Deep learning for time series classification

ISPRS Congress 2022 – Tutorial Session

Charlotte Pelletier and Marc Rußwurm

Univ. Bretagne Sud / IRISA and
EPFL-Environmental Computational Science and Earth Observation Laboratory (ECEO)

About us

We are working on time series analysis, mainly classification problems

- using deep-learning and tree-based approaches
- in various contexts: few supervision, mislabelled data, multimodal, etc.

Charlotte Pelletier
Ass. Professor
Univ. Bretagne Sud / IRISA—France
charlotte.pelletier@univ-ubs.fr

Marc Rußwurm
Postdoctoral Researcher
EPFL-ECEO—Switzerland
marc.russwurm@epfl.ch

About you

We would like to know you better and learn about your expertise.

Please go to `menti.com`
Enter the following code to participate to the survey: 1068 0919

link to results

June 5th, 2022, from 09:00 to 12:30

Timeline	Topic
09:00 - 09:05	Opening
09:05 - 09:30	Introduction to Satellite Image Time Series (Lecture)
09:30 - 10:30	Data and Features (Google Colab Notebook 1)
10:30 - 11:00	Break
11:00 - 11:25	Deep learning for SITS (Lecture)
11:25 - 12:25	Deep Learning (Google Colab Notebook 2)
12:25 - 12:30	Closing remarks

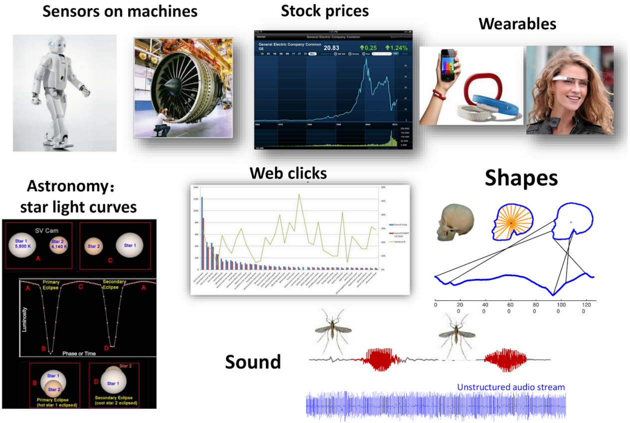
Links to all materials available: *`https://tinyurl.com/isprs2022`*

Content

Time Series

Time series

- describe the evolution of a process over time
- are everywhere and ubiquitous: daily life, medical, food security, financial, environmental...
- increase in quantity



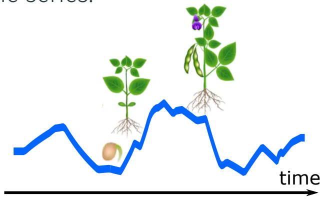
Time Series

Formally, a time series

- is a sequence of values ordered in time
- either univariate or multivariate
- possibly of different lengths

An example univariate time series:

time	value
t1	0.236
t2	0.563
t3	0.748
t4	0.692
...	
tL	0.167



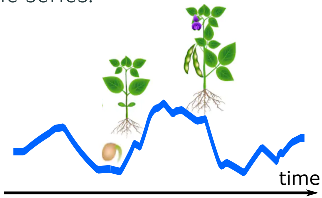
Time Series

Formally, a time series

- is a sequence of values ordered in time
- either univariate or multivariate
- possibly of different lengths

An example univariate time series:

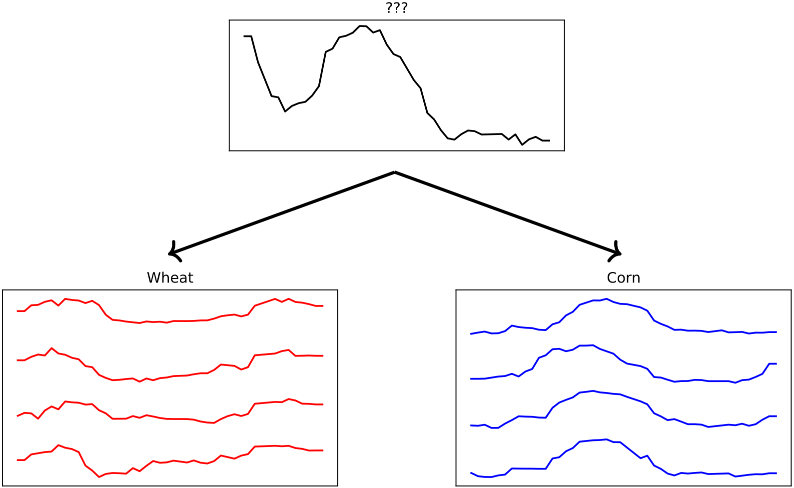
time	value
t1	0.236
t2	0.563
t3	0.748
t4	0.692
...	
tL	0.167



