RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

METEO FRANCE
À VOS CÔTÉS, DANS UN CLIMAT QUI CHANGE

Retour sur le TP n°2 (structure des CNN & exemples classiques) + Notion de champ réceptif + Quelques stages M2 + Introduction (rapide) au TP n°3

Pierre Lepetit
ENM, le 24/10/2025

Structure des ConvNets

Les « neurones » d'un réseau sont organisés en « couches ». Nous avons vu deux types de couches entraînables :

- Dans une couche type perceptron (fully connected layer), un neurone correspond à une fonction affine du vecteur d'entrée.

Exemple :

```
model.fc = Linear(1024, 512, bias=True)
# 1024 : taille du vecteur d'entrée
# 512 : Nb de neurones

# Nb de poids : 1024 x 512 + 512

x.shape = (32, 1024)
model.fc(x).shape = (32, 64)
```

Taille du batch

Biais

Structure des ConvNets

Les « neurones » d'un réseau sont organisés en « couches ». Nous avons vu deux types de couches entraînables :

- Dans une couche type perceptron (fully connected layer),
  un neurone correspond à une fonction affine du vecteur d'entrée.

- Dans une couche de convolution (convolutional layer),
  chaque neurone code pour une opération de « convolution » (moyenne glissante).

  Exemple :

```
model.conv = Conv2d(3, 64, kernel_size=7, padding=3, stride=2, bias=True)
# 3 : nb de canaux en entrée # 64 : nb de canaux en sortie (nb de neurones)
# kernel_size=7 : côté (en nb de pixels) du noyau de convolution.
# padding=3 : lignes supplémentaires ajoutées (e.g., pour conserver les dim. spatiales)
# stride=2 : pas de l'opération de moyenne glissante

# Nb de poids : 64 x (3 x 7² + 1)

x.shape = (32, 3, 256, 256)
model.conv(x).shape = (32, 64, 128, 128)  # Cartes de caractéristiques (features maps)
```

Structure des ConvNets

Les « neurones » d'un réseau sont organisés en « couches ». Nous avons vu deux types de couches entraînables :

- Dans une couche type perceptron (fully connected layer),
  un neurone correspond à une fonction affine du vecteur d'entrée.

- Dans une couche de convolution (convolutional layer),
  chaque neurone code pour une opération de « convolution » (moyenne glissante).

  Exemple :

```
self.conv = Conv1d(2, 16, kernel_size=5, padding=2, stride=4, bias=False)

# Nb de poids : ...

x.shape = (8, 2, 10201)
model.conv(x).shape = (   ...   )
```

Structure des ConvNets

Les « neurones » d'un réseau sont organisés en « couches ». Nous avons vu deux types de couches entraînables :

- Dans une couche type perceptron (fully connected layer),
  un neurone correspond à une fonction affine du vecteur d'entrée.

- Dans une couche de convolution (convolutional layer),
  chaque neurone code pour une opération de « convolution » (moyenne glissante).

  Exemple :

```
self.conv = Conv1d(2, 16, kernel_size=5, padding=2, stride=4, bias=False)

# Nb de poids : 160

x.shape = (8, 2, 10201)
model.conv(x).shape = (8, 16, 2501)
```

Structure des ConvNets

Les « neurones » d'un réseau sont organisés en « couches ». Nous avons vu deux types de couches entraînables :

- Dans une couche type perceptron (fully connected layer),
  un neurone correspond à une fonction affine du vecteur d'entrée.

- Dans une couche de convolution (convolutional layer),
  chaque neurone code pour une opération de « convolution » (moyenne glissante).

  Exemple :

```
self.conv = Conv3d(1, 32, kernel_size=3, padding=1)

# Nb de poids : ...

x.shape = (32, 1, 60, 64, 64)
y = model.conv(x)
y.shape = ...
```

Structure des ConvNets

Les « neurones » d'un réseau sont organisés en « couches ». Nous avons vu deux types de couches entraînables :

- Dans une couche type perceptron (fully connected layer),
  un neurone correspond à une fonction affine du vecteur d'entrée.

- Dans une couche de convolution (convolutional layer),
  chaque neurone code pour une opération de « convolution » (moyenne glissante).

