IMAGE SEGMENTATION BY SUPERPIXELS

T.Dumont[a], **B.Figliuzzi**[b]

a. MINES ParisTech, theo.dumont@mines-paristech.fr
b. MINES ParisTech CMM, bruno.figliuzzi@mines-paristech.fr

Key-words:
deep learning; convolutional neural networks; image segmentation

Abstract:
In this paper, we study different options to improve the performance of a deep learning convolutional neural network

Contents

1 Introduction **3**
1.1 Segmentation . . . 3
1.2 Superpixels . . . 3
1.3 Ce qu'est une bonne superpixelsegmentation . . . 3
1.3.1 Metrics . . . 3
1.3.2 Autres algorithmes . . . 4
1.4 Motivations/ambitions . . . 4

2 Dataset generation **4**
2.1 COCO dataset . . . 4
2.1.1 The COCO dataset . . . 4
2.1.2 Characteristics . . . 5
2.2 Eikonal . . . 5
2.3 Notre utilisation de eikonal . . . 5

3 The model **5**
3.1 Approach . . . 5
3.2 Network architecture . . . 5
3.2.1 Layers definitions . . . 5
3.2.2 Chen . . . 7
3.2.3 UNet . . . 8
3.2.4 Chen + UNet . . . 8
3.3 Total Variation (TV) Loss . . . 8
3.3.1 MSE . . . 8
3.3.2 TV . . . 8
3.4 Implementation . . . 8

4 Expérience et résultats **9**
4.1 Hyperparameters . . . 9
4.1.1 Learning rate . . . 9
4.1.2 Network size d . . . 9
4.1.3 Number of epochs . . . 9
4.1.4 TV regularization . . . 9
4.1.5 Runs . . . 9

4.2 Results on dataset 10

5 Conclusion/Discussion **10**

Todo

-

1 Introduction

1.1 Segmentation

What it is piche

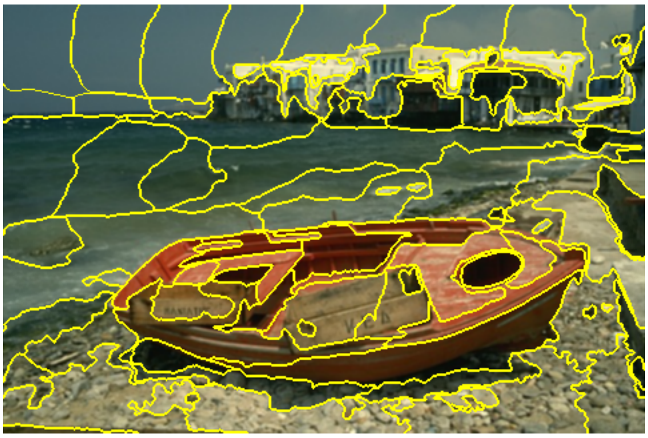


Figure 1: *An image and its segmented image*

Motivation

1.2 Superpixels

What it is piche

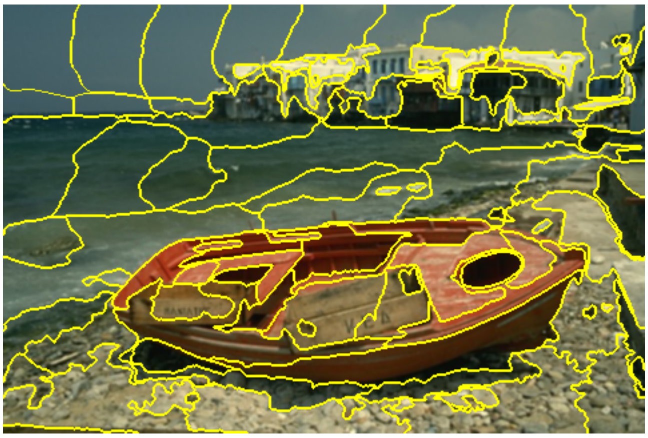


Figure 2: *Partition d'une image en superpixels*

Applications & Motivation démarrer une segmentation
fournir un support sur lequel faire de la classification (couleur/texture moyenne, etc)

