Deep Learning

A course about theory & practice

UNIVERSITÀ DI PAVIA

Recurrent Neural Networks

Marco Piastra

The Basic Architecture

Recurrent Neural Networks

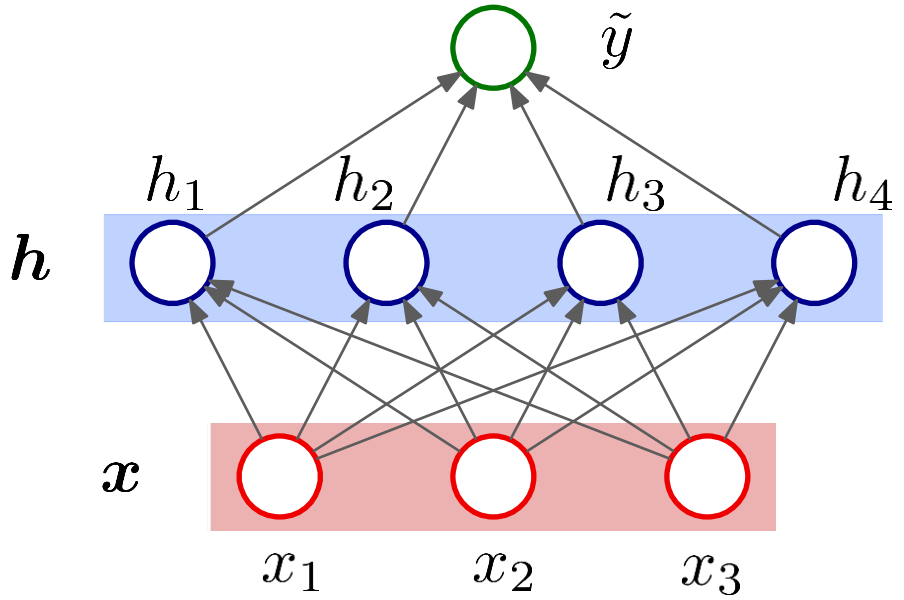
- **Feed-forward neural network**

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

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:

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

$\boldsymbol{h}$: hidden; $\boldsymbol{x}$: input

- **Recurrent Neural Network**

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

where:

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

$\boldsymbol{h}^{(t)}$: hidden at t; $\boldsymbol{x}^{(t)}$: input at t; $\boldsymbol{h}^{(t-1)}$: hidden at t-1

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

Recurrent Neural Networks

- **Recurrent Neural Network**

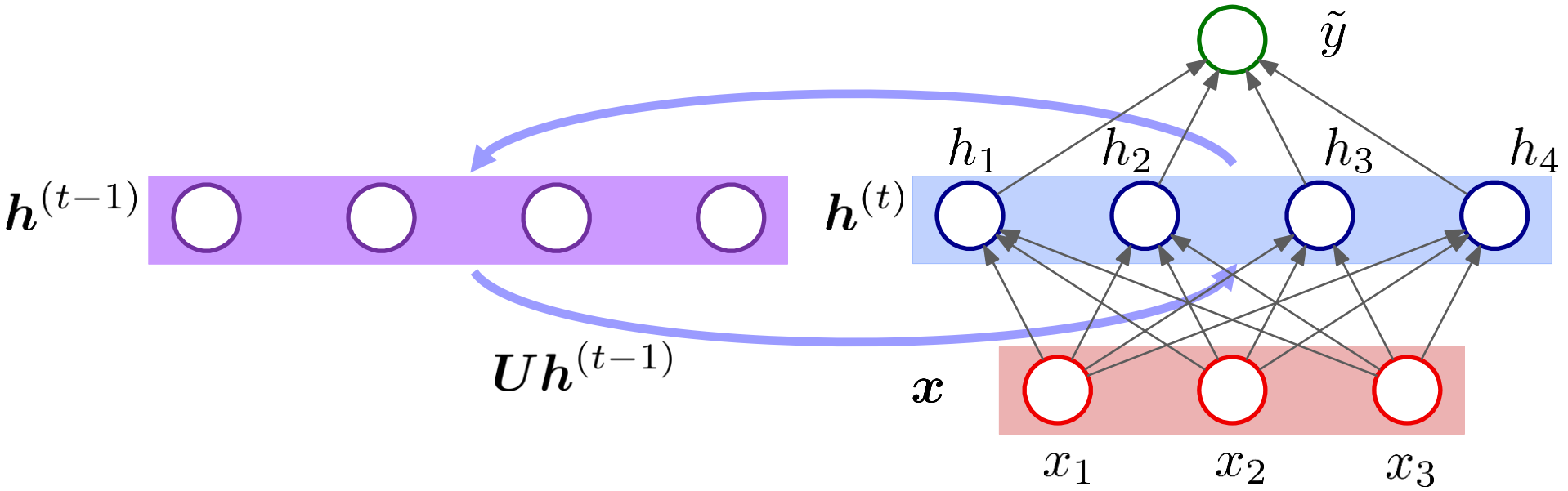
$$\tilde{y}^{(t)} = \boldsymbol{w} \cdot \boldsymbol{h}^{(t)} + b$$

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 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*

Recurrent Neural Networks

- **Recurrent Neural Network**

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

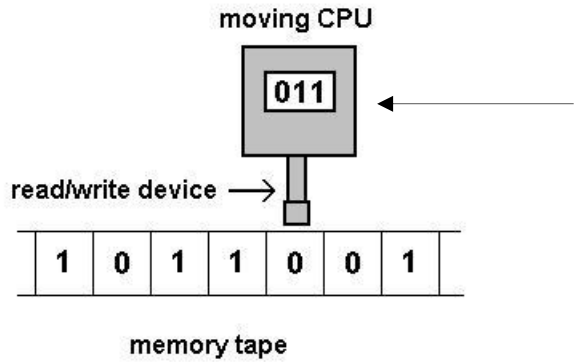
where:

