Deep learning for time series forecasting

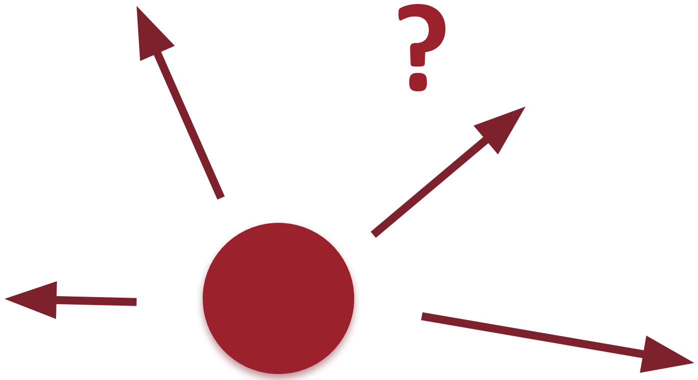
Pedro Lara Benítez

Manuel Carranza García

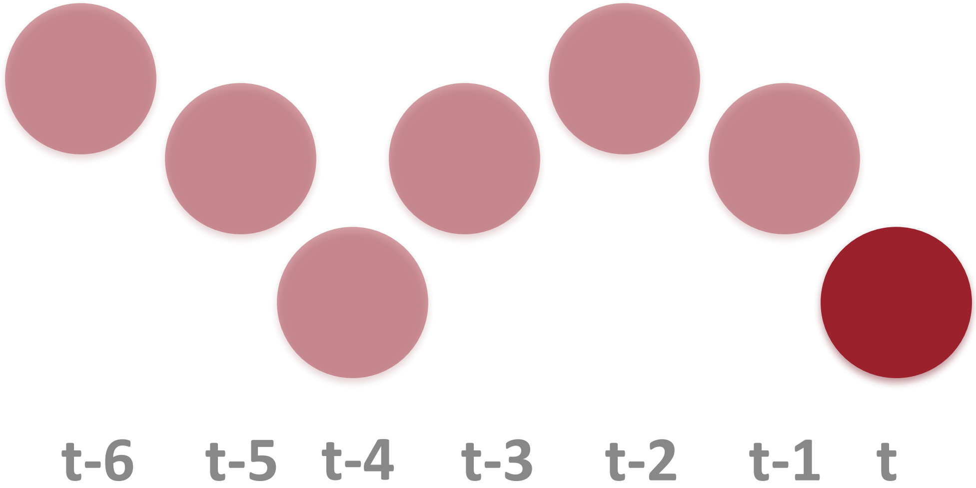
Time series forecasting

- Data particularities
- Problem to solve

Given an image of a ball, can you predict where it will go next?

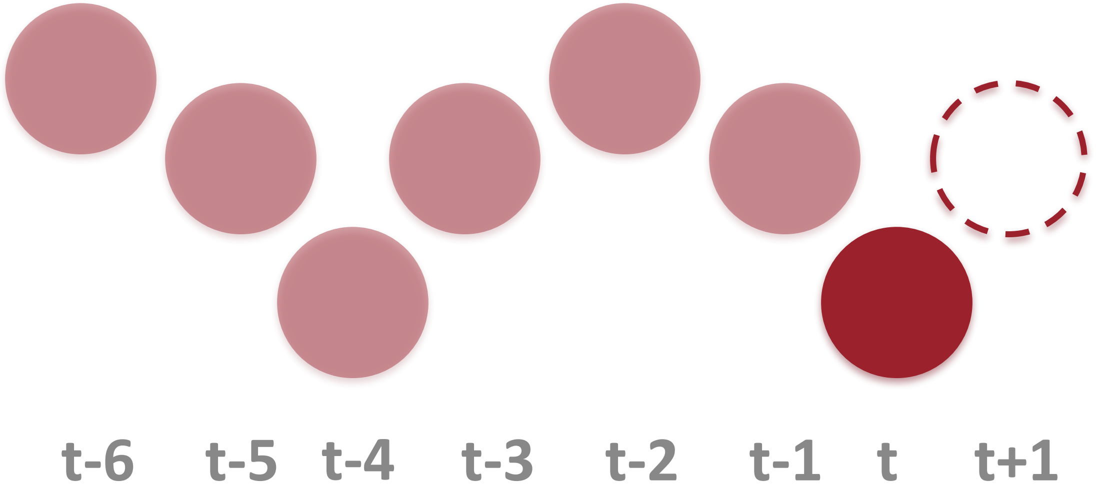


Given an image of a ball, can you predict where it will go next?



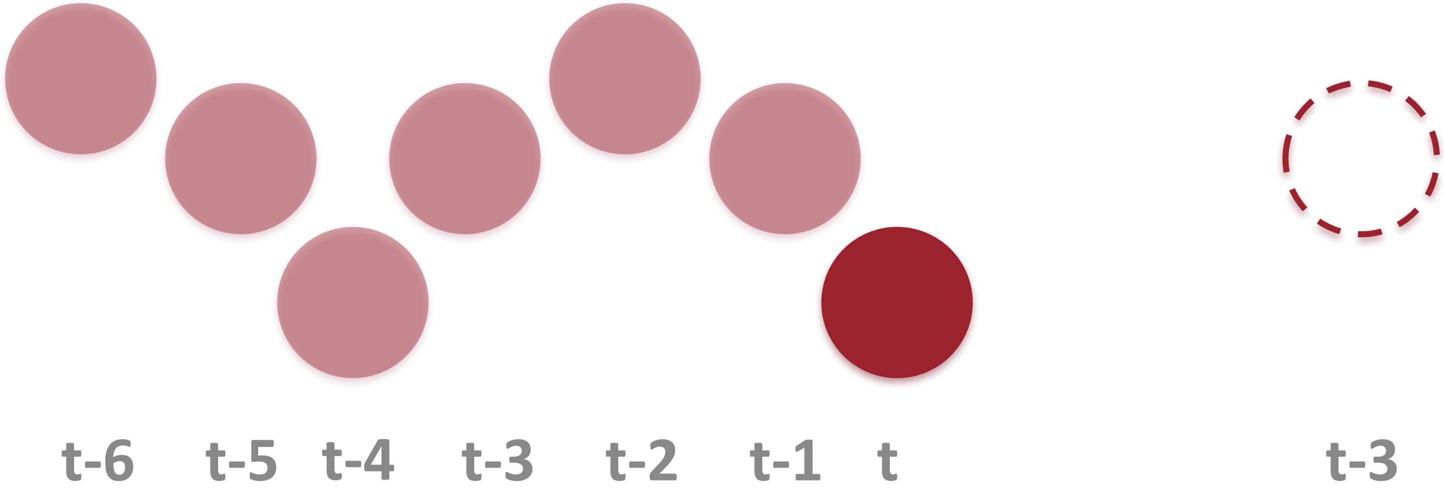
Given an image of a ball, can you predict where it will go next?

Single step forecasting



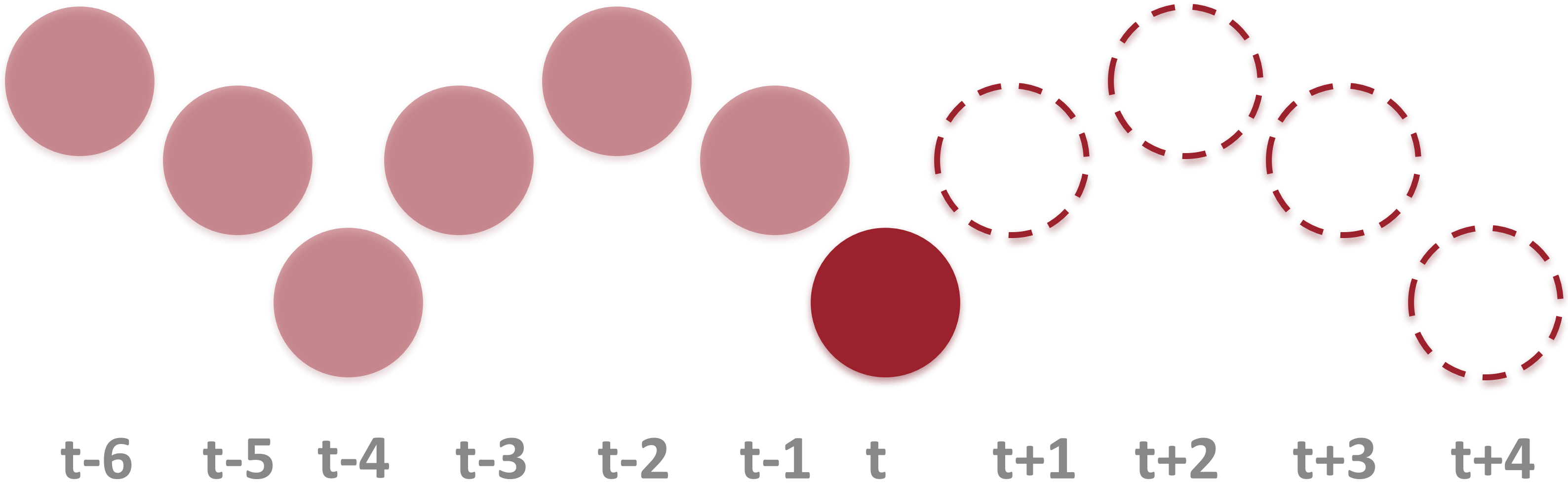
Given an image of a ball, can you predict where it will go next?

Single step forecasting

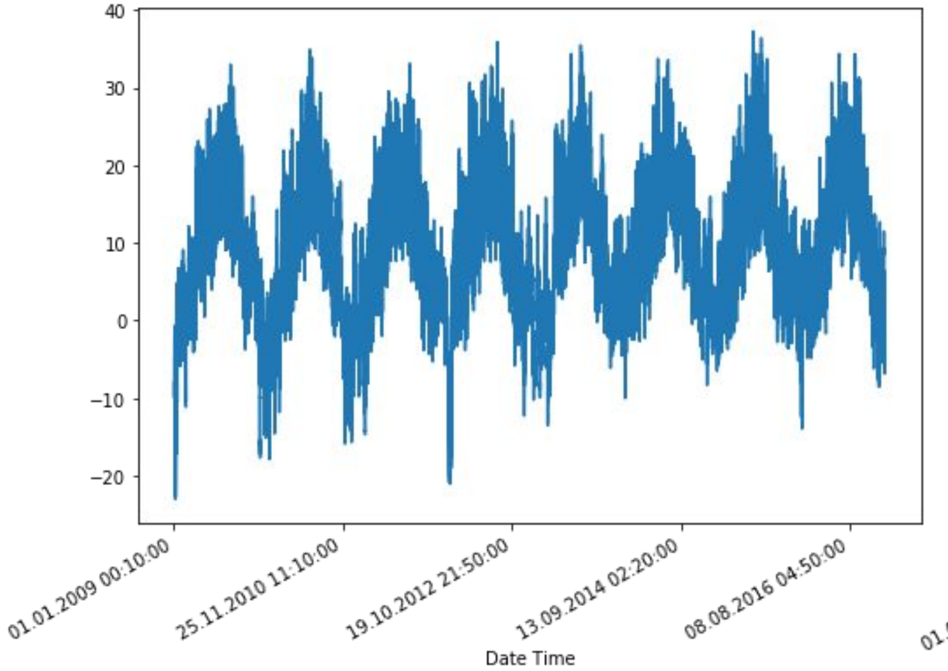


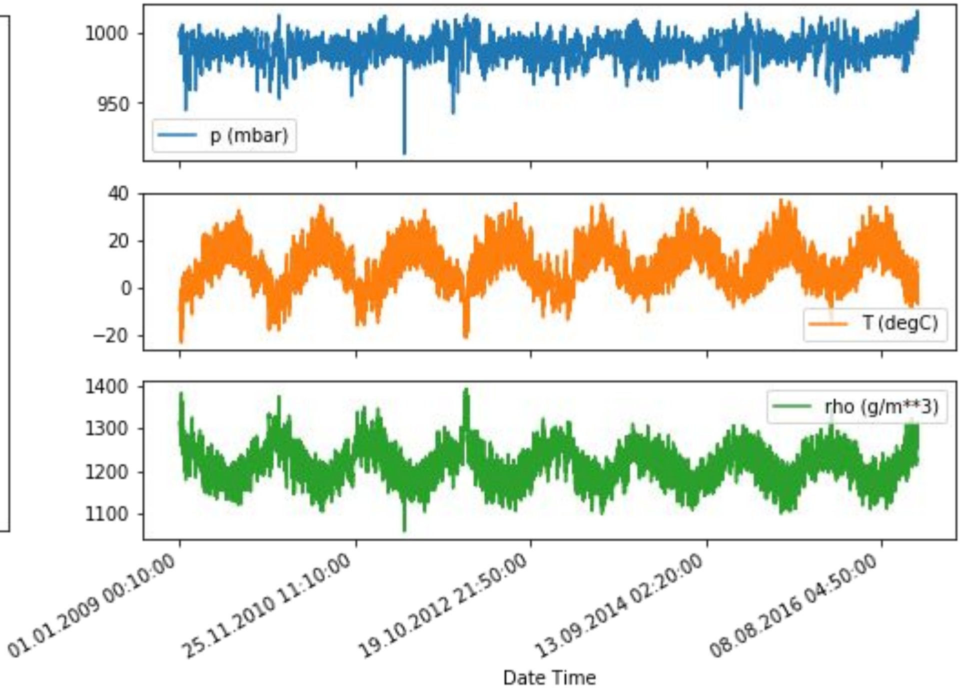
Given an image of a ball, can you predict where it will go next?

Multi step forecasting



Univariate vs Multivariate time series





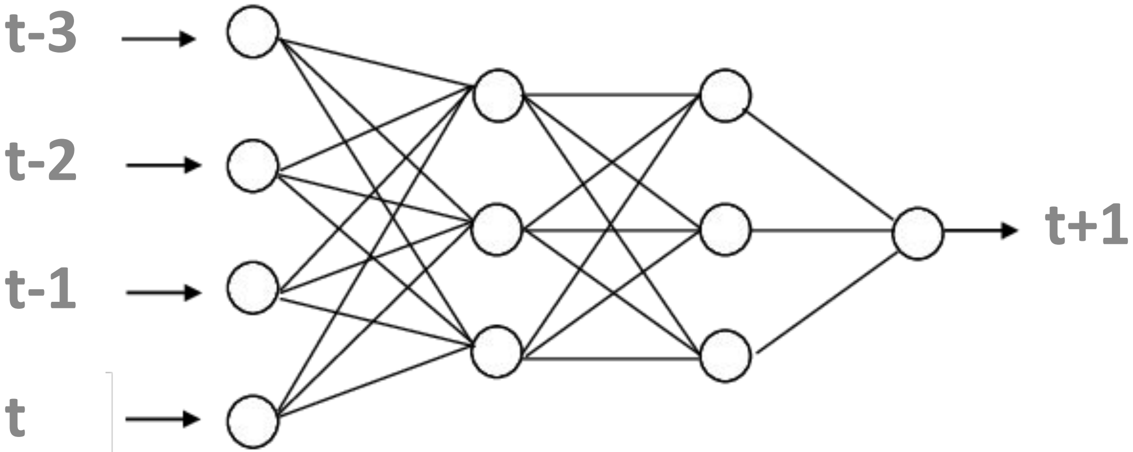
Deep learning models for TSF

- Fully connected
  - MLP
- Recurrent neural network
  - Elman
  - LSTM
  - GRU
- Convolutional neural network
  - CNN
  - TCN

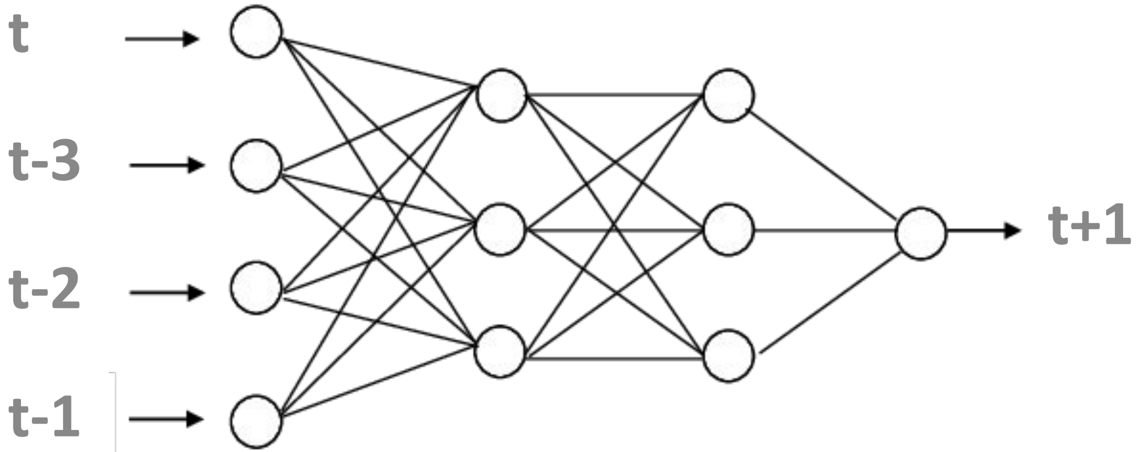
Deep learning models for TSF

- Fully connected
  - MLP
- Recurrent neural network
  - Elman
  - LSTM
  - GRU
- Convolutional neural network
  - CNN
  - TCN

Multi-layer Perceptron



Multi-layer Perceptron



Deep learning models for TSF

- Fully connected
  - MLP
- Recurrent neural network
  - Elman
  - LSTM
  - GRU
- Convolutional neural network
  - CNN
  - TCN

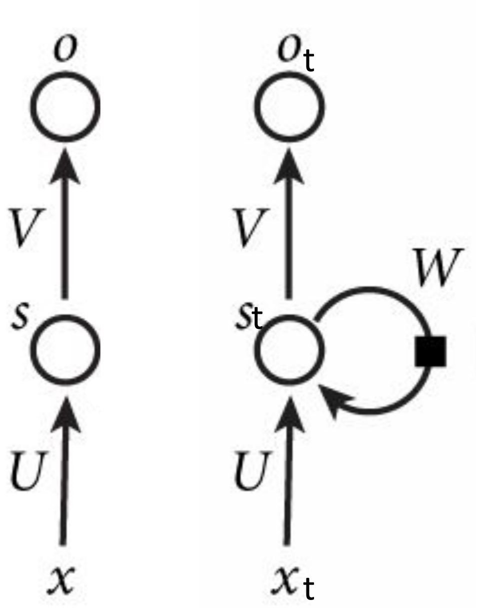
Recurrent Neural Network

Requirements for time series modelling

- Handle **variable-length** sequences
- Track **long-term** dependencies
- Maintain information about **order**
- **Share parameters** across the sequence

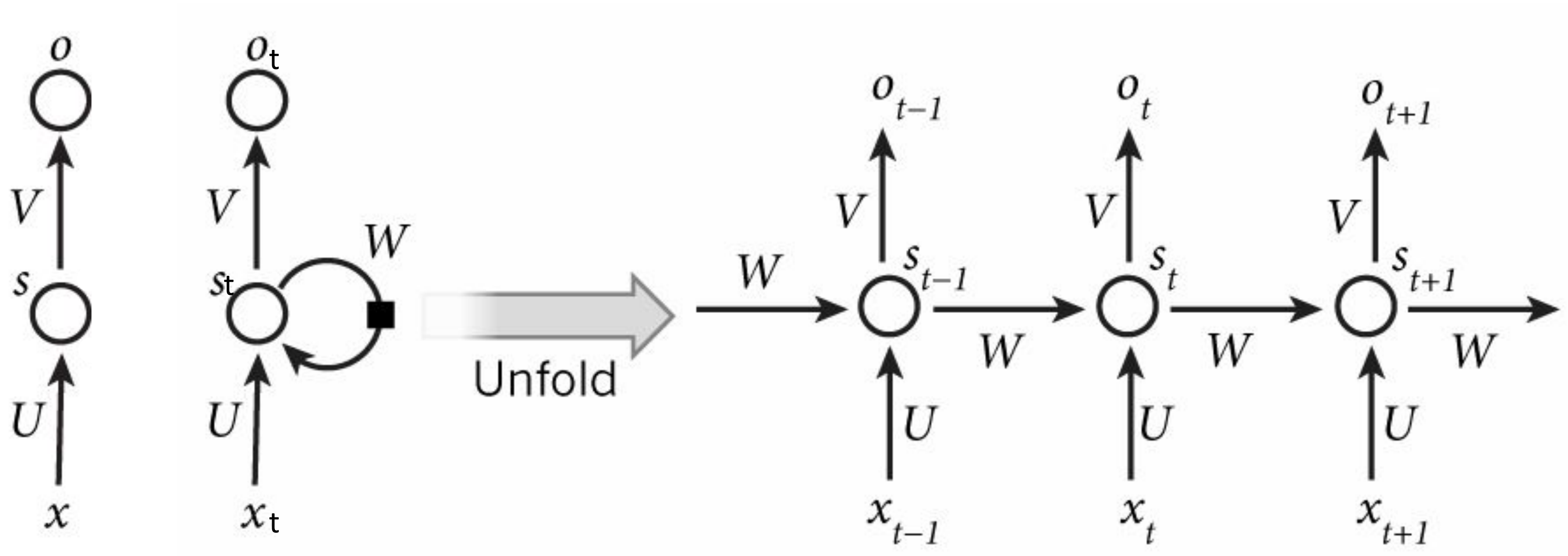
Elman Recurrent Neural Network

Elman Recurrent Neural Network



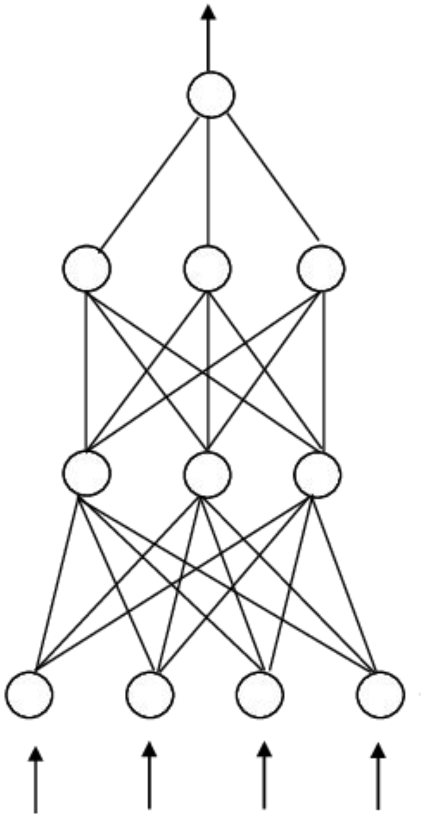
Perceptron RNN $S_t = f(W, U, X_t, S_{t-1}) = tanh(U \times X_t + W \times S_{t-1})$

Elman Recurrent Neural Network

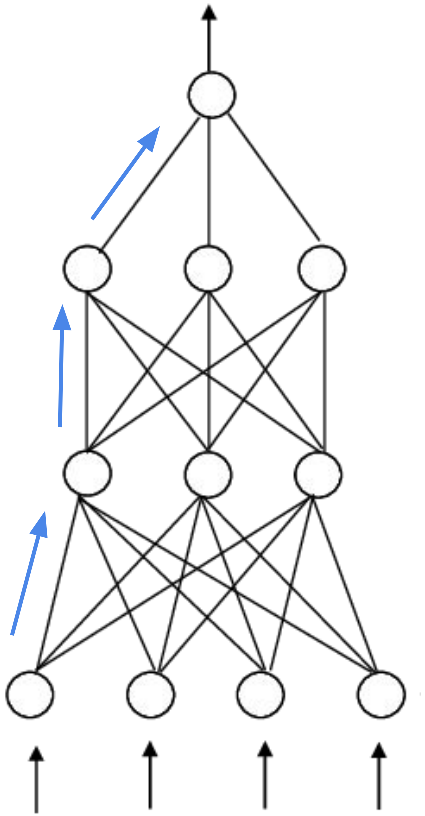


Perceptron RNN $S_t = f(W, U, X_t, S_{t-1}) = tanh(U \times X_t + W \times S_{t-1})$

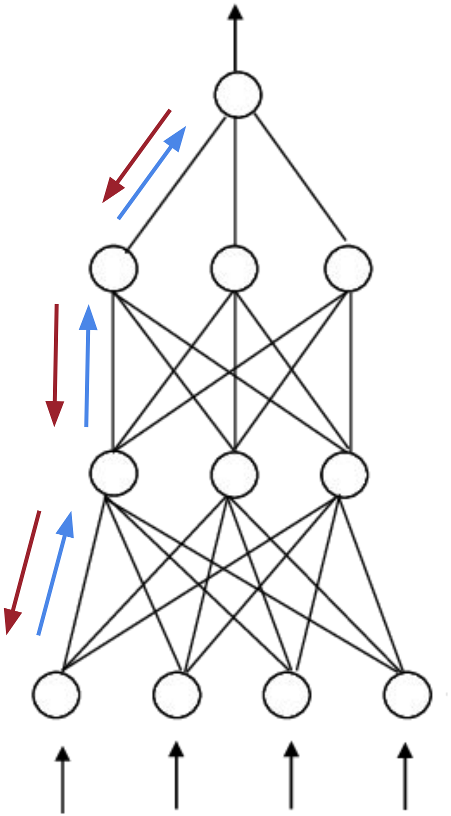
Backpropagation through time



Backpropagation through time

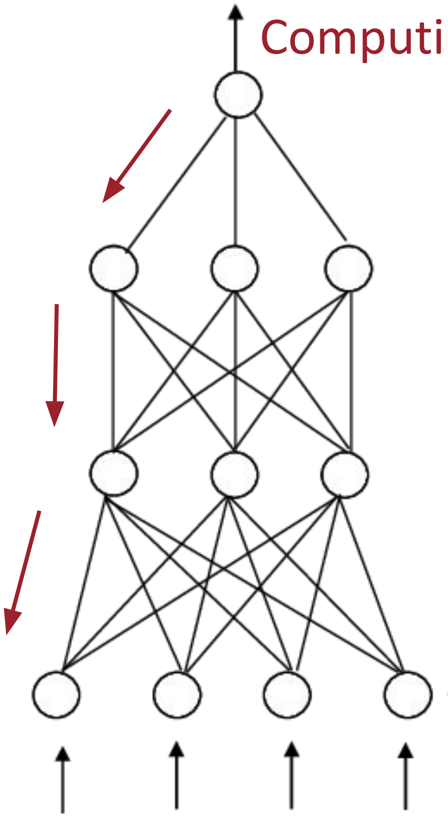


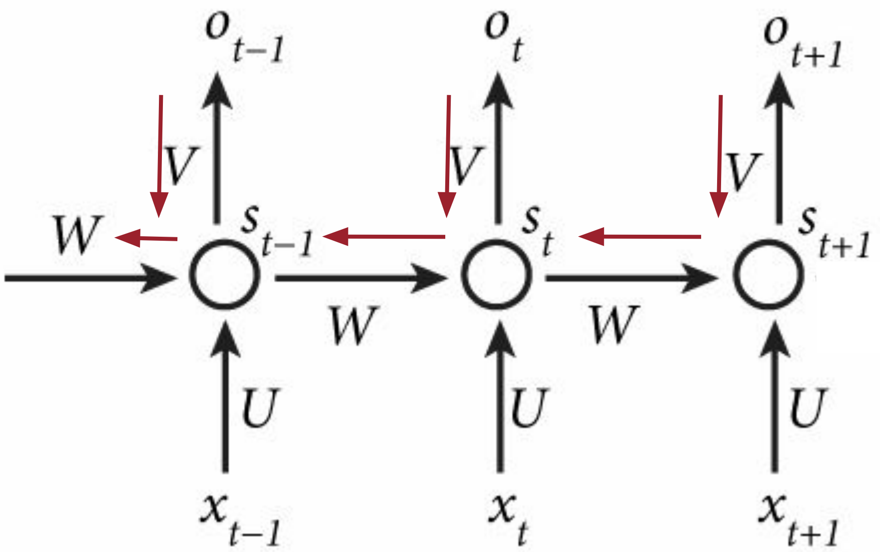
Backpropagation through time



Backpropagation through time

Computing the gradient involves **many factors of W** (and repeated **f’**)





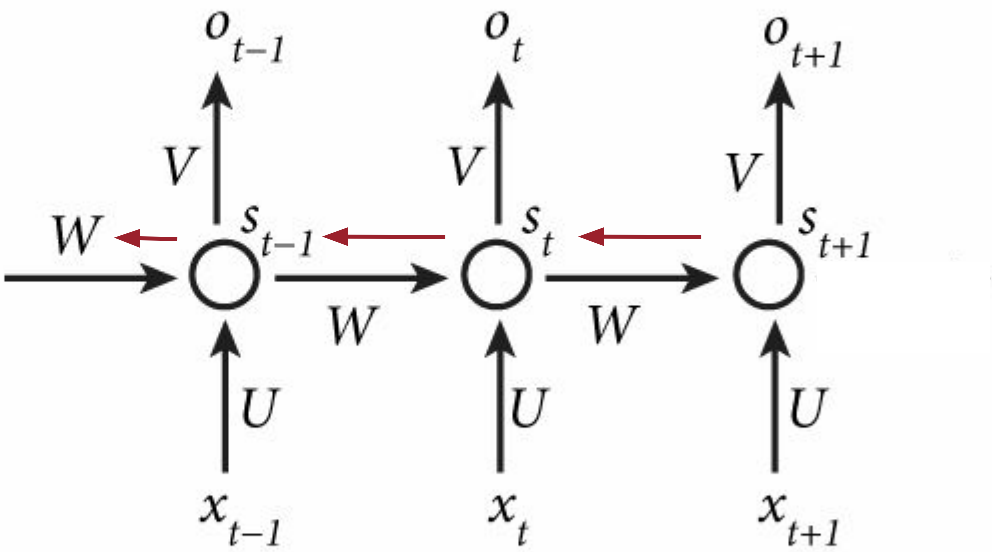
RNNs problems

Exploding gradient

Many values to compute

Vanishing gradient

Multiply by small numbers



RNNs problems

Exploding gradient

Many values to compute

Vanishing gradient

Multiply by small numbers

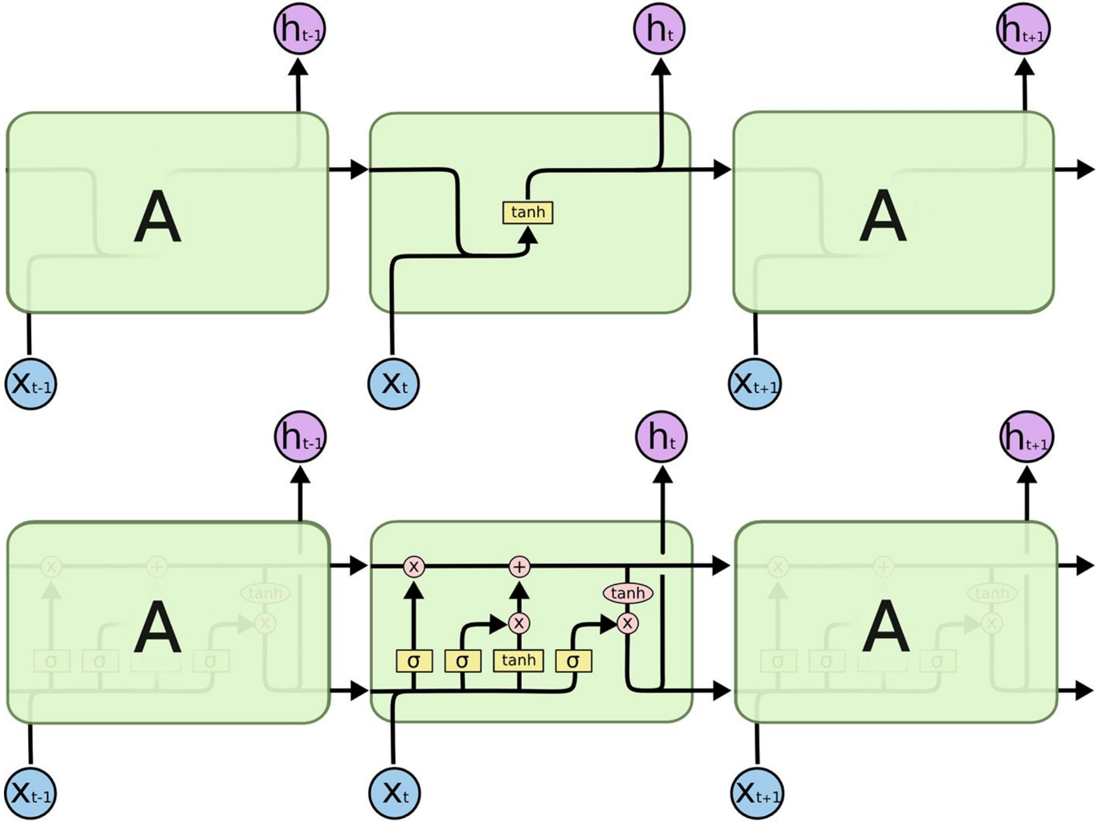
LSTM as the solution

(Long short-term memory)

Deep learning models for TSF

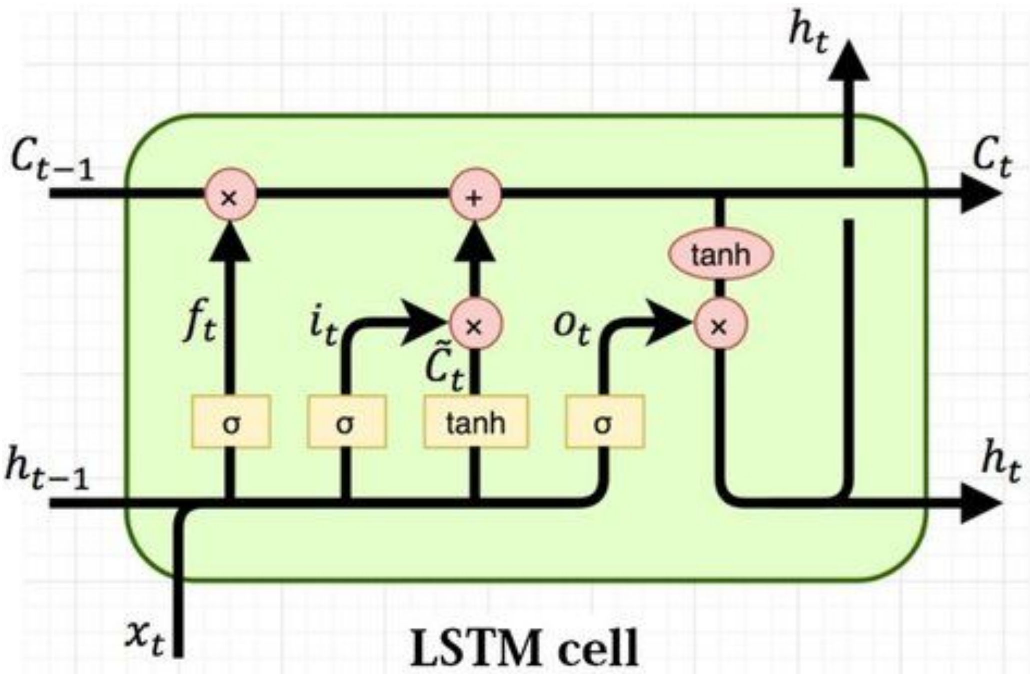
- Fully connected
  - MLP
- Recurrent neural network
  - Elman
  - LSTM
  - GRU
- Convolutional neural network
  - CNN
  - TCN

LSTM RNN



h_{t-1}
h_t
h_{t+1}
A
tanh
A
X_{t-1}
X_t
X_{t+1}
h_{t-1}
h_t
h_{t+1}
A
σ
σ
tanh
σ
tanh
A
X_{t-1}
X_t
X_{t+1}

LSTM RNN



$$i_t = \sigma\left(x_t U^i + h_{t-1} W^i\right)$$

$$f_t = \sigma\left(x_t U^f + h_{t-1} W^f\right)$$

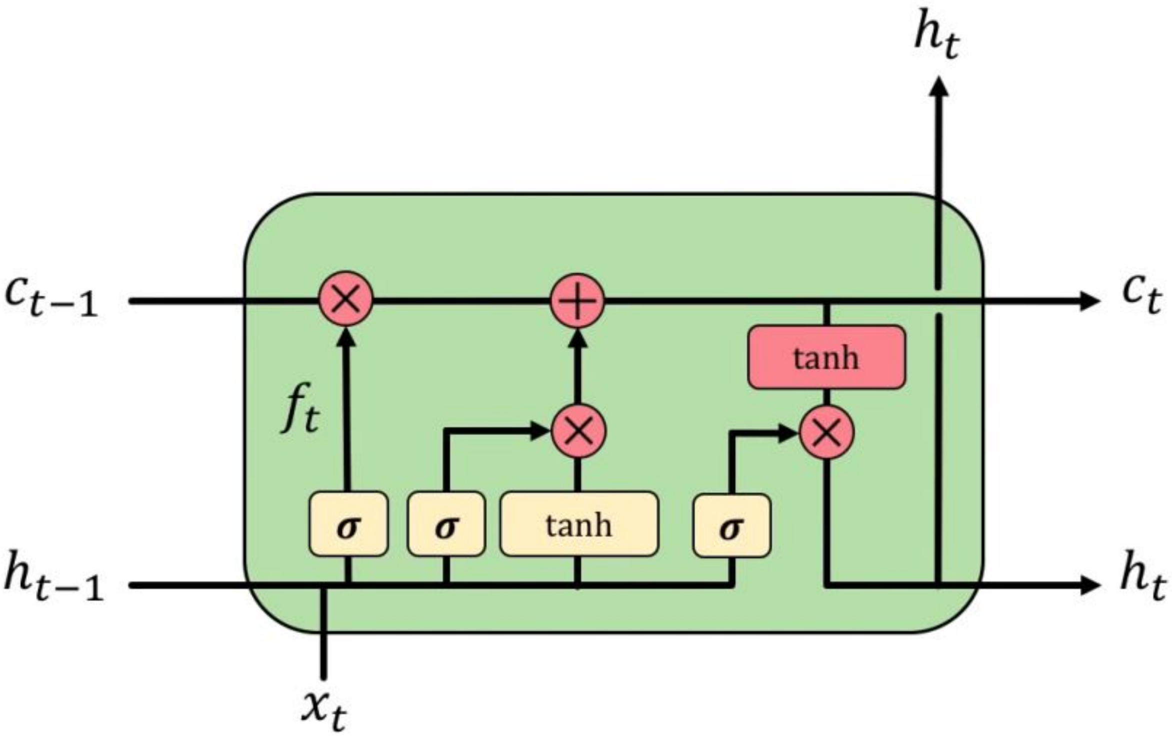
$$o_t = \sigma\left(x_t U^o + h_{t-1} W^o\right)$$

$$\tilde{C}_t = \tanh\left(x_t U^g + h_{t-1} W^g\right)$$

$$C_t = \sigma\left(f_t * C_{t-1} + i_t * \tilde{C}_t\right)$$

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

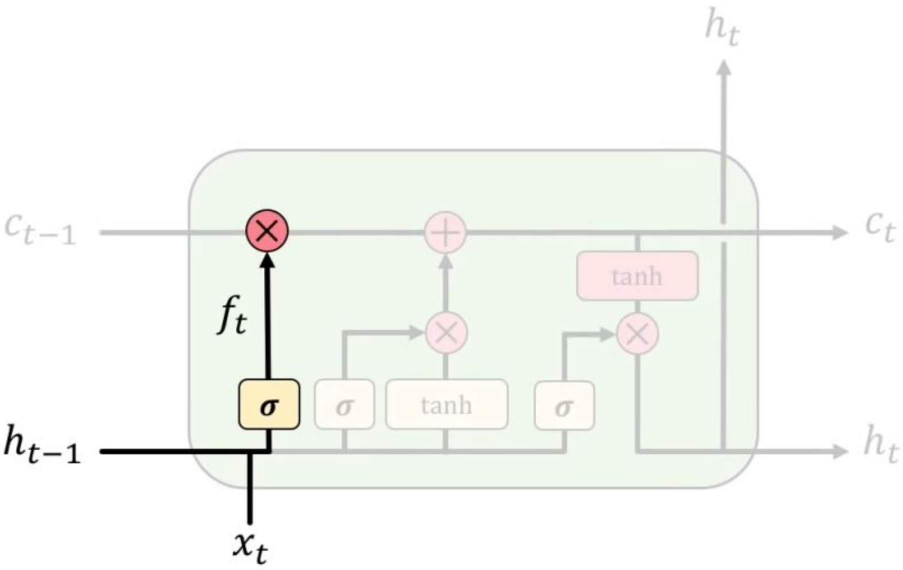
LSTM RNN



1. Forget
2. Update
3. Output

LSTM RNN - Forget

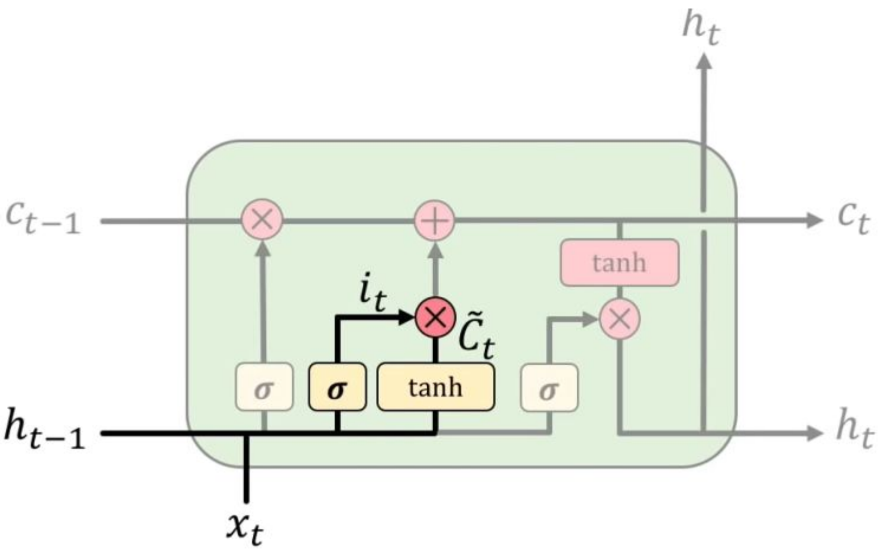
LSTMs **forget irrelevant** parts of the previous state



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

- Use previous cell output and input
- Sigmoid: value 0 and 1 – "completely forget" vs. "completely keep"

LSTM RNN - Update

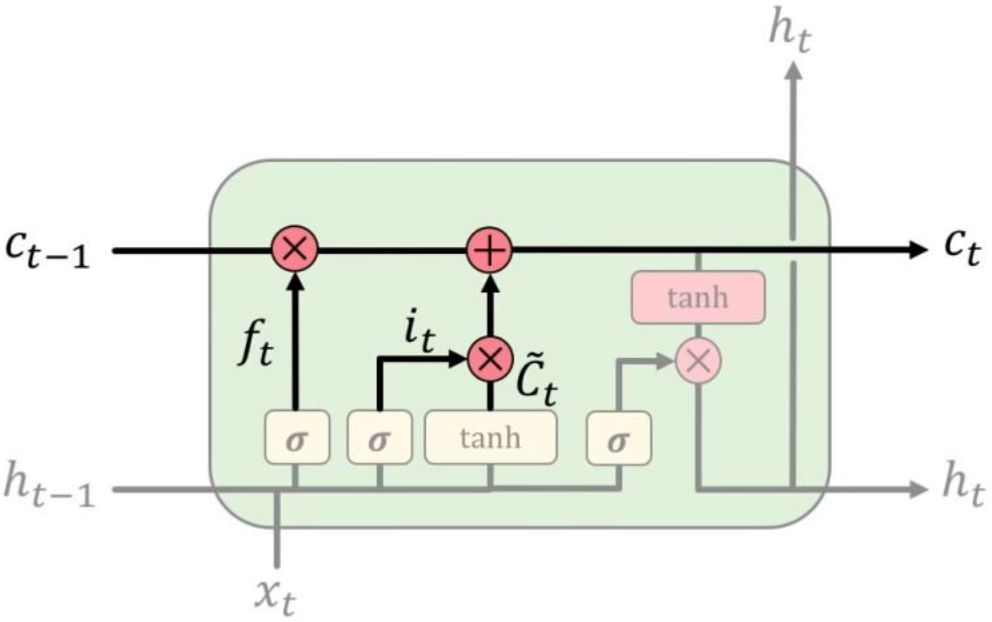


$$i_t = \sigma(\boldsymbol{W_i}[\, h_{t-1}, x_t \,] + b_i)$$

$$\tilde{C}_t = \tanh(\boldsymbol{W_C}[\, h_{t-1}, x_t \,] + b_C)$$

- Sigmoid layer: decide what values to update
- Tanh layer: generate new vector of "candidate values" that could be added to the state

