Introduction aux réseaux de neurones pour l'apprentissage supervisé (suite)

Guillaume Bourmaud

PLAN

I. Introduction
II. Apprentissage supervisé
III. Approches paramétriques
IV. Réseaux de neurones
V. Risques
VI. Apprentissage des paramètres d'un réseau de neurones
VII. Réseaux de neurones à convolution
VIII. Réseaux de neurones profonds
IX. Spécialisation d'un réseau de neurones
X. « Adversarial examples »
XI. PyTorch : Bibliothèque de « Deep Learning »
XII. Application à la détection d'objets dans une image

IX) Spécialisation d’un réseau de neurones

IX)

Spécialisation (“Fine-tuning”)

- Exemple : détection du frelon asiatique

Présence (1042 images)

Absence (1844 images)

- Vision par ordinateur
  - réseaux pré-entraînés sur ImageNet (e.g Inceptionv3 ou ResNet)

- Traitement automatique du langage
  - réseaux pré-entraînés sur Wikipedia (e.g BERT)

Exemple de spécialisation d'un ResNet 18 : couches gelées

Image « normalisée »

3x224x224

7x7 Conv No bias
K=64 Pad=3 Stride=2

64x112x112

BN

64x112x112

ReLu

64x112x112

3x3 MaxPool
Pad=1 Stride=2

64x56x56

ResBlock A

64x56x56

ResBlock A

64x56x56

ResBlock B

128x28x28

$$\arg\max(\mathbf{o}) = 2 \doteq \text{"Présence"}$$

2x1

2x512 FC

512x1

7x7 Average Pool

512x7x7

ResBlock A

512x7x7

ResBlock B

256x14x14

ResBlock A

256x14x14

ResBlock B

128x28x28

ResBlock A

Couches "gelées" préalablement optimisées sur ImageNet

Nouvelle couche

Exemple de spécialisation d'un ResNet 18 : couches non gelées

Image « normalisée »

3x224x224

7x7 Conv No bias
K=64 Pad=3 Stride=2

64x112x112

BN

64x112x112

ReLu

64x112x112

3x3 MaxPool
Pad=1 Stride=2

64x56x56

ResBlock A

64x56x56

ResBlock A

64x56x56

ResBlock B

128x28x28

$$\arg\max(\mathbf{o}) = 2 \doteq \text{"Présence"}$$

2x1

2x512 FC

512x1

7x7 Average Pool

512x7x7

ResBlock A

512x7x7

ResBlock B

256x14x14

ResBlock A

256x14x14

ResBlock B

128x28x28

ResBlock A

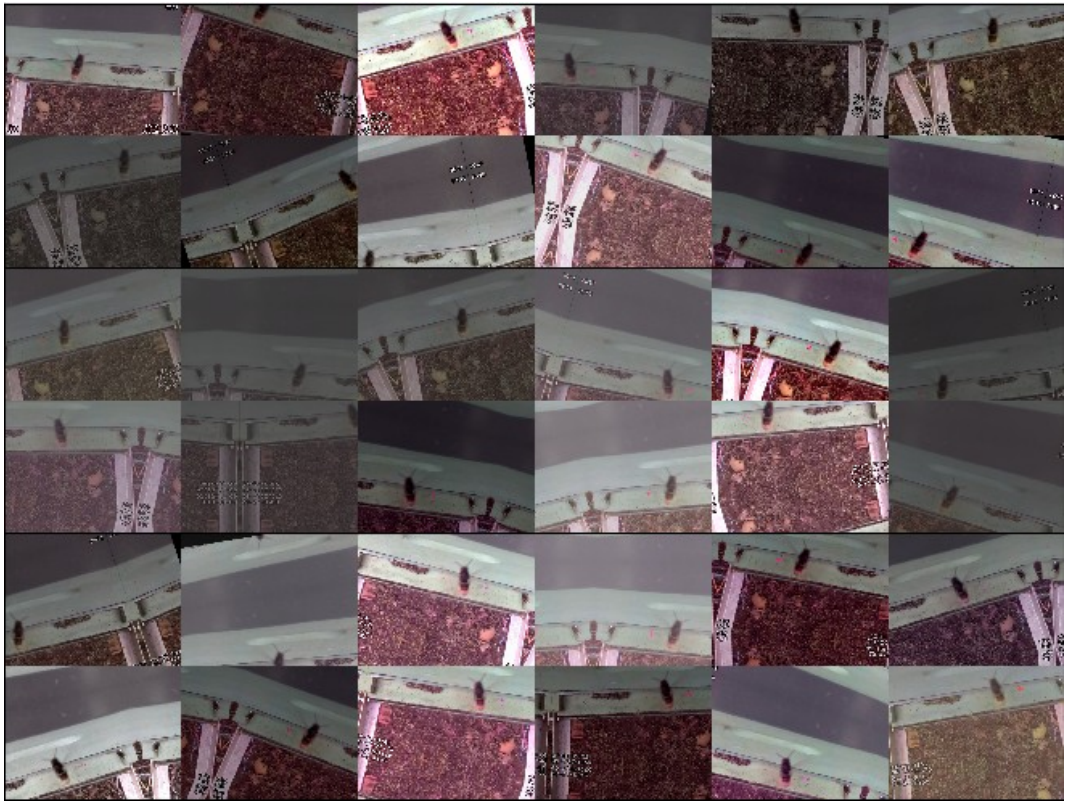
Paramètres initialisés par les valeurs d'un ResNet18 entraîné sur ImageNet

Initialisation classique (aléatoire)

IX)

Augmentation de données

- Mirroir
- Transformation affine
- Perturbation couleur
- Effacement
- Bruit
- …



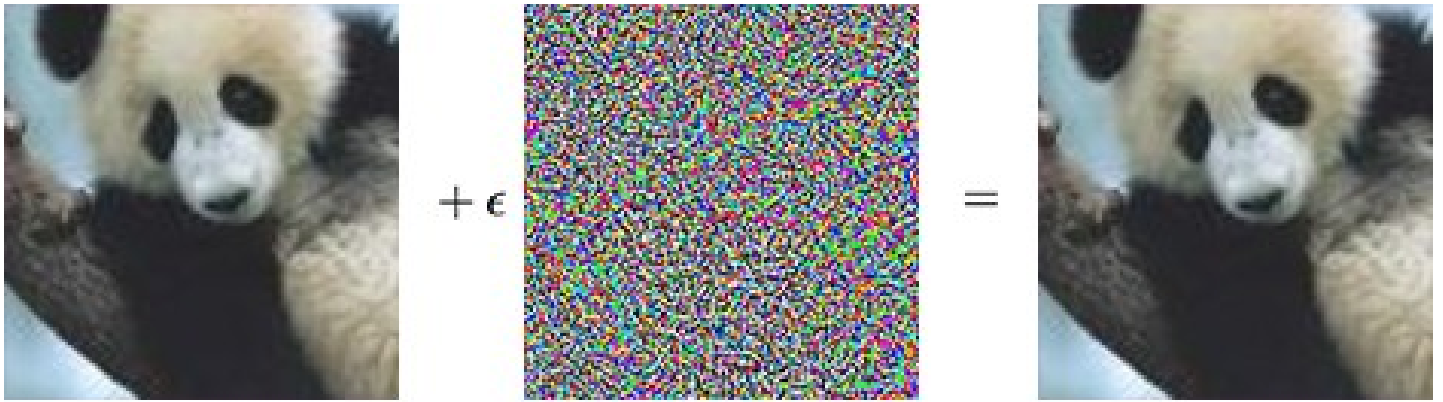
Cubuk et al. (2019). Autoaugment: Learning augmentation strategies from data. CVPR

Cubuk et al. (2020). Randaugment: Practical automated data augmentation with a reduced search space. CVPR

X) « Adversarial examples »

X)

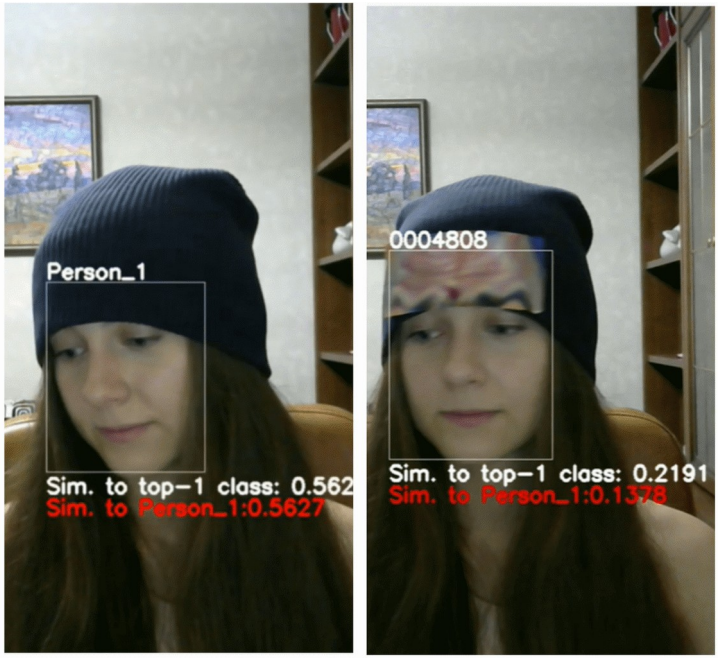
“Adversarial examples”