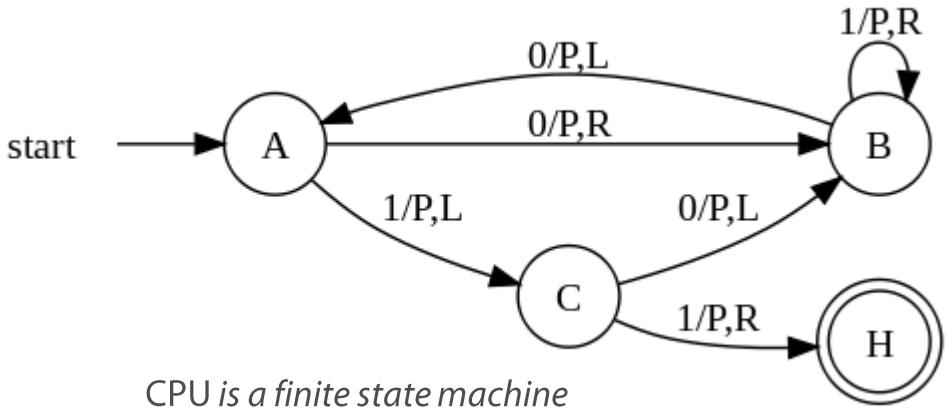
$$\boldsymbol{h}^{(t)} := g(\boldsymbol{W}\boldsymbol{x}^{(t)} + \boldsymbol{U}\boldsymbol{h}^{(t-1)} + \boldsymbol{b})$$

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

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

"RNNs can simulate any Turing machine"





CPU *is a finite state machine*

They are *much harder* to **train** than FF neural networks

There is no known general method to do this

Recurrent Neural Networks

- **Recurrent Neural Network**

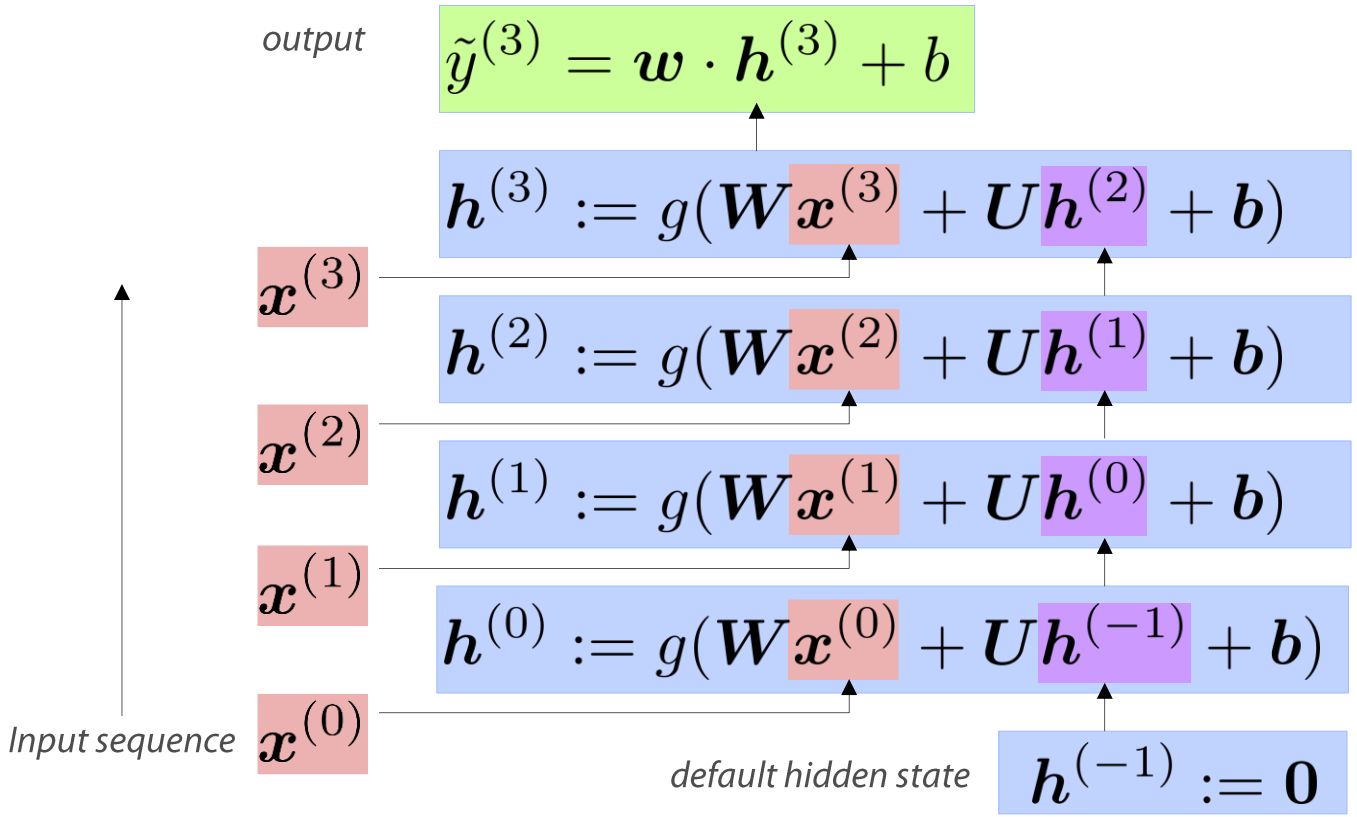
$$\tilde{y}^{(t)} = \boldsymbol{w} \cdot \boldsymbol{h}^{(t)} + b$$

