Deep Learning

A course about theory & practice

UNIVERSITÀ DI PAVIA

Deep Learning and Time Series

Marco Piastra

Recurrent Neural Networks

Recurrent Neural Networks

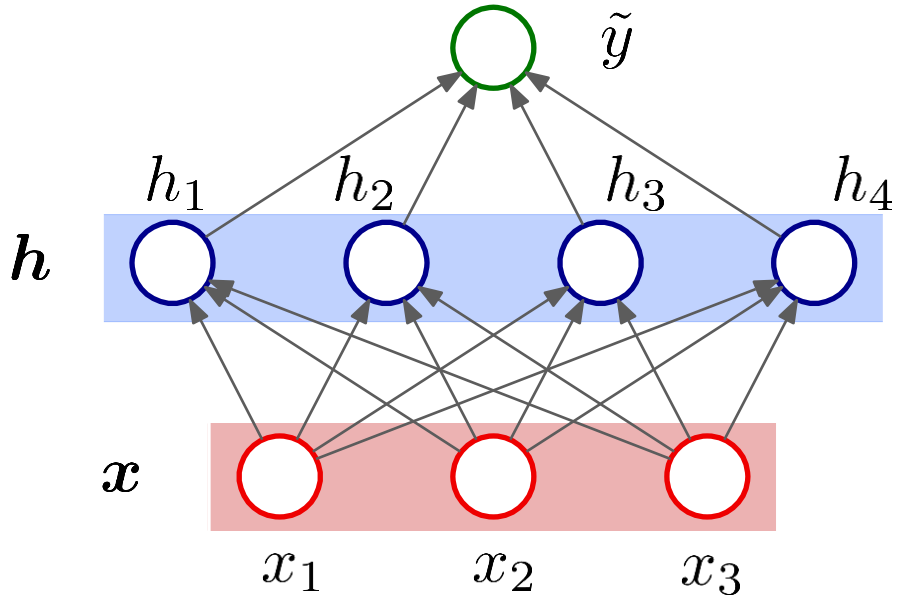
- **Feed-forward neural network**

$$\tilde{y} = \boldsymbol{w} \cdot \boldsymbol{h} + b$$

where:

$$\boldsymbol{h} := g(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$$

hidden — input



Recurrent Neural Networks

- **Feed-forward neural network**

$$\tilde{y} = \boldsymbol{w} \cdot \boldsymbol{h} + b$$

where: $\boldsymbol{h} := g(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$

hidden *input*

- **Recurrent Neural Network**

$$\tilde{y}^{(t)} = \boldsymbol{w} \cdot \boldsymbol{h}^{(t)} + b$$

where: $\boldsymbol{h}^{(t)} := g(\boldsymbol{W}\boldsymbol{x}^{(t)} + \boldsymbol{U}\boldsymbol{h}^{(t-1)} + \boldsymbol{b})$

hidden at t *input at* t *hidden at t-1*

The basic idea is to make the network output depend on the past 'history'

Recurrent Neural Networks

- **Recurrent Neural Network**

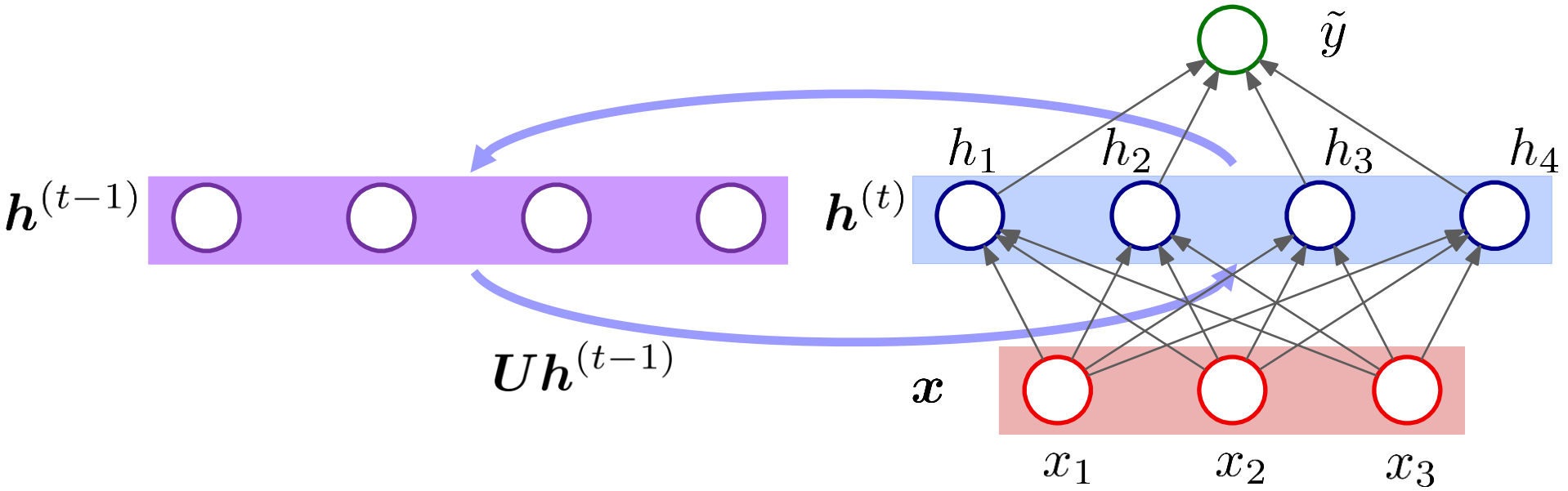
$$\tilde{y}^{(t)} = \boldsymbol{w} \cdot \boldsymbol{h}^{(t)} + b$$

where:

$$\boldsymbol{h}^{(t)} := g(\boldsymbol{W}\boldsymbol{x}^{(t)} + \boldsymbol{U}\boldsymbol{h}^{(t-1)} + \boldsymbol{b})$$

hidden at t *input at t* *hidden at $t-1$*

The basic idea is to make the network output depend on the past 'history'



Recurrent Neural Networks

- **Recurrent Neural Network**

$$\tilde{y}^{(t)} = \boldsymbol{w} \cdot \boldsymbol{h}^{(t)} + b$$

where:

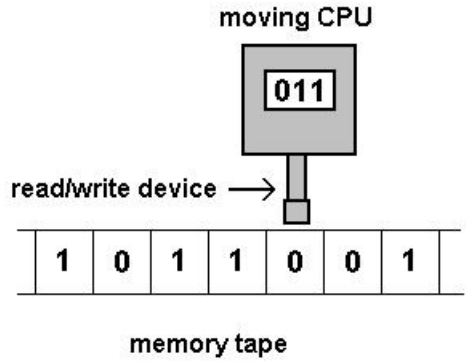
$$\boldsymbol{h}^{(t)} := g(\boldsymbol{W}\boldsymbol{x}^{(t)} + \boldsymbol{U}\boldsymbol{h}^{(t-1)} + \boldsymbol{b})$$

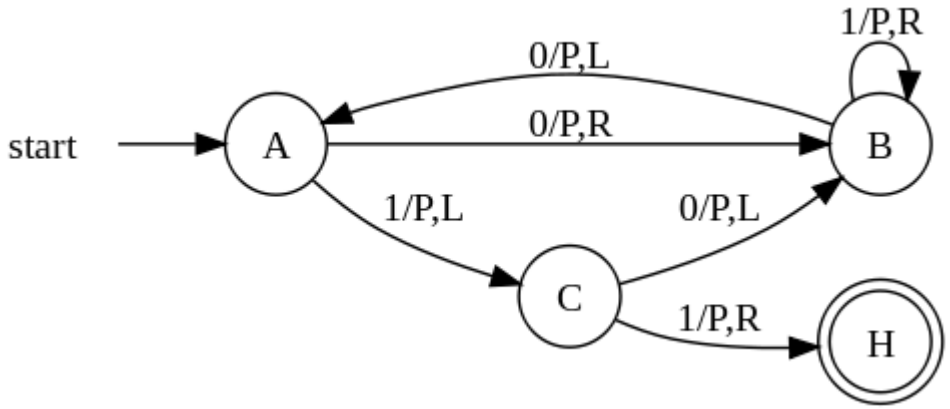
hidden at t *input at* t *hidden at* t-1

RNN are Turing Machines

- **Computational power of RNNs** (Siegelmann & Sontag, 1992)

  "RNNs can simulate any Turing machine"





This means that they can compute anything a Turing Machine could

[image from https://en.wikipedia.org/wiki/Turing_machine]

Recurrent Neural Networks

- **Recurrent Neural Network**

$$\tilde{y}^{(t)} = \boldsymbol{w} \cdot \boldsymbol{h}^{(t)} + b$$

where

$$\boldsymbol{h}^{(t)} := g(\boldsymbol{W}\boldsymbol{x}^{(t)} + \boldsymbol{U}\boldsymbol{h}^{(t-1)} + \boldsymbol{b})$$

hidden at t — *input at* t — *hidden at* t-1

- **General Properties**

Recurrent neural networks (RNNs) are more powerful than FF neural networks

It can approximate any Turing machine (i.e. a general, theoretical model of computation)

They are much harder to train than FF neural networks

Although, with *temporal unfolding (see next slide)*, *gradient descent* methods can be applied

Recurrent Neural Networks

- **Recurrent Neural Network**

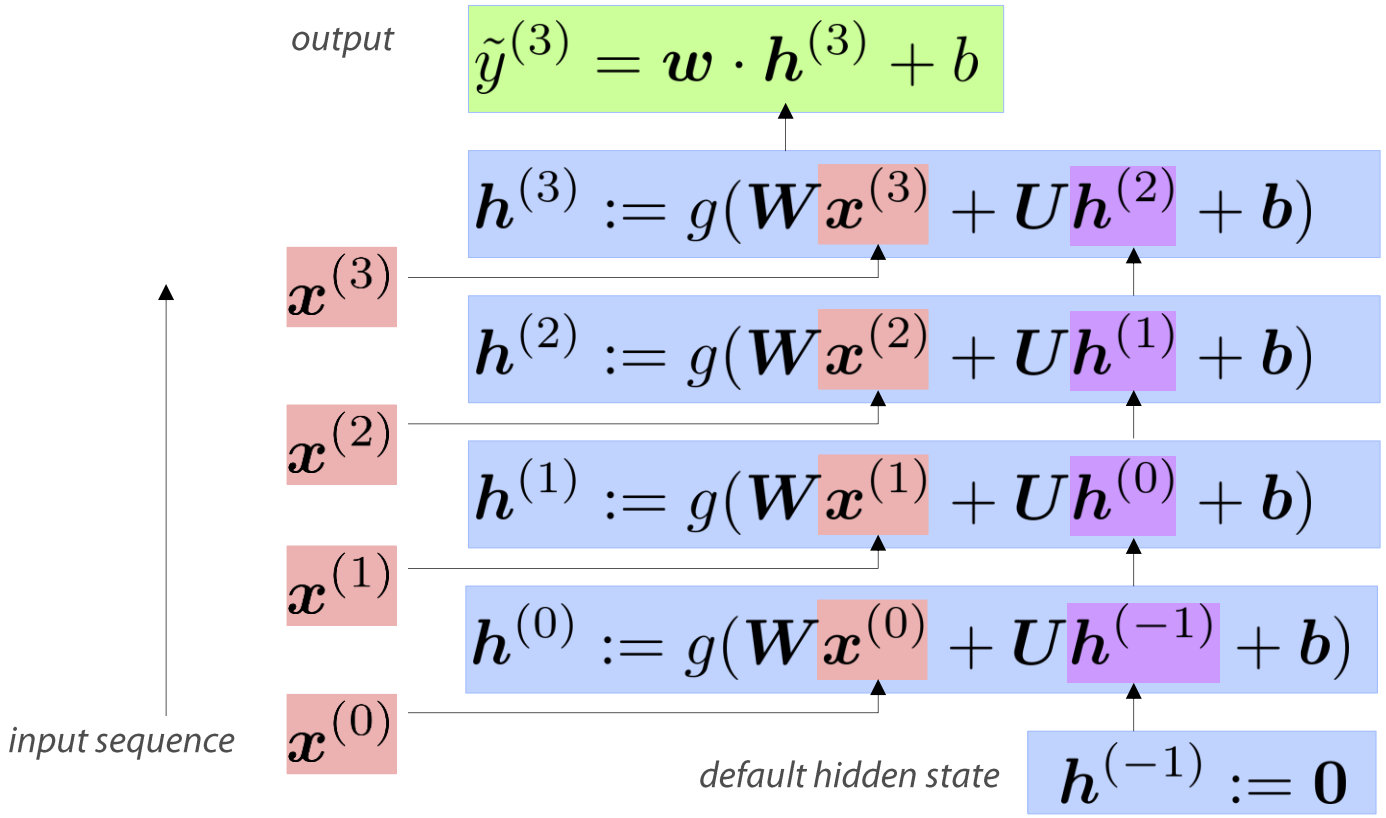
$$\tilde{y}^{(t)} = \boldsymbol{w} \cdot \boldsymbol{h}^{(t)} + b$$

where:

$$\boldsymbol{h}^{(t)} := g(\boldsymbol{W}\boldsymbol{x}^{(t)} + \boldsymbol{U}\boldsymbol{h}^{(t-1)} + \boldsymbol{b})$$

hidden at t | input at t | hidden at t-1

- **Temporal Unfolding**



Recurrent Neural Networks

- **Recurrent Neural Network**

$$\tilde{y}^{(t)} = \boldsymbol{w} \cdot \boldsymbol{h}^{(t)} + b$$

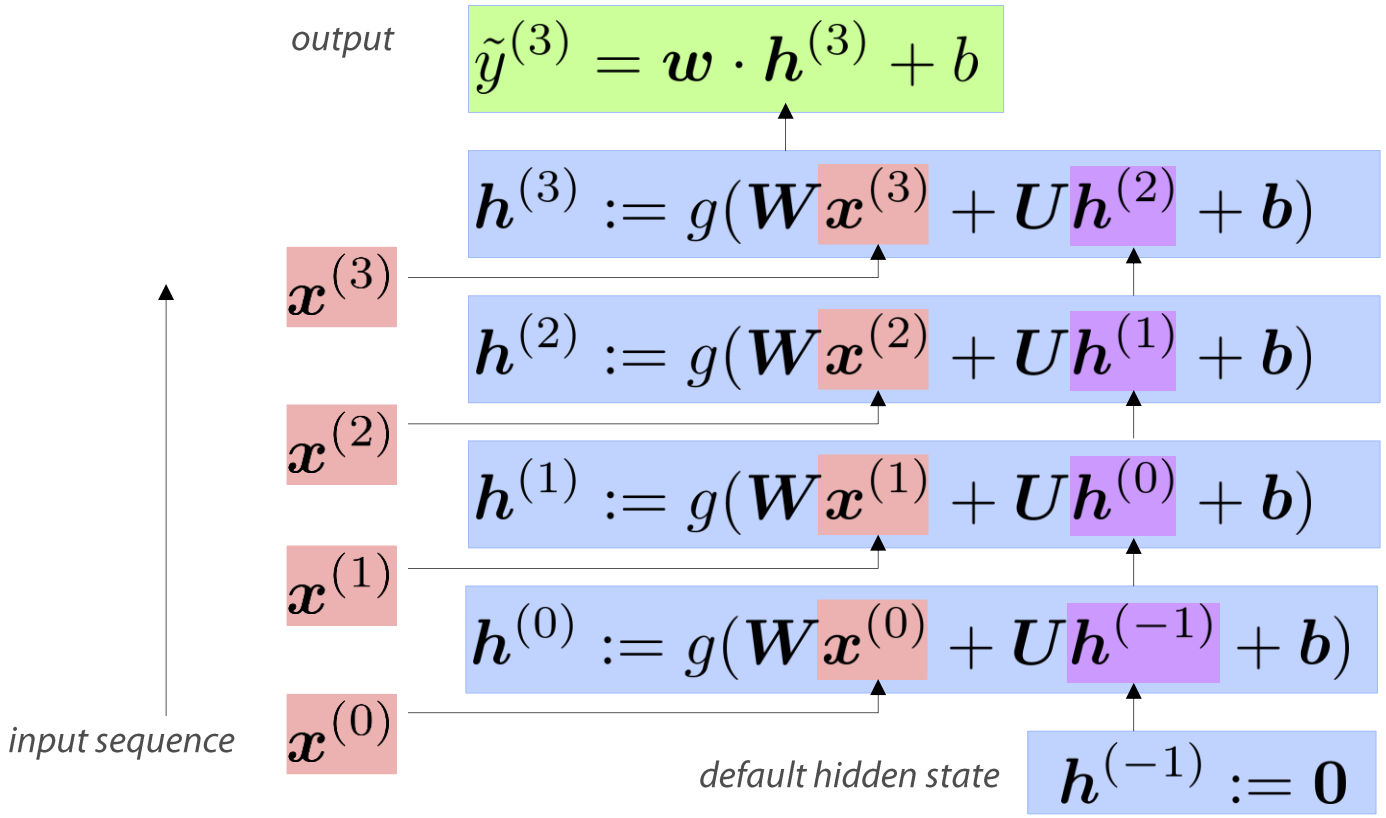
where:

$$\boldsymbol{h}^{(t)} := g(\boldsymbol{W}\boldsymbol{x}^{(t)} + \boldsymbol{U}\boldsymbol{h}^{(t-1)} + \boldsymbol{b})$$

hidden at t — *input at t* — *hidden at t-1*

- **Temporal Unfolding**

This looks very similar to a deep feed-forward neural network …

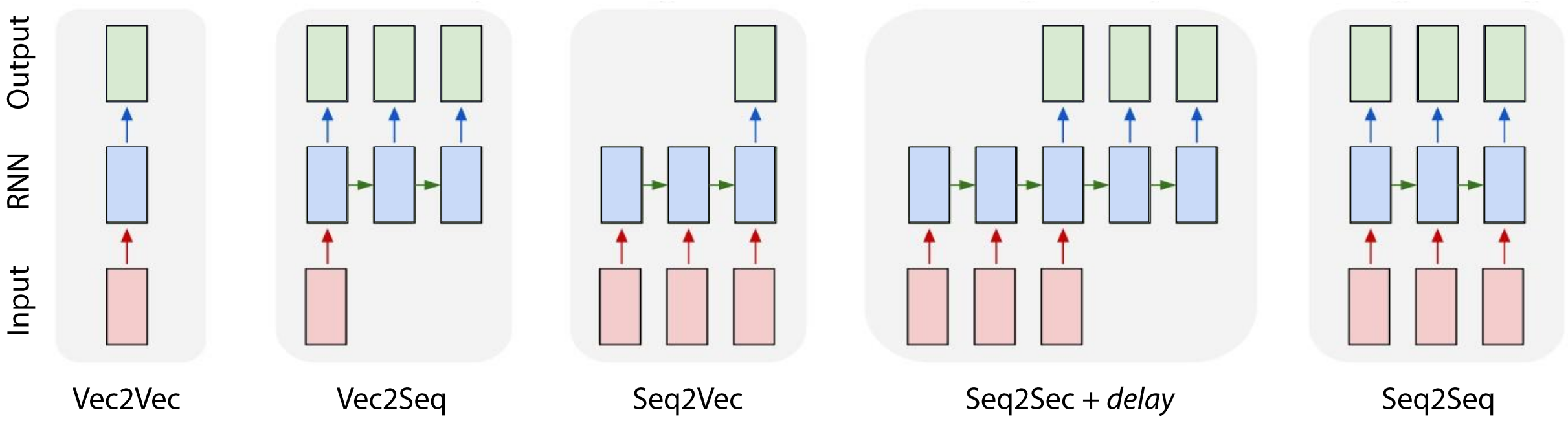


Recurrent Neural Networks

- **Recurrent Neural Network**

$$\tilde{y}^{(t)} = \boldsymbol{w} \cdot \boldsymbol{h}^{(t)} + b \qquad \textit{where:} \qquad \boldsymbol{h}^{(t)} := g(\boldsymbol{W}\boldsymbol{x}^{(t)} + \boldsymbol{U}\boldsymbol{h}^{(t-1)} + \boldsymbol{b})$$

- **Input-Output Modes**



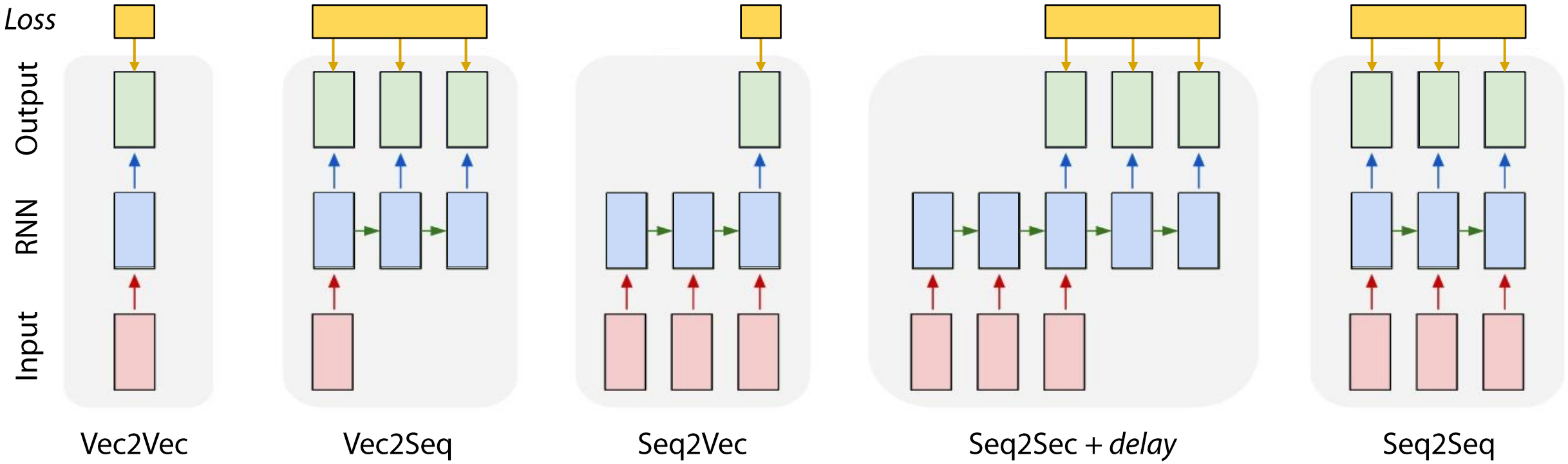
[image adapted from https://karpathy.github.io/2015/05/21/rnn-effectiveness/]

Recurrent Neural Networks

- **Recurrent Neural Network**

$$\tilde{y}^{(t)} = \boldsymbol{w} \cdot \boldsymbol{h}^{(t)} + b \qquad \textit{where:} \qquad \boldsymbol{h}^{(t)} := g(\boldsymbol{W}\boldsymbol{x}^{(t)} + \boldsymbol{U}\boldsymbol{h}^{(t-1)} + \boldsymbol{b})$$

- **Input-Output Modes**



[image adapted from https://karpathy.github.io/2015/05/21/rnn-effectiveness/]

Long-Short Term Memory (LSTM)

LSTM Cell

- ***Long-Short Term Memory*** (Hochreiter & Schmidhuber, 1995)

$$\tilde{y}^{(t)} = \boldsymbol{w} \cdot \boldsymbol{h}^{(t)} + b$$

elementwise product

$$\boldsymbol{h}^{(t)} := \boldsymbol{o}^{(t)} \odot \tanh(\boldsymbol{c}^{(t)})$$

$$\boldsymbol{c}^{(t)} := \boldsymbol{f}^{(t)} \odot \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \odot \boldsymbol{c}_{in}^{(t)}$$

