Università degli
Studi di Pavia

Deep Learning

09- Deep Learning and Time Series

Marco Piastra & Andrea Pedrini(*)

(*) Dipartimento di Matematica F. Casorati

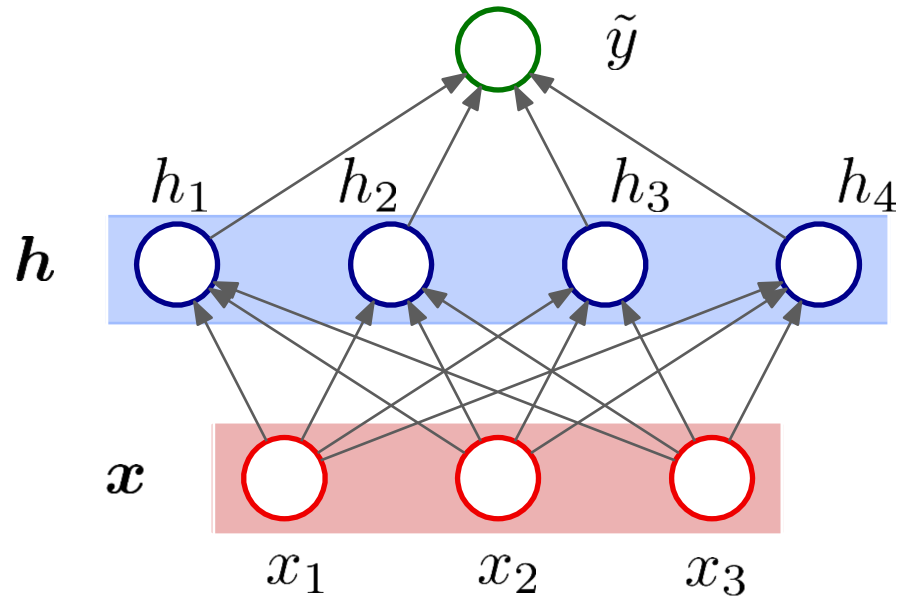
This presentation can be downloaded at:
http://vision.unipv.it/DL

Recurrent Neural Networks

Recurrent Neural Networks

- ***Feed-forward neural network***

$$\tilde{y} = \boldsymbol{w} \cdot \boldsymbol{h} + b \qquad \textit{where} \qquad \underbrace{\boldsymbol{h}}_{\text{hidden}} := g(\boldsymbol{W}\underbrace{\boldsymbol{x}}_{\text{input}} + \boldsymbol{b})$$



Recurrent Neural Networks

- **Feed-forward neural network**

$$\tilde{y} = \boldsymbol{w} \cdot \boldsymbol{h} + b \qquad \text{where} \qquad \underbrace{\boldsymbol{h}}_{\text{hidden}} := g(\boldsymbol{W} \underbrace{\boldsymbol{x}}_{\text{input}} + \boldsymbol{b})$$

- **Recurrent Neural Network**

$$\tilde{y}^{(t)} = \boldsymbol{w} \cdot \boldsymbol{h}^{(t)} + b \qquad \text{where} \qquad \underbrace{\boldsymbol{h}^{(t)}}_{\text{hidden at } t} := g(\boldsymbol{W} \underbrace{\boldsymbol{x}^{(t)}}_{\text{input at } t} + \boldsymbol{U} \underbrace{\boldsymbol{h}^{(t-1)}}_{\text{hidden at } t\text{-}1} + \boldsymbol{b})$$

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

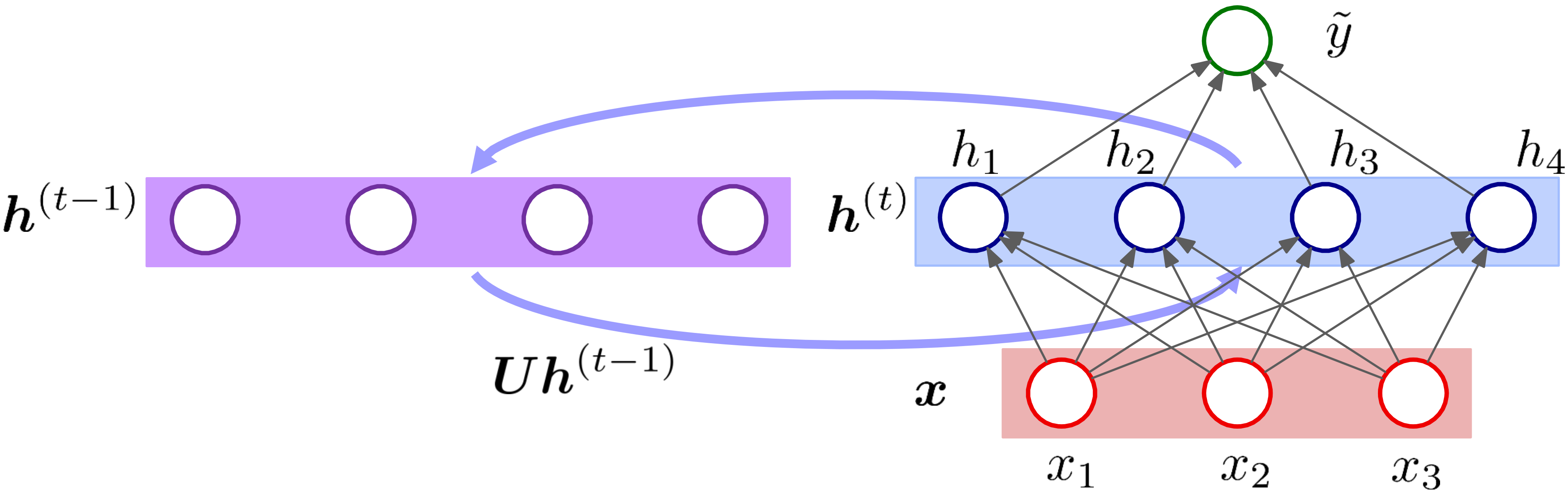
Recurrent Neural Networks

- **Recurrent Neural Network**

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

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

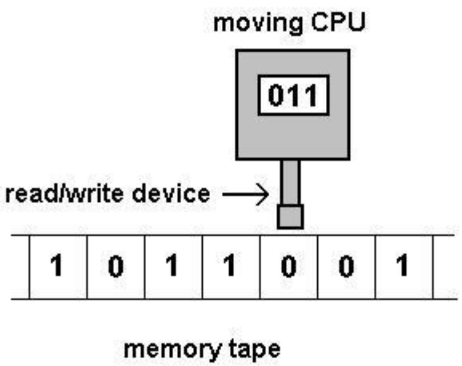
- ***Recurrent Neural Network***

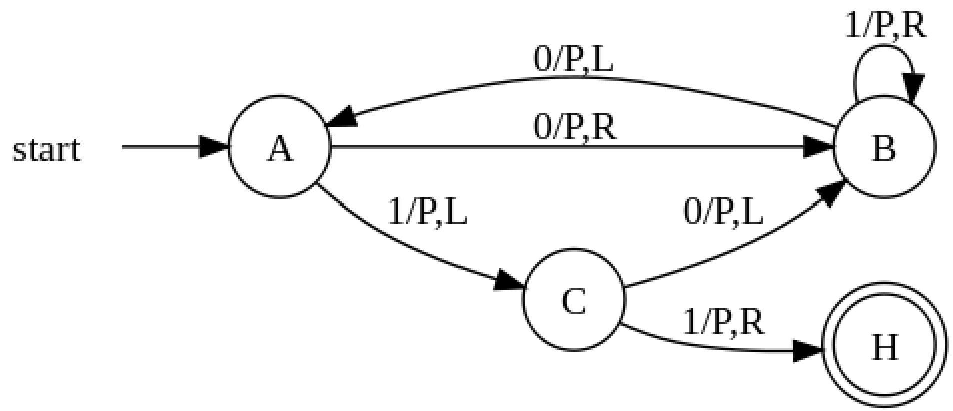
$$\tilde{y}^{(t)} = \boldsymbol{w} \cdot \boldsymbol{h}^{(t)} + b \qquad \textit{where} \qquad \boxed{\underbrace{\boldsymbol{h}^{(t)}}_{\text{hidden at } t} := g(\boldsymbol{W}\underbrace{\boldsymbol{x}^{(t)}}_{\text{input at } t} + \boldsymbol{U}\underbrace{\boldsymbol{h}^{(t-1)}}_{\text{hidden at } t\text{-}1} + \boldsymbol{b})}$$