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**

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

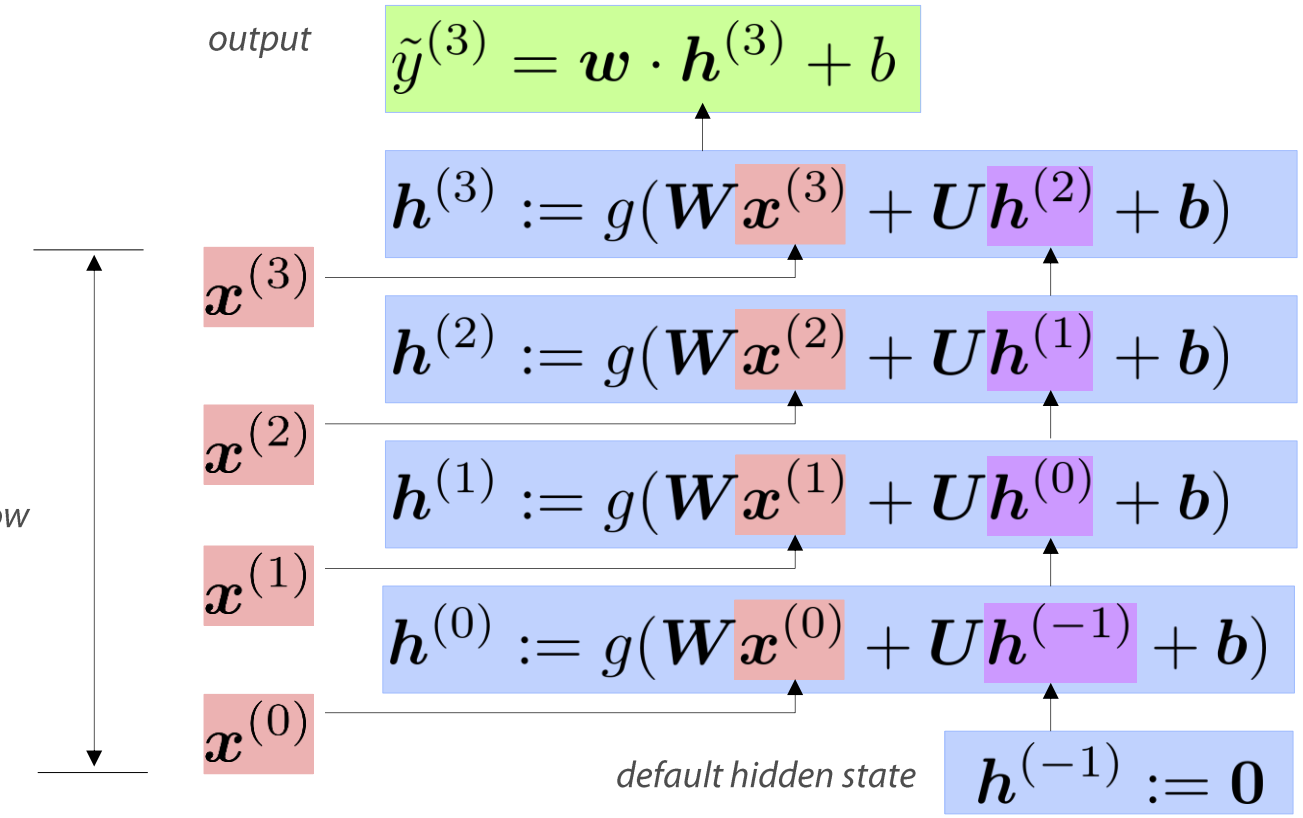
where:

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

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

- **Temporal Unfolding**

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

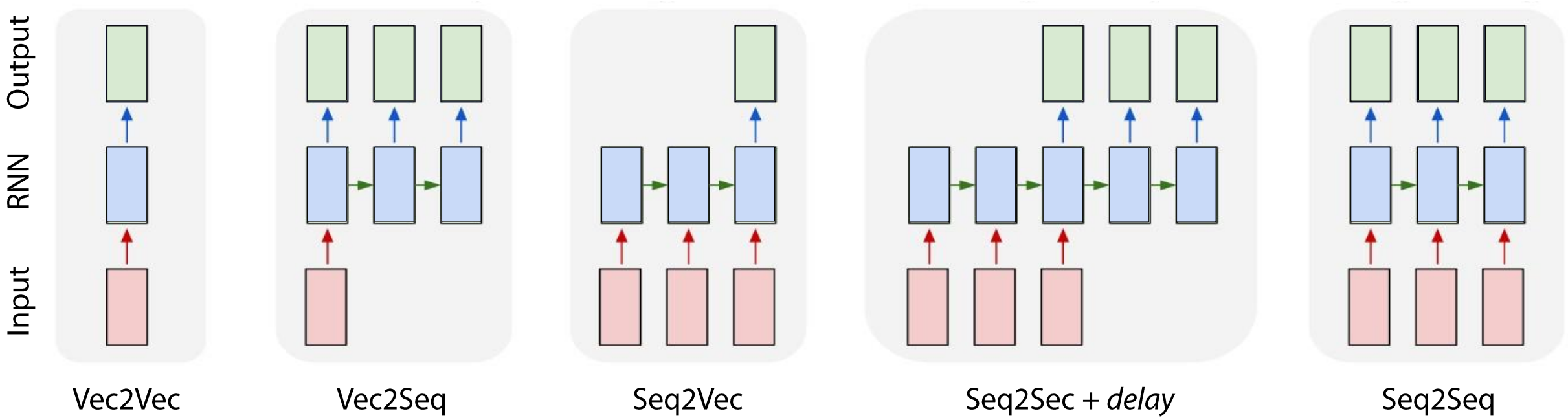


Recurrent Neural Networks

- **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

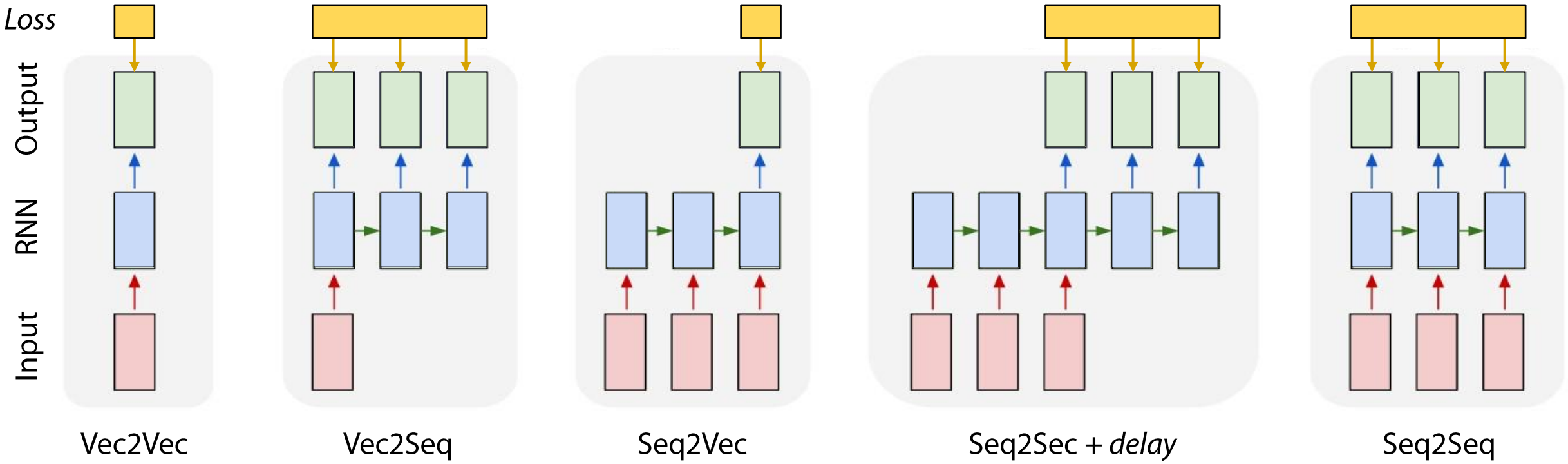
- **Recurrent Neural Network**

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

where:

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

- **Input-Output Modes**

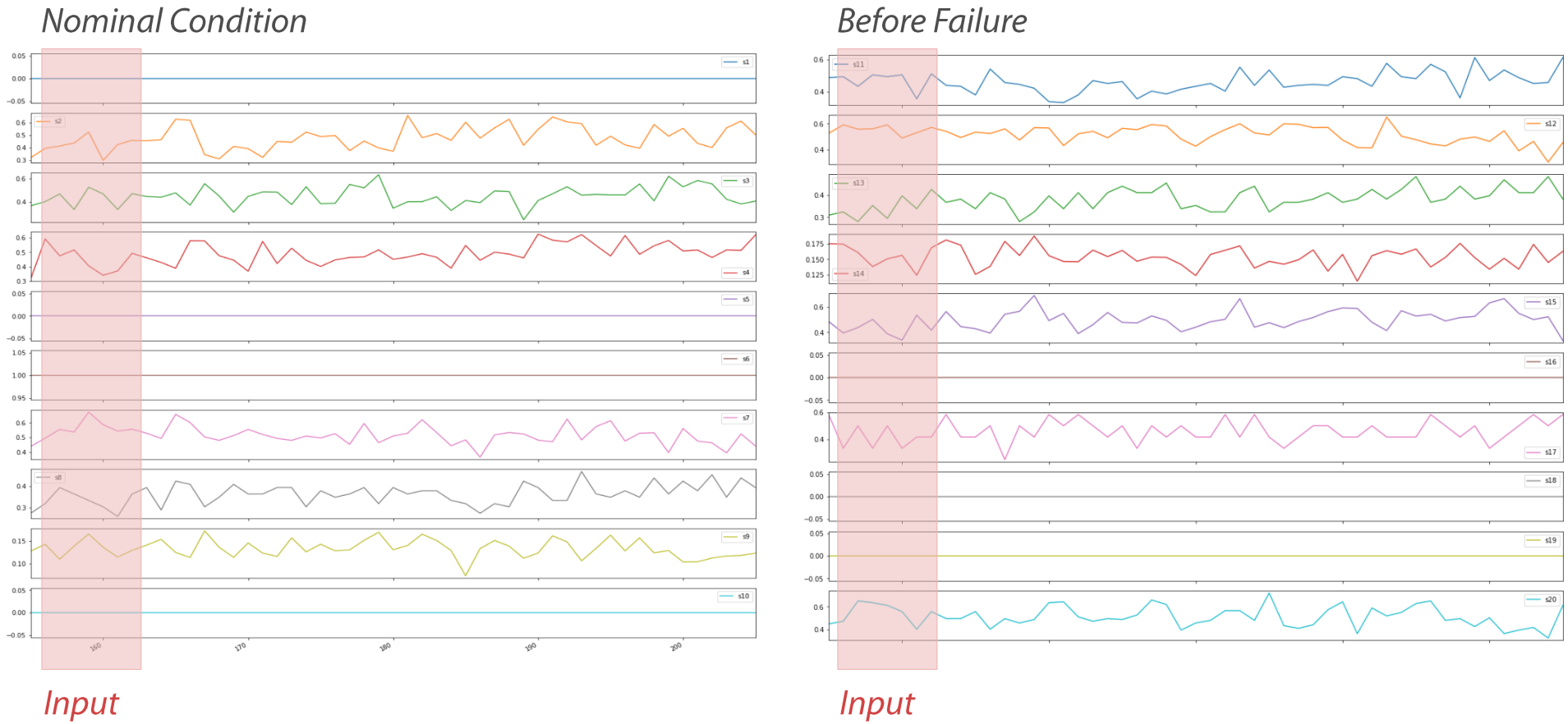


[image adapted from https://karpathy.github.io/2015/05/21/rnn-effectiveness/]