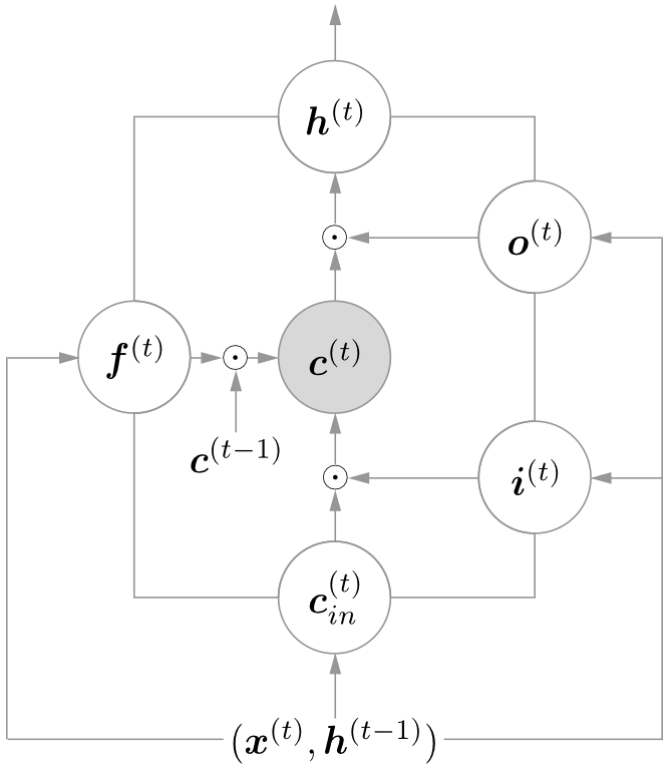
sigmoid function

$$\boldsymbol{o}^{(t)} := \sigma(\boldsymbol{W}_o \boldsymbol{x}^{(t)} + \boldsymbol{U}_o \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_o)$$

$$\boldsymbol{f}^{(t)} := \sigma(\boldsymbol{W}_f \boldsymbol{x}^{(t)} + \boldsymbol{U}_f \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_f)$$

$$\boldsymbol{i}^{(t)} := \sigma(\boldsymbol{W}_i \boldsymbol{x}^{(t)} + \boldsymbol{U}_i \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_i)$$

$$\boldsymbol{c}_{in}^{(t)} := \tanh(\boldsymbol{W}_c \boldsymbol{x}^{(t)} + \boldsymbol{U}_c \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_c)$$



LSTM Cell

- ***Long-Short Term Memory*** (Hochreiter & Schmidhuber, 1995)

$$\tilde{y}^{(t)} = \boldsymbol{w} \cdot \boldsymbol{h}^{(t)} + b$$

$$\boldsymbol{h}^{(t)} := \boldsymbol{o}^{(t)} \odot \tanh(\boldsymbol{c}^{(t)})$$

$$\boldsymbol{c}^{(t)} := \boldsymbol{f}^{(t)} \odot \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \odot \boldsymbol{c}_{in}^{(t)}$$

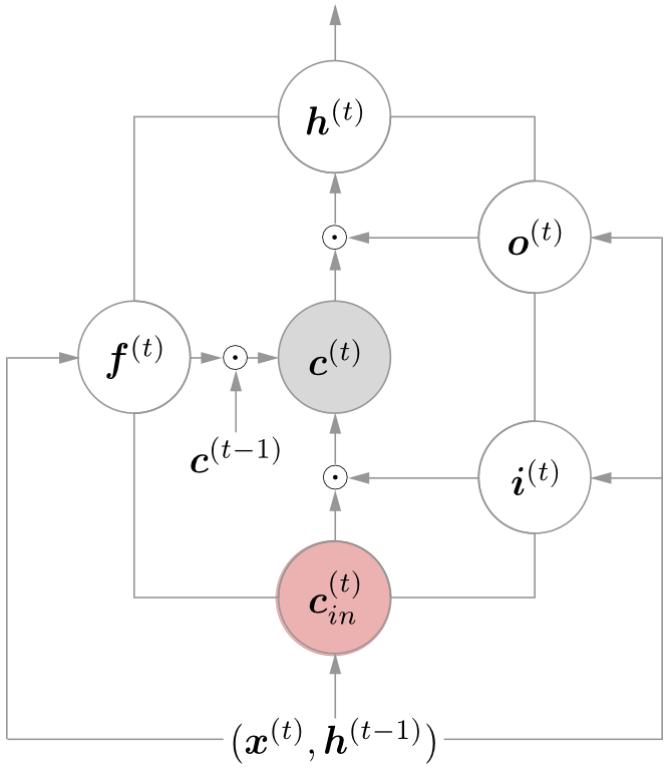
$$\boldsymbol{o}^{(t)} := \sigma(\boldsymbol{W}_o \boldsymbol{x}^{(t)} + \boldsymbol{U}_o \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_o)$$

$$\boldsymbol{f}^{(t)} := \sigma(\boldsymbol{W}_f \boldsymbol{x}^{(t)} + \boldsymbol{U}_f \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_f)$$

$$\boldsymbol{i}^{(t)} := \sigma(\boldsymbol{W}_i \boldsymbol{x}^{(t)} + \boldsymbol{U}_i \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_i)$$

Combined input

$$\boldsymbol{c}_{in}^{(t)} := \tanh(\boldsymbol{W}_c \boldsymbol{x}^{(t)} + \boldsymbol{U}_c \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_c)$$



LSTM Cell

- ***Long-Short Term Memory*** (Hochreiter & Schmidhuber, 1995)

$$\tilde{y}^{(t)} = \boldsymbol{w} \cdot \boldsymbol{h}^{(t)} + b$$

$$\boldsymbol{h}^{(t)} := \boldsymbol{o}^{(t)} \odot \tanh(\boldsymbol{c}^{(t)})$$

$$\boldsymbol{c}^{(t)} := \boldsymbol{f}^{(t)} \odot \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \odot \boldsymbol{c}_{in}^{(t)}$$

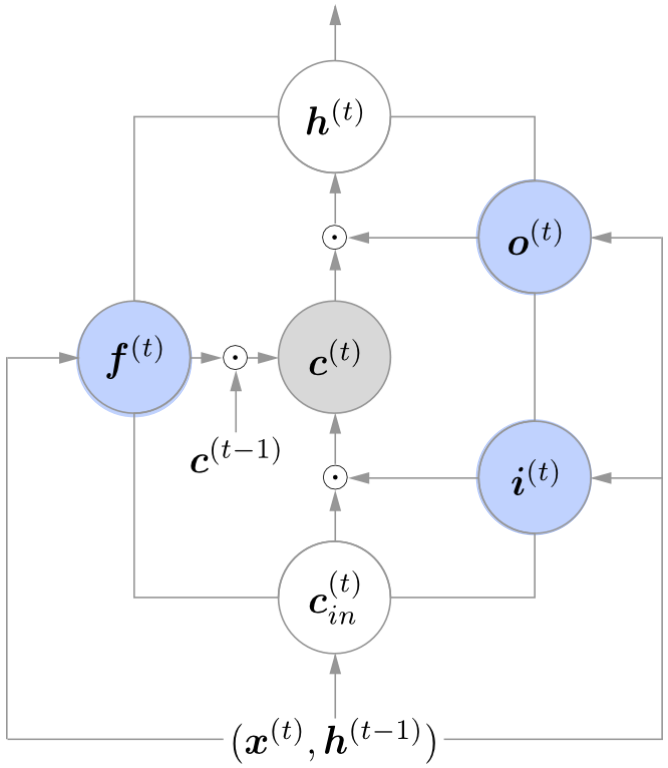
Gating values

$$\boldsymbol{o}^{(t)} := \sigma(\boldsymbol{W}_o \boldsymbol{x}^{(t)} + \boldsymbol{U}_o \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_o) \quad \textit{output}$$

$$\boldsymbol{f}^{(t)} := \sigma(\boldsymbol{W}_f \boldsymbol{x}^{(t)} + \boldsymbol{U}_f \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_f) \quad \textit{forget}$$

$$\boldsymbol{i}^{(t)} := \sigma(\boldsymbol{W}_i \boldsymbol{x}^{(t)} + \boldsymbol{U}_i \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_i) \quad \textit{input}$$

$$\boldsymbol{c}_{in}^{(t)} := \tanh(\boldsymbol{W}_c \boldsymbol{x}^{(t)} + \boldsymbol{U}_c \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_c)$$



LSTM Cell

- ***Long-Short Term Memory*** (Hochreiter & Schmidhuber, 1995)

$$\tilde{y}^{(t)} = \boldsymbol{w} \cdot \boldsymbol{h}^{(t)} + b$$

Applying gates

$$\boldsymbol{h}^{(t)} := \boldsymbol{o}^{(t)} \odot \tanh(\boldsymbol{c}^{(t)})$$ *hidden*

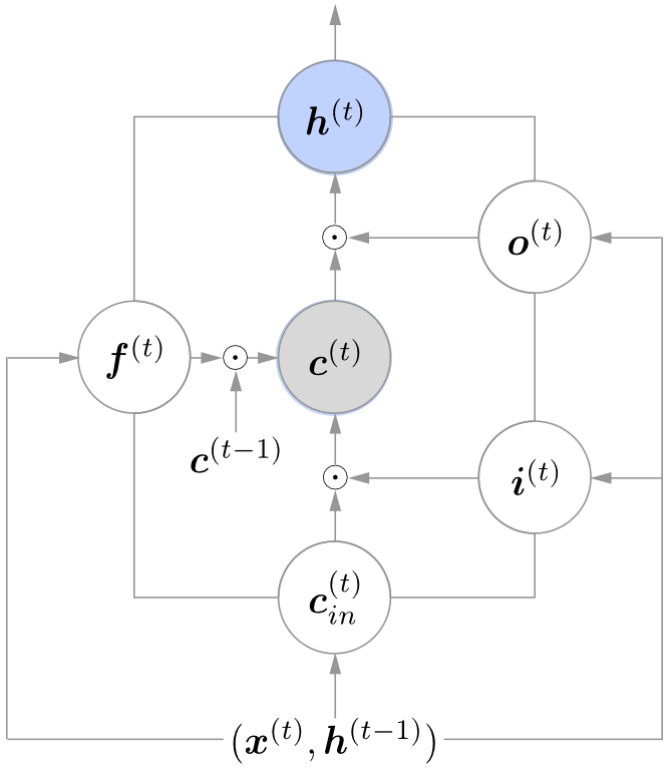
$$\boldsymbol{c}^{(t)} := \boldsymbol{f}^{(t)} \odot \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \odot \boldsymbol{c}_{in}^{(t)}$$ *memory*

$$\boldsymbol{o}^{(t)} := \sigma(\boldsymbol{W}_o \boldsymbol{x}^{(t)} + \boldsymbol{U}_o \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_o)$$

$$\boldsymbol{f}^{(t)} := \sigma(\boldsymbol{W}_f \boldsymbol{x}^{(t)} + \boldsymbol{U}_f \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_f)$$

$$\boldsymbol{i}^{(t)} := \sigma(\boldsymbol{W}_i \boldsymbol{x}^{(t)} + \boldsymbol{U}_i \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_i)$$

$$\boldsymbol{c}_{in}^{(t)} := \tanh(\boldsymbol{W}_c \boldsymbol{x}^{(t)} + \boldsymbol{U}_c \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_c)$$



LSTM Cell

- ***Long-Short Term Memory*** (Hochreiter & Schmidhuber, 1995)

$$\tilde{y}^{(t)} = \boldsymbol{w} \cdot \boldsymbol{h}^{(t)} + b$$ *Cell output*

$$\boldsymbol{h}^{(t)} := \boldsymbol{o}^{(t)} \odot \tanh(\boldsymbol{c}^{(t)})$$

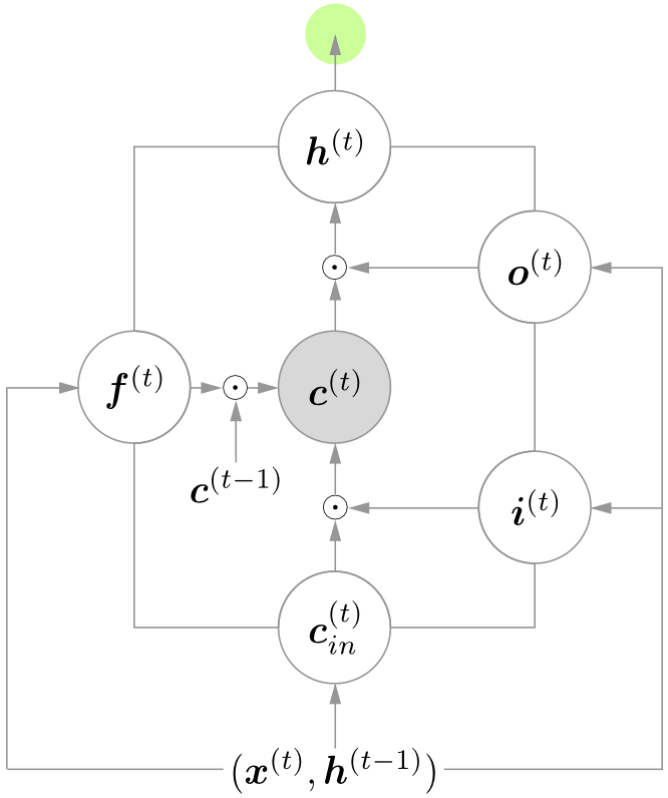
$$\boldsymbol{c}^{(t)} := \boldsymbol{f}^{(t)} \odot \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \odot \boldsymbol{c}_{in}^{(t)}$$

$$\boldsymbol{o}^{(t)} := \sigma(\boldsymbol{W}_o \boldsymbol{x}^{(t)} + \boldsymbol{U}_o \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_o)$$

$$\boldsymbol{f}^{(t)} := \sigma(\boldsymbol{W}_f \boldsymbol{x}^{(t)} + \boldsymbol{U}_f \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_f)$$

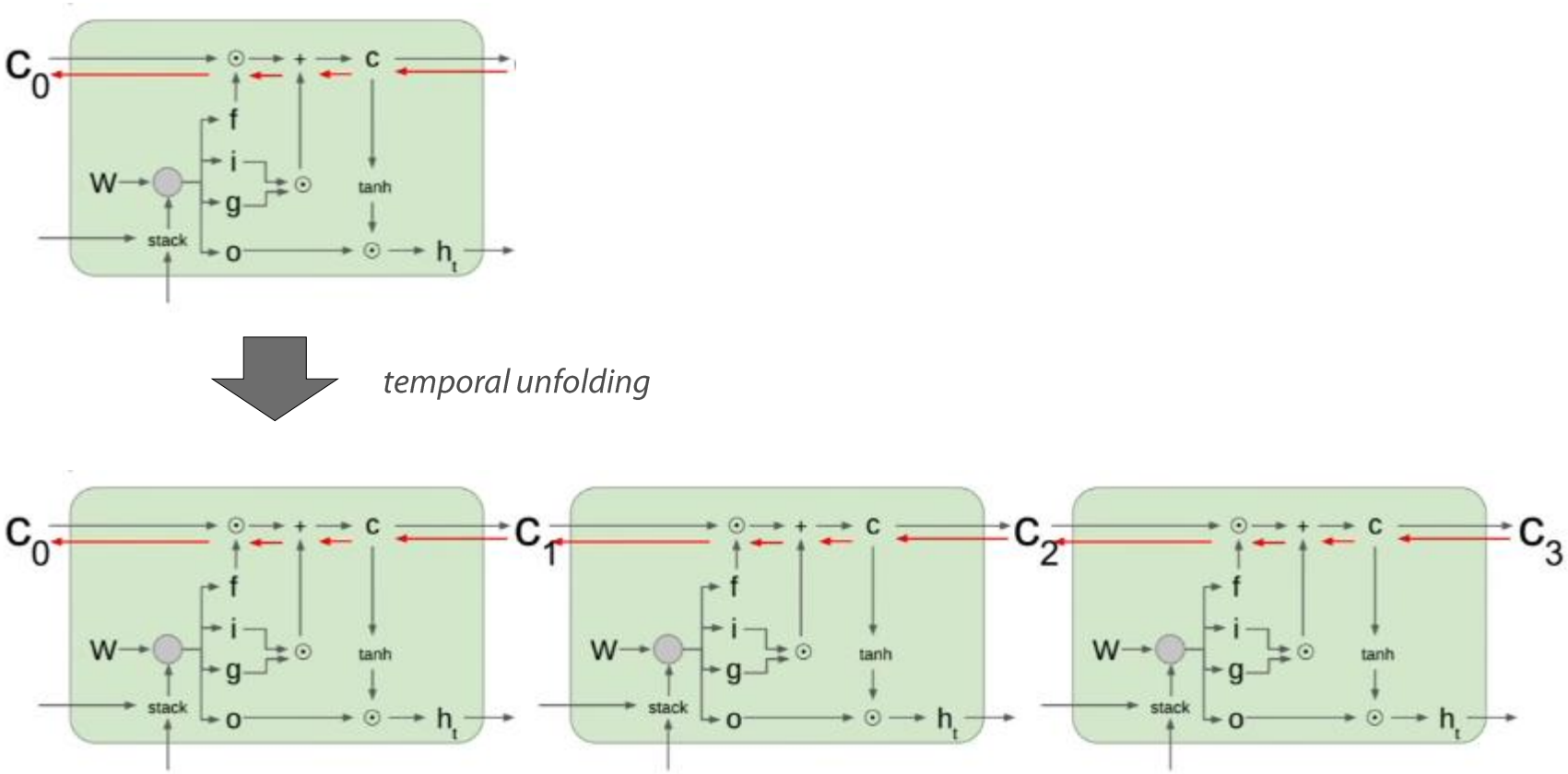
$$\boldsymbol{i}^{(t)} := \sigma(\boldsymbol{W}_i \boldsymbol{x}^{(t)} + \boldsymbol{U}_i \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_i)$$

$$\boldsymbol{c}_{in}^{(t)} := \tanh(\boldsymbol{W}_c \boldsymbol{x}^{(t)} + \boldsymbol{U}_c \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_c)$$