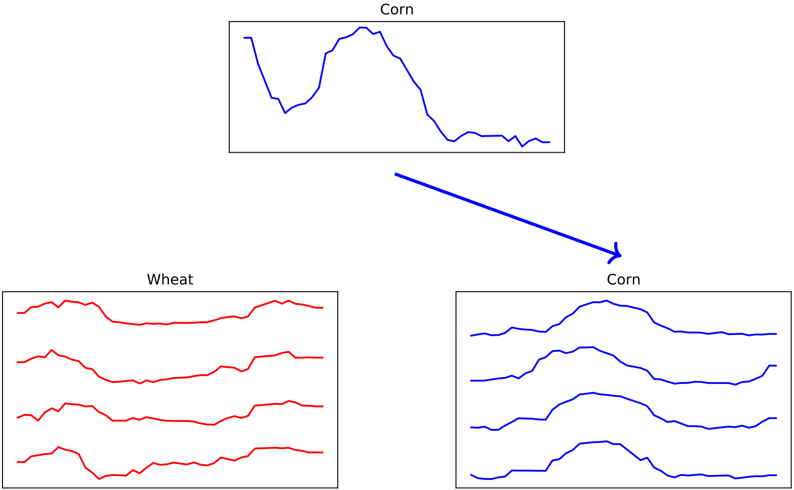
Time series analysis include

- forecasting: predicting future values
- regression: predicting a continuous scalar variable
- retrieval: finding similar time series
- segmentation: dividing a time series into ”homogeneous” subseries
- classification: today’s tutorial

The goal is to associate an unlabelled time series with a class with the help of some labelled time series (supervised learning).



The goal is to associate an unlabelled time series with a class with the help of some labelled time series (supervised learning).



An example satellite image time series

Sentinel-2 images over Brittany, France

Time Series in Remote Sensing

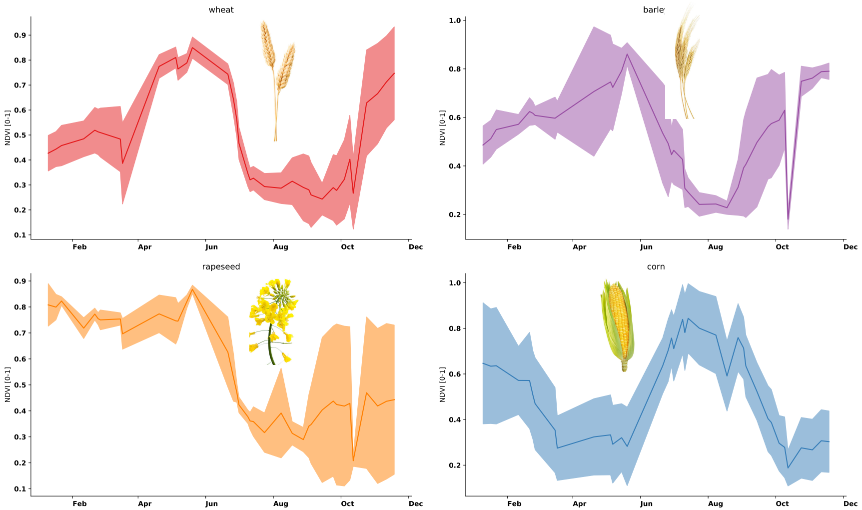
An example satellite image time series

Sentinel-2 images over Brittany, France

Applications

- vegetation monitoring
- landscape changes
- large scale study

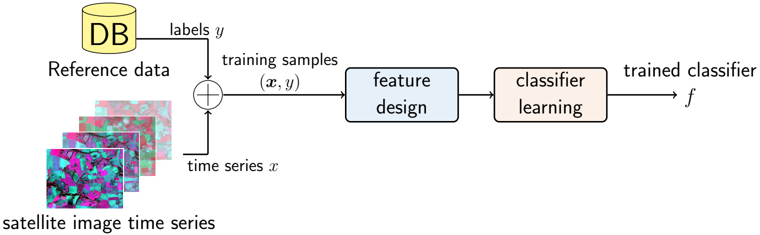
An example application: crop type mapping at large scale



Supervised classification framework

Two main steps:

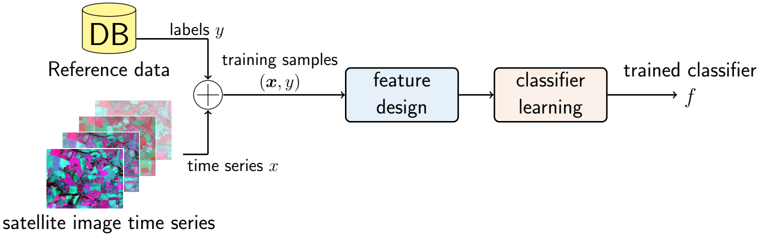
1. Learning a model f such that $f(\mathbf{x}) \approx y$



