Deep learning

RNNs

Announcements

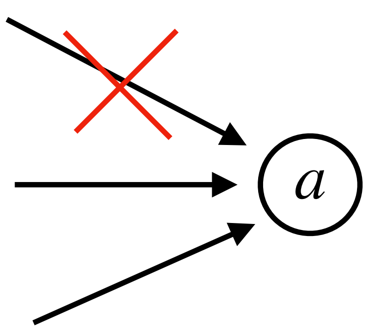
- Environment
  - It is a lot faster now, next week it will be even faster.
  - Docker image 1.2
- Assignment 2 will be set on 9th of April, due 23rd of April.
  - RNN based.

Recap / Questions?

$$J(\boldsymbol{w}) = \frac{1}{n}\sum_i^n (l(f(x_i, \boldsymbol{w}), y_i) + \boxed{\frac{\lambda}{2n}\sum_j w_j^2}$$

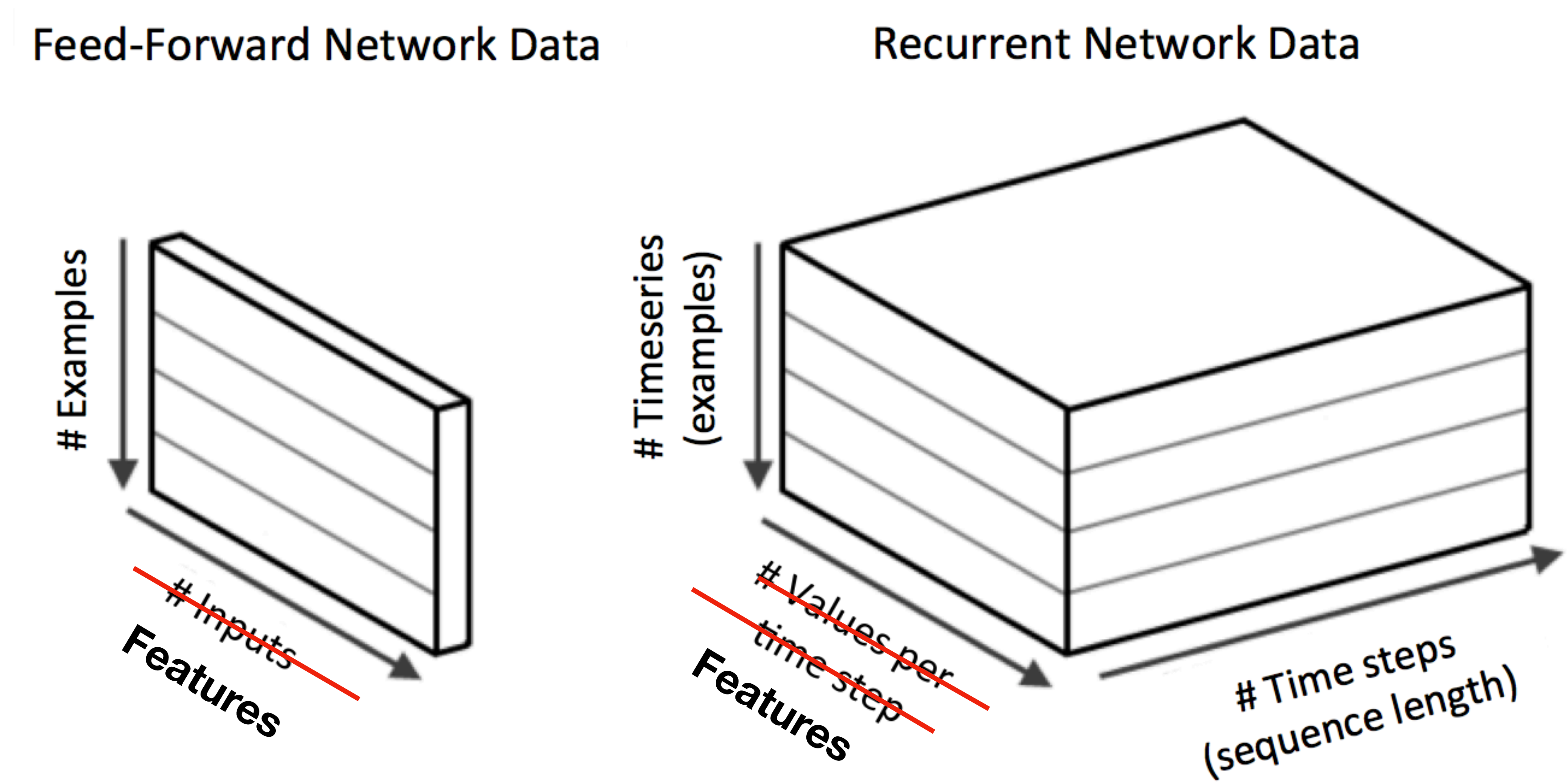
- Regularisation
  - L2 regularisation
- Practiced BN, Dropout and L2.

Set weight to 0



Recap / Questions?

- Sequential data



Data = (examples, features)

Data = (examples, sequence_length, features)

Source: https://www.oreilly.com/library/view/deep-learning/9781491924570/ch04.html

Recap / Questions?

Data = (examples, sequence_length, features)

Dataset of sentences

"hi"	"hoe"	"gaat"	"het"	"<EOS>"
"goed"	"<EOS>"	0	0	0
"leuk"	"<EOS>"	0	0	0
"mag"	"je"	"iets"	"<EOS>".	0

Dataset's dimensions = (4, 5, ?)

Dataset of measurements

(21, 998)	(20, 998)	(19, 980)
(10, 1040)	(13, 1000)	(11, 981)
(40, 970)	(40, 970)	(41, 978)

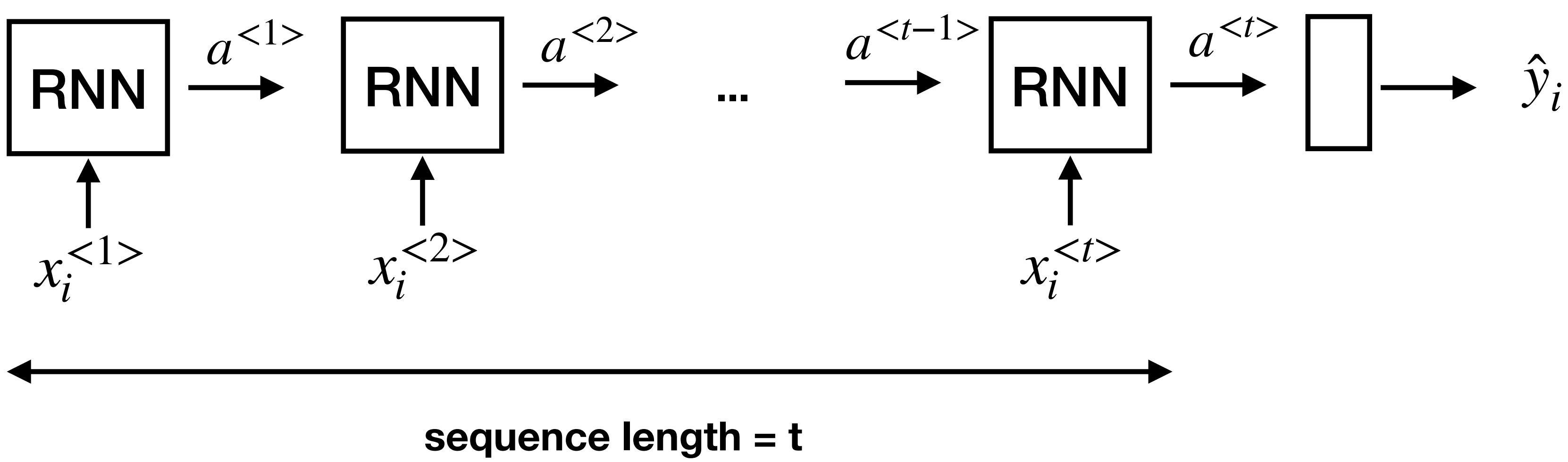
Dataset's dimensions = (3, 3, 2)