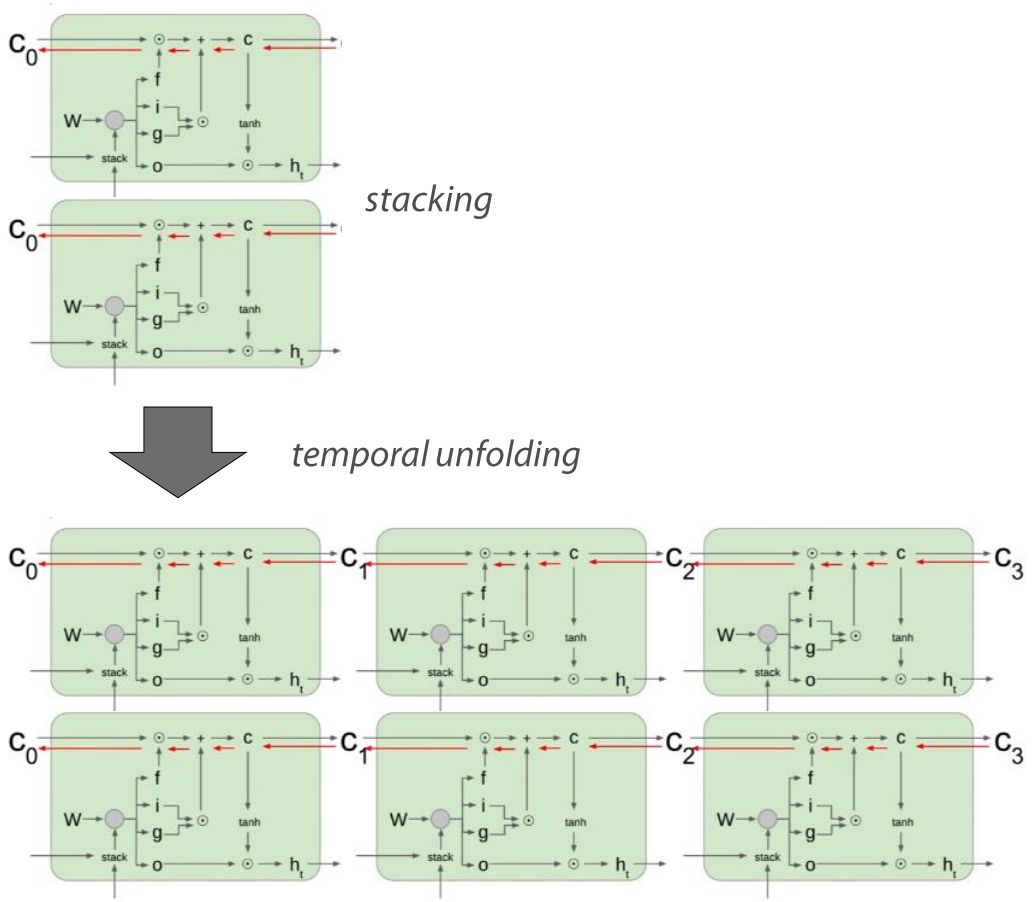
LSTM Training

- **Temporal Unfolding**



LSTM Training

- ***Stacking and Temporal Unfolding***



GRU

- ***Gated Recurrent Unit*** (Kyunghyun Cho et al., 1995)

Simpler structure, no internal memory

$$\tilde{y}^{(t)} = \boldsymbol{w} \cdot \boldsymbol{h}^{(t)} + b$$

(exponential moving average)

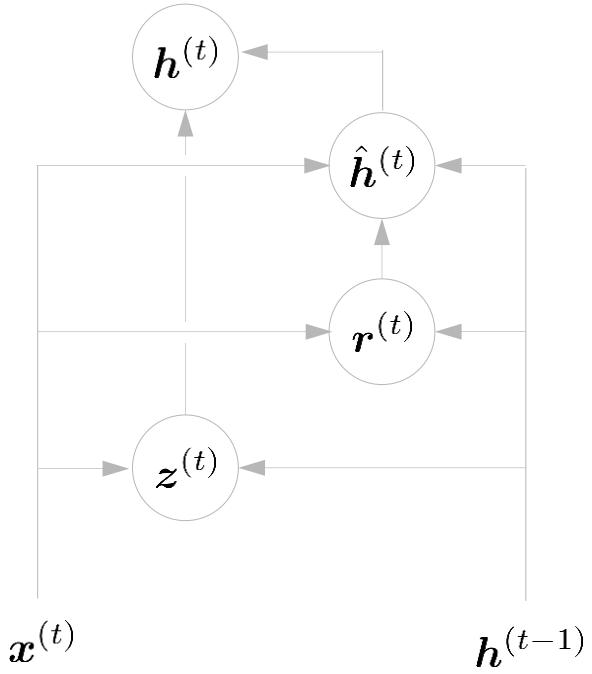
$$\boldsymbol{h}^{(t)} := (1 - \boldsymbol{z}^{(t)}) \odot \boldsymbol{h}^{(t-1)} + \boldsymbol{z}^{(t)} \odot \hat{\boldsymbol{h}}^{(t)}$$

$$\hat{\boldsymbol{h}}^{(t)} := \tanh(\boldsymbol{W}_h \boldsymbol{x}^{(t)} + \boldsymbol{U}_h(\boldsymbol{r}^{(t)} \odot \boldsymbol{h}^{(t-1)}) + \boldsymbol{b}_h)$$

sigmoid function

$$\boldsymbol{r}^{(t)} := \sigma(\boldsymbol{W}_r \boldsymbol{x}^{(t)} + \boldsymbol{U}_r \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_r)$$

$$\boldsymbol{z}^{(t)} := \sigma(\boldsymbol{W}_z \boldsymbol{x}^{(t)} + \boldsymbol{U}_z \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_z)$$



GRU

- ***Gated Recurrent Unit*** (Kyunghyun Cho et al., 1995)

Simpler structure, no internal memory

$$\tilde{y}^{(t)} = \boldsymbol{w} \cdot \boldsymbol{h}^{(t)} + b$$

(exponential moving average)

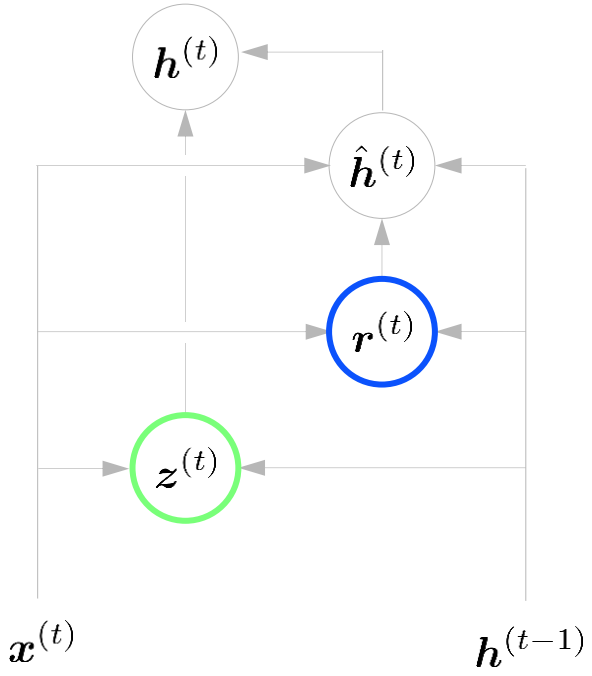
$$\boldsymbol{h}^{(t)} := (1 - \boldsymbol{z}^{(t)}) \odot \boldsymbol{h}^{(t-1)} + \boldsymbol{z}^{(t)} \odot \hat{\boldsymbol{h}}^{(t)}$$

$$\hat{\boldsymbol{h}}^{(t)} := \tanh(\boldsymbol{W}_h \boldsymbol{x}^{(t)} + \boldsymbol{U}_h(\boldsymbol{r}^{(t)} \odot \boldsymbol{h}^{(t-1)}) + \boldsymbol{b}_h)$$

sigmoid function — *Gating values*

$$\boldsymbol{r}^{(t)} := \sigma(\boldsymbol{W}_r \boldsymbol{x}^{(t)} + \boldsymbol{U}_r \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_r)$$ *reset*