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

$$\underbrace{\boldsymbol{h}^{(t)}}_{\text{hidden at } t} := g(\boldsymbol{W}\underbrace{\boldsymbol{x}^{(t)}}_{\text{input at } t} + \boldsymbol{U}\underbrace{\boldsymbol{h}^{(t-1)}}_{\text{hidden at } t-1} + \boldsymbol{b})$$

- ## *General Properties*

A recurrent neural network (RNN) is even more powerful than a FF neural network

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

It is much harder to train than a FF neural network

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

Recurrent Neural Networks

- **Recurrent Neural Network**

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

where

$$\underbrace{\boldsymbol{h}^{(t)}}_{\text{hidden at } t} := g(\boldsymbol{W} \underbrace{\boldsymbol{x}^{(t)}}_{\text{input at } t} + \boldsymbol{U} \underbrace{\boldsymbol{h}^{(t-1)}}_{\text{hidden at } t\text{-}1} + \boldsymbol{b})$$

- **Temporal Unfolding**

output at (3)

$$\tilde{y}^{(3)} = \boldsymbol{w} \cdot \boldsymbol{h}^{(3)} + b$$

$$\boldsymbol{h}^{(3)} := g(\boldsymbol{W}\boldsymbol{x}^{(3)} + \boldsymbol{U}\boldsymbol{h}^{(2)} + \boldsymbol{b})$$

$$\boldsymbol{h}^{(2)} := g(\boldsymbol{W}\boldsymbol{x}^{(2)} + \boldsymbol{U}\boldsymbol{h}^{(1)} + \boldsymbol{b})$$

$$\boldsymbol{h}^{(1)} := g(\boldsymbol{W}\boldsymbol{x}^{(1)} + \boldsymbol{U}\boldsymbol{h}^{(0)} + \boldsymbol{b})$$

$$\boldsymbol{h}^{(0)} := g(\boldsymbol{W}\boldsymbol{x}^{(0)} + \boldsymbol{U}\boldsymbol{h}^{(-1)} + \boldsymbol{b})$$

default hidden state $\boldsymbol{h}^{(-1)} := \boldsymbol{0}$

input sequence $\boldsymbol{x}^{(0)}, \boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}$

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

output at (3)

$$\tilde{y}^{(3)} = \boldsymbol{w} \cdot \boldsymbol{h}^{(3)} + b$$

This looks very similar to a <u>*deep*</u> *feed-forward neural network …*

$$\boldsymbol{h}^{(3)} := g(\boldsymbol{W}\boldsymbol{x}^{(3)} + \boldsymbol{U}\boldsymbol{h}^{(2)} + \boldsymbol{b})$$

$$\boldsymbol{h}^{(2)} := g(\boldsymbol{W}\boldsymbol{x}^{(2)} + \boldsymbol{U}\boldsymbol{h}^{(1)} + \boldsymbol{b})$$

$$\boldsymbol{h}^{(1)} := g(\boldsymbol{W}\boldsymbol{x}^{(1)} + \boldsymbol{U}\boldsymbol{h}^{(0)} + \boldsymbol{b})$$

$$\boldsymbol{h}^{(0)} := g(\boldsymbol{W}\boldsymbol{x}^{(0)} + \boldsymbol{U}\boldsymbol{h}^{(-1)} + \boldsymbol{b})$$

$\boldsymbol{x}^{(3)}$

$\boldsymbol{x}^{(2)}$

$\boldsymbol{x}^{(1)}$

$\boldsymbol{x}^{(0)}$

input sequence

default hidden state $\boldsymbol{h}^{(-1)} := \boldsymbol{0}$

Long-Short Term Memory (LSTM)

LSTM

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

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

elementwise product

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

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

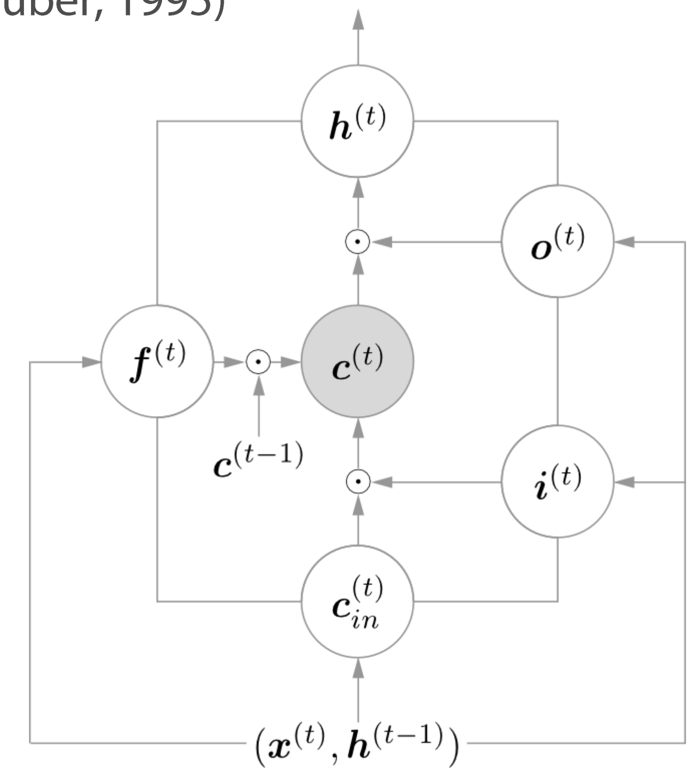
sigmoid function

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

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

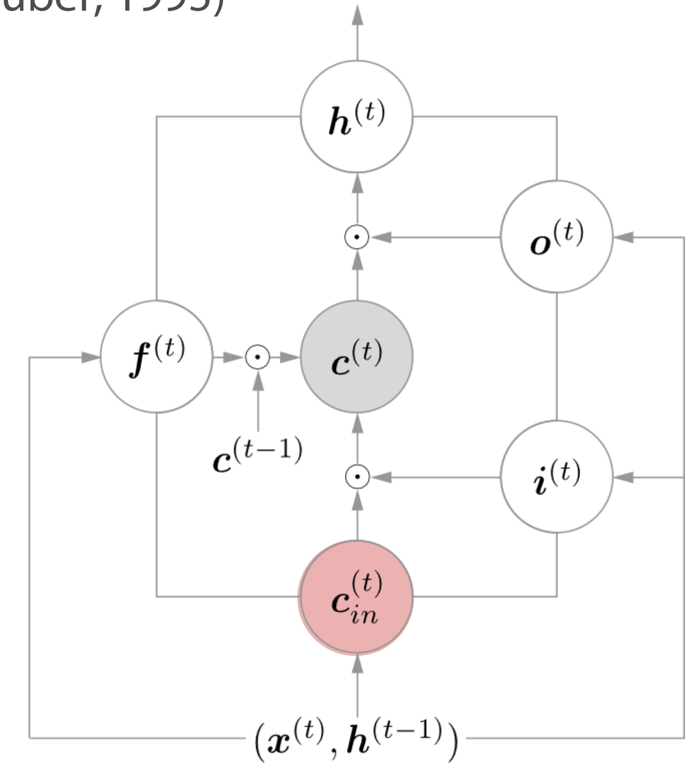
$$\boldsymbol{i}^{(t)} := \sigma(\boldsymbol{W}_i \boldsymbol{x}^{(t)} + \boldsymbol{U}_i \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_i)$$