Source : https://openai.com/blog/adversarial-example-research/

X)

“Adversarial patches”



Sources : “ADVHAT: Real-world adversarial attack on ArcFace face ID system” (https://arxiv.org/pdf/1908.08705.pdf)
”Robust Physical-World Attacks on Deep Learning Visual Classification” (https://arxiv.org/pdf/1707.08945.pdf)

XI) PyTorch : Bibliothèque de « Deep Learning »

XI)

Bibliothèques

XI)

PyTorch

Numpy sur GPU

```
# Create a numpy array.
x = np.array([[1, 2], [3, 4]])

# Convert the numpy array to a torch tensor.
y = torch.from_numpy(x)

# Convert the torch tensor to a numpy array.
z = y.numpy()
```

Autograd

```
# Create tensors.
x = torch.tensor(1., requires_grad=True)
w = torch.tensor(2., requires_grad=True)
b = torch.tensor(3., requires_grad=True)

# Build a computational graph.
y = w * x + b    # y = 2 * x + 3

# Compute gradients.
y.backward()

# Print out the gradients.
print(x.grad)    # x.grad = 2
print(w.grad)    # w.grad = 1
print(b.grad)    # b.grad = 1
```

XI)

PyTorch : Chargement de données (« Dataloader »)

Définition du Dataloader

```
train_loader = t.utils.data.DataLoader(dataset=train_set,
                                       batch_size=batch_size,
                                       shuffle=True,
                                       num_workers=2)
```

Nombre de processus qui vont préparer des minibatches en parallèle sur CPU(s).

Définition du GPU

```
device = t.device('cuda:0' if t.cuda.is_available() else 'cpu')
```

Boucle principale d'apprentissage

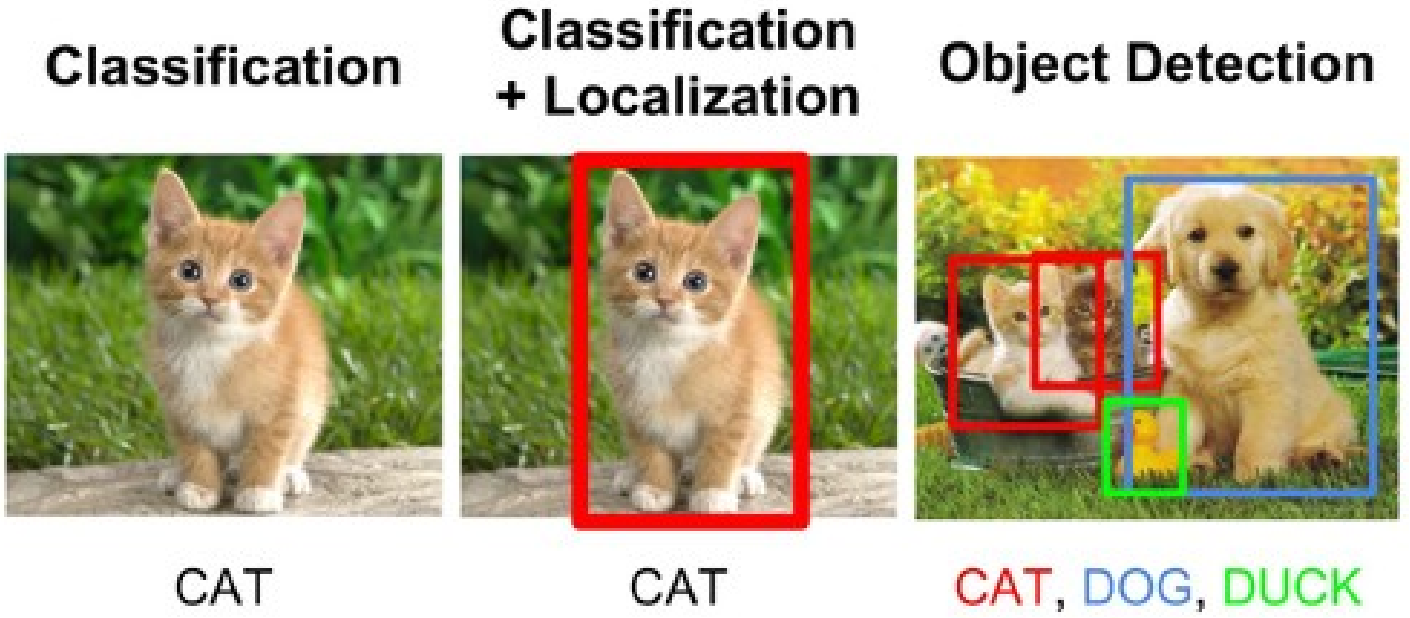
```
for epoch in range(num_epochs):
    for i, (images, labels) in enumerate(train_loader):
        images = images.to(device)
        labels = labels.to(device)
        ...
```

