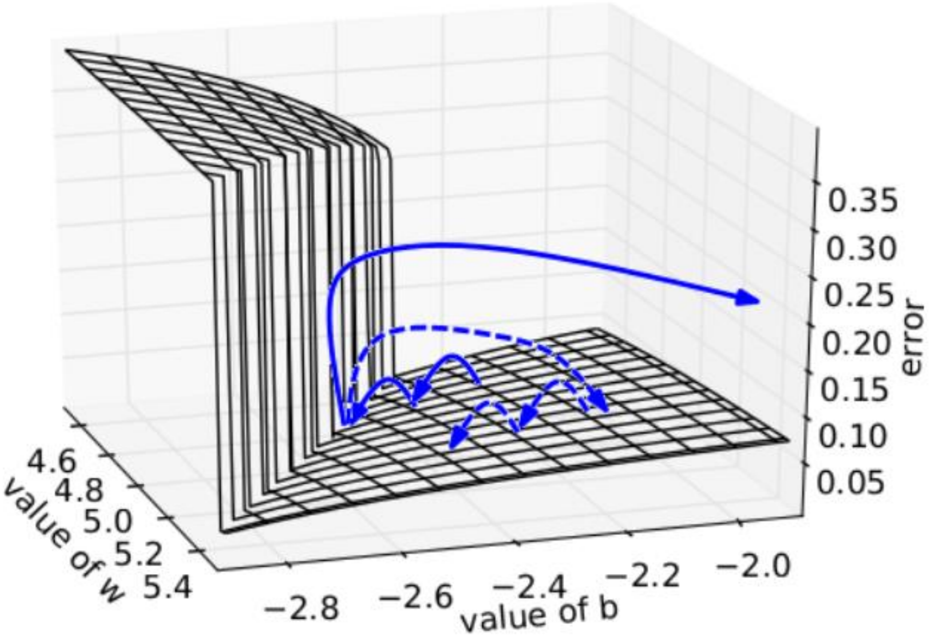
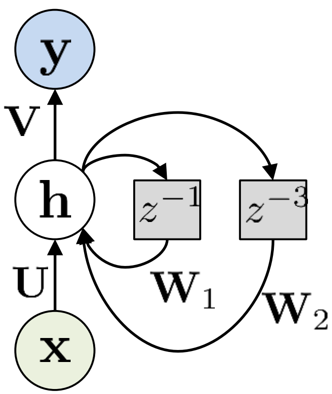
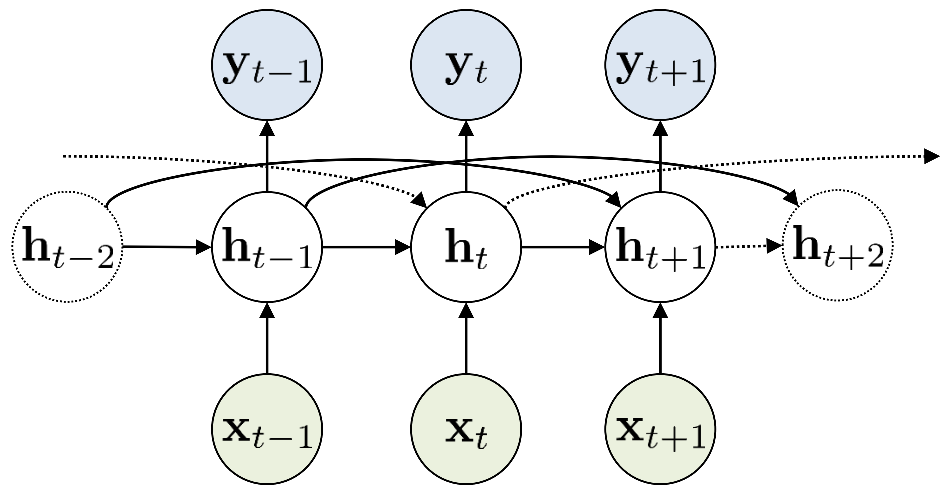


Image from [7]

3.0 Vanishing and Exploding Gradients

Vanishing gradient: skip connections



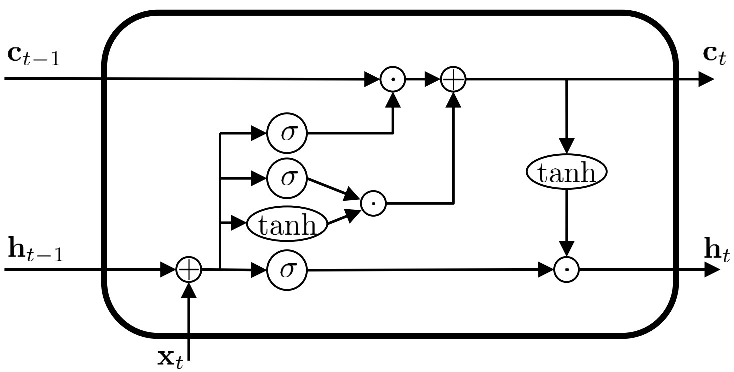


3.0 Vanishing and Exploding Gradients

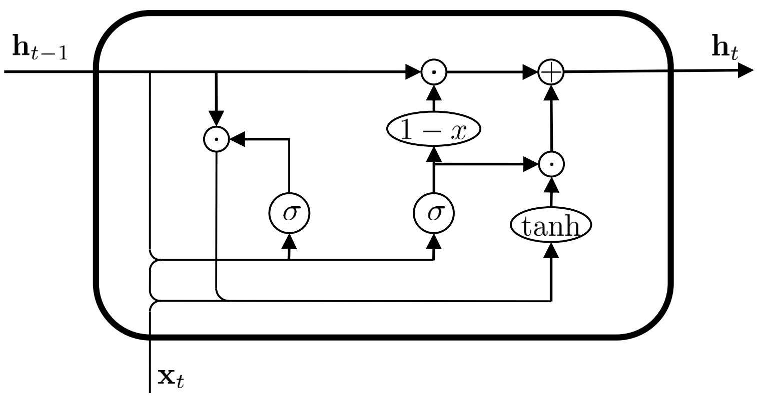
Vanishing gradient: gated networks

Use **different RNN structures** with „gates“

LSTM



GRU



3. Wrap Up

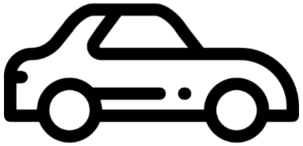
- Reusing the same weights can lead to **very large or very small gradients** that can slow down training.
- **Large gradients** should be **diminished in some way**, e.g. by rescaling.
- **Small gradients** can be tackled by using **inputs from multiple steps in the past.**
- Other RNN structures can mitigate the problem of vanishing gradients.
- A good **initialization of the recurrent** weights can avoid small or large gradients at the beginning of the training.

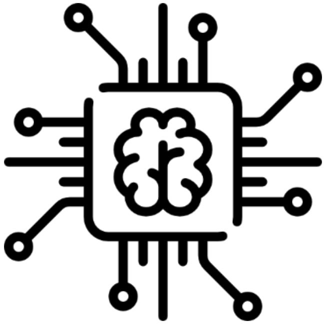
Recurrent Neural Networks

Maximilian Geißlinger / Fabian Netzler / Prof. Dr. Markus Lienkamp
(Matthias Rowold, M. Sc.)

Agenda

1. Introduction
   1. Motivating example
   2. The “hidden state”
   3. Connection to dynamical systems
2. Training RNNs
   1. Backpropagation through time
   2. Methods
3. Vanishing and exploding gradients
4. **Advanced RNN structures**
5. Examples of RNNs in automotive applications

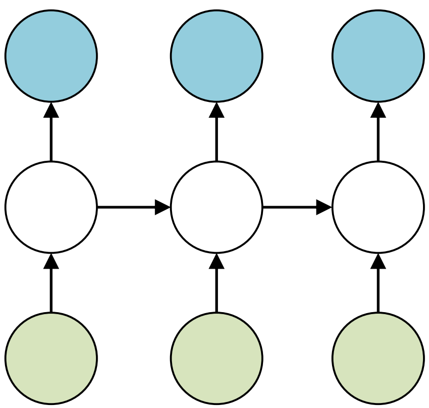




4. Advanced RNN Structures

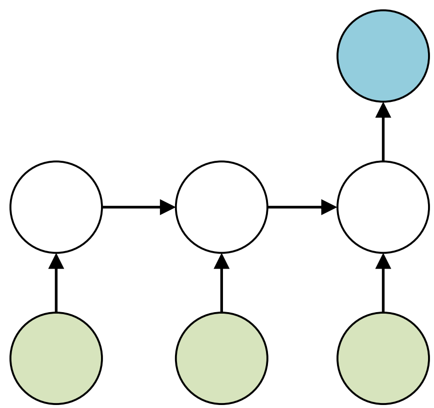
Input – output relations

Many to many



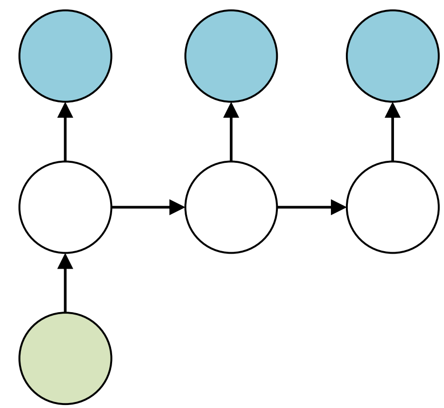
E.g. Approximate the dynamics of a physical system