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



- ***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

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

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

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

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

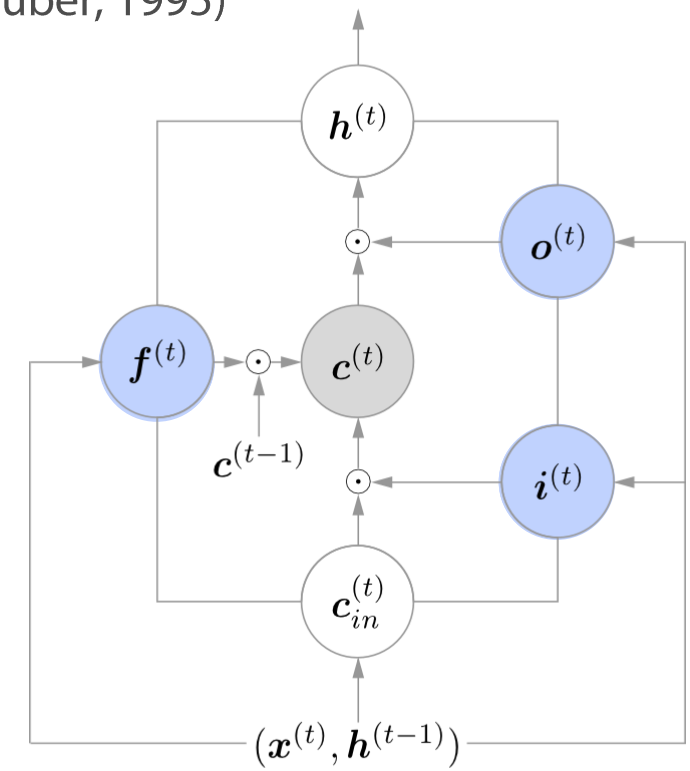
Gating values

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

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

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

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



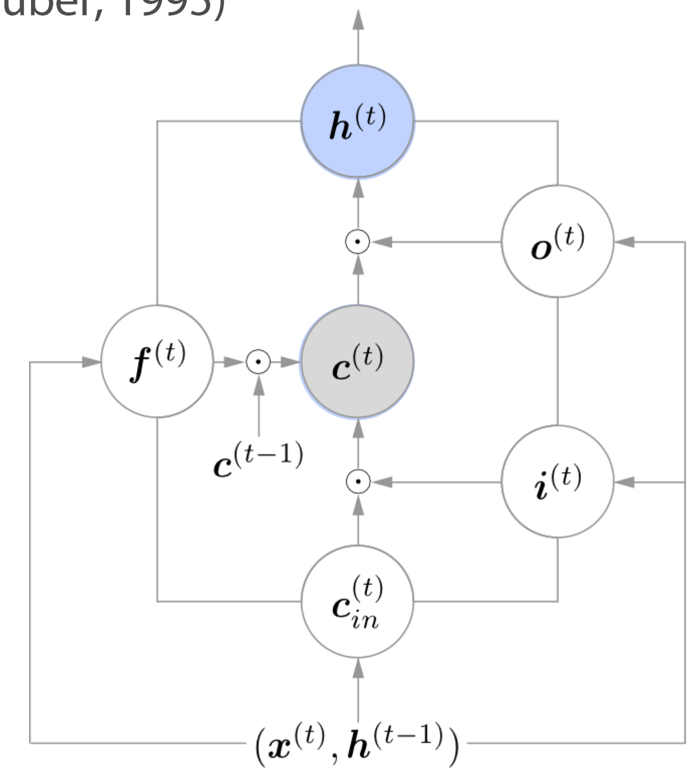
- ***Long-Short Term Memory*** (Hochreiter & Schmidhuber, 1995)

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

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*



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

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

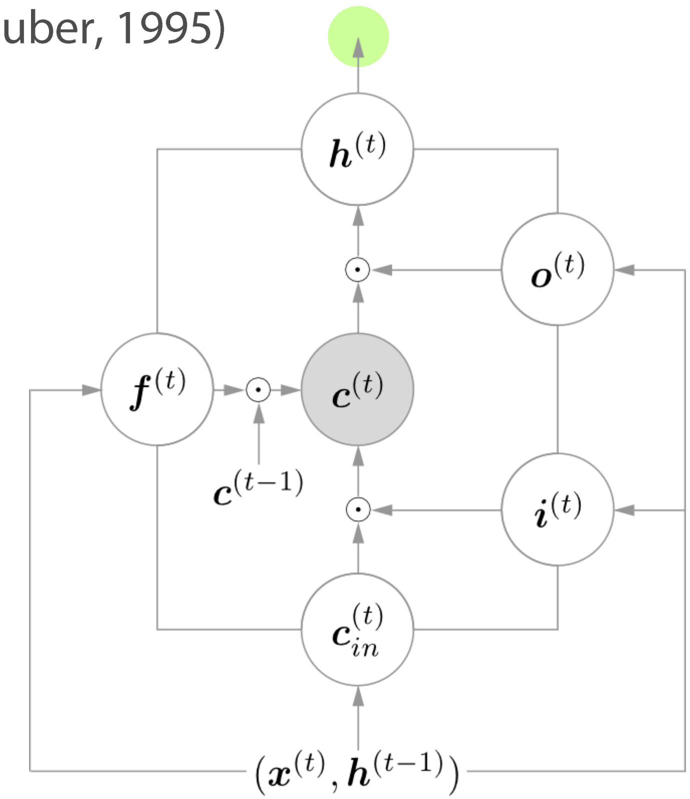
$$\boldsymbol{i}^{(t)} := \sigma(\boldsymbol{W}_i \boldsymbol{x}^{(t)} + \boldsymbol{U}_i \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_i)$$

