

Week 3: Deep Networks and Sequence Models

## Unit 1: The Need for Deeper Networks

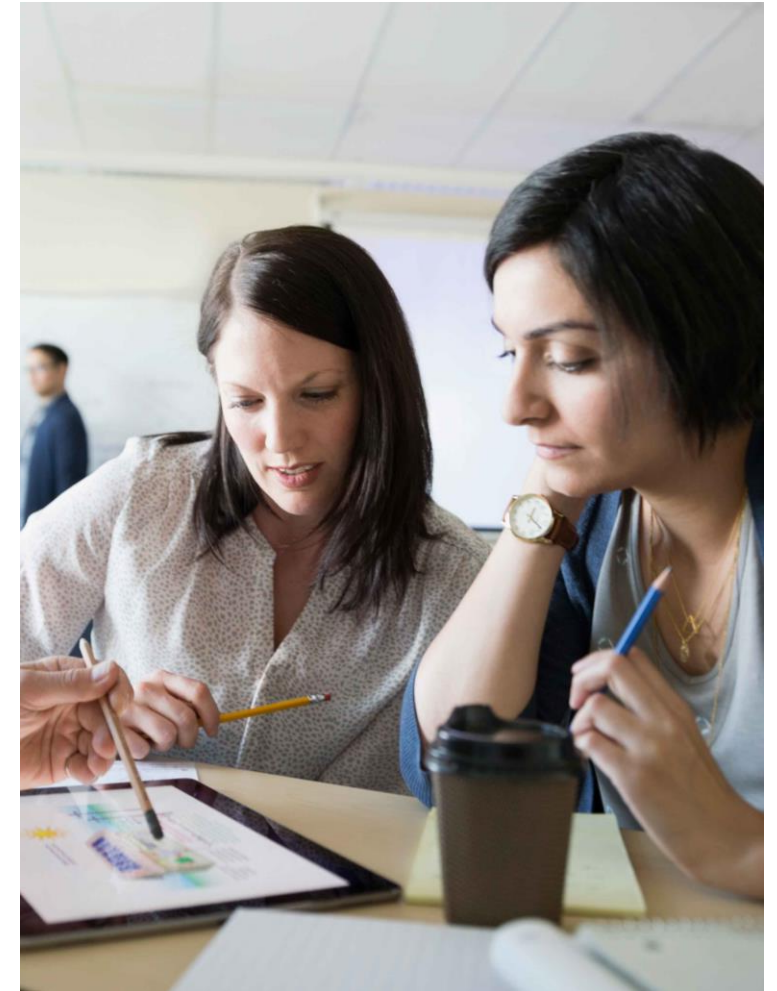
# The Need for Deeper Networks

What we covered in the last week

Experiment setup for developing machine learning applications

Classifying structure data with TensorFlow estimators

TF serving and architectures for deep learning



# The Need for Deeper Networks

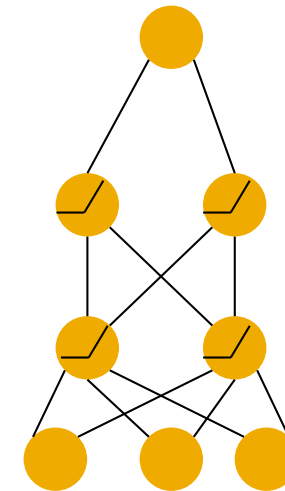
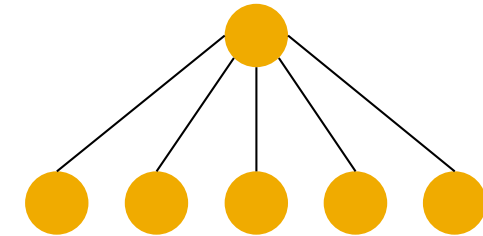
## Motivation

### Limitations of shallow networks

- Shallow, wide networks are good at memorization
- Generalize poorly on new data

### Capability of deeper networks

- Global features are learnt as a combination of local features along the depth of the network
- Less prone to memorization, better generalization



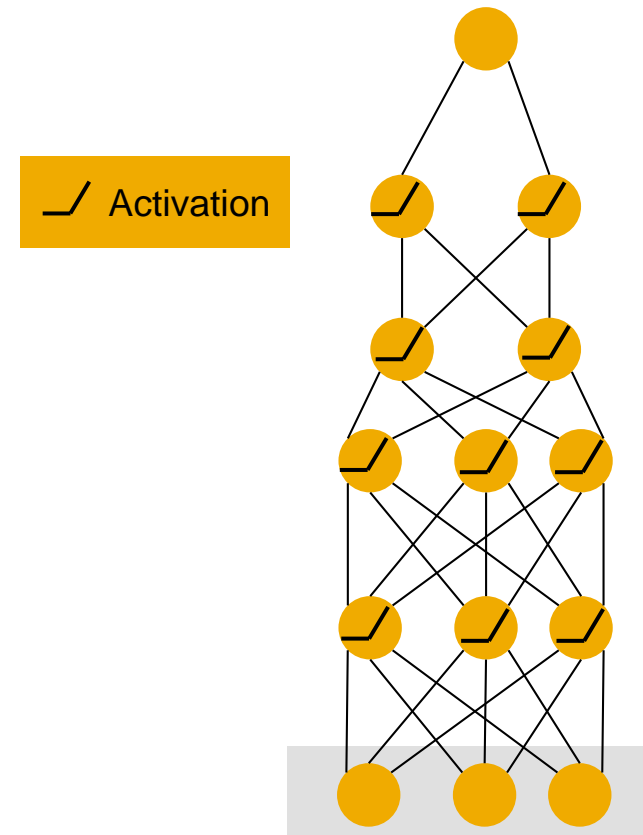
# The Need for Deeper Networks

Need for non-linearity

Deep networks without non-linearity behave like a single-layer network

Complex data cannot be separated with linear transformations

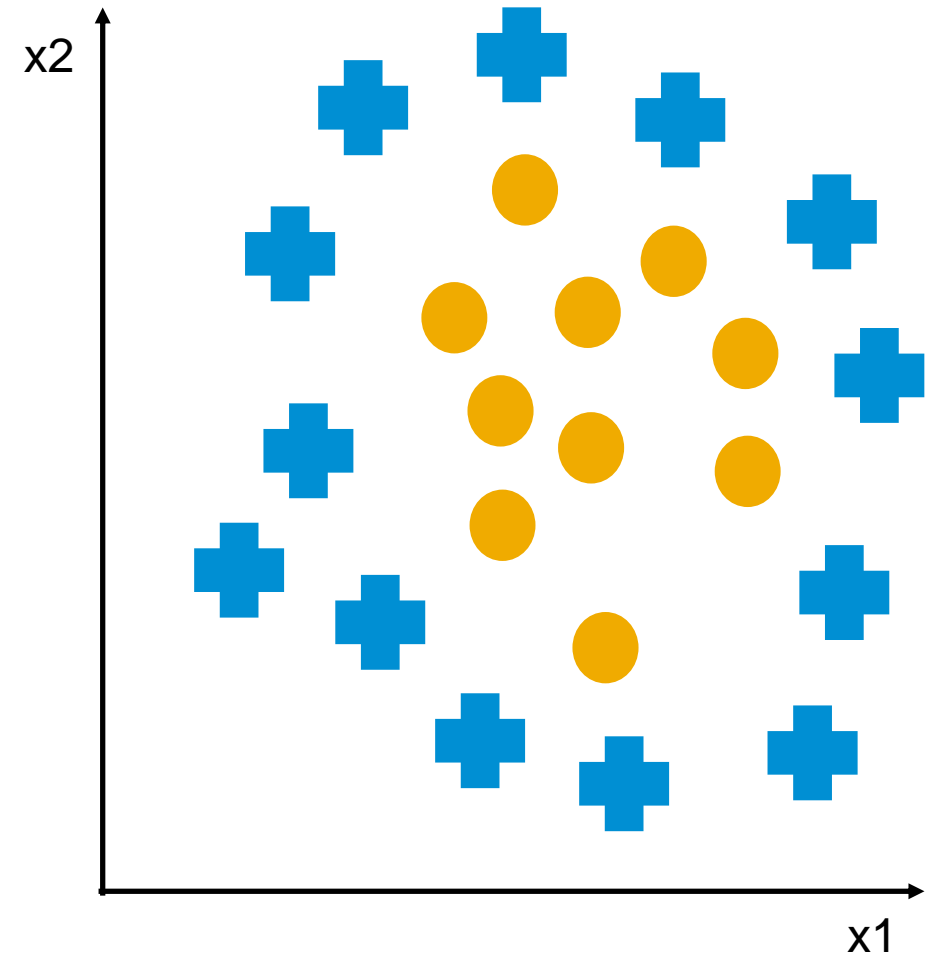
Non-linear activations can map input into a hyperspace where they are linearly separable



## Need for non-linearity

## Complex data cannot be separated with linear transformations

## Non-linear activations can map input into a hyperspace where they are linearly separable



# The Need for Deeper Networks

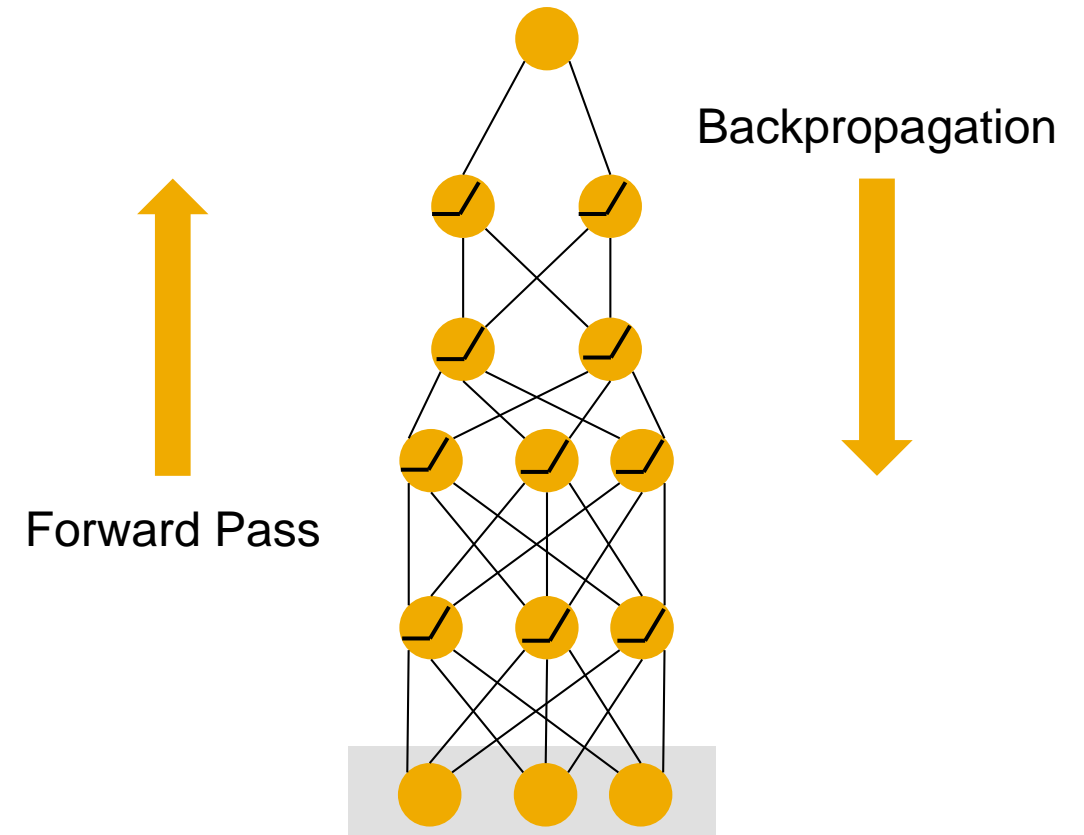
Learning process

## Forward pass

- Calculates scores based on weights of hidden nodes

## Backpropagation

- Calculates error contributed by each neuron after each batch is processed
- Weights are modified based on error calculated



# The Need for Deeper Networks

Learning objectives

## Loss and cost function

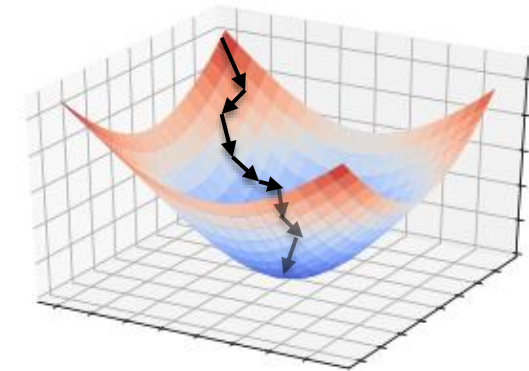
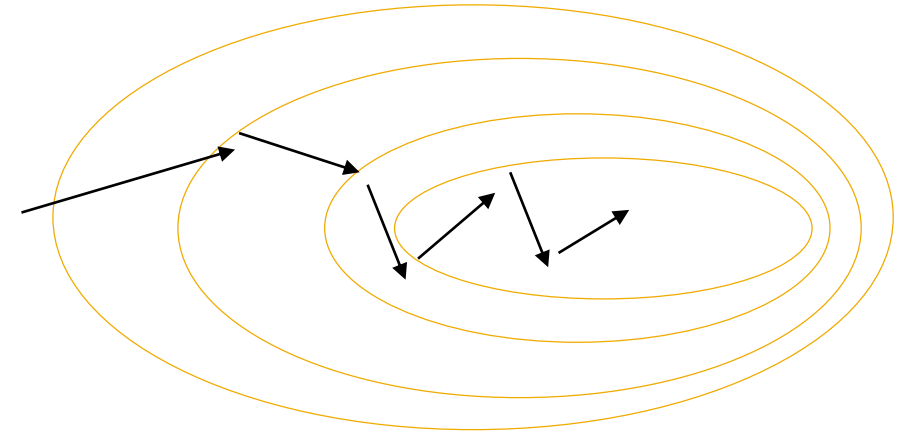
- Loss is computed on a single training example
- Cost is defined as average loss over all training data

## Softmax

- A classifier that converts scores to probabilities for each class

## Stochastic gradient descent (SGD)

- An optimizer commonly used for parameter updates
- Mini-batch of data to minimize the objective function



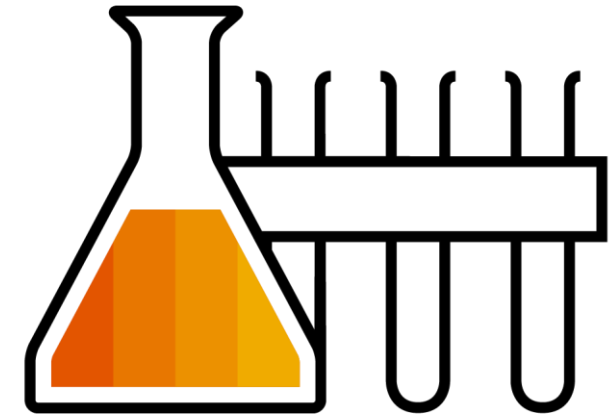
|      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 1.1  | 1.01 | 0.97 | 0.91 | 0.86 | 0.81 | 0.77 | 0.74 | 0.71 | 0.7  | 0.69 | 0.7  | 0.71 | 0.74 | 0.77 | 0.81 | 0.86 | 0.91 | 0.97 | 1.03 |
| 1.03 | 0.96 | 0.9  | 0.83 | 0.77 | 0.72 | 0.68 | 0.64 | 0.62 | 0.6  | 0.59 | 0.6  | 0.62 | 0.64 | 0.68 | 0.72 | 0.77 | 0.83 | 0.9  | 0.96 |
| 0.97 | 0.9  | 0.82 | 0.76 | 0.69 | 0.64 | 0.59 | 0.55 | 0.52 | 0.5  | 0.49 | 0.5  | 0.52 | 0.55 | 0.59 | 0.64 | 0.69 | 0.76 | 0.82 | 0.9  |
| 0.91 | 0.83 | 0.76 | 0.68 | 0.62 | 0.55 | 0.5  | 0.46 | 0.43 | 0.41 | 0.4  | 0.41 | 0.43 | 0.46 | 0.5  | 0.55 | 0.62 | 0.68 | 0.76 | 0.83 |
| 0.86 | 0.77 | 0.69 | 0.62 | 0.54 | 0.48 | 0.42 | 0.37 | 0.34 | 0.31 | 0.31 | 0.31 | 0.34 | 0.37 | 0.42 | 0.48 | 0.54 | 0.62 | 0.69 | 0.77 |
| 0.81 | 0.72 | 0.64 | 0.55 | 0.48 | 0.41 | 0.34 | 0.29 | 0.25 | 0.23 | 0.22 | 0.23 | 0.25 | 0.29 | 0.34 | 0.41 | 0.48 | 0.55 | 0.64 | 0.72 |
| 0.77 | 0.68 | 0.59 | 0.5  | 0.42 | 0.34 | 0.28 | 0.22 | 0.18 | 0.16 | 0.15 | 0.16 | 0.18 | 0.22 | 0.28 | 0.34 | 0.42 | 0.5  | 0.59 | 0.68 |
| 0.74 | 0.64 | 0.55 | 0.46 | 0.37 | 0.29 | 0.22 | 0.17 | 0.12 | 0.1  | 0.09 | 0.1  | 0.12 | 0.17 | 0.22 | 0.29 | 0.37 | 0.46 | 0.55 | 0.64 |
| 0.71 | 0.62 | 0.52 | 0.43 | 0.34 | 0.25 | 0.18 | 0.12 | 0.08 | 0.05 | 0.04 | 0.05 | 0.08 | 0.12 | 0.18 | 0.25 | 0.34 | 0.43 | 0.52 | 0.62 |
| 0.7  | 0.6  | 0.5  | 0.41 | 0.31 | 0.23 | 0.16 | 0.1  | 0.05 | 0.02 | 0.01 | 0.02 | 0.05 | 0.1  | 0.16 | 0.23 | 0.31 | 0.41 | 0.5  | 0.6  |
| 0.69 | 0.59 | 0.49 | 0.4  | 0.31 | 0.22 | 0.15 | 0.09 | 0.04 | 0.01 | 0    | 0.01 | 0.04 | 0.09 | 0.15 | 0.22 | 0.31 | 0.4  | 0.49 | 0.59 |
| 0.7  | 0.6  | 0.5  | 0.41 | 0.31 | 0.23 | 0.16 | 0.1  | 0.05 | 0.02 | 0.01 | 0.02 | 0.05 | 0.1  | 0.16 | 0.23 | 0.31 | 0.41 | 0.5  | 0.6  |
| 0.71 | 0.62 | 0.52 | 0.43 | 0.34 | 0.25 | 0.18 | 0.12 | 0.08 | 0.05 | 0.04 | 0.05 | 0.08 | 0.12 | 0.18 | 0.25 | 0.34 | 0.43 | 0.52 | 0.62 |
| 0.74 | 0.64 | 0.55 | 0.46 | 0.37 | 0.29 | 0.22 | 0.17 | 0.12 | 0.1  | 0.09 | 0.1  | 0.12 | 0.17 | 0.22 | 0.29 | 0.37 | 0.46 | 0.55 | 0.64 |
| 0.77 | 0.68 | 0.59 | 0.5  | 0.42 | 0.34 | 0.28 | 0.22 | 0.18 | 0.16 | 0.15 | 0.16 | 0.18 | 0.22 | 0.28 | 0.34 | 0.42 | 0.5  | 0.59 | 0.68 |
| 0.81 | 0.72 | 0.64 | 0.55 | 0.48 | 0.41 | 0.34 | 0.29 | 0.25 | 0.23 | 0.22 | 0.23 | 0.25 | 0.29 | 0.34 | 0.41 | 0.48 | 0.55 | 0.64 | 0.72 |
| 0.86 | 0.77 | 0.69 | 0.62 | 0.54 | 0.48 | 0.42 | 0.37 | 0.34 | 0.31 | 0.31 | 0.31 | 0.34 | 0.37 | 0.42 | 0.48 | 0.54 | 0.62 | 0.69 | 0.77 |
| 0.91 | 0.83 | 0.76 | 0.68 | 0.62 | 0.55 | 0.5  | 0.46 | 0.43 | 0.41 | 0.4  | 0.41 | 0.43 | 0.46 | 0.5  | 0.55 | 0.62 | 0.68 | 0.76 | 0.83 |
| 0.97 | 0.9  | 0.82 | 0.76 | 0.69 | 0.64 | 0.59 | 0.55 | 0.52 | 0.5  | 0.49 | 0.5  | 0.52 | 0.55 | 0.59 | 0.64 | 0.69 | 0.76 | 0.82 | 0.9  |
| 1.03 | 0.96 | 0.9  | 0.83 | 0.77 | 0.72 | 0.68 | 0.64 | 0.62 | 0.6  | 0.59 | 0.6  | 0.62 | 0.64 | 0.68 | 0.72 | 0.77 | 0.83 | 0.9  | 0.96 |

# The Need for Deeper Networks

Classification example

## Classification with Fashion MNIST

- Develop a simple single-layer neural network
- Evaluate performance on Fashion MNIST
- Add more layers and evaluate improvement



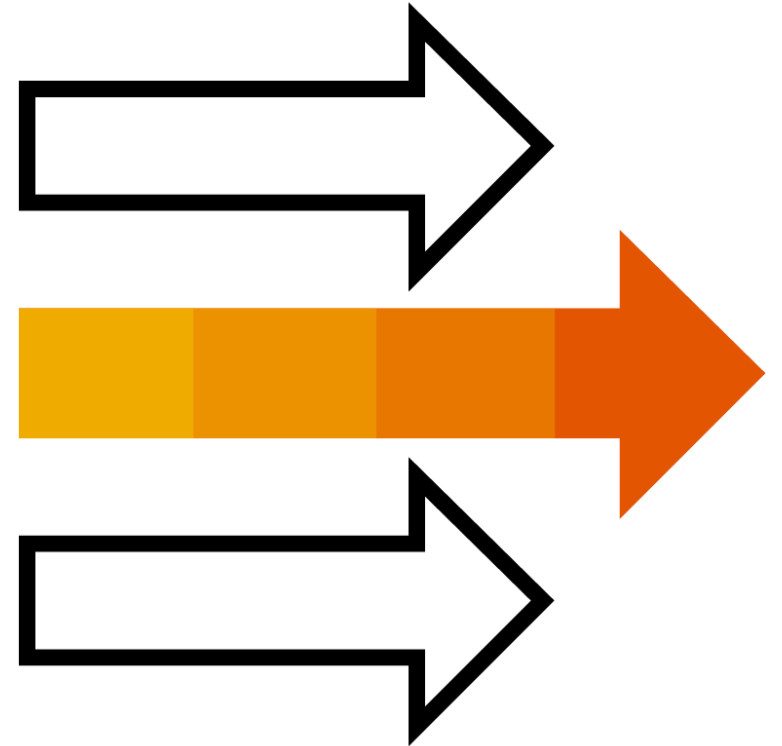


# The Need for Deeper Networks

Coming up next

Introduction to sequence models

How to process sequence data



# Thank you.

**Contact information:**

**open@sap.com**

© 2017 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company.

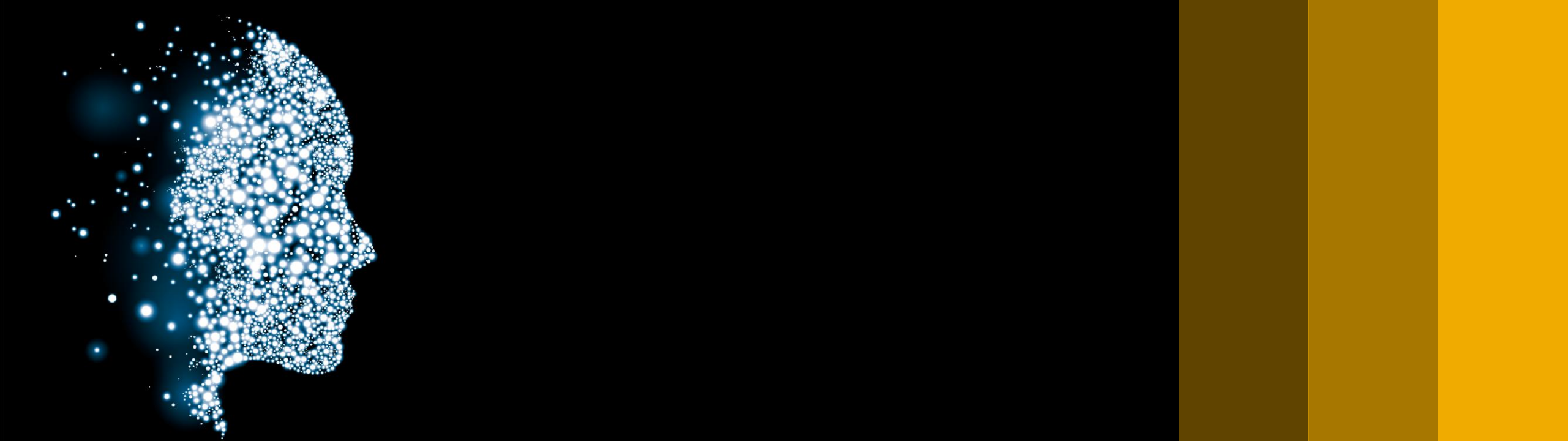
The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.