Many to one



E.g. classify a video

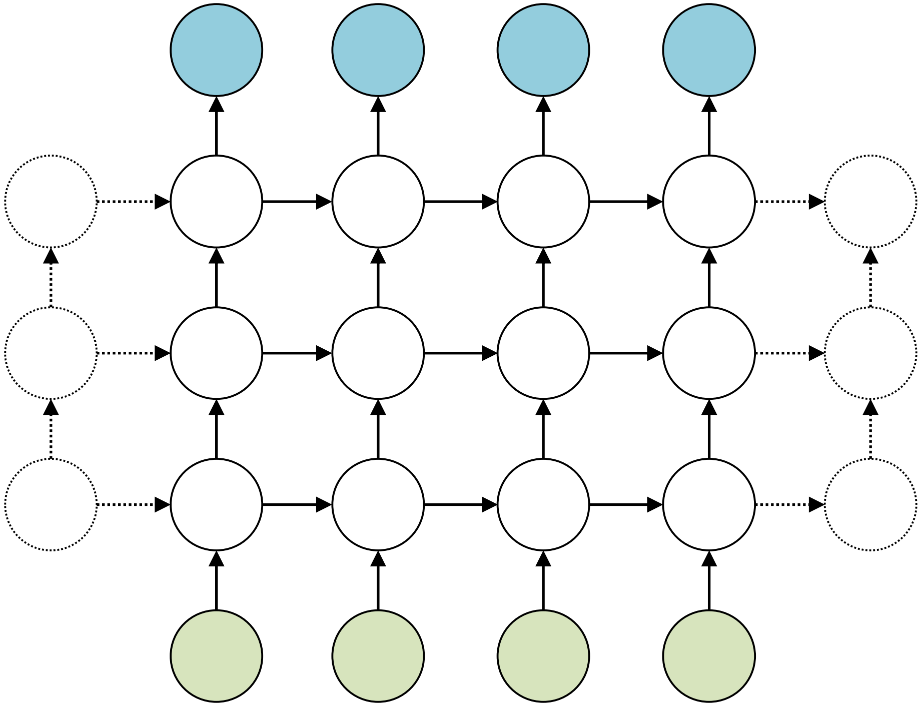
One to many



E.g. describe an image with a sentence

4. Advanced RNN Structures

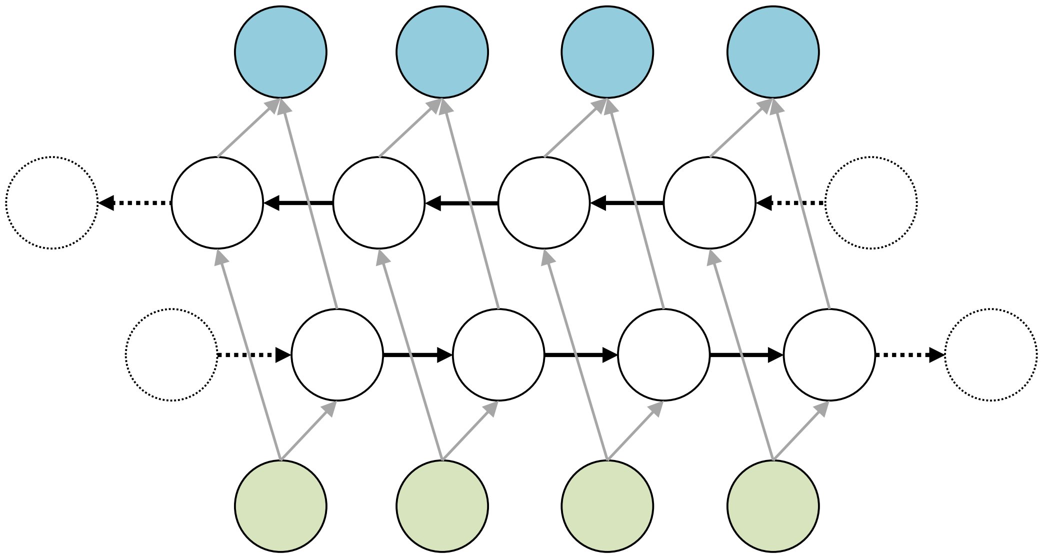
Multilayer RNN



Short analysis of the influence of multiple layers in [10].

4. Advanced RNN Structures

Bidirectional RNN



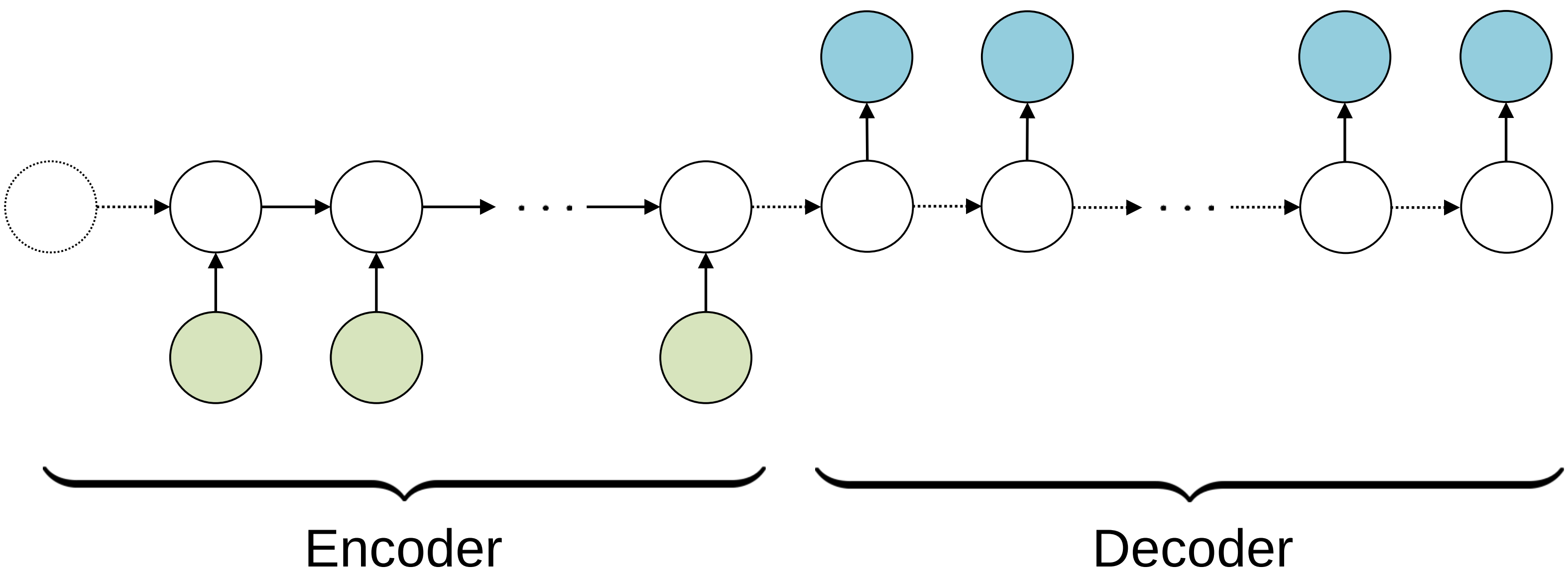
Including future information can be helpful. E.g. handwriting recognition.

4. Advanced RNN Structures

Sequence to sequence

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



4. Advanced RNN Structures

Long short-term memory (LSTM)

Idea: Do not update the whole hidden state each time-step.

Protect the state from being overwritten by useless information.

