

Introduction à l'apprentissage profond

Juste Raimbault¹

¹LaSTIG, IGN-ENSG-UGE



Introduction

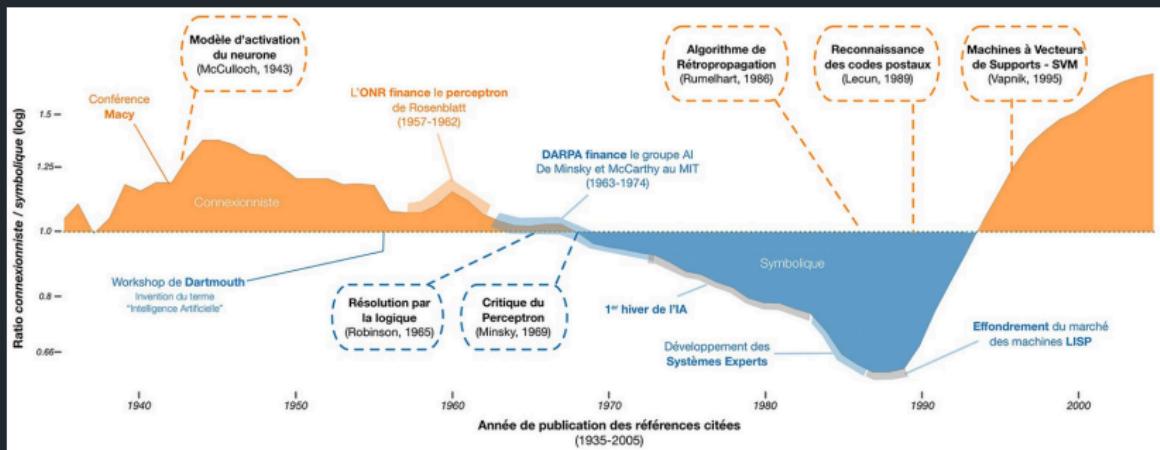
Réseaux de Neurones Artificiels

Entraînement

Implémentation

Données raster

Histoire des réseaux de neurones



[Cardon et al., 2018]

- **Apprentissage profond** : une sous-classe de méthodes en apprentissage machine, basée sur des réseaux de neurones
- **Construction endogène** des descripteurs, contrairement au ML classique (ex. SIFT en analyse d'image)
- Relativement simple théoriquement, succès grâce à la quantité de données et à la puissance de calcul (implémentations GPU)
- Pertinent et efficace pour : perception (image, vidéo, son, texte, 3D), données relationnelles (graphe), analyse sémantique, génération
- Non pertinent pour : optimisation (inverse, convexe), problèmes avec algorithme exact, environnements critiques, données tabulaires, géométrie, combinatoire

Introduction

Réseaux de Neurones Artificiels

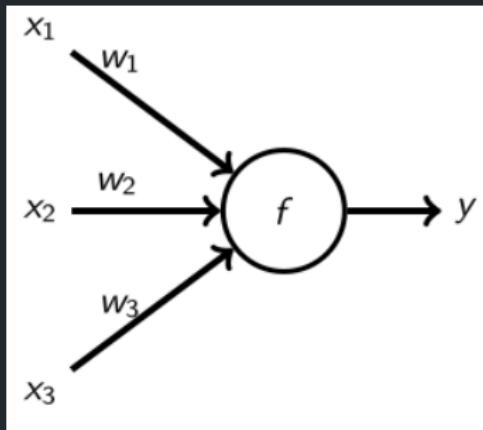
Entraînement

Implémentation

Données raster

Transforme des input x_i en une output y par une combinaison linéaire (w_i poids et b biais) et une fonction f non linéaire

$$y = f \left(\sum_i w_i x_i + b \right)$$



Rectified Linear Unit : $f = ReLu(x) = \max(0, x)$

Softmax : transforme en distribution de probabilité

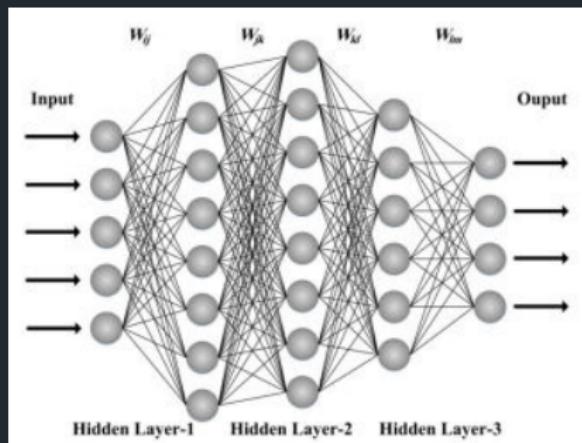
$$f(\vec{x}) = (\exp(x_i) / \sum_j \exp(x_j))_i$$

Sigmoide : lissage:

$$f(x) = \frac{1}{1 + \exp(-x)}$$

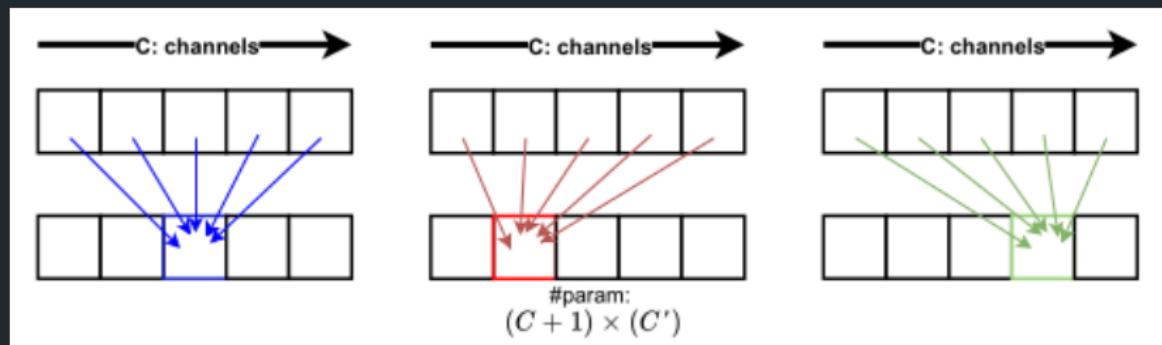
Combinaison des neurones en réseau “séquentiel” :

- **Largeur** : neurones dans une couche
- **Profondeur** : nombre de couches
- Réseaux intermédiaires : “maps” capturant des descripteurs de l’entrée

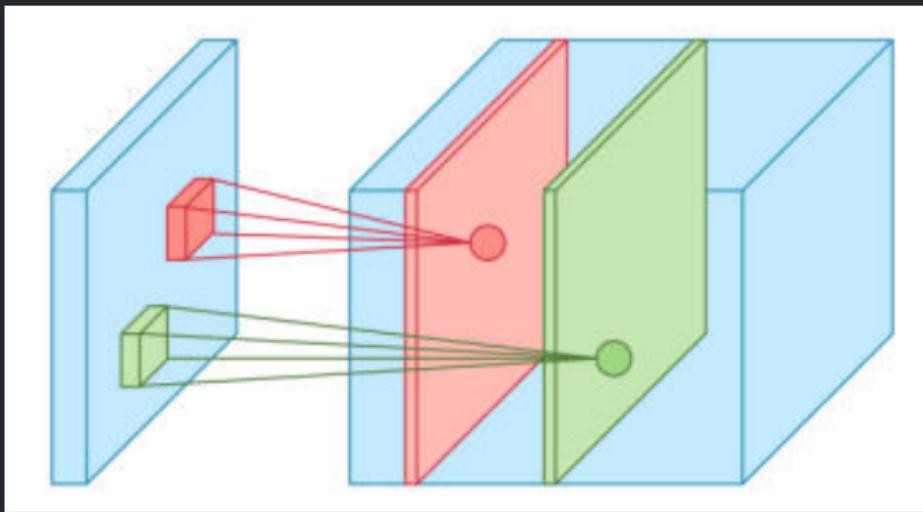


Produit matriciel $F(X) = WX + B$

→ brique élémentaire d'architectures plus compliquées



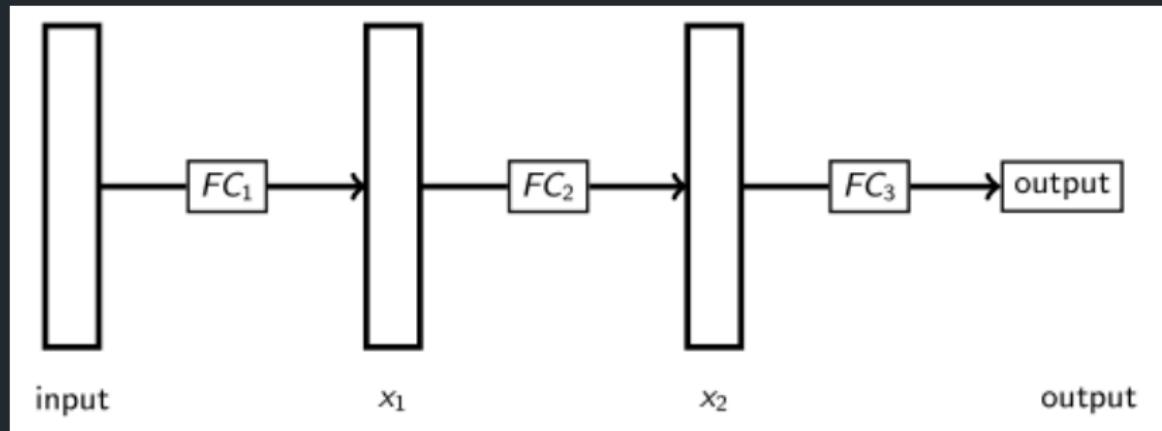
- Pour des données image : représentation des données en tenseur (map: $D_{in} \times H \times W \rightarrow D_{out} \times H \times W$)
- Passage d'un filtre local pour ne pas connecter tous les pixels
- Apprentissage d'un filtre par channel dans l'image cible :
 $params = (K \times K \times D_{in} + 1) \times D_{out}$



Architecture : Multi-layer Perceptron

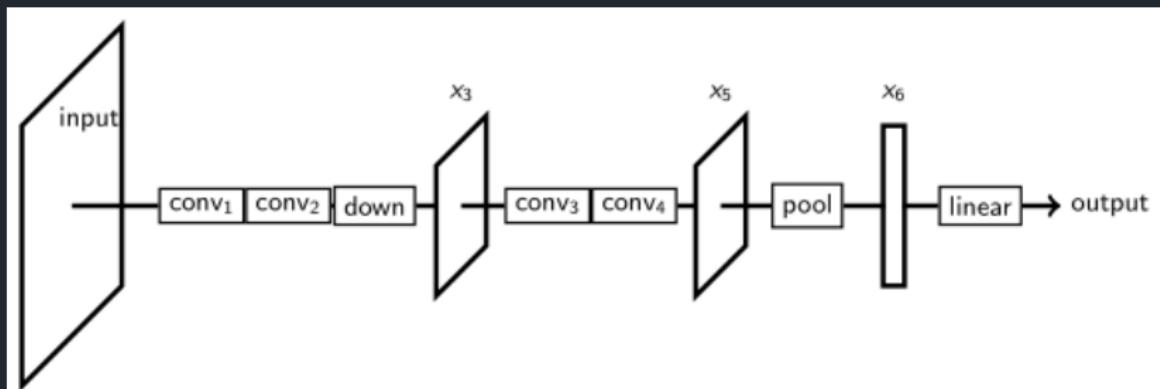
Couches linéaires :

$$x_{i+1} = \text{ReLU}(\text{Norm}(FC_i(x_i)))$$



Architecture : Classifieur convolutionnel

- Blocs de convolution : $x_{i+1} = \text{ReLU}(\text{Norm}(\text{Conv}_i(x_i)))$
- Blocs de downsampling
- Blocs de pooling : max ou moyenne sur la dimension spatiale



Introduction

Réseaux de Neurones Artificiels

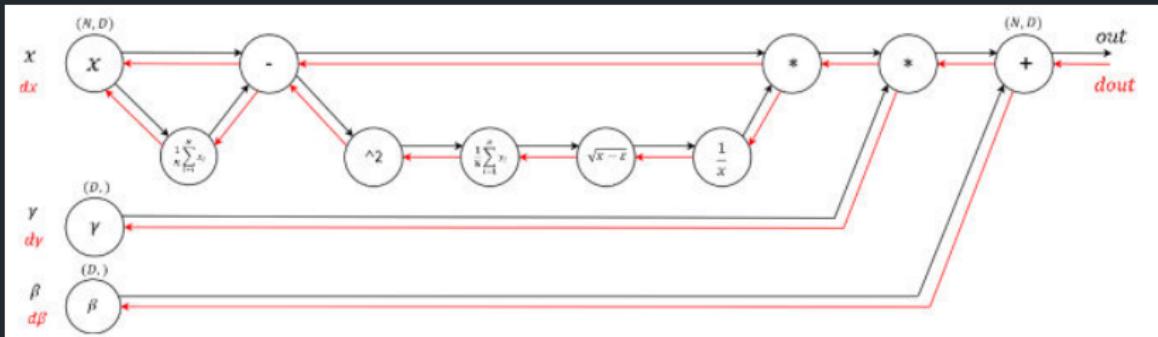
Entraînement

Implémentation

Données raster

Entraînement et Back-propagation

- Fonction de Loss : classique (cross-entropy) ou plus spécifique (ex. Triplet Loss)
- Pour optimiser les poids, descente du gradient de la fonction de Loss
- Algorithme de back-propagation : calcul des couches (forward) gardés en mémoire, puis retour pour le calcul du gradient (dérivée d'une composée)
- **Batch Stochastic Gradient Descent** : gradient le long de samples ("batch") aléatoires (accéléré avec un momentum et un adaptive learning rate)



Notation : L couche de sortie, C fonction de coût, σ activation, \odot produit d'Hadamard (produit de matrices terme-à-terme), $a^l = \sigma(z^l)$ activation de la couche l , $z^l = W^l a^{l-1} + b^l$ entrée de la couche l .