See <http://global.sap.com/corporate-en/legal/copyright/index.epx> for additional trademark information and notices.



Week 3: Deep Networks and Sequence Models

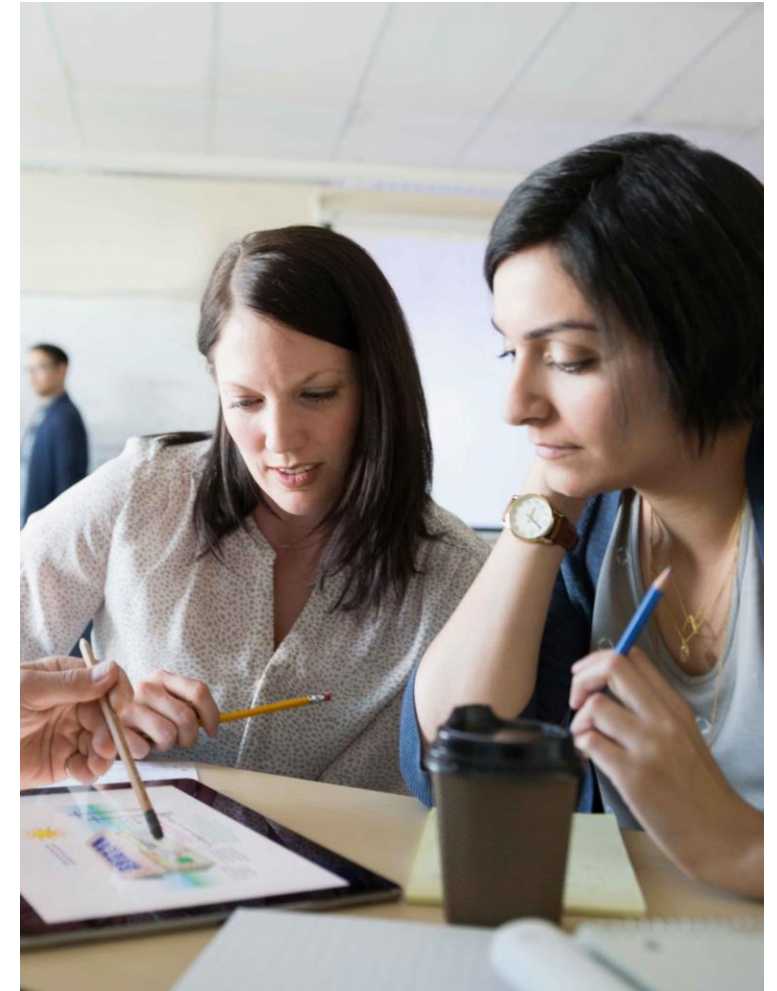
## Unit 2: Introduction to Sequence Models

# Introduction to Sequence Models

What we covered in the last unit

Deep networks

Deep feed-forward networks

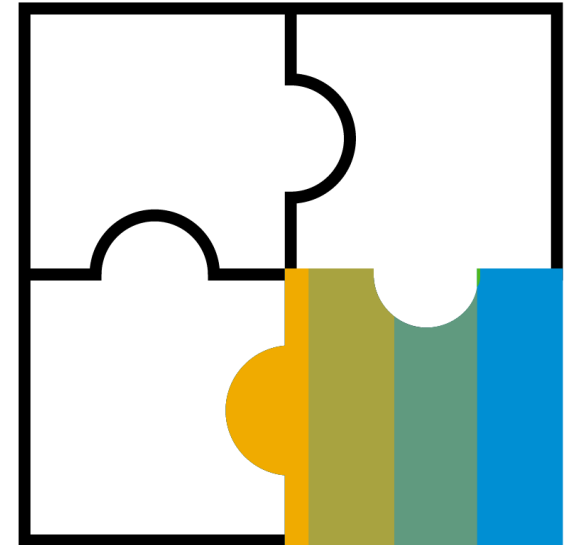


# Introduction to Sequence Models

## Overview

### Content:

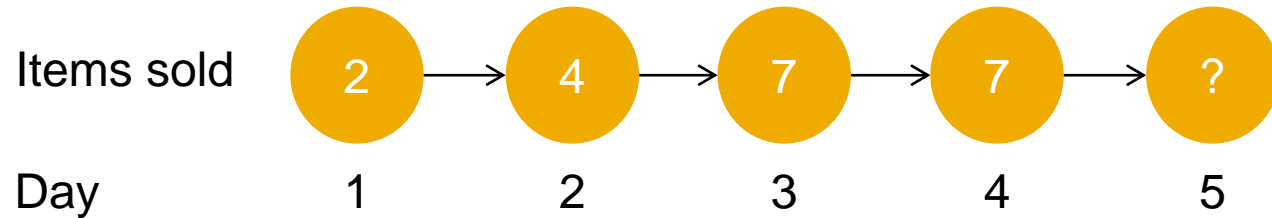
- Sequence data
- Sequence models
- Applications of sequence models



# Introduction to Sequence Models

Sequence data

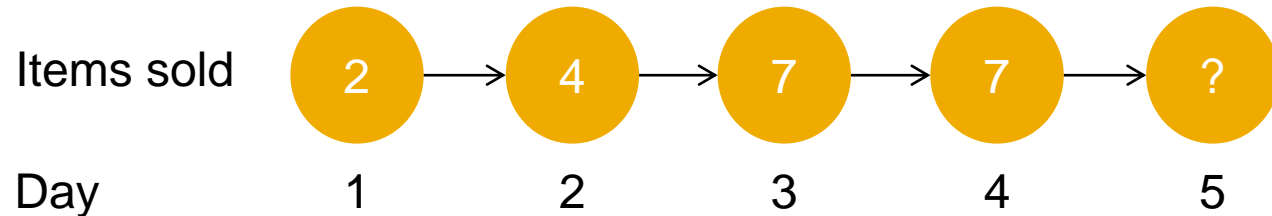
## Time Series



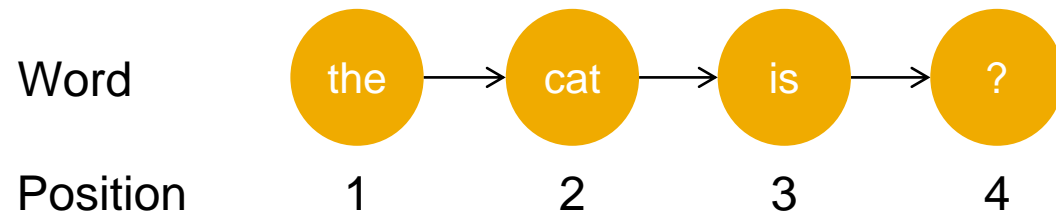
# Introduction to Sequence Models

Sequence data

## Time Series



## Sentences

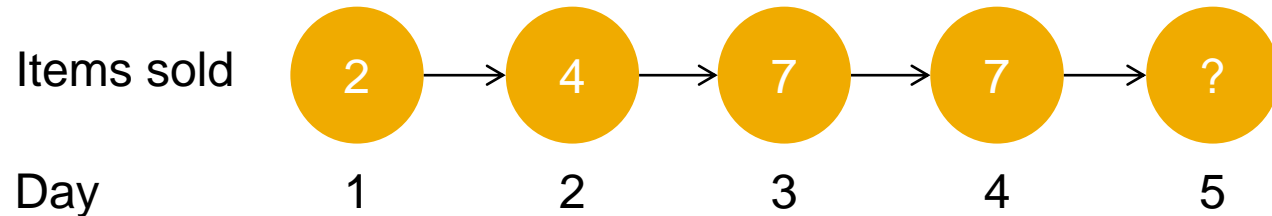




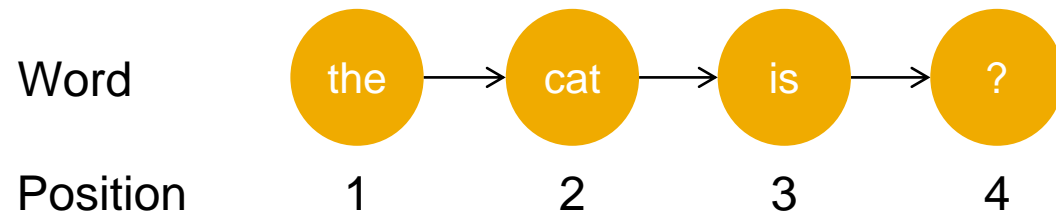
# Introduction to Sequence Models

Sequence data

## Time Series



## Sentences



## Characteristics:

- Ordered elements
- Number of elements can vary

# Introduction to Sequence Models

What decides the next element?

- Time series
  - External factors: weather, competing products, etc.
  - Internal factors: previous values
- Sentences
  - External factors: paragraph, conversation context, etc.
  - Internal factors: previous words
- Primary focus is on *internal factors*
  - Previous elements determine the next element, up to noise
  - Mathematically:  $e_{t+1} = f(e_1, e_2, \dots, e_t) + \epsilon$

↑  
error in the prediction  
(unexplained noise)

# Introduction to Sequence Models

Inferring the next word in a sentence

- Given a fixed vocabulary  $V$  and present words  $w_1, w_2, \dots, w_t$ ,
  - Compute the probability of each candidate next word  $x$  in  $V$  given  $w_1, w_2, \dots, w_t$
  - Equivalently compute  $P(x \mid w_1, w_2, \dots, w_t)$  for each  $x$  in  $V$
  - Set  $w_{t+1}$  to  $x$  with maximal conditional probability
  - Mathematically:  $w_{t+1} = \arg \max_{x \in V} P(x \mid w_1, w_2, \dots, w_t)$

# Introduction to Sequence Models

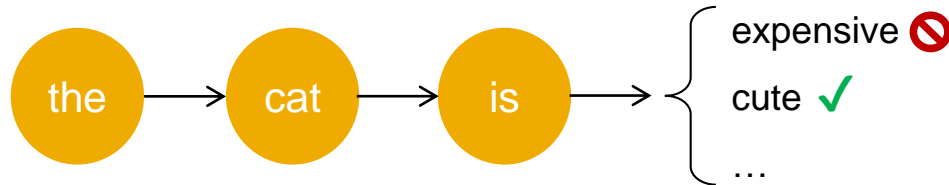
Inferring the next word in a sentence – Traditional approaches

- n-gram models
  - Infer  $w_{t+1}$  based on a fixed number  $n$  of previous words (e.g., hidden Markov models)

# Introduction to Sequence Models

Inferring the next word in a sentence – Traditional approaches

- n-gram models
  - Infer  $w_{t+1}$  based on a fixed number  $n$  of previous words (e.g., hidden Markov models)
  - Example: let  $n = 3$

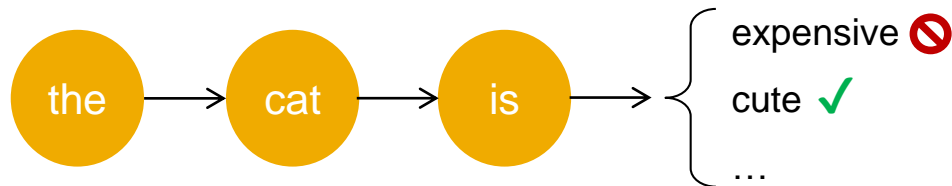


- Issue: the parameter  $n$  is restrictive and somewhat arbitrary

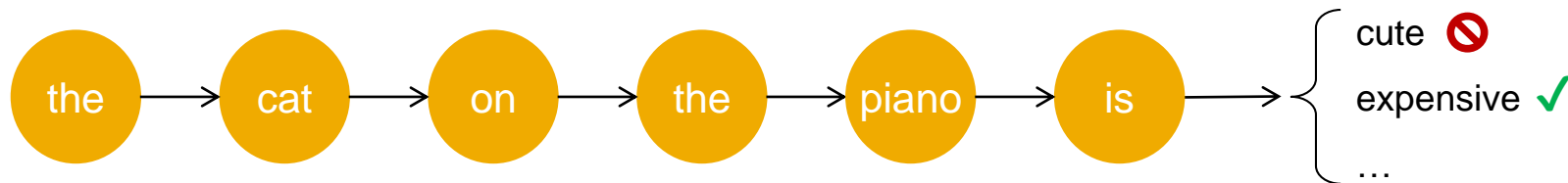
# Introduction to Sequence Models

Inferring the next word in a sentence – Traditional approaches

- n-gram models
  - Infer  $w_{t+1}$  based on a fixed number  $n$  of previous words (e.g., hidden Markov models)
  - Example: let  $n = 3$



- Issue: the parameter  $n$  is restrictive and somewhat arbitrary



# Introduction to Sequence Models

Inferring the next word in a sentence – Traditional approaches

- Bag of words
  - Infer  $w_{t+1}$  based on a fixed-length vector of word count built on previous words

# Introduction to Sequence Models

Inferring the next word in a sentence – Traditional approaches

- Bag of words
  - Infer  $w_{t+1}$  based on a fixed-length vector of word count built on previous words
  - Example: assume the dictionary consists of words “the”, “cat”, “nice”, “caught”, “jump”

|                          | the | cat | nice | caught | jump |
|--------------------------|-----|-----|------|--------|------|
| the cat is nice          | 1   | 1   | 1    | 0      | 0    |
| the cat caught the mouse | 2   | 1   | 0    | 1      | 0    |



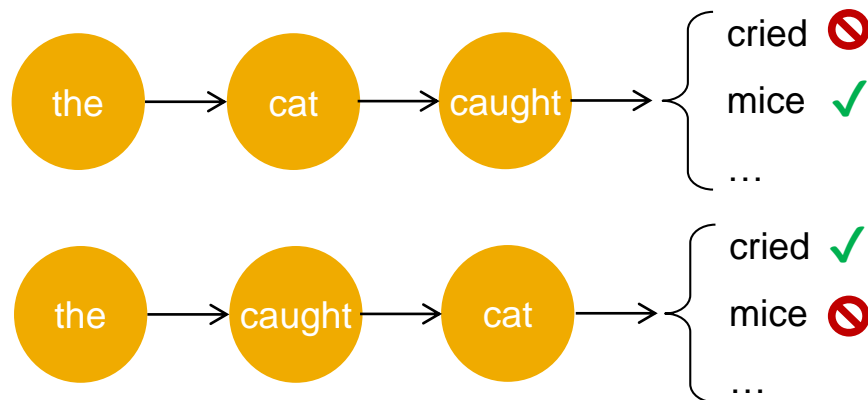
# Introduction to Sequence Models

Inferring the next word in a sentence – Traditional approaches

- Bag of words
  - Infer  $w_{t+1}$  based on a fixed-length vector of word count built on previous words
  - Example: assume the dictionary consists of words “the”, “cat”, “nice”, “caught”, “jump”

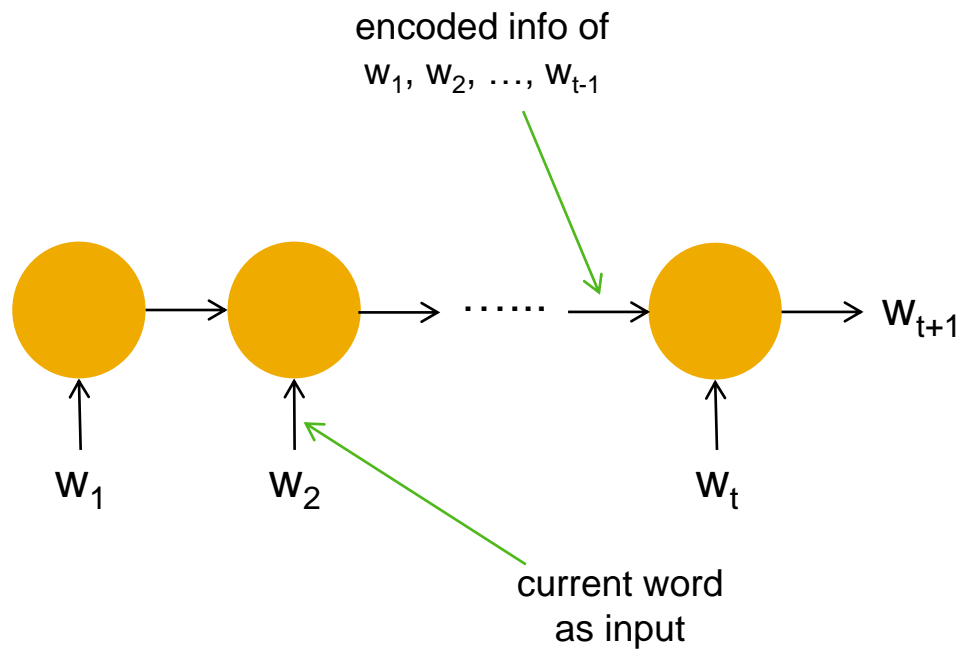
|                          | the | cat | nice | caught | jump |
|--------------------------|-----|-----|------|--------|------|
| the cat is nice          | 1   | 1   | 1    | 0      | 0    |
| the cat caught the mouse | 2   | 1   | 0    | 1      | 0    |

- Issue: the ordering is lost and hence the semantics of words



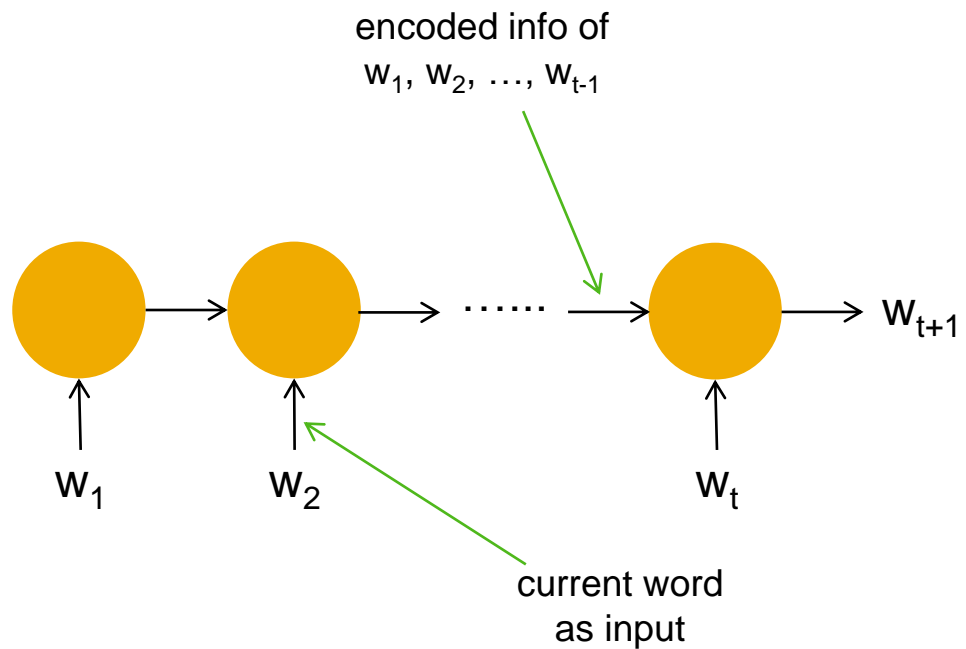
# Introduction to Sequence Models

## Sequence models



# Introduction to Sequence Models

## Sequence models



### Motivation:

We change

$$w_{t+1} = f(w_1, w_2, \dots, w_{t-1}, w_t)$$

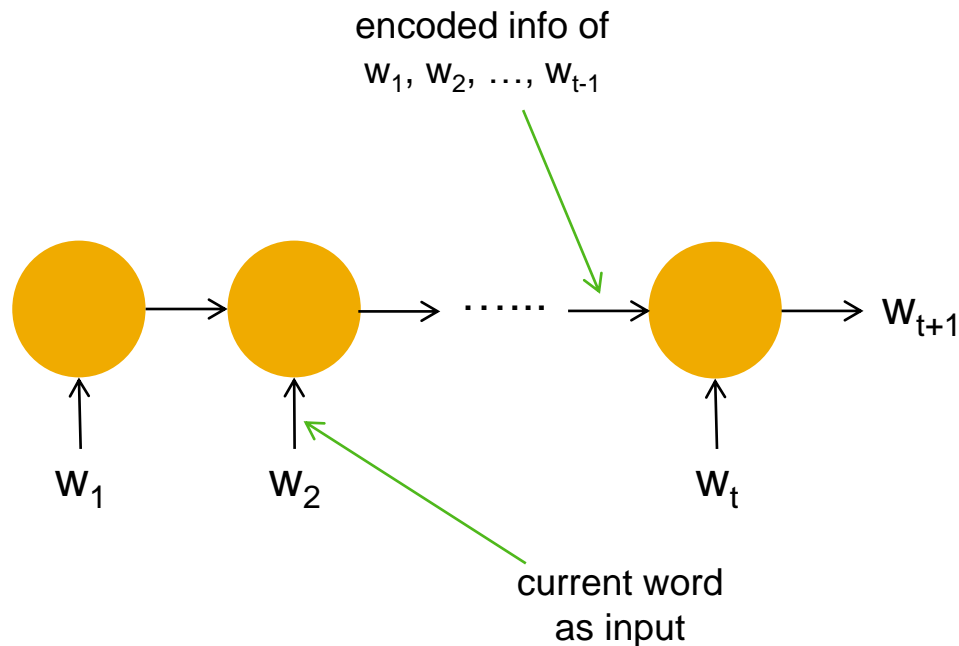
to

$$w_{t+1} = f(h, w_t)$$

where  $h$  is the encoded info of  $w_1, w_2, \dots, w_{t-1}$

# Introduction to Sequence Models

## Sequence models



### Motivation:

We change

$$w_{t+1} = f(w_1, w_2, \dots, w_{t-1}, w_t)$$

to

$$w_{t+1} = f(h, w_t)$$

where  $h$  is the encoded info of  $w_1, w_2, \dots, w_{t-1}$

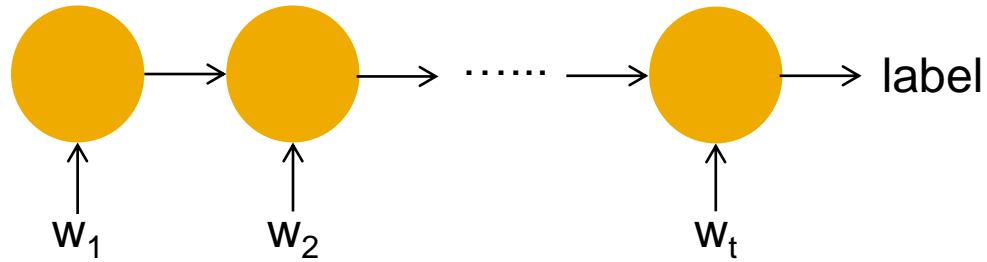
**Remarks:** When  $h$  can be expressed as a fixed-length feature vector:

- The function  $f$  can be learned in a similar way across different positions/states
- Sentences with different lengths are handled consistently