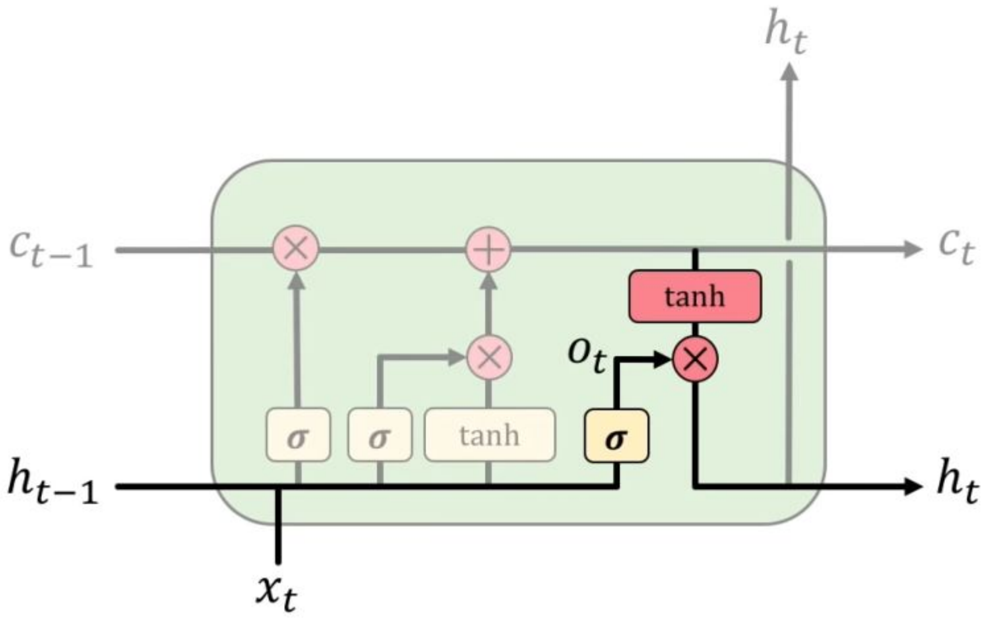
LSTM RNN - Update



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

- Apply forget operation to previous internal cell state: $f_t * C_{t-1}$
- Add new candidate values, scaled by how much we decided to update: $i_t * \tilde{C}_t$

LSTM RNN - Output



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

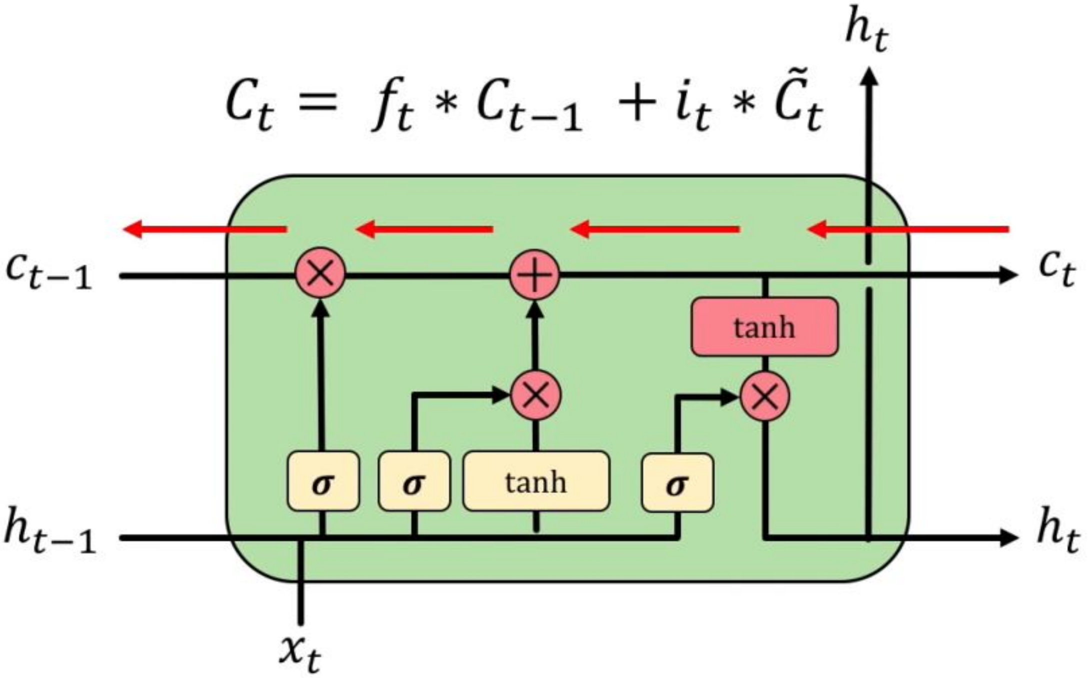
$$h_t = o_t * \tanh(C_t)$$

- Sigmoid layer: decide what parts of state to output
- Tanh layer: squash values between -1 and 1
- $o_t * \tanh(C_t)$: output filtered version of cell state

LSTM RNN - Backpropagation flow

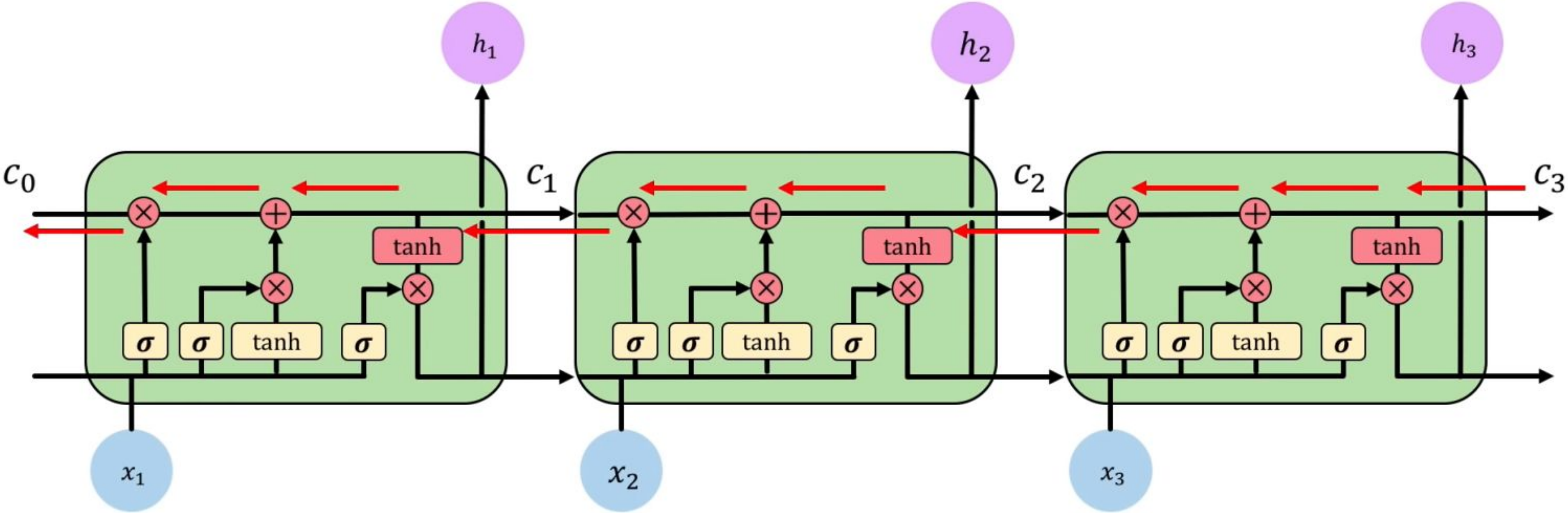
Backpropagation from C_t to C_{t-1} requires only elementwise multiplication!
No matrix multiplication → avoid vanishing gradient problem.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$



LSTM RNN - Backpropagation flow

Uninterrupted gradient flow



LSTM - Key concept

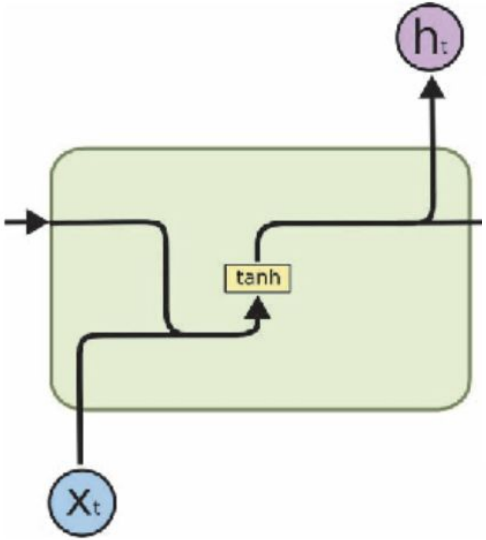
1. Maintain a **separate cell state** from what is outputted
2. Use **gates** to control the **flow of information**
   - Forget gate gets rid of irrelevant information
   - Selectively update cell state
   - Output gate return a filtered version of the cell state
3. Backpropagation from C_t to C_{t-1} doesn't require matrix multiplication: **uninterrupted gradient flow**

Deep learning models for TSF

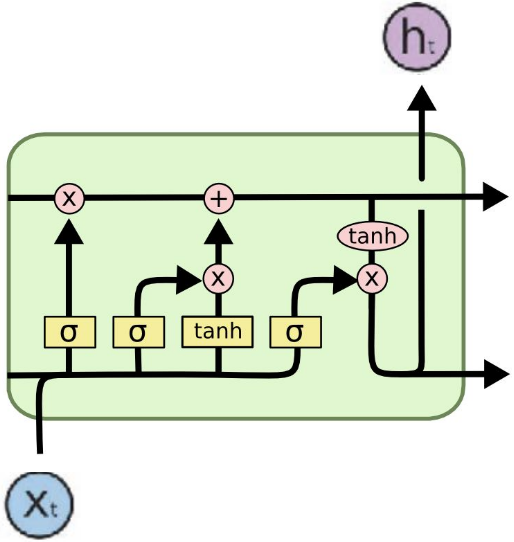
- Fully connected
  - MLP
- Recurrent neural network
  - Elman
  - LSTM
  - GRU
- Convolutional neural network
  - CNN
  - TCN