1. Erreur à la couche de sortie (L)

Mesure de l'impact des activations sur le coût final :

$$\delta^L = \nabla_a C \odot \sigma'(z^L)$$

2. Propagation de l'erreur (Couche l)

Calcul de l'erreur de la couche l en fonction de la couche suivante $l+1$:

$$\delta^l = ((W^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$$

3. Calcul des Gradients et Mise à jour

Ajustement des poids via la descente de gradient (pas η) :

$$\frac{\partial C}{\partial W^l} = \delta^l (a^{l-1})^T$$

$$W^l \leftarrow W^l - \eta \frac{\partial C}{\partial W^l}$$

Introduction

Réseaux de Neurones Artificiels

Entraînement

Implémentation

Données raster

- Packages python pour les architectures, l'optimisation et les calculs GPU: Tensorflow, Pytorch
- Grande banque de couches, architectures, fonctions d'activation, algorithmes d'optimisation, etc.
- Communauté active



- epoch : une itération sur le dataset complet
- choix des hyperparamètres, de l'optimiseur
- suivi des performances au fil des itérations

```
model = create_model(options)
optimizer = SGD(model.parameters())
for i_epoch in range(n_epochs):
    perf_train = train_one_epoch(model, optimizer)
    if i_epoch % n_epoch_test == 0:
        perf_test = test(model)
```

- batch : sample des données, avec gt ground truth correspondante
- criterion : fonction de Loss
- Evaluation : idem sans calculer le gradient

```
def train_one_epoch(model, optimizer):  
    dataset = dataloader(train_set, batch_size = 16)  
    metric_meter = empty_meter()  
    for batch, gt in dataset:  
        model.grad = 0  
        pred = model.run(batch)  
        loss = criterion(pred, gt)  
        loss.backward()  
        optimizer.step()  
        accuracy = compute_accuracy(pred, gt)  
        metric_meter.add(loss, accuracy)  
    return metric_meter.values()
```

Spécification de l'architecture selon des briques prédéfinies

```
def MyModel(nn.Module):
    def __init__(parameters):
        #set up layers
        layer1 = Conv2D(in_channel=3, out_channel=32 ...)
        layer2 = ...
        self.layers = nn.Sequential([layer1, layer2])

    def forward(input):
        #apply model to input
        return self.layers(input)
```