# Introduction to Sequence Models

Other applications of sequence models – Sequence classification

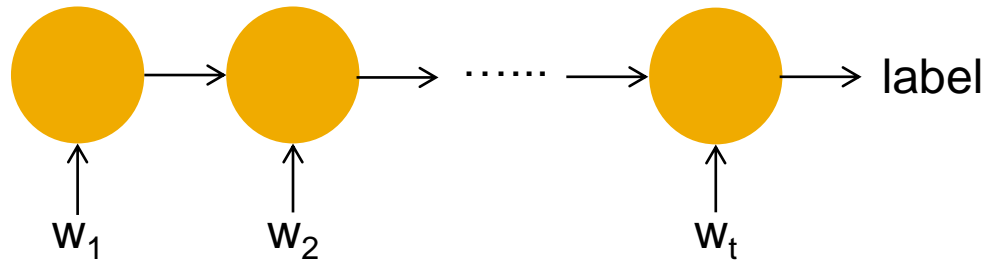
- Text classification



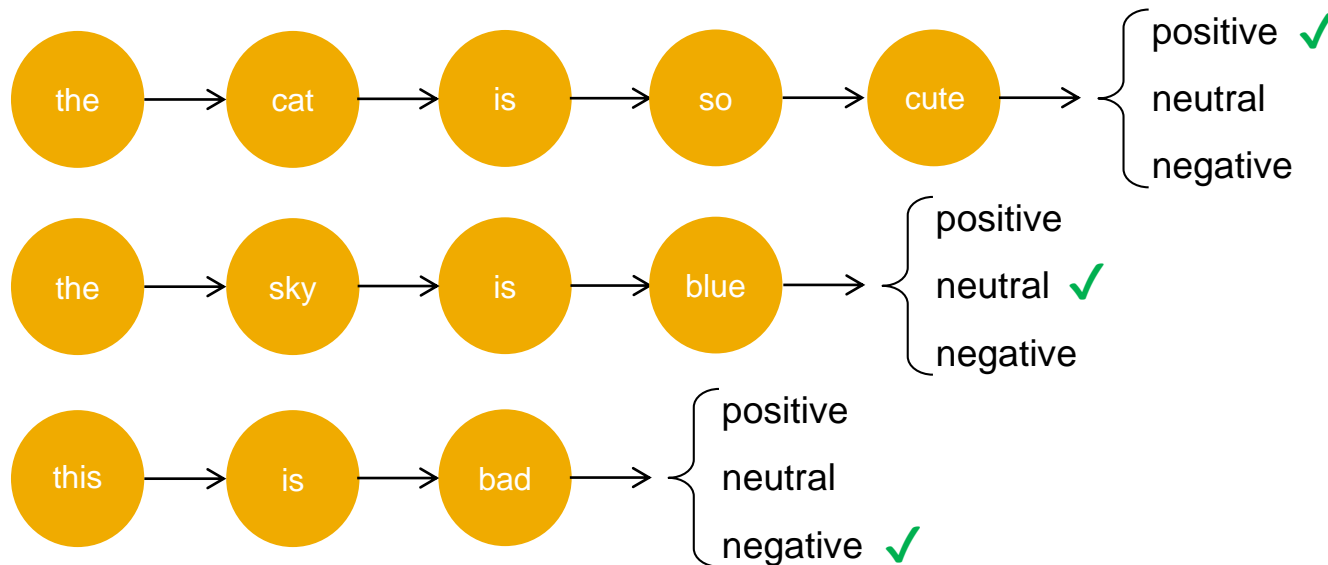
# Introduction to Sequence Models

Other applications of sequence models – Sequence classification

- Text classification



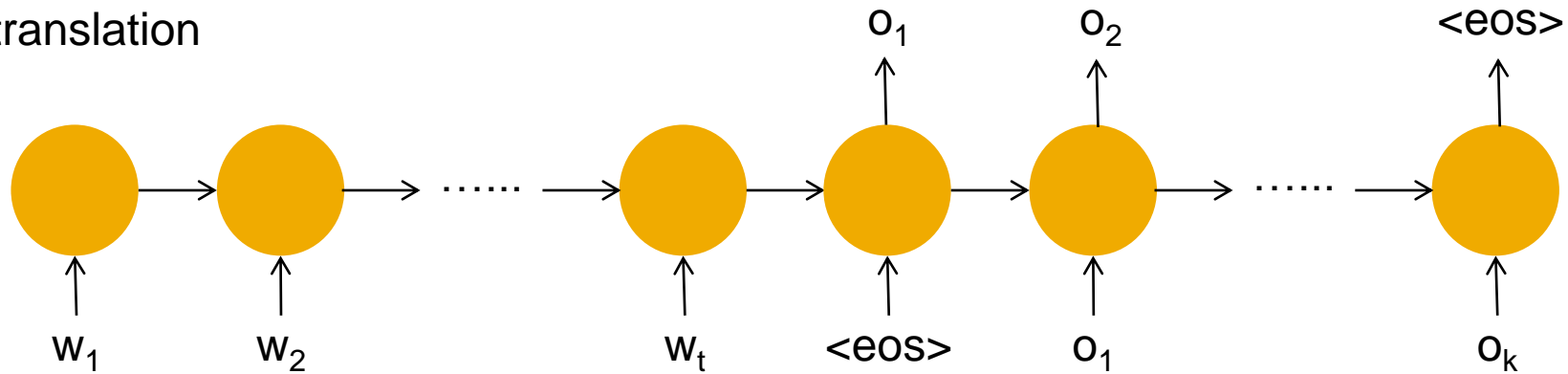
- Example: Sentiment analysis



# Introduction to Sequence Models

Other applications of sequence models – Sequence to sequence

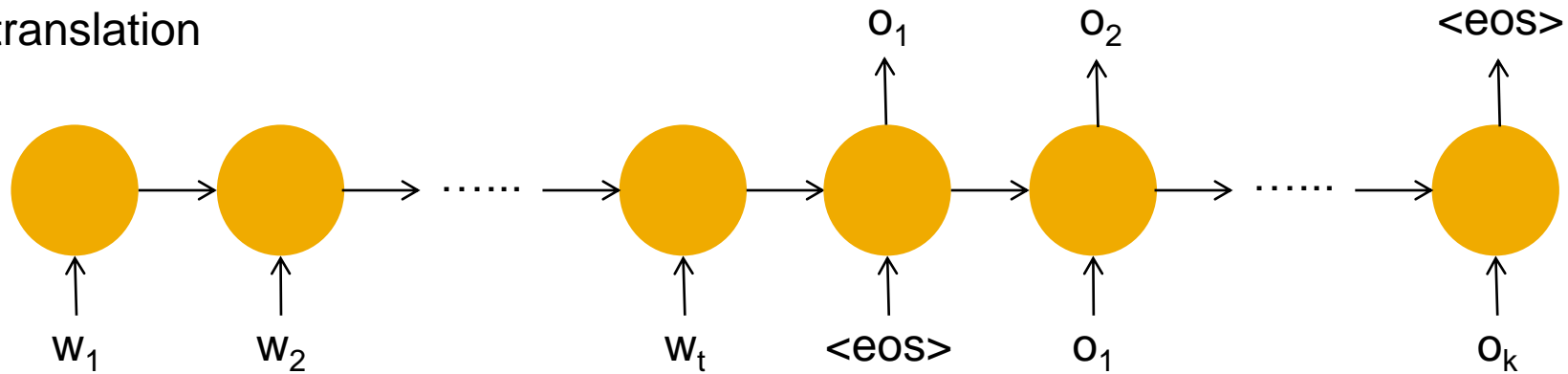
- Machine translation



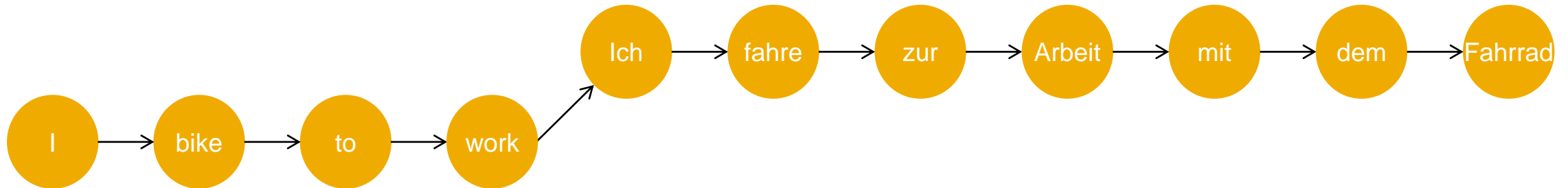
# Introduction to Sequence Models

Other applications of sequence models – Sequence to sequence

- Machine translation



- Example: English to German translation





# Introduction to Sequence Models

## Word representations

- Words need to be represented numerically, as dense vectors, for easy computation
- The representation needs to capture semantic information about the words
- How to achieve this is explained in the next lecture



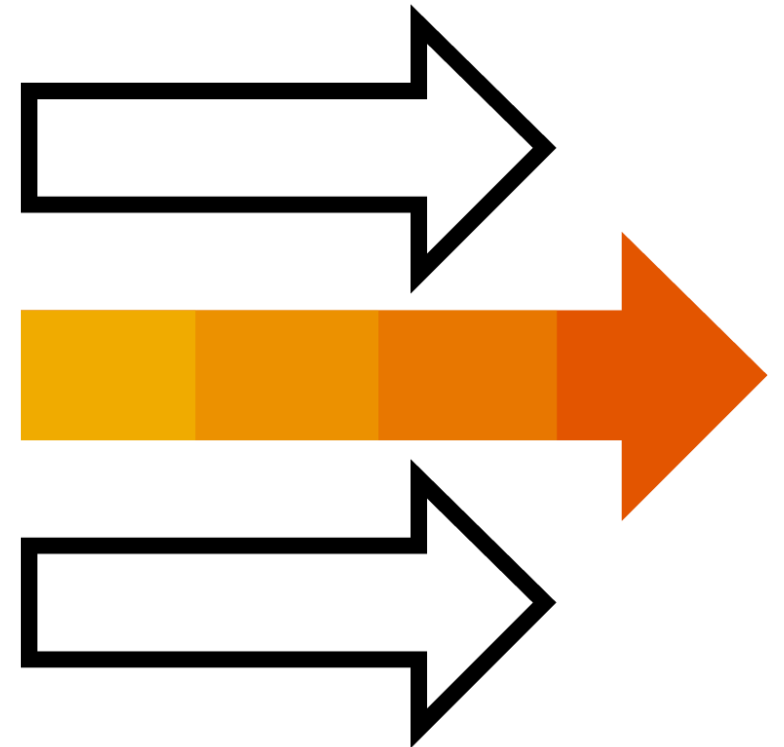
# Introduction to Sequence Models

Coming up next

Vector representations of words

Distributed representations

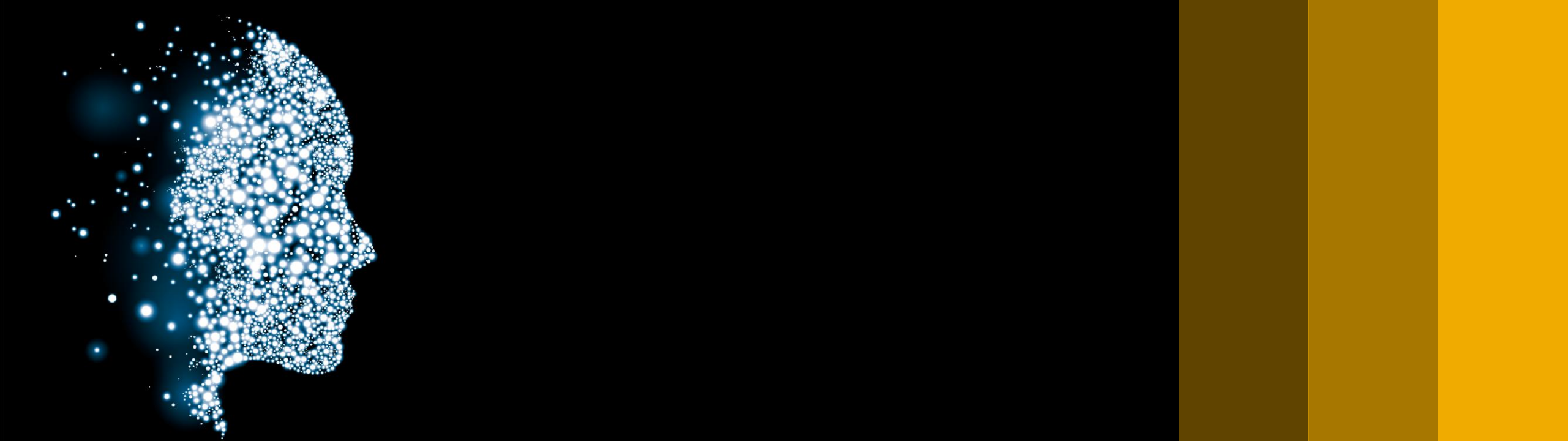
Word2Vec



# Thank you.

**Contact information:**

**open@sap.com**



Week 3: Deep Networks and Sequence Models

## Unit 3: Representation Learning for NLP

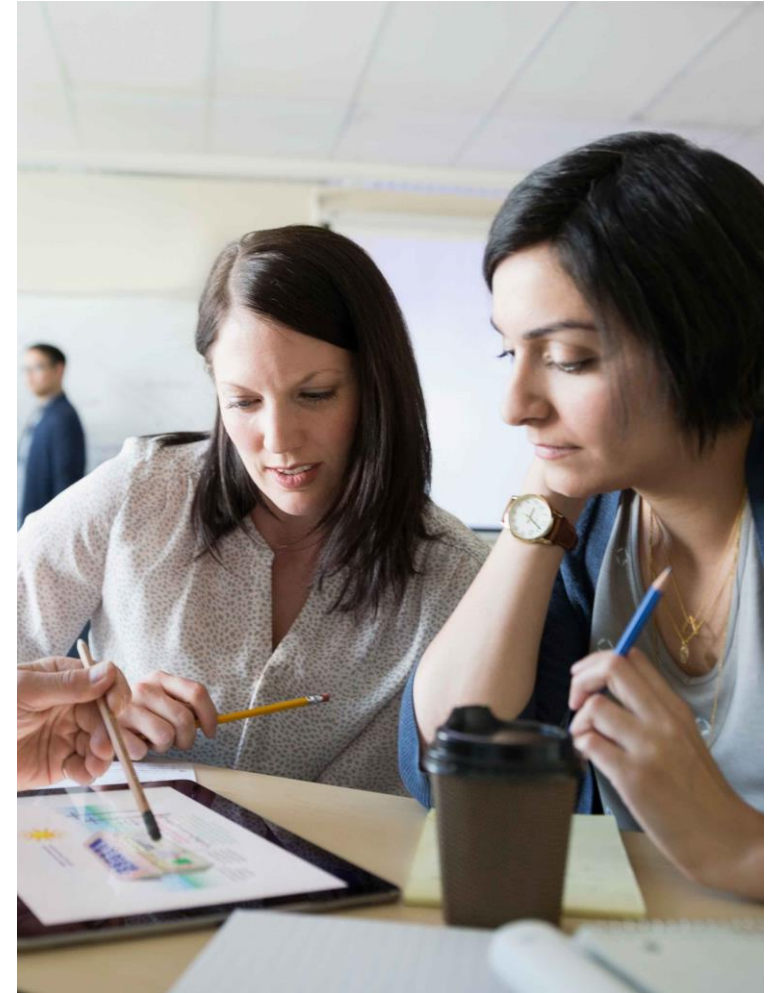
# Representation Learning for NLP

What we covered in the last unit

Sequential data

Sequence models

Applications of sequence models

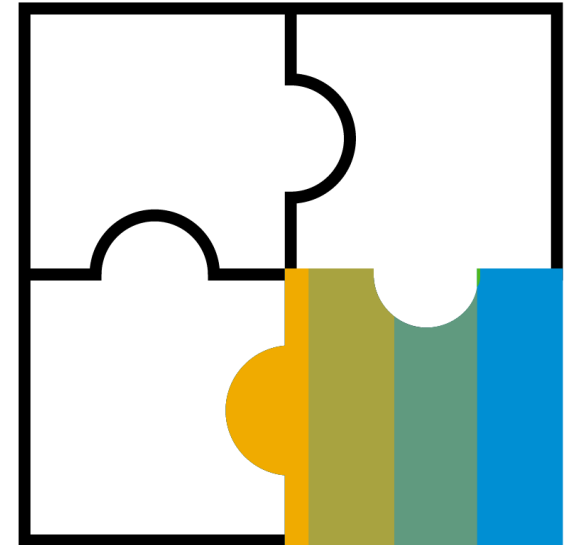


# Representation Learning for NLP

## Overview

### Content:

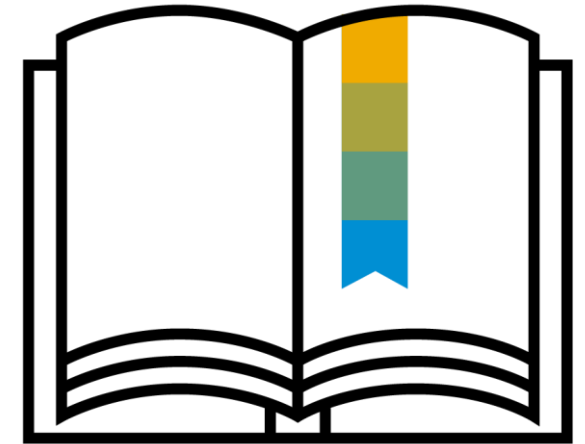
- Vector representations of words
- Distributed representations
- Word2Vec



# Representation Learning for NLP

How do we represent natural language?

- Machine learning applications in natural language processing (NLP) are, for example,
  - Text classification
  - Language modeling
  - Sentiment analysis
  - Machine translation
  - Document summarization
  - Caption generation
- NLP tasks require a meaningful representation of text as input
- An input text can be thought of as a sequence where the units composing the sequence can either be characters, words, or even full documents



# Representation Learning for NLP

Vector representation of words

## One-hot encoding

- Definition: Given a vocabulary  $V$ , every word is converted to a vector in  $\mathbb{R}^{|V| \times 1}$  that is 0 for all indices but 1 at the index of the respective word represented

- Example:

*“Everybody likes dogs and hates bananas.”*

| Everybody | likes | dogs |
|-----------|-------|------|
| 1         | 0     | 0    |
| 0         | 0     | 1    |
| 0         | 1     | 0    |
| ...       | ...   | ...  |
| ...       | ...   | ...  |
| ...       | ...   | ...  |
| 0         | 0     | 0    |



# Representation Learning for NLP

Vector representation of words

## One-hot representation

- Straightforward implementation
- Meaning of word is not encoded in representation
- Vector dimensionality scales with vocabulary size

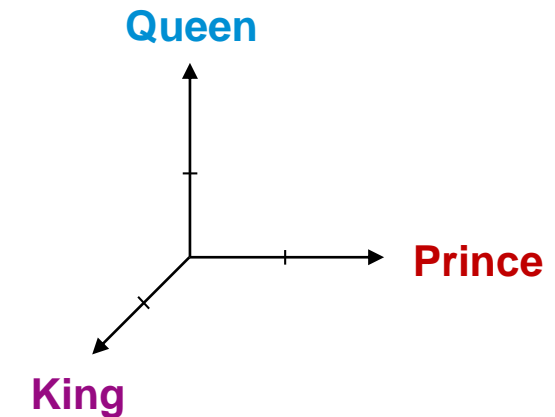
$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}^T \times \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = 0$$

## Easy implementation

- Straightforward implementation

## Missing semantics

- Meaning of word is not encoded in representation



## High dimensionality

- Vector dimensionality scales with vocabulary size

# Representation Learning for NLP

Vector representation of words

## Distributed Representations

- Instead of storing all information in one dimension, we distribute the meaning across a fixed number of dimensions
- Encodes semantic and syntactic features of words
- Reduces the necessary number of dimensions
- Wouldn't it be great to have an algorithm that learns those representations from unstructured text?

|                    | Queen | Woman | King | Prince |     |
|--------------------|-------|-------|------|--------|-----|
| <i>Femininity</i>  | 0.99  | 0.99  | 0.01 | 0.02   |     |
| <i>Masculinity</i> | 0.01  | 0.01  | 0.99 | 0.92   |     |
| <i>Royalty</i>     | 0.99  | 0.99  | 0.99 | 0.81   | ... |
| <i>Age</i>         | 0.67  | 0.53  | 0.75 | 0.22   |     |

# Representation Learning for NLP

Distributed representation of words

## Representations of words based on distributional similarity

*“You shall know a word by the company it keeps”  
(Firth, 1957)*

- Represent a word based on neighboring words
- Word representations are defined through their context
- Words occurring in similar contexts should have similar representations



My neighbor has trees in his yard.

The diagram illustrates word co-occurrence in the sentence "My neighbor has trees in his yard." Each word is highlighted in a blue box. Yellow curved arrows connect the word "My" to "neighbor", "neighbor" to "has", "has" to "trees", "trees" to "in", "in" to "his", and "his" to "yard", showing the sequential context of the words.

My neighbor has bushes in his yard.

The trunks of trees have many rings.

~~The trunks of bushes have many rings.~~

# Representation Learning for NLP

Word2Vec: skip-gram

## Skip-Gram Model

- For each word in a corpus, predict the neighboring words in a context window of length  $c$

Everybody likes dogs and hates bananas.

Everybody likes dogs and hates bananas.

Everybody likes dogs and hates bananas.

$V$ : Vocabulary size

$x^i$ : one-hot vector for word  $i$  in  $V$

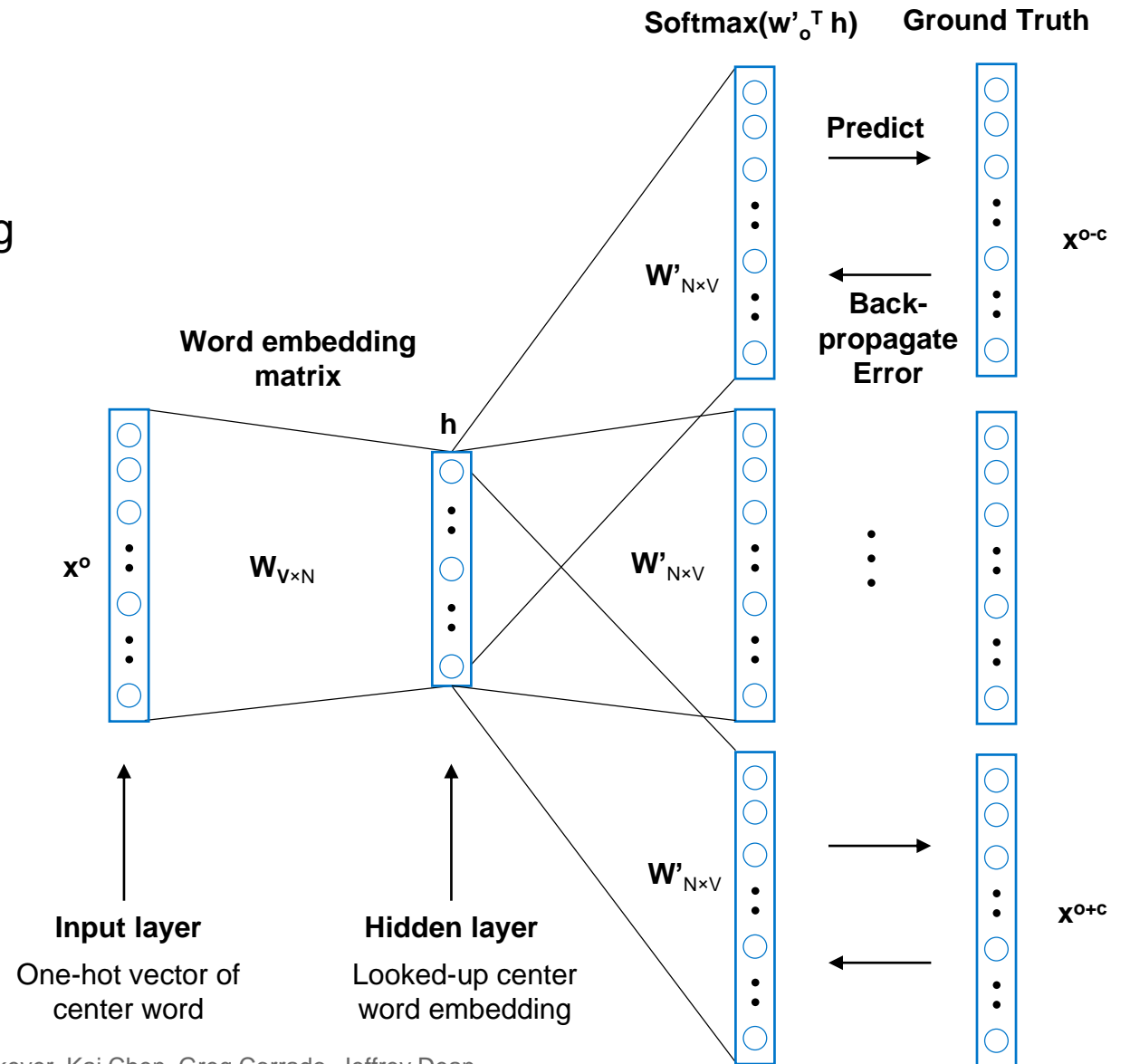
$N$ : size of embedding space

$W$ : Word embedding matrix  $W \in V \times N$

$w$ : Word embedding  $w \in 1 \times N$

$W'$ : Word embedding matrix  $W' \in N \times V$

$C$ : Context window



# Representation Learning for NLP

Word2Vec: CBOW

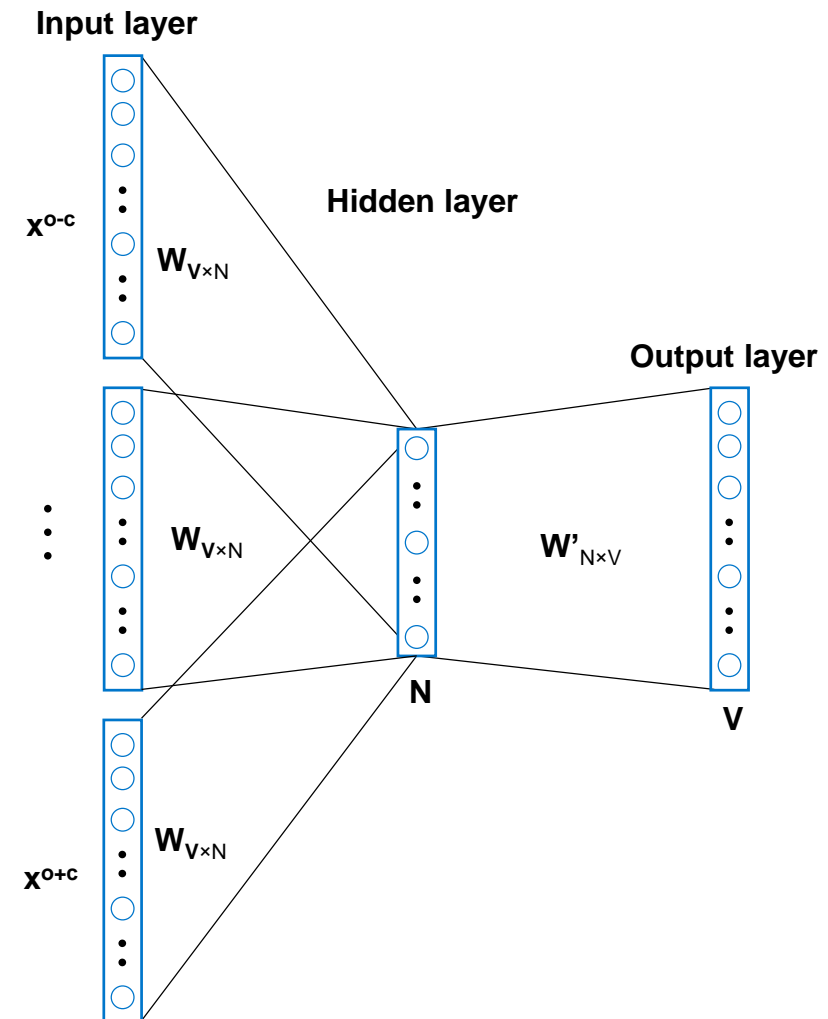
## Continuous Bag of Words Model

- Predict the output word based on the neighboring words in a context window of length  $c$

Everybody likes dogs and hates bananas.

