

Opening

Part I. Satellite Image Time Series

Time Series

Time Series Classification

Part II. Deep learning for SITS

Introduction

Convolutional Neural Networks

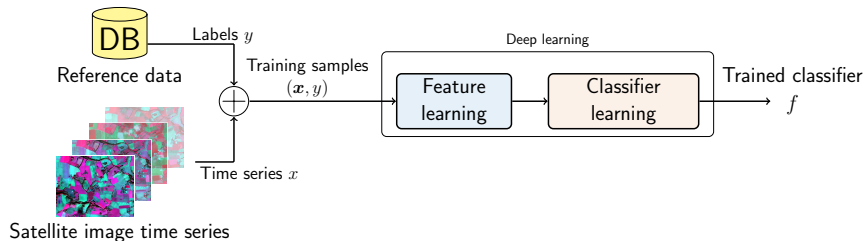
Recurrent Neural Networks

Attention-based architectures

Closing remarks

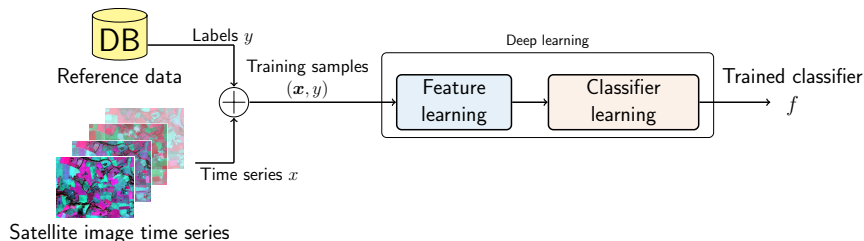
# From Machine Learning to Deep Learning

Features are extracted automatically in deep learning



# From Machine Learning to Deep Learning

Features are extracted automatically in deep learning

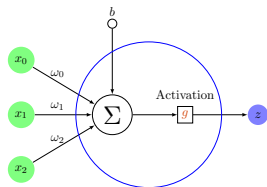


**Architecture design** is the new feature engineering! One needs to choose

- ◇ the type of network,
- ◇ the number of layers (depth)
- ◇ the number of units per layer (width)
- ◇ the learning strategy (optimizer, learning rate)
- ◇ *etc.*

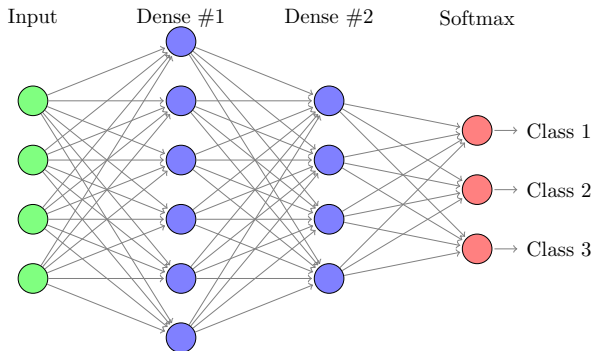
# How to train a network?

Training a network = finding parameter values that minimize the cost function



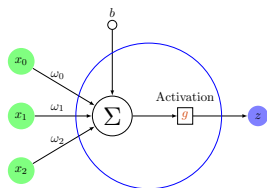
$$y = g\left(\sum_i \omega_i \cdot x_i + b\right)$$

$$y = g(\mathbf{w}^T \mathbf{x} + b)$$



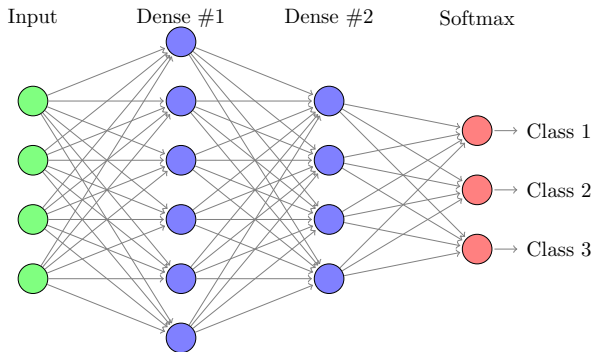
# How to train a network?

Training a network = finding parameter values that minimize the cost function



$$y = g\left(\sum_i \omega_i \cdot x_i + b\right)$$

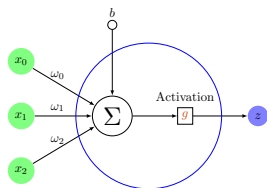
$$y = g(\mathbf{w}^T \mathbf{x} + b)$$



1. **Forward** step: estimate the cost function

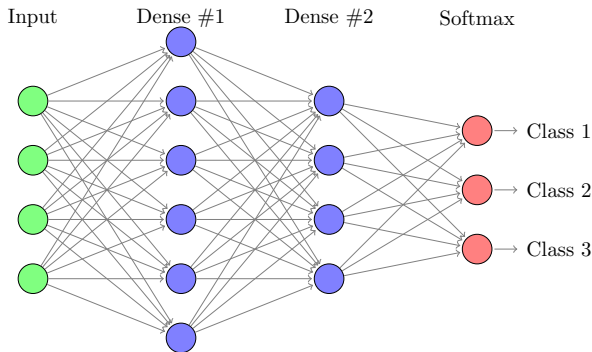
# How to train a network?

Training a network = finding parameter values that minimize the cost function



$$y = g\left(\sum_i \omega_i \cdot x_i + b\right)$$

$$y = g(\mathbf{w}^T \mathbf{x} + b)$$

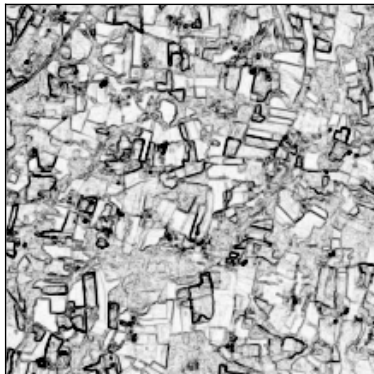
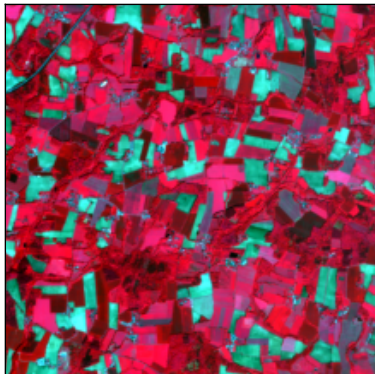


2. **Backward** step: update the parameter values through gradient descent

## A. Convolutional Neural Networks

## Convolution for images

The convolution is a common image processing techniques for images and signals.



The result of applying a convolution filter (here an edge detection filter) on a Sentinel-2 image.



### How it works?

- ◇ The result of applying an edge detection on a time series:

### How it works?

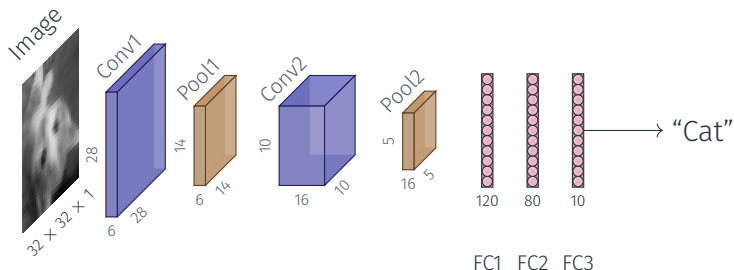
◇ The result of applying an edge detection on a time series:

◇ A convolution (actually a cross-correlation) between a time series  $x$  and a filter  $w$  at instant  $t$  can be expressed as:  $(x * w)(t) = \sum_{i+j=t} x_i \cdot w_j$



# Convolutional Neural Networks

- ◇ **Learn** the weight of the convolution filter during the network training
- ◇ **Stack** several convolution layers
  - ◇ first convolution layers extract simple features such as edges
  - ◇ last convolution layers extract more complex features

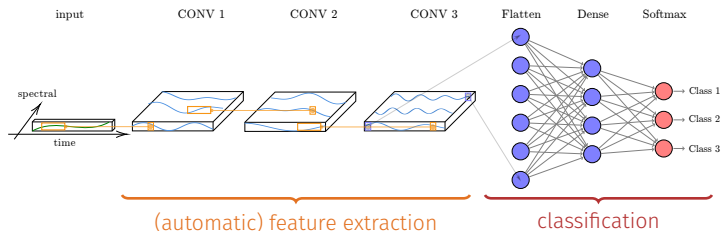


LeNet architecture [1]

[1] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.