Be selective in:

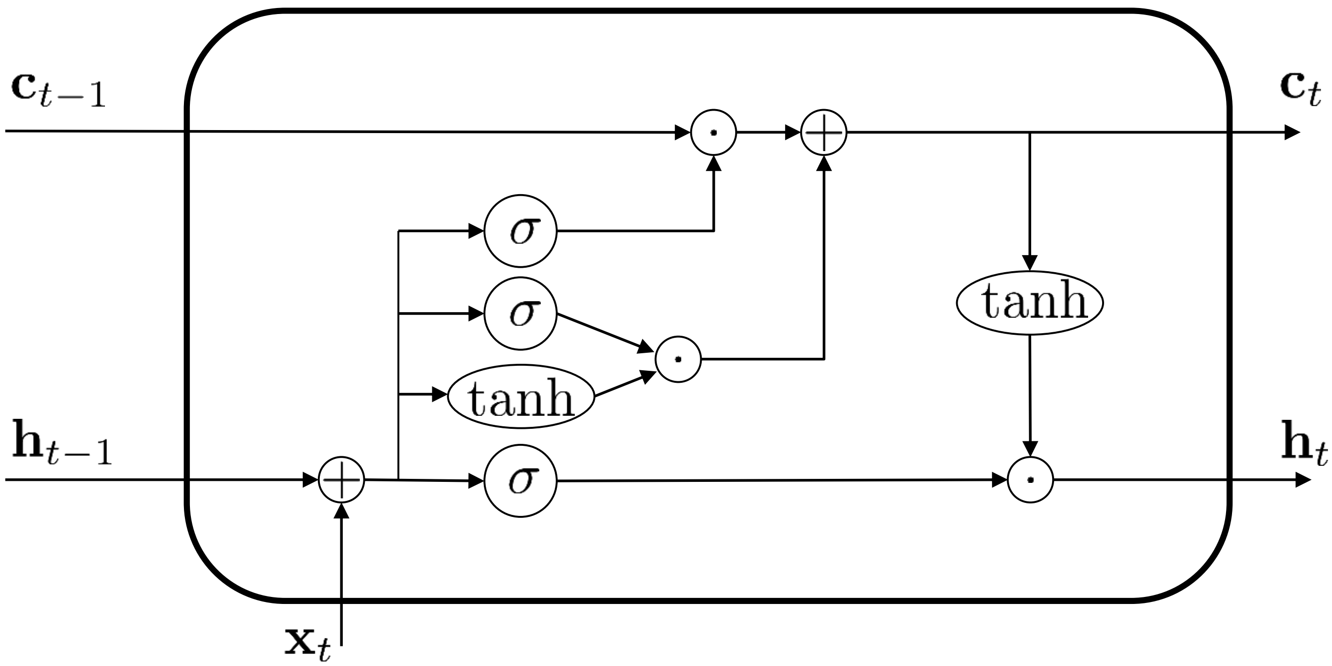
- What to forget (forget gate) $\mathbf{f}_t = \sigma(\mathbf{W}_f \mathbf{h}_{t-1} + \mathbf{U}_f \mathbf{x}_t + \mathbf{b}_f)$
- What to write (input gate) $\mathbf{i}_t = \sigma(\mathbf{W}_i \mathbf{h}_{t-1} + \mathbf{U}_i \mathbf{x}_t + \mathbf{b}_i)$
- What to output (output gate) $\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{h}_{t-1} + \mathbf{U}_o \mathbf{x}_t + \mathbf{b}_o)$

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

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(c_t)$$

4. Advanced RNN Structures

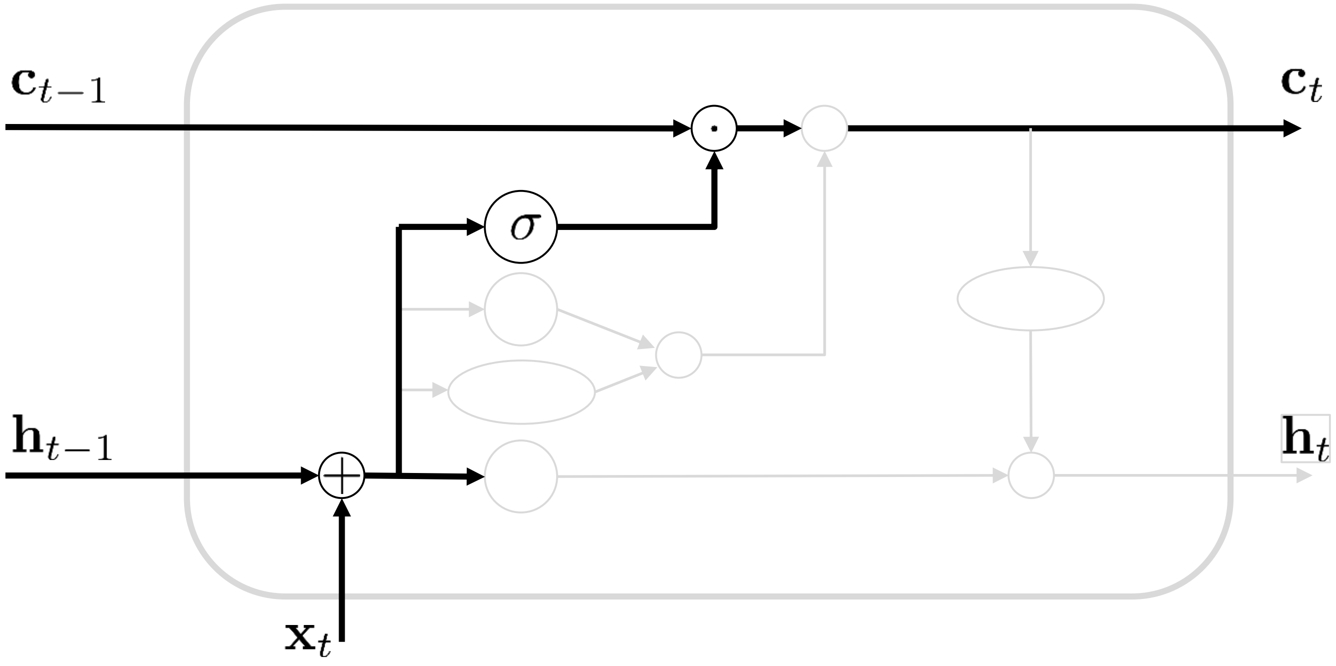
Long short-term memory (LSTM)



4. Advanced RNN Structures

LSTM, forget

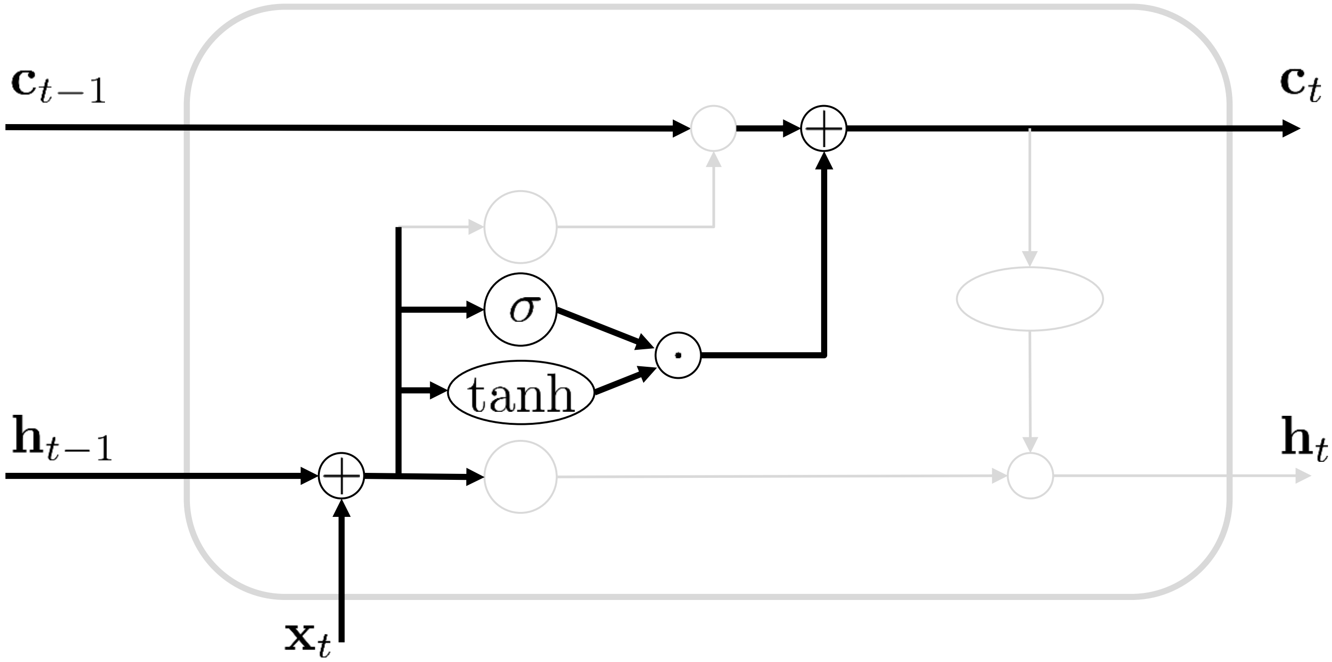
$$\mathbf{f}_t = \sigma(\mathbf{W}_f \mathbf{h}_{t-1} + \mathbf{U}_f \mathbf{x}_t + \mathbf{b}_f)$$