Introduction

Réseaux de Neurones Artificiels

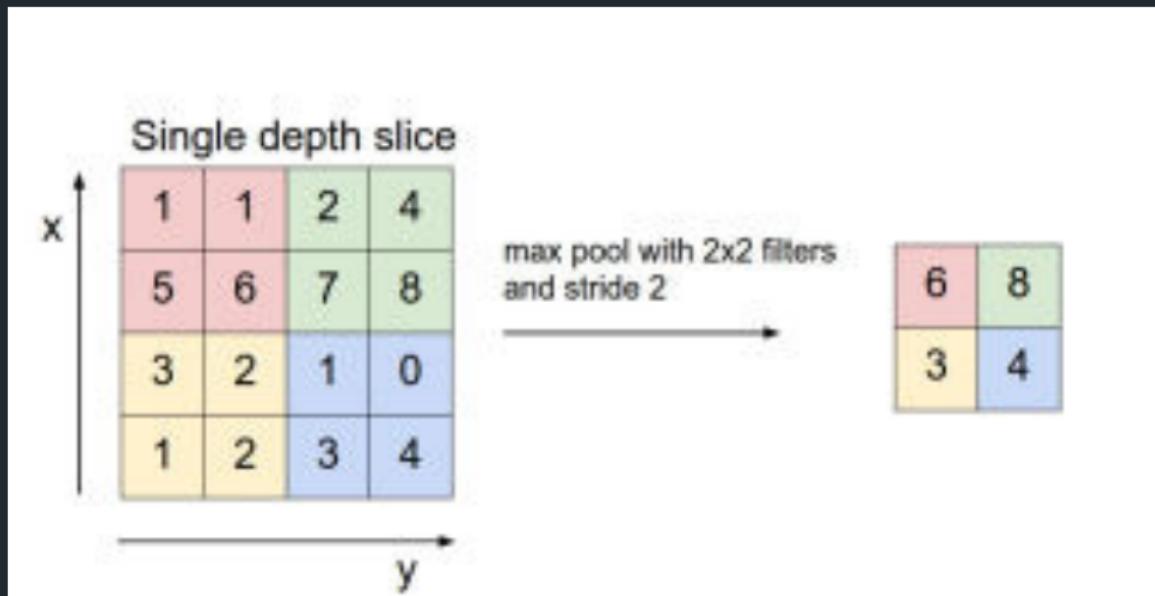
Entraînement

Implémentation

Données raster

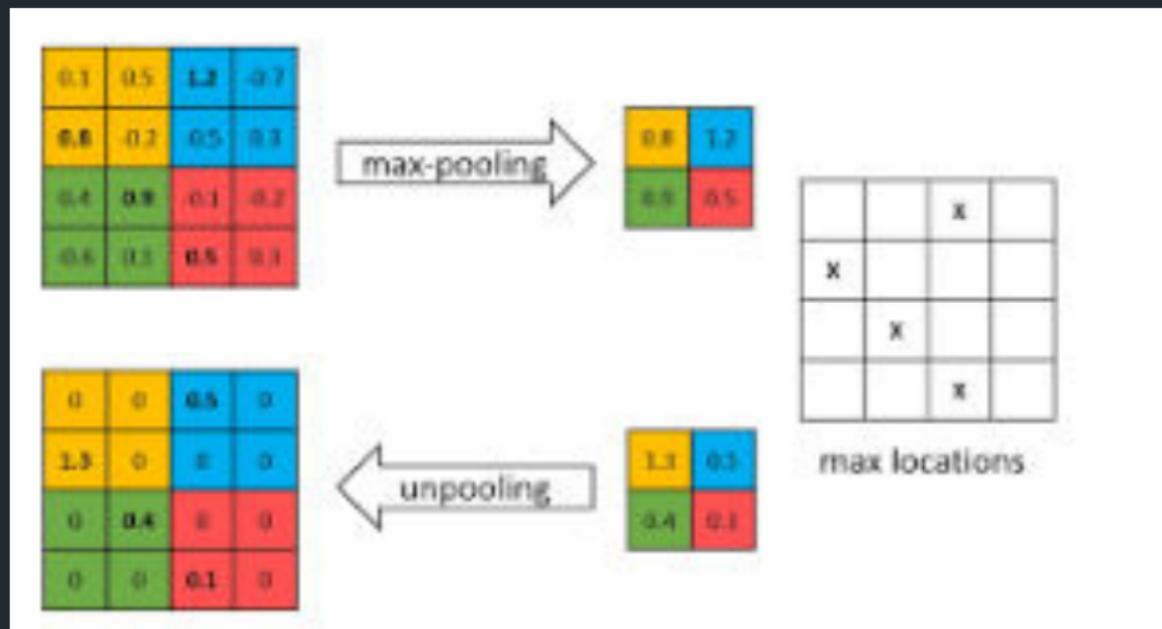
Couche de Pooling

Diminue la dimension de l'image



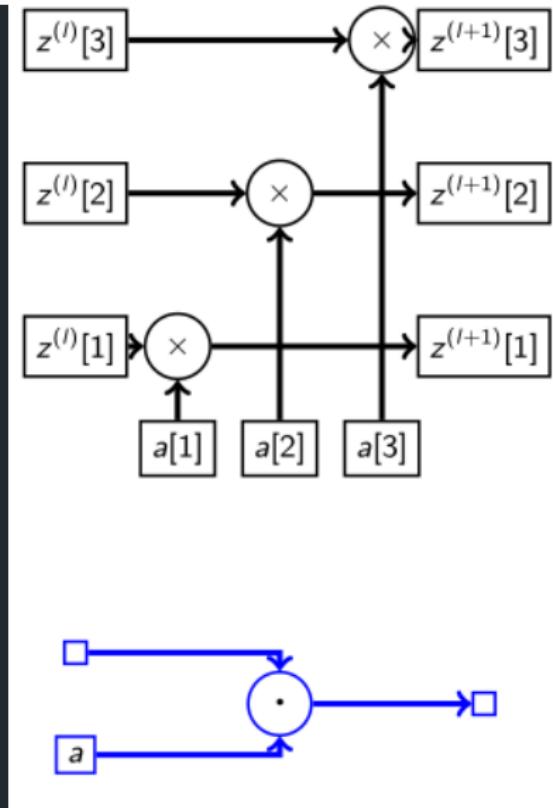
Couche de Un-pooling

Augmente la dimension, à partir d'une carte des indices du pooling

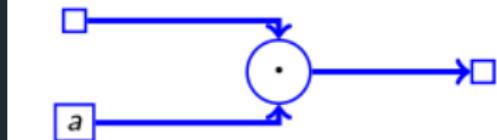