## Temporal Convolutional Neural Networks (TempCNN) [1]

- ◇ fixed filter size
- ◇ no pooling layers between convolutional layers



[1] Pelletier, C., Webb, G. I., & Petitjean F. (2019). Temporal convolutional neural network for the classification of satellite image time series. *Remote Sensing*, 11(5), 523.

## B. Recurrent Neural Networks

## Recurrent Neural Networks

---

Recurrent Neural Networks (RNNs) are intrinsically designed for sequence data:

- ◇ able to explicitly consider the **temporal correlation** of the data
- ◇ state-of-the-art architectures for forecasting tasks

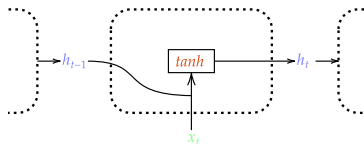
# Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are intrinsically designed for sequence data:

- ◇ able to explicitly consider the **temporal correlation** of the data
- ◇ state-of-the-art architectures for forecasting tasks

**A recurrent cell:** at each timestamp  $t$ ,

- ◇ the state of the recurrent cell is affected by past information  $h_{t-1}$  and the current time-series element  $x_t$
- ◇  $(W_x, W_h, b_h)$  are the trainable weights and bias learned with backpropagation through time



$$h_t = \tanh(W_x x_t + W_h h_{t-1} + b_h)$$



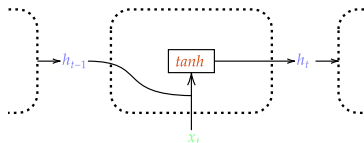
# Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are intrinsically designed for sequence data:

- ◇ able to explicitly consider the **temporal correlation** of the data
- ◇ state-of-the-art architectures for forecasting tasks

**A recurrent cell:** at each timestamp  $t$ ,

- ◇ the state of the recurrent cell is affected by past information  $h_{t-1}$  and the current time-series element  $x_t$
- ◇  $(W_x, W_h, b_h)$  are the trainable weights and bias learned with backpropagation through time



$$h_t = \tanh(W_x x_t + W_h h_{t-1} + b_h)$$

RNNs are good at

- ◇ considering past (possibly future) information during computations
- ◇ considering time series of different lengths
- ◇ sharing weights across time

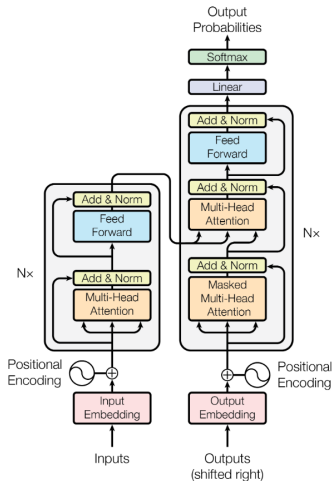
but they are slow to train due to backpropagation through time, and fail to extract long temporal dependencies

## C. Attention-based architectures

# Transformers

Attention mechanisms were initially proposed by [1], they become popular with Transformers in 2017 [2]

- ◇ make the most of GPU
- ◇ encoder-decoder architecture similar to RNNs
- ◇ develop for language translation or sentence generation



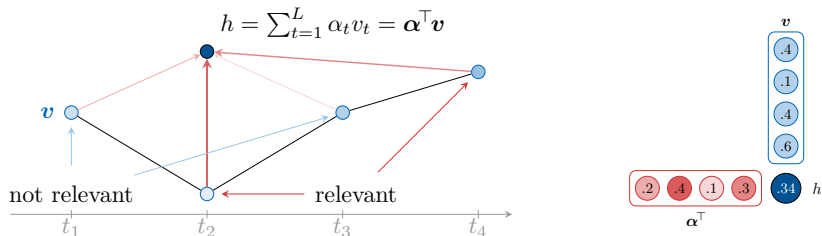
[1] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

[2] Waswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L & Polosukhin, I. (2017). Attention is all you need. In *Conference on Neural Information Processing Systems (NIPS)*

## Attention mechanism

**Objective:** focusing on the relevant elements of the time series

- Given **values**  $\mathbf{v} \in \mathbb{R}^L$  as a sequence of observations .
- We want to calculate an output  $\mathbf{h}$  based only on **classification-relevant** observations.
- This is realized by a weighted sum over **attention scores**  $\alpha \in \mathbb{R}^L$ .

