Réseaux de neurones

IFT 780

Visualisation

Par

Pierre-Marc Jodoin

Jusqu’à présent : réseaux entraînés

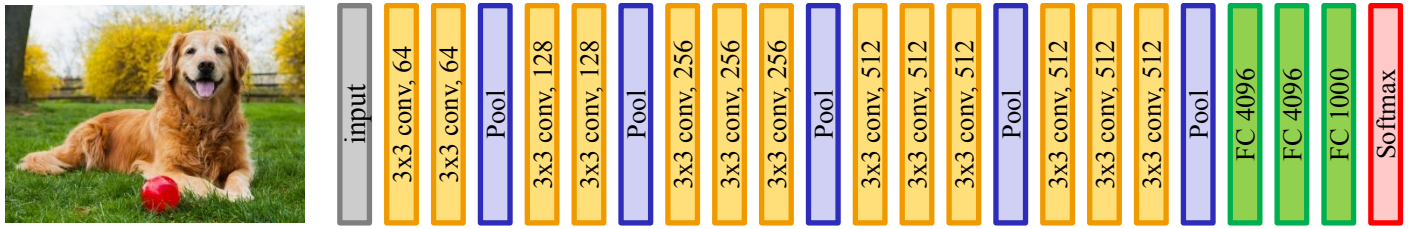
Classification

Génération

Comment visualiser ce qu'un réseau a appris?

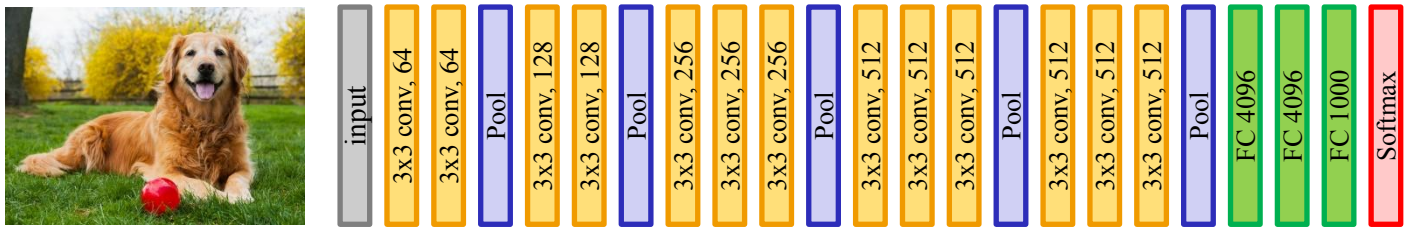
Comment visualiser ce qu'un réseau a appris?

VGG16



p(chien) = 0.9

VGG16



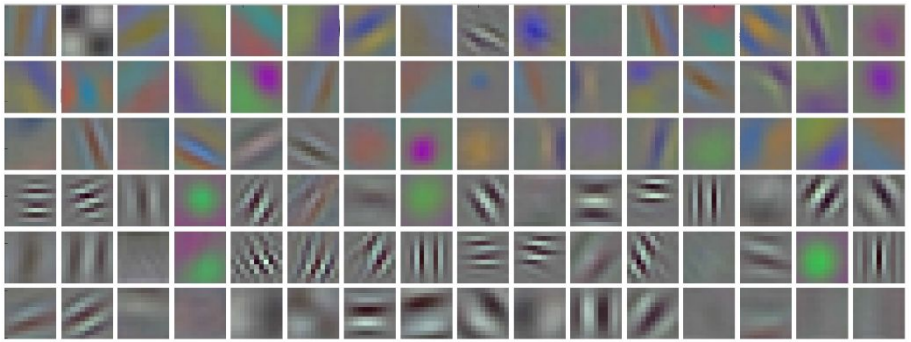
Prédiction
1000 valeurs

3 options:

1. Visualiser les **filtres**
2. Visualiser les **cartes d'activation** (neurones en sortie d'une couche)
3. **Construire/sélectionner des images** qui excitent certains neurones

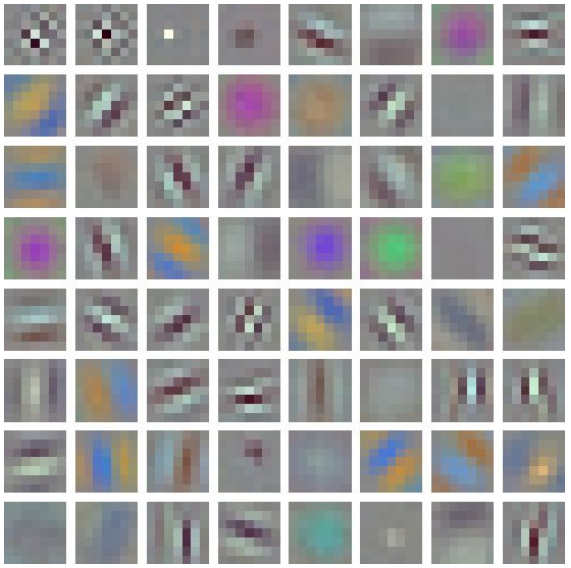
Visualiser les filtres de la **première couche**

Peu importe la structure du réseau, les premiers filtres sont **toujours** des filtres de détection de contours, de coins et de « blobs ».
Ressemble à des filtre de Gabor



CaffