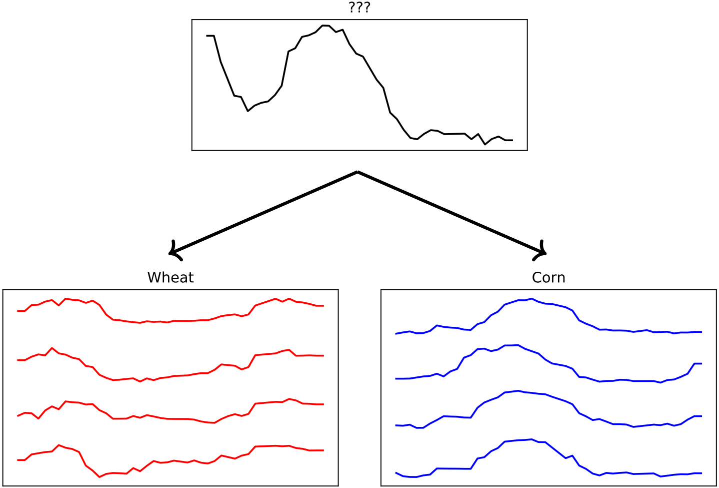
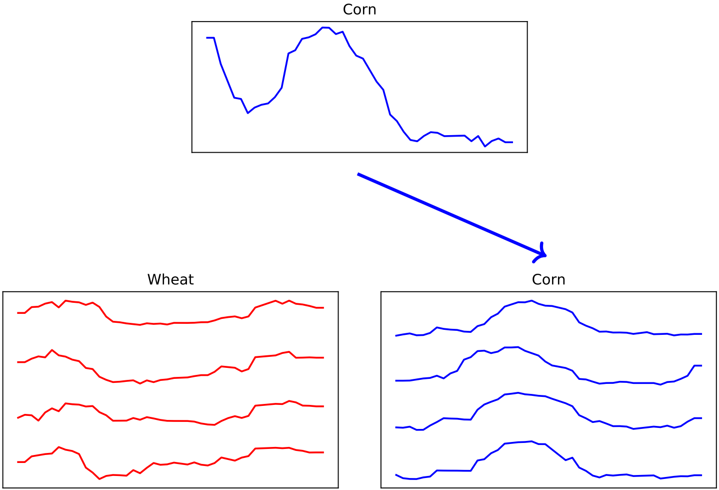
Content

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



Time Series Classification

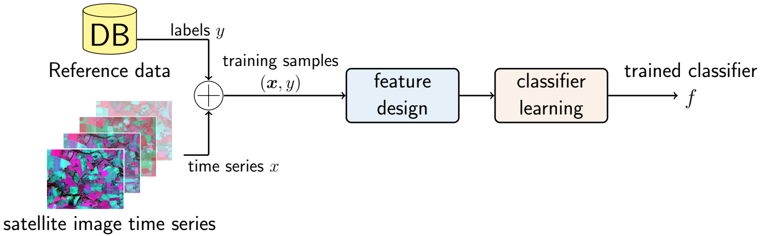
The goal of time-series **classification** is to associate an unlabelled time series with a class with the help of some labelled time series (supervised learning).



Supervised classification framework

How does it work in practice?

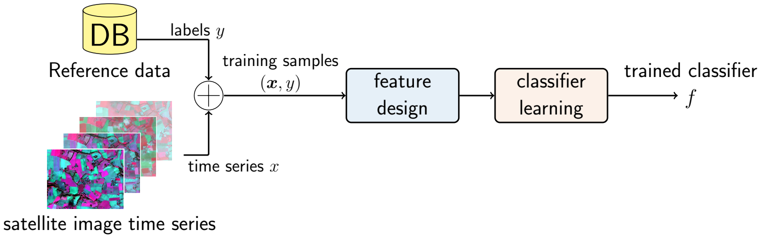
1. Learning a model f such that $f(\mathbf{x}) \approx y$



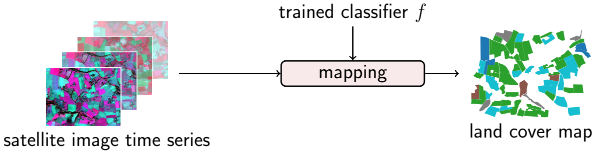
Supervised classification framework

How does it work in practice?

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



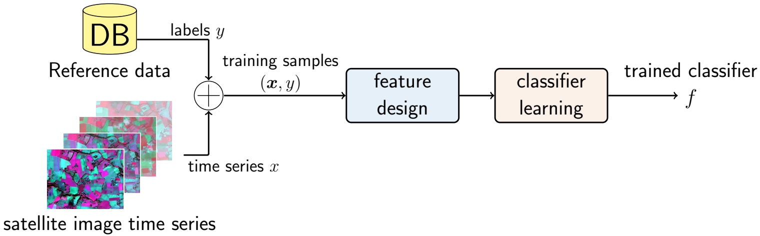
2. Using the model f to map the study area



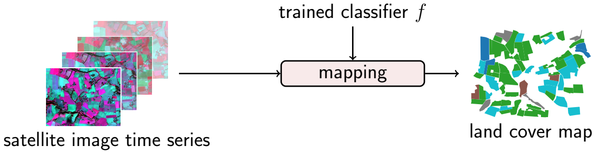
Supervised classification framework

How does it work in practice?

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



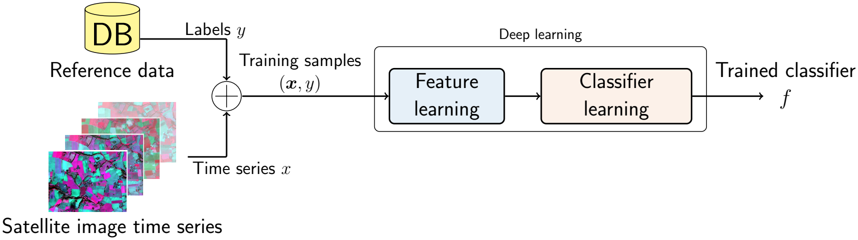
2. Using the model f to map the study area



This framework requires the extraction of discriminative and relevant features.

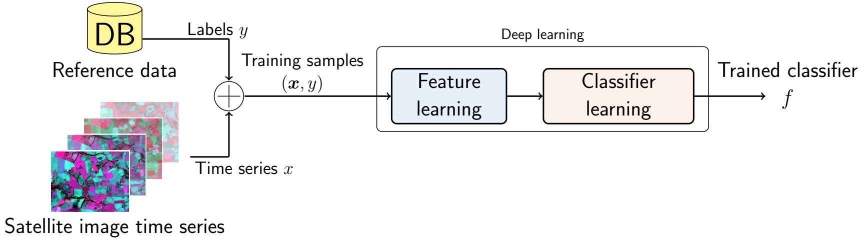
From Machine Learning to Deep Learning

Features are extracted **automatically** in deep learning



From Machine Learning to Deep Learning

Features are extracted **automatically** in deep learning

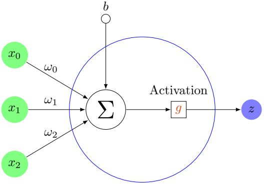


Architecture design is the new feature engineering! One needs to choose

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

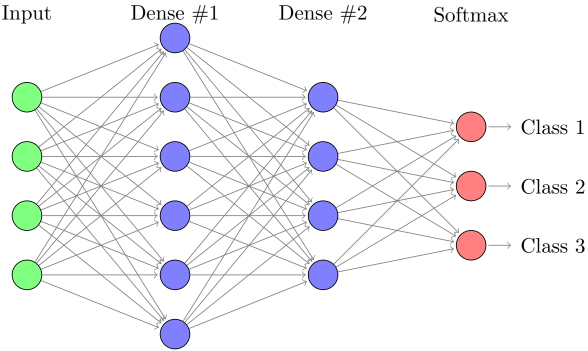
How to train a network?

