Hyperspectral remote sensing data analysis using deep learning

Nicolas Audebert

Séminaire MLDL @ LAM – 1 Feb. 2021

Multispectral imaging and why we use it

Multispectral imaging for remote sensing

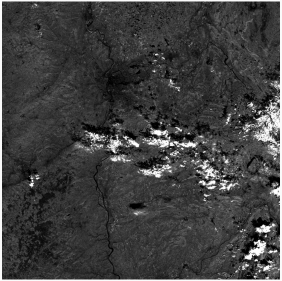
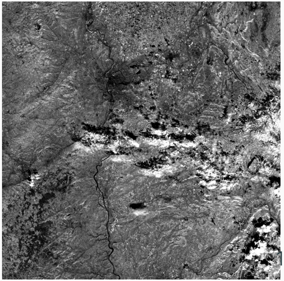
Remote sensing

Airborne or spaceborne sensors, e.g. cameras/radars mounted on planes and satellites.

Multispectral imaging

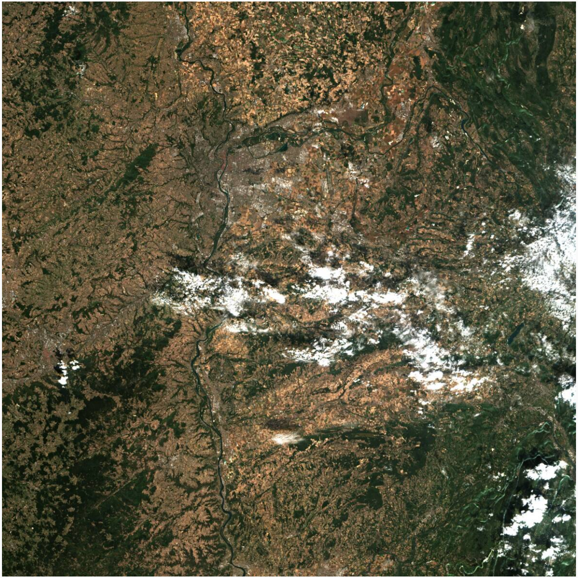
Visible wavelengths are not always the most interesting.

ESA/**Sentinel-2** constellation uses a camera that "sees" through 12 wavelengths carefully chosen.

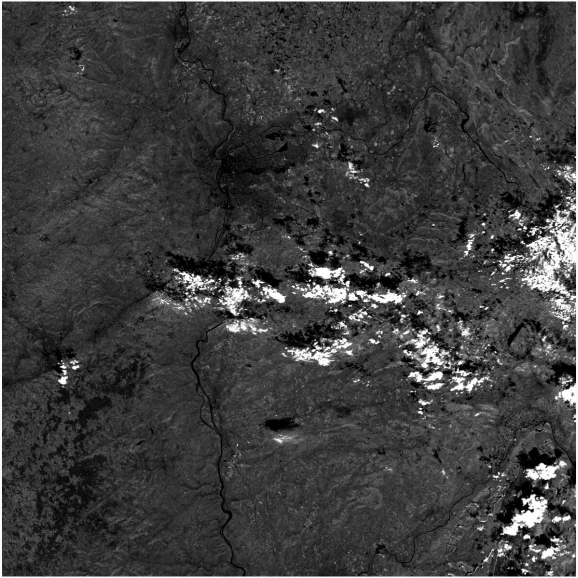
Seeing the invisible

Visible light



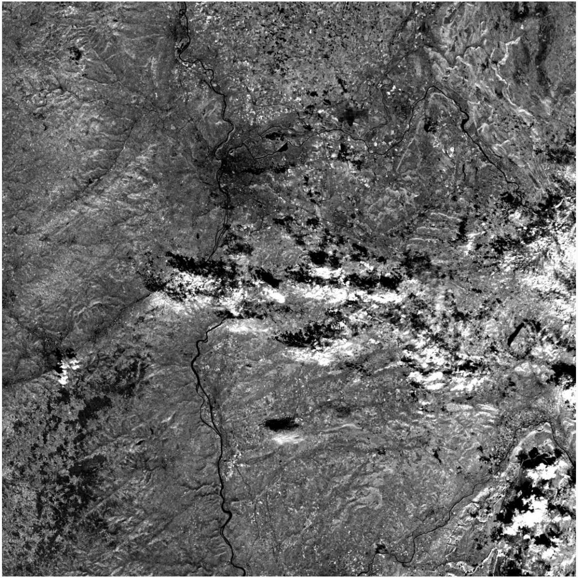
Seeing the invisible

Water vapour ($\simeq$ clouds)

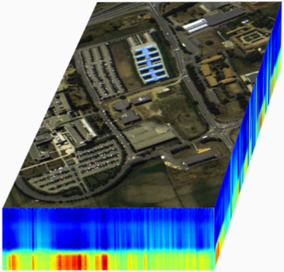


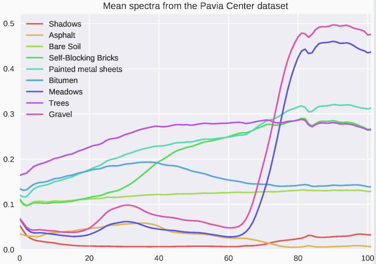
Seeing the invisible

Vegetation "red edge"



Why is multi/hyperspectral imaging useful?





What is an hyperspectral image?

Hyperspectral cameras acquire light intensities for hundreds of wavelengths $\rightarrow$ one pixel = one spectrum $\rightarrow$ see invisible things for the human eye

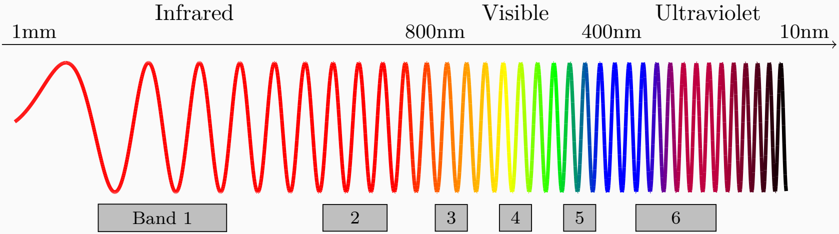
Why do we use hyperspectral imaging?

Different materials have different **spectral signatures** that can be measured through an hyperspectral sensor.

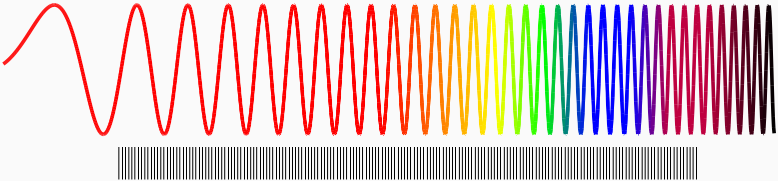
$\rightarrow$ huge discriminative power for land cover mapping, health vegetation monitoring, plastic recycling, etc.

Multi or hyper? Spectral resolution matters

Multispectral : a few bands with irregular widths



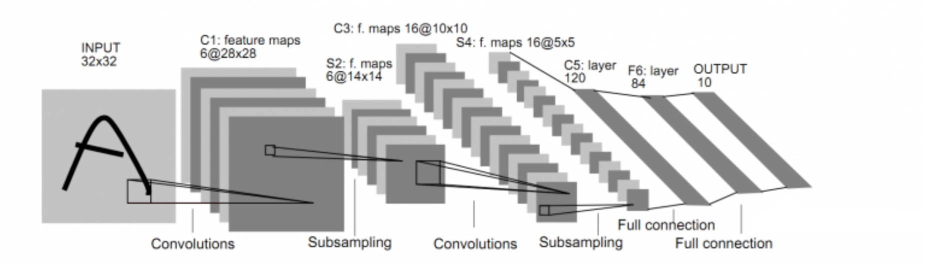
Hyperspectral : several dozen bands of identical width