4. Advanced RNN Structures

LSTM, input

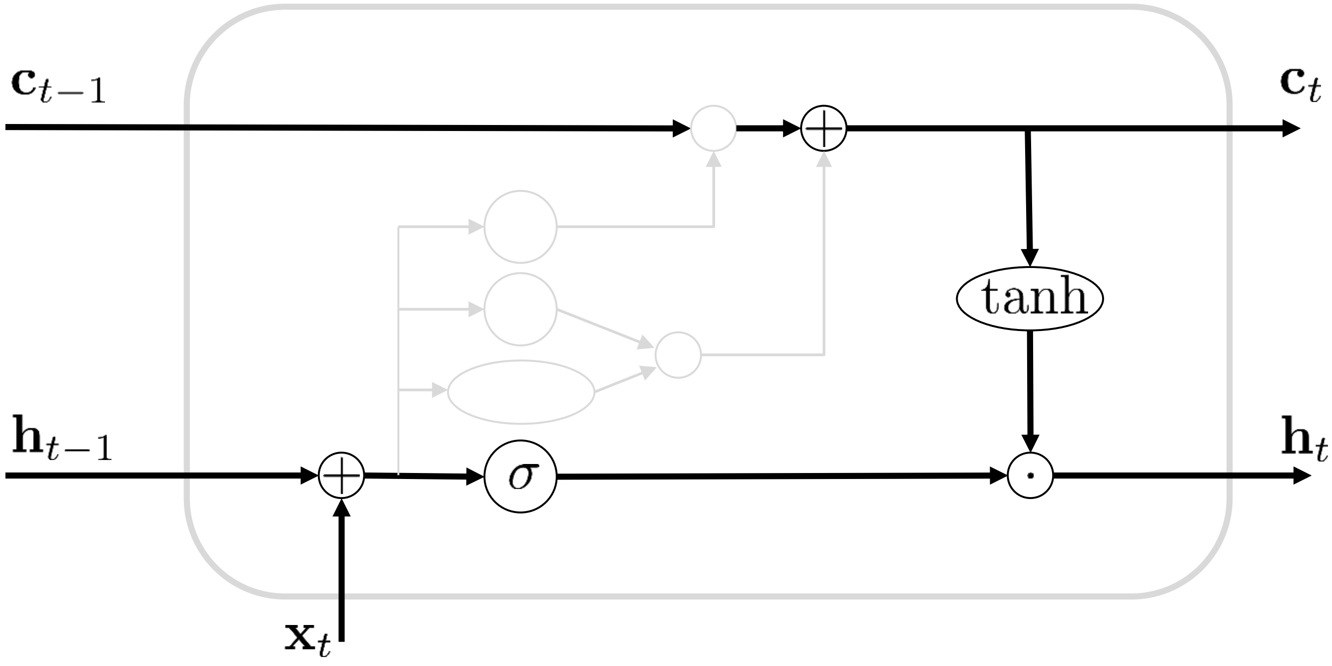
$$\mathbf{i}_t = \sigma(\mathbf{W}_i \mathbf{h}_{t-1} + \mathbf{U}_i \mathbf{x}_t + \mathbf{b}_i)$$



4. Advanced RNN Structures

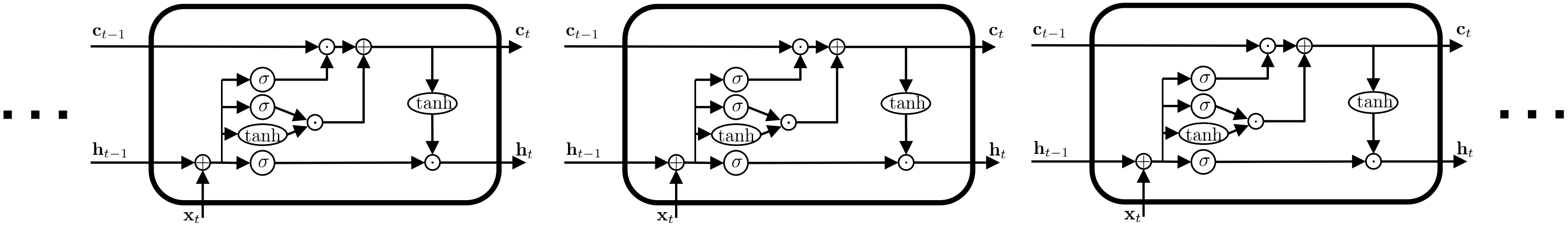
LSTM, output

$$\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{h}_{t-1} + \mathbf{U}_o \mathbf{x}_t + \mathbf{b}_o)$$



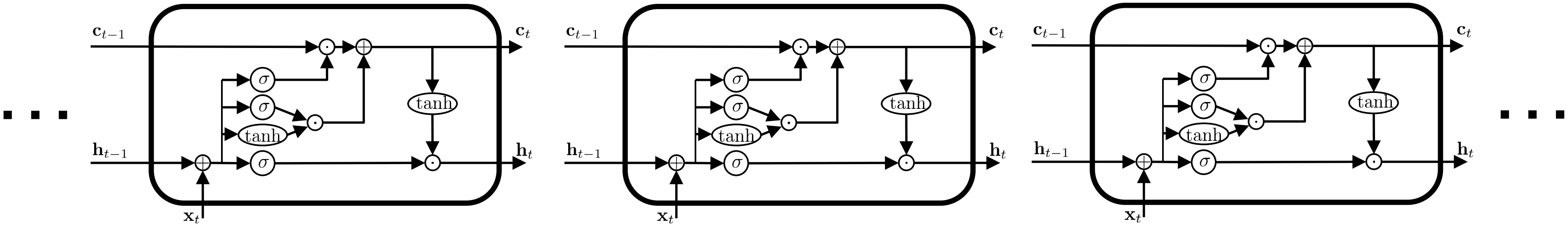
4. Advanced RNN Structures

Long short-term memory (LSTM)



4. Advanced RNN Structures

Long short-term memory (LSTM)



DIPLOMARBEIT
IM FACH INFORMATIK

Untersuchungen zu dynamischen neuronalen Netzen

Josef Hochreiter Institut für Informatik
Technische Universität München
Arcisstr. 21, 8000 München 2, Germany
hochreit@kiss.informatik.tu-muenchen.de

Aufgabensteller: Professor Dr. W. Brauer

Betreuer: Dr. Jürgen Schmidhuber

15 Juni 1991

[11]

LONG SHORT-TERM MEMORY

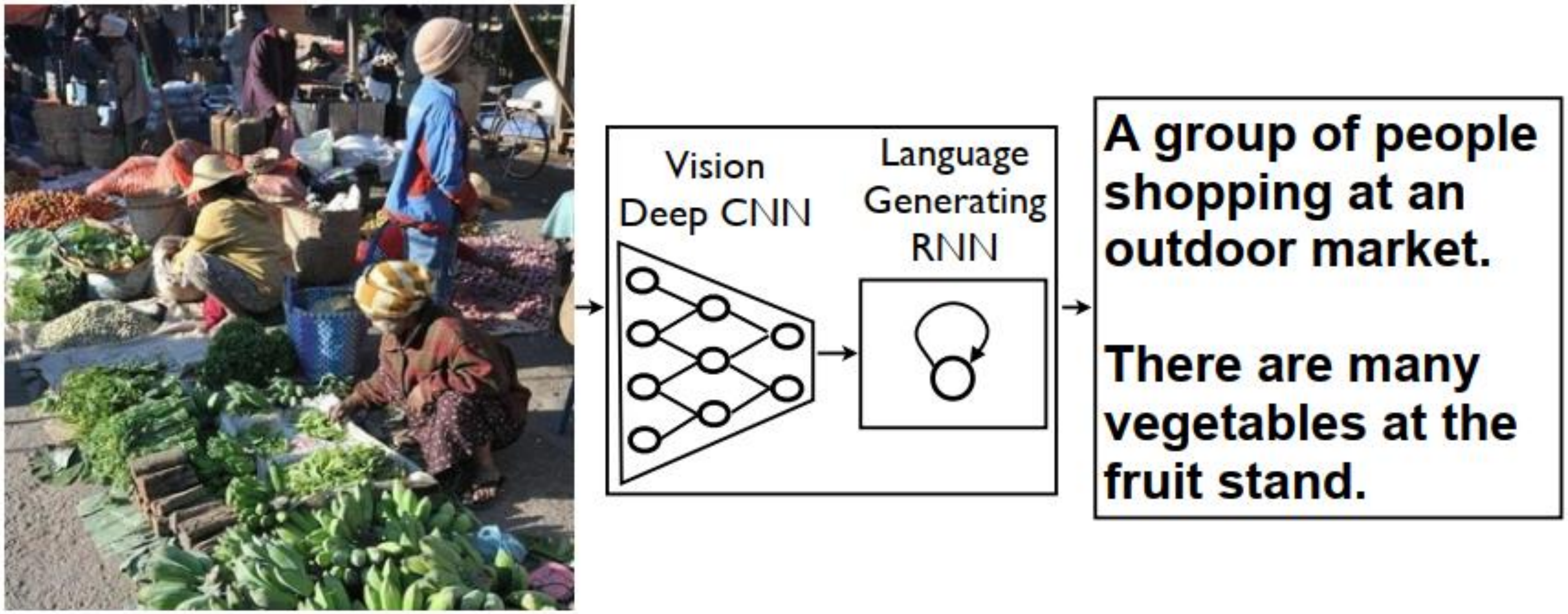
NEURAL COMPUTATION 9(8):1735–1780, 1997

Sepp Hochreiter	Jürgen Schmidhuber
Fakultät für Informatik	IDSIA
Technische Universität München	Corso Elvezia 36
80290 München, Germany	6900 Lugano, Switzerland
hochreit@informatik.tu-muenchen.de	juergen@idsia.ch
http://www7.informatik.tu-muenchen.de/~hochreit	http://www.idsia.ch/~juergen