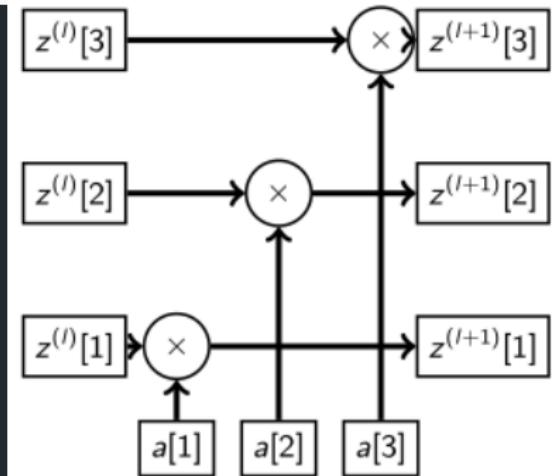


Couche d'attention

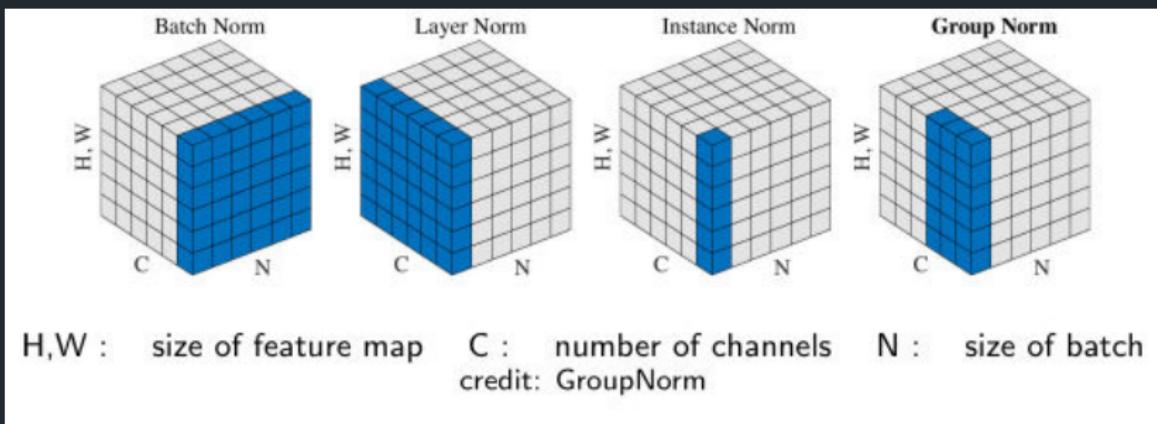
- Eteindre les channel les moins importants
- Utilisation d'un Softmax pour simuler une capacité d'attention limitée



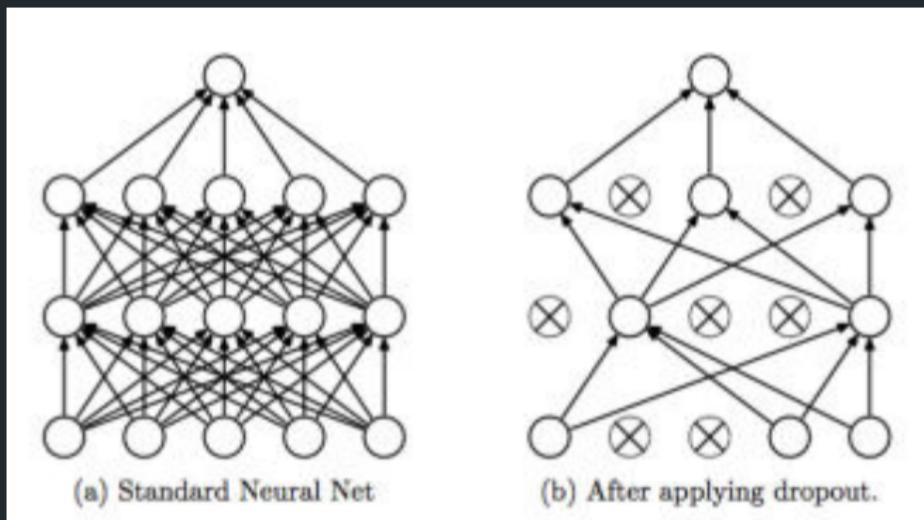
- Problème de la perte d'information exponentielle avec la profondeur du réseau
- Réinjection de l'input pour les réseaux très profonds