RNN Applications

Predictive Maintenance

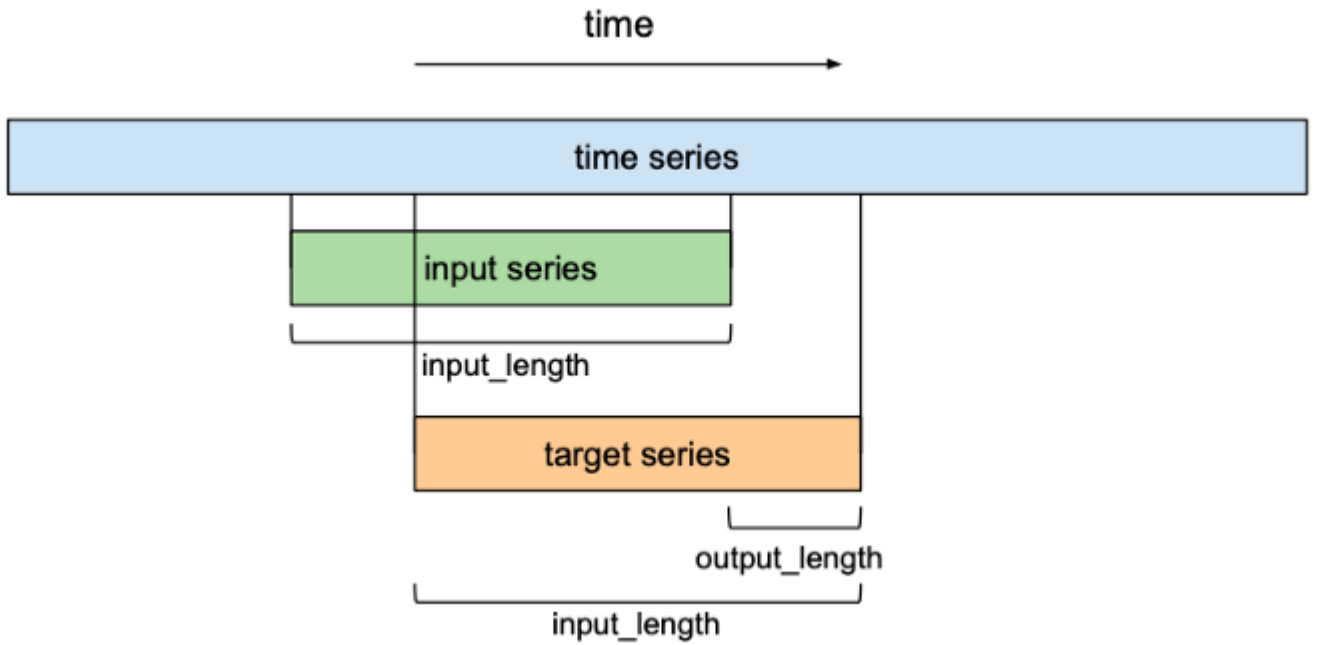
- **Detecting failure conditions from sensor readings**



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

Time Series Forecasting

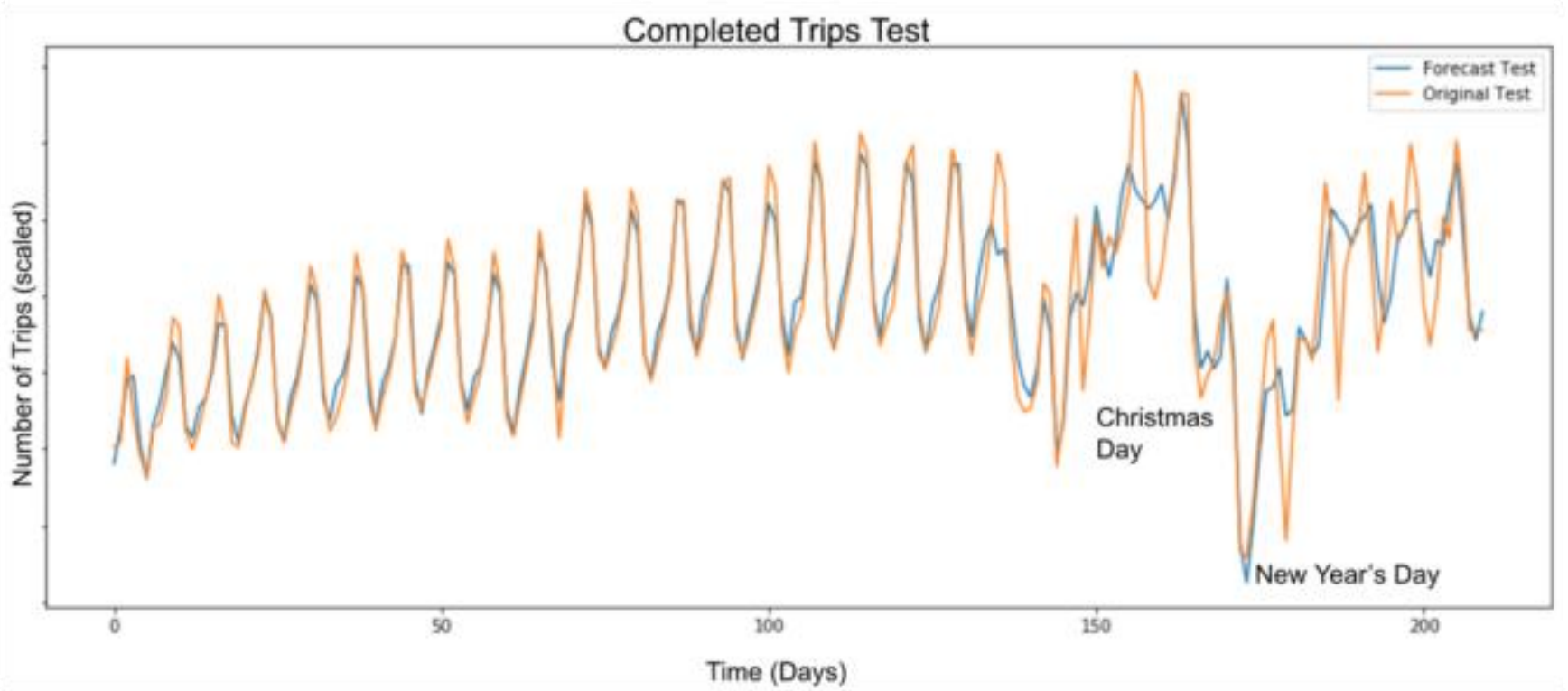
Forecasting time series ahead of time



[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/]

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

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

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

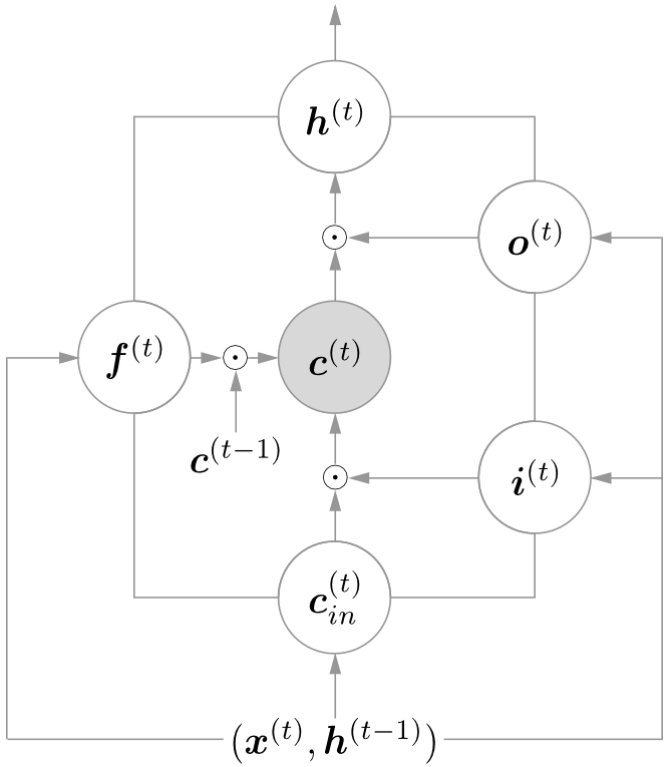
sigmoid function

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

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

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

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



LSTM Cell

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

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

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

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

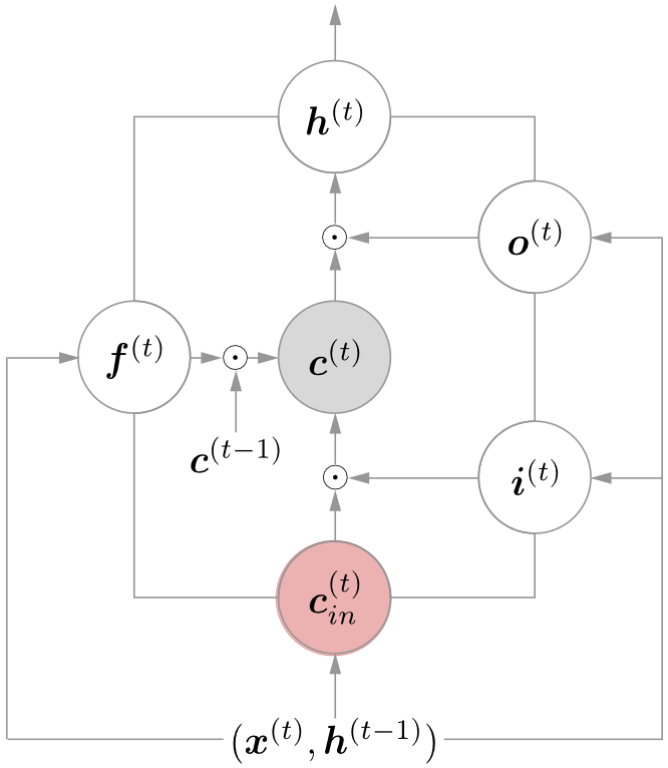
$$\boldsymbol{o}^{(t)} := \sigma(\boldsymbol{W}_o \boldsymbol{x}^{(t)} + \boldsymbol{U}_o \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_o)$$

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

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

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)