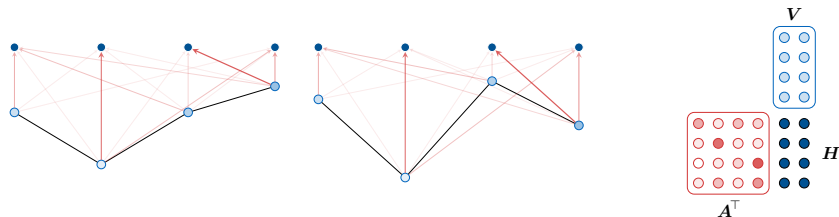


$$\mathbf{h} = \text{Attention}(\alpha, \mathbf{v}) = \alpha^T \mathbf{v} = \sum_{t=1}^L \alpha_t v_t,$$
$$\alpha \in [0, 1]^L, \mathbf{v} \in \mathbb{R}^L$$

# Attention mechanism

**Objective:** focusing on the relevant elements of the time series

- ◇ Given **values**  $\mathbf{v} \in \mathbb{R}^L$  as a sequence of observations .
- ◇ We want to calculate an output  $\mathbf{h}$  based only on **classification-relevant** observations.
- ◇ This is realized by a weighted sum over **attention scores**  $\alpha \in \mathbb{R}^L$ .



$$\mathbf{H} = \text{Attention}(\mathbf{A}, \mathbf{V}) = \mathbf{A}^T \mathbf{V}, \quad \mathbf{A} \in [0, 1]^{L \times L}, \mathbf{V} \in \mathbb{R}^{L \times D_V}$$

where  $D_V$  is the dimension of the time series  $\mathbf{v}$ .

## How to compute the attention scores?

- ◇ We calculate scores from one **query**  $\mathbf{q} \in \mathbb{R}^{D_k}$  and  $L$  **keys**  $\mathbf{K} = (\mathbf{k}_t)_{t \in [1, L]} \in \mathbb{R}^{L \times D_k}$

$$\alpha_t(\mathbf{q}, \mathbf{K}) = \frac{\exp(\text{sim}(\mathbf{q}, \mathbf{k}_t))}{\sum_{\tau=1}^L \exp(\text{sim}(\mathbf{q}, \mathbf{k}_\tau))}$$

- ◇ The **query**  $\mathbf{q}$  provides a semantic **context** that is compared to a **key**  $\mathbf{k}_t$  for each sequence element  $t$  using a similarity measure  $\text{sim}$ .
- ◇ The softmax normalization  $\frac{\exp(\cdot)}{\sum \exp(\cdot)}$  ensures that  $\sum_{t=1}^L \alpha_t = 1$ .

## How to compute the attention scores?

- ◇ We calculate scores from one **query**  $\mathbf{q} \in \mathbb{R}^{D_r}$  and  $L$  **keys**  $\mathbf{K} = (\mathbf{k}_t)_{t \in [1, L]} \in \mathbb{R}^{L \times D_r}$

$$\alpha_t(\mathbf{q}, \mathbf{K}) = \frac{\exp(\text{sim}(\mathbf{q}, \mathbf{k}_t))}{\sum_{\tau=1}^L \exp(\text{sim}(\mathbf{q}, \mathbf{k}_\tau))}$$

- ◇ The **query**  $\mathbf{q}$  provides a semantic **context** that is compared to a **key**  $\mathbf{k}_t$  for each sequence element  $t$  using a similarity measure  $\text{sim}$ .
- ◇ The softmax normalization  $\frac{\exp(\cdot)}{\sum \exp(\cdot)}$  ensures that  $\sum_{t=1}^L \alpha_t = 1$ .