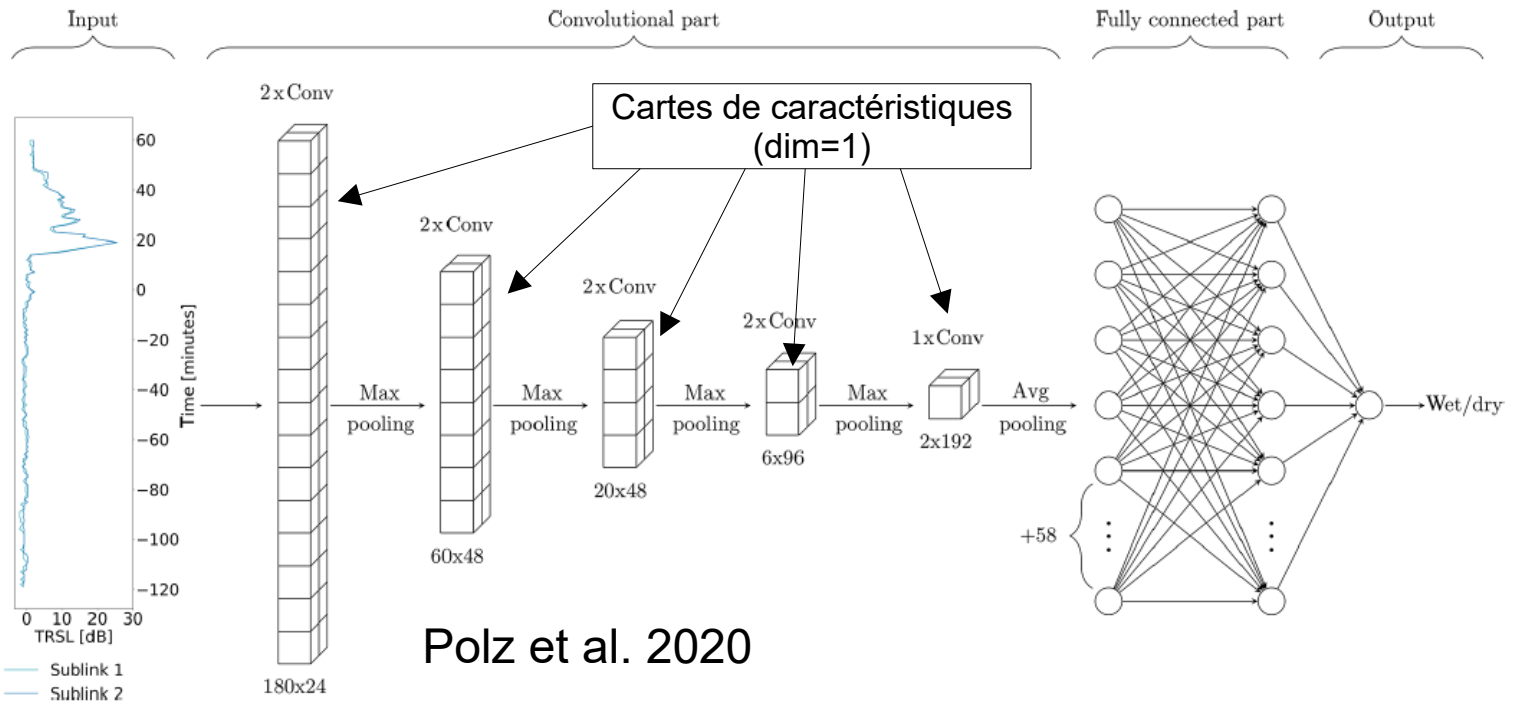
  Exemple :

```
self.conv = Conv3d(1, 32, kernel_size=3, padding=1) # default stride = 1, default bias = True

# Nb de poids : 32 x (27 + 1)

x.shape = (32, 1, 60, 64, 64)
y = model.conv(x)
y.shape = (32, 32, 60, 64, 64)
# Attention avec les conv3d :
y.numel ~ 250 M
```