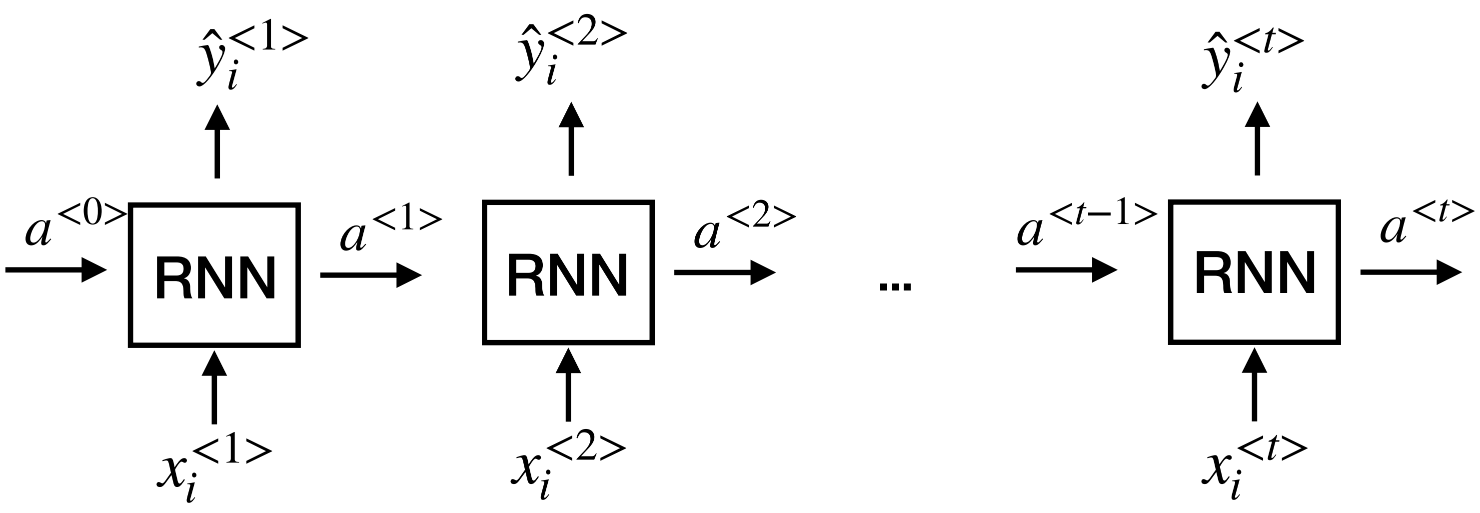
(21, 998, -4)	(20, 998, -4)	(19, 980, -6)	(19, 980, -6)
(21, 998, -4)	(20, 998, -4)	(19, 980, -6)	(19, 980, -6)

Dataset's dimensions = ?

Recap / Questions?

- Basic RNN cell





Overview

Today we will cover

Topic: RNNs.

- Training RNNs.
- Long term dependencies, LSTM & GRU.
- Residual connections.

Notebook: RNN using GRU.

Topic: Improving RNNS.

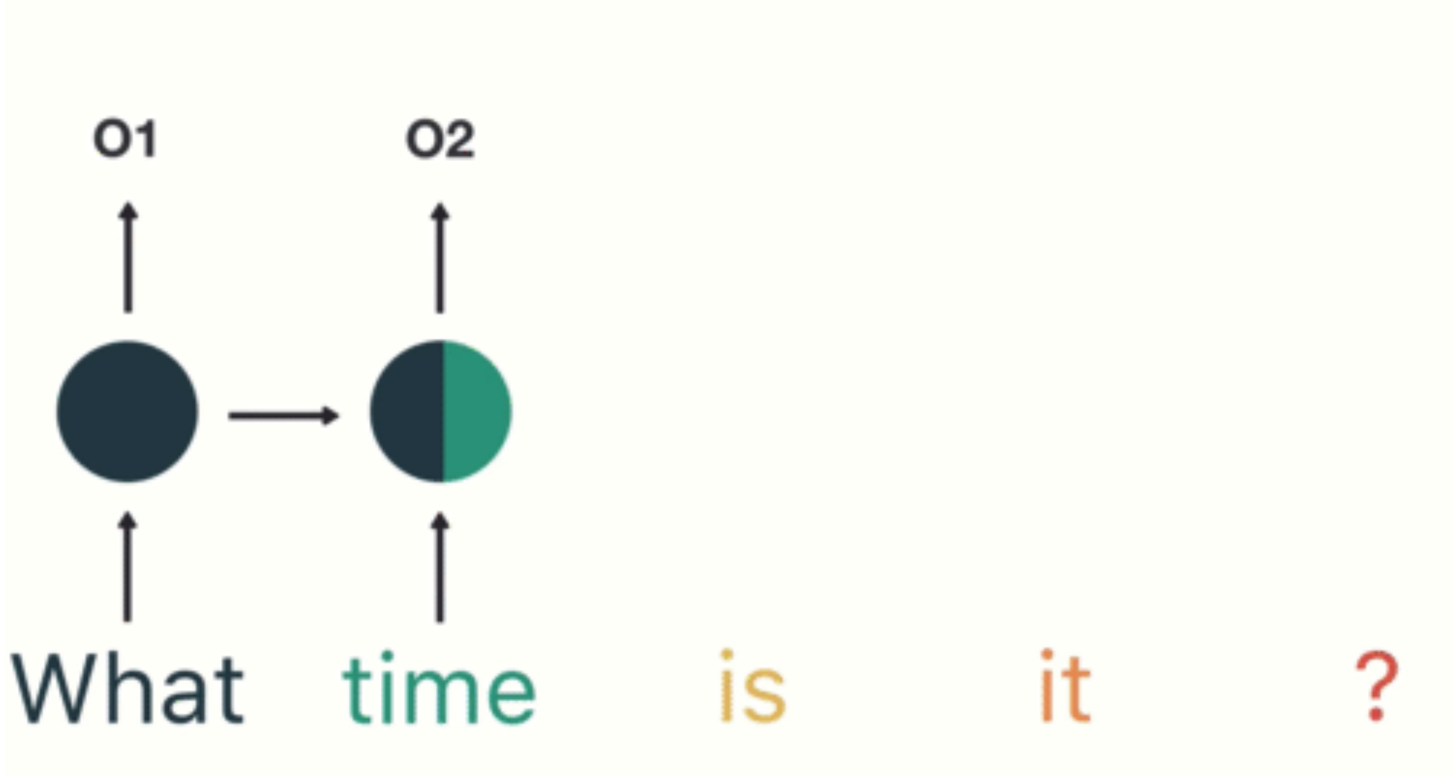
- Regularisation in RNNs.
- Going deep, stacking layers.