GRU RNN

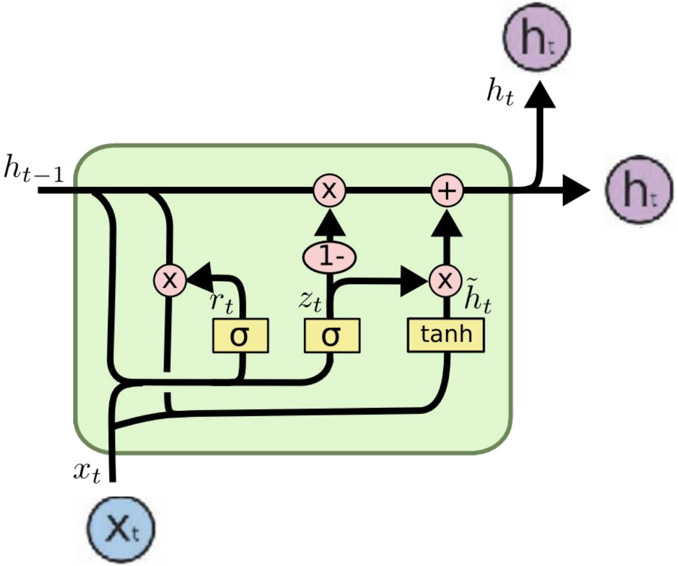
Elman RNN



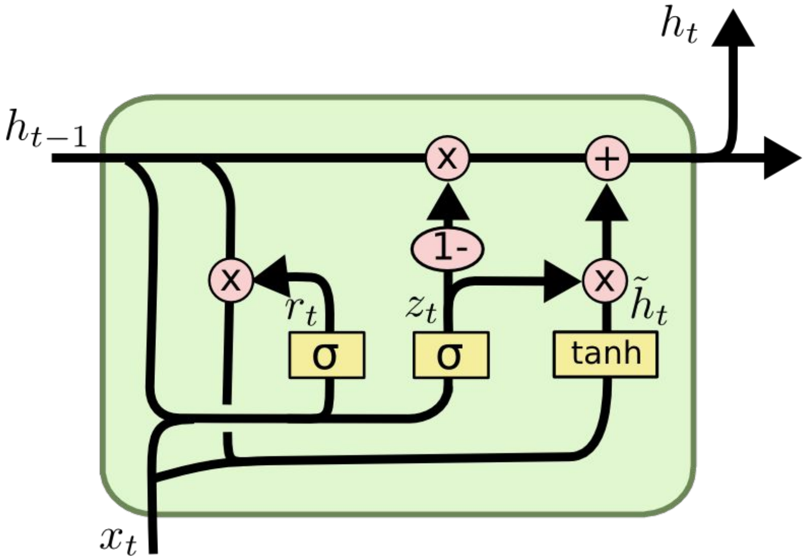
LSTM RNN



GRU RNN



GRU RNN



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

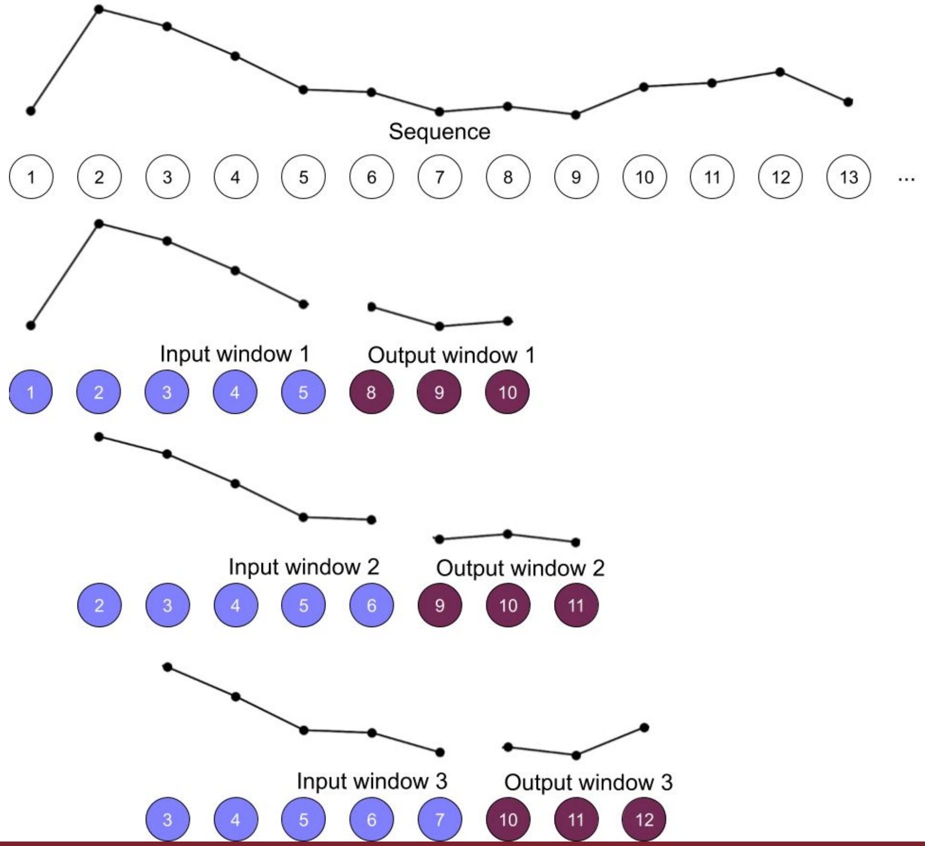
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Code example

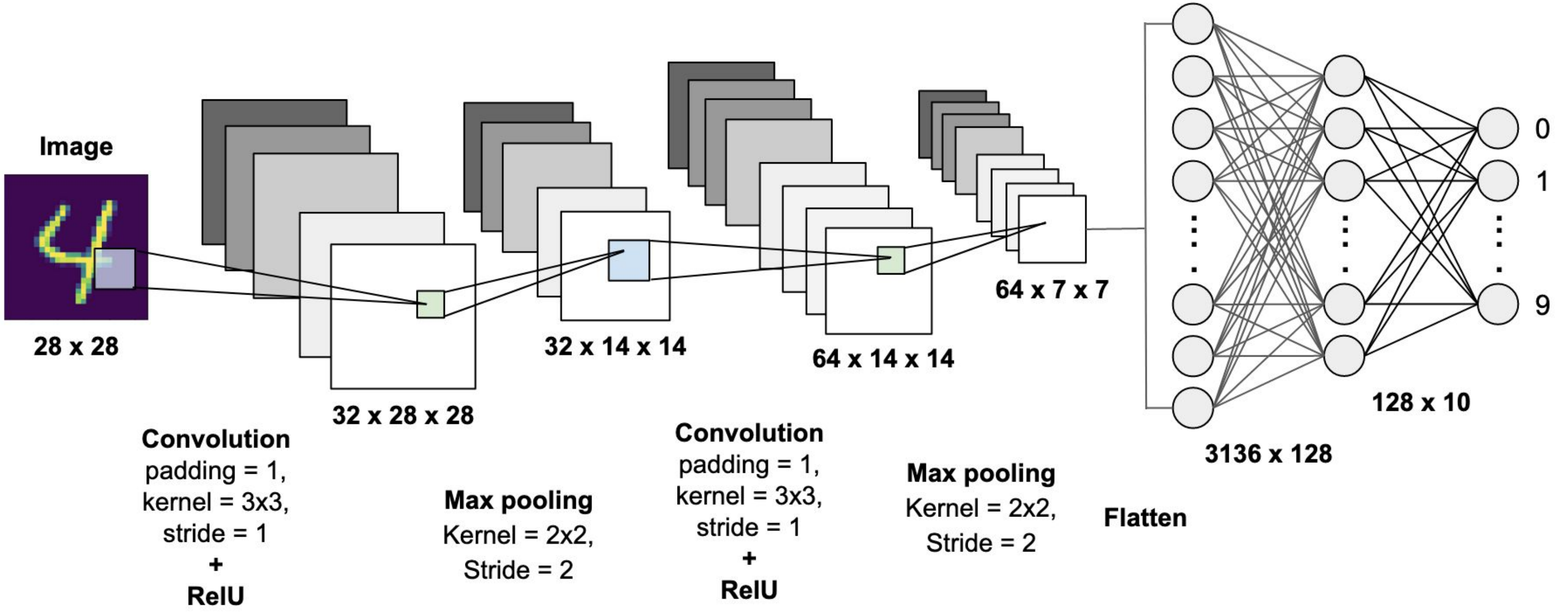
Moving window strategy



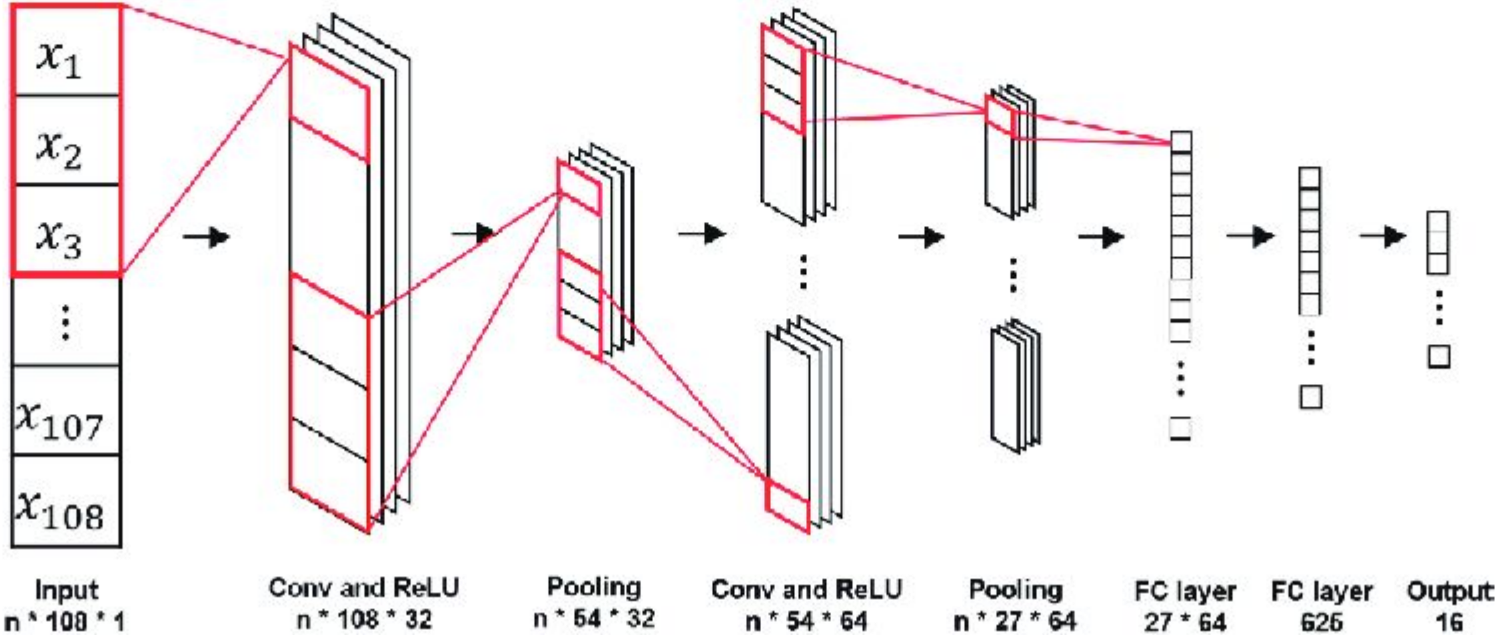
Deep learning models for TSF

- Fully connected
  - MLP
- Recurrent neural network
  - Elman
  - LSTM
  - GRU
- Convolutional neural network
  - CNN
  - TCN

Convolutional NN

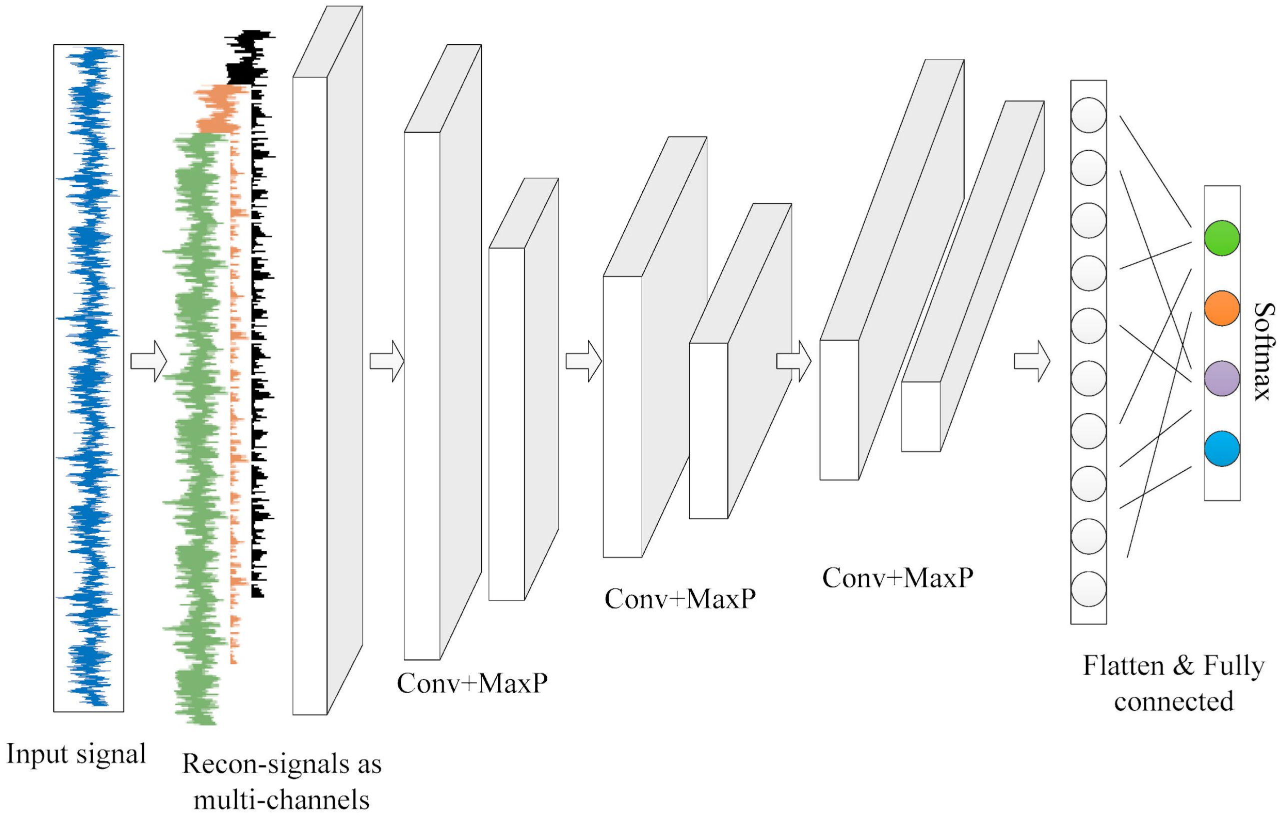


CNN

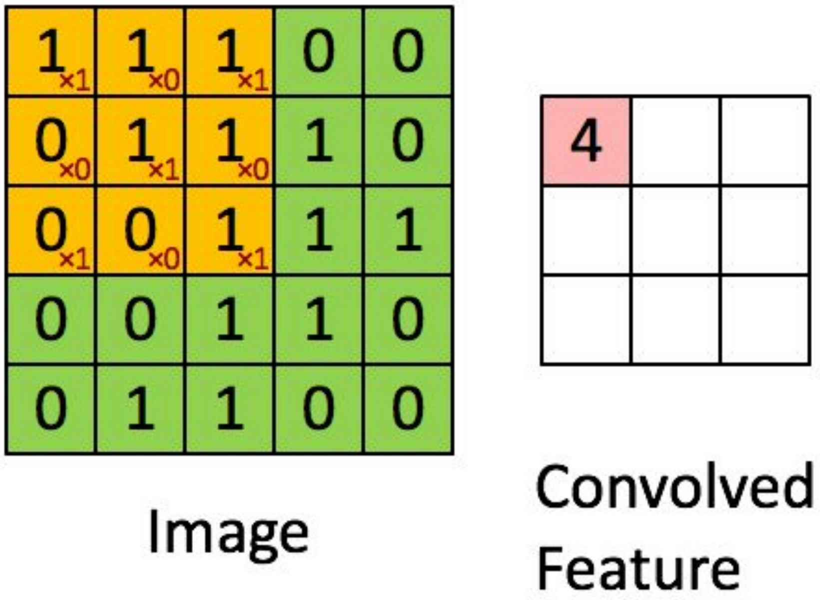


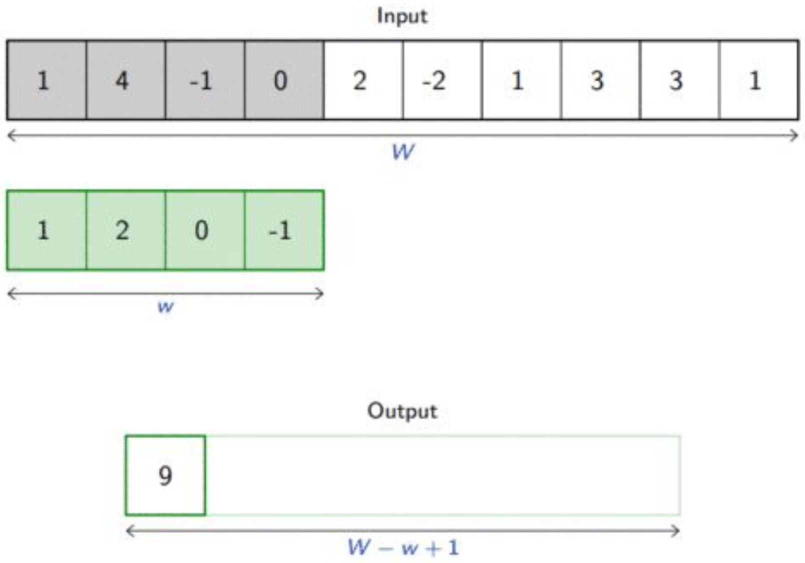
x_1
x_2
x_3
x_{107}
x_{108}
Input
n * 108 * 1
Conv and ReLU
n * 108 * 32
Pooling
n * 54 * 32
Conv and ReLU
n * 54 * 64
Pooling
n * 27 * 64
FC layer
27 * 64
FC layer
625
Output
16

CNN

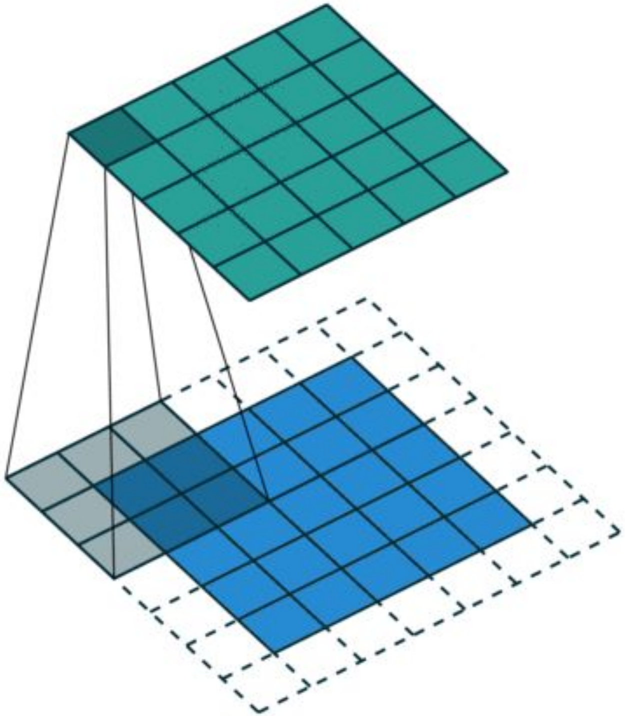

Softmax
Conv+MaxP
Conv+MaxP
Conv+MaxP
Input signal
Recon-signals as multi-channels
Flatten & Fully connected

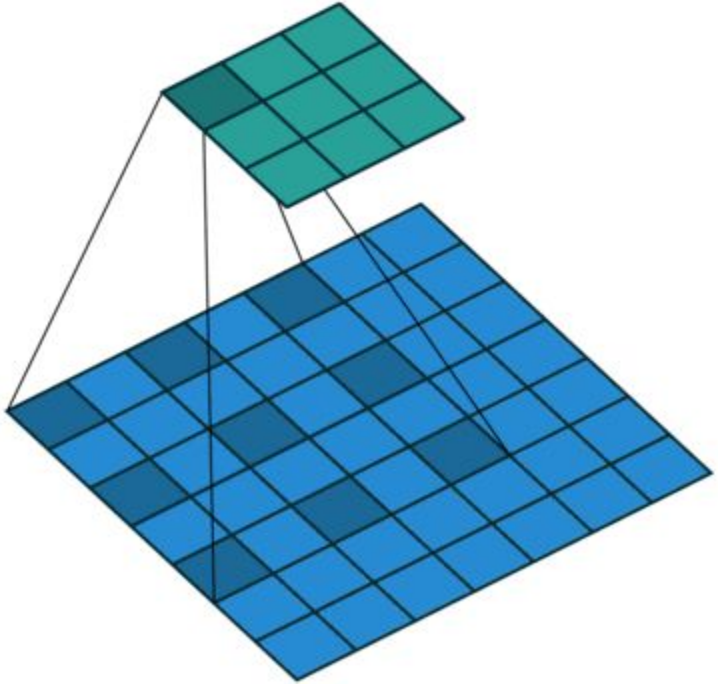
Convolutional operation



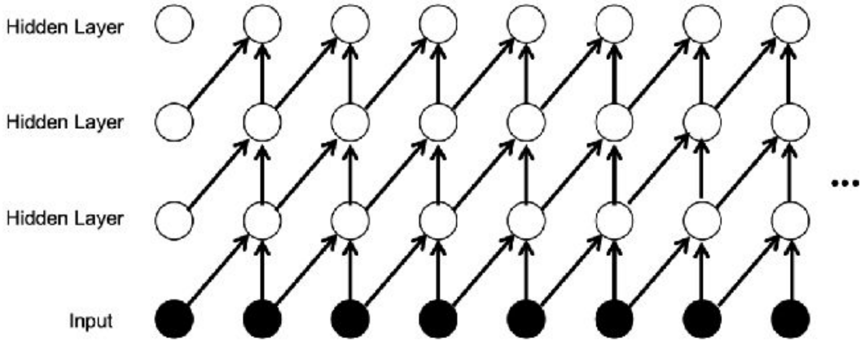


Dilated CNN

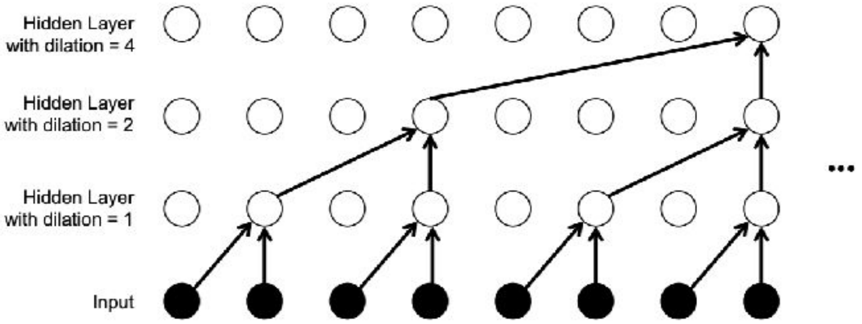




Dilated CNN



(a) Standard 1D casual convolution

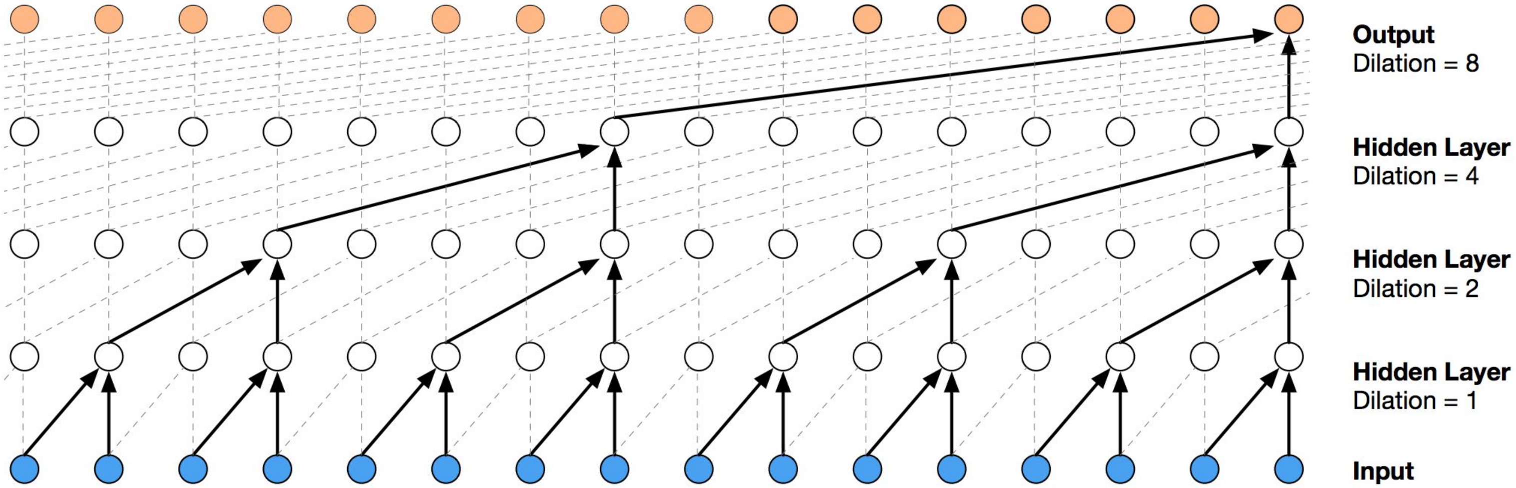


(b) Dilated 1D casual convolution

Deep learning models for TSF

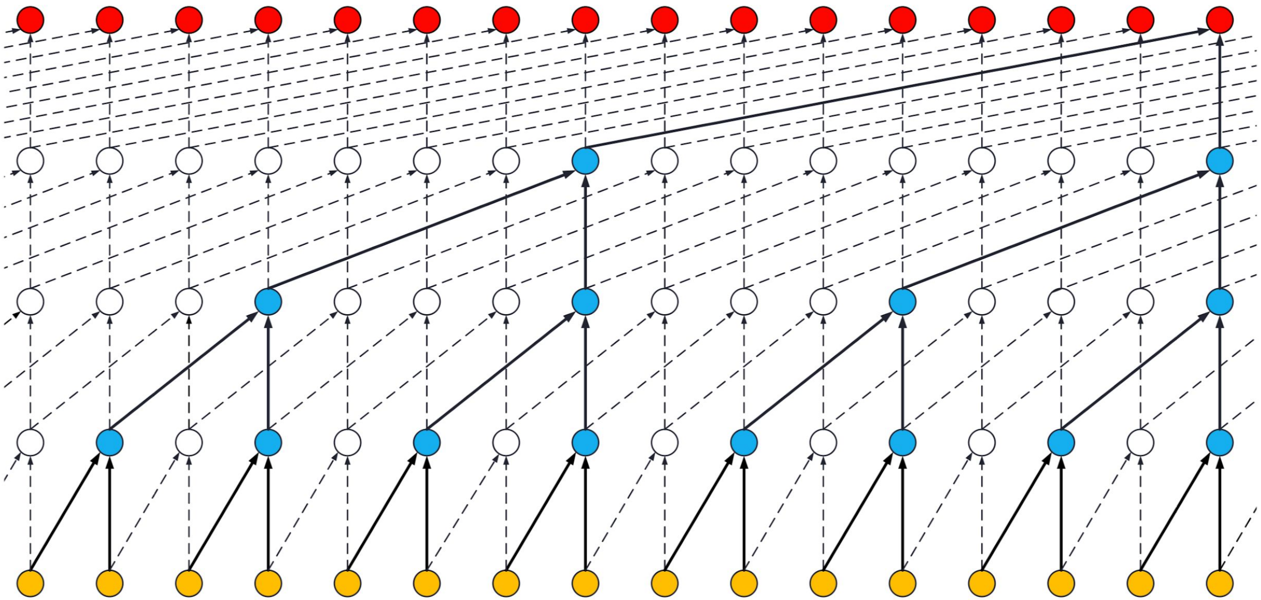
- Fully connected
  - MLP
- Recurrent neural network
  - Elman
  - LSTM
  - GRU
- Convolutional neural network
  - CNN
  - TCN

Temporal Convolutional Network (TCN)



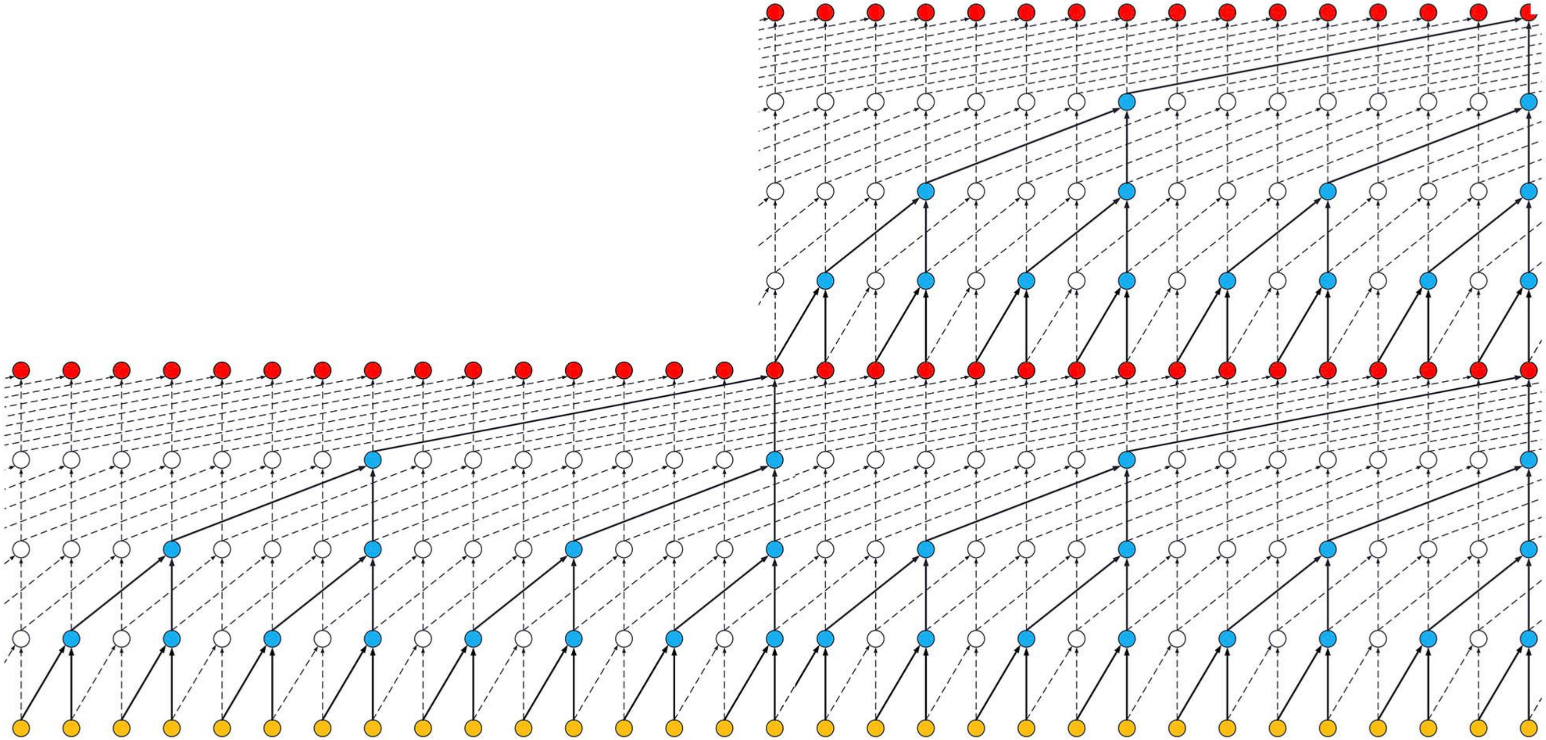
TCN - Receptive field

Receptive field = nb_stacks_of_residuals_blocks * kernel_size * last_dilation



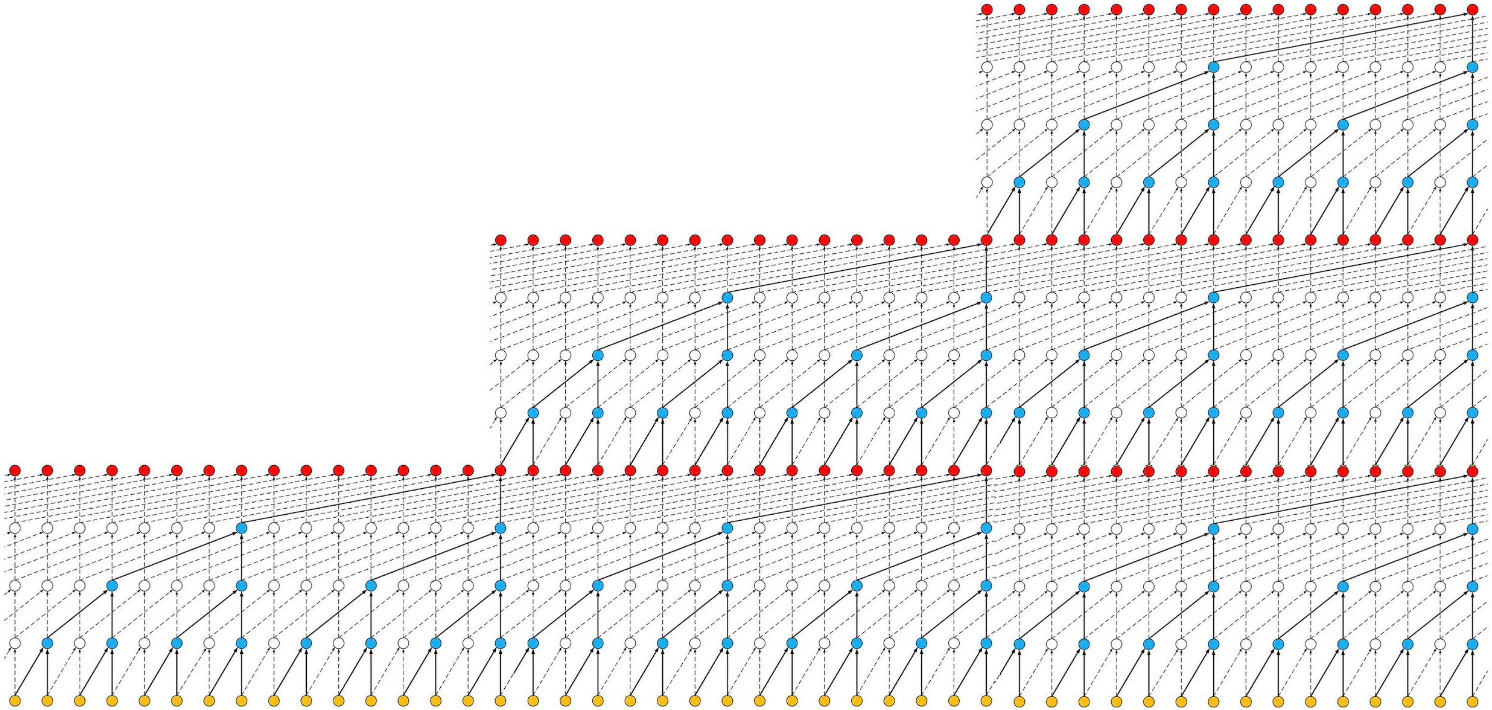
kernel size = 2, dilations = [1,2,4,8], 1 block

TCN - Receptive field



kernel size = 2, dilations = [1,2,4,8], 2 blocks

TCN - Receptive field



kernel size = 2, dilations = [1,2,4,8], 3 blocks

Code example

TensorFlow + Keras

Resources

TCN implementation: https://github.com/philipperemy/keras-tcn

LSTM video explanation: https://www.youtube.com/watch?v=_h66BW-xNgk