Structure des ConvNets

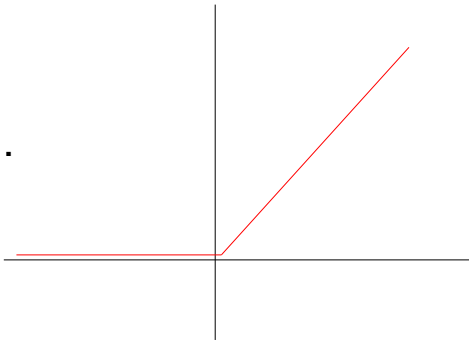


Polz et al. 2020

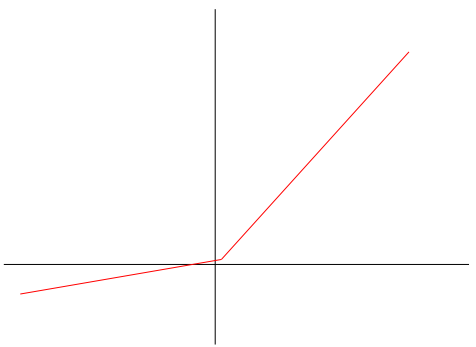
Structure des ConvNets

Les couches sont complétées par des opérations non linéaires...

- Activation (couches intermédiaires) :
  - ReLU (Recitified Linear Unit)
  - Softplus, Leaky ReLU, ELU



ReLU



Leaky ReLU

Structure des ConvNets

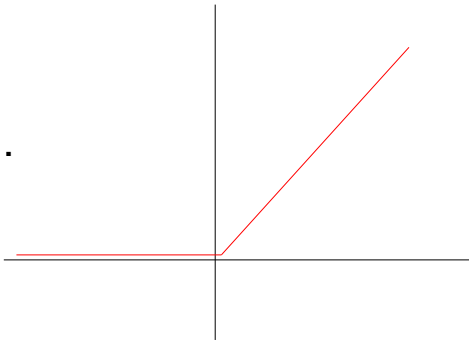
Les couches sont complétées par des opérations non linéaires...

- Activation (couches intermédiaires) :
  - ReLU (Recitified Linear Unit)
  - Softplus, Leaky ReLU, ELU

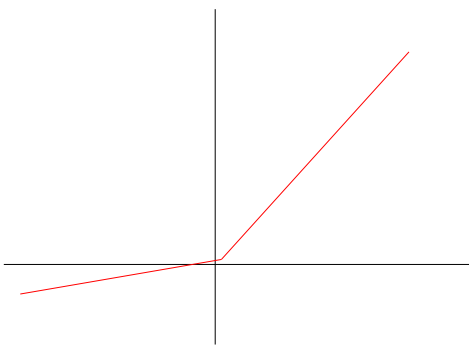
- Activations (couche finale)
  - Classification binaire : sigmoïde
  - Classification *n* classes : softmax
  - Autres : tanh, etc

```
x.shape = (32, 1 , 224, 224)
self.sigmoid(x).shape = (32, 1 , 224, 224)

x.shape = (64, 3, 224, 224)
self.softmax(x).shape = (64, 3, 224, 224)
```



ReLU



Leaky ReLU

Structure des ConvNets

Les couches sont complétées par des opérations non linéaires…
...et des opérations de réduction des dimensions spatiales :

- Réduction des dimensions spatiale : Maxpooling2D

- Autres opérateurs :
  - Maxpooling1D, 3D
  - Average Pooling/ Adaptive MaxPooling / Adaptive Average Pooling
  - flatten()