Notebook: Improving RNNs.

Sequential data

Long term dependencies

- After each time-step we store some information.
- When we train RNNs we are training them to do two things.
  - Store the correct information between time-steps. This is **hard**.
  - Map the stored information to solve the task. This is **easy**.



Source: https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9

Sequential data

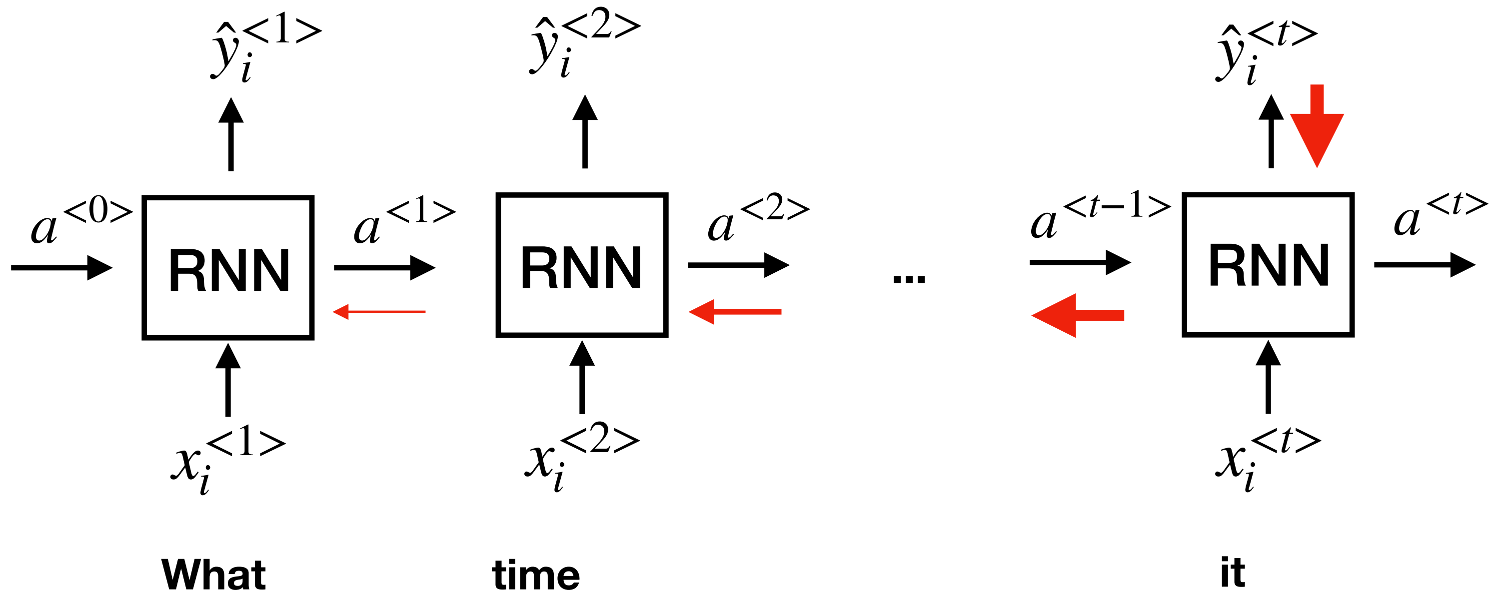
Long term dependencies

- Why is it hard?
- RNNs are trying to learn to **represent sequences by remembering what they contain**.
  - In question detection we want to remember if we saw "what", ...
- We learn to represent the sequence in order to solve the task at hand.
  - At **start** we are doing **poorly** (random weights) and we see almost no indication that "what" was used previously.
  - We want to update our RNN cell so that next time we remember when we see "what".

Sequential data

Long term dependencies

- If the sequence is long, little information is passed all the way to the end so a small error signal is sent back for the word "what".
- Small error signal = small updates

