RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

METEO FRANCE
À VOS CÔTÉS, DANS UN CLIMAT QUI CHANGE

Machine Learning – Recap' n°2

Pierre Lepetit
ENM, le 25/10/2024

Structure des ConvNets

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

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

  Exemple :

```
Self.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)
```

Structure des ConvNets

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

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

  Exemple :

```
Self.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 :

- 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 = 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.fc(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 :

- 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 :

- 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 : 80

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 :

- 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, 60, 64, 64)
model.conv(x).shape = ( ... )
```

Structure des ConvNets

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

- 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).

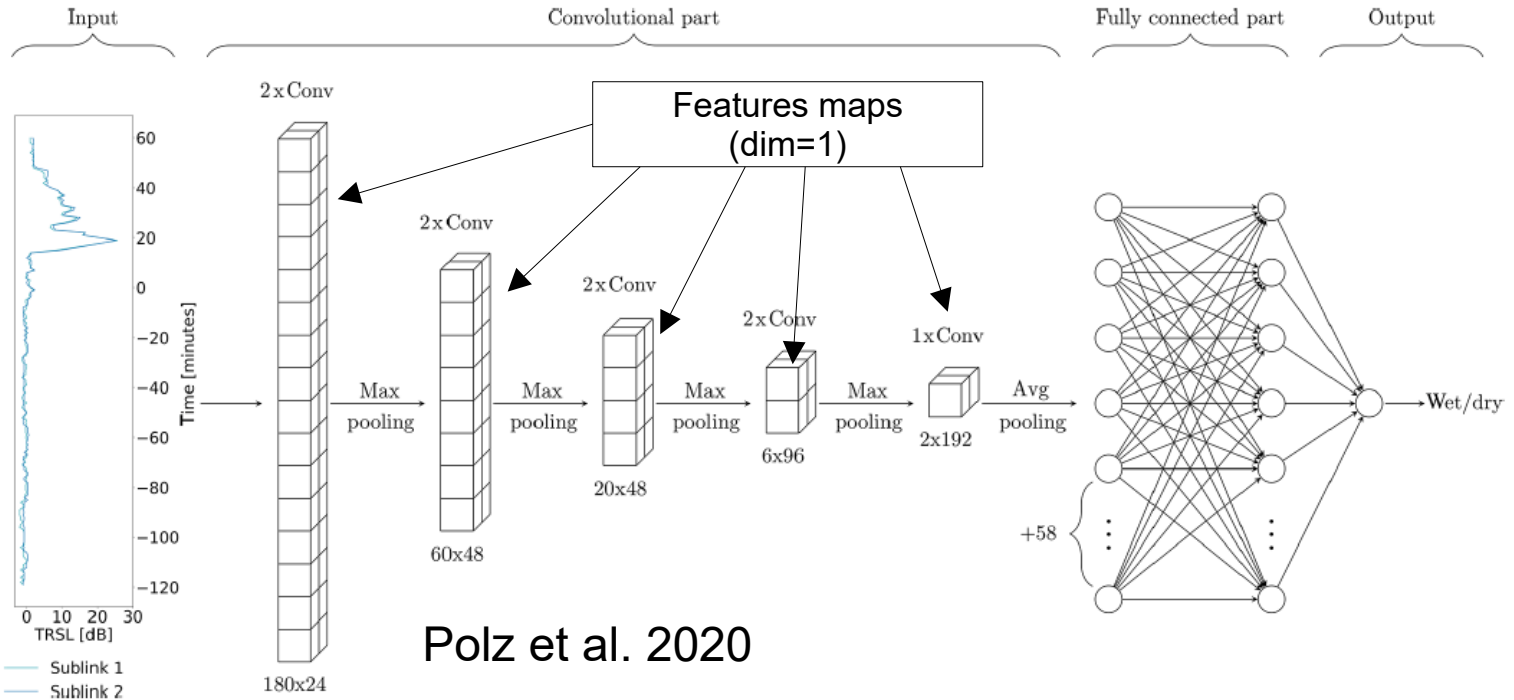
  Exemple :

```
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

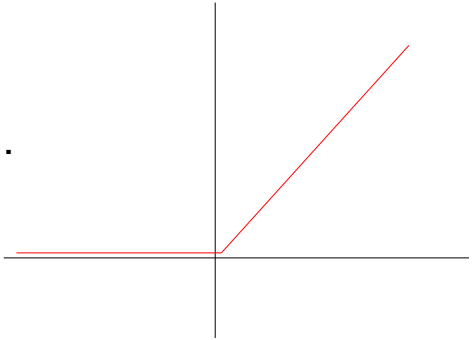


Polz et al. 2020

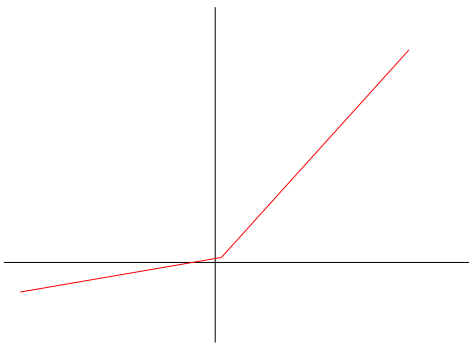
Structure des ConvNets

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

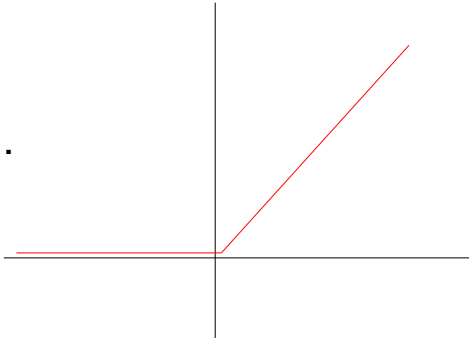
Les couches sont complétées par des opérations non linéaires...

- Activation (couches intermédiaires) :
  - ReLU (recitified linear Unit)
  - Softplus, Leaky ReLU, ELU

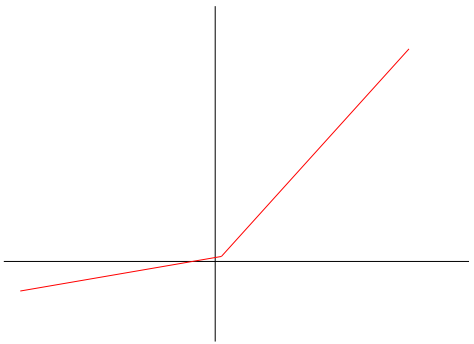
- Activations (couche finale)
  - Classification binaire : sigmoïde
  - Classification *n* classes : softmax
  - Autres : tanh, etc