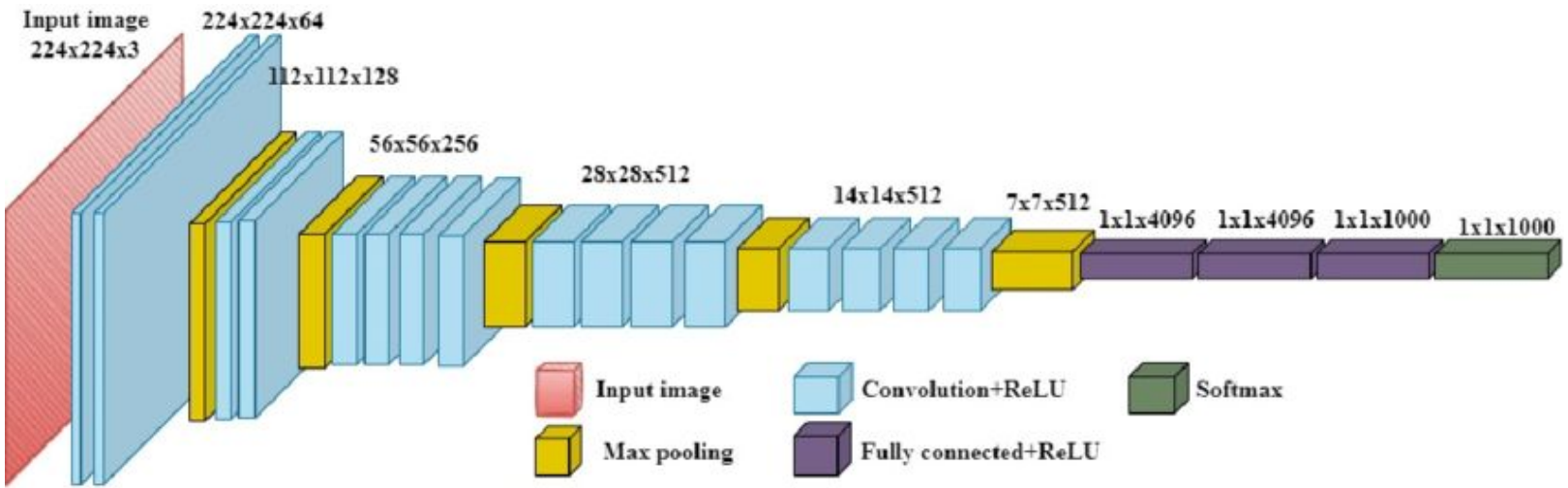
```
model.maxpool2D = MaxPool2D(2)
x.shape = (32, 16 , 224, 224)
model.maxpool2D(x).shape = (32, 16 , 112, 112)


model.adaptiveavgpool = AdaptiveAvgPool2D(64)
x.shape = (32, 16 , 100, 100)
model.maxpool2D(x).shape = (32, 16 , 64, 64)
```

Exemples de ConvNets

Deux exemples « star » à connaître

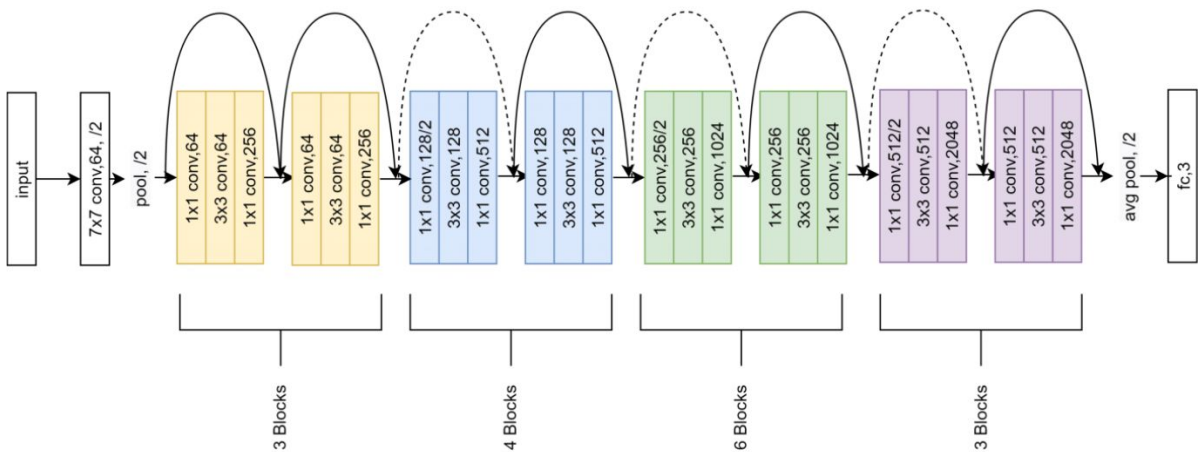
- VGG**N** avec **N** = 16, 19 (VGG : Visual Geometry Group)



Exemples de ConvNets

Deux exemples « star » à connaître

- ResNet**N** avec **N** = 18, 34, 50, 101, 152 (Residual Network)