Basics of convolutional neural networks

(Convolutional) neural networks

Stack of **optimizable** convolutional kernels learnt via **gradient descent**
$\rightarrow$ similar to wavelet decomposition but using a learnt kernel set



Gradient-based learning applied to document recognition, LeCun et al., 1998

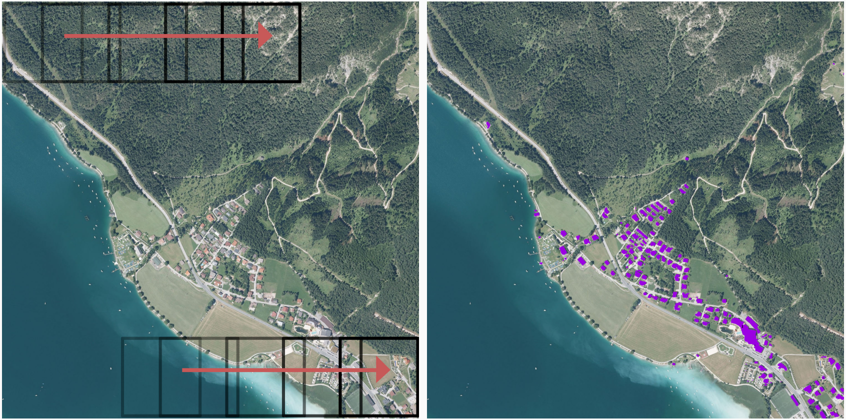
Objective function

Minimize the "error" on some samples (the *train set*), e.g. :

- **Regress** some quantity (% of water, building height...)
- **Classify** the sample into some categories (forest, building, crop...).

An example : extracting buildings from aerial images

Objective : for every pixel, predict if it belongs to a building or not (*binary classification*)



Slide a window on the image, for each patch, apply the model on the relevant pixels.

Training a neural network

- f_θ be the NN function with θ the parameters of the NN (its *weights*),
- X the training samples and y their labels,
- $\mathcal{L}$ the loss function (quantifies the errors between NN prediction and the truth).

Examples of $\mathcal{L}$: MSE, Kullback-Leibler div., anything differentiable...

Gradient descent : repeat until bored or $\mathcal{L}(y, \hat{y})$ stops decreasing

1. For step i, select at random some $x_i \in X$.
2. Compute NN predictions $\hat{y} = f_\theta(x_i)$.
3. Compute the "loss" $\mathcal{L}(y, \hat{y})$.
4. Compute gradient of loss w.r.t. the weights : $\nabla_\theta(\mathcal{L}) \leftarrow$ **we use the chain rule to "backpropagate" the gradient value into the network**
5. Update the weights : $\theta' \leftarrow \theta - \alpha \nabla_\theta(\mathcal{L})$
   α controls the "*learning rate*" (how much we update the weights)

Training a neural network

- f_θ be the NN function with θ the parameters of the NN (its *weights*),
- X the training samples and y their labels,
- $\mathcal{L}$ the loss function (quantifies the errors between NN prediction and the truth). $\leftarrow$ **we want to minimize this**

Examples of $\mathcal{L}$: MSE, Kullback-Leibler div., anything differentiable...

Gradient descent : repeat until bored or $\mathcal{L}(y, \hat{y})$ stops decreasing

1. For step i, select at random some $x_i \in X$.
2. Compute NN predictions $\hat{y} = f_\theta(x_i)$.
3. Compute the "loss" $\mathcal{L}(y, \hat{y})$.
4. Compute gradient of loss w.r.t. the weights : $\nabla_\theta(\mathcal{L}) \leftarrow$ **we use the chain rule to "backpropagate" the gradient value into the network**
5. Update the weights : $\theta' \leftarrow \theta - \alpha\nabla_\theta(\mathcal{L})$
   α controls the "*learning rate*" (how much we update the weights)

Training a neural network

- f_θ be the NN function with θ the parameters of the NN (its *weights*),
- X the training samples and y their labels,
- $\mathcal{L}$ the loss function (quantifies the errors between NN prediction and the truth). $\leftarrow$ **we want to minimize this**

Examples of $\mathcal{L}$: MSE, Kullback-Leibler div., anything differentiable...