Mise à disposition d'un minibatch

Transfert du minibatch au GPU

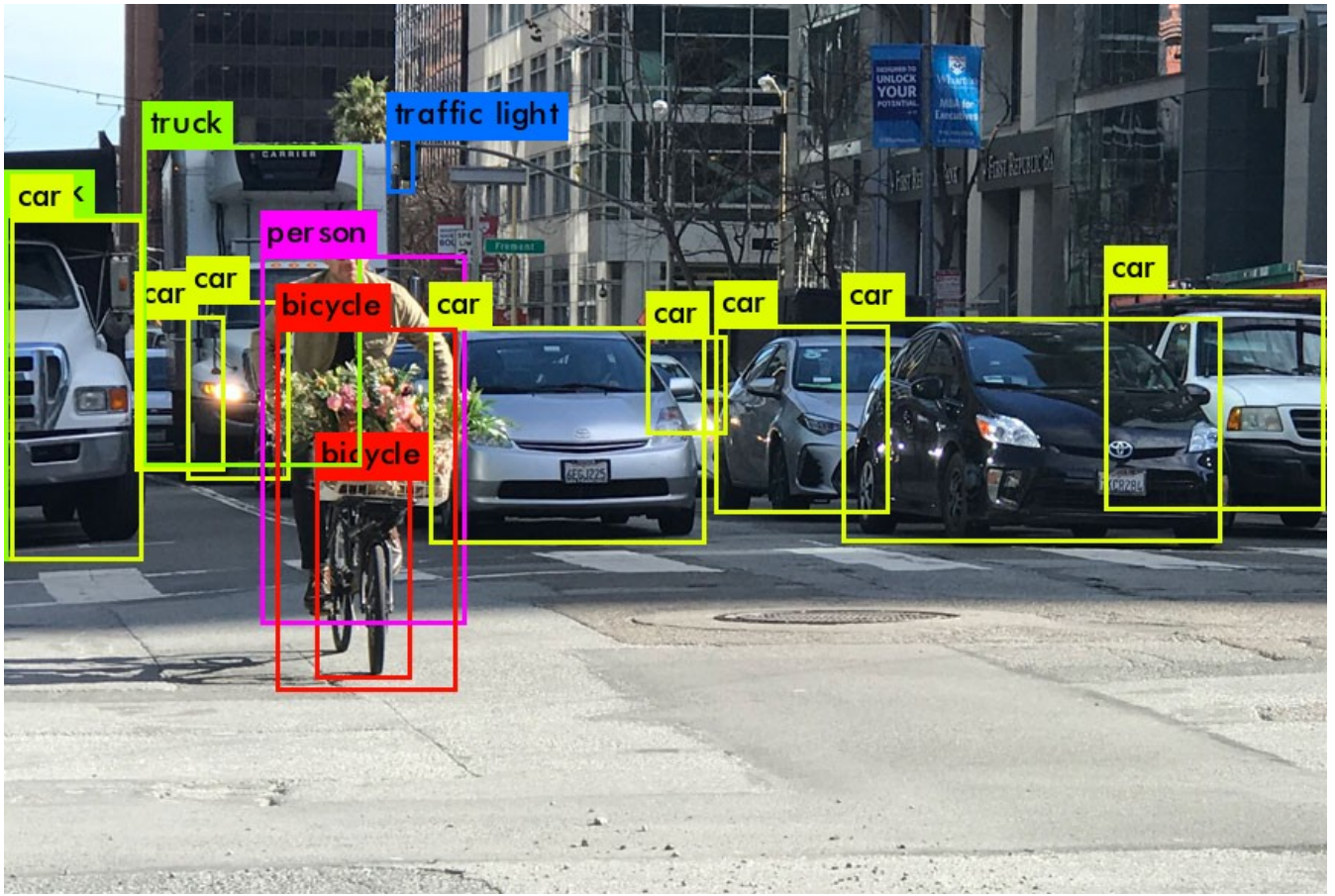
XII) Application à la détection d’objets dans une image

Détection d'objets



Source : https://medium.com/analytics-vidhya/yolov3-real-time-object-detection-54e69037b6d0

Exemple de détection d'objets



Source : https://towardsdatascience.com/yolo-you-only-look-once-17f9280a47b0

Formulation du problème

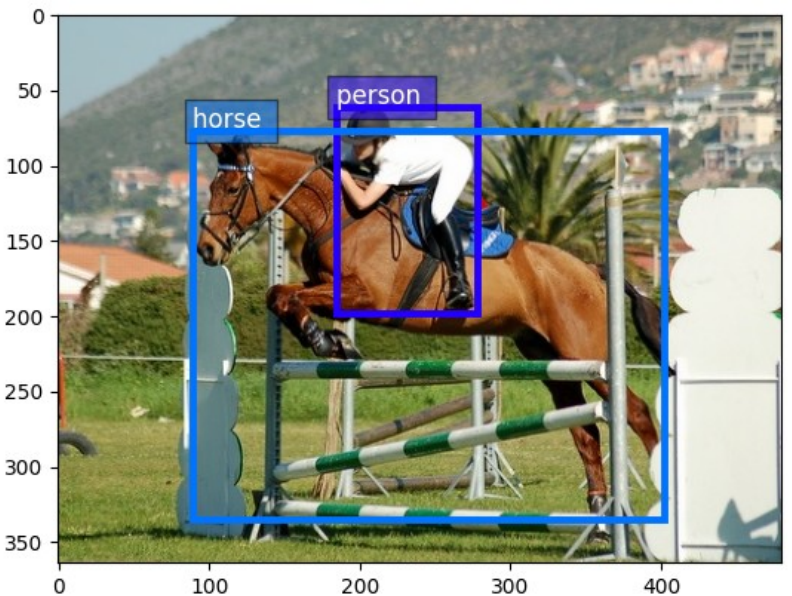
- Objectif
  - Prédire une boîte englobante autour de chaque objet,
  - Prédire la classe de l'objet,
  - En utilisant un réseau de neurones en apprentissage supervisé.

- Questions
  - Quelles données annotées disponibles ?
  - Quelle architecture ?
  - Quelle fonction de coût ?

Quelles données annotées disponibles ?

Microsoft COCO dataset (80 classes)

1: 'person',
2: 'bicycle',
3: 'car',
4: 'motorcycle',
5: 'airplane',
6: 'bs',
7: 'train',
8: 'trck',
9: 'boat',
10: 'traffic light',
11: 'fire hydrant',
12: 'stop sign',
13: 'parking meter',
14: 'bench',
15: 'bird',
16: 'cat',
17: 'dog',
18: 'horse',
19: 'sheep',
20: 'cow',
21: 'elephant',
22: 'bear',
23: 'zebra',
24: 'giraffe',
25: 'backpack',
26: 'mbrella',
27: 'handbag',
28: 'tie',
29: 'sitcase',
30: 'frisbee',
31: 'skis',
32: 'snowboard',
33: 'sports ball',
34: 'kite',
35: 'baseball bat',
36: 'baseball glove',
37: 'skateboard',
38: 'srfboard',
39: 'tennis racket',
40: 'bottle',
41: 'wine glass',
42: 'cp',
43: 'fork',
44: 'knife',
45: 'spoon',
46: 'bowl',
47: 'banana',
48: 'apple',
49: 'sandwich',
50: 'orange',
51: 'broccoli',
52: 'carrot',
53: 'hot dog',
54: 'pizza',
55: 'dont',
56: 'cake',
57: 'chair',
58: 'coch',
59: 'potted plant',
60: 'bed',
61: 'dining table',
62: 'toilet',
63: 'tv',
64: 'laptop',
65: 'mose',
66: 'remote',
67: 'keyboard',
68: 'cell phone',
69: 'microwave',
70: 'oven',
71: 'toaster',
72: 'sink',
73: 'refrigerator',
74: 'book',
75: 'clock',
76: 'vase',
77: 'scissors',
78: 'teddy bear',
79: 'hair drier',
80: 'toothbrsh'



Pascal VOC dataset (20 classes)