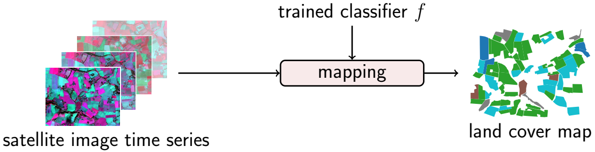
Supervised classification framework

Two main steps:

1. Learning a model f such that $f(\mathbf{x}) \approx y$



2. Using the model f to map the study area



Inputs for learning

Satellite data, *e.g.*, Sentinel-2 images

- Where to download images?
  - Sentinels Scientific Data Hub
  - Copernicus DIAS
  - cloud platforms: GEE, Amazon, Microsoft Planetary Computer
  - THEIA, USGSS, *etc.*
- Common pre-processing steps:
  - coregistration
  - atmospheric correction
  - gapfilling
  - *etc.*

Inputs for learning

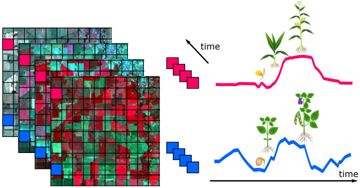
Satellite data, *e.g.*, Sentinel-2 images

- Where to download images?
  - Sentinels Scientific Data Hub
  - Copernicus DIAS
  - cloud platforms: GEE, Amazon, Microsoft Planetary Computer
  - THEIA, USGSS, *etc.*

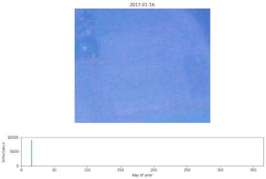
- Common pre-processing steps:
  - coregistration
  - atmospheric correction
  - gapfilling
  - *etc.*

From satellite images to time series

Pixel-based analysis



Object-based analysis, *e.g.*, averaging the reflectance values within an agricultural parcel



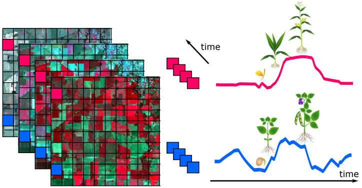
Inputs for learning

Satellite data, *e.g.*, Sentinel-2 images

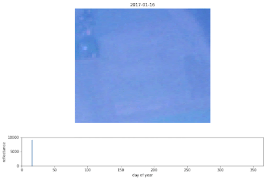
- Where to download images?
  - Sentinels Scientific Data Hub
  - Copernicus DIAS
  - cloud platforms: GEE, Amazon, Microsoft Planetary Computer
  - THEIA, USGSS, *etc.*
- Common pre-processing steps:
  - coregistration
  - atmospheric correction
  - gapfilling
  - *etc.*

From satellite images to time series

Pixel-based analysis



Object-based analysis, *e.g.*, averaging the reflectance values within an agricultural parcel



Reference data

Usually vector files

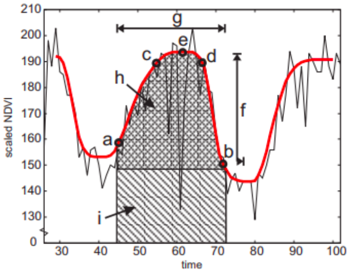
↳ label $y \in \{1, \cdots, \mathcal{C}\}$

- photo-interpretation
- field campaigns
- governmental data (*e.g.* Corine Land Cover)
- collaborative data (*e.g.* Open Street Map)

From time series to feature vectors

Feature design is a key step when using traditional machine learning algorithms
- flatten reflectance time series
- compute spectral features, *e.g.*, Normalized Difference Vegetation Index
- extract temporal features: statistical and phenological features
- and even compute spatial features, *e.g.* Haralick or attribute profiles

TIMESAT example: extraction of key phenological stages [1]



[1] Jönsson, P., & Eklundh, L. (2004). TIMESAT—a program for analyzing time-series of satellite sensor data. *Computers & Geosciences*, 30(8), 833-845.

Evaluation

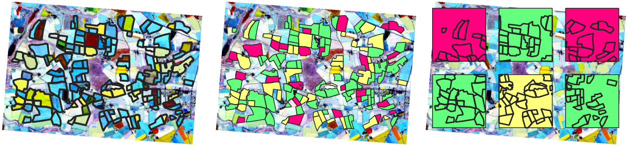
Types of evaluation: quantitative (accuracy, computational complexity, explainability), visual, evaluation on a downstream task

Types of evaluation: quantitative (accuracy, computational complexity, explainability), visual, evaluation on a downstream task

Quantitative evaluation: split the labeled data into 3 spatially independent sets

- a train set to learn the model's parameters
- a validation set to tune the hyperparameter values of the model
- a test set to obtain a non-biased estimation of the model's performance

Labeled data | Polygon-split | Grid-split



Evaluation

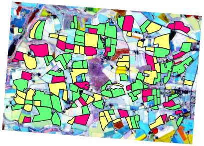
Types of evaluation: quantitative (accuracy, computational complexity, explainability), visual, evaluation on a downstream task

Quantitative evaluation: split the labeled data into 3 spatially independent sets

- a train set to learn the model's parameters
- a validation set to tune the hyperparameter values of the model
- a test set to obtain a non-biased estimation of the model's performance

Labeled data

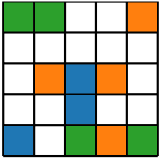
Polygon-split



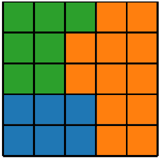
Grid-split

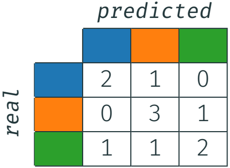
The confusion matrix:

Ground truth



Dense predictions





Practical activity

How to implement this framework?