Gradient descent : repeat until bored or $\mathcal{L}(y, \hat{y})$ stops decreasing

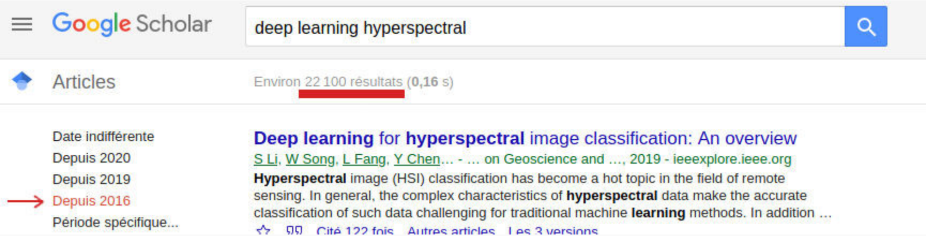
1. For step i, select at random some $x_i \in X$.
2. Compute NN predictions $\hat{y} = f_\theta(x_i)$.
3. Compute the "loss" $\mathcal{L}(y, \hat{y})$.
4. Compute gradient of loss w.r.t. the weights : $\nabla_\theta(\mathcal{L}) \leftarrow$ **we use the chain rule to "backpropagate" the gradient value into the network**
5. Update the weights : $\theta' \leftarrow \theta - \alpha \nabla_\theta(\mathcal{L})$
   α controls the "*learning rate*" (how much we update the weights)

Applying deep learning on hyperspectral data

Deep learning for hyperspectral image processing is difficult

How to choose a model?

New papers and neural architectures are published every week.



Processing hyperspectral data is difficult...

- Hyperspectral cameras are costly and low spatial resolution
- Large gap between RGB and hyperspectral
- Annotated hyperspectral datasets are very small
  → Indian Pines (hyperspectral) = 1 image of 21,025 pixels
  → ImageNet (photos) = 1 million 224 × 224 images

Unsupervised/supervised learning

Supervised learning : loss function depends on external labels (e.g. from manual human annotations) :

$$\mathcal{L}_\theta : (x, y) \to \mathcal{L}(f_\theta(x), y)$$

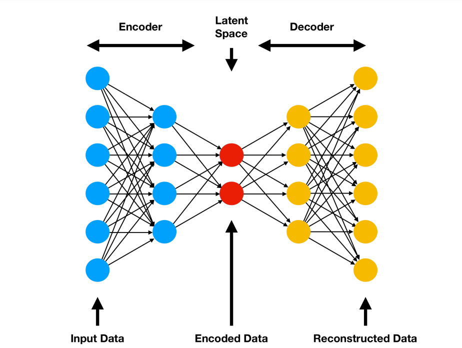
- ☺ Perfect for most tasks (classification, regression) where we have a target we want to predict
- ☹ Requires external labels that can be hard to obtain

Unsupervised learning : loss function depends only on data

$$\mathcal{L}_\theta : x \to \mathcal{L}(f_\theta(x), x)$$

- ☺ Great to learn representations without prior, can replace traditional dimension reduction algorithms (e.g. PCA)
- ☹ No labels means less possibilities regarding what can be learnt

Learning to compress : the autoencoder



https://www.compthree.com/blog/autoencoder/

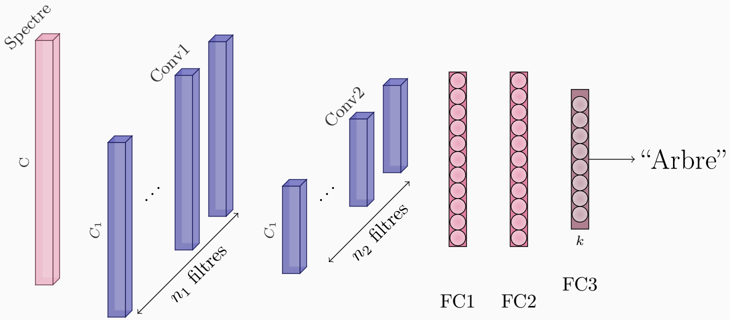
A two-part neural network : encoder + decoder