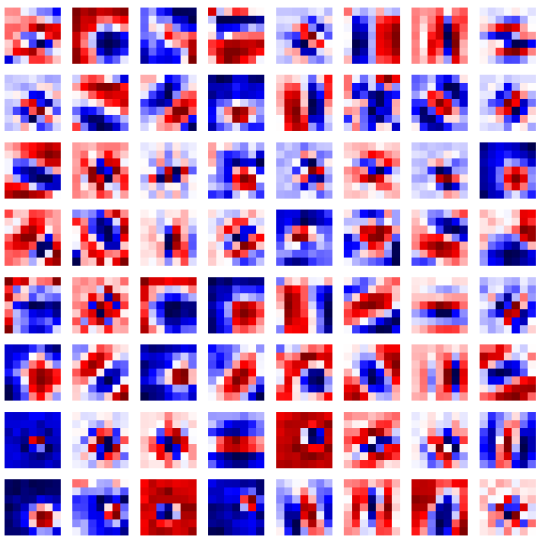


ResNet50

Exemples de ConvNets

Deux exemples « star » à connaître

- ResNet**N** avec **N** = 18, 34, 50, 101, 152 (Residual Network)



Noyaux (premier canal) de la première couche de convolution
- ResNet50 entraîné sur ImageNet -

Notion de champ réceptif

Champ réceptif associé à la composante d'une carte de caractéristique
= région du champ d'entrée qui a influencé sa valeur.

- Exemples :
  - En sortie du premier maxpool. du ResNet50, le champ réceptif est un carré de taille :

(…)

  - En sortie du premier bloc du ResNet50, le champ réceptif est un carré de taille :

(…)

Notion de champ réceptif

Champ réceptif associé à la composante d'une carte de caractéristique
= région du champ d'entrée qui a influencé sa valeur.

- Exemples :
  - En sortie du premier maxpool. du ResNet50, le champ réceptif est un carré de taille :

    (9 x 9)

  - En sortie du premier bloc du ResNet50, le champ réceptif est un carré de taille :

    (13 x 13)

- On peut obtenir la taille du champ par :

$$R_0 = 1$$

Notion de champ réceptif

Champ réceptif associé à la composante d'une carte de caractéristique
= région du champ d'entrée qui a influencé sa valeur.

- Exemples :
  - En sortie du premier maxpool. du ResNet50, le champ réceptif est un carré de taille :

    (9 x 9)

  - En sortie du premier bloc du ResNet50, le champ réceptif est un carré de taille :

    (17 x 17)

- On peut obtenir la taille du champ par :

$R_0 = 1, J_0 = 1$;
$R_n = R_{n-1} + (k_n - 1) J_{n-1}$ k_n est la taille du noyau,
$J_n = J_{n-1} s_n$ s_n est la « stride » de la couche n

Notion de champ réceptif

Champ réceptif associé à la composante d'une carte de caractéristique
= région du champ d'entrée qui a influencé sa valeur.

- Exemples :
  - En sortie du premier maxpool. du ResNet50, le champ réceptif est un carré de taille :

    (9 x 9) | k1 = 7, s1 = 2, k2 = 2 → R1 = 7, J1 = 2, R2 = 9, J2 = 4

  - En sortie du premier bloc du ResNet50, le champ réceptif est un carré de taille :

    (17 x 17) | k3 = 1, s3 = 1, k4 = 3, s4 = 1, k5 = 1, s5 = 1
    → R5 = 17

- On peut obtenir la taille du champ par :

$R_0 = 1, J_0 = 1$;
$R_n = R_{n-1} + (k_n - 1) J_{n-1}$ k_n est la taille du noyau,
$J_n = J_{n-1} s_n$ s_n est la « stride » de la couche n

Quelques stages

- Quelques PFE du CNRM et du Cerfacs
- Fiches dans https://cnrm.sedoo.fr/stages/
  et https://cerfacs.fr/offer/?pt=offer&pg=1&pol=1&loa=0
- Sélection ici : https://github.com/nanopiero/ML_S5_etudiants/tree/main/trainings

Introduction au TP n°3

But : savoir exploiter un CNN pour de la classification

- Entraînement « from scratch » sur MNIST ;
  comparaison avec un perceptron multicouche
  - Notion de validation (préfigure le jeu de test)
  - Courbes d'apprentissage

- Utilisation d'un ResNet pré-entraîné sur ImageNet
  - Data Augmentation (DAUG) adaptée
  - Accélération du fine tuning sur GPU (x10 – x100)
  - Méthodes de fine-tuning (freezing) : cas particulier d'un apprentissage par transfert