[12]

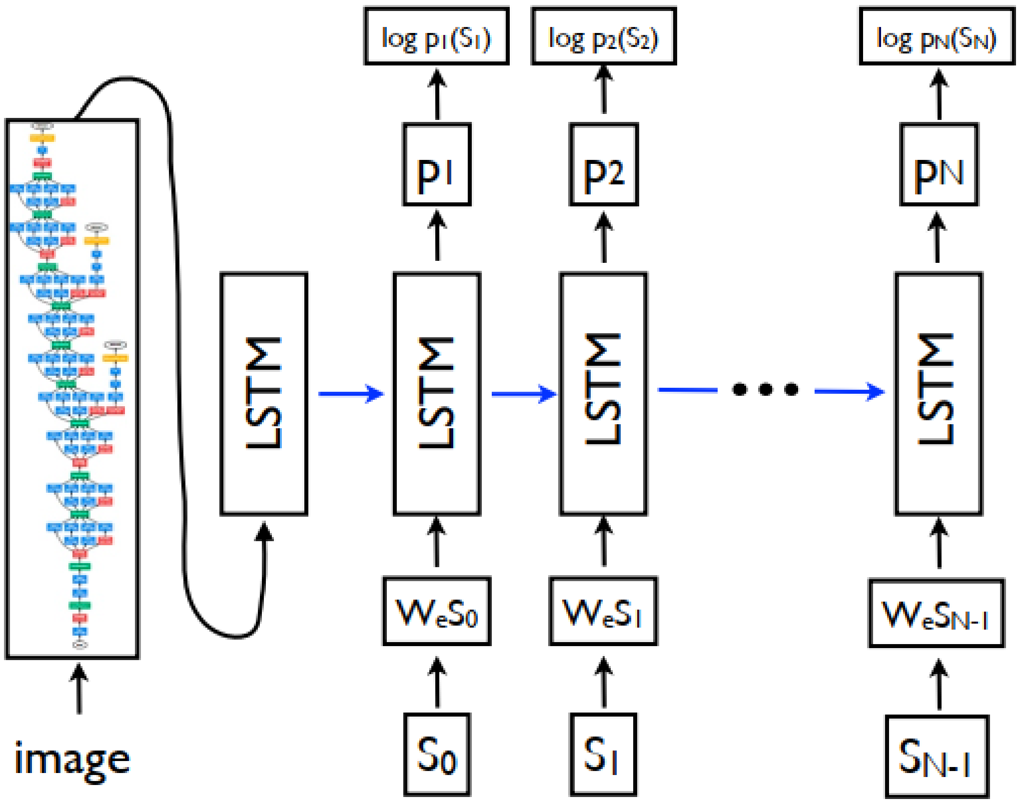
4. Advanced RNN Structures

LSTM example: image captioning [13]



4. Advanced RNN Structures

LSTM example: image captioning [13]



4. Advanced RNN Structures

LSTM example: image captioning [13]

4. Advanced RNN Structures

Gated Recurrent Unit (GRU) [14]

The same idea of using gates:

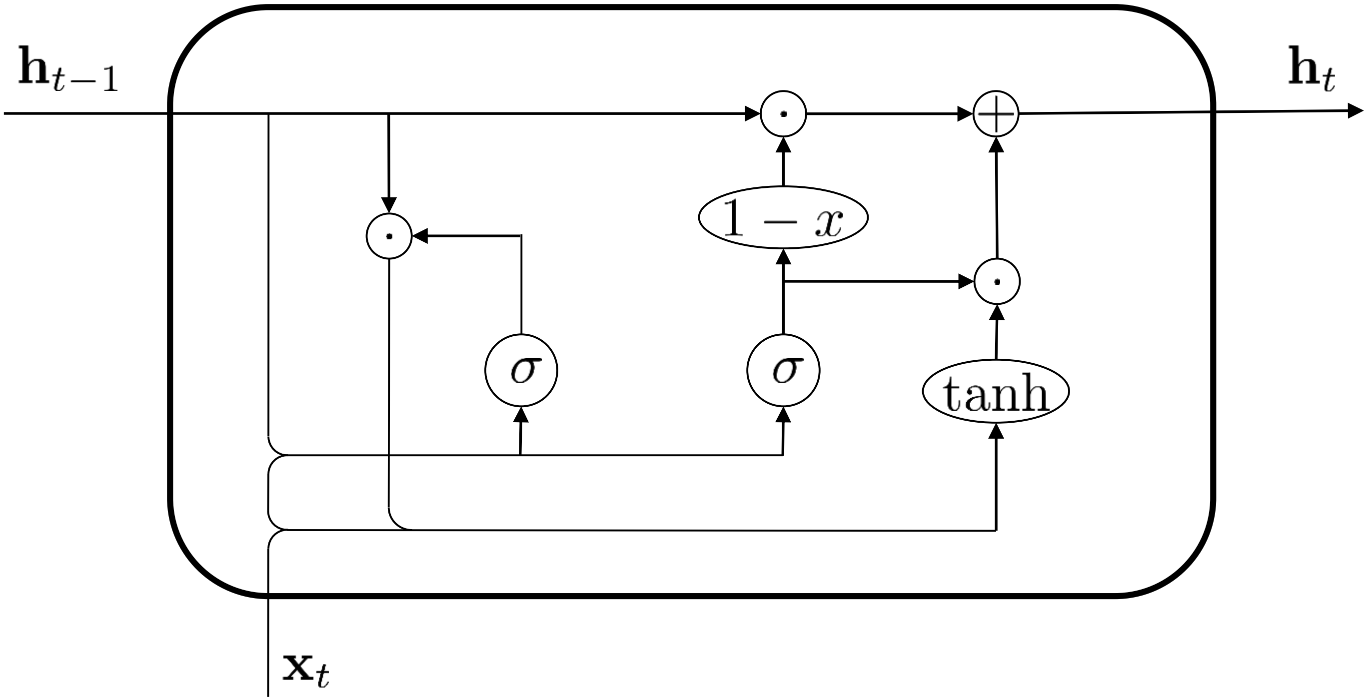
- Reset gate $\mathbf{r}_t = \sigma(\mathbf{W}_r \mathbf{h}_{t-1} + \mathbf{U}_r \mathbf{x}_t + \mathbf{b}_r)$
- Update gate $\mathbf{z}_t = \sigma(\mathbf{W}_z \mathbf{h}_{t-1} + \mathbf{U}_z \mathbf{x}_t + \mathbf{b}_z)$

$$\hat{\mathbf{h}}_t = \tanh(\mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{U}_h (\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}_h)$$

$$\mathbf{h}_t = (\mathbf{1} - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \hat{\mathbf{h}}_t$$

4. Advanced RNN Structures

Gated Recurrent Unit (GRU) [14]



4. Wrap Up

- **Many structures and activations are possible**: many to many, many to one, multilayer, bidirectional, LSTM, GRU, …
- Hard to know a priori what will work best. Currently LSTM and GRU are used a lot.