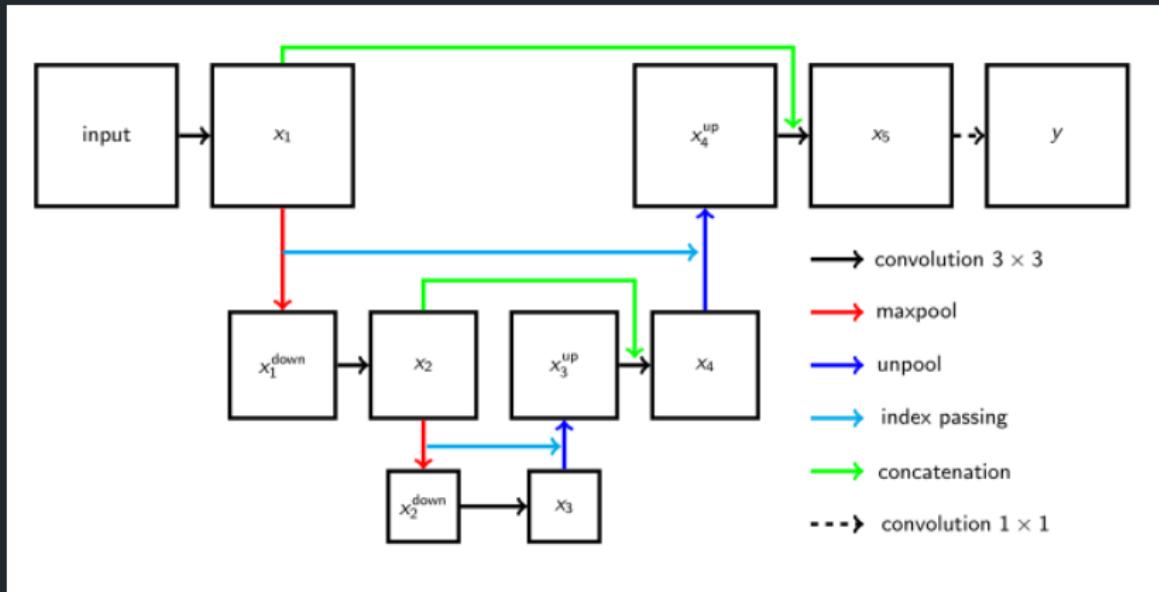
- Limite entre descripteur (encoder) et output (decoder) est floue : besoin d'ajouter des étapes de normalisation dans le réseau
- Selon différentes dimensions : BatchNorm, LayerNorm, InstanceNorm, GroupNorm



- Régularisation pour diminuer l'overfitting
- Extinction de neurones aléatoires pendant l'entraînement

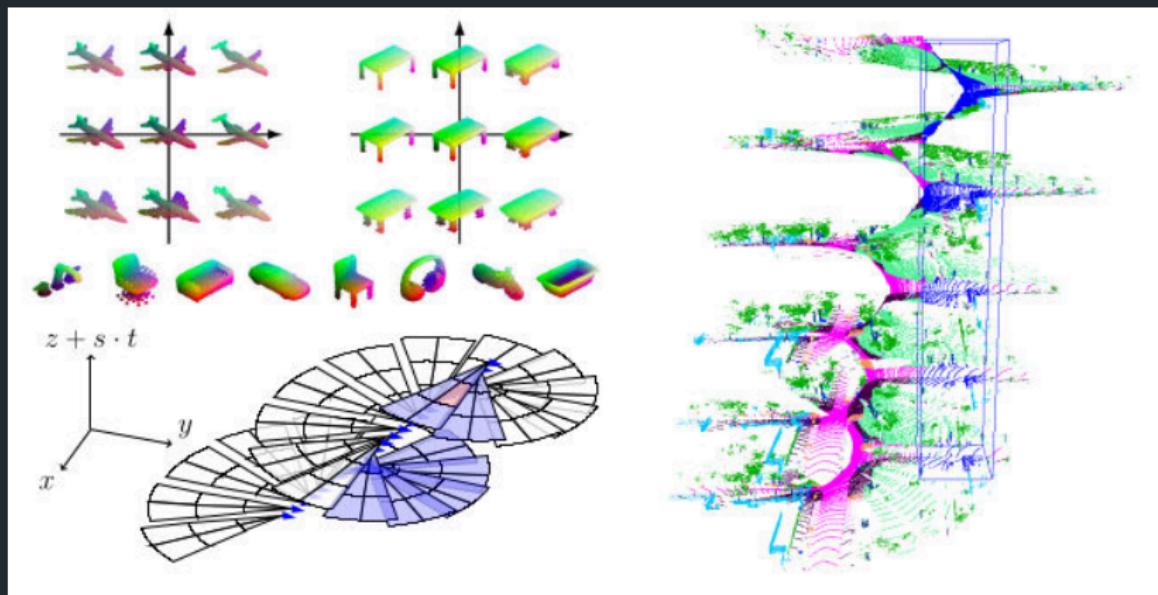


Exemple d'architecture pour la segmentation sémantique des images : classification pour chaque pixel