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

LSTM

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

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



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

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

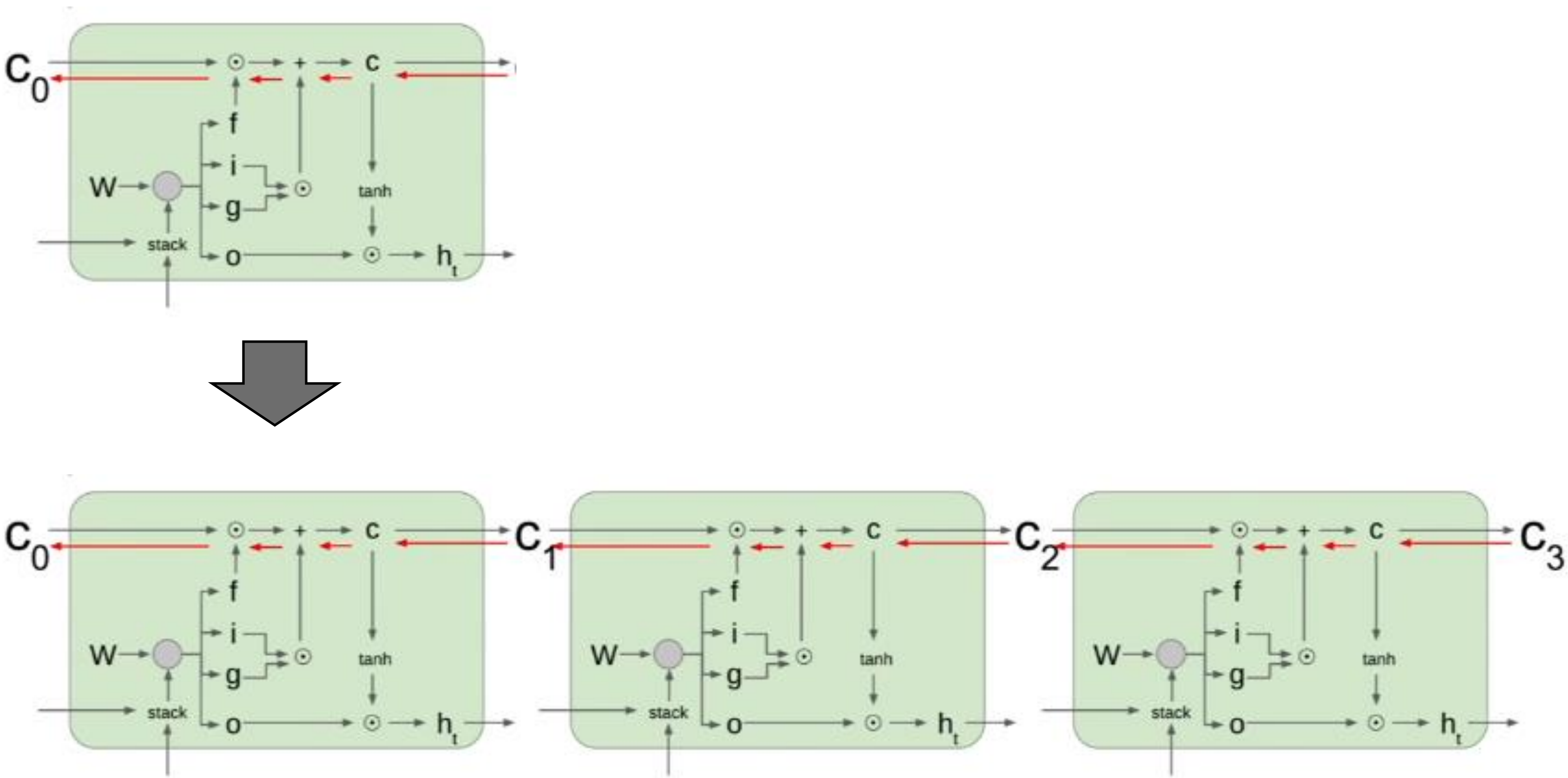
$$\boldsymbol{o}^{(t)} := \sigma(\boldsymbol{W}_o \boldsymbol{x}^{(t)} + \boldsymbol{U}_o \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_o)$$

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

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

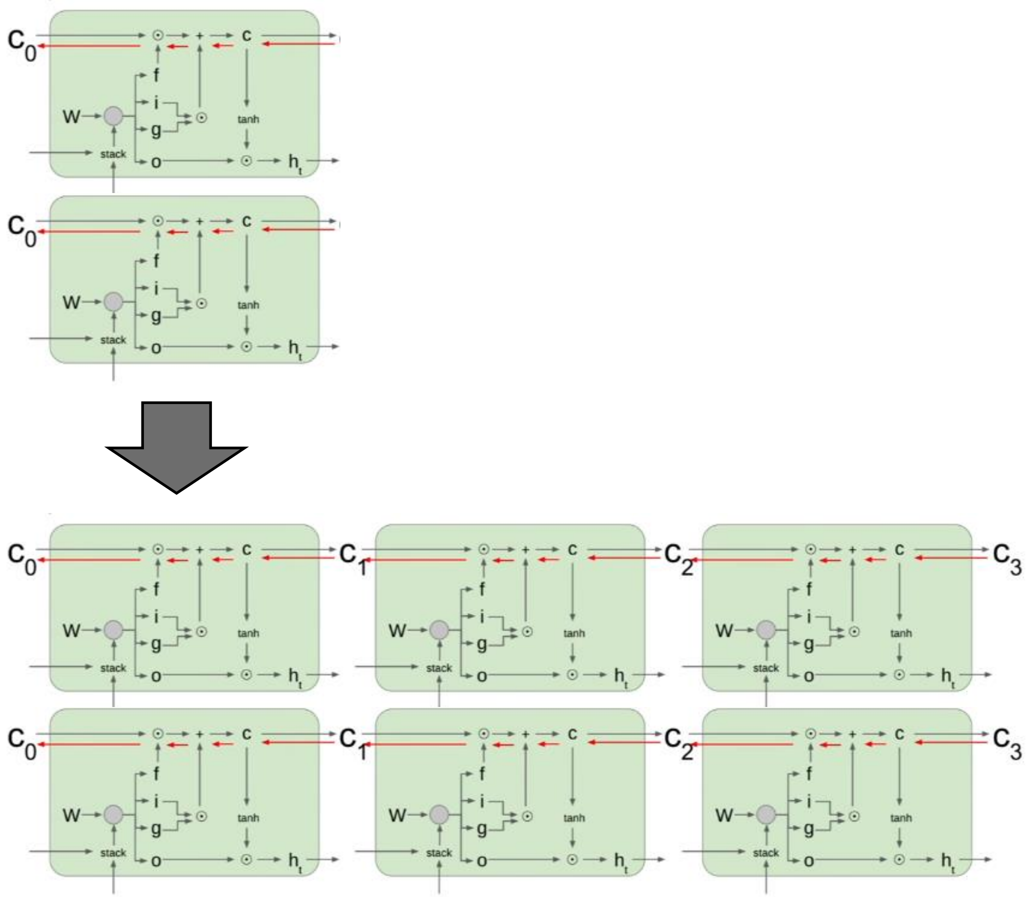
$$\boldsymbol{c}_{in}^{(t)} := \tanh(\boldsymbol{W}_c \boldsymbol{x}^{(t)} + \boldsymbol{U}_c \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_c)$$

- ***Temporal Unfolding***



LSTM

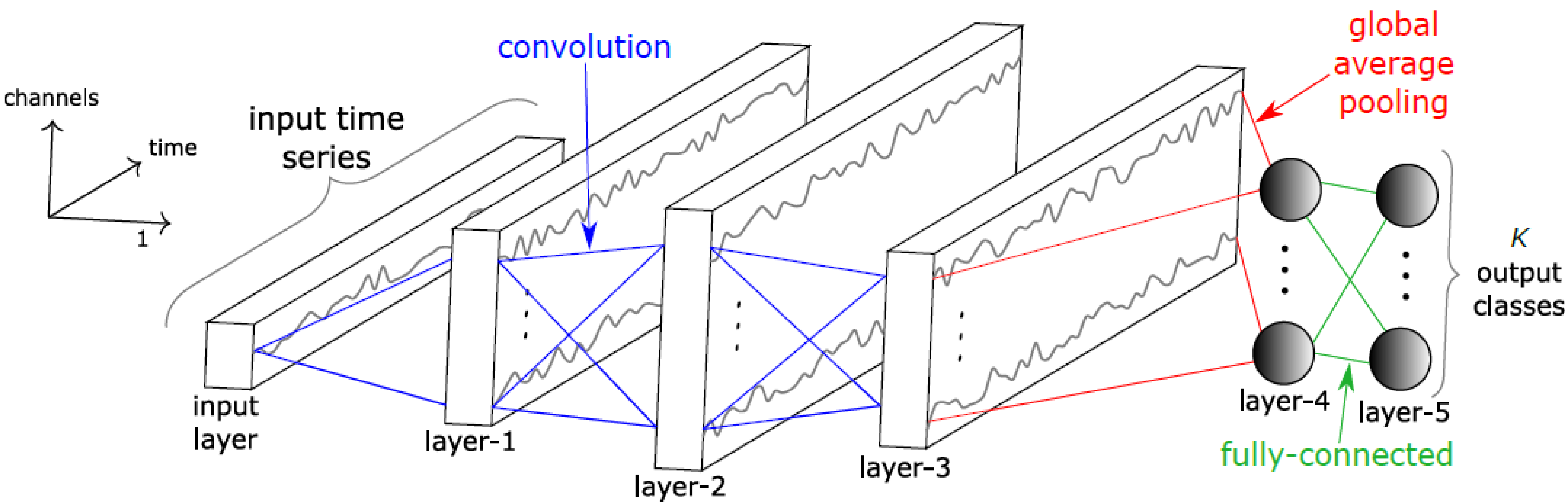
- ***Stacking and Temporal Unfolding***



Deep Convolutional Networks for Time Series Analysis

1D Convolution Over Time

Time windows are treated as 1D 'images'

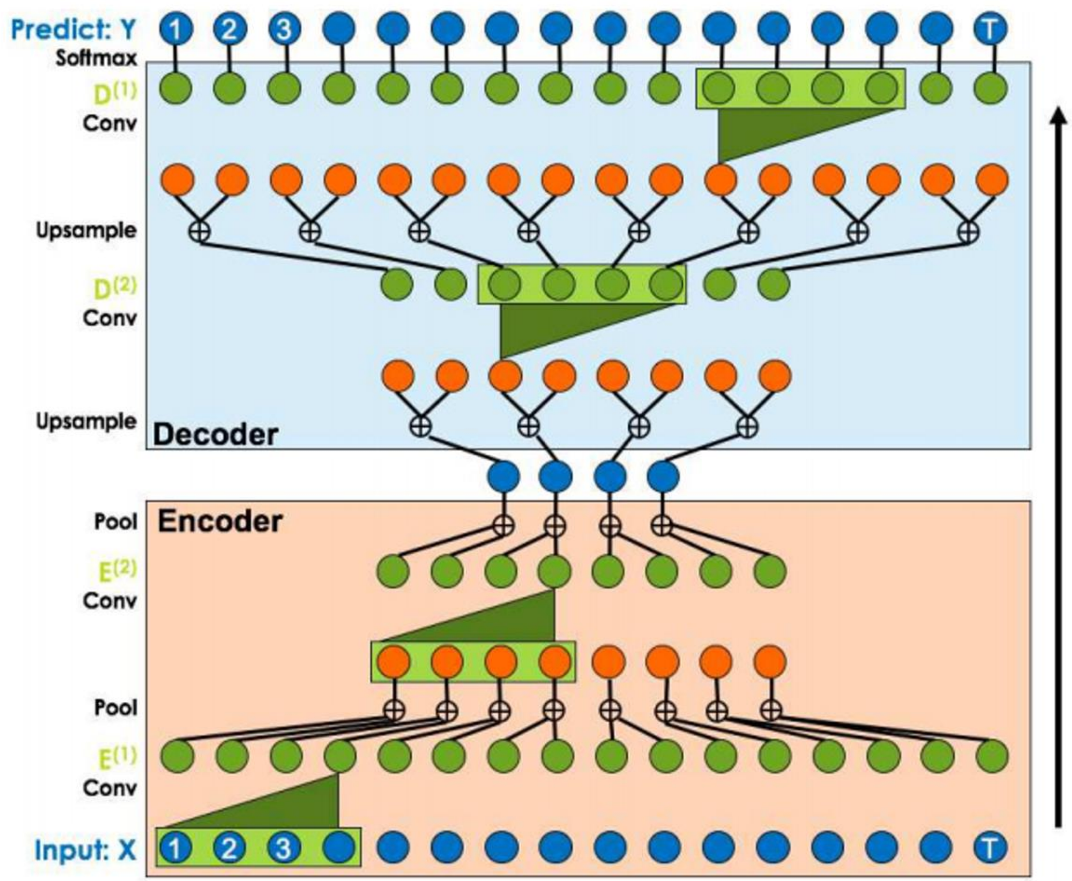


[image from https://link.springer.com/article/10.1007%2Fs10618-019-00619-1]

