Imagerie numérique

Projet inpainting

Mathis Petrovich et Raphaël Bricout

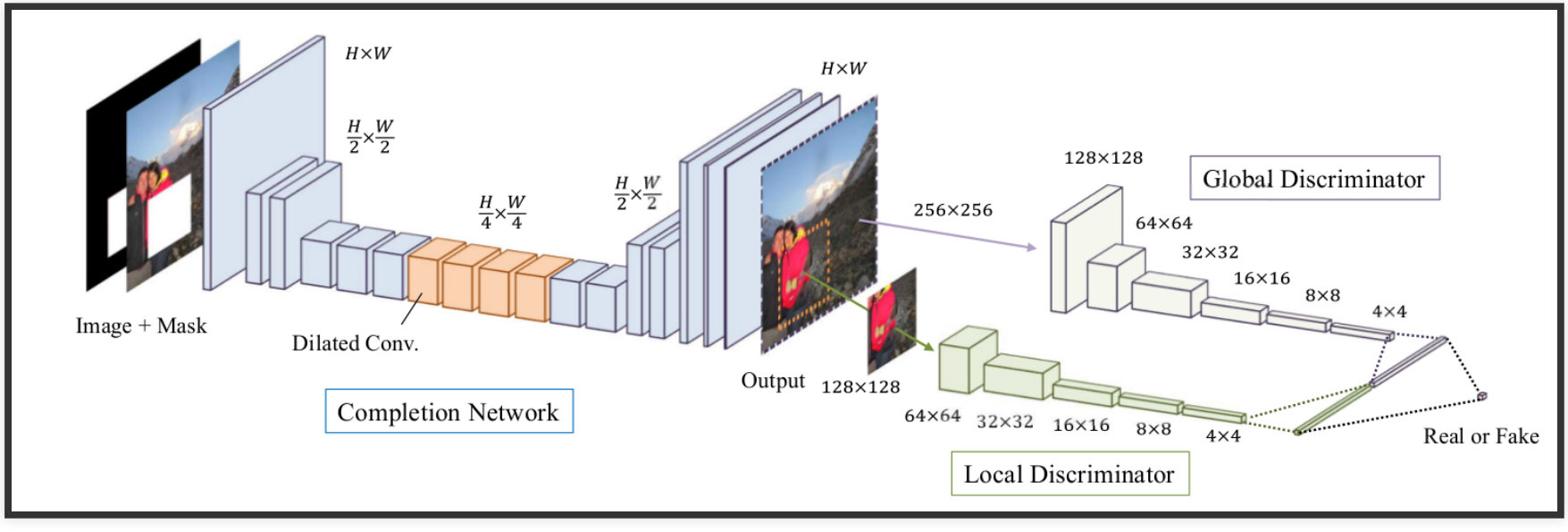
Méthodes usuelles pour ce problème

- Méthodes par patchs
- Context encoder

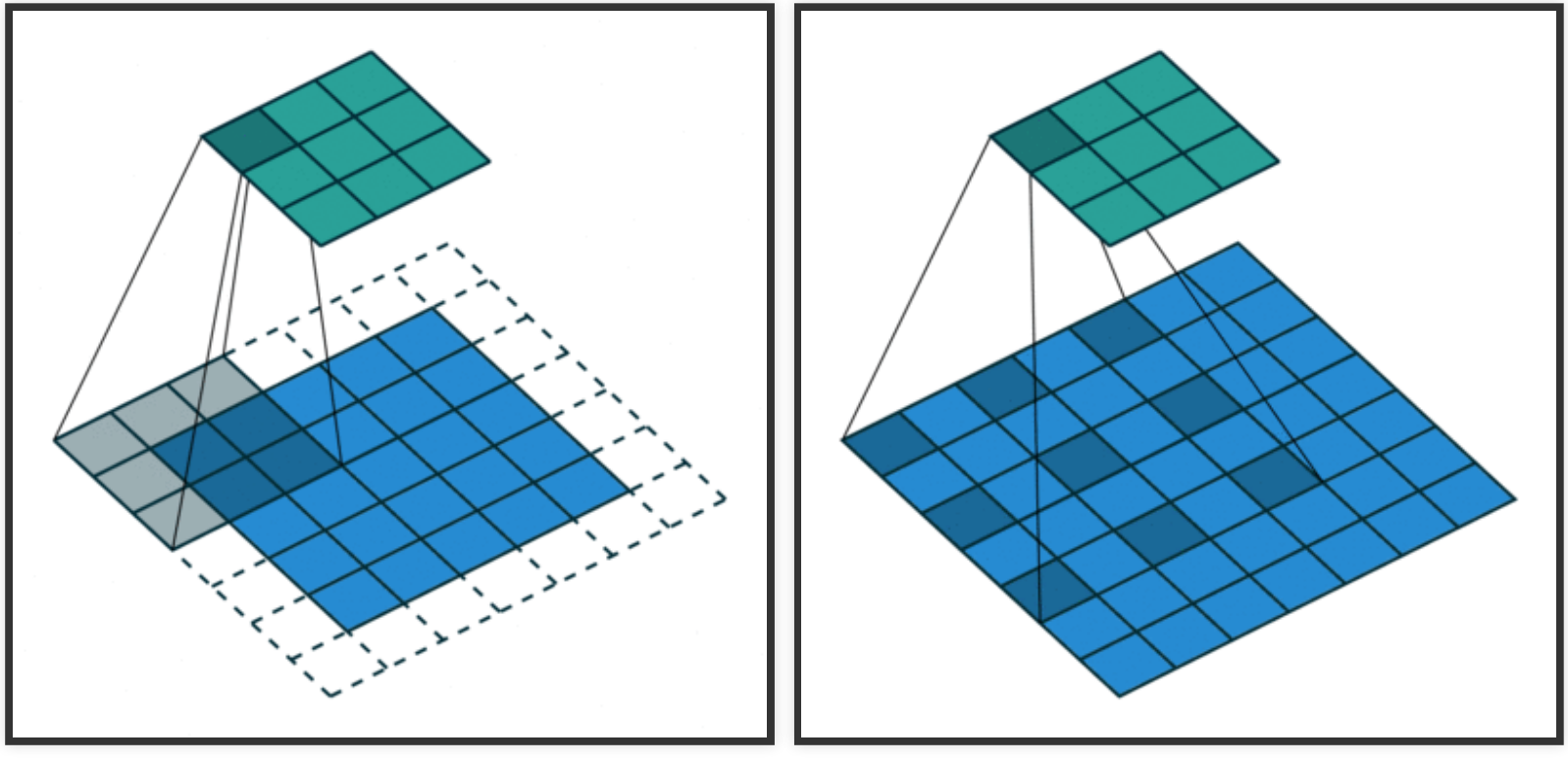
Caractéristiques

	Patch-based	**Context Encoder**	**Network**
Image size	**Any**	Fixed	**Any**
Local Conistsency	**Yes**	No	**Yes**
Semantics	No	**Yes**	**Yes**
Creates objects	No	**Yes**	**Yes**

Structure du réseau



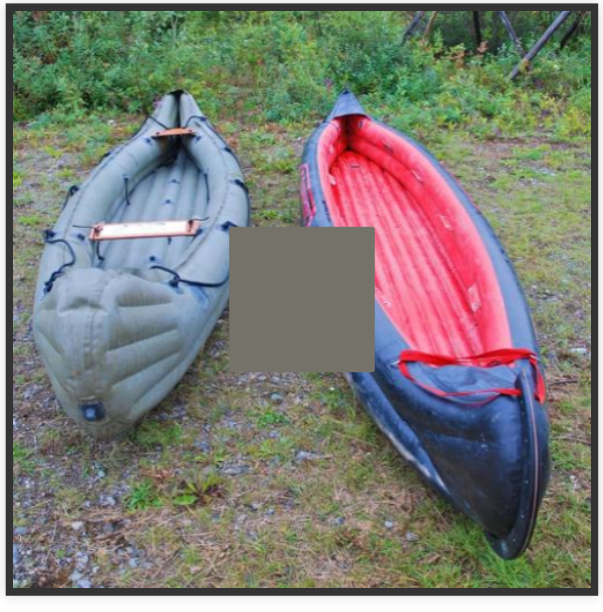
Dilated convolutions



Tests sur le réseau

Différents masques

Entrées


Différents masques

Sorties



Différents masques

Entrées



Différents masques

Sorties



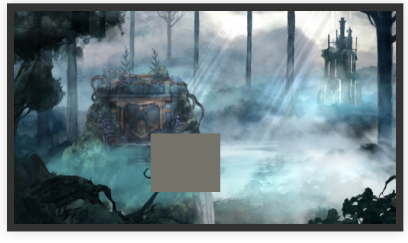
Zero-padding

Artefacts dus au 0-padding

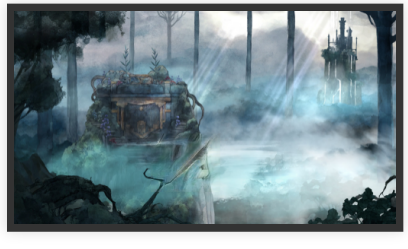
0 pixels	3 pixels	6 pixels
		
		