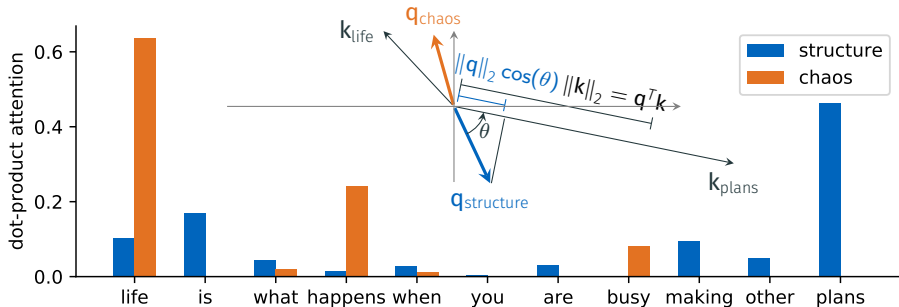
A variety of similarity measures:

cosine distance [1]	$\text{sim}(\mathbf{q}, \mathbf{k}) = \frac{\mathbf{q}^T \mathbf{k}}{\ \mathbf{q}\ _2 \ \mathbf{k}\ _2}$
dot-product [2]	$\text{sim}(\mathbf{q}, \mathbf{k}) = \mathbf{q}^T \mathbf{k}$
scaled dot-product [3]	$\text{sim}(\mathbf{q}, \mathbf{k}) = \frac{\mathbf{q}^T \mathbf{k}}{\sqrt{D_r}}$

# Dot-Product Attention on Word Embeddings

## Text example

- ◇ Each word is embedded into a 300-dimensional semantic GloVe Vector, e.g.,  $\mathbf{e}_{\text{structure}} = E(\text{"structure"}) \in \mathbb{R}^{300}$ .
- ◇ Embeddings of two query words "structure" and "chaos" are compared to a sentence of keys "life is what happens when you are busy making other plans"



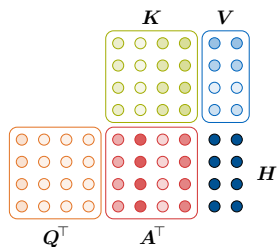
## Core idea:

If two words point in the same direction ( $\theta \approx 0$ ) **attention** is high.



## Self-attention

How to determine the values, keys, and queries?

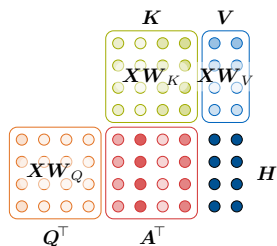


$$\text{Attention}(K, Q, V) = \overbrace{\text{softmax}(Q^T K)}^{A^T} V,$$
$$V \in \mathbb{R}^{L \times D_V}, Q, K \in \mathbb{R}^{D_R \times L}, A \in \mathbb{R}^{L \times L}$$

## Self-attention

How to determine the values, keys, and queries?

- ◇ the self-attention mechanism uses linear projection of the input sequences  $X$



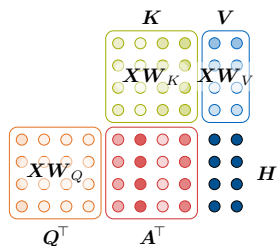
$$\text{Attention}(K, Q, V) = \text{softmax} \left( \overbrace{Q^T K}^{A^T} \right) V,$$
$$V \in \mathbb{R}^{L \times D_V}, Q, K \in \mathbb{R}^{D_R \times L}, A \in \mathbb{R}^{L \times L}$$

$$\begin{aligned} \text{Self-Attention}_W(X) &= \text{Attention}(XW_K, XW_Q, XW_V) \\ &= \text{softmax} \left( (XW_Q)^T (XW_K) \right) (XW_V) \end{aligned}$$

# Self-attention

How to determine the values, keys, and queries?

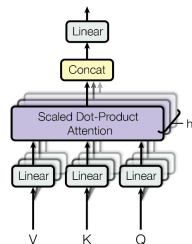
- the self-attention mechanism uses linear projection of the input sequences  $X$



$$\text{Attention}(K, Q, V) = \text{softmax} \left( \overbrace{Q^T K}^{A^T} \right) V,$$
$$V \in \mathbb{R}^{L \times D_V}, Q, K \in \mathbb{R}^{D_R \times L}, A \in \mathbb{R}^{L \times L}$$

$$\text{Self-Attention}_W(X) = \text{Attention}(XW_K, XW_Q, XW_V)$$
$$= \text{softmax} \left( (XW_Q)^T (XW_K) \right) (XW_V)$$

Self-attention is usually applied in parallel heads, which is known as **multi-head attention**.