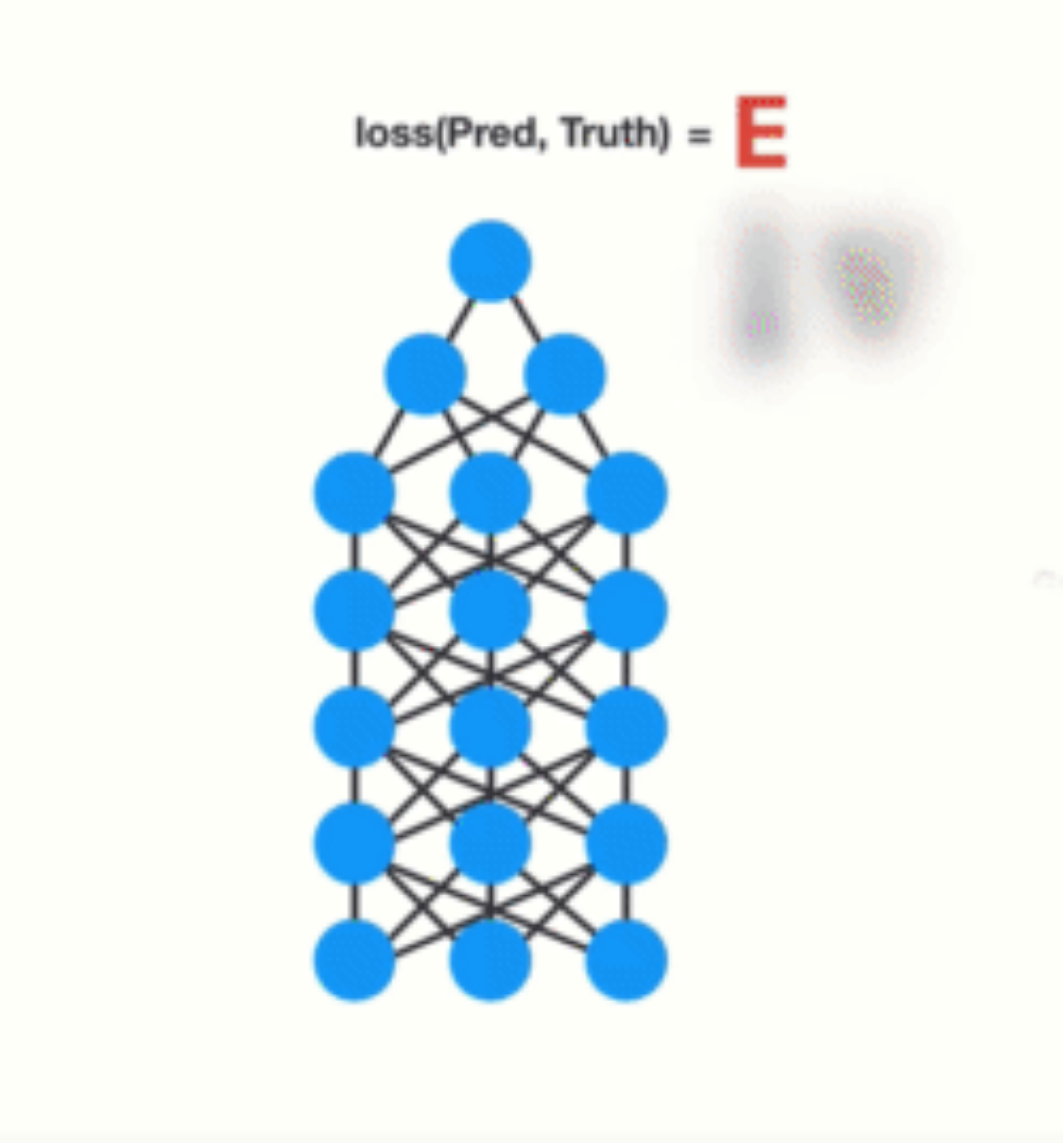


Sequential data

Long term dependencies

- The is the problem of **long term dependencies**, which is because the **gradient vanishes** (the error signal).

- This is **a general problem** in neural networks trained with gradient descent, but very tangible in RNNs due to their depth.



Source: https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9

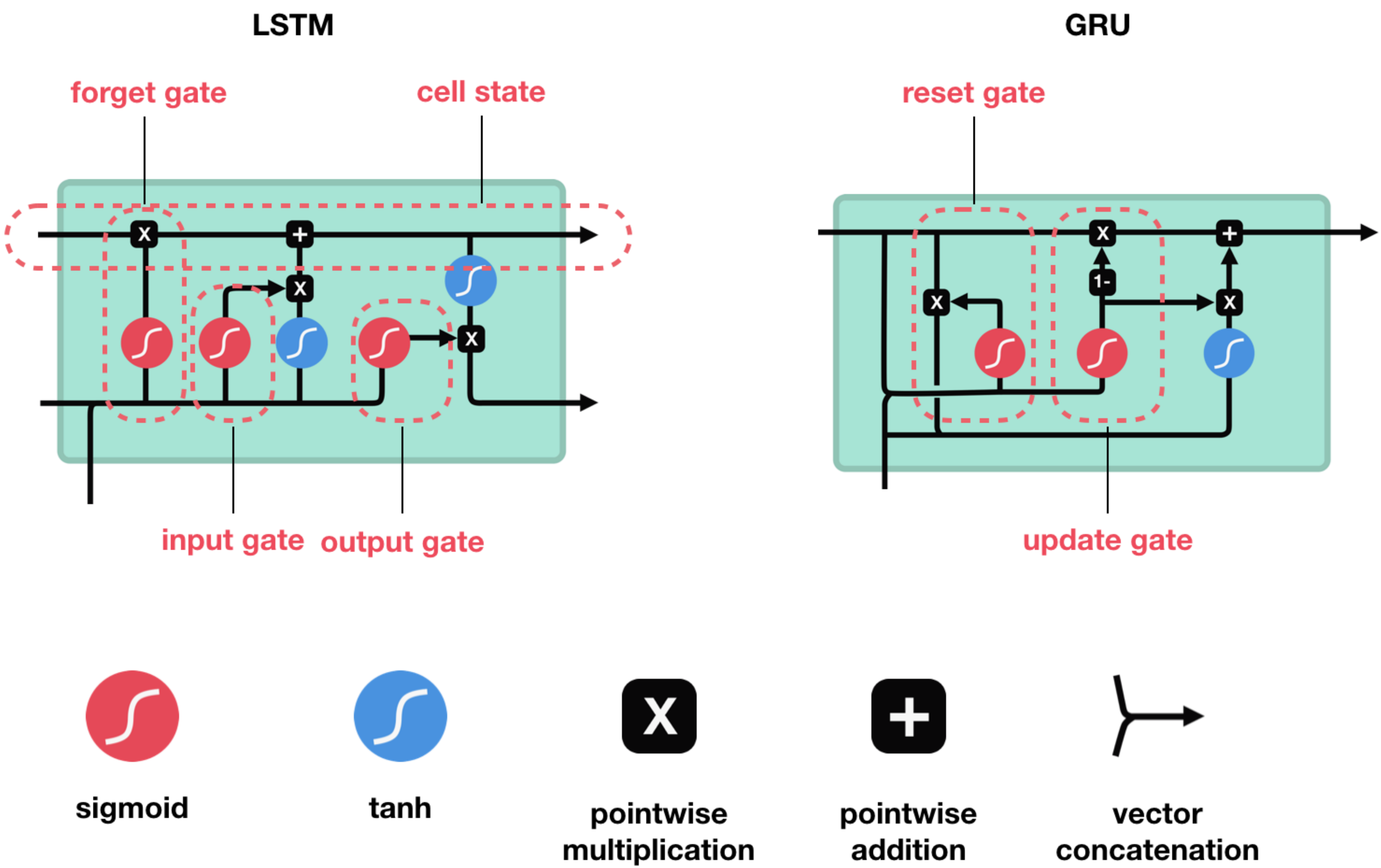
LSTMs and GRUs

Long term dependencies

- The solution to the vanishing gradient problem in RNNs was a different implementation of the RNN cell.
  - LSTM (1997)
  - GRU (2014)
- They are more complex and expensive but are able to deal better with long term dependencies.
  - LSTM is heavier than GRU.

```
layer_gru(units = 10)          layer_lstm(units = 10)
```