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

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

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

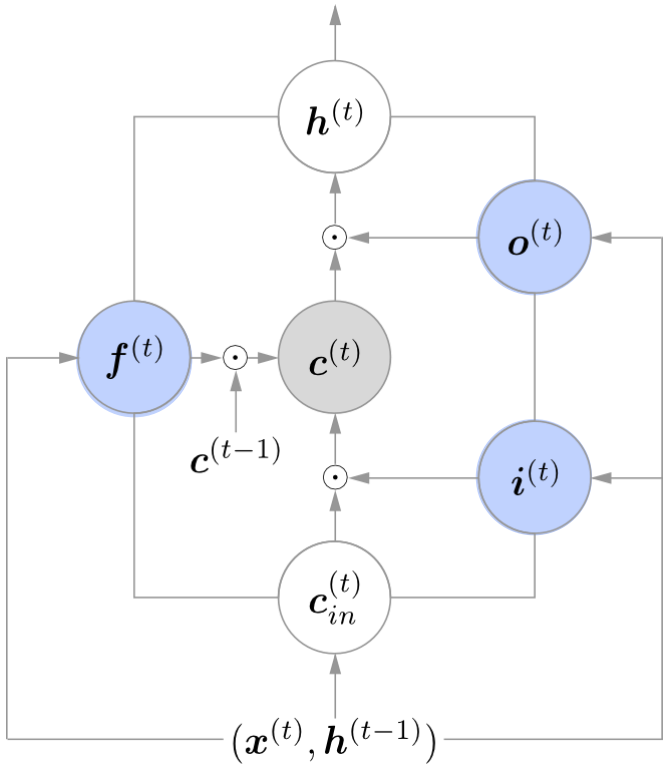
Gating values

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

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

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

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



LSTM Cell

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

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

Applying gates

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

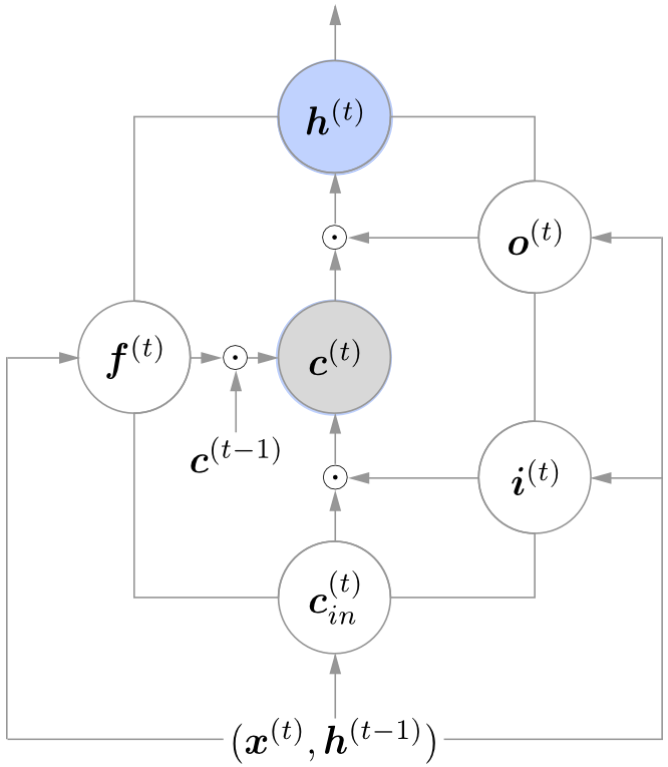
$$\boldsymbol{c}^{(t)} := \boldsymbol{f}^{(t)} \odot \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \odot \boldsymbol{c}_{in}^{(t)} \quad \textit{memory}$$

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

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

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

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



LSTM Cell

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

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

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

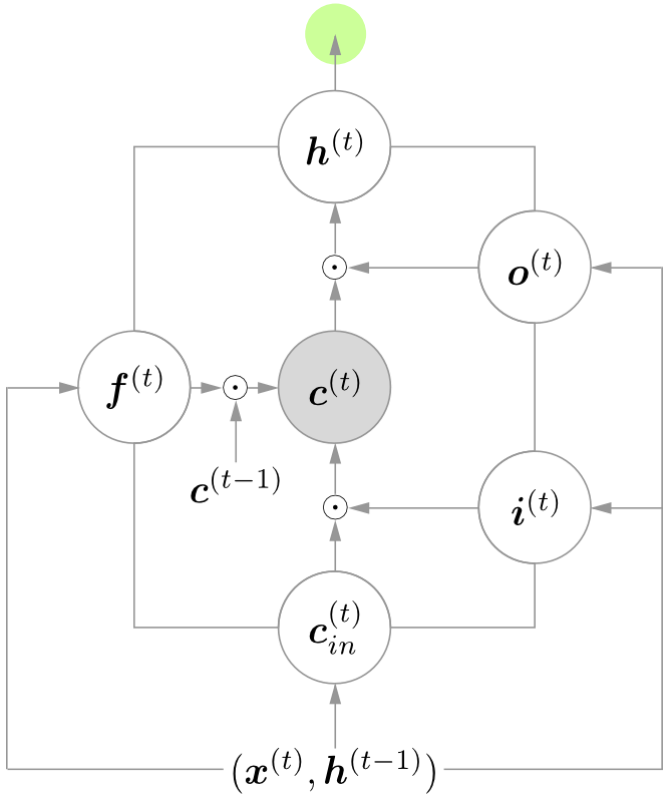
$$\boldsymbol{c}^{(t)} := \boldsymbol{f}^{(t)} \odot \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \odot \boldsymbol{c}_{in}^{(t)}$$

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

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

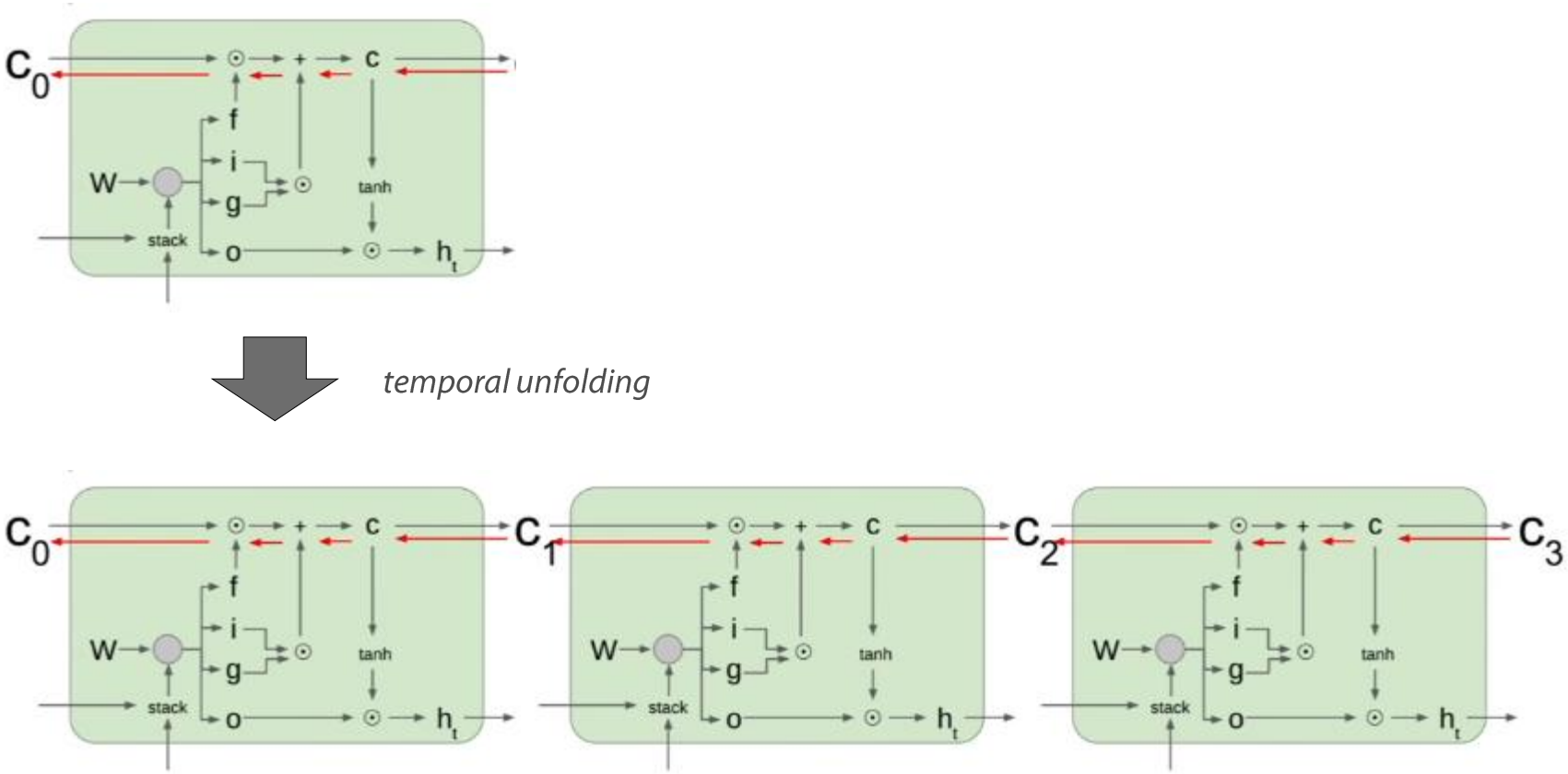
$$\boldsymbol{i}^{(t)} := \sigma(\boldsymbol{W}_i \boldsymbol{x}^{(t)} + \boldsymbol{U}_i \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_i)$$