- develop your own Python code
- use dedicated libraries, *e.g.*, snap, OTB
- use existing frameworks, *e.g.*, ***`iota2`*** or R-SITS package

Practical activity

How to implement this framework?

- ◇ develop your own Python code
- ◇ use dedicated libraries, *e.g.*, snap, OTB
- ◇ use existing frameworks, *e.g.*, ***iota2*** or R-SITS package

Let us now move to our first practical activity!

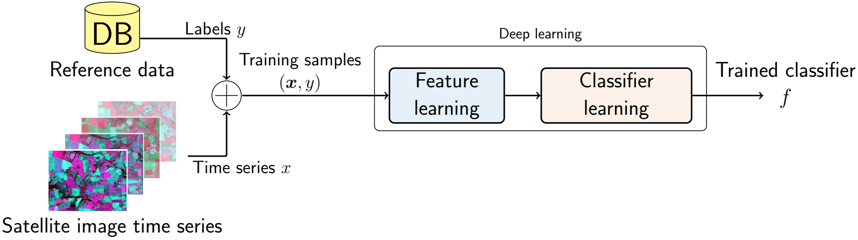
Notebook 1: Feature Engineering

		pre-processing	feature design	classification
feat. engineering	raw data X	→ processed data	→ features	→ labels y
realization	Sentinel-2 Series	→ cloud filtering	→ NDVI-statistics	→ crop type y

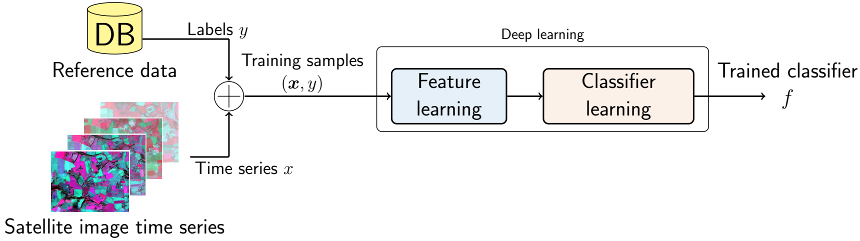
Link for the notebooks: ***https://tinyurl.com/isprs2022***

Content

Features are extracted automatically in deep learning



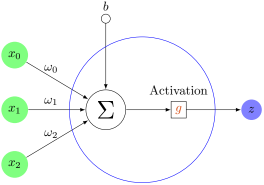
Features are extracted automatically in deep learning



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

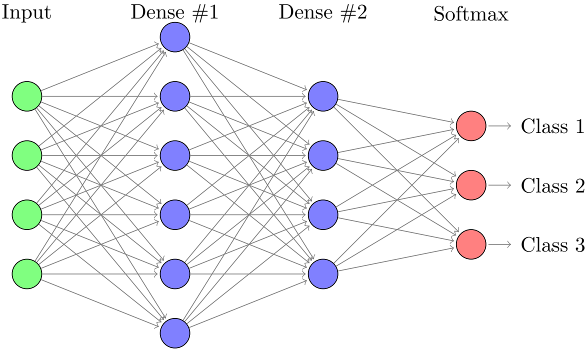
- the type of network,
- the number of layers (depth)
- the number of units per layer (width)
- the learning strategy (optimizer, learning rate)
- *etc.*

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



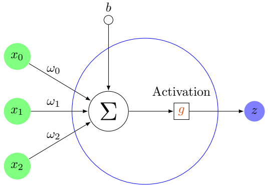
$$y = g\left(\sum_i \omega_i \cdot x_i + b\right)$$

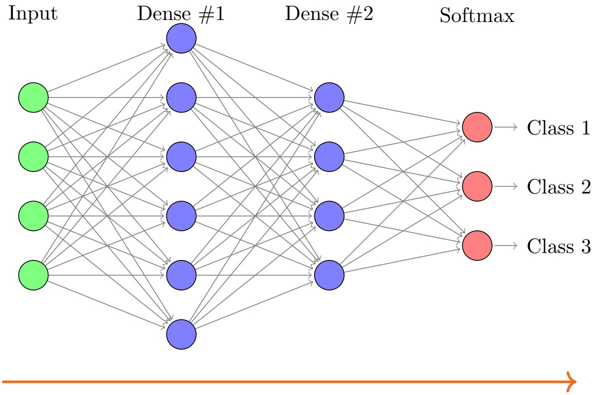
$$y = g(\mathbf{w}^T \mathbf{x} + b)$$



How to train a network?