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



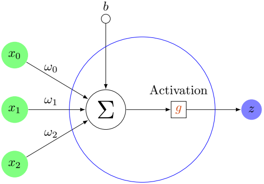
$$y = g\left(\sum_i \omega_i \cdot x_i + b\right)$$

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



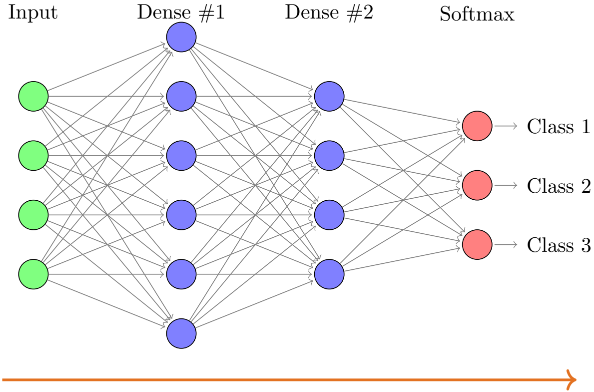
How to train a network?

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



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

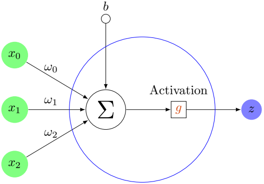
$$y = g(\mathbf{w}^T \mathbf{x} + b)$$



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

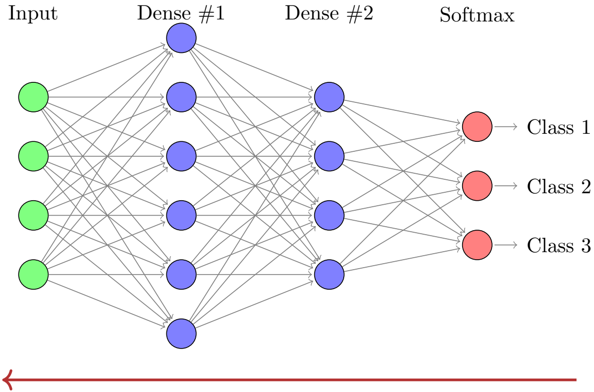
How to train a network?

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



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

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



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

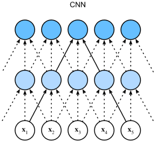
Deep Learning for SITS

Extensive research using deep learning techniques to **exploit spatio-temporal dependencies** in SITS

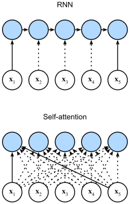
◇

We reviewed the use of four main architectures [1], including

- ◇ convolutional neural networks (CNN)
- ◇ recurrent networks
- ◇ attention-based approaches
- ◇ graph-based techniques



Source



Source

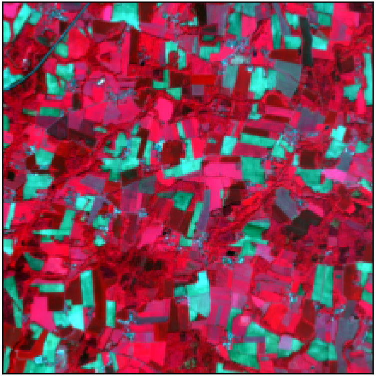
[1] Miller, L., Pelletier, C., & Webb, G. I. (2024). Deep Learning for Satellite Image Time-Series Analysis: A review. *IEEE Geoscience and Remote Sensing Magazine.*

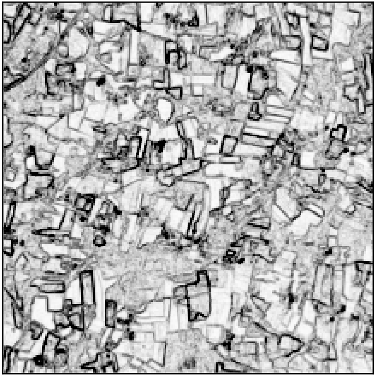
[2] Moskolaï, W. R., Abdou, W., Dipanda, A., & Kolyang. (2021). Application of deep learning architectures for satellite image time series prediction: A review. *Remote Sensing*, 13(23), 4822.