```
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 :

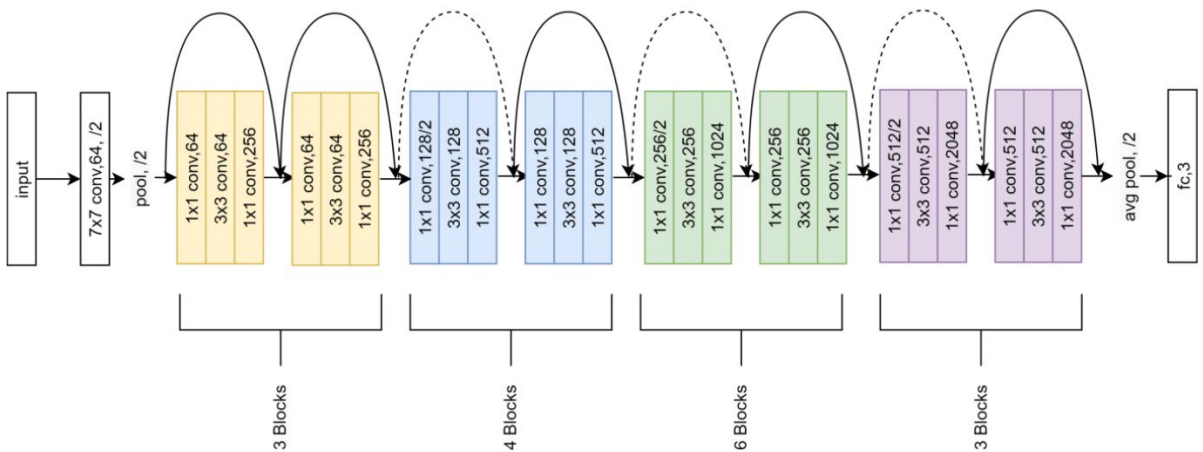
- Réduction des dimensions spatiale : Maxpooling2D

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

Exemples de ConvNets

Deux exemples « star » à connaître

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

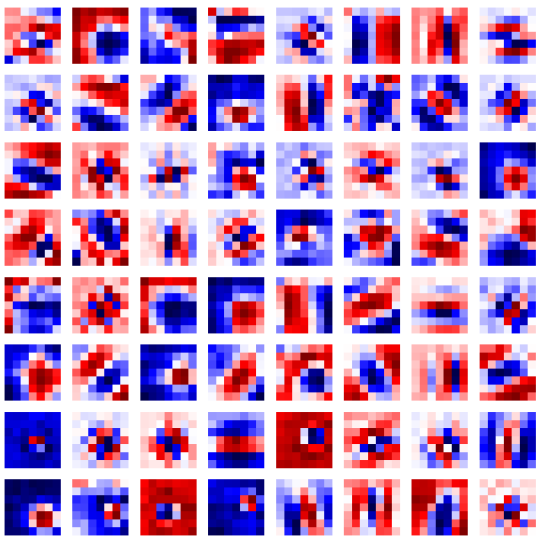


ResNet18

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 -

Exemples de ConvNets

Deux exemples « star » à connaître

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