Temporal Convolution Networks

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

Effective in segmenting actions and <u>predicting</u> time series



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

RNN applications

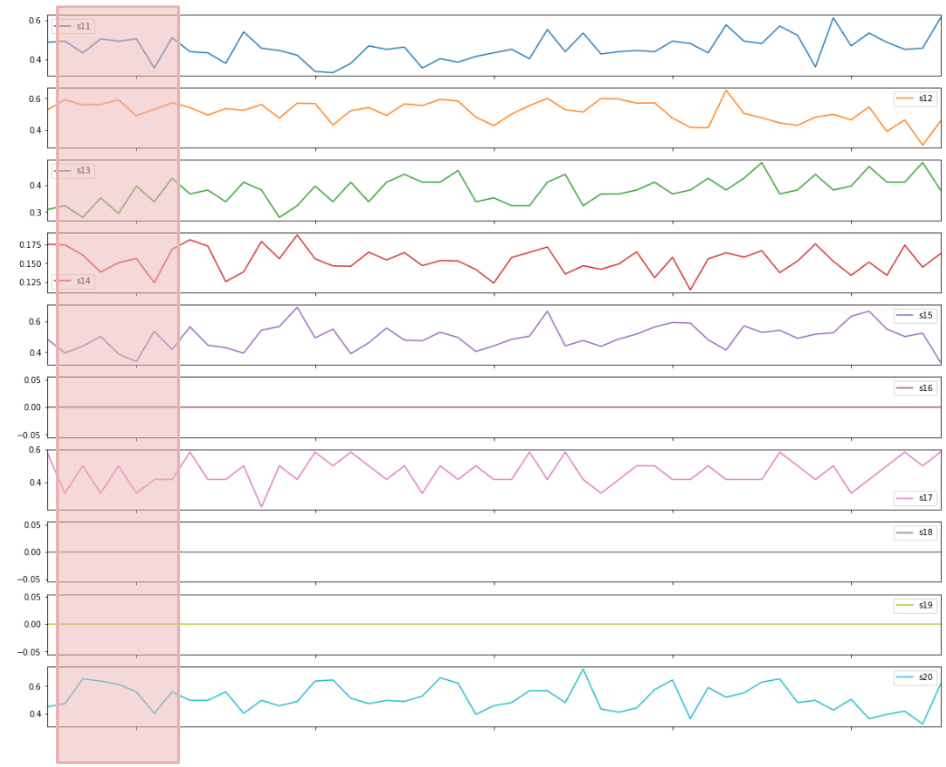
Predictive Maintenance

- ***Detecting failure conditions from sensor readings***

Nominal Condition

Input

Before Failure



Input

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

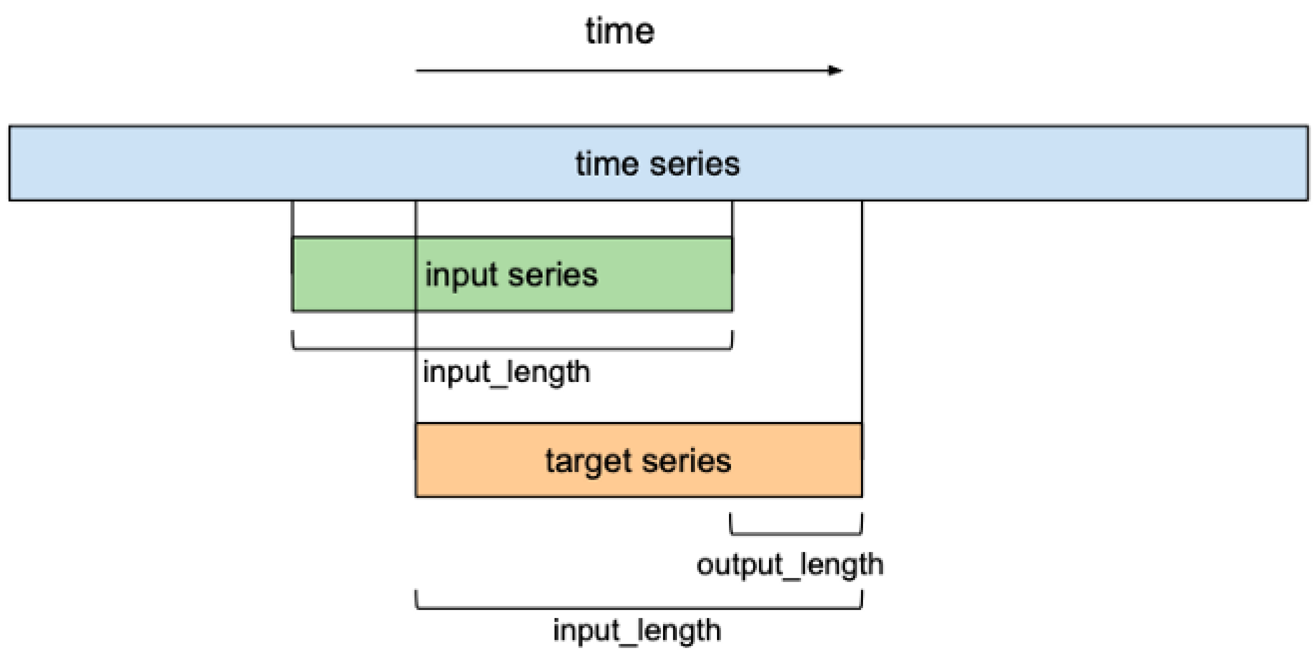
Action Detection

Find relevant actions in video sequences
(See video)

[video from https://www.youtube.com/watch?v=9XphWB9w7p8]

Time Series Forecasting

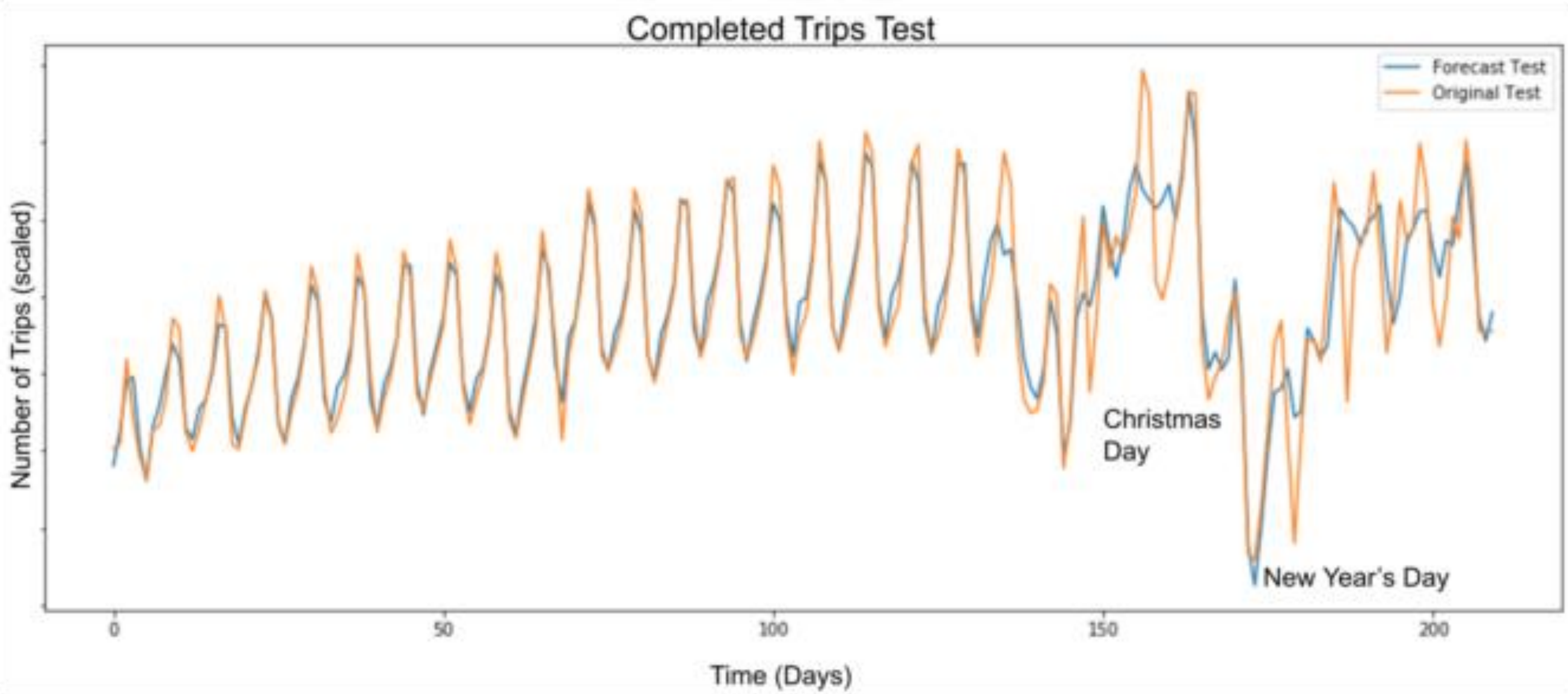
Forecasting time series ahead of time



[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

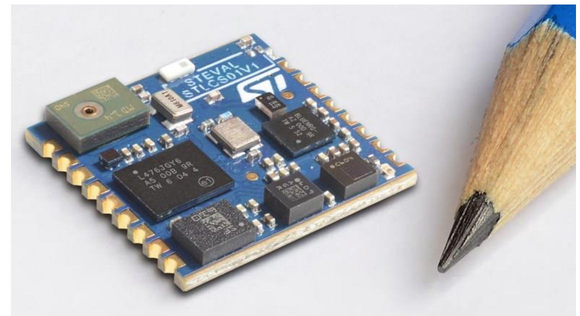


[image from https://eng.uber.com/neural-networks/]

An Aside: Embedded AI (at UniPV)

Credits are due to M. Musci and E. Torti

STM SensorTile for Wearable Devices



- Microcontroller STM32 (ARM Cortex M4)
  - 80MHz Clock Frequency
  - 128KB SRAM
  - 1MB Flash memory
- Multiple Onboard Sensors
  - Microphone
  - 3D Accelerometer + 3D Gyroscope
  - e-Compass, 3D Accelerometer, 3D Magnetic sensor
  - Barometer
- Peripherals
  - 100 mAh Li-Ion battery
  - Bluetooth Low Energy (BLE) radio module

Why Embedding?