1.3 Ce qu'est une bonne superpixelsegmentation

1.3.1 Metrics

cf article https://arxiv.org/pdf/1612.01601.pdf let $S = S_{j}{}_{j=1}^{K}$ and $G = G_i$ be partitions of the same image $I : x_n \mapsto I(x_n)$, $1 \leq n \leq N$ S is a segmented image G is the ground truth

Boundary Recall - most commonly used metric to assess boundary adherence. - Let $\mathrm{TP}(G,S)$ be the number of true positive boundary pixels and $\mathrm{FN}(G,S)$ be the number of false negative boudary pixels in the segmented image S.

$$\mathrm{Rec}(G,S)=\frac{\mathrm{TP}(G,S)}{\mathrm{TP}(G,S)+\mathrm{FN}(G,S)}$$

Undersegmentation Error

Compactness - evaluates the compactness of the superpixels.

$$\mathrm{CO}(G,S)=\frac{1}{N}\sum_{S_j}|S_j|\frac{4\pi A\left(S_j\right)}{P\left(S_j\right)}$$

- the CO operator computes how close the area $A(S_j)$ of each superpixel S_j is from a circle with same perimeter $P(S_j)$.

1.3.2 Autres algorithmes

SLIC

metrics Here are the previously defined metrics of some well-known superpixel segmentation algorithms.

Algorithm	BR	UE	CO
SLIC			
Reference			

Table 1: Metrics for different superpixel segmentation algorithms

We use the ?? alorithm as a reference to evaluate the performances of our model.

1.4 Motivations/ambitions

Difficultés que l'on cherche à résoudre

Pas de vraie approche DL pour segmentation avec superpixels

Ambitions améliorer les métriques

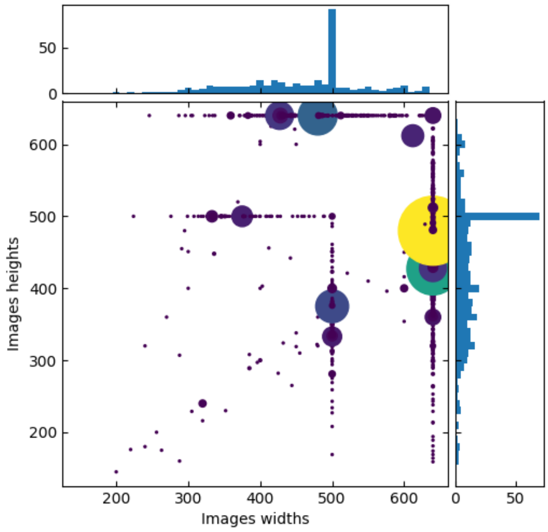
2 Dataset generation

2.1 COCO dataset

2.1.1 The COCO dataset

COCO dataset[1], nb of images, examples

(a) *plot*



(b) *plot*

Training	Height	Width
Min	?	?
Max	?	?

(c) Tabular

Validation	Height	Width
Min	145	200
Max	640	640

(d) Tabular

Figure 3: *Training and validation sets characterization*

2.1.2 Characteristics

2.2 Eikonal

2.3 Notre utilisation de eikonal

en plus réutilisé derrière sur image qui sort du réseau
faire un petit résumé

3 The model

3.1 Approach

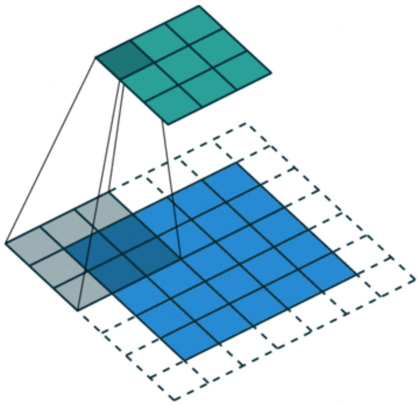
description générale de l'approche (NN puis eikonal)

3.2 Network architecture

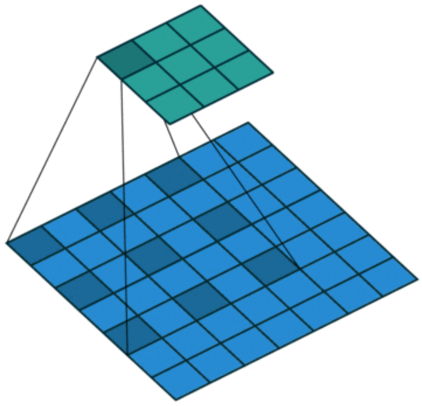
3.2.1 Layers definitions

Dilated convolution We consider a layer $L = (L_j)_{j \in [\![1,w]\!]}$, w being the number of feature maps L_j of L. We also consider $K = (K_{i,j})_{i,j}$, each $K_{i,j}$ being a 3×3 convolutional kernel. The dilated convolution operation of $K_{i,j}$ on L_j is denoted by $L_j *_r K_{i,j}$, r being the dilation parameter. The output $C(x)$ of a

[1]site de COCO



(a) *A simple convolution ($r = 1$)*



(b) *A dilated convolution ($r = 2$)*

Figure 4: *Illustration of two types of convolutions*

pixel x is:

$$\begin{aligned} C(x) &:= (L_j *_r K_{i,j})(x) \\ &= \sum_{a+rb=x} L_j(a)K_{i,j}(b) \\ &= \sum_b L_j(x-rb)K_{i,j}(b) \end{aligned}$$

and we recognize the simple convolution when $r = 1$.
A dilated convolution enables the network getting larger receptive fields while preserving the input resolution[2]

Adaptative Batch Normalization (ABN) As we have seen in (2.1.2), page 5, we need to normalize the data. We define the *adaptative normalization function* Ψ as:

$$\Psi(x) = a\ x + b\ BN(x),$$

where BN is the classic batch normalization[3], defined as:

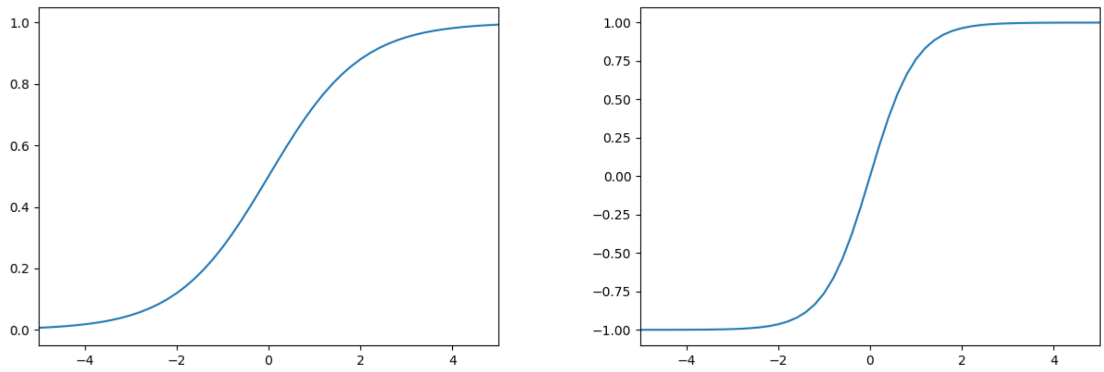
$$BN(x) = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}} * \gamma + \beta.$$

As such, Ψ combines identity mapping and batch normalization. a, b, γ and β are learned parameters[4] by backpropagation. It allows the model to adapt to each dataset, choosing whether or not giving a big importance to the identity term and the normalization term.

Leaky rectifier (LReLU) In order to let our neural network model complex patterns in the data, we have to add a non-linear property to the model. It often is an activation function, such as a sigmoid or a tanh (Figure 5).