A. Convolutional Neural Networks

Convolution for images

Convolution is a common image-processing technique for images and signals.

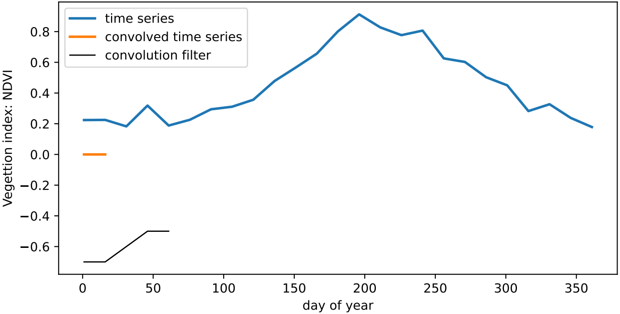




The result of applying a convolution filter (here an edge detection filter) on a Sentinel-2 image.

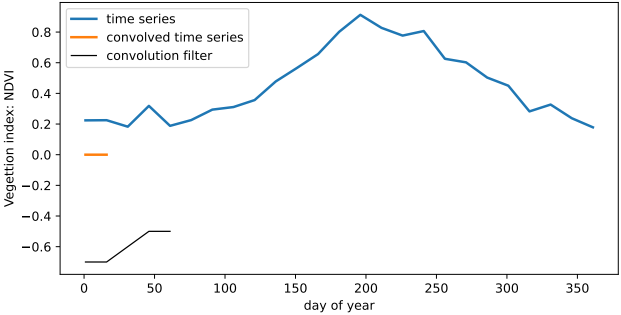
How it works?

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



How it works?

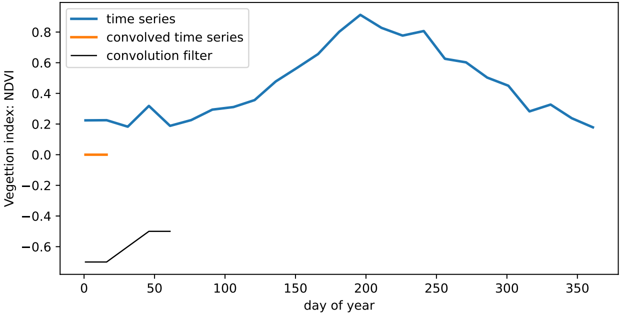
- The result of applying an edge detection on a time series:



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

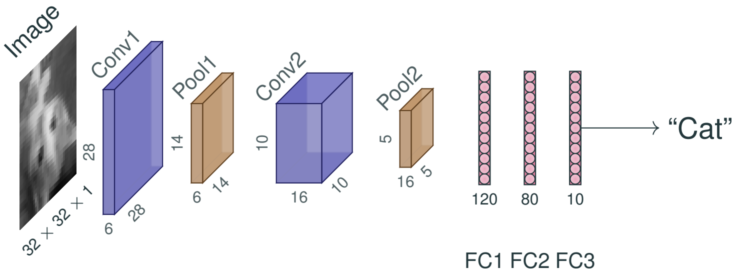
How it works?

- 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

- **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



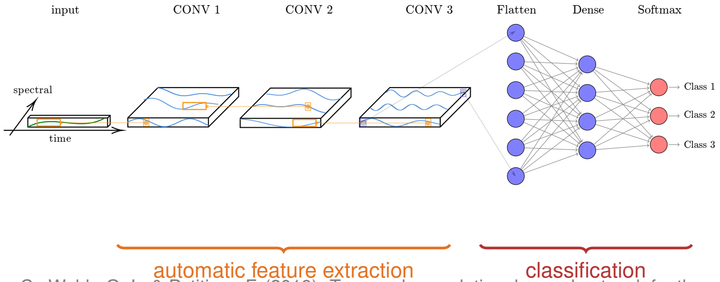
LeNet architecture [1]

[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.