Images non naturelles








Images non naturelles



Architecture

```
nn.Sequential {
  [input -> (1) -> (2) -> (3) -> (4) -> (5) -> (6) -> (7) -> (8) -> (9) -> (10)
-> (11) -> (12) -> (13) -> (14) -> (15) -> (16) -> (17) -> (18) -> (19) -> (20)
-> (21) -> (22) -> (23) -> (24) -> (25) -> (26) -> (27) -> (28) -> (29) -> (30)
-> (31) -> (32) -> (33) -> (34) -> (35) -> (36) -> (37) -> (38) -> (39) -> (40)
-> (41) -> (42) -> (43) -> (44) -> (45) -> (46) -> (47) -> (48) -> (49) -> (50)
-> output]
  (1): nn.SpatialConvolution(4 -> 64, 5x5, 1,1, 2,2)
  (2): nn.SpatialBatchNormalization (4D) (64)
  (3): nn.ReLU
  (4): nn.SpatialConvolution(64 -> 128, 3x3, 2,2, 1,1)
  (5): nn.SpatialBatchNormalization (4D) (128)
  (6): nn.ReLU
  (7): nn.SpatialConvolution(128 -> 128, 3x3, 1,1, 1,1)
  (8): nn.SpatialBatchNormalization (4D) (128)
  (9): nn.ReLU
  (10): nn.SpatialConvolution(128 -> 256, 3x3, 2,2, 1,1)
  (11): nn.SpatialBatchNormalization (4D) (256)
  (12): nn.ReLU
  (13): nn.SpatialDilatedConvolution(256 -> 256, 3x3, 1,1, 1,1, 1,1)
  (14): nn.SpatialBatchNormalization (4D) (256)
  (15): nn.ReLU
  (16): nn.SpatialDilatedConvolution(256 -> 256, 3x3, 1,1, 1,1, 1,1)
  (17): nn.SpatialBatchNormalization (4D) (256)
  (18): nn.ReLU
  (19): nn.SpatialDilatedConvolution(256 -> 256, 3x3, 1,1, 2,2, 2,2)
  (20): nn.SpatialBatchNormalization (4D) (256)
  (21): nn.ReLU
  (22): nn.SpatialDilatedConvolution(256 -> 256, 3x3, 1,1, 4,4, 4,4)
  (23): nn.SpatialBatchNormalization (4D) (256)
  (24): nn.ReLU
  (25): nn.SpatialDilatedConvolution(256 -> 256, 3x3, 1,1, 8,8, 8,8)
  (26): nn.SpatialBatchNormalization (4D) (256)
  (27): nn.ReLU
  (28): nn.SpatialDilatedConvolution(256 -> 256, 3x3, 1,1, 16,16, 16,16)
  (29): nn.SpatialBatchNormalization (4D) (256)
  (30): nn.ReLU
  (31): nn.SpatialDilatedConvolution(256 -> 256, 3x3, 1,1, 1,1, 1,1)
  (32): nn.SpatialBatchNormalization (4D) (256)
  (33): nn.ReLU
  (34): nn.SpatialDilatedConvolution(256 -> 256, 3x3, 1,1, 1,1, 1,1)
  (35): nn.SpatialBatchNormalization (4D) (256)
  (36): nn.ReLU
  (37): nn.SpatialFullConvolution(256 -> 128, 4x4, 2,2, 1,1)
  (38): nn.SpatialBatchNormalization (4D) (128)
  (39): nn.ReLU
  (40): nn.SpatialConvolution(128 -> 128, 3x3, 1,1, 1,1)
  (41): nn.SpatialBatchNormalization (4D) (128)
  (42): nn.ReLU
  (43): nn.SpatialFullConvolution(128 -> 64, 4x4, 2,2, 1,1)
  (44): nn.SpatialBatchNormalization (4D) (64)
  (45): nn.ReLU
  (46): nn.SpatialConvolution(64 -> 32, 3x3, 1,1, 1,1)
  (47): nn.SpatialBatchNormalization (4D) (32)
  (48): nn.ReLU
  (49): nn.SpatialConvolution(32 -> 3, 3x3, 1,1, 1,1)
  (50): nn.Sigmoid
}
```

Résultats : couches de convolution (1,4)







Résultats : couches de convolution (7,10)







Résultats : couches de convolution (13,16)







Résultats : couches de convolution (46,49)







Résultats : couches de convolution dilatées (19,22)







Résultats : couche de convolution dilatée (25,28)







Résultats : SpatialBatchNormalization (2,11)







Résultats : SpatialBatchNormalization (20,29)







Résultats : SpatialBatchNormalization (38,47)







Résultats : bruit sur les couches batch









Résultats : bruit sur les couches convolutives





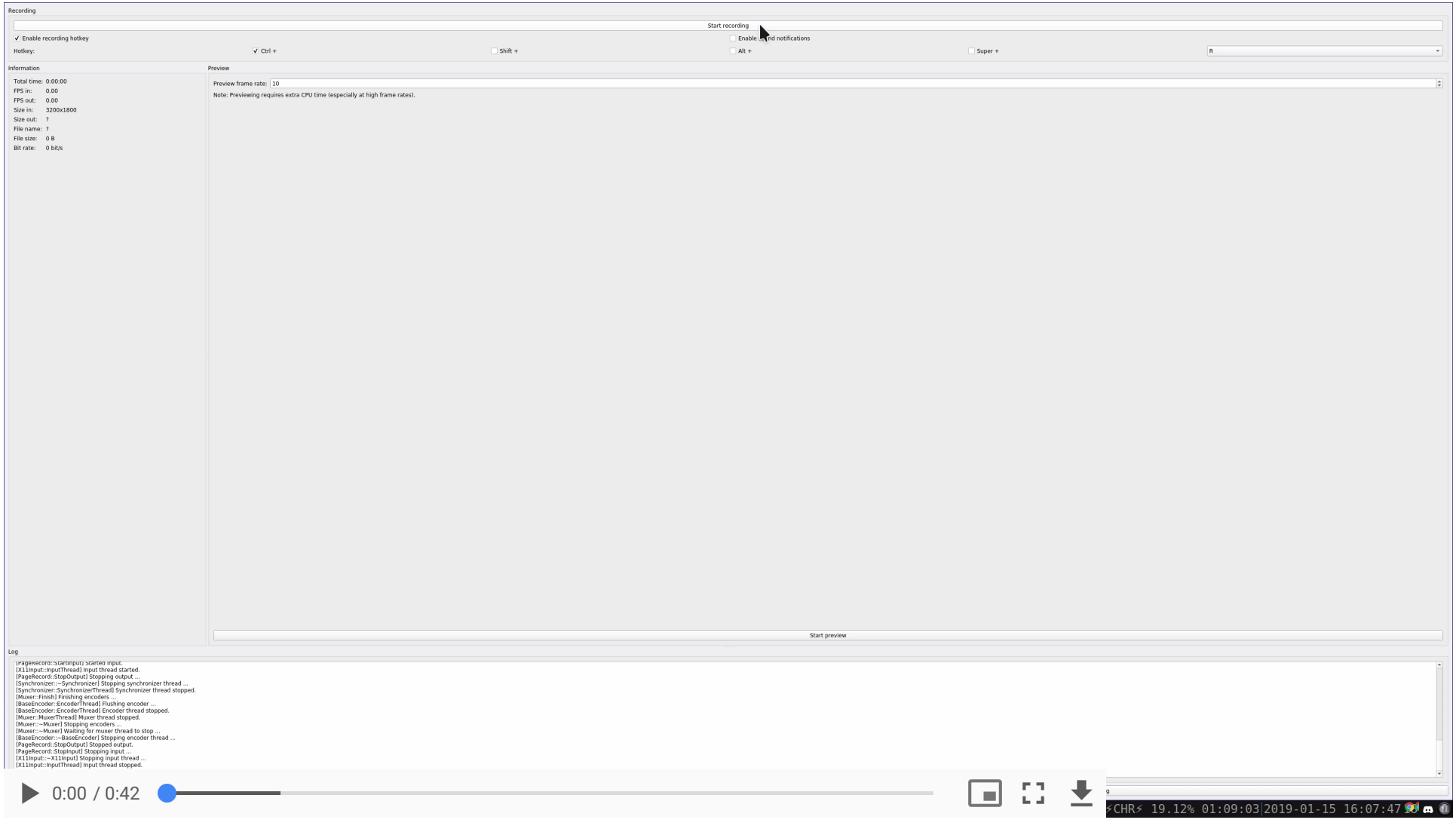




Fin

Outil pour les créer des masques

Outil pour les créer des masques



Resultat de l'article sur les visages:

Test avec mon visage!

Entrées

Test avec mon visage!

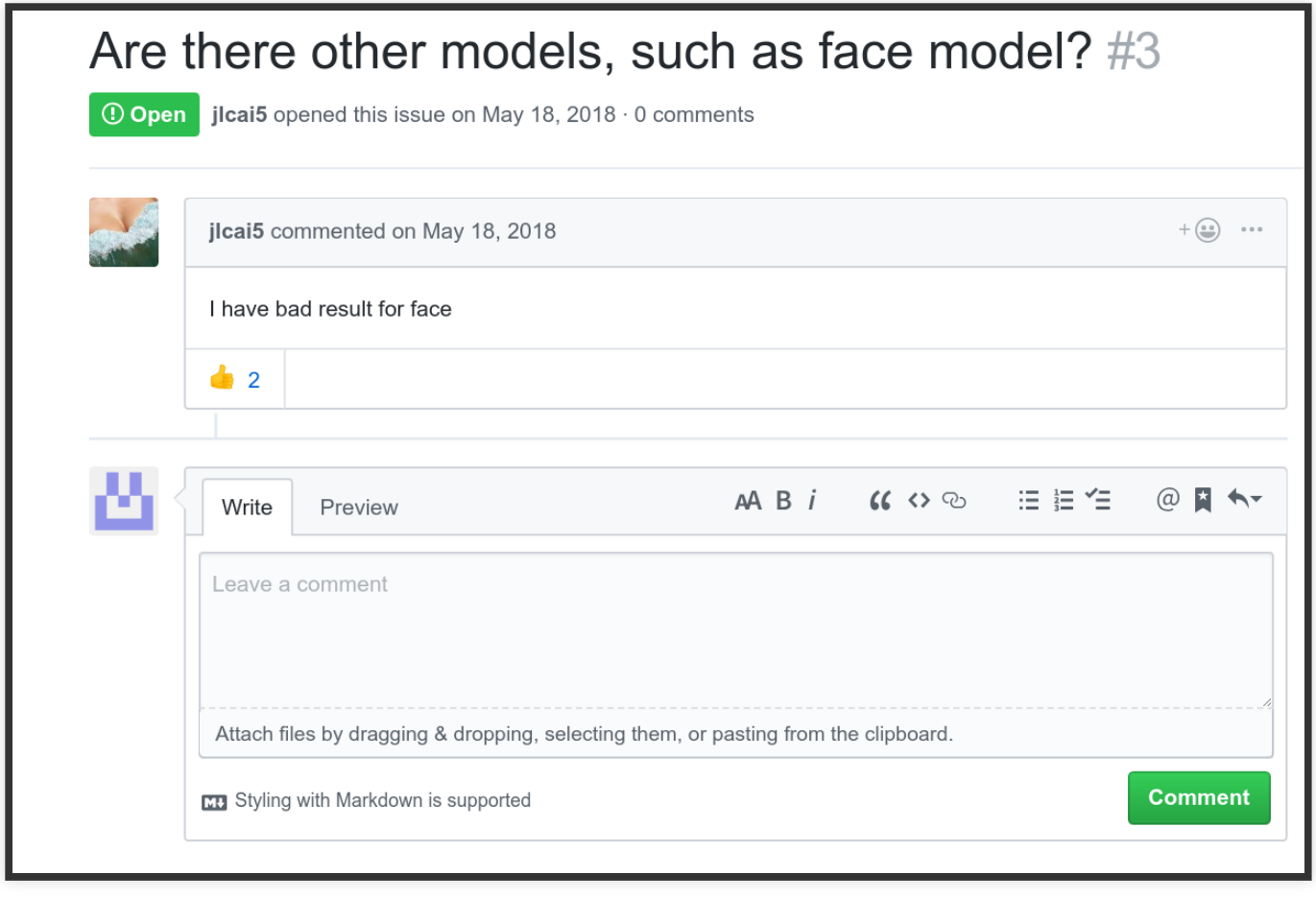
Sorties

Dataset utilisés

- Places2/ImageNet => entrainé globalement
- CelebA, CMP Facade dataset => finetunning

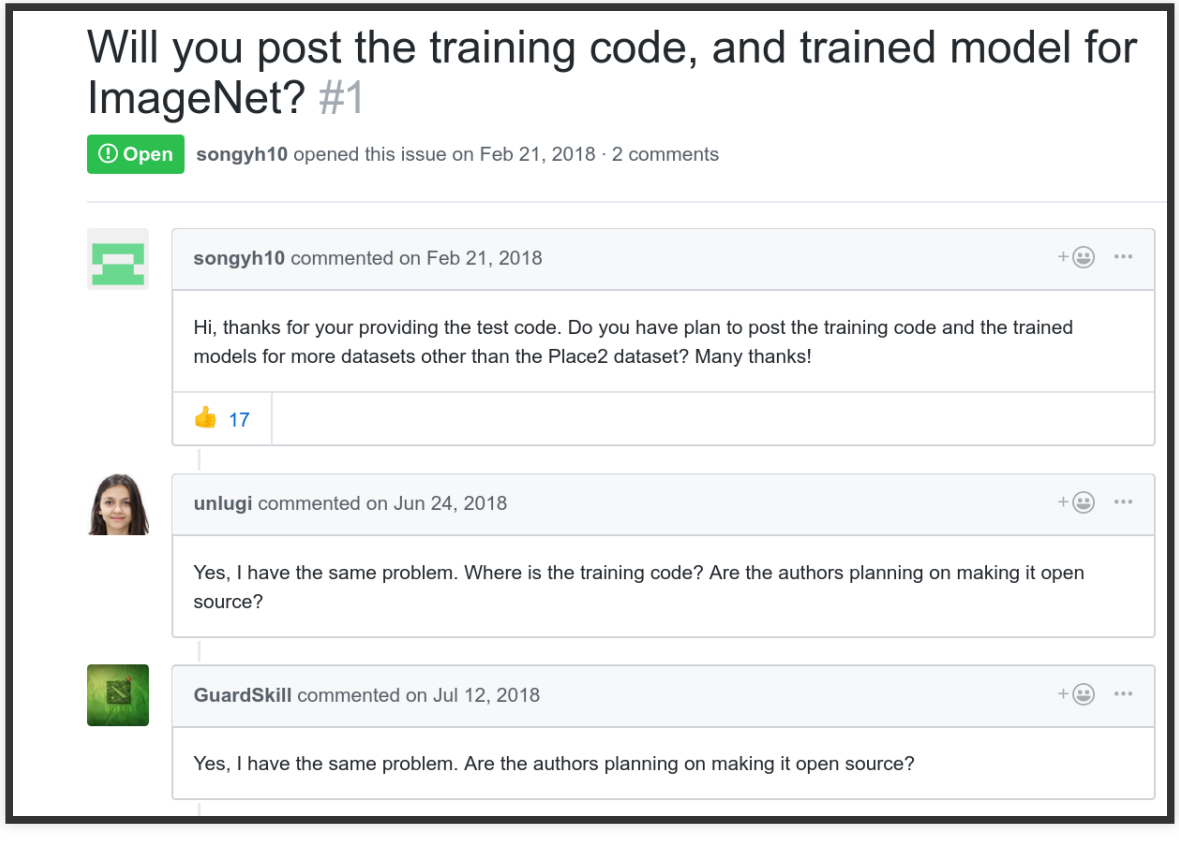
Visages

Pas accès à leur model ☹



Expérimentations sur le réseau

Entraînement du réseau



Will you post the training code, and trained model for ImageNet? #1

Open **songyh10** opened this issue on Feb 21, 2018 · 2 comments

songyh10 commented on Feb 21, 2018

Hi, thanks for your providing the test code. Do you have plan to post the training code and the trained models for more datasets other than the Place2 dataset? Many thanks!

17

unlugi commented on Jun 24, 2018

Yes, I have the same problem. Where is the training code? Are the authors planning on making it open source?

GuardSkill commented on Jul 12, 2018

Yes, I have the same problem. Are the authors planning on making it open source?

Implémentation

Première étape (lua)

- Installer torch7 lua (en mode gpu)
- Comprendre leur code
- Rajouter des parametres

Implementation

Deuxième étape (python)

- Gérer le système de fichiers
- Lancer les tests à la suite en série
- Déplacer les fichiers au bon endroit

Tests effectués:

- Plus de 16000 images inpaintés
- Environ 2Go d'output d'images

Credit

- https://towardsdatascience.com/review-dilated-convolution-semantic-segmentation-9d5a5bd768f5
- https://arxiv.org/pdf/1604.07379.pdf