- **Encoder** : maps the input $x \in \mathbb{R}^n$ into a vector $z \in \mathbb{R}^m$. We assume $k << n$.
- **Decoder** : maps z to a "decoded" output $\hat{x}$ of same size as x.

Train the net to minimize the **reconstruction error** $||x - \hat{x}||^2$

Spectral classification : 1D CNN

- 1D convolutional kernels are applied on the spectral dimension



Deep CNN for Hyperspectral Image Classification, Hu et al., 2015

Strengths

☺ Simple, fast

☺ Scale from tens to hundreds of wavelengths

Weaknesses

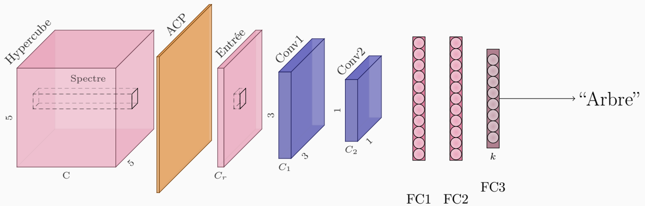
☹ No spatial awareness

Spatial-spectral classification : 2D+1D

2D+1D approaches

Reduce spectral dimension to only a few bands + 2D CNN

- Unsupervised reduction : PCA, autoencoder...
- Supervised reduction : alternate 2D and 1D convolutions



Deep supervised learning for hyperspectral data classification through CNN, Makantasis et al., 2015

Strengths

☺ Can reuse RGB models

☺ Can learn spatial patterns

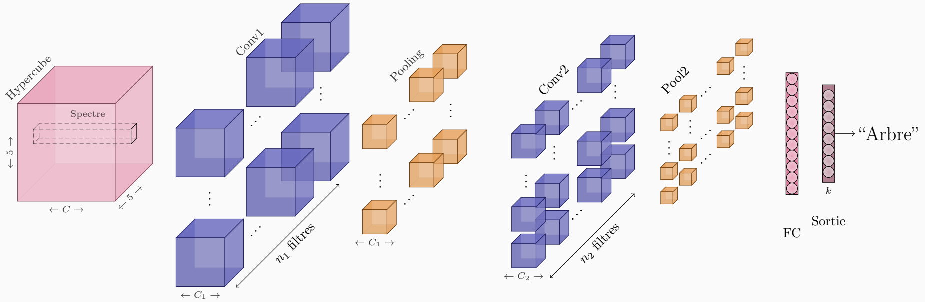
Weaknesses

☹ Shoehorning the problem

☹ Unefficient use of spectral information

3D approaches

End-to-end 3D pattern recognition : apply learnable (w, h, B) filters on the hypercube



Deep Feature Extraction and Classification of Hyperspectral Images Based on CNN, Chen et al., 2016

Strengths

- ☺ Superior on-paper abilities
- ☺ Spatial-spectral patterns !

Weaknesses

- ☹ Can be slower → **there are tricks to avoid this**

Study case : Pavia University

Dataset

Hyperspectral image of the University of Pavia (Italy) : 103 bands, 610 $\times$ 610px, 1px=1.3m (courtesy of Prof. Paolo Gamba).

Left to right : color image, 1D SVM, 3D CNN, ground truth.

one color = one class (meadows, bare soil, metal sheets...)