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



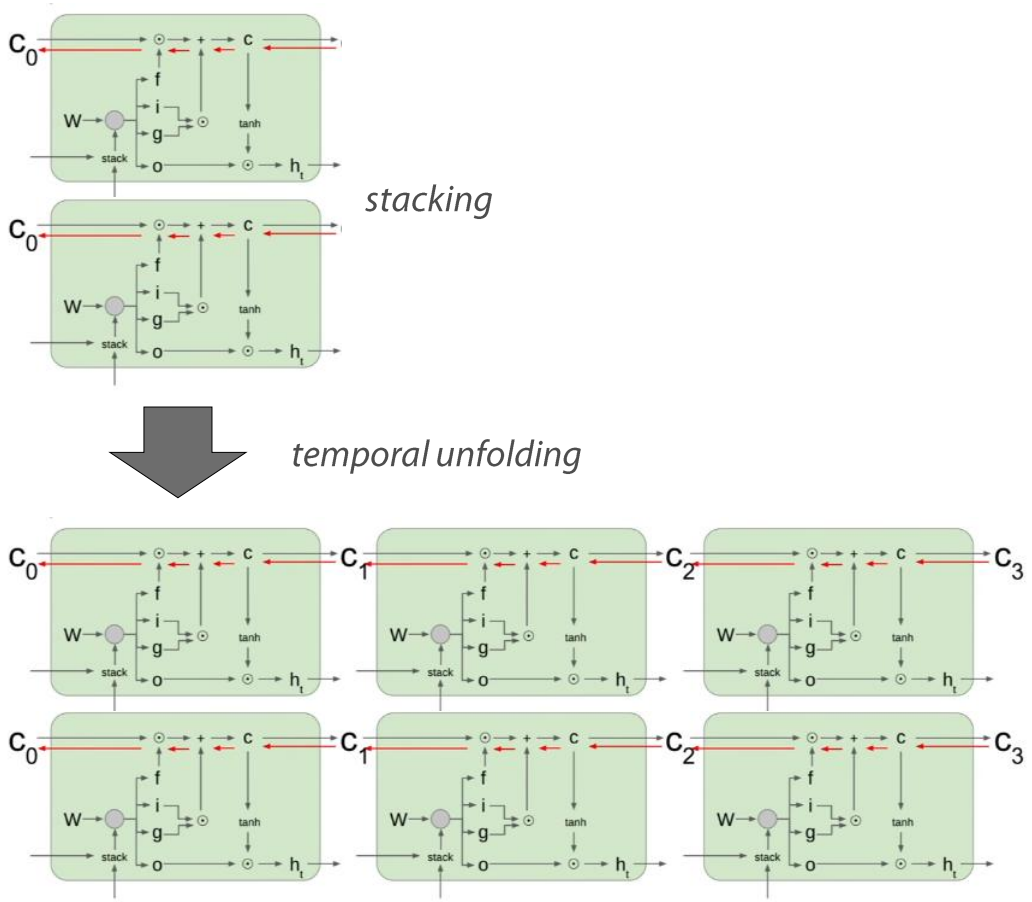
LSTM Training

- **Temporal Unfolding**



LSTM Training

- **Stacking and Temporal Unfolding**



GRU

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

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

(exponential moving average)

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

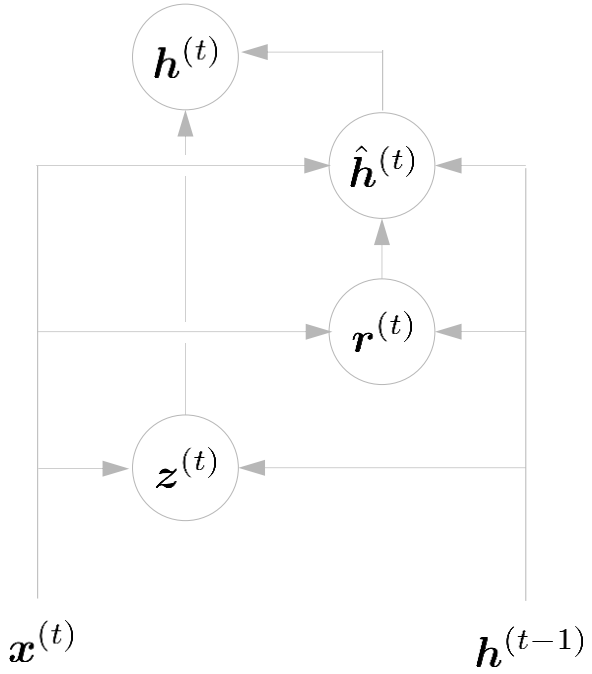
$$\hat{\boldsymbol{h}}^{(t)} := \tanh(\boldsymbol{W}_h \boldsymbol{x}^{(t)} + \boldsymbol{U}_h(\boldsymbol{r}^{(t)} \odot \boldsymbol{h}^{(t-1)}) + \boldsymbol{b}_h)$$

sigmoid function

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

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

Simpler structure, no internal memory



GRU

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

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

(exponential moving average)

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

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

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*

Simpler structure, no internal memory