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





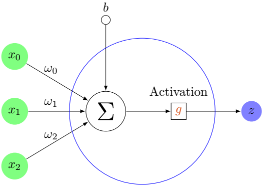
$$y = g\left(\sum_i \omega_i \cdot x_i + b\right)$$

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

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

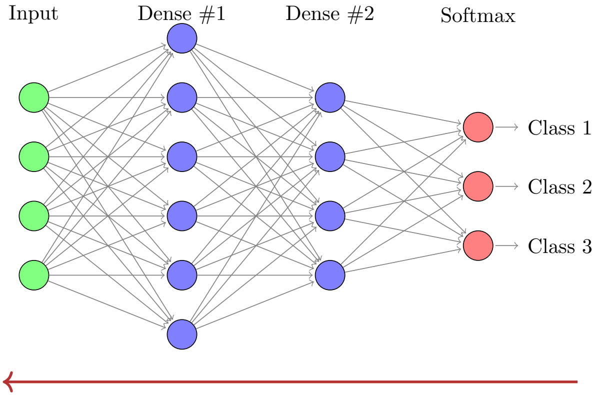
How to train a network?

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



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

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

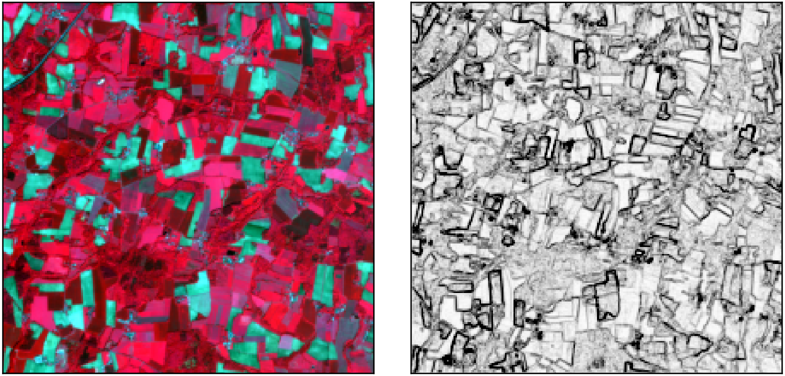


2. **Backward** step: update the parameter values through gradient descent

A. Convolutional Neural Networks

Convolution for images

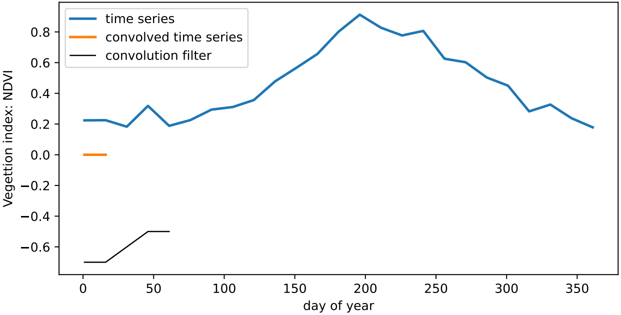
The convolution is a common image processing techniques for images and signals.



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

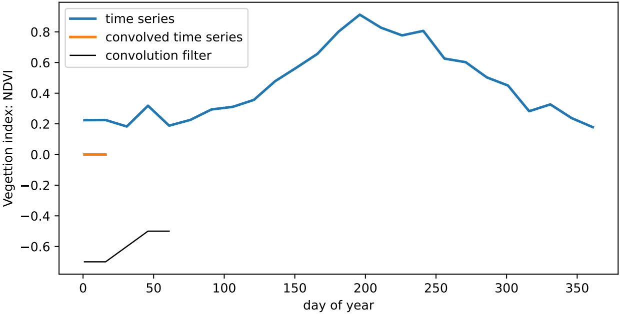
How it works?

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



How it works?

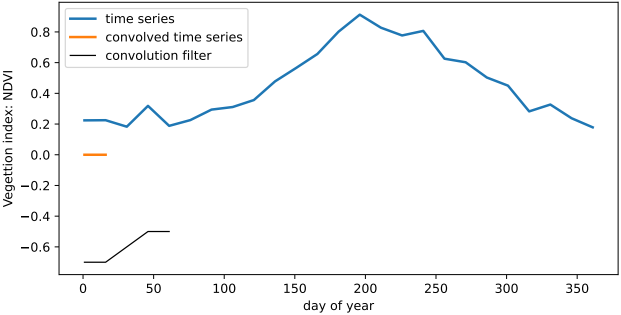
- The result of applying an edge detection on a time series:



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

How it works?