Everybody likes dogs and hates bananas.

Everybody likes dogs and hates bananas.



# Representation Learning for NLP

Word2Vec captures similarity of words...

## Word similarity

- Closest vectors in terms of (cosine) distance according to GoogleNews-vectors (trained on about 100 billion words)
- Excluded plurals for illustration

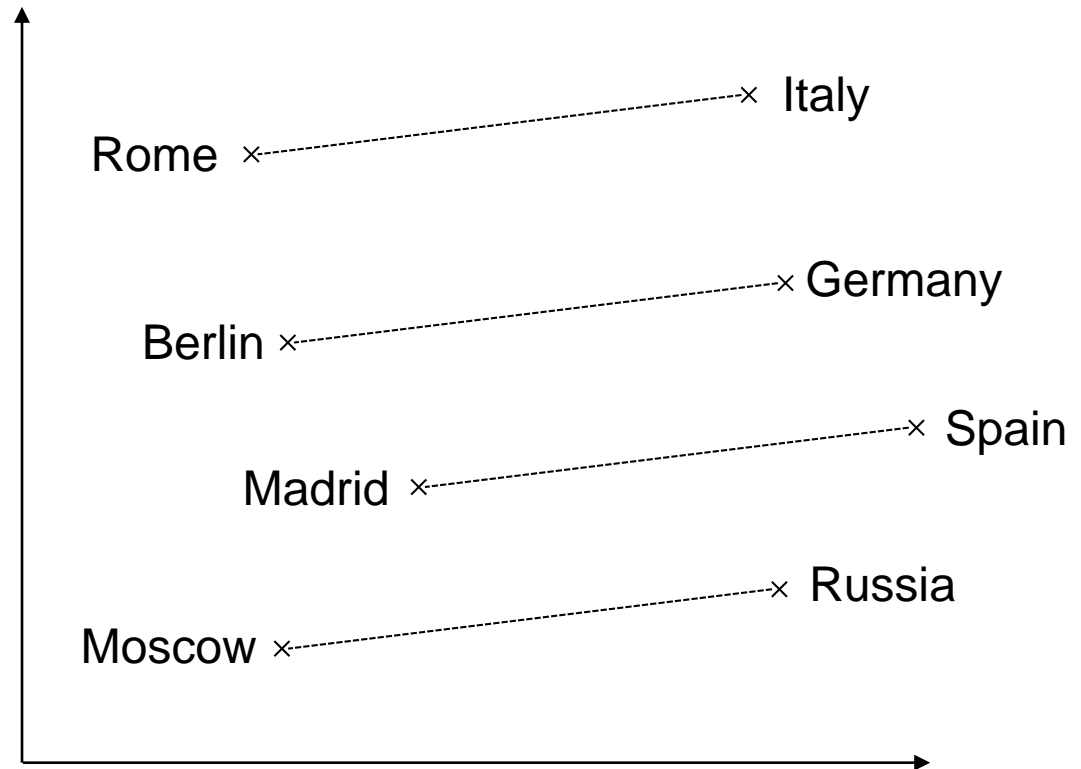
| red    | reddish   | banana    | two   | parrot    | PS4      |
|--------|-----------|-----------|-------|-----------|----------|
| yellow | brownish  | pineapple | three | parrots   | PSP2     |
| blue   | yellowish | mango     | four  | macaw     | SONY NGP |
| purple | pinkish   | papaya    | five  | parakeet  | Wii2     |
| orange | grayish   | coconut   | six   | cockatiel | PS3      |

# Representation Learning for NLP

...and beyond

## Word analogies

- Trained on a large corpus, the distance of distributed representations encodes certain semantic concepts
- This can be e.g. gender, capital city
- Rome relates to Italy, as Berlin relates to Germany
- This applies for different languages



$v(\text{Rome}) - v(\text{Italy}) + v(\text{Berlin}) \sim$

$v(\text{King}) - v(\text{Man}) + v(\text{Woman}) \sim$

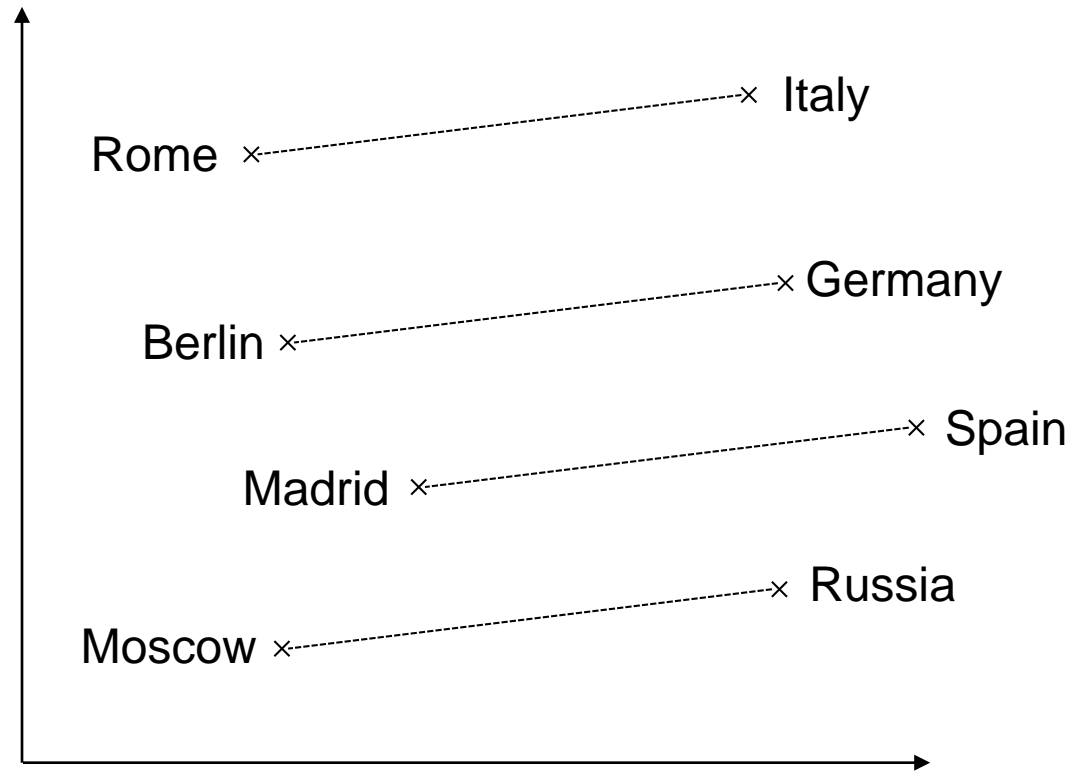
$v(\text{Building}) - v(\text{Architect}) + v(\text{Software}) \sim$

# Representation Learning for NLP

...and beyond

## Word analogies

- Trained on a large corpus, the distance of distributed representations encodes certain semantic concepts
- This can be e.g. gender, capital city
- Rome relates to Italy, as Berlin relates to Germany
- This applies for different languages



$$\begin{aligned} \mathbf{v}(\text{Rome}) - \mathbf{v}(\text{Italy}) + \mathbf{v}(\text{Berlin}) &\sim \mathbf{v}(\text{Germany}) \\ \mathbf{v}(\text{King}) - \mathbf{v}(\text{Man}) + \mathbf{v}(\text{Woman}) &\sim \\ \mathbf{v}(\text{Building}) - \mathbf{v}(\text{Architect}) + \mathbf{v}(\text{Software}) &\sim \end{aligned}$$

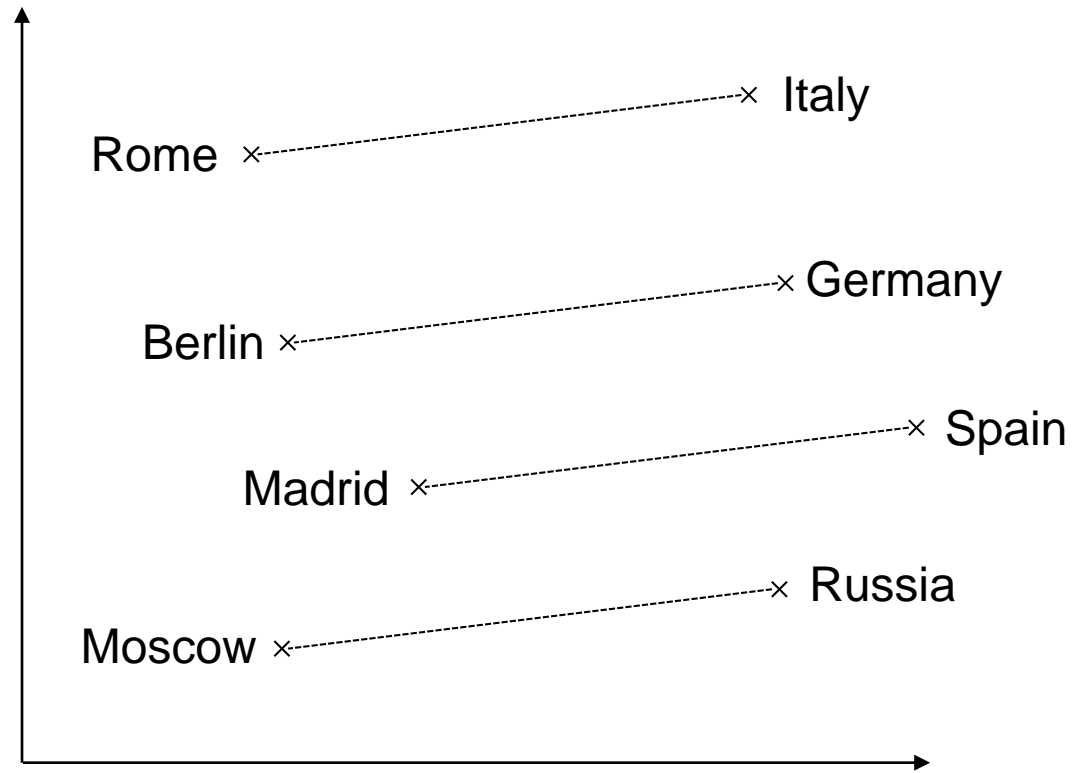


# Representation Learning for NLP

...and beyond

## Word analogies

- Trained on a large corpus, the distance of distributed representations encodes certain semantic concepts
- This can be e.g. gender, capital city
- Rome relates to Italy, as Berlin relates to Germany
- This applies for different languages



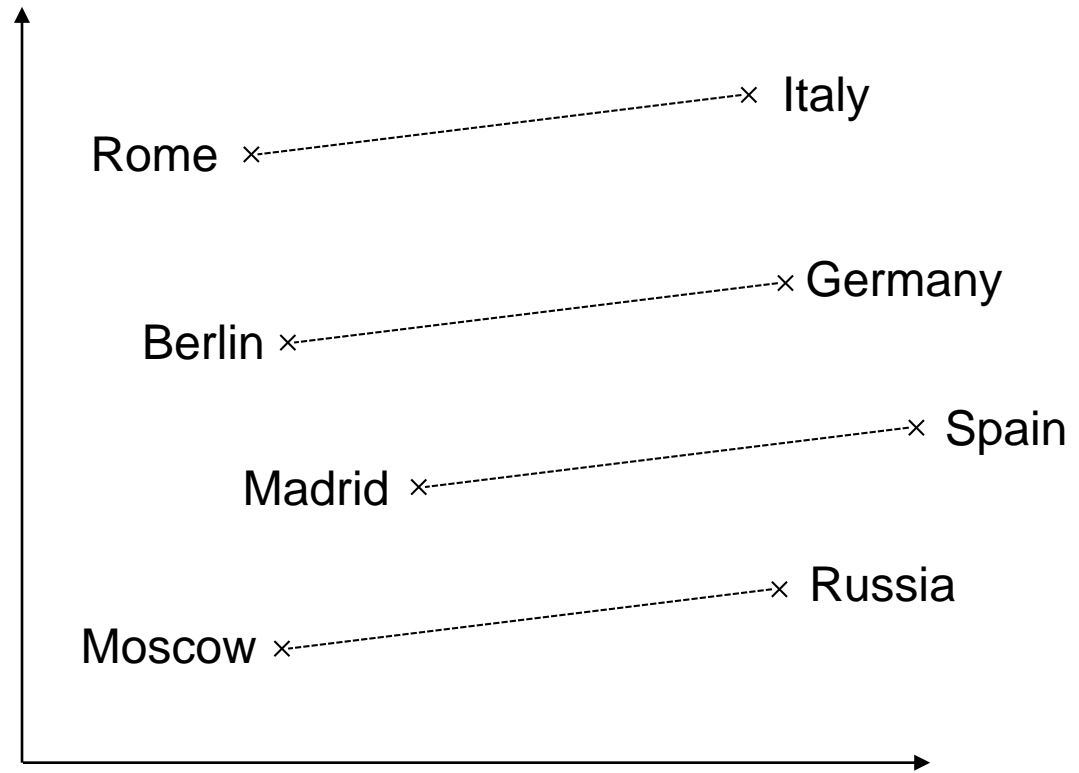
$$\begin{aligned} v(\text{Rome}) - v(\text{Italy}) + v(\text{Berlin}) &\sim v(\text{Germany}) \\ v(\text{King}) - v(\text{Man}) + v(\text{Woman}) &\sim v(\text{Queen}) \\ v(\text{Building}) - v(\text{Architect}) + v(\text{Software}) &\sim \end{aligned}$$

# Representation Learning for NLP

...and beyond

## Word analogies

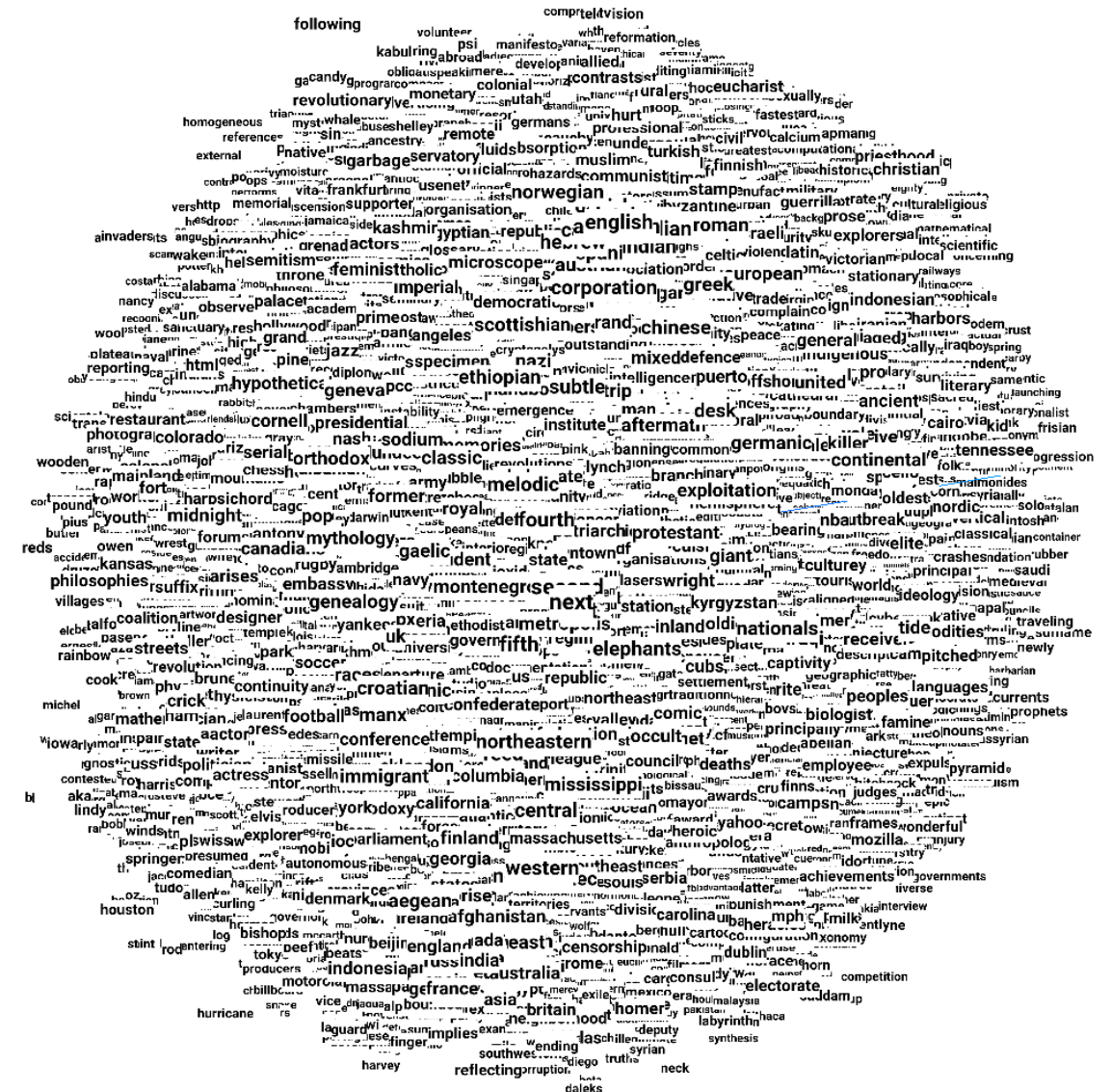
- Trained on a large corpus, the distance of distributed representations encodes certain semantic concepts
- This can be e.g. gender, capital city
- Rome relates to Italy, as Berlin relates to Germany
- This applies for different languages



$$\begin{aligned} \mathbf{v}(\text{Rome}) - \mathbf{v}(\text{Italy}) + \mathbf{v}(\text{Berlin}) &\sim \mathbf{v}(\text{Germany}) \\ \mathbf{v}(\text{King}) - \mathbf{v}(\text{Man}) + \mathbf{v}(\text{Woman}) &\sim \mathbf{v}(\text{Queen}) \\ \mathbf{v}(\text{Building}) - \mathbf{v}(\text{Architect}) + \mathbf{v}(\text{Software}) &\sim \mathbf{v}(\text{Programmer}) \end{aligned}$$

# Vector representation for words

- Show visualization of embeddings in TensorBoard, either by t-SNE or PCA
- 10,000 word vectors with dimensionality of 128
- Reduced to  $d = 3$  for visualization



# Representation Learning for NLP

## Outlook

### Adaptations of the Word2Vec algorithm

- Doc2Vec
- DNA2Vec
- Product2Vec
- App2Vec
- Emoji2Vec

Word2Vec and its implementations are versatile algorithms for creating fixed-size embeddings of input features

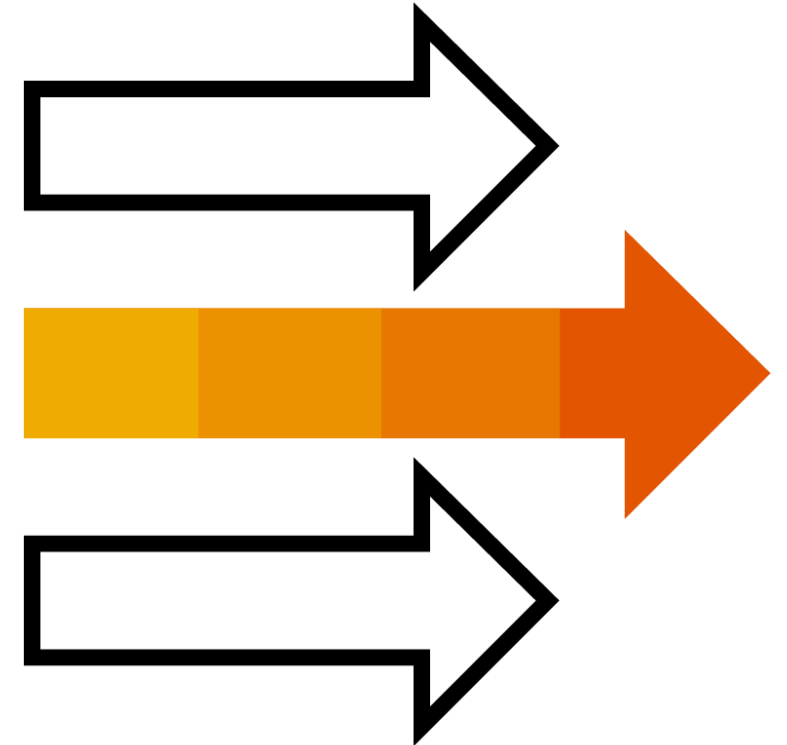


# Representation Learning for NLP

Coming up next

## Basic Recurrent Neural Networks (RNNs)

- Types of sequences
- Basic RNN cell structure
- Problems with training RNNs



# Thank you.

**Contact information:**

**open@sap.com**

© 2017 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company.

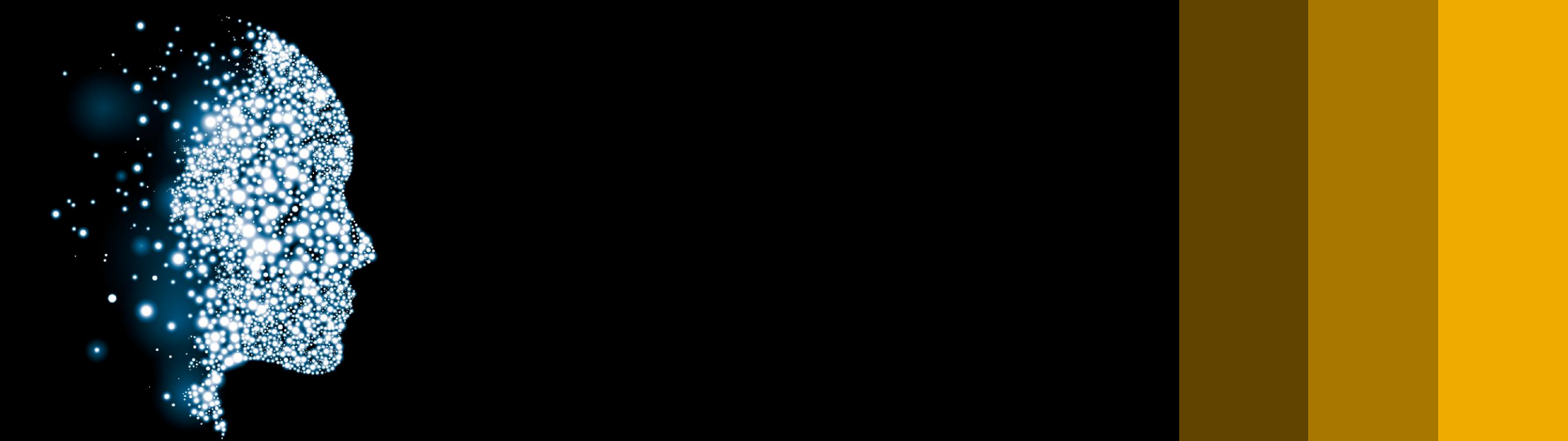
The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.

See <http://global.sap.com/corporate-en/legal/copyright/index.epx> for additional trademark information and notices.