SURF

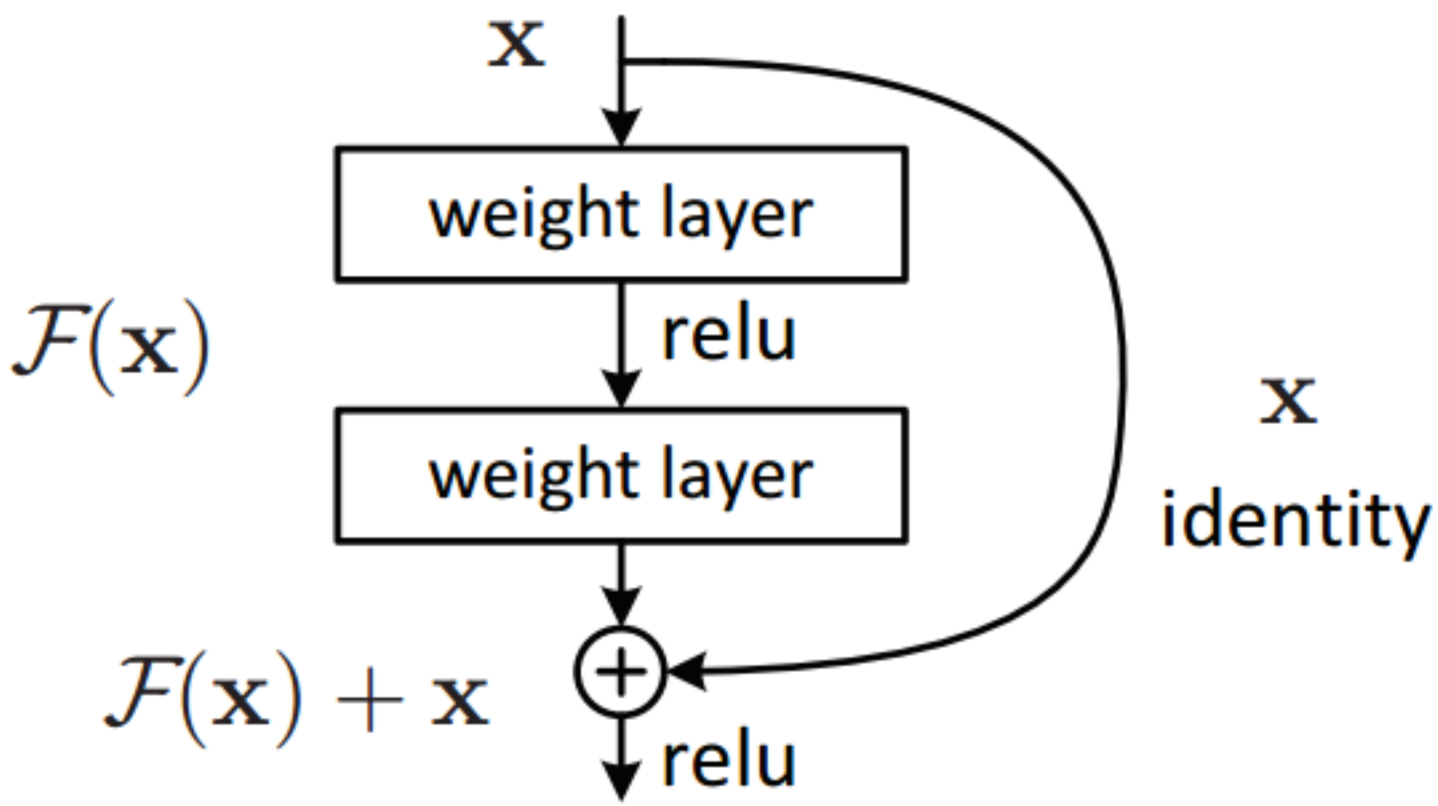


Source: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb8

Residual connections

Long term dependencies

- The solution to this problem in general are **residual connections**, (ResNet, 2015).
- We add connections which **bypass non-linear activations** (or go through fewer).
- This allows the error signal to flow directly to earlier layers.



Source: https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9

Generating sequences

We want to break this long sequence into many sequences

(1, ...) (2, ...) (3, ...) (4, ...) (5, ...) (6, ...) (7, ...) (8, ...) (9, ...) (10, ...) (11, ...) (12, ...) (13, ...) (14, ...)

Reshape approach - sequence length = 7

(1, ...) (2, ...) (3, ...) (4, ...) (5, ...) (6, ...) (7, ...) (8, ...) (9, ...) (10, ...) (11, ...) (12, ...) (13, ...) (14, ...)

↓

1st example	(1, ...) (2, ...) (3, ...) (4, ...) (5, ...) (6, ...) (7, ...)
2nd example	(8, ...) (9, ...) (10, ...) (11, ...) (12, ...) (13, ...) (14, ...)

Generating sequences

We want to break this long sequence into many sequences

(1, ...) (2, ...) (3, ...) (4, ...) (5, ...) (6, ...) (7, ...) (8, ...) (9, ...) (10, ...) (11, ...) (12, ...) (13, ...) (14, ...)

Shift approach, using shift = 2, sequence length = 7

(1, ...) (2, ...) (3, ...) (4, ...) (5, ...) (6, ...) (7, ...) (8, ...) (9, ...) (10, ...) (11, ...) (12, ...) (13, ...) (14, ...)

↓

1st example (1, ...) (2, ...) (3, ...) (4, ...) (5, ...) (6, ...) (7, ...)

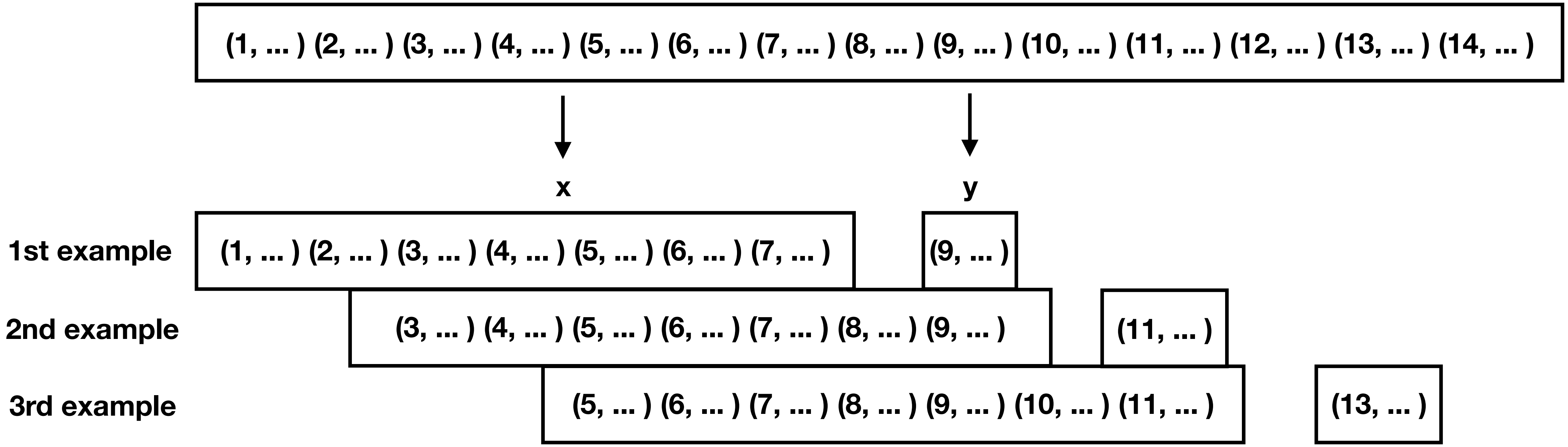
2nd example (3, ...) (4, ...) (5, ...) (6, ...) (7, ...) (8, ...) (9, ...)

3rd example (5, ...) (6, ...) (7, ...) (8, ...) (9, ...) (10, ...) (11, ...)

4th example (7, ...) (8, ...) (9, ...) (10, ...) (11, ...) (12, ...) (13, ...)