CNN in remote sensing

Temporal Convolutional Neural Networks (TempCNN) [1] (and also [2])

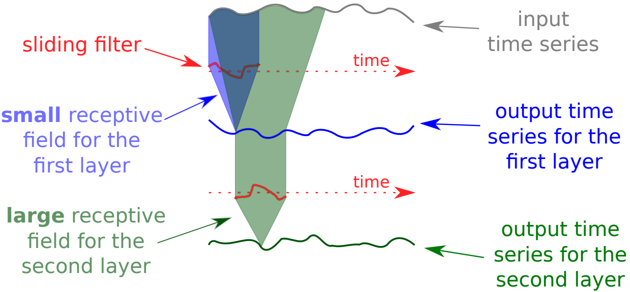
- small architecture, especially when adding a global average pooling after the convolution layers
- requires regular-spaced time series



[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.

[2] Zhong, L., Hu, L. & Zhou, H. (2019). Deep learning based multi-temporal crop classification. *Remote Sensing of Environment*, 221, 430-443.

Receptive field illustration for two (Temp)CNN layers



The **effective receptive field** is the part of the input that affects a given neuron indirectly through previous convolutional layers. It grows linearly with depth.

B. Recurrent Neural Networks

Recurrent Neural Networks

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

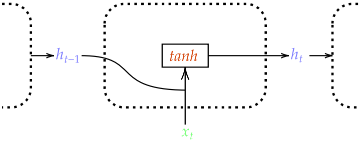
- able to explicitly consider the **temporal correlation** of the data
- state-of-the-art architectures for forecasting tasks

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



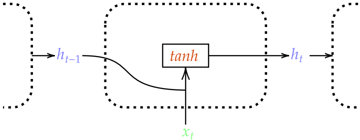
$$h_t = \tanh(W_x x_t + W_h h_{t-1} + b_h)$$

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

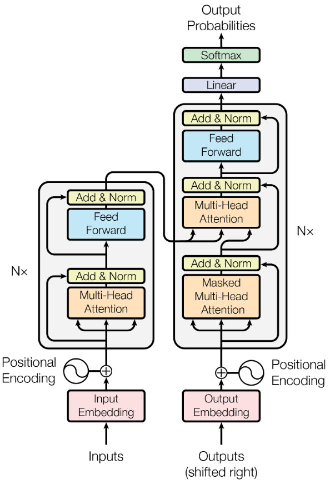
but they are slow to train due to backpropagation through time, and fail to extract long temporal dependencies

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



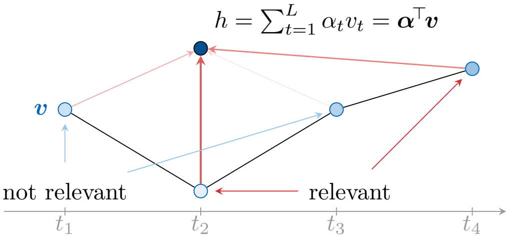
[1] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

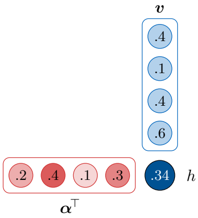
[2] Waswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L & Polosukhin, I. (2017). Attention is all you need. In *Conference on Neural Information Processing Systems (NIPS)*

Attention mechanism

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** $\boldsymbol{\alpha} \in \mathbb{R}^L$.





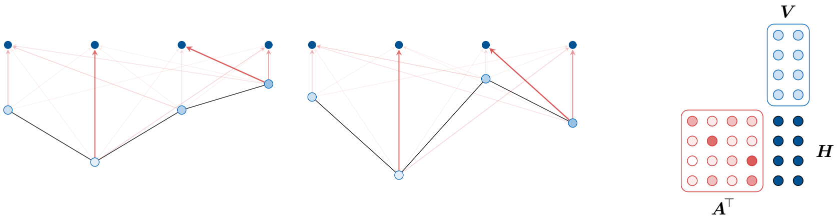
$$\boldsymbol{h} = \text{Attention}(\boldsymbol{\alpha}, \boldsymbol{v}) = \boldsymbol{\alpha}^\top \boldsymbol{v} = \sum_{t=1}^{L} \alpha_t v_t,$$

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

Attention mechanism

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** $\alpha \in \mathbb{R}^L$.