Person:
- 1: person

Animal:
- 2: bird
- 3: cat
- 4: cow
- 5: dog
- 6: horse
- 7: sheep

Vehicle:
- 8: aeroplane
- 9: bicycle
- 10: boat
- 11: bus
- 12: car
- 13: motorbike
- 14: train

Indoor:
- 15: bottle
- 16: chair
- 17: dining table
- 18: potted plant
- 19: sofa
- 20: tv/monitor

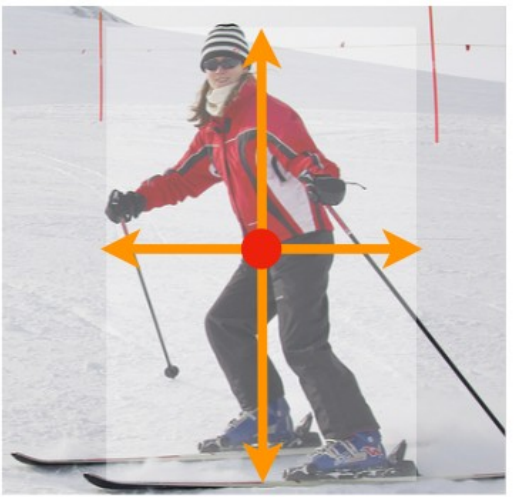
XII)

Comment prédire le nombre d'objets et pour chaque objet sa boîte et sa classe ?

Exemple de solution : CenterNet (Zhou et. al, Objects as points, 2019)

keypoint heatmap [C]

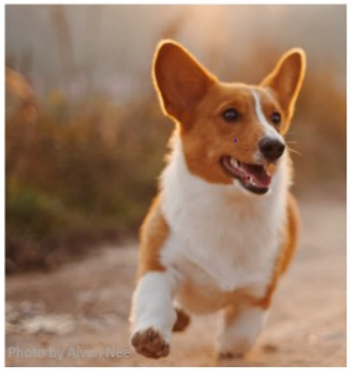
local offset [2]

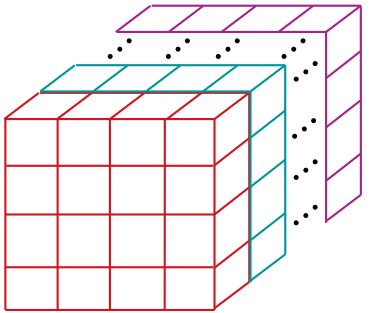


object size [2]

XII)

CenterNet





3 x H x W

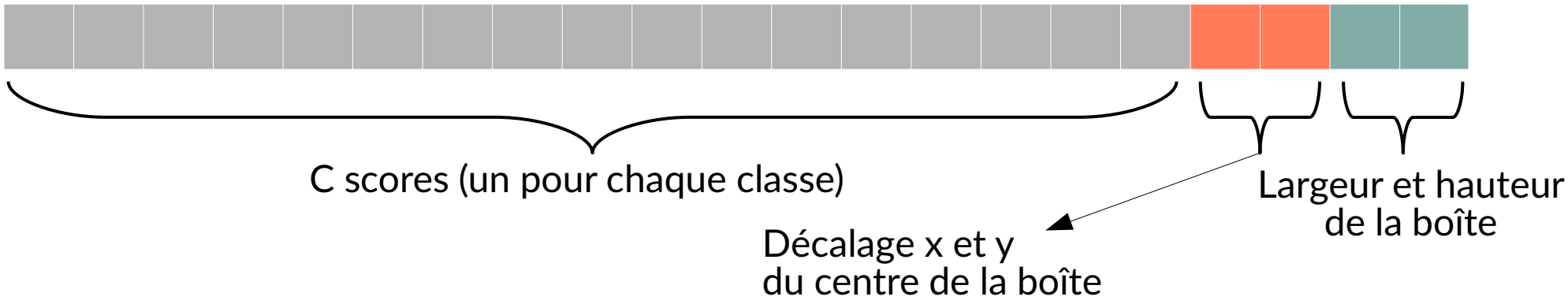
(C+2+2) x H/4 x W/4

C scores (un pour chaque classe)

Décalage x et y
du centre de la boîte

Largeur et hauteur
de la boîte

CenterNet (suite)



Fonction de coût = somme de trois fonctions de coût

- Pour les scores : « pixelwise logistic regression » (i.e. sur chaque case grise)

- Pour le décalage : régression L1 (sur les cases oranges s'il y a une boîte sinon rien)

- Pour la largeur et la hauteur : régression L1 (sur les cases vertes s'il y a une boîte sinon rien)

XII)

CenterNet (suite)