Generating sequences

Shift approach, using shift = 2, sequence length = 7, target shift = 1



Hands-on

Go to https://dba.projects.sda.surfsara.nl/

Notebook: `05a-rnns.ipynb`

Break at 11:00 / 15:00

Second part at 11:10 / 15:10

Improving RNNs

- Regularisation
  - L1/L2
  - Dropout, recurrent dropout
- Improving RNNs
  - Stacking
  - Stateful
  - Bi-directional

Improving RNNs

L2/L1 regularisation

- Just like with normal dense layers.
- we add L2/L1 regularisation to the weights learnt in the RNN cell.

```
layer_gru(units = 10, kernel_regularizer = regularizer_l2(l = 0.001))

layer_gru(units = 10, kernel_regularizer = regularizer_l1(l = 0.001))
```

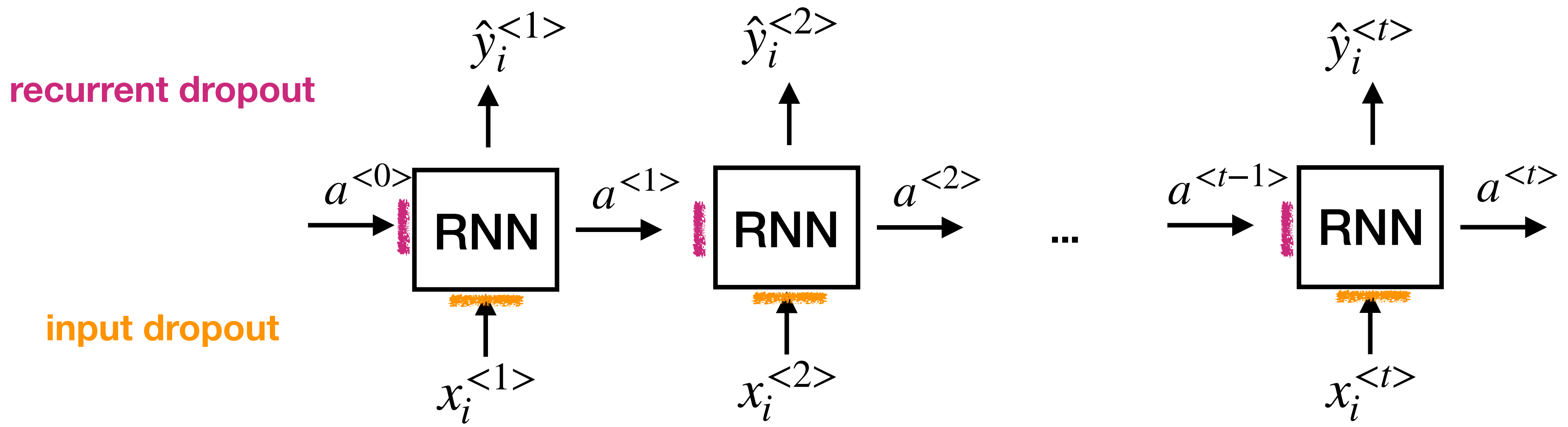
SURF

Improving RNNs

Dropout

- In RNNs we consider dropouts in two locations.
  - **Input**
  - **Recurrent dropout**

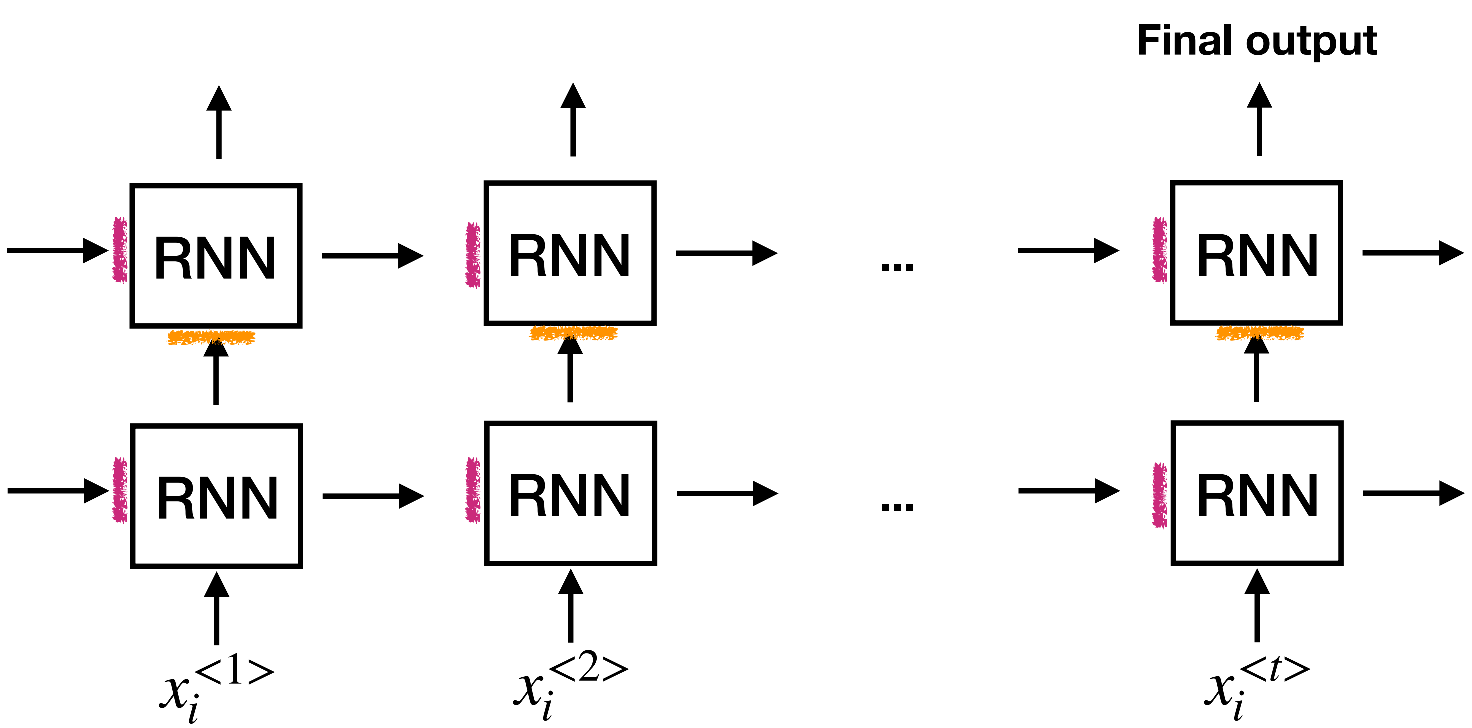
layer_gru(units = 10, dropout = 0.2, recurrent_dropout = 0.3)



Improving RNNs

Stacking RNNs

- Why would we consider input dropout?
  - Maybe in production we might not always get all inputs.
- More likely, we are stacking RNNs.
- Stacking RNNs is like adding additional layers in a dense network.
  - We never go that deep, 1-6 layers. Long training time.