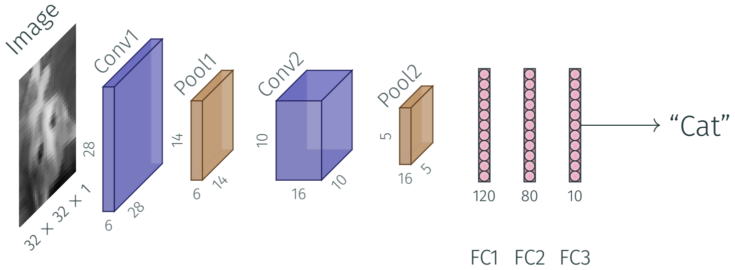
- The result of applying an edge detection on a time series:



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

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

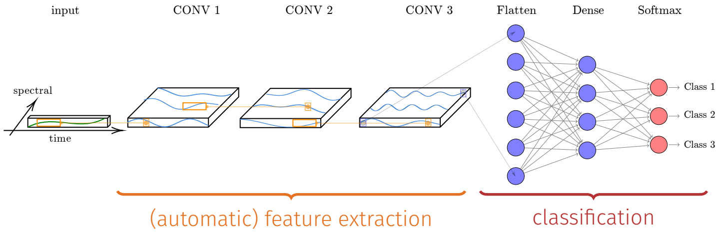


LeNet architecture [1]

[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:

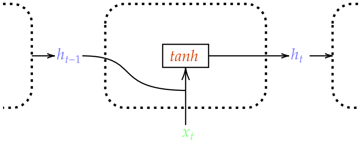
- able to explicitly consider the temporal correlation of the data
- state-of-the-art architectures for forecasting tasks

Recurrent Neural Networks (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



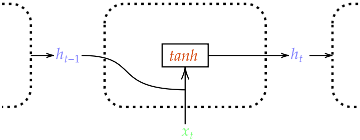
$$h_t = \tanh(W_x x_t + W_h h_{t-1} + b_h)$$

Recurrent Neural Networks (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

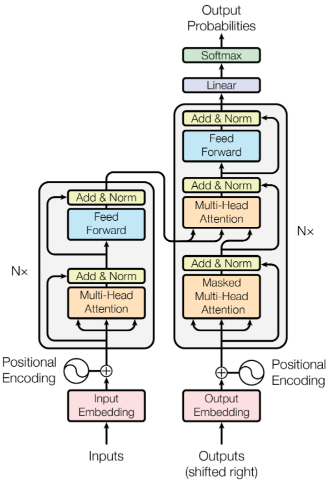
but they are slow to train due to backpropagation through time, and fail to extract long temporal dependencies

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

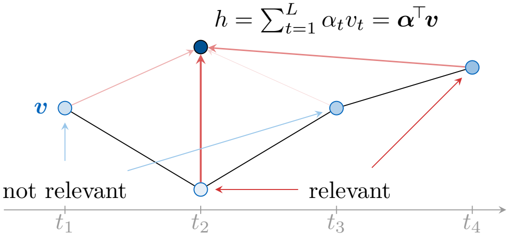


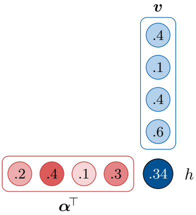
[1] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

[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)*

Objective: focusing on the relevant elements of the time series

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



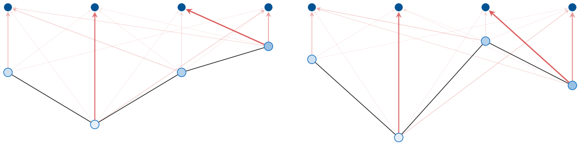


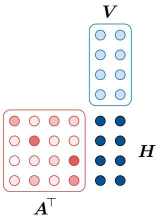
$$\boldsymbol{h} = \text{Attention}(\boldsymbol{\alpha}, \boldsymbol{v}) = \boldsymbol{\alpha}^\top \boldsymbol{v} = \sum_{t=1}^{L} \alpha_t v_t,$$

$$\boldsymbol{\alpha} \in [0, 1]^L, \boldsymbol{v} \in \mathbb{R}^L$$

Objective: focusing on the relevant elements of the time series

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





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

where D_v is the dimension of the time series $\boldsymbol{v}$.

How to compute the attention scores?

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

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

- 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 sim.
- The softmax normalization $\frac{\exp(\cdot)}{\sum \exp(\cdot)}$ ensures that $\sum_{t=1}^{L} \alpha_t = 1$.