Week 3: Deep Networks and Sequence Models

## Unit 4: Basic RNNs in TensorFlow

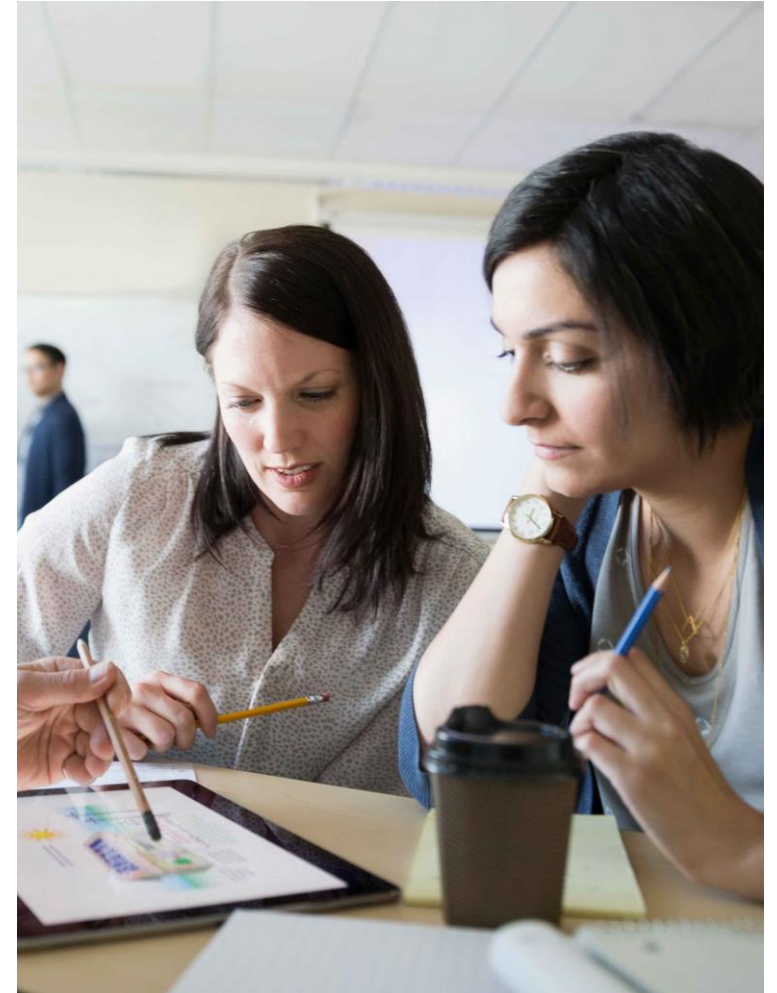


# Basic RNNs in TensorFlow

What we covered in the last unit

## Representation Learning for NLP

- Vector representation of words
- Distributed representations

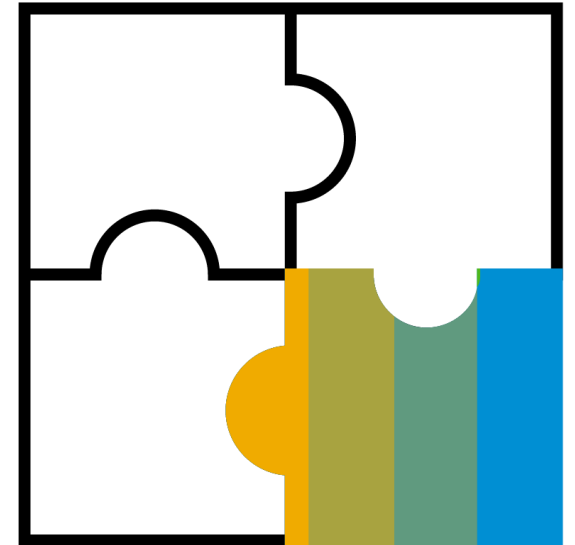


# Basic RNNs in TensorFlow

## Overview

### Content:

- Recap: Sequence types
- Basic RNN cells
- Training RNNs



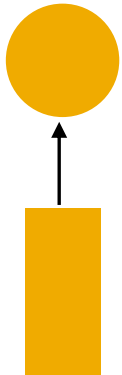
# Basic RNNs in TensorFlow

Sequence types

Example:

Height

Parent's height  $\rightarrow$  Child's height



**One-to-One**

# Basic RNNs in TensorFlow

Sequence types

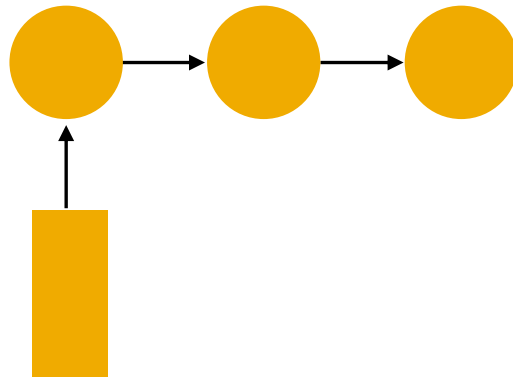
Example:

Image captioning

Image  $\rightarrow$  Sequence of words



**One-to-One**



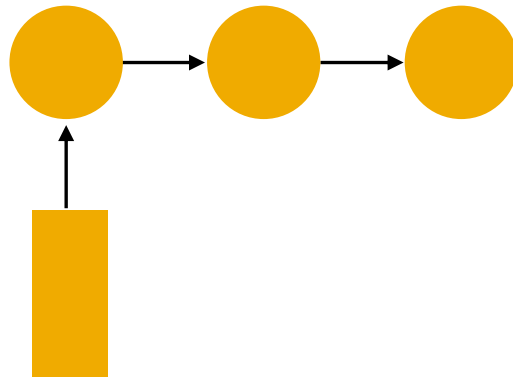
**One-to-Many**

# Basic RNNs in TensorFlow

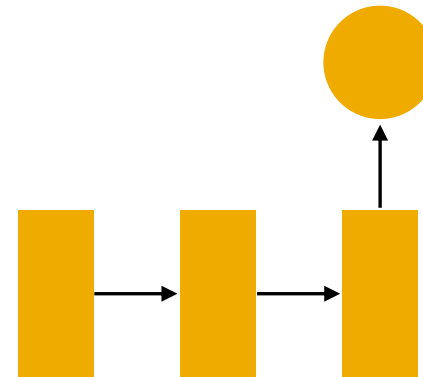
Sequence types



**One-to-One**



**One-to-Many**



**Many-to-One**

Example:

Sentiment Analysis

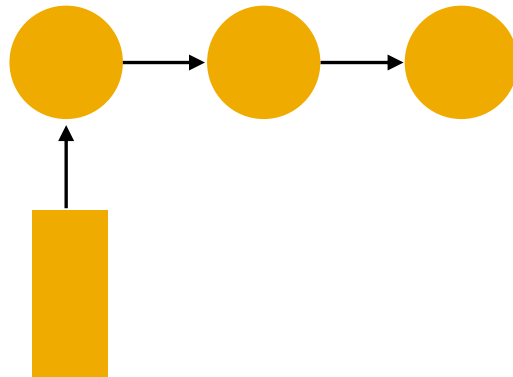
Sequence of words  $\rightarrow$  Sentiment

# Basic RNNs in TensorFlow

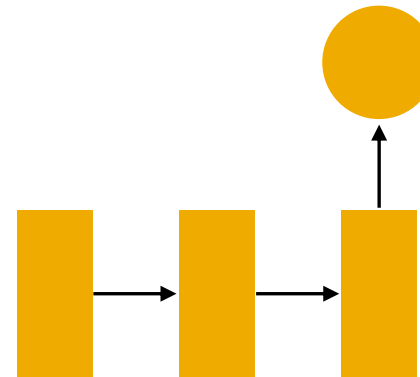
Sequence types



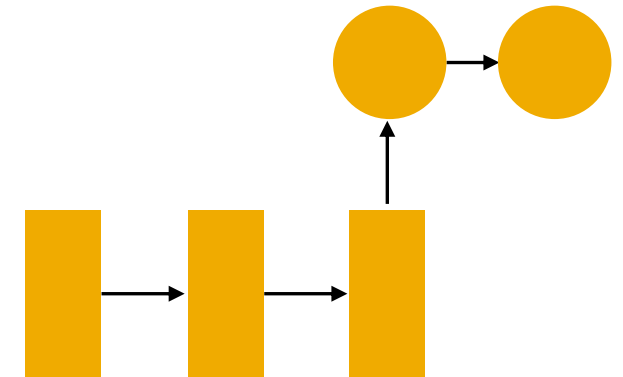
One-to-One



One-to-Many



Many-to-One

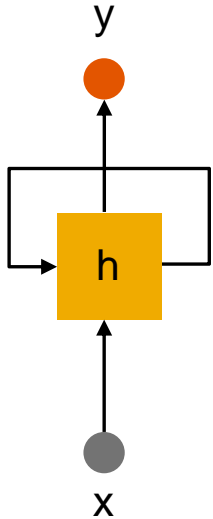


Many-to-Many

Example:  
Machine Translation  
Sequence of words →  
Sequence of words

# Basic RNNs in TensorFlow

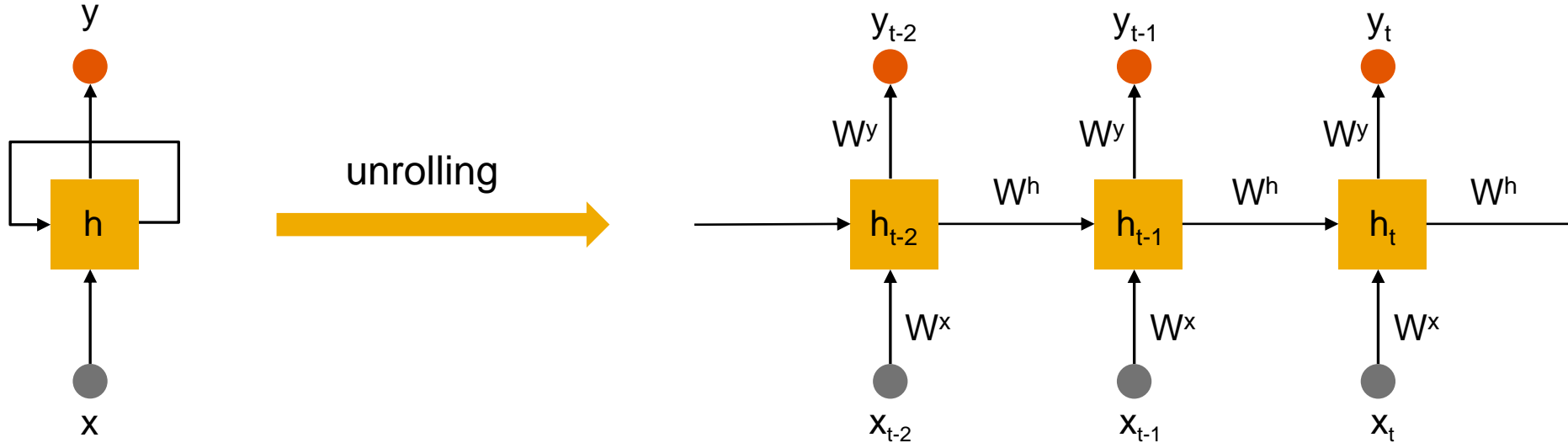
Understanding recurrent neural networks (RNNs)



- *'Recurrent'* implies that weights are shared across time steps

# Basic RNNs in TensorFlow

Understanding recurrent neural networks (RNNs)

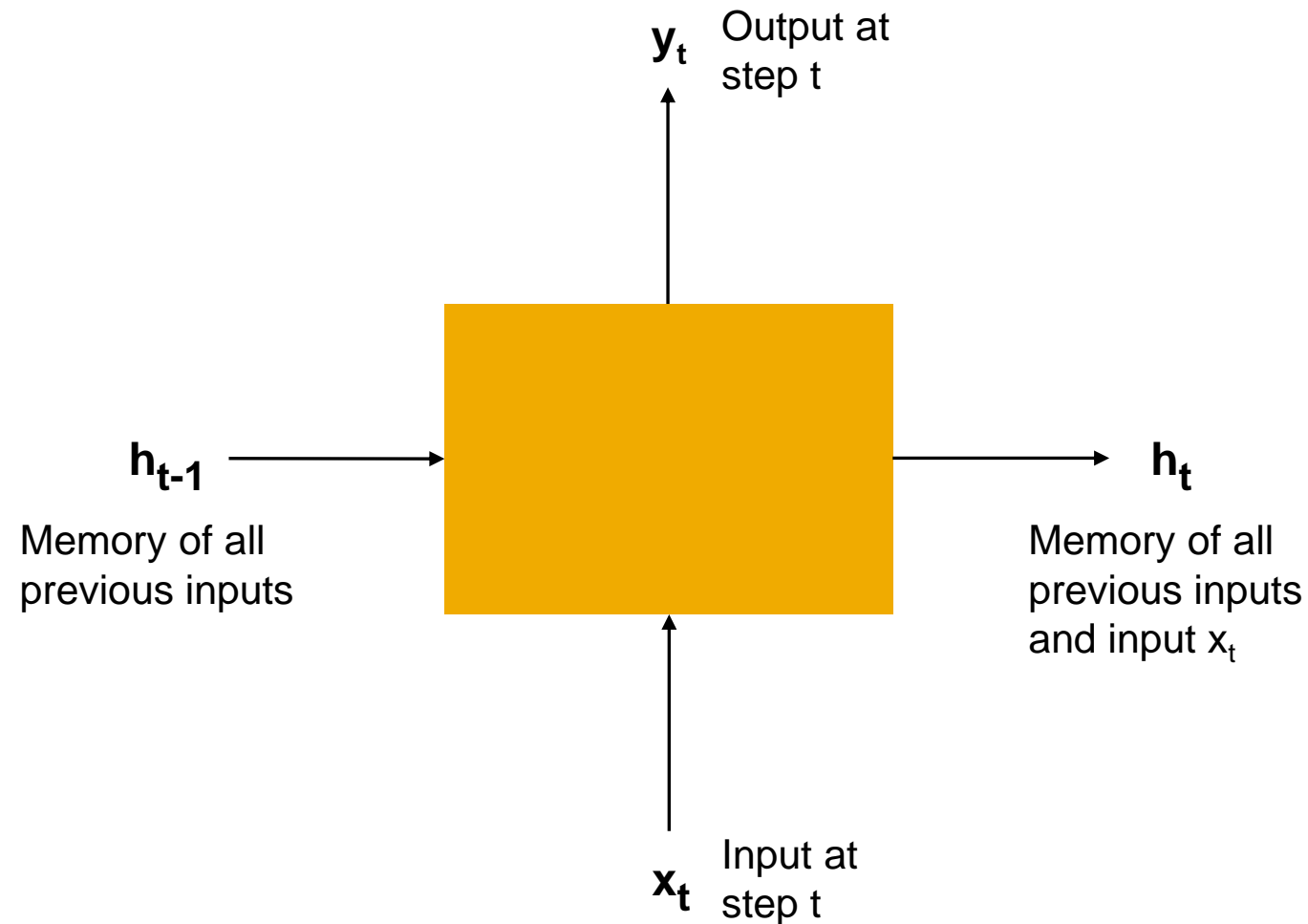


- *'Recurrent'* implies that weights are shared across time steps
- The network is unrolled in time
- The same weight matrix is applied at each time step



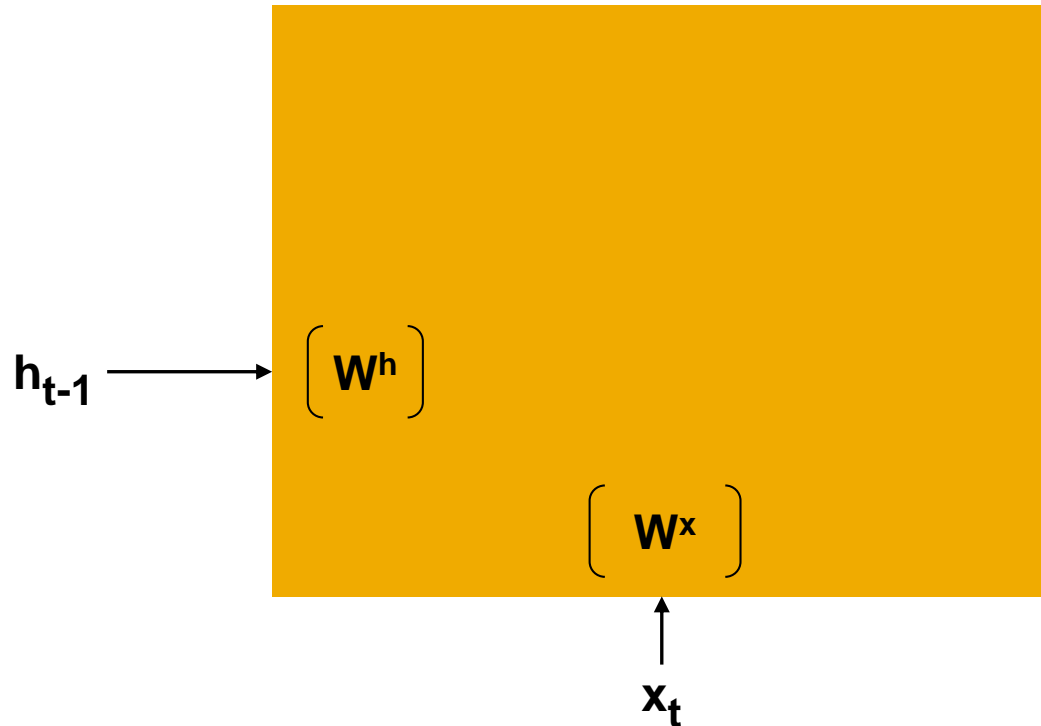
# Basic RNNs in TensorFlow

## Understanding an RNN cell



# Basic RNNs in TensorFlow

## Understanding an RNN cell



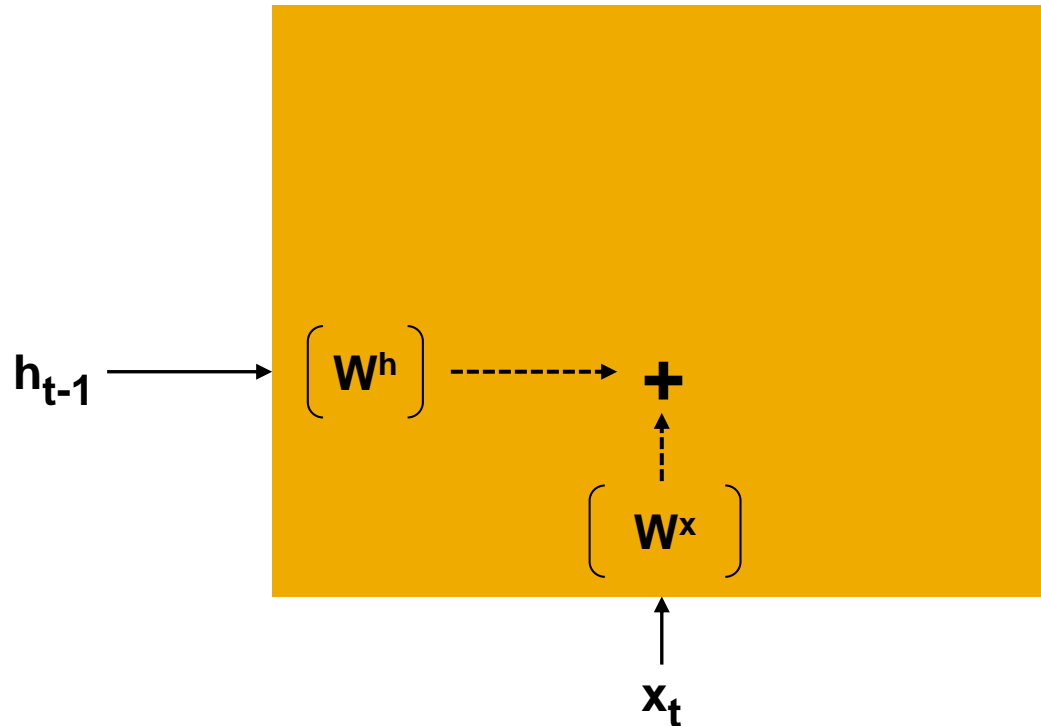
- $x_t$  = input at step  $t$
- $h_{t-1}$  = memory of all previous inputs until  $t-1$
- $W^h$  = weight parameters for hidden state  $h_t$
- $W^x$  = weight parameters for input  $x_t$

- Calculation of the hidden state  $h_t$ :

$$W^h h_{t-1} + W^x x_t$$

# Basic RNNs in TensorFlow

## Understanding an RNN cell



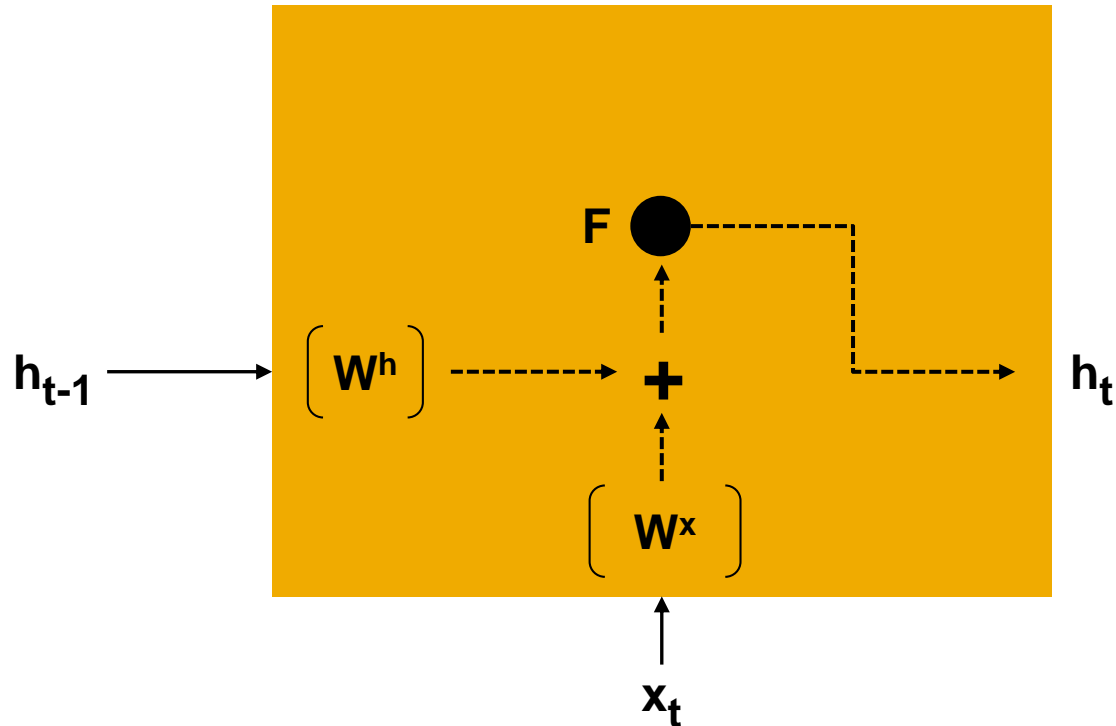
- $x_t$  = input at step  $t$
- $h_{t-1}$  = memory of all previous inputs until  $t-1$
- $W^h$  = weight parameters for hidden state  $h_t$
- $W^x$  = weight parameters for input  $x_t$

- Calculation of the hidden state  $h_t$ :

$$W^h h_{t-1} + W^x x_t$$

# Basic RNNs in TensorFlow

## Understanding an RNN cell



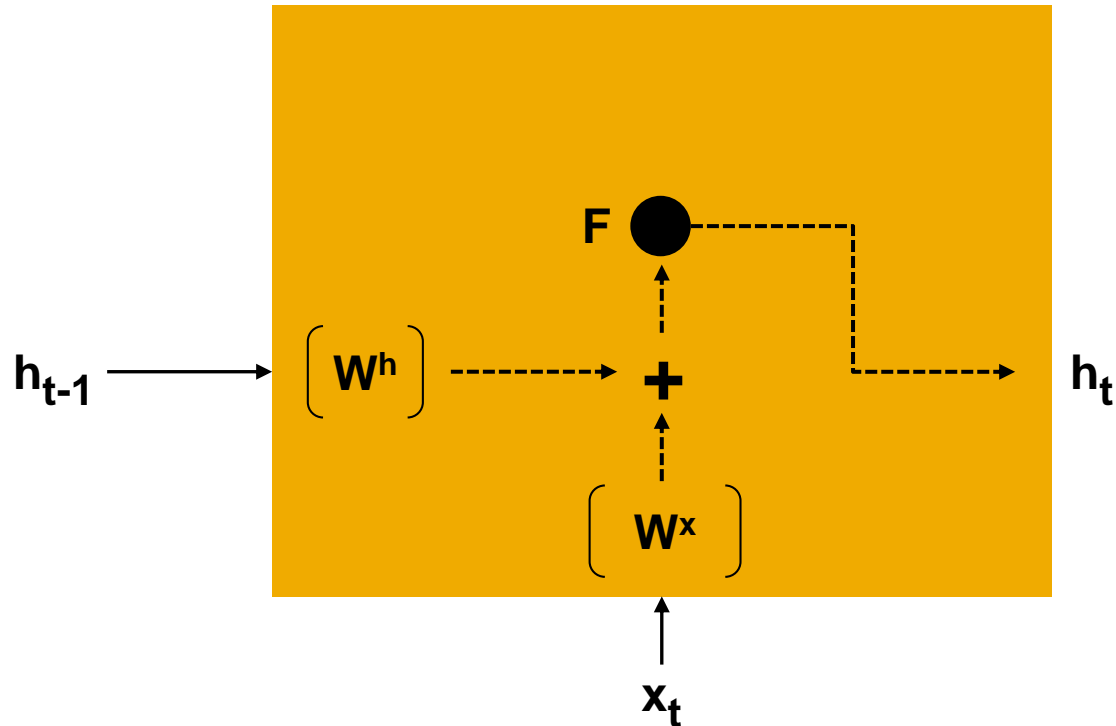
- $x_t$  = input at step  $t$
- $h_{t-1}$  = memory of all previous inputs until  $t-1$
- $W^h$  = weight parameters for hidden state  $h_t$
- $W^x$  = weight parameters for input  $x_t$
- $F$  = activation function e.g. tanh