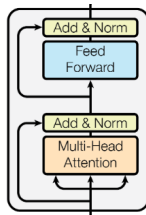
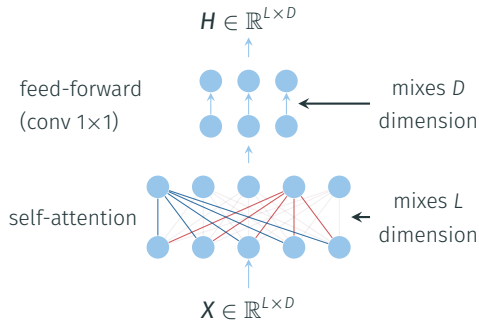


## Transformers encoder

The Transformers are composed of **encoder blocks** that map a  $D$ -dimensional input times series  $X$  of length  $L$  into a higher-level representation  $H \in \mathbb{R}^{L \times D}$ . Each block is composed of:

1. multi-head attention that mixes dimension  $L$
2. feed-forward networks (convolutions of size  $1 \times 1$ ) that mixes dimension  $D$

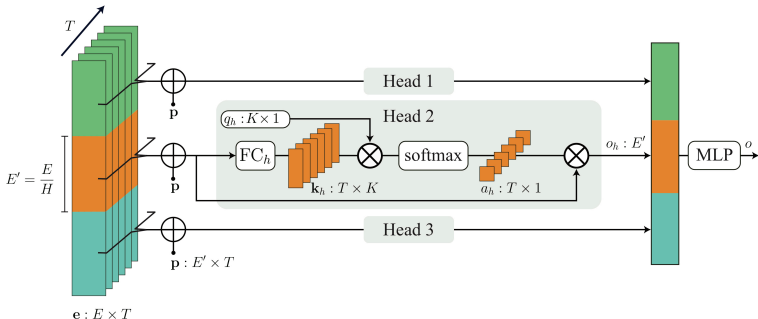
A block also includes skip connections and normalization.



# Transformers in remote sensing

In SITS classification, we want to **predict one label** per time series, not a sequence of words as in sentence translation or generation.

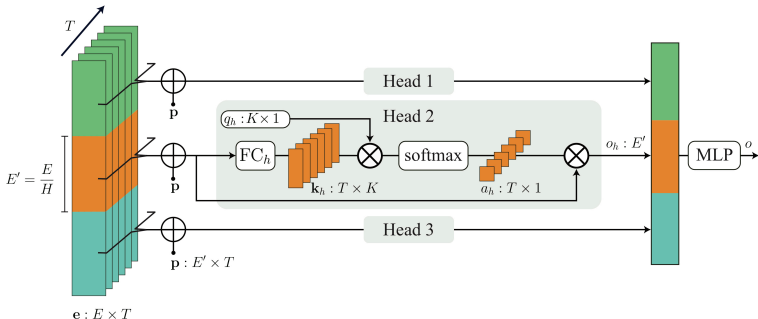
⇒ No need to compute the full attention matrix



# Transformers in remote sensing

In SITS classification, we want to **predict one label** per time series, not a sequence of words as in sentence translation or generation.

⇒ No need to compute the full attention matrix

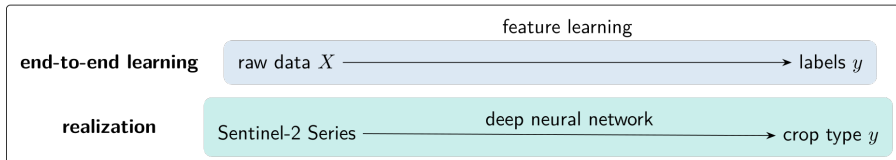


Want to learn more?

- ◇ Join us on **Monday at 1 pm**.
- ◇ Loic Landrieux will discuss *Temporal Attention For SITS* in the special session on *Satellite image time series analysis*.

Let us now move to the second notebook to put into practice deep learning for satellite image time series

## Notebook 2: End-to-End Learning



Link for the notebooks: <https://tinyurl.com/isprs2022>