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

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 $\sum_{t=1}^{L} \alpha_t = 1$.

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 $\sum_{t=1}^{L} \alpha_t = 1$.

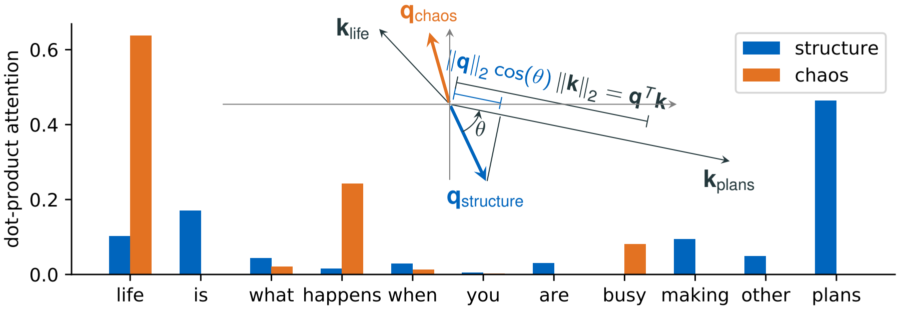
A variety of similarity measures:

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

Dot-Product Attention on Word Embeddings

Text example

- Each word is embedded into a 300-dimensional semantic Glove Vector, *e.g.*, $\mathbf{e}_{\text{structure}} = E(\texttt{"structure"}) \in \mathbb{R}^{300}$.
- 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"`.

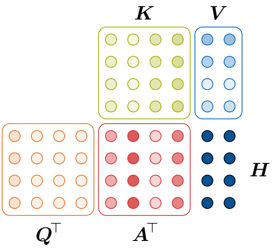


Core idea:

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

Self-attention

How to determine the values, keys, and queries?



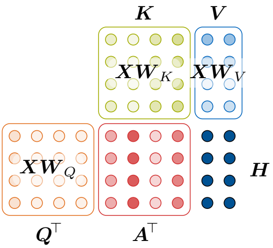
$$\text{Attention}(\boldsymbol{K}, \boldsymbol{Q}, \boldsymbol{V}) = \overbrace{\text{softmax}\left(\boldsymbol{Q}^\top \boldsymbol{K}\right)}^{\boldsymbol{A}^\top} \boldsymbol{V},$$

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

Self-attention

How to determine the values, keys, and queries?

- the self-attention mechanism uses linear projection of the input sequences $\boldsymbol{X}$



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

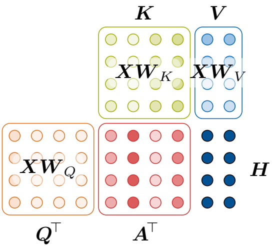
$$\boldsymbol{V} \in \mathbb{R}^{L \times D_v}, \boldsymbol{Q}, \boldsymbol{K} \in \mathbb{R}^{D_k \times L}, \boldsymbol{A} \in \mathbb{R}^{L \times L}$$

$$\begin{aligned}\text{Self-Attention}_{\boldsymbol{W}}(\boldsymbol{X}) &= \text{Attention}(\boldsymbol{X}\boldsymbol{W}_K, \boldsymbol{X}\boldsymbol{W}_Q, \boldsymbol{X}\boldsymbol{W}_V) \\ &= \text{softmax}\Big((\boldsymbol{X}\boldsymbol{W}_Q)^T (\boldsymbol{X}\boldsymbol{W}_K) \Big) \Big(\boldsymbol{X}\boldsymbol{W}_V\Big)\end{aligned}$$

Self-attention

How to determine the values, keys, and queries?