Choosing a model or *how not to drown in the state of the art*

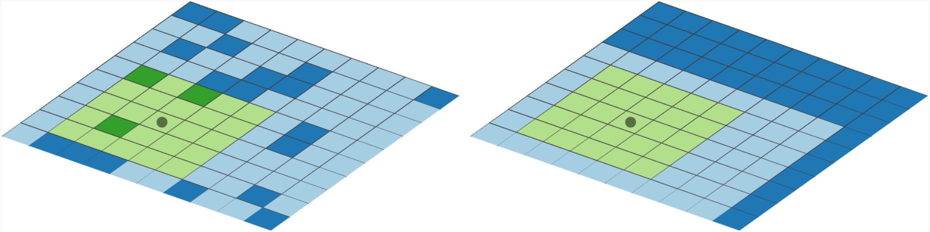
Validating my architecture

- Choose a public dataset (e.g. Pavia University, Indian Pines...)
- Split the dataset between train/test/validation
- Compare my accuracy with the state-of-the-art

It is easy to get it wrong

- Random splitting of train/test sets is unrealistic
- Different authors do not always consider the same classes
- Hyperparameters tuning is sometimes done directly on the test set $\rightarrow$ **this results in optimistic performances since we overfit the model on test data...**

Choosing a model or *how not to drown in the state of the art*



Random train/test — Disjoint train/test

It is easy to get it wrong

- Random splitting of train/test sets is unrealistic
- Different authors do not always consider the same classes
- Hyperparameters tuning is sometimes done directly on the test set $\rightarrow$ **this results in optimistic performances since we overfit the model on test data...**

Some guidelines

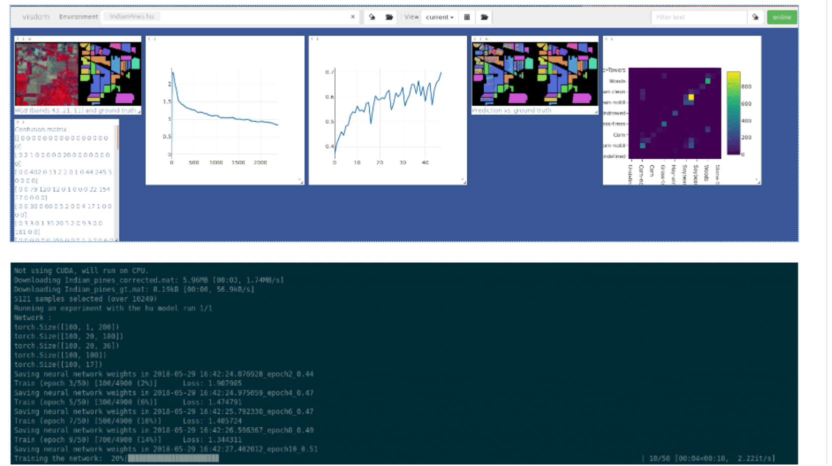
Designing my network : keeping it simple

- Start small, add layers until it starts overfitting
- Don't reinvent the wheel, use proofed optimizers (SGD, Adam...)
- Not enough data : create some more! $\leftarrow$ **data augmentation can significantly improve your models**

Validating the model

- Keep your test set hidden so that you can evaluate your model on new data $\leftarrow$ **measure generalization, not memorization**
- Be skeptical of too-good-to-be-true results, e.g. 99% accuracy...
- Be careful on any spurious correlation or info leak that could defeat your objective

A unified toolbox : DeepHyperX



https://github.com/nshaud/DeepHyperX

Deep Learning for Classification of Hyperspectral Data : A Comparative Review, Audebert et al., 2018

WIP with J.-F. Robitaille and I. Joncour (IPAG) to adapt the toolbox to astro data !

Conclusion

- Multispectral and hyperspectral imaging is a great tool for **fine-grained characterization** of objects and surfaces.
- **Deep learning and convolutional neural networks** are very strong for **image classification** but hyperspectral cubes are not the same as color images.

→ Yet, we can manage using **3D neural networks** to capture spatial-spectral patterns in the data.

- Many models have been published but no clear winner yet.
- Since datasets are small and scarce, extra care should be taken to ensure the validity of the results.

→ `DeepHyperX` tries to provide a collection of sate-of-the-art models inside the same toolbox for easier use.