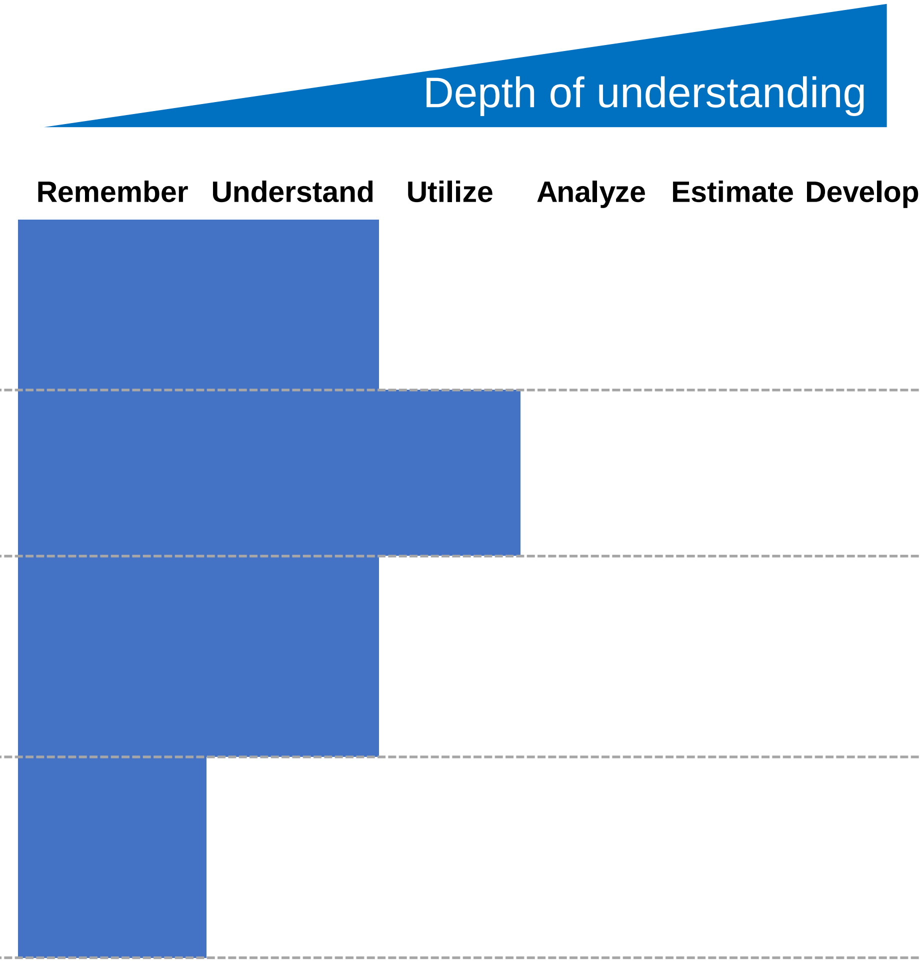
Objectives of Lecture 10

Clarify when Recurrent Neural Networks (RNN) are preferred over ‘static’ Neural Networks.

Show the basic methods for training a RNN.

Clarify problems that might occur when training a RNN for long sequences and how to avoid / mitigate them.

Give a short overview of different architectures for RNNs and their respective use cases.

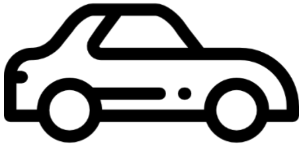


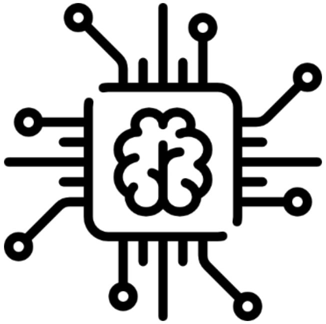
Recurrent Neural Networks

Maximilian Geißlinger / Fabian Netzler / Prof. Dr. Markus Lienkamp
(Matthias Rowold, M. Sc.)

Agenda

1. Introduction
    1. Motivating example
    2. The “hidden state”
    3. Connection to dynamical systems
2. Training RNNs
    1. Backpropagation through time
    2. Methods
3. Vanishing and exploding gradients
4. Advanced RNN structures
5. **Examples of RNNs in automotive applications**





5. Automotive Applications

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**

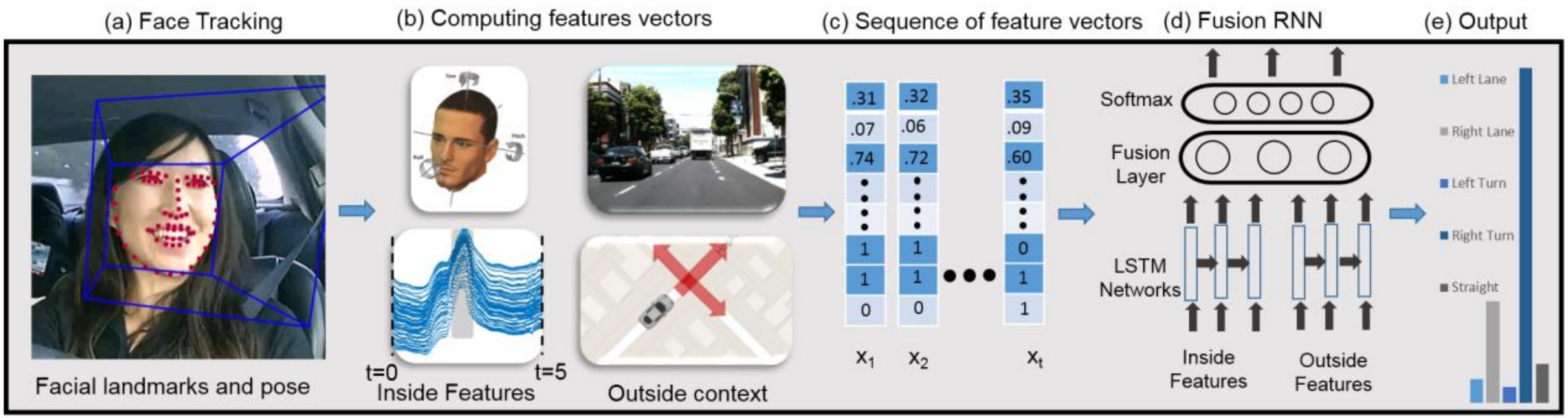
[15]

Maneuver anticipation

Predict the drivers action multiple seconds ahead.

→multiple sensors and LSTMs fused

5. Automotive Applications



[15]

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

5. Automotive Applications

Method		Lane change			Turns		
		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

[15]

- Comparison of different modifications, and other works.

5. Automotive Applications

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

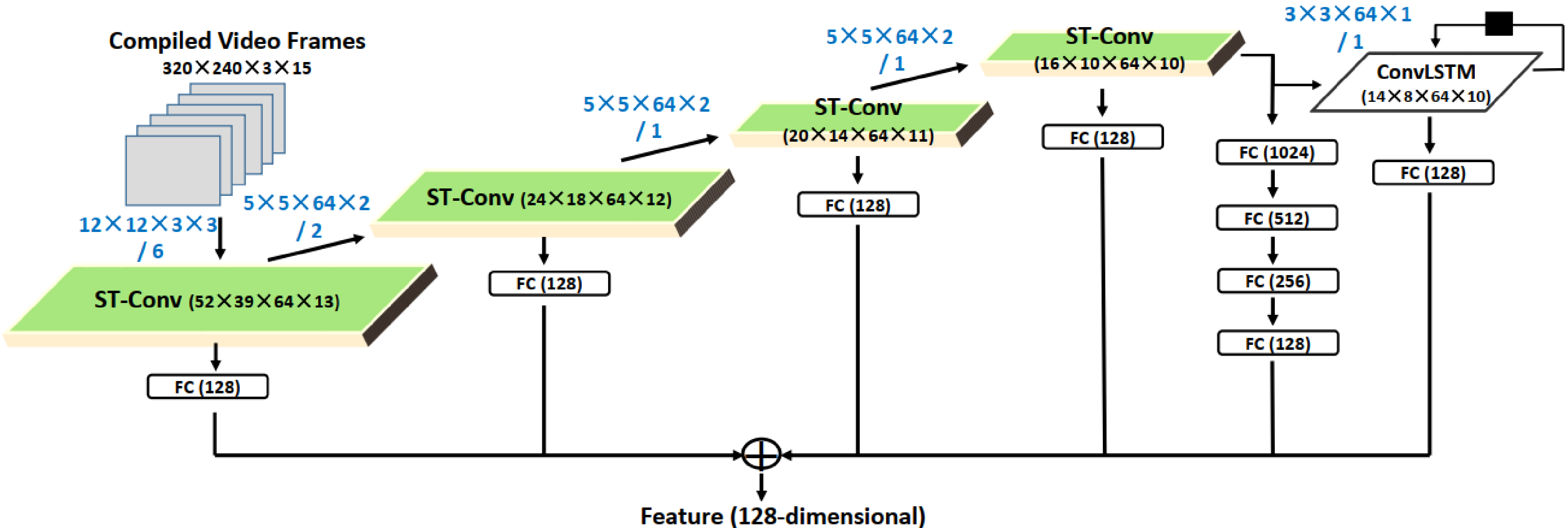
Chi, L. & Mu, Y.

arXiv preprint, **2017**

[16]