- the self-attention mechanism uses linear projection of the input sequences $\boldsymbol{X}$

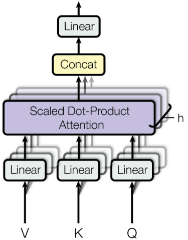


$$\text{Attention}(\boldsymbol{K}, \boldsymbol{Q}, \boldsymbol{V}) = \overbrace{\text{softmax}\left(\boldsymbol{Q}^\top \boldsymbol{K}\right)}^{\boldsymbol{A}^\top} \boldsymbol{V},$$

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

$$\begin{aligned} \text{Self-Attention}_{\boldsymbol{W}}(\boldsymbol{X}) &= \text{Attention}(\boldsymbol{XW}_K, \boldsymbol{XW}_Q, \boldsymbol{XW}_V) \\ &= \text{softmax}\left((\boldsymbol{XW}_Q)^T (\boldsymbol{XW}_K)\right)\left(\boldsymbol{XW}_V\right) \end{aligned}$$

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

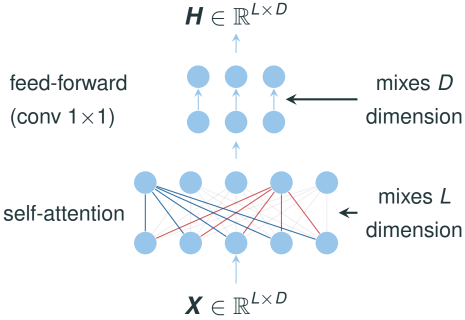


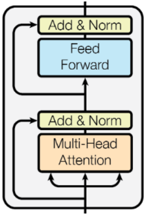
Transformers encoder

The Transformers are composed of **encoder blocks** that map a D-dimensional input times series $\boldsymbol{X}$ 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.

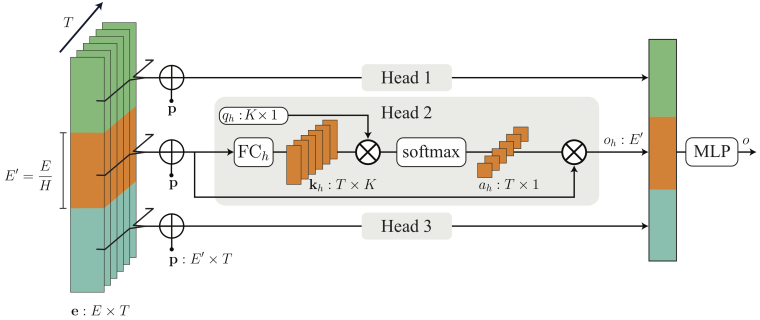




Transformers in remote sensing

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

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



[1] Sainte Fare Garnot, V., & Landrieu, L. (2020). 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.

Conclusions

Deep learning has a high potential and impact in real remote-sensing applications

- various models inspired by natural language processing (NLP) and computer vision
- specific SITS-architectures
- time-first, space-later strategies

Conclusions

Deep learning has a high potential and impact in real remote-sensing applications

- ◇ various models inspired by natural language processing (NLP) and computer vision
- ◇ specific SITS-architectures
- ◇ time-first, space-later strategies

... with current limitations:

- ◇ use on very high spatial resolution dense SITS (below 1-meter), at continental and global scales, still requires the development of efficient techniques
- ◇ small volumes of training sets, especially for the temporal dimension, requires new architectures and learning paradigms
- ◇ difficult to adapt to different climatic and anthropic regions, especially when marked by seasonal effects.

[1] Rolf, E., Klemmer, K., Robinson, C., & Kerner, H. (2024). Mission Critical–Satellite Data is a Distinct Modality in Machine Learning. arXiv preprint arXiv:2402.01444.

Practical Session

Let us now move to the **practical session** to put into practice break detection and deep learning for satellite image time series

Link for the notebooks: **https://dl4sits.github.io/igarss2024/tutorial/**