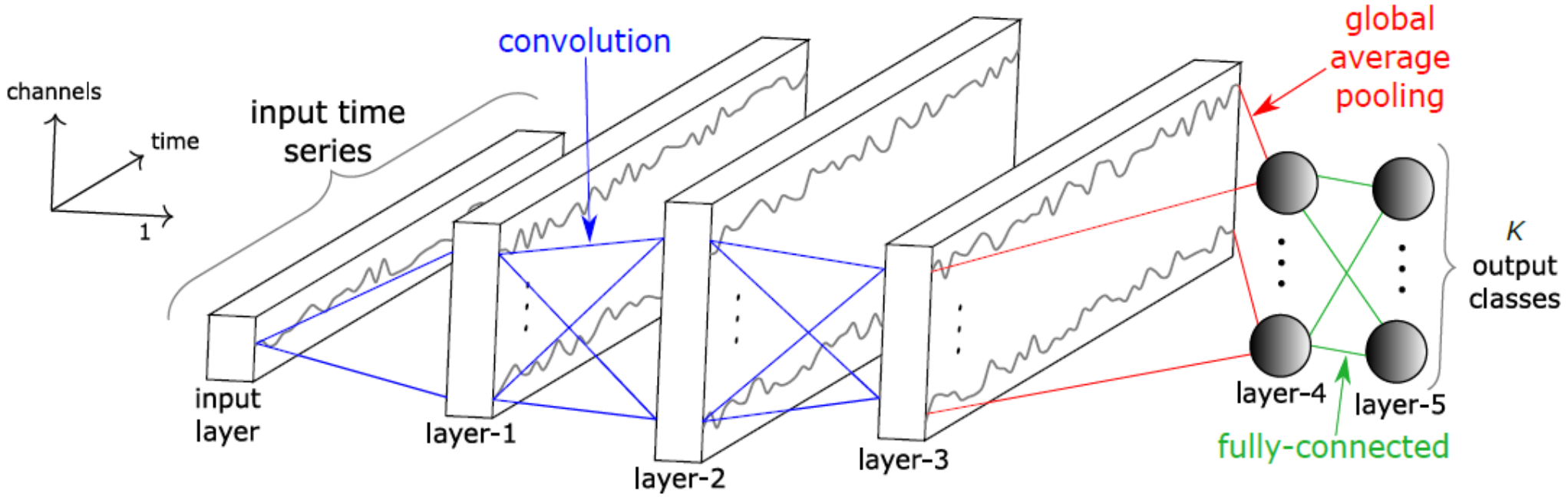
$$\boldsymbol{z}^{(t)} := \sigma(\boldsymbol{W}_z \boldsymbol{x}^{(t)} + \boldsymbol{U}_z \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_z)$$ *update*



Deep Convolutional Networks for Time Series Analysis

1D Convolution Over Time

Time windows are treated as 1D 'images'

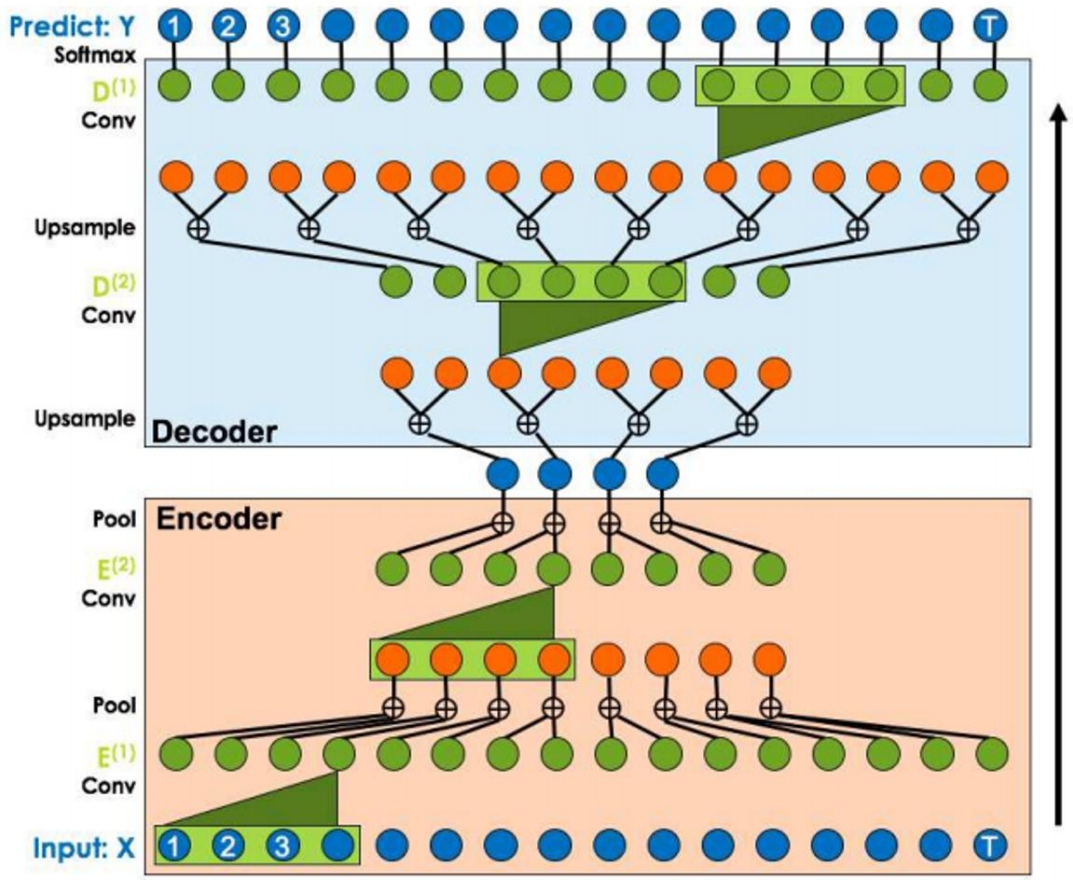


[image from https://link.springer.com/article/10.1007%2Fs10618-019-00619-1]

Temporal Convolution Networks

1D Convolution-Deconvolution in an autoencoder architecture (Lea et al., 2016)