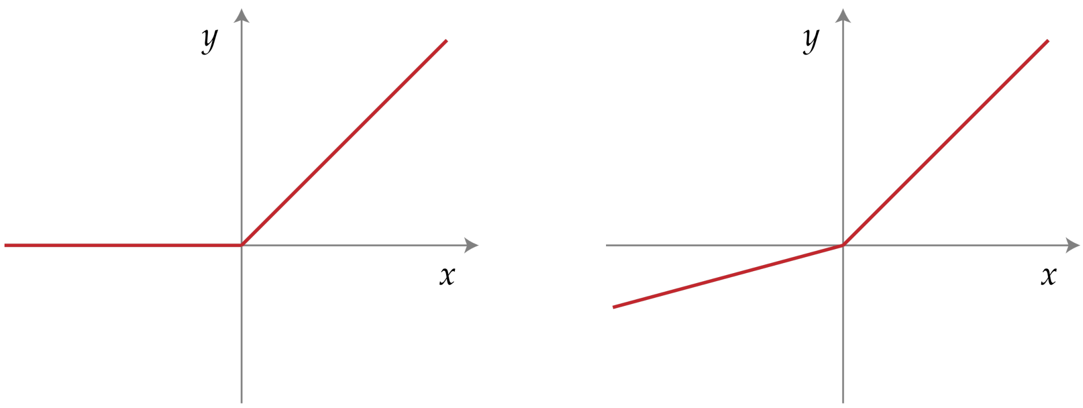
The problem with these activation functions is that they are bounded and their gradient is very low on the edges. Because we want are going to manipulate high scalar values, we have to use an unbounded activation function, such as ReLU, $\Phi(x) = \max(0, x)$ (Figure 6a). But the issue with ReLU is that all

[2] ref ?
[3] reference ?
[4] ref : `https://pytorch.org/docs/stable/_modules/torch/nn/modules/batchnorm.html`



(a) *Sigmoid* **(b)** *tanh*

Figure 5: *Illustration of two bounded rectifiers*



(a) *ReLU* **(b)** *LReLU* ($\alpha = 0.2$)

Figure 6: *Illustration of two unbounded rectifiers*

the negative values become zero immediately, which decreases the ability of our model to train from the data. Hence the implementation of a *leaky rectifier*, LReLU:

$$\Phi(x) = \max(\alpha x, x), \text{ with } 0 < \alpha < 1.$$

By implementing a Leaky Rectifier, we are able to take into account the negative valued pixels.

3.2.2 Chen

Context Aggregation Network (CAN) [5] blabla sur le RGB en entrée, RGB en sortie I -> f(I)

input I	$\longrightarrow$	L^1	$\longrightarrow$	$\cdots$	$\longrightarrow$	L^s	$\longrightarrow$	$\cdots$	$\longrightarrow$	output (L^d)
$m \times n \times 3$		$m \times n \times w_1$				$m \times n \times w_s$				$m \times n \times 3$

Table 2: *Layers*

[5]reference

Architecture of a block Each block L_s is made of 3 layers:

1. *A dilated convolution*, $r_s = 2^s$
2. *An adaptative batch normalization*
3. *A leaky rectifier (ReLU)*

so that the content of an intermediate layer L^s can be computed from the content of the previous layer L^{s-1}:

$$L_i^s = \Phi\left(\Psi^s\left(b_i^s + \sum_j L_j^{s-1} *_{r_s} K_{i,j}^s\right)\right). \tag{1}$$

where ... is ...
and

$$L_j^{s-1} *_{r_s} K_{i,j}^s = \sum_{a+r_s b=x} L_j^{s-1}(a) K_{i,j}^s(b) \tag{2}$$

because of 3.2.1, page 5.

Layer	1	2	3	4	5	6	7
Convolution	3×3						
Dilation	1						
Batch Normalization	Yes						
LReLU	Yes						

Table 3: *Chen*

3.2.3 UNet

3.2.4 Chen + UNet

3.3 Total Variation (TV) Loss

3.3.1 MSE

$$L_{MSE} = \frac{1}{N}\sum_{i=1}^{N} |\hat{f}(I)_i - f(I)_i|^2$$

3.3.2 TV

Formula

$$L_{TV} = \frac{1}{N}\sum_{i=1}^{N} |\hat{f}(I)_i - f(I)_i|^2 + \frac{1}{N}$$

Why

3.4 Implementation

The network was implemented with PyTorch[6] and we used GPU acceleration [...] (pytorch), se renseigner (section assez courante) GPU acceleraation code sur github