Exemples d'application

Segmentation sémantique de nuages de points 3D dynamiques
[Loiseau et al., 2022]



Exemples d'application

Production automatique de cartes d'usage du sol
[Postadjian et al., 2017] (cf données OCS-GE en production)

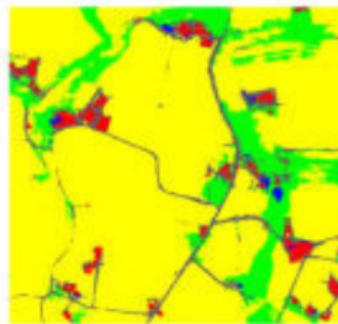
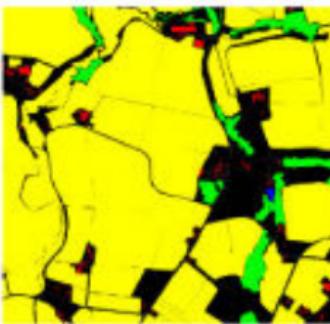
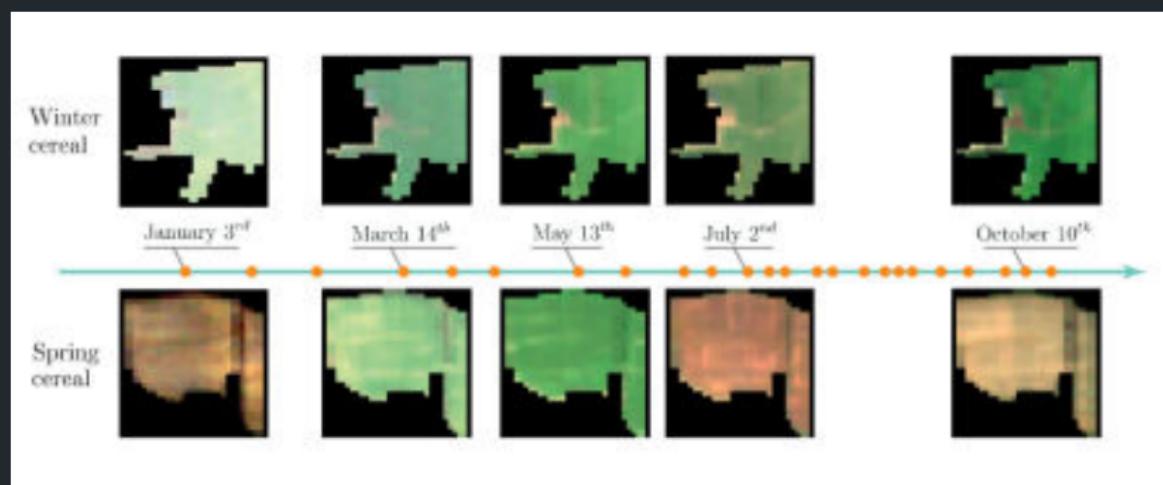


Figure 1. Coverage of the training data. From left to right:
Spot 6 image, training data, classification. ● *no training data*, ● *buildings*, ● *roads*, ● *crops*, ● *forest*, ● *water*.

Exemples d'application

Classification de parcelles agricoles à partir de séries temporelles satellite [Garnot et al., 2020]



Classification du type de parcelle agricoles dans des données d'image satellite Sentinel-2 [Garnot et al., 2020]

Notebook python sur Google Colab :

[https://colab.research.google.com/drive/
1BMX84dvLf6Uwcj8YnG16r9T0f7LKcx0X](https://colab.research.google.com/drive/1BMX84dvLf6Uwcj8YnG16r9T0f7LKcx0X)

-  Cardon, D., Cointet, J.-P., and Mazières, A. (2018).
La revanche des neurones.
Réseaux, 211(5):173–220.
-  Garnot, V. S. F., Landrieu, L., Giordano, S., and Chehata, N. (2020).
Satellite image time series classification with pixel-set encoders and temporal self-attention.
In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12325–12334.

-  Loiseau, R., Aubry, M., and Landrieu, L. (2022).
Online segmentation of lidar sequences: Dataset and algorithm.
In *European Conference on Computer Vision*, pages 301–317.
Springer.
-  Postadjian, T., Le Bris, A., Sahbi, H., and Mallet, C. (2017).
Investigating the potential of deep neural networks for large-scale classification of very high resolution satellite images.
ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 4:183–190.