Effective in segmenting actions and predicting time series

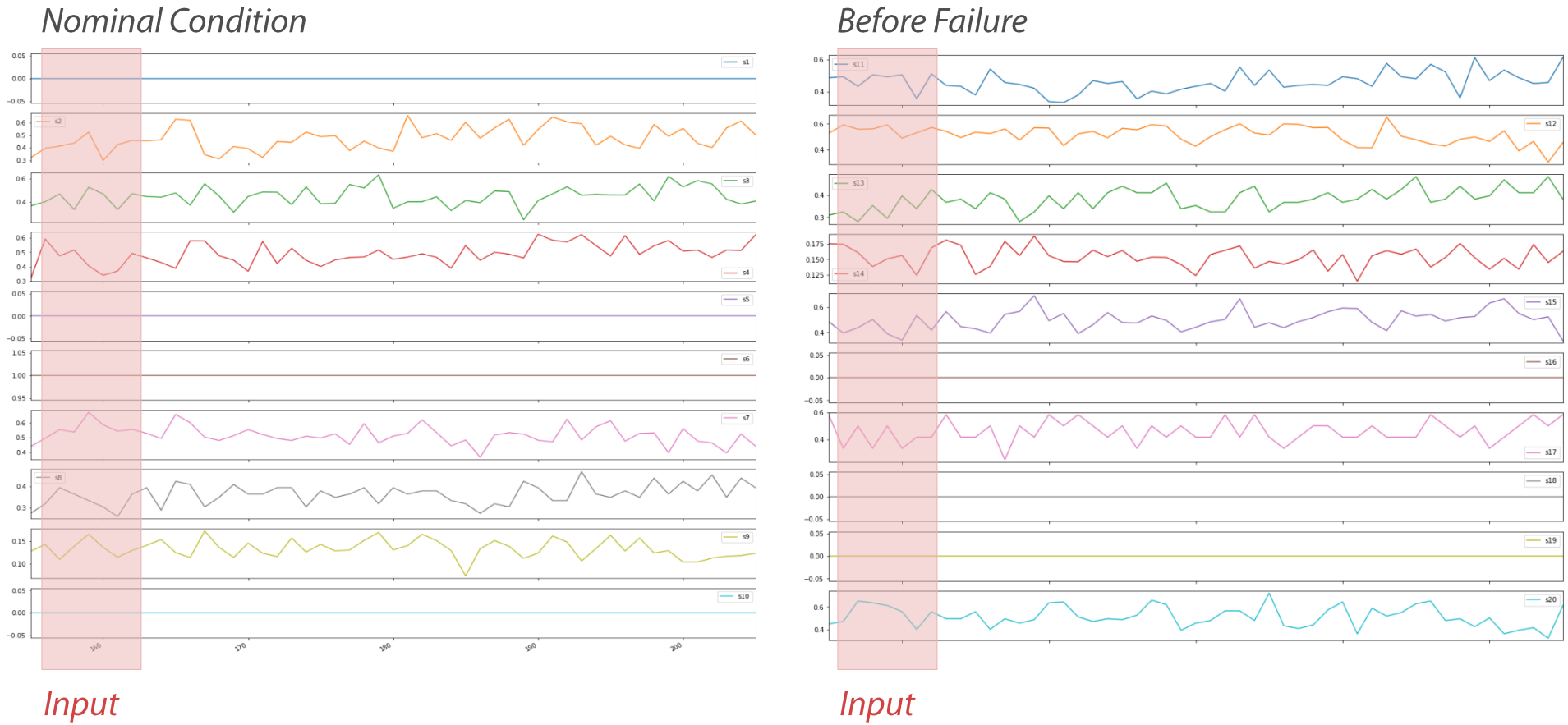


[image from https://arxiv.org/abs/1611.05267]

RNN applications

Predictive Maintenance

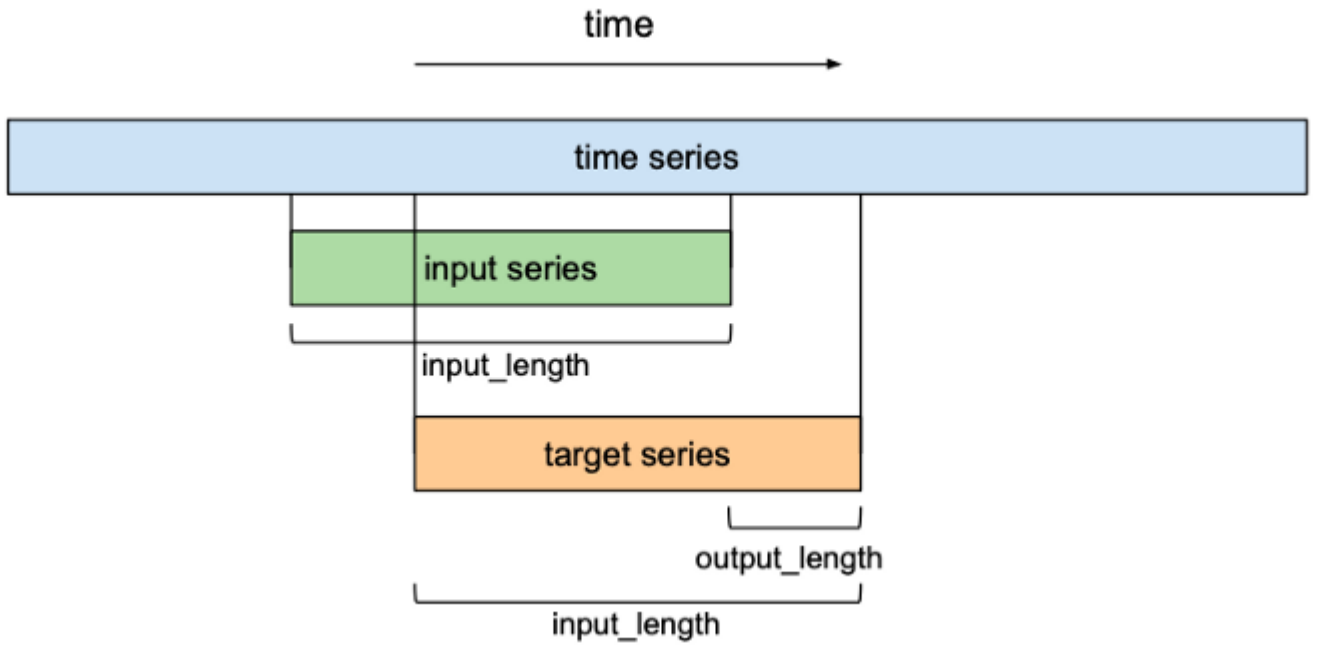
- **Detecting failure conditions from sensor readings**



Training and Prediction occurs by using a sliding window of sensor readings as input

Time Series Forecasting

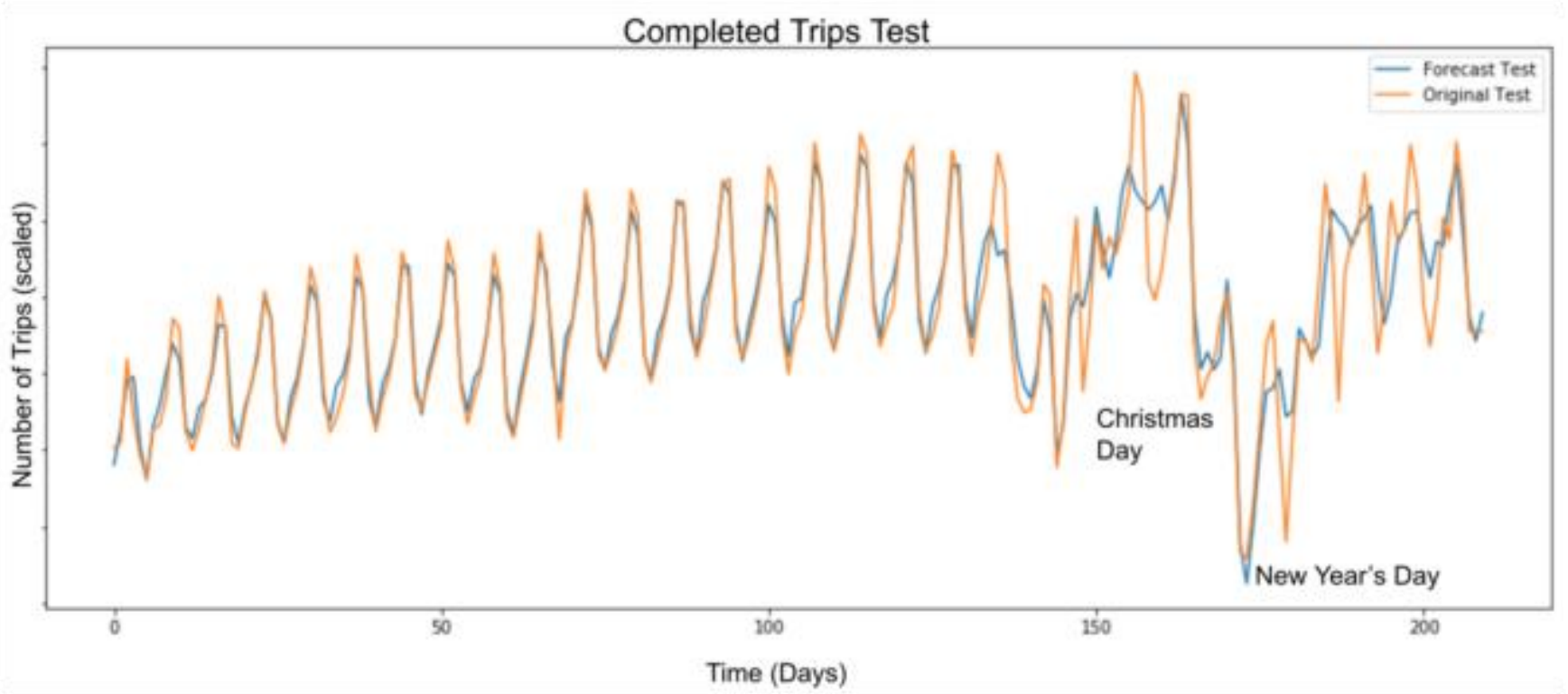
Forecasting time series ahead of time



[image from https://medium.com/unit8-machine-learning-publication/temporal-convolutional-networks-and-forecasting-5ce1b6e97ce4]

Time Series Forecasting

Detecting anomalies as differences from forecasted and actual



[image from https://eng.uber.com/neural-networks/]