> ***Edge computing*** *is a method of optimizing* ***cloud computing systems***
> *by taking some portion of an application, its data, or services*
> *away from one or more central nodes (the "core") to the other logical extreme*
> *(the "edge") of the Internet which makes contact with the physical world or end users*
> [Wikipedia]

- **A *critical* problem for (any) Intelligent *Wireless* Sensor**

  Wireless transmission of raw sensor data to the cloud requires
  a substantial amount of power

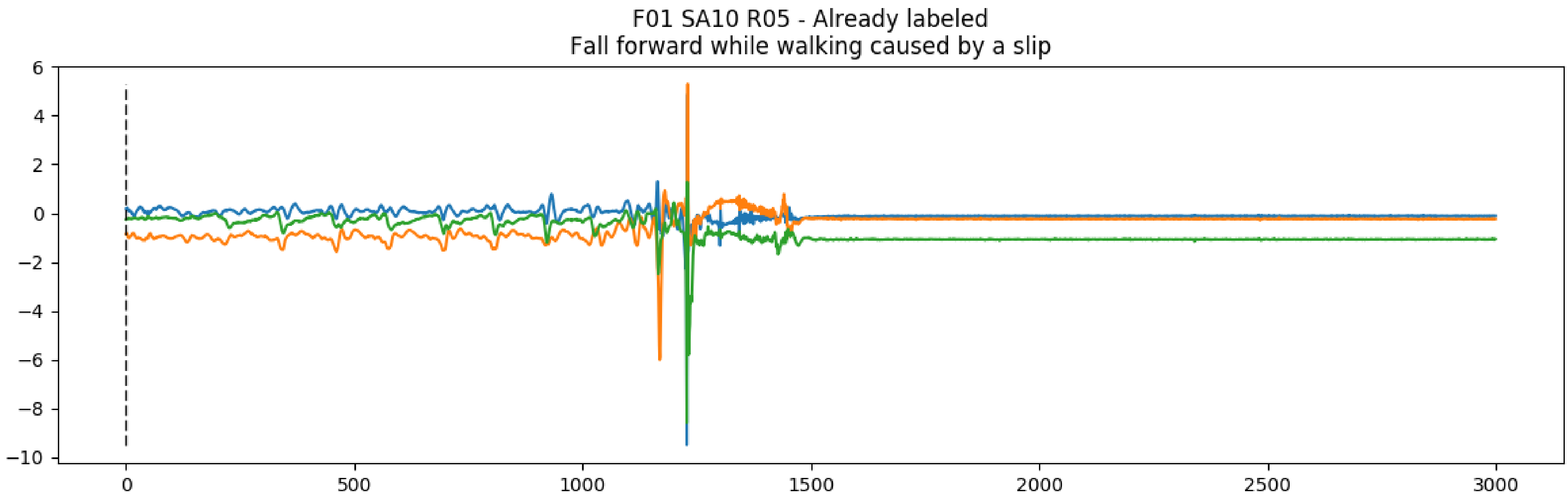
  All Internet-of-Things Low-Power Wide Area protocols (LPWA: Lora, Sigfox, NB-IoT)
  but also BLE, ZigBee, etc. are optimized for sparse and infrequent <u>*short*</u> messages

  Ideally, in an Intelligent Sensor, data processing must be performed <u>***onboard***</u>*:*
  short messages should be sent only when relevant events or state transitions occur

Human Fall (rehearsed)

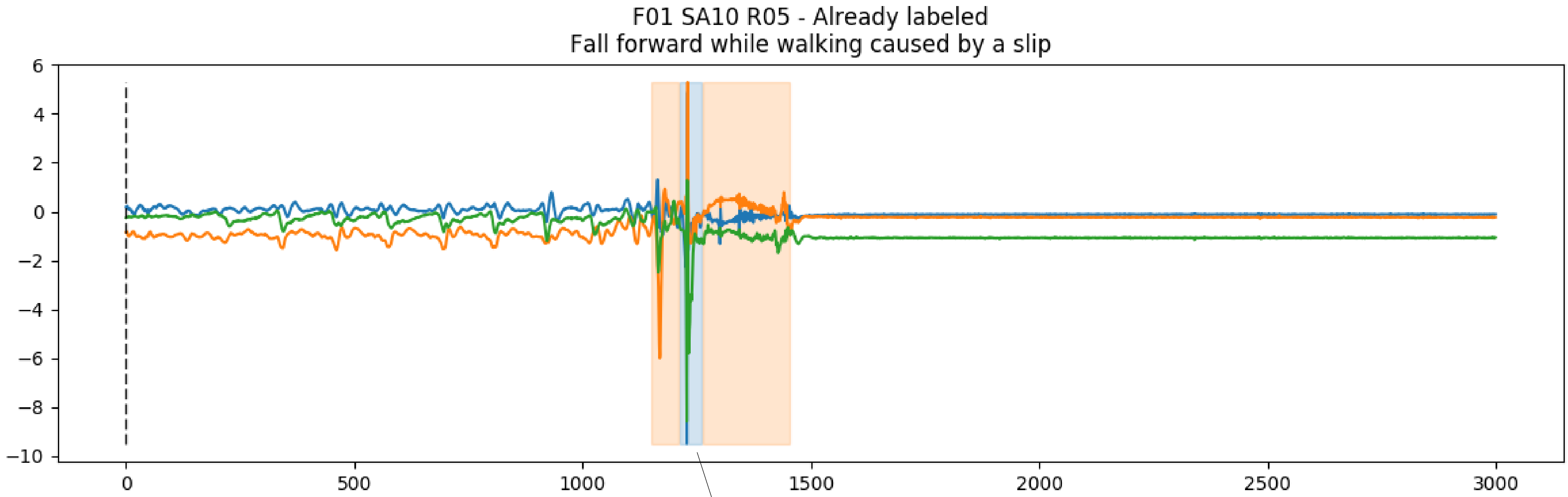
Human Fall (rehearsed)

- Accelerometers



Human Fall (rehearsed)

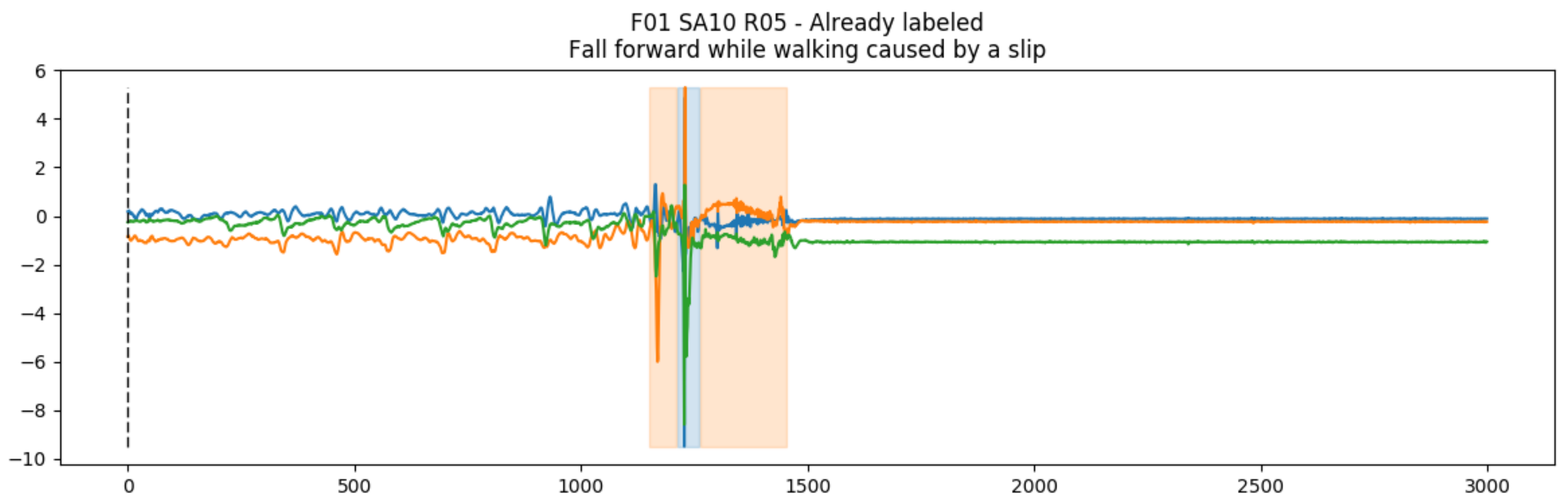
- Accelerometers (annotated)



the actual fall occurred here

Fall Detection

- *Detecting human falls from accelerometer signals is difficult*
  - The input is time-variant (*unlike a still image*)
  - Information is limited (*just three scalar values at each time instant*)
  - It is the 'history' (*i.e. the shape*) of signals that describes the event
  - False negatives are to be avoided, but even false positives….