- Calculation of the hidden state  $h_t$ :

$$h_t = F(W^h h_{t-1} + W^x x_t)$$

# Basic RNNs in TensorFlow

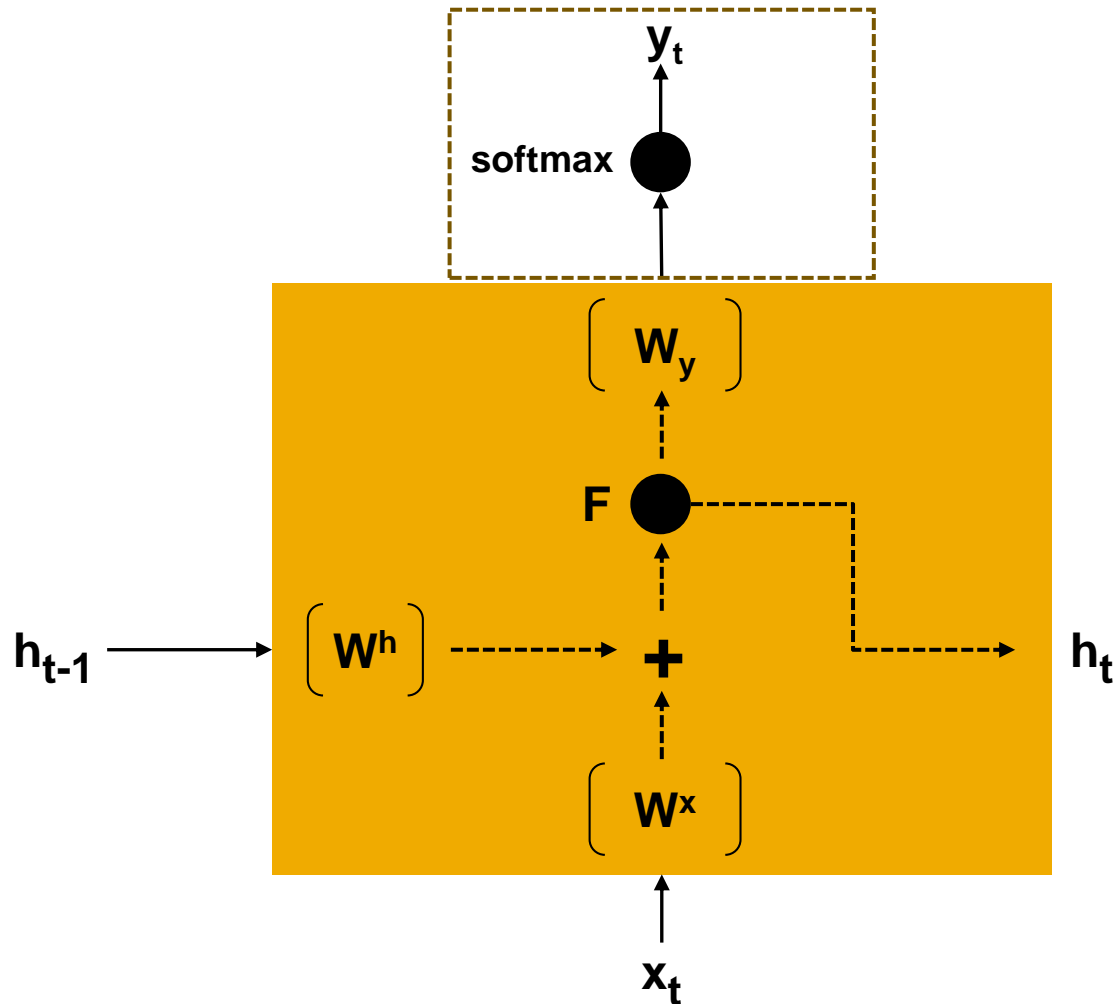
## Understanding an RNN cell



- $x_t$  = input at step  $t$
  - $h_{t-1}$  = memory of all previous inputs until  $t-1$
  - $W^h$  = weight parameters for hidden state  $h_t$
  - $W^x$  = weight parameters for input  $x_t$
  - $F$  = activation function e.g. tanh
- 
- Apply the same operation at each time step:  
$$h_t = F(W^h h_{t-1} + W^x x_t)$$
$$= F(W^h F(W^h h_{t-2} + W^x x_{t-1}) + W^x x_t)$$
$$\dots$$

# Basic RNNs in TensorFlow

## Understanding an RNN cell



- $x_t$  = input at step  $t$
- $h_{t-1}$  = memory of all previous inputs until  $t-1$
- $W^h$  = weight parameters for hidden state  $h_t$
- $W^x$  = weight parameters for input  $x_t$
- $F$  = activation function e.g.  $\tanh$
- $W^y$  = weight parameters for hidden  $\rightarrow$  output
- $y_t$  = output at step  $t$

- Calculation of the hidden state  $h_t$ :

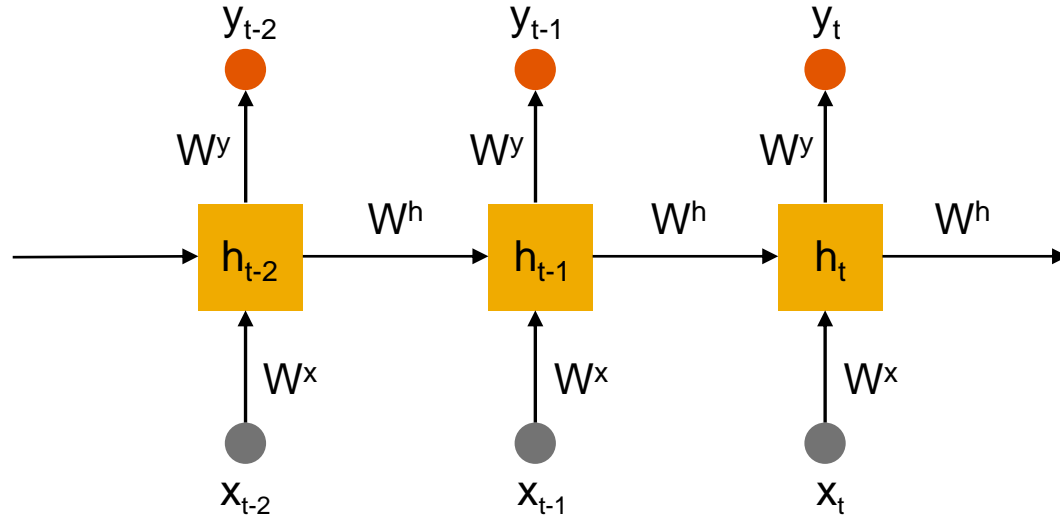
$$h_t = F(W^h h_{t-1} + W^x x_t)$$

- Calculation of the output  $y_t$ :

$$y_t = \text{softmax}(W^y h_t)$$

# Basic RNNs in TensorFlow

## Understanding an RNN cell

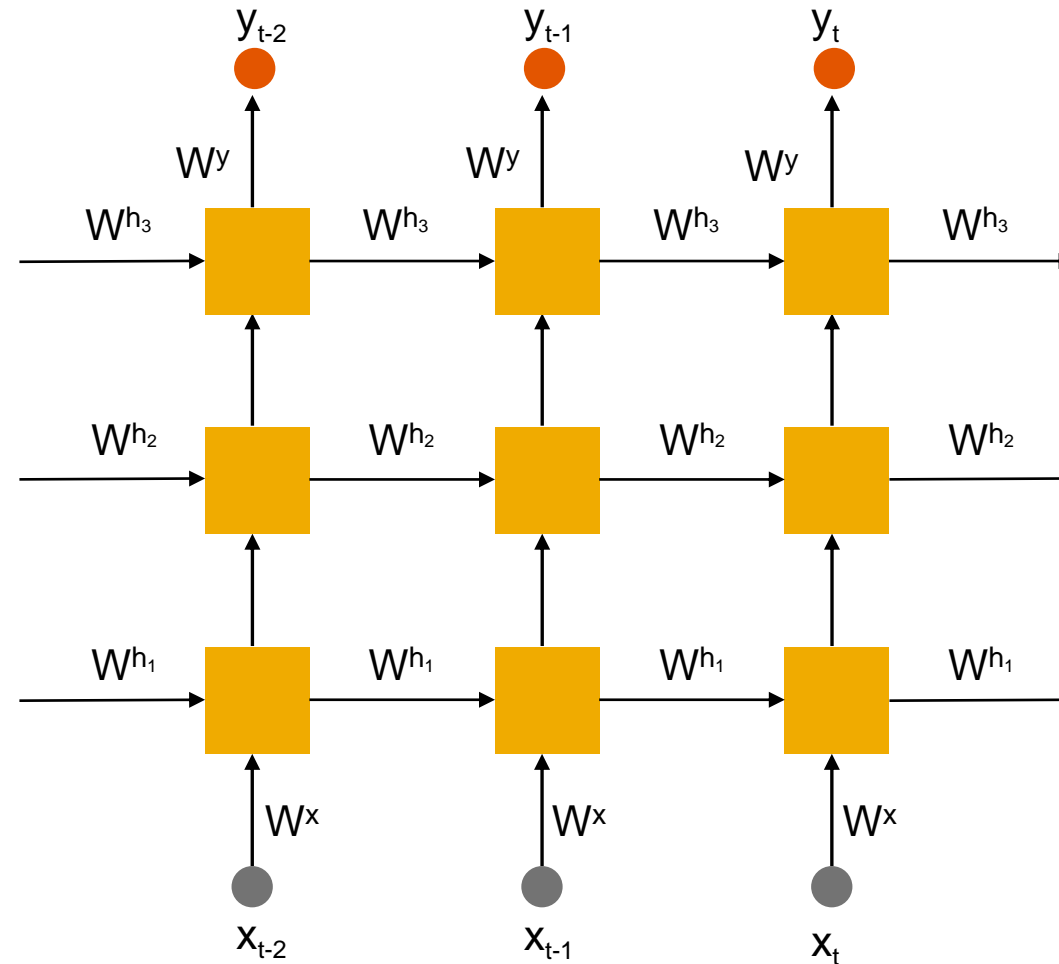


### Takeaway

- We use the same weight parameters for all  $t$
- Output depends on all previous inputs ( $h_{t-1}$ ) and the current input  $x_t$
- Calculating the output means matrix multiplication of the same matrices over and over again

# Basic RNNs in TensorFlow

Getting deep with RNNs





# Basic RNNs in TensorFlow

## Training of RNNs

Example problem: Predict the last word in a sentence

### Case 1: Short sequence:

Tom was cooking pasta. Jenny walked in. Both ate *pasta*.

### Case 2: Long sequence:

Tom was cooking pasta in the kitchen. Jenny walked in. They talked about what happened during the day. Then she asked him what he was cooking. Tom replied that he was cooking *pasta*.

# Basic RNNs in TensorFlow

## Training of RNNs

Example problem: Predict the last word in a sentence

**Case 1:** Short sequence:

Tom was cooking pasta. Jenny walked in. Both ate *pasta*.

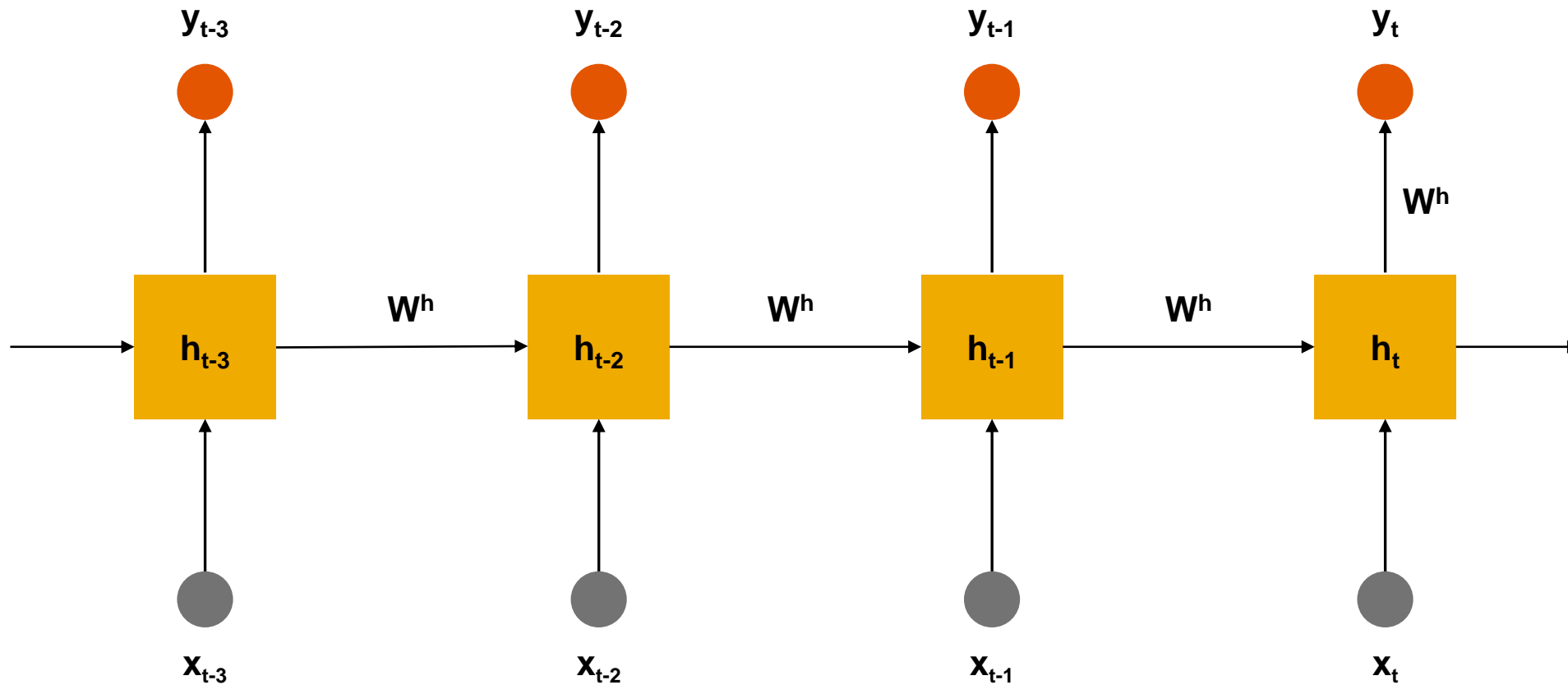
**Case 2:** Long sequence:

Tom was cooking pasta in the kitchen. Jenny walked in. They talked about what happened during the day. Then she asked him what he was cooking. Tom replied that he was cooking *pasta*.

*Vanilla RNNs are poor at modeling long-term dependency*

# Basic RNNs in TensorFlow

## Training of RNNs



# Basic RNNs in TensorFlow

## Vanishing gradient problem in detail

- Our main problem: The same operation (matrix multiplication followed by activation function) is repeated over and over

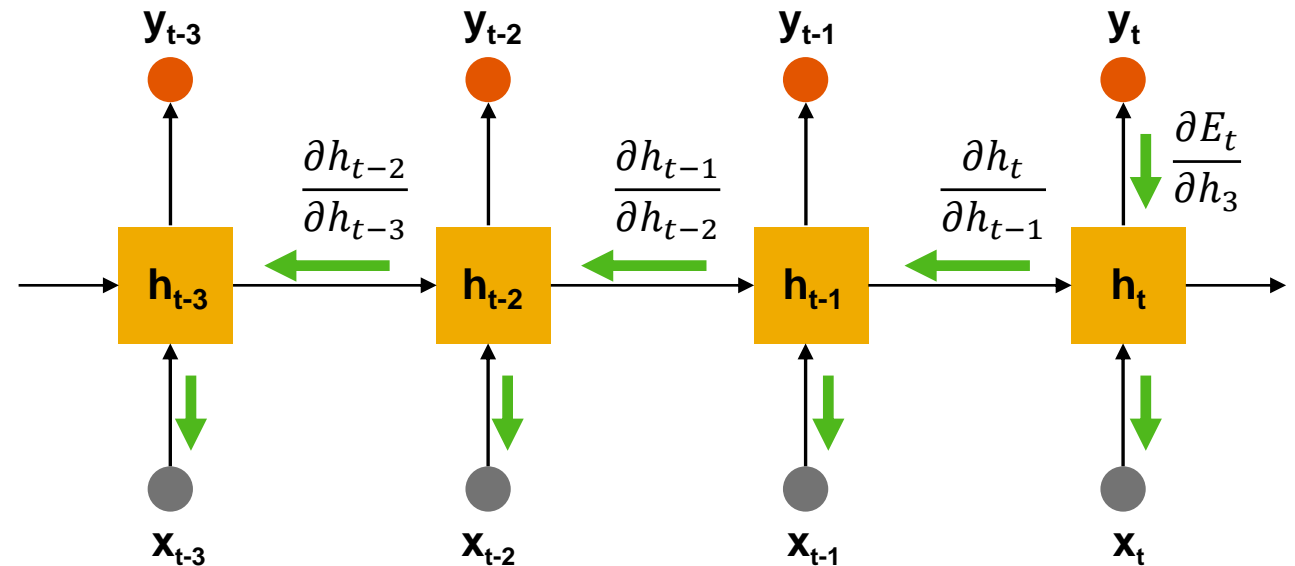
- Example: A Simpler RNN without nonlinear activation and input:

$$h_t \sim W^h h_{t-1}$$

$$h_t \sim (W^h)^t h_0$$

- with t as our last time step

**Gradient:** Rate of change of cost function with respect to each parameter



# Basic RNNs in TensorFlow

## Vanishing gradient problem in detail

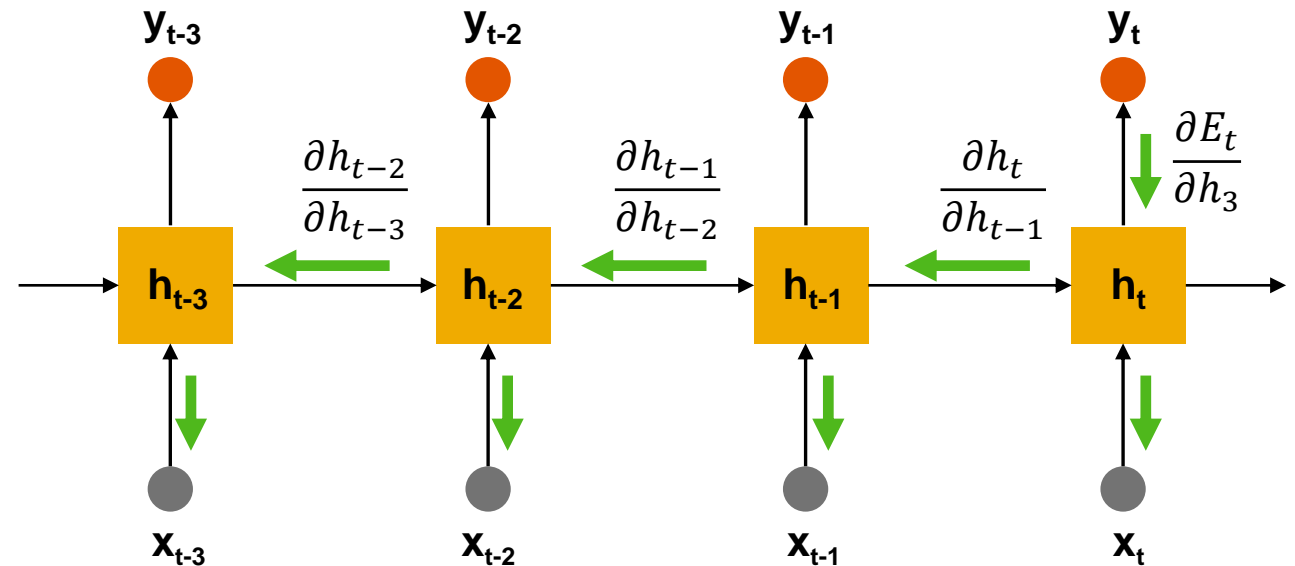
- We can factorize  $W$  via eigendecomposition:

$$W^h = Q\Lambda Q^{-1}$$

$$h_t \sim Q\Lambda^t Q^{-1}$$

- Eigenvalues are raised to the power of  $t$
- Any eigenvalue smaller than one becomes zero
- Any eigenvalue larger than one will explode

**Gradient:** Rate of change of cost function with respect to each parameter



# Basic RNNs in TensorFlow

Vanishing gradient problem in detail

- The same applies for backpropagating the error:

Case 1: **Exploding gradient**



# Basic RNNs in TensorFlow

## Vanishing gradient problem in detail

- The same applies for backpropagating the error:

Case 1: **Exploding gradient** → **Gradient clipping**

*Scales the gradient if its norm gets big*

if `gradient_norm > threshold`:

`gradient = (threshold / gradient_norm) gradient`



# Basic RNNs in TensorFlow

## Vanishing gradient problem in detail

- The same applies for backpropagating the error:

Case 1: **Exploding gradient** → **Gradient clipping**

*Scales the gradient if its norm gets big*  
if `gradient_norm > threshold`:  
$$\text{gradient} = (\text{threshold} / \text{gradient\_norm}) \text{ gradient}$$