5. Automotive Applications

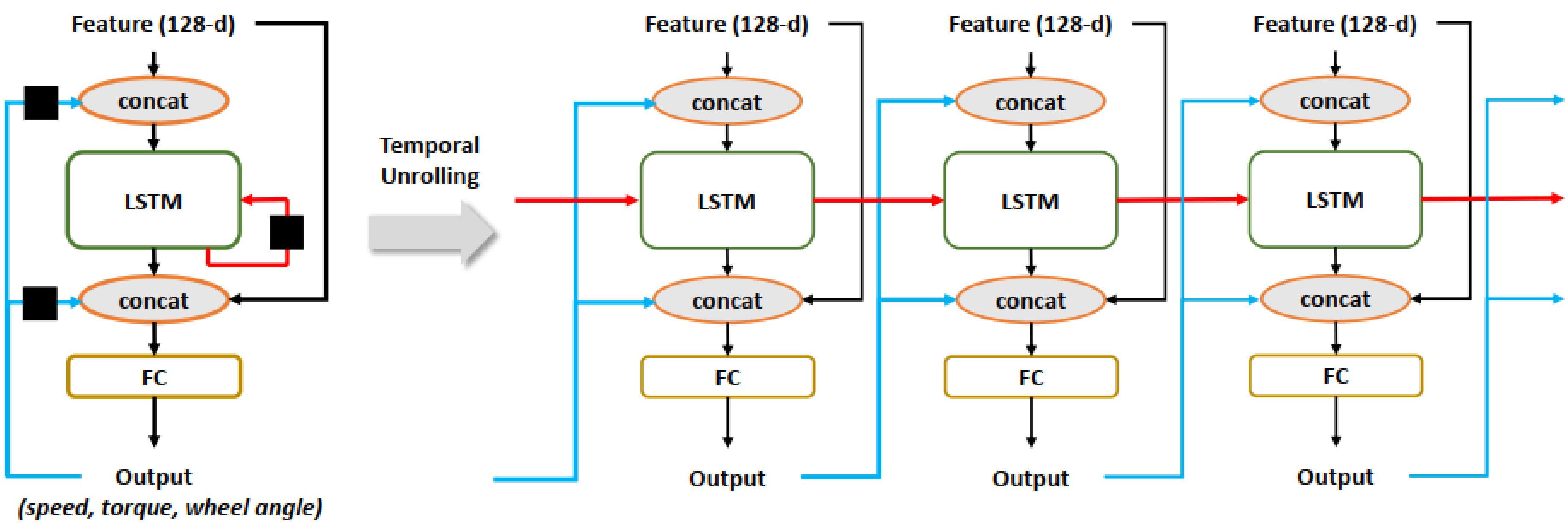
„Feature extraction sub-network“



[16]

5. Automotive Applications

„Steering-prediction sub-network“



[16]

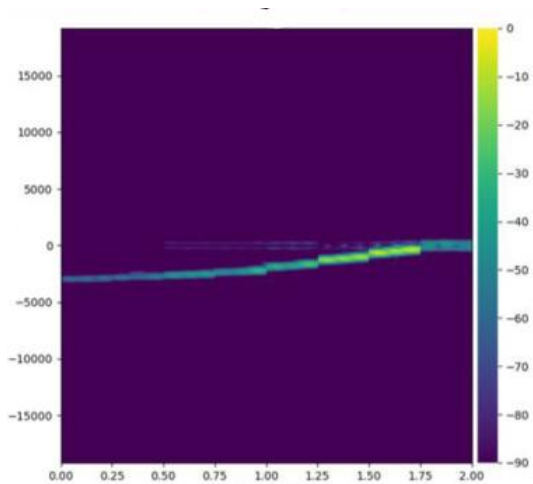
5. Automotive Applications

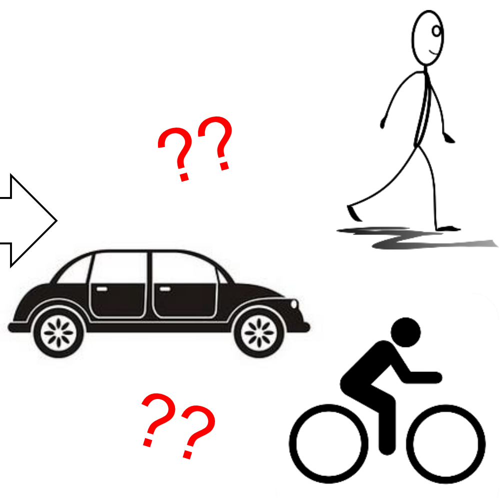
[16]

5. Automotive Applications

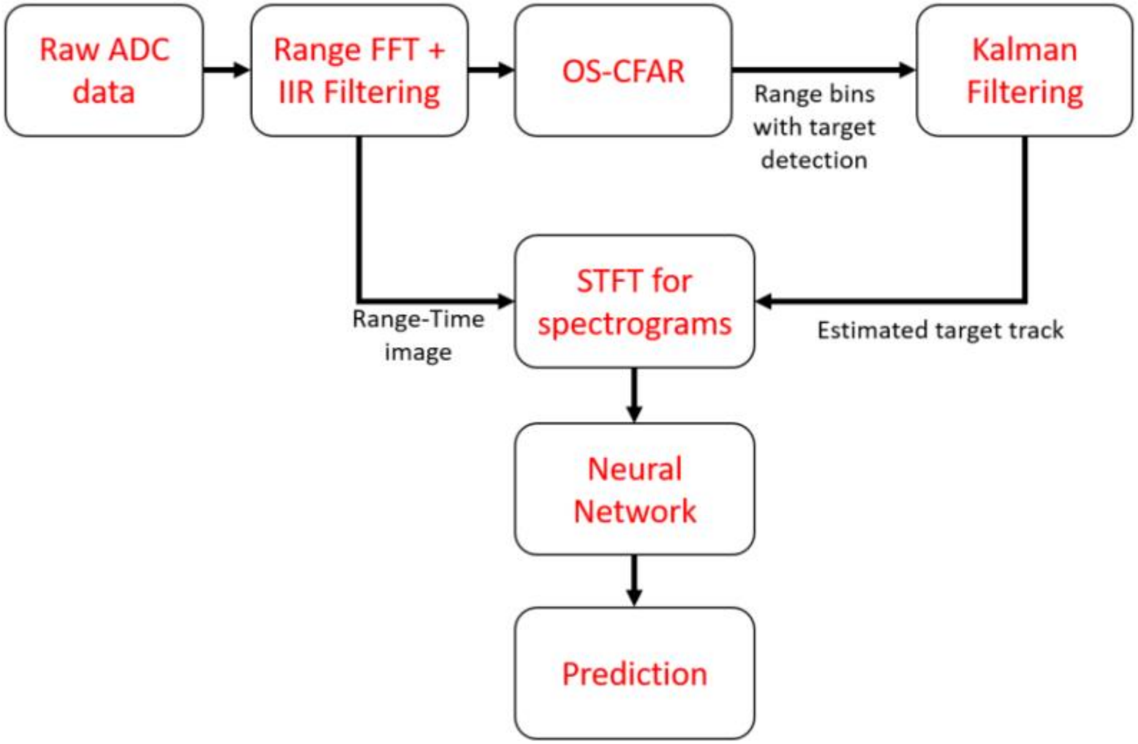
Practical classification of different moving targets using automotive radar and deep neural networks
Angelov, A.; Robertson, A.; Murray-Smith, R. & Fioranelli, F.
IET Radar, Sonar & Navigation, IET, **2018**

[17]





5. Automotive Applications



[17]

5. Automotive Applications

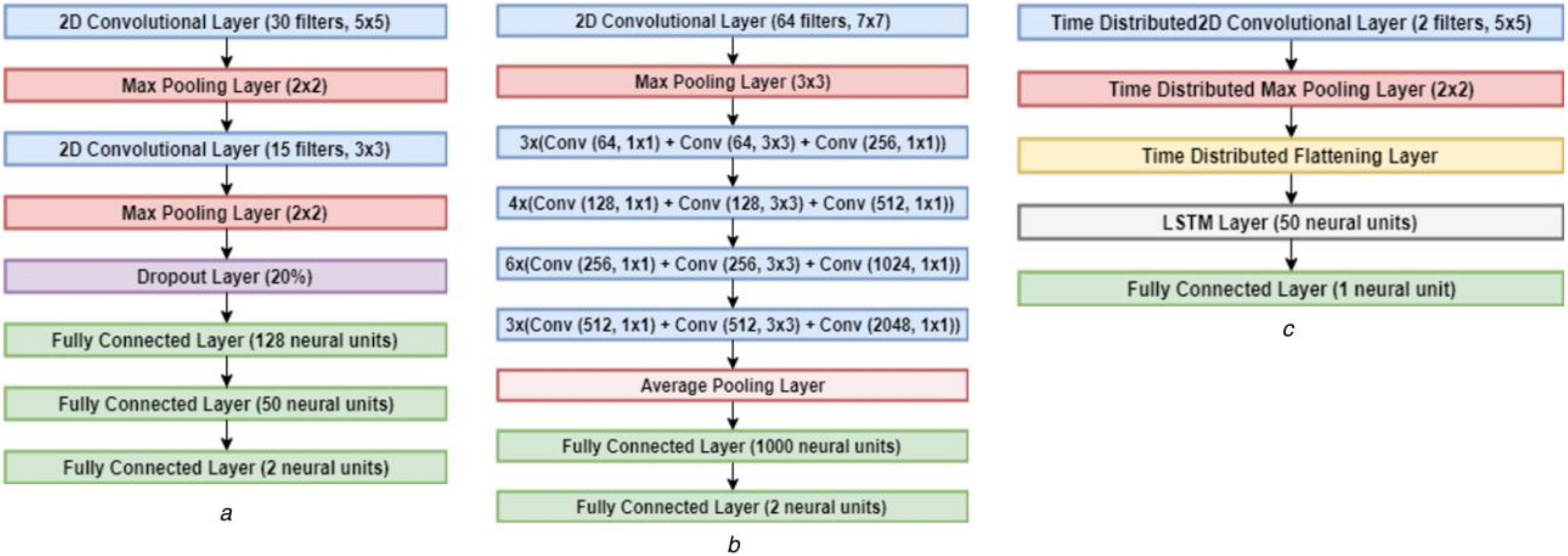


Fig. 3 *Representation of the different network architectures*
(a) Convolutional neural network similar to VGG type, ***(b)*** Convolutional residual network, ***(c)*** Combination of convolutional and recurrent LSTM network