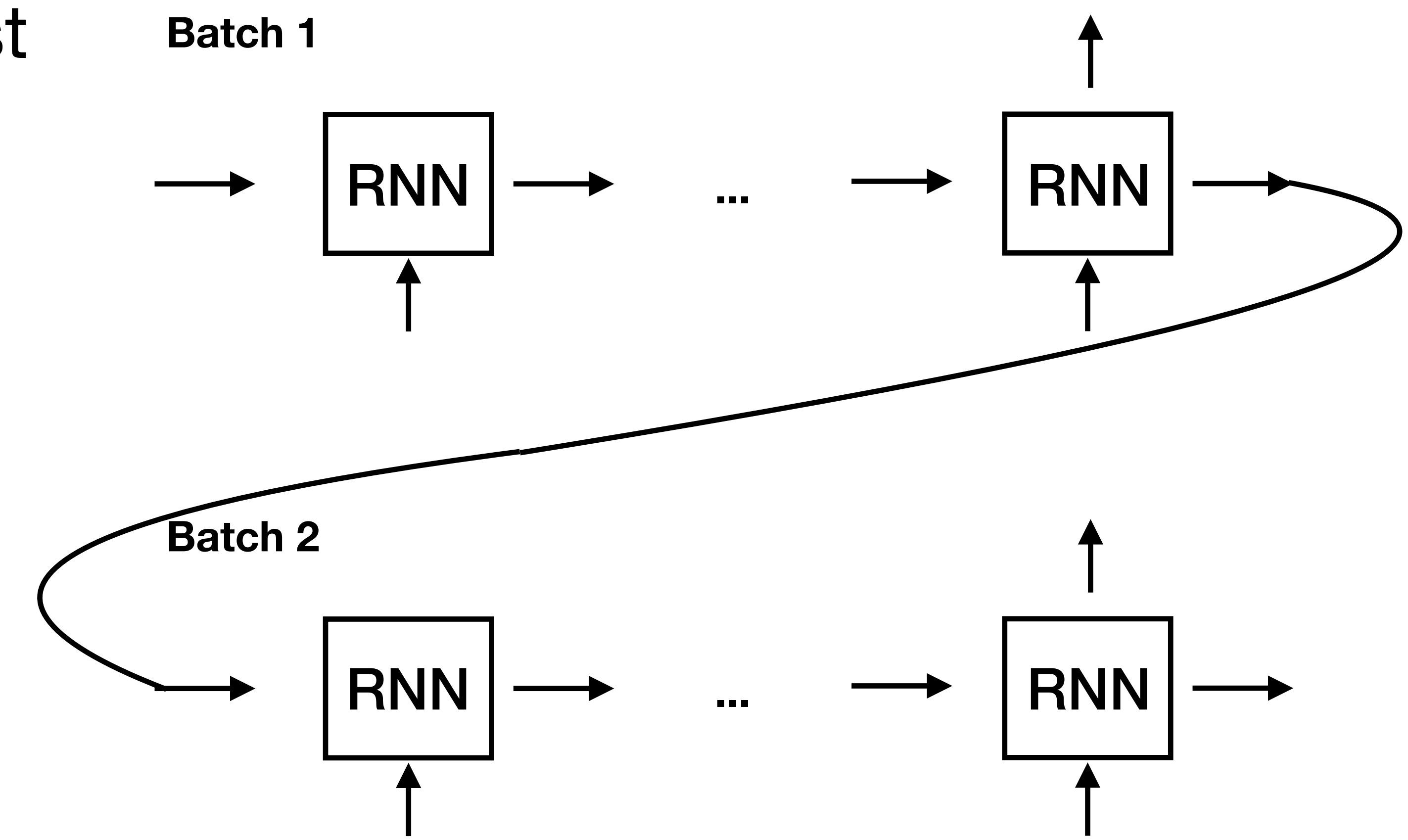



```
layer_gru(units = 10, return_sequences = TRUE) %>%
layers_gru(units = 10)
```

Improving RNNs

Stateful

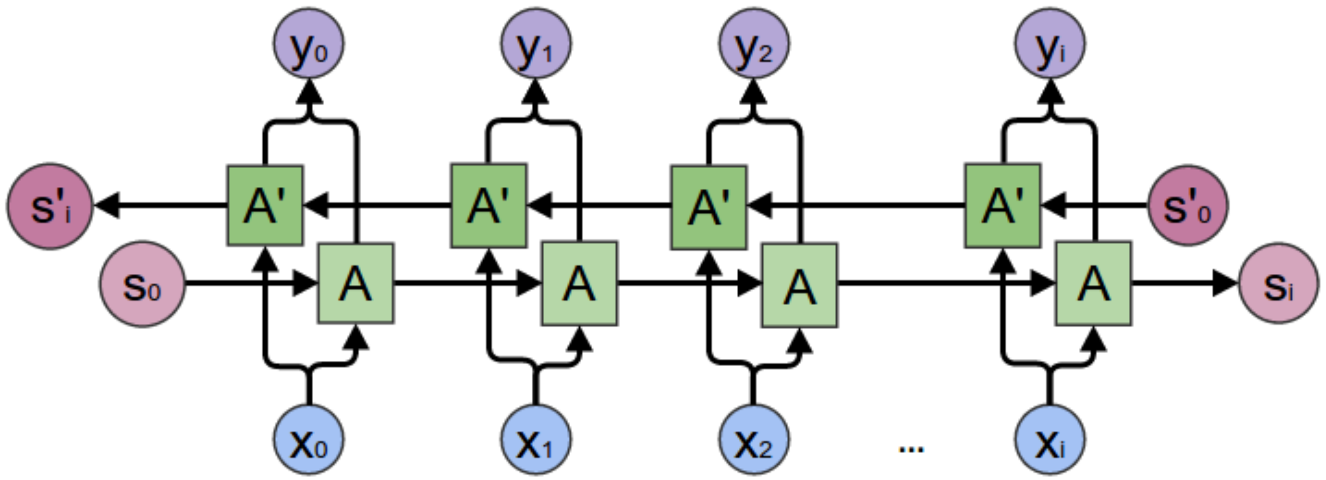
- A stateful RNN passes the last state of the previous batch to as an initial state to the next batch.
    - Otherwise the initial state is "all zeroes".
- This is useful if there is some connection between batches.
    - For example, the batches are in sequence.



Improving RNNs

Bi-directional

- We process the sequence in both directions.
- Very helpful for example in named entity recognition, in which we classify every word as a "person", "place", ...
  - He said, "Teddy Roosevelt ...
  - He said, "Teddy bears ...



Summary

- Regularisation
  - L1/L2
  - Dropout, recurrent dropout
- Improving RNNs
  - Stacking
  - Stateful
  - Bi-directional