Case 2: **Vanishing gradient**

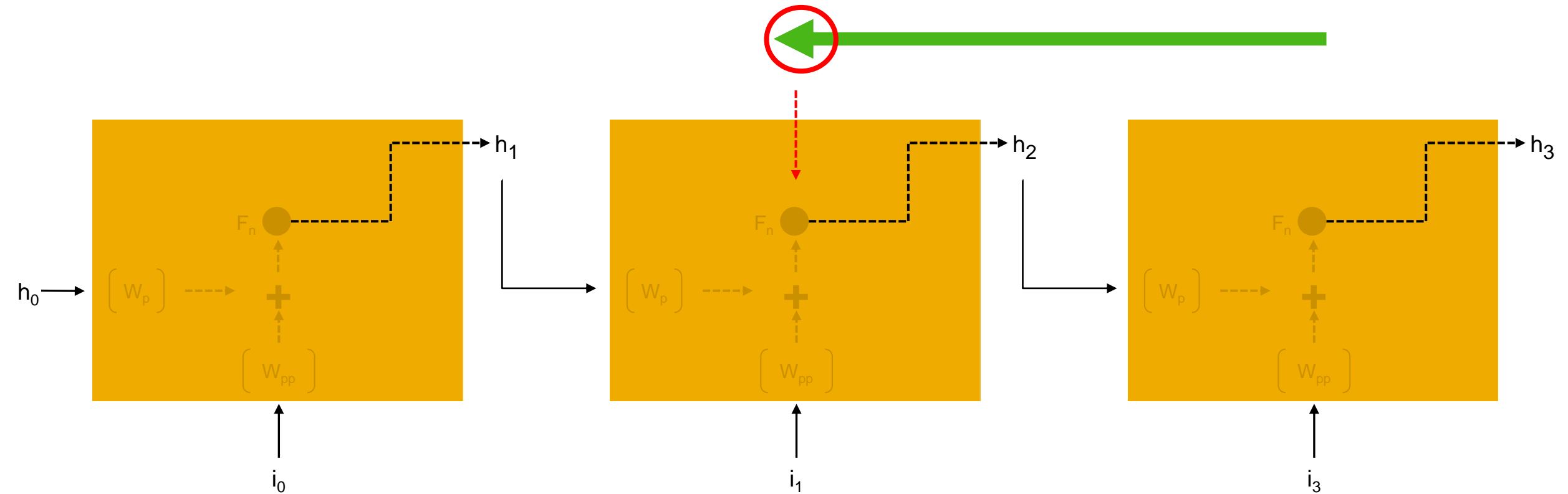




# Basic RNNs in TensorFlow

Vanishing gradient problem in detail

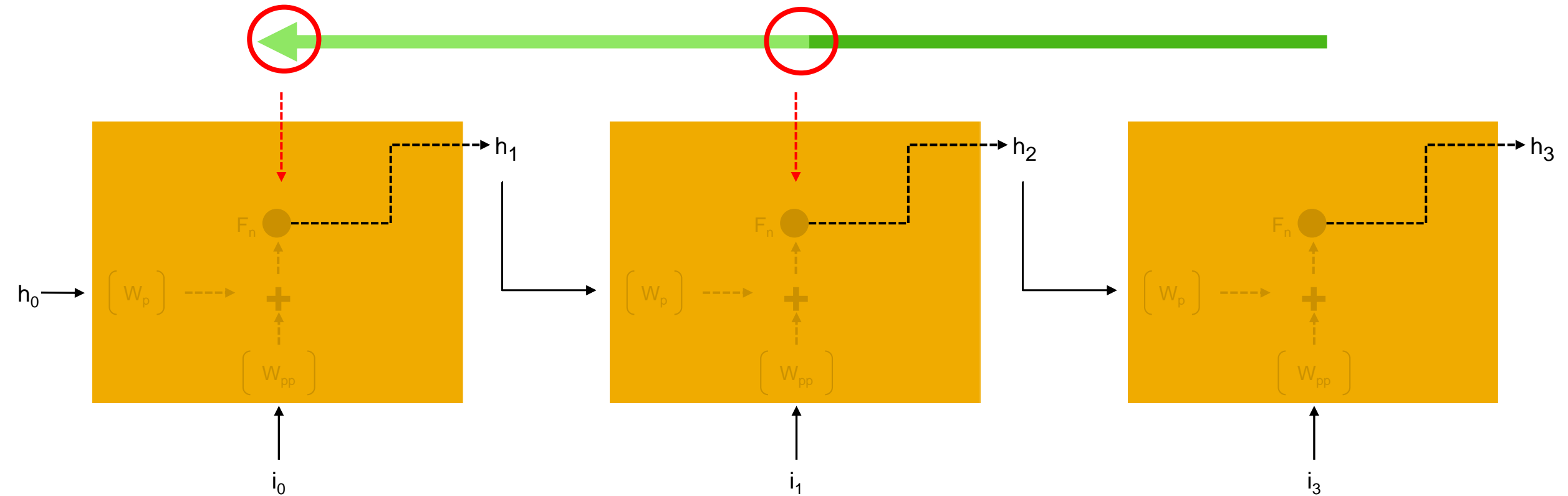
gradient flow is interrupted



# Basic RNNs in TensorFlow

Vanishing gradient problem in detail

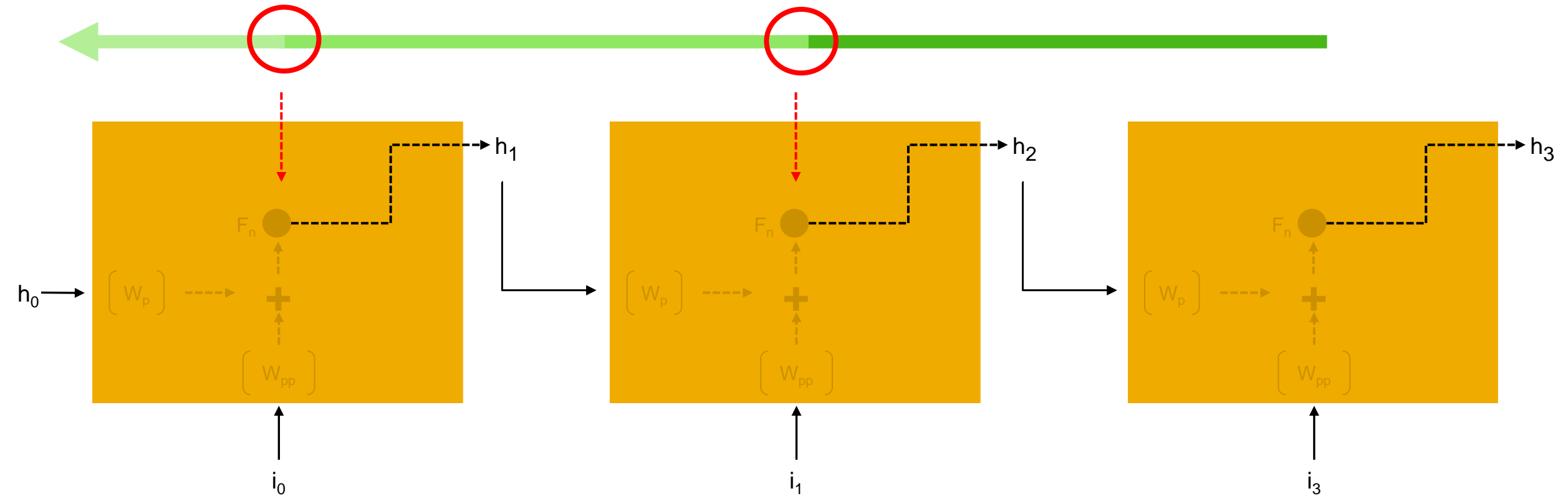
gradient flow is interrupted again



# Basic RNNs in TensorFlow

Vanishing gradient problem in detail

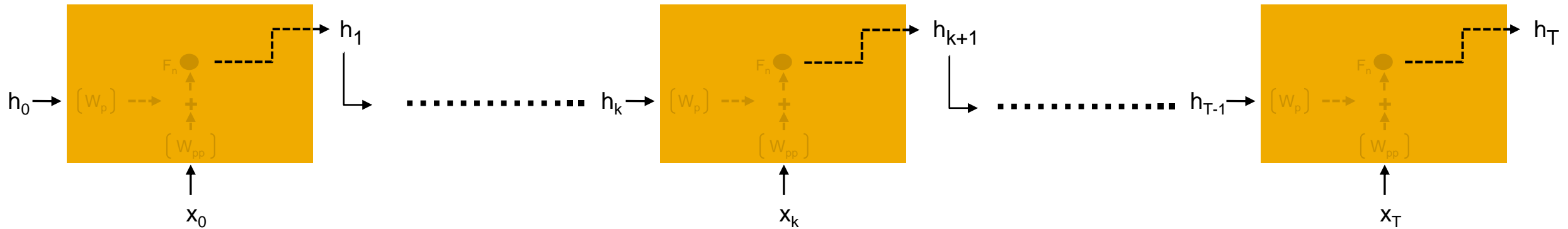

gradient flow is interrupted again



# Basic RNNs in TensorFlow

Vanishing gradient problem in deep network

small gradient flow is interrupted at each step, thus making learning difficult large



# Basic RNNs in TensorFlow

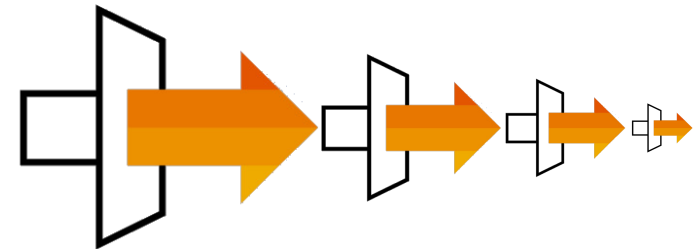
## Vanishing gradient problem in detail

- The same applies for backpropagating the error:

Case 1: **Exploding gradient** → **Gradient clipping**

*Scales the gradient if its norm gets big*  
if `gradient_norm > threshold`:  
$$\text{gradient} = (\text{threshold} / \text{gradient\_norm}) \text{gradient}$$

Case 2: **Vanishing gradient**



# Basic RNNs in TensorFlow

## Vanishing gradient problem in detail

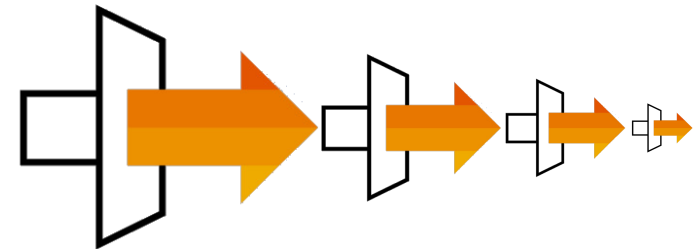
- The same applies for backpropagating the error:

Case 1: **Exploding gradient** → **Gradient clipping**

*Scales the gradient if its norm gets big*  
if `gradient_norm > threshold`:  
$$\text{gradient} = (\text{threshold} / \text{gradient\_norm}) \text{gradient}$$

Case 2: **Vanishing gradient** → **Change RNN architecture**

*This is the subject of our next unit!*

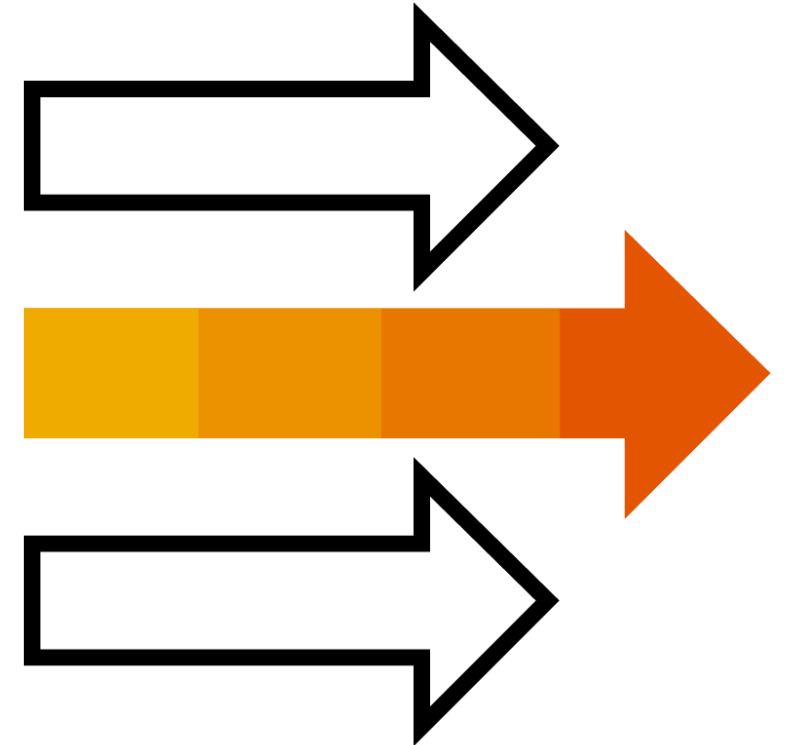


# Basic RNNs in TensorFlow

Coming up next

## Introduction to Long Short-Term Memory and Gated Recurrent Networks

- Vanilla LSTM cell structure
- Solution to vanishing gradient problem
- Further simplification in LSTM using GRU



# Thank you.

**Contact information:**

**open@sap.com**



© 2017 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company.

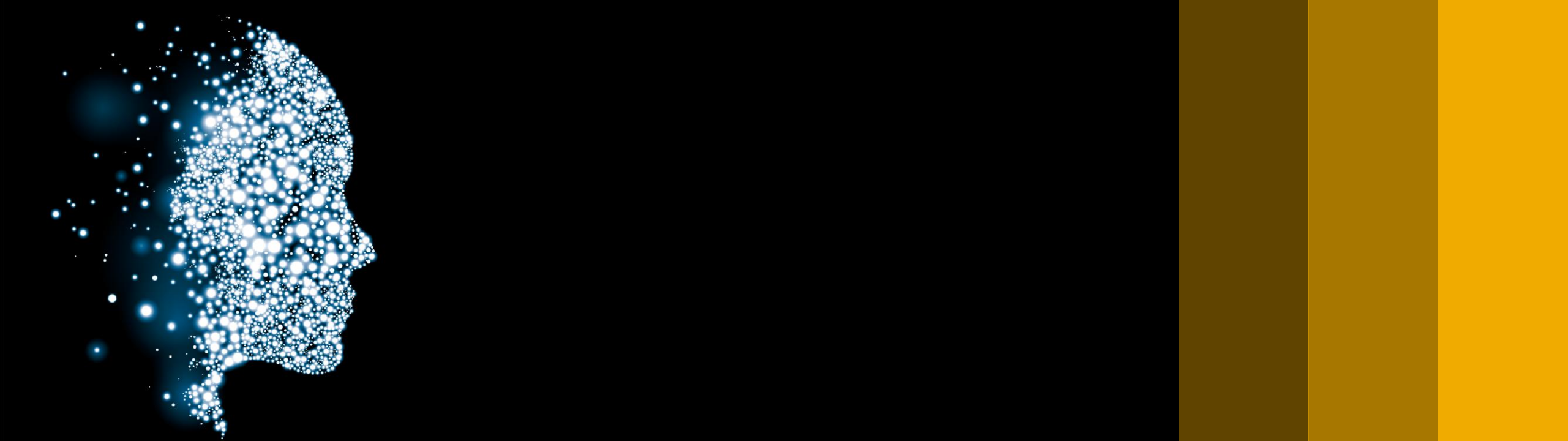
The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.

See <http://global.sap.com/corporate-en/legal/copyright/index.epx> for additional trademark information and notices.



## Week 3: Deep Networks and Sequence Models

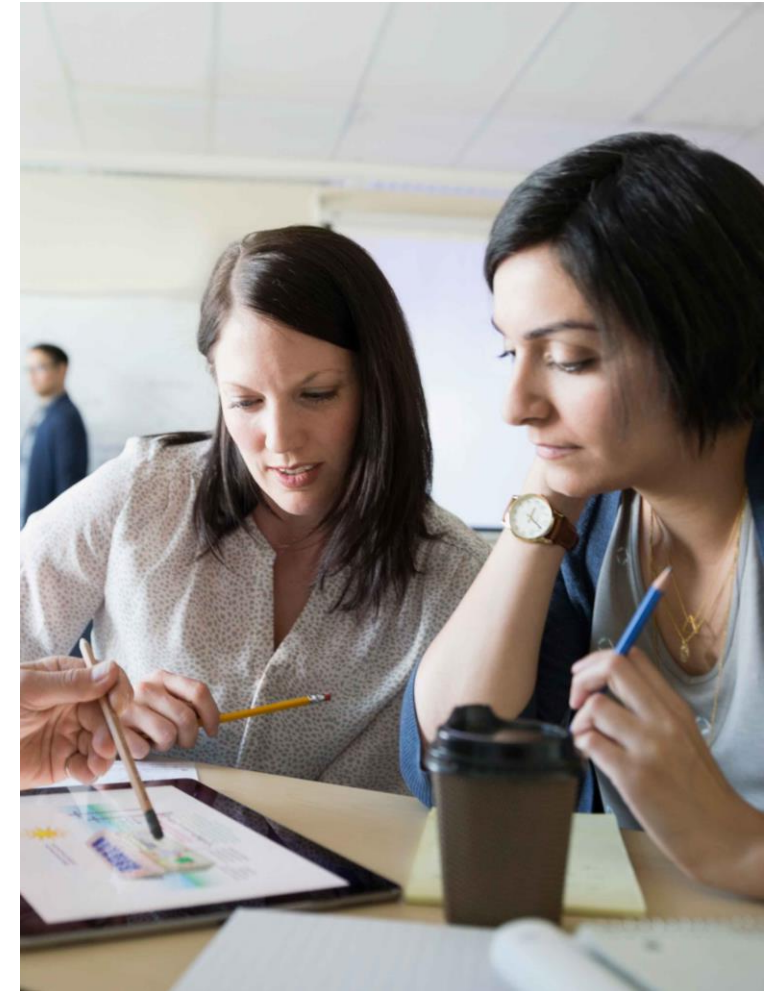
# Unit 5: Introduction to LSTM, GRU

# Introduction to LSTM, GRU

What we covered in the last unit

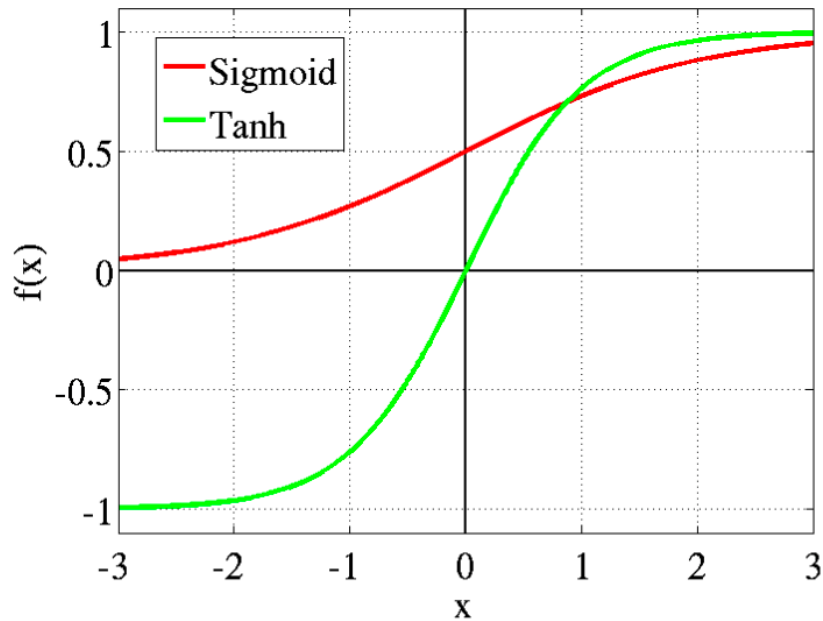
## Basic Recurrent Neural Network in TensorFlow

- Types of sequences
- Basic RNN cell structure
- Problems with RNN



# Introduction to LSTM, GRU

## Activation functions



- Sigmoid functions map values to the range (0,1)

$$\text{Sig}(x) = \frac{e^x}{e^x + 1}$$

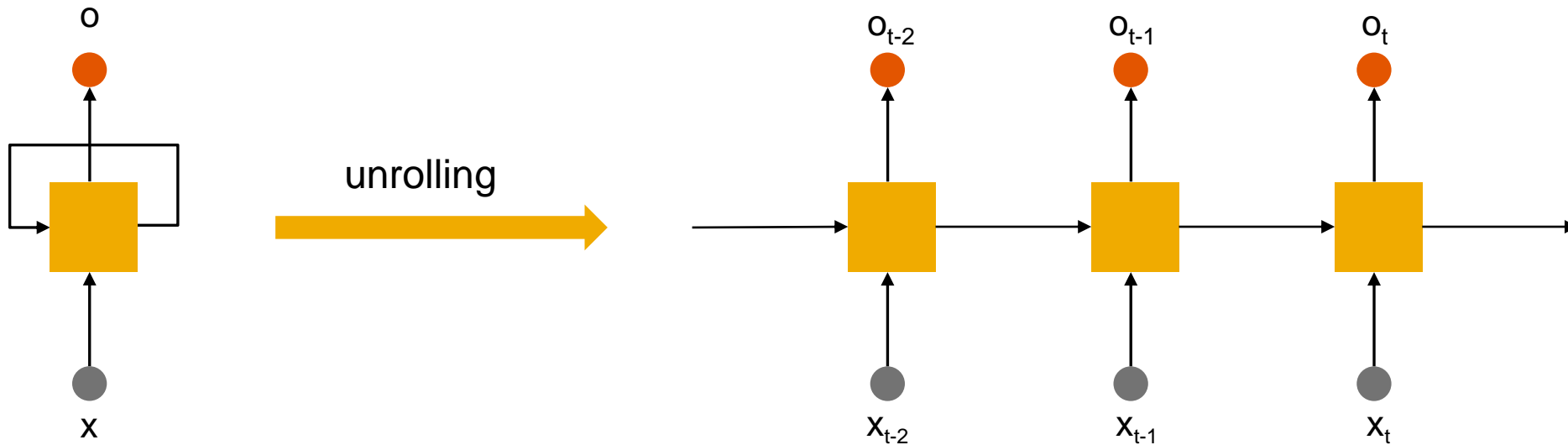
$$\forall x: 0 \leq \text{Sig}(x) \leq 1$$

- tanh is also bounded like a sigmoid, but is zero-centered (improves statistical properties of layer output)

$$-1 \leq \tanh(x) \leq 1$$

# Introduction to LSTM, GRU

Introduction to long short-term memory networks (LSTMs)



End-to-end architecture to model the sequence remains the same, but the cell structure changes

# Introduction to LSTM, GRU

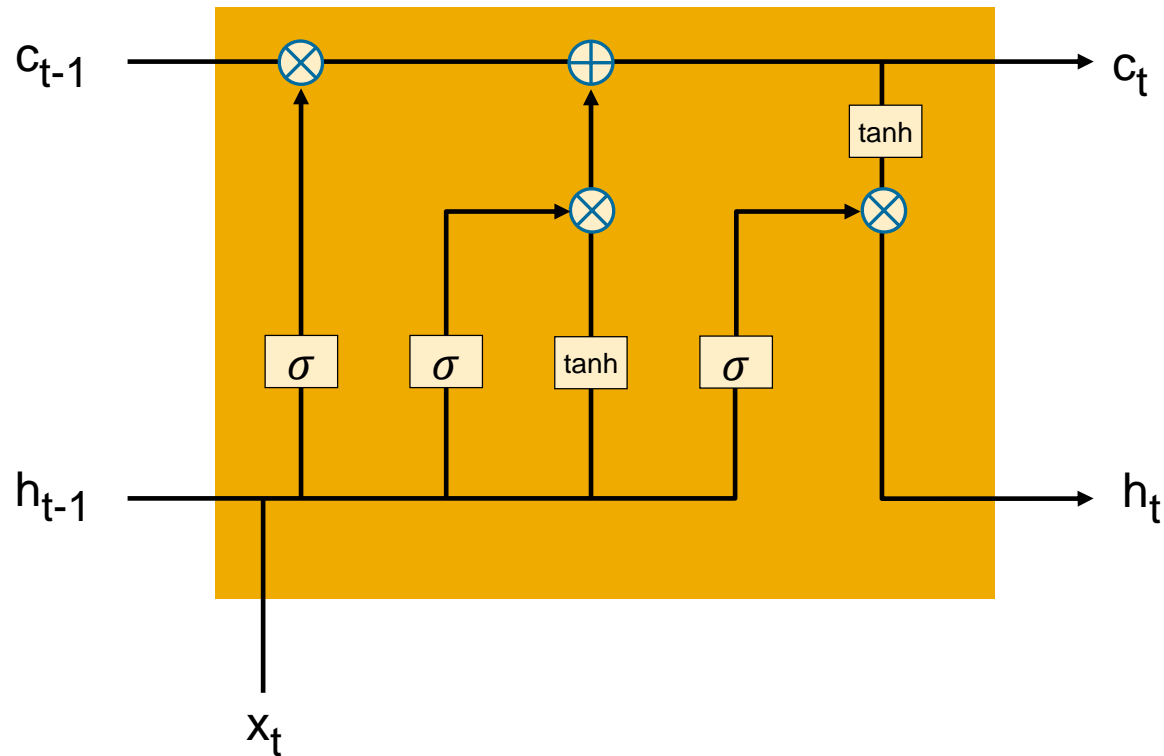
Introduction to long short-term memory networks (LSTMs)

## The three main components of an LSTM cell:

- The **memory cell** captures long-term information
- The **working memory** captures short-term information
- **Gates** regulate the information exchange between the memory cell and working memory

# Introduction to LSTM, GRU

## LSTM cell structure



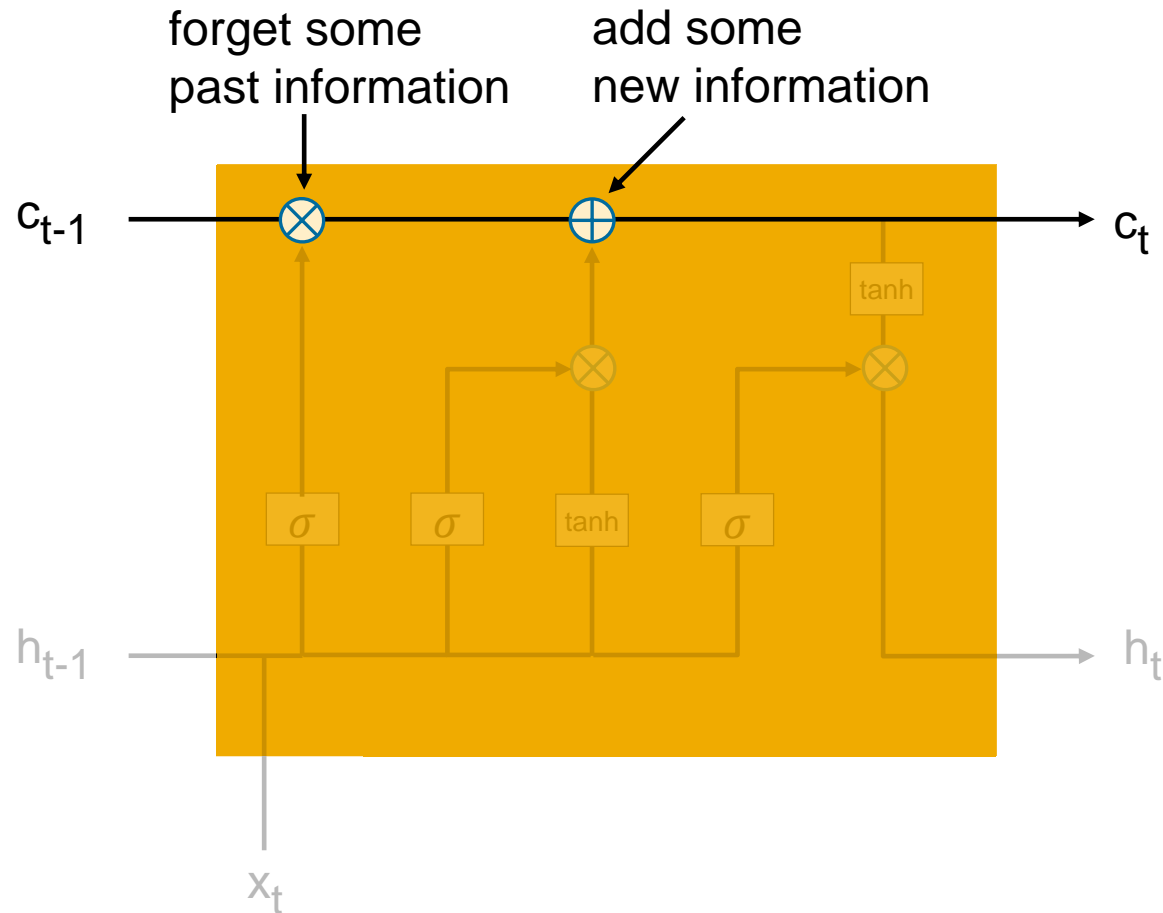
## LSTM:

- Decide whether current input matters
- Decide which part of the memory to forget
- Decide what to output at the current time

**LSTM:** Sepp Hochreiter, Jürgen Schmidhuber (1997). *Long Short-Term Memory*.

# Introduction to LSTM, GRU

## LSTM cell structure – Memory cell

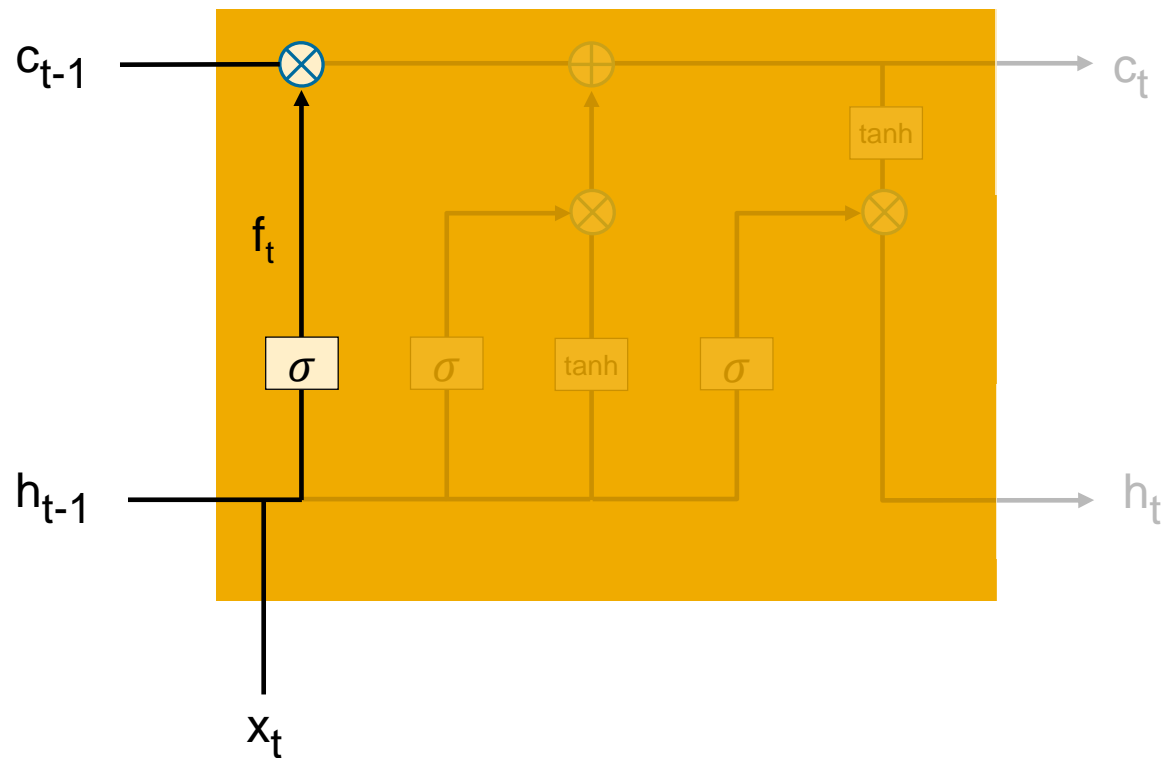


- Captures long-term dependency
- Element-wise operation
- Gradients can flow freely without suppression



# Introduction to LSTM, GRU

## LSTM cell structure – Forget gate

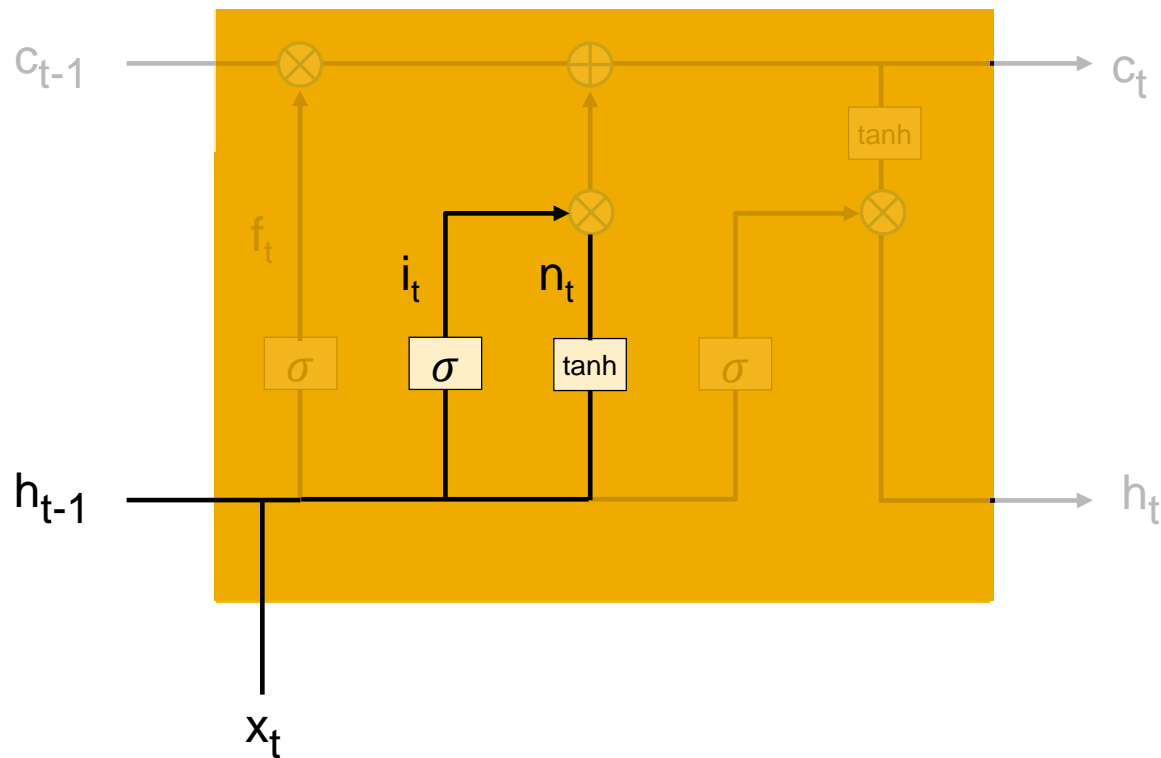


$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$

- Based on the past sequence and current input, a **forget regulator ( $f_t$ )** is created
- It dampens certain elements of  $c_{t-1}$
- As a result, some past information is forgotten
- This helps in remembering only the important part of a sequence

# Introduction to LSTM, GRU

## LSTM cell structure – Input gate



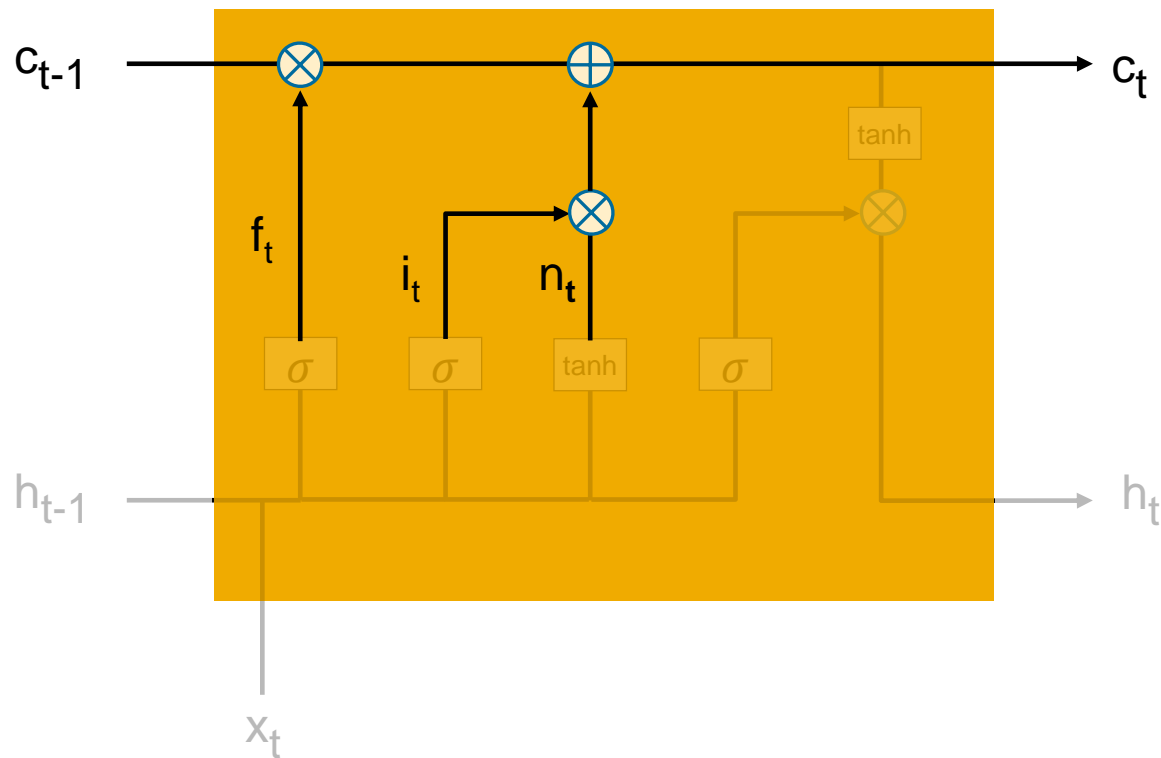
$$i_t = \sigma(W_i [h_{t-1}, x_t] + b_i)$$

$$n_t = \tanh(W_n [h_{t-1}, x_t] + b_n)$$

- $n_t$  contributes new information from the current input
- $i_t$  acts as an **input regulator**
- It suppresses some new information which is not relevant for the long-term update

# Introduction to LSTM, GRU

LSTM cell structure – Updating the memory cell

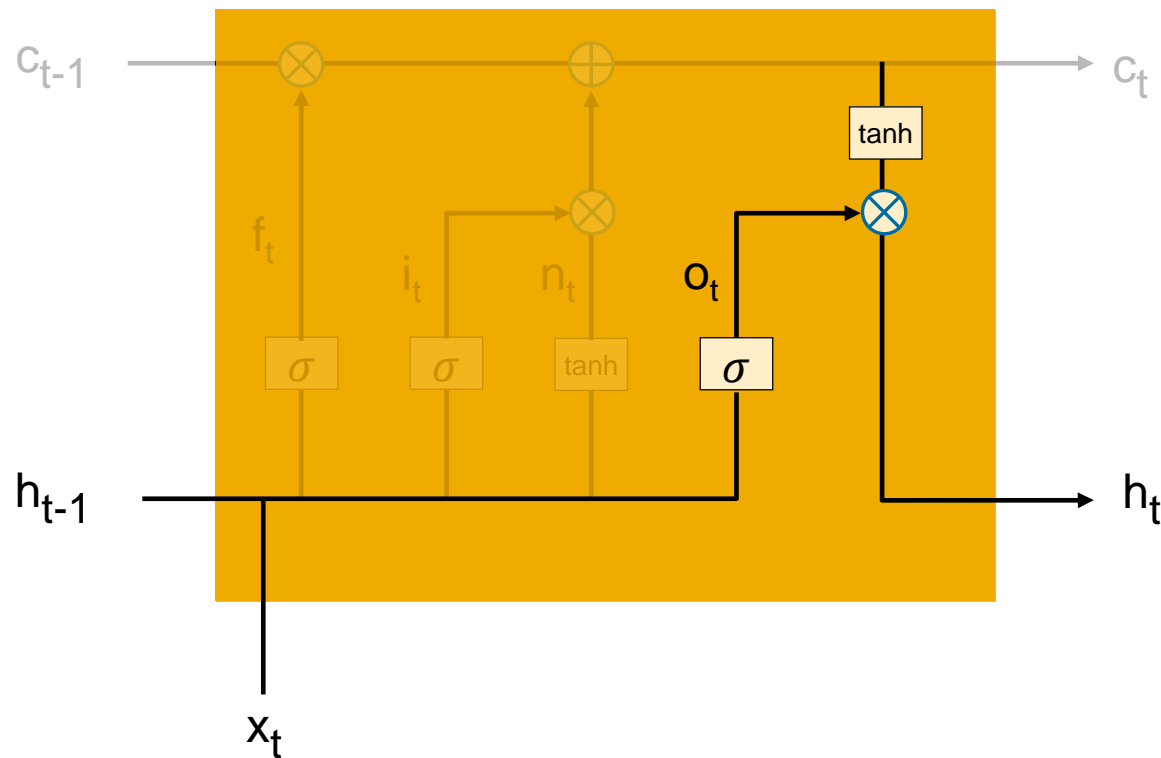


$$c_t = f_t * c_{t-1} + i_t * n_t$$

- After all updates in the memory cell, we get new long-term information in  $c_t$
- The inputs to  $c_t$  are between 0 and 1, but the elements can be unbounded because of addition to each element over multiple steps

# Introduction to LSTM, GRU

## LSTM cell structure – Output gate



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

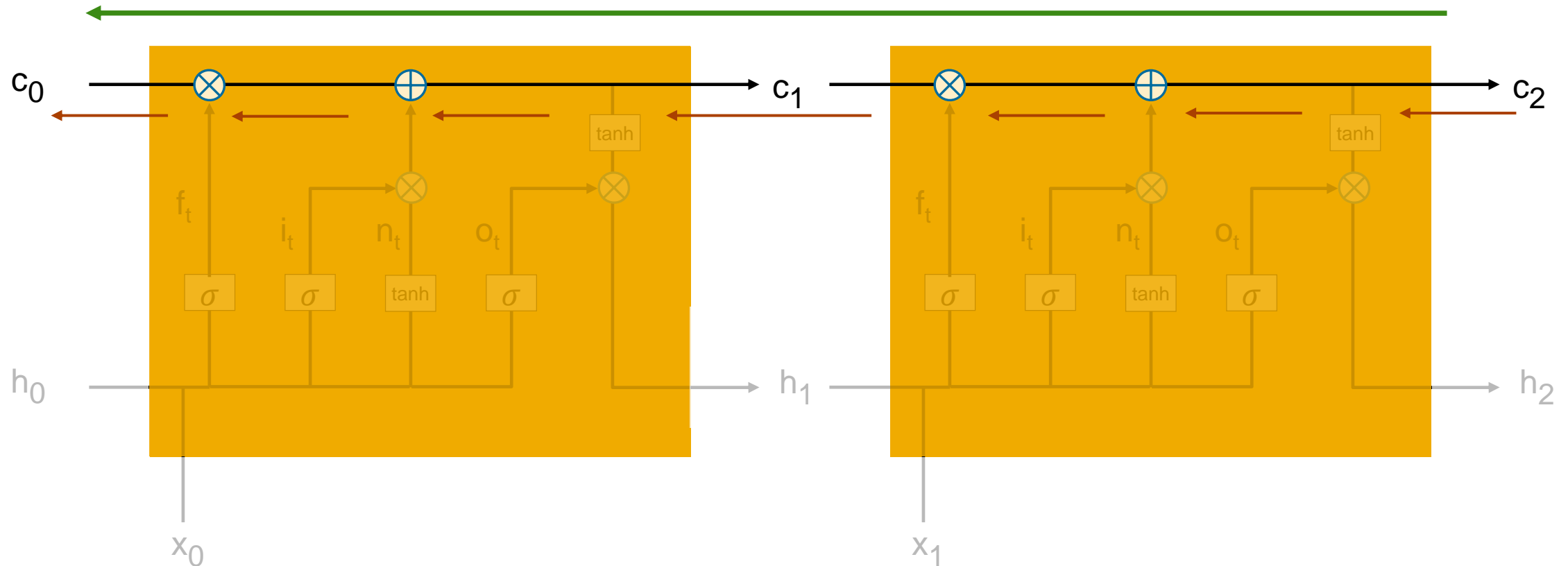
$$h_t = o_t * \tanh(c_t)$$

- Computation of new short-term information
- Since  $c_t$  is unbounded, we again bound it with an activation function, in this case  $\tanh$
- $o_t$  acts as an **output regulator**
- It decides what fraction of long-term information is considered for generating current working memory

# Introduction to LSTM, GRU

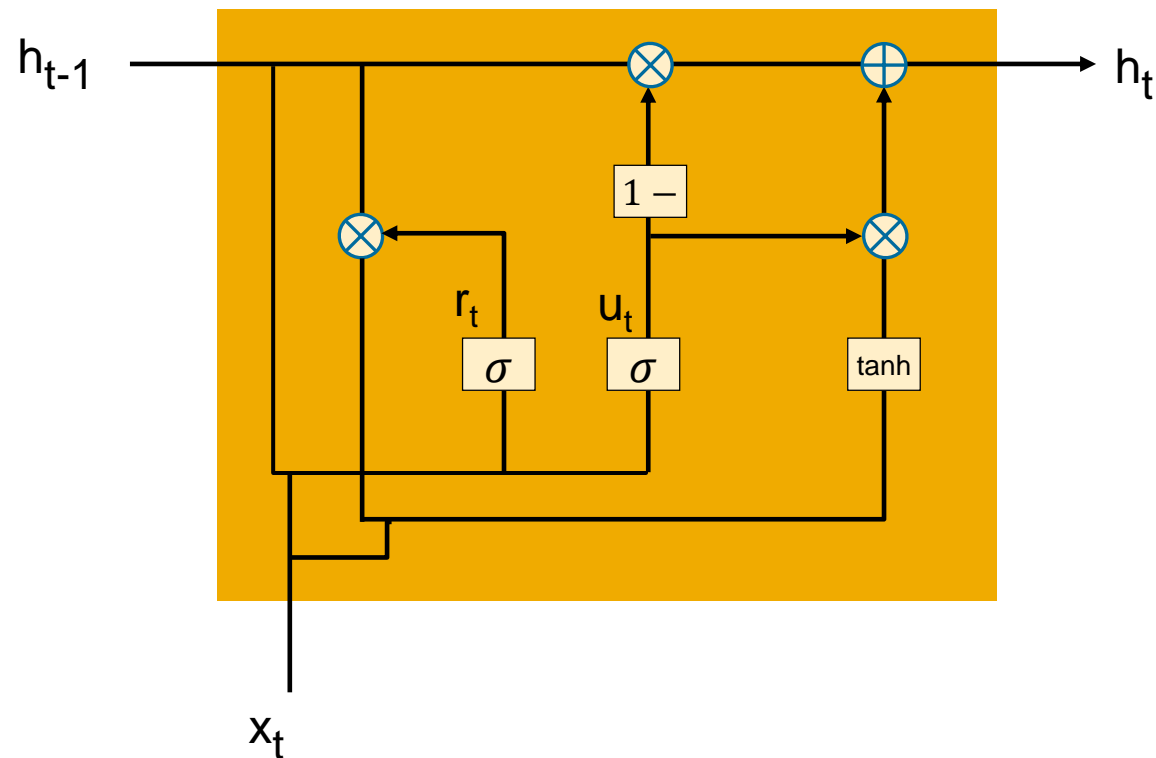
## LSTM cell structure and the vanishing gradient

Uninterrupted gradient flow by virtue of memory cell



# Introduction to LSTM, GRU

Gated recurrent unit (GRU) – Another variant of RNN



- Simplifies the standard LSTM cell
- Combines forget and input gate

$$u_t = \sigma(W_u[h_{t-1}, x_t])$$

$$r_t = \sigma(W_r[h_{t-1}, x_t])$$

$$n_t = \tanh(W_n[r_t * h_{t-1}, x_t])$$

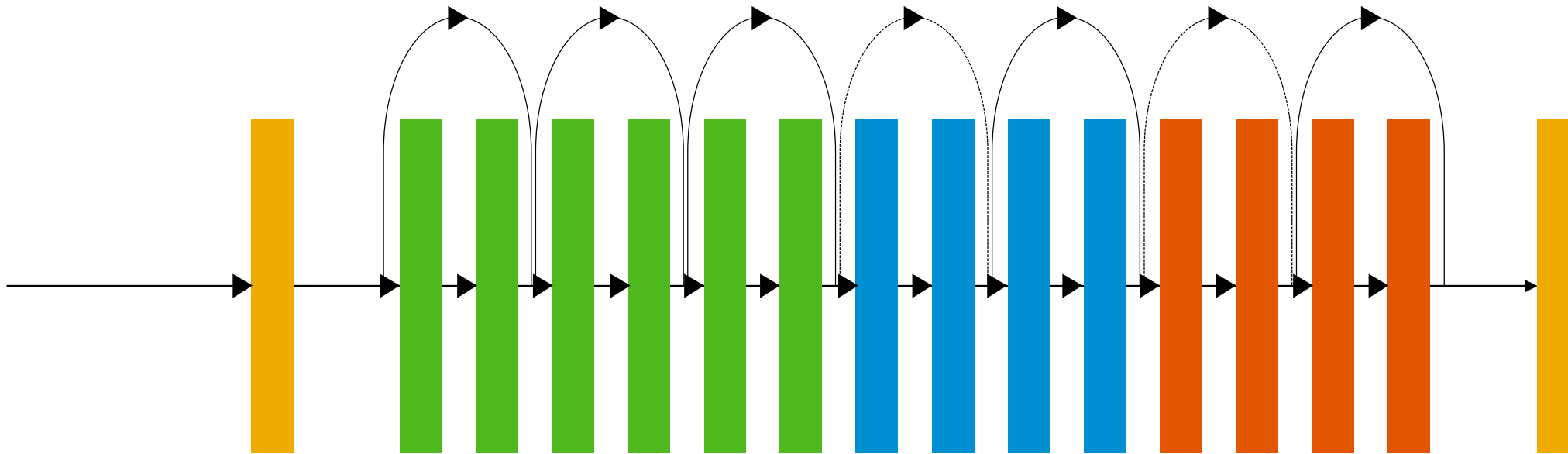
$$h_t = u_t * h_{t-1} + (1 - u_t) * n_t$$

- **$r_t$  regulates** the information in previous state
- **$u_t$  regulates** the new information and forgettable information simultaneously

**GRU:** Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, Yoshua Bengio (Dec, 2014). *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*.

# Introduction to LSTM, GRU

## Vanilla LSTM cell structure



A similar idea is used in other state-of-the-art networks like ResNet

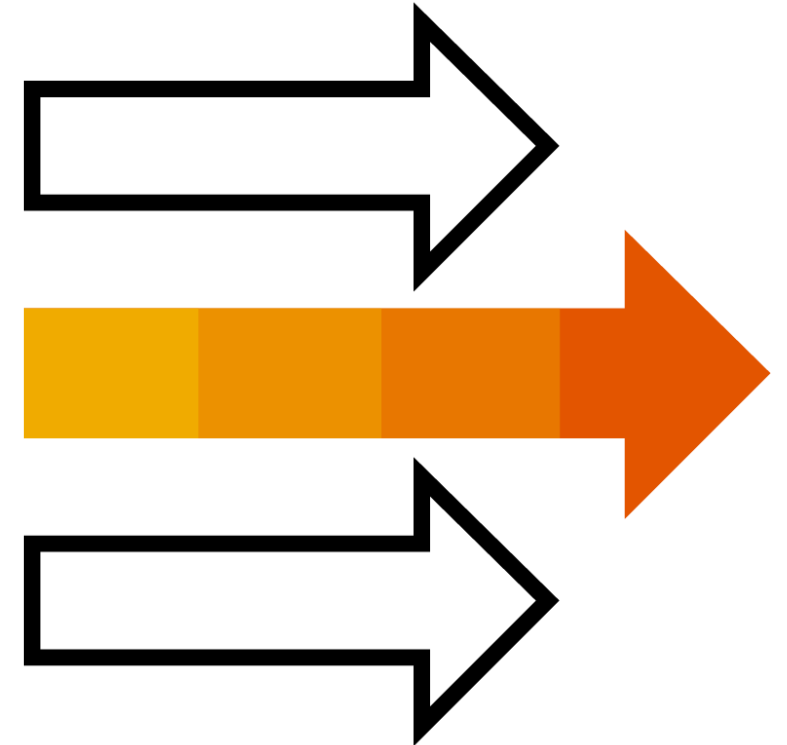
**ResNet:** Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun (Dec, 2015). *Deep Residual Learning for Image Recognition*.

# Introduction to LSTM, GRU

Coming up next

## Convolutional Networks

- Introduction to CNNs
- CNN architecture
- Accelerating deep CNN training
- Applications of CNNs





# Thank you.

**Contact information:**

**open@sap.com**

© 2017 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company.

The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.

See <http://global.sap.com/corporate-en/legal/copyright/index.epx> for additional trademark information and notices.