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

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

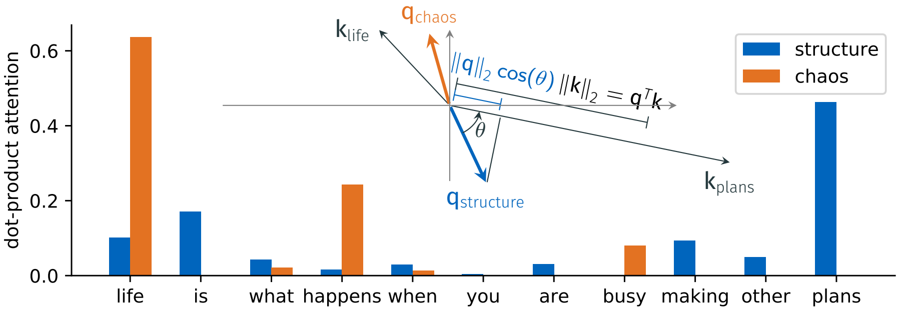
- The query $\mathbf{q}$ provides a semantic **context** that is compared to a key $\mathbf{k}_t$ for each sequence element t using a similarity measure *sim*.
- The softmax normalization $\frac{\exp(\cdot)}{\sum \exp(\cdot)}$ ensures that $\sum_{t=1}^{L} \alpha_t = 1$.

A variety of similarity measures:

cosine distance [1]	$sim(\mathbf{q}, \mathbf{k}) = \frac{\mathbf{q}^\top \mathbf{k}}{\lVert\mathbf{q}\rVert_2 \lVert\mathbf{k}\rVert_2}$
dot-product [2]	$sim(\mathbf{q}, \mathbf{k}) = \mathbf{q}^\top \mathbf{k}$
scaled dot-product [3]	$sim(\mathbf{q}, \mathbf{k}) = \frac{\mathbf{q}^\top \mathbf{k}}{\sqrt{D_k}}$

Text example

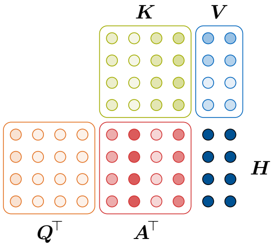
- Each word is embedded into a 300-dimensional semantic Glove Vector, *e.g.*, $\mathbf{e}_{\text{structure}} = E(\texttt{"structure"}) \in \mathbb{R}^{300}$.
- Embeddings of two query words *"structure"* and *"chaos"* are compared to a sentence of keys *"life is what happens when you are busy making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



Core idea:

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

How to determine the values, keys, and queries?

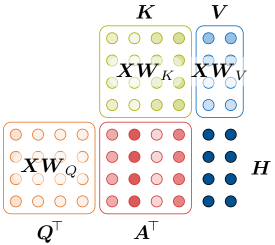


$$\mathrm{Attention}(K, Q, V) = \overbrace{\mathrm{softmax}\left(Q^\top K\right)}^{A^\top} V,$$

$$V \in \mathbb{R}^{L \times D_v}, Q, K \in \mathbb{R}^{D_k \times L}, A \in \mathbb{R}^{L \times L}$$

How to determine the values, keys, and queries?

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



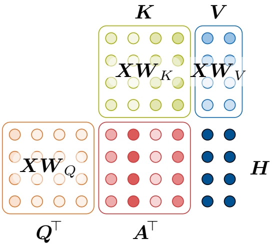
$$\text{Attention}(K, Q, V) = \overbrace{\text{softmax}\left(Q^\top K\right)}^{A^\top} V,$$

$$V \in \mathbb{R}^{L \times D_V}, Q, K \in \mathbb{R}^{D_k \times L}, A \in \mathbb{R}^{L \times L}$$

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

How to determine the values, keys, and queries?

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

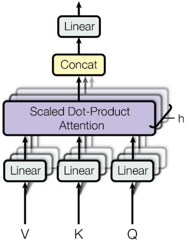


$$\text{Attention}(K, Q, V) = \overbrace{\text{softmax}\left(Q^\top K\right)}^{A^\top} V,$$

$$V \in \mathbb{R}^{L \times D_V}, Q, K \in \mathbb{R}^{D_k \times L}, A \in \mathbb{R}^{L \times L}$$

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

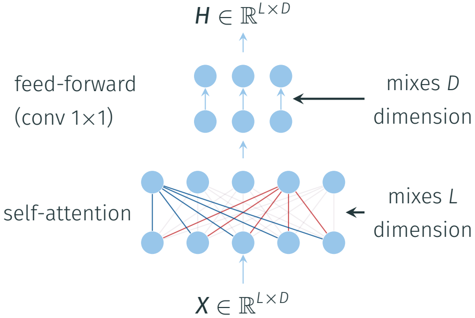
Self-attention is usually applied in parallel heads, which is known as **multi-head attention**.