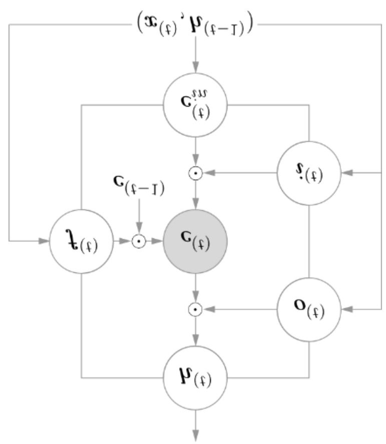
Smart Sensors: a Case Study

- ***Fall detection with wearable sensors (IPHSDM)***

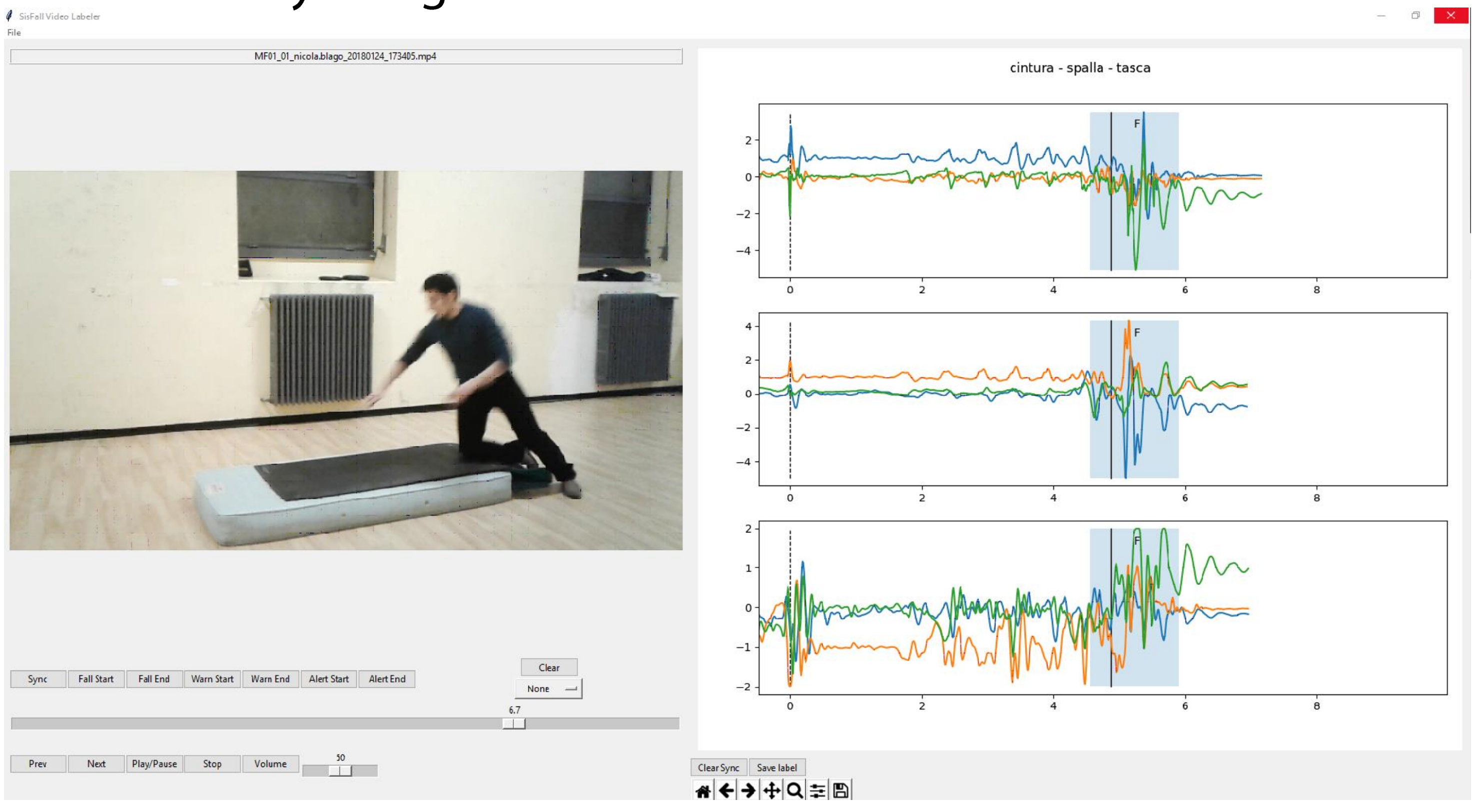
A project co-funded by Regione Lombardia

Event detection is performed by the smart sensors (SensorTile)
Messages are sent over BLE only to signal relevant events

Long-Short Term Memory (LSTM)

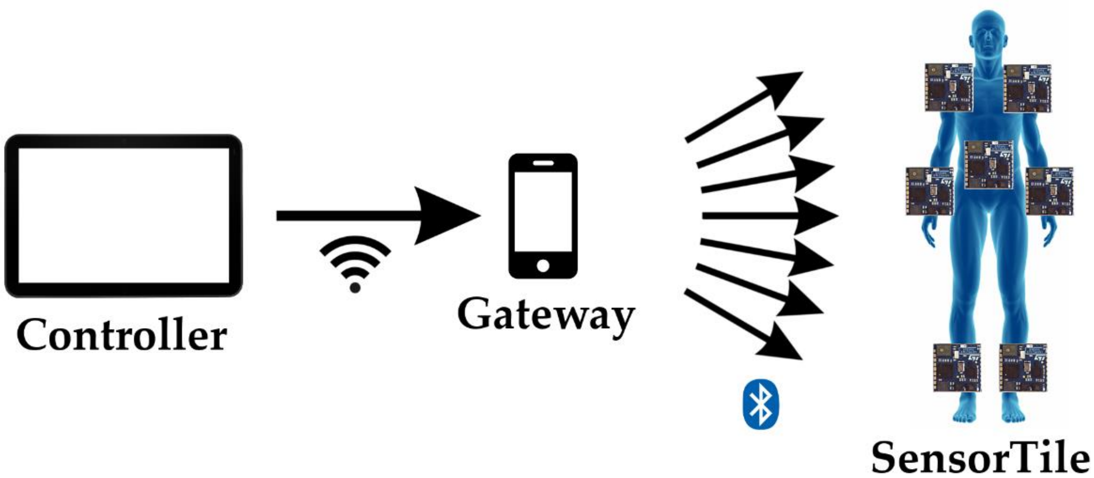


Embedded on Device



Creating a dataset
(for fall detection)

Body Network



Body Network

Simulated Falls

Dataset Annotation