[17]

5. Automotive Applications

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

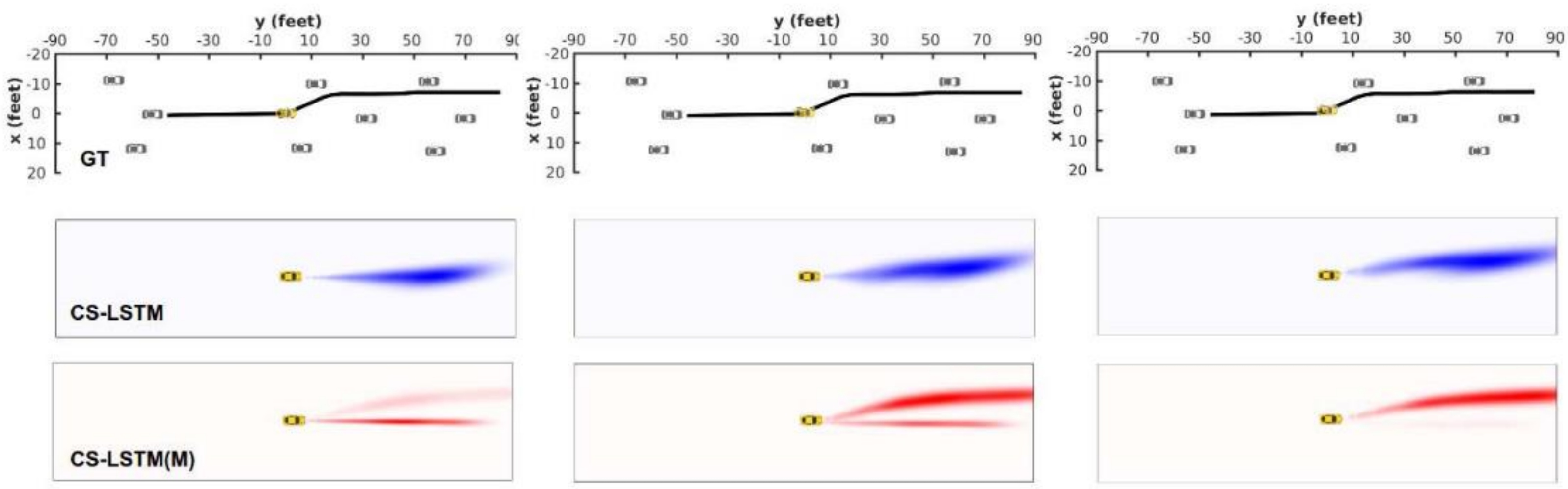
Evaluation/ network type	VGG-like CNN (2 s long datasets)	VGG-like CNN (0.5 s long datasets)	CNN-LSTM (2 s long datasets)	CNN-LSTM (0.5 s long datasets)
car-person-bicycle classification	78.6%	81.1%	50%	73.5%
car-person-2 people classification	77.8%	88.6%	44.4%	78.3%
all-4-classes-classification (VGG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

[17]

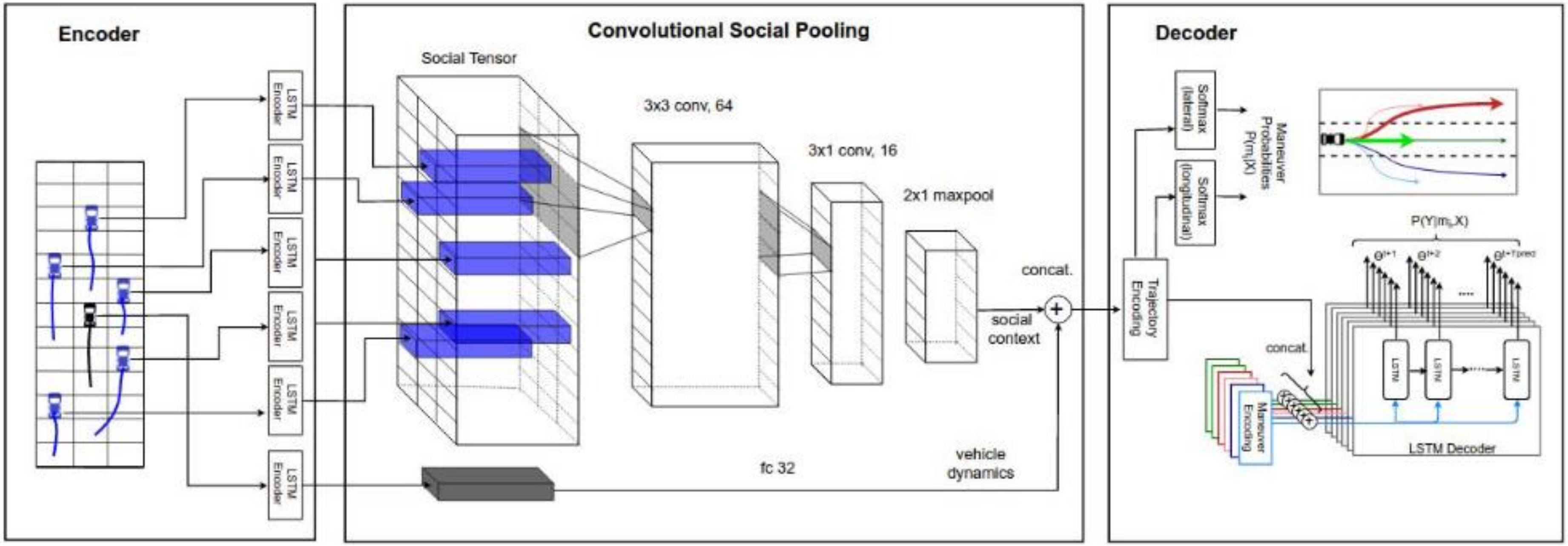
5. Automotive Applications

Convolutional Social Pooling for Vehicle Trajectory Prediction
Deo N.; Trivedi M.M.
2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, **2018**

[18]



5. Automotive Applications



[18]

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

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

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

[5] 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

[6] Zaremba, W. & Sutskever, I.; Vinyals, O.: Recurrent Neural Network Regularization; https://arxiv.org/abs/1409.2329, 2015

A.1 Literature

[7] Pascanu, R.; Mikolov, T.; Bengio Y.: On the difficulty of training recurrent neural networks; Proceedings of the 30th International Conference on Machine Learning; 2013

[8] Lillicrap, T. P.; Santoro, A.; Backpropagation through time and the brain; Current Opinion in Neurobiology; 2019

[9] Glorot, X.; Bengio, Y.; Understanding the difficulty of training deep feedforward neural networks; Proceedings of the 13th International Conference on Artificial Intelligence and Statistics; 2010

[10] Hermans, M.; Schrauwen, B.; Training and Analyzing Deep Recurrent Neural Networks; Advances in Neural Information Processing Systems 26; 2013

[11] Hochreiter, J.; Untersuchungen zu dynamischen neuronalen Netzen; Diplomarbeit; 1991

[12] Hochreiter, S.; Schmidhuber, J.; Long Short-Term Memory; Neural Computation 9(8); 1997

A.1 Literature

[13] Vinyals, O.; Toshev, A.; Bengio, S.; Erhan, D.; Show and Tell: Lessons learned from the 2015 MSCOCO Image Captioning Challenge; IEEE Transaction on Pattern Analysis and Machine Intelligence; 2016

[14] Cho, K.; Bahdanau, D.; Bougares, F; Schwenk, H.; Bengio, Y.; Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation; Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing; 2014

[15] Jain, A.; Singh, A.; Koppula, H. S.; Soh, S.; Saxena, A.; Recurrent Neural Networks for Driver Activity Anticipation via Sensory-Fusion Architecture; 2016 IEEE International Conference on Robotics and Automation; 2016

[16] Chi, L.; Mu, Y.; Deep Steering: Learning End-to-End Driving Model from Spatial and Temporal Visual Cues; Computer Science, ArXiv; 2017

[17] 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 12(10); 2018

A.1 Literature

[18] Deo, N.; Trivedi M. M.; Convolutional Social Pooling for Vehicle Trajectory Prediction; 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops; 2018