[6]Repository can be found at `https://github.com/theodumont/superpixels-segmentation`.

lr_0	decay?	saturation?	d	TV?
0.001	No	No	7	No
0.01	No	No	7	No
0.01	$\times 0.5$ every 2 epochs	10^{-4}	7	No
0.001	$\times 0.5$ every 2 epochs	10^{-4}	7	No

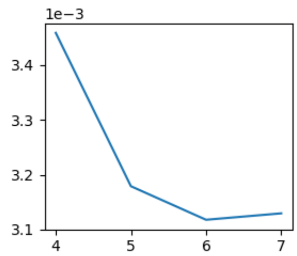
Table 4: Runs for learning rate tuning

4 Expérience et résultats

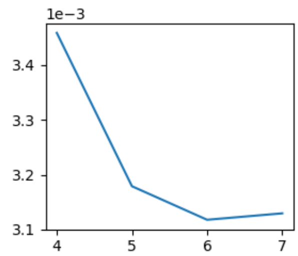
4.1 Hyperparameters

petit bilan des valeurs choisies évolution des paramètres a et b ?

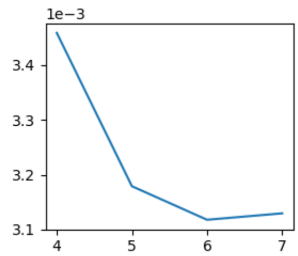
4.1.1 Learning rate



(a) *Graph*



(b) *Graph*



(c) *Graph*

Figure 7: *Tuning of learning rate*

Constant value entrainement (lr, alpha) -> courbes de loss, et loss qui sature (cluster) d'où changement de lr au cours des epochs

Non constant value

4.1.2 Network size d

- Learning rate initialisé à 0.01, divisé par 2 toutes les 2 époques, saturation à 1e-4 - Pas de régularisation TV CONCLUSION des run 3 à 5: Il est préférable de laisser d=7. Entre d=6 et d=7, l'amélioration semble relativement faible. bon intermédiaire entre temps de calcul et performances

4.1.3 Number of epochs

nb epochs: on le sélectionne en prenant le minimum de la validation loss

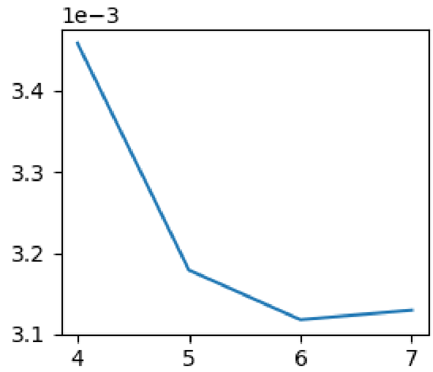
4.1.4 TV regularization

4.1.5 Runs

tableaux et graphes

d	4	5	6	7	8
loss$\times 10^3$	3.46	3.18	3.12	3.13	???

(a) *Loss values on validation set*



(b) *Graph*

Figure 8: *Tuning of network size*

4.2 Results on dataset

image originale -> CNN -> résultat du filtre dans eikonal -> superpixels sans couleurs + couleur moyenne pour chaque spp de l'image originale cf results/images

5 Conclusion/Discussion

On a présenté un nouveau...
On a prouvé...
Il reste à faire...
relire tous les mails pour avoir toutes les infos sur performances etc

Special thanks

Sources

[1] [1] C. Smith, J.C. Green, Titre de l'article, Titre du journal, 10 (2009) 55-72
[2] M. Truk, C. Bidul. Titre du bouquin, John Wiley and Sons, New York, 1973
[3] P. Machin, Titre de la thèse, Thèse, Université Poitiers, 1992
[4] D. Pierre, J.-P. Paul, B. Jacques, Titre communication, in: D. Editor, G. Editeur, (éd.), Proceedings of Conference XXX , Publisher, Paris, France, 1995, pp. 3–6