Hands-on

Go to https://dba.projects.sda.surfsara.nl/

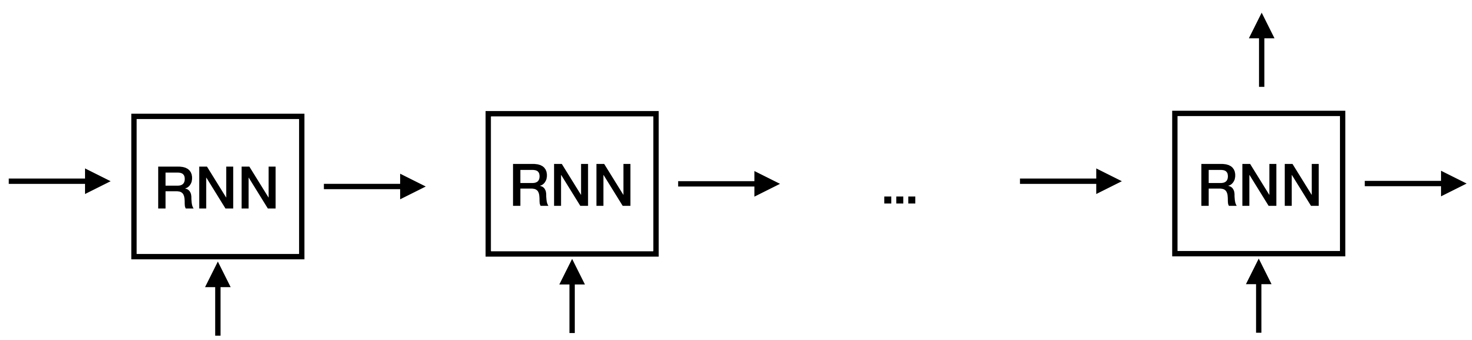
Notebook: `05b-.ipynb`

Wrap-up at 12:20 / 16:20

RNN architectures

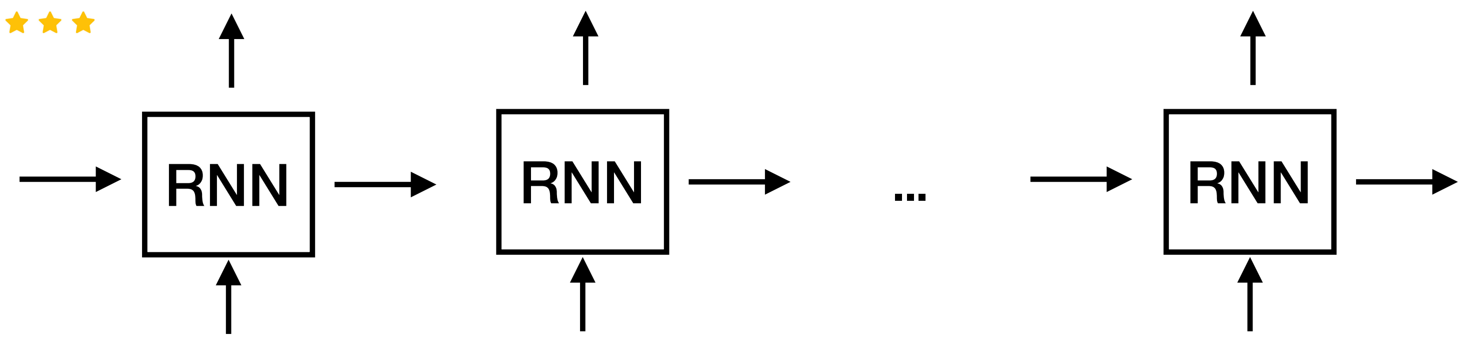
Intput / Output

Many-to-One



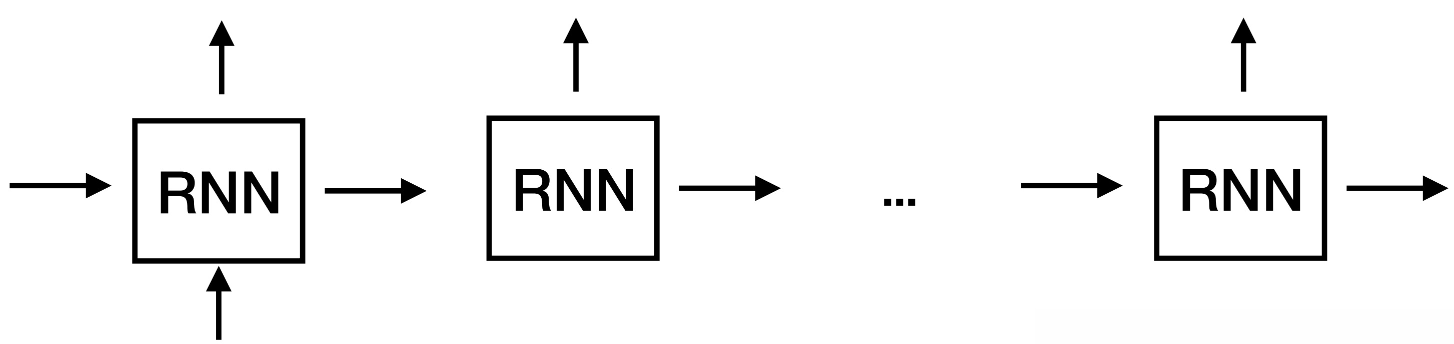
Sentiment analysis "Great food!" → 

Many-to-Many*



Machine translation "Hoi, hoe gaat het?" → "Hi, how are you?"

One-to-Many



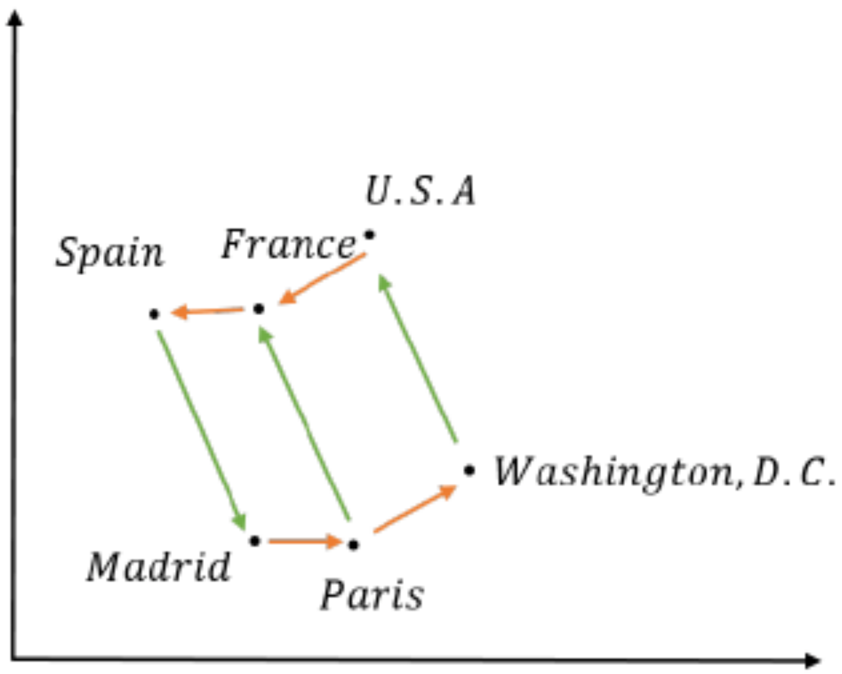
Music generation Some style / nothing → 

SURF

Will not cover

Bi-directional

- We did not cover any natural language processing (NLP).
  - **Word embeddings**, representing words as vectors
- RNNs have been very successful in NLP over the years.
- NLP requires **a lot** of data preprocessing and large models.
- Same models used.



Word Embedding Space

Summary

Topic: RNNs.

- Training RNNs.
- Long term dependencies, LSTM & GRU.
- Residual connections.

Notebook: RNN using GRU.

Topic: Improving RNNS.

- Regularisation in RNNs.
- Going deep, stacking layers.

Notebook: Improving RNNs.