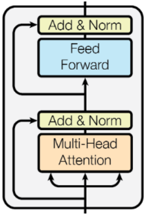


The Transformers are composed of **encoder blocks** that map a D-dimensional input times series $\boldsymbol{X}$ of length L into a higher-level representation $\boldsymbol{H} \in \mathbb{R}^{L \times D}$. Each block is composed of:

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

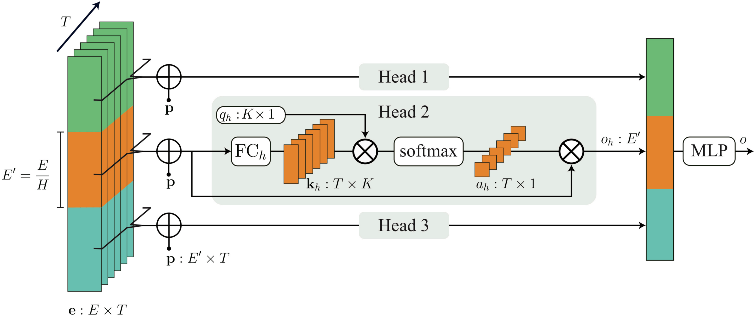
A block also includes skip connections and normalization.





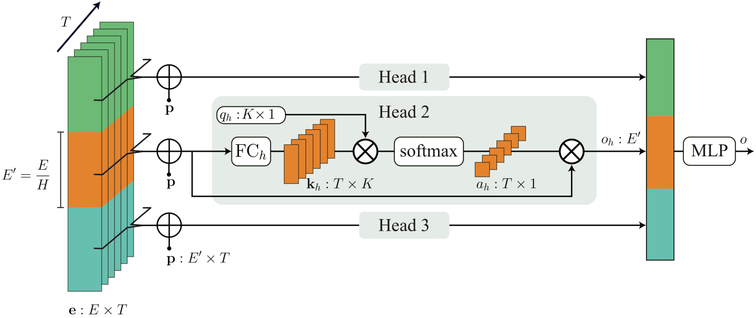
In SITS classification, we want to **predict one label** per time series, not a sequence of words as in sentence translation or generation.

$\Rightarrow$ No need to compute the full attention matrix



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

$\Rightarrow$ No need to compute the full attention matrix



Want to learn more?

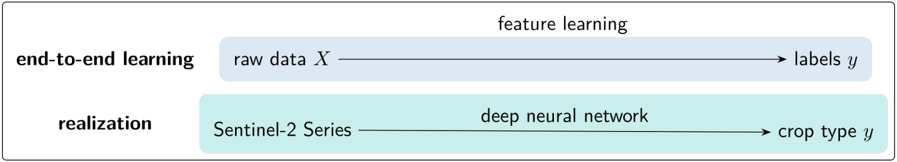
- Join us on **Monday** at **1 pm**.
- Loic Landrieux will discuss *Temporal Attention For SITS* in the special session on *Satellite image time series analysis*.

[1] Sainte Fare Garnot, V., & Landrieu, L. (2020, September). Lightweight temporal self-attention for classifying satellite images time series. In *International Workshop on Advanced Analytics and Learning on Temporal Data* (pp. 171-181). Springer, Cham.

Notebooks

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`*

Content

Take home messages

- deep learning is the **data-driven extension** (second notebook) of previously **hand-defined features** (first notebook)

Take home messages

- deep learning is the **data-driven extension** (second notebook) of previously **hand-defined features** (first notebook)
- deep learning requires **far more labeled data** to avoid overfitting.

Take home messages

- deep learning is the **data-driven extension** (second notebook) of previously **hand-defined features** (first notebook)
- deep learning requires **far more labeled data** to avoid overfitting.
- do **not under-estimate the pre-processing** steps: good pre-processing will lead to better performance

Take home messages

- deep learning is the **data-driven extension** (second notebook) of previously **hand-defined features** (first notebook)
- deep learning requires **far more labeled data** to avoid overfitting.
- do **not under-estimate the pre-processing** steps: good pre-processing will lead to better performance
- check existing classification networks for SITS (*e.g.*, pre-trained weights available in the BreizhCrops package)

Take home messages

- deep learning is the **data-driven extension** (second notebook) of previously **hand-defined features** (first notebook)
- deep learning requires **far more labeled data** to avoid overfitting.
- do **not under-estimate the pre-processing** steps: good pre-processing will lead to better performance
- check existing classification networks for SITS (*e.g.*, pre-trained weights available in the BreizhCrops package)
- Test your own algorithm using existing datasets: BreizhCrops, DENETHOR, TimeSen2Crop, EuroCrops

See you at the conference!

Special Session @ISPRS:

Analysis of satellite image time series **tomorrow** (June 6th) at **1 pm.**

Special Issue in *Remote Sensing* (until Nov. 22):

Advances in Deep Learning Techniques for the Analysis of Remote Sensing Time Series

Closing remarks

See you at the conference!

Special Session @ISPRS:
Analysis of satellite image time series **tomorrow** (June 6th) at **1 pm.**

Special Issue in *Remote Sensing* (until Nov. 22):
Advances in Deep Learning Techniques for the Analysis of Remote Sensing Time Series

---

If you liked this tutorial, (re)tweet under #TimeSeriesISPRS
and follow us on twitter @marccoru, @CharlottePlltr

all material remains available under https://dl4sits.github.io/isprs2022/tutorial/
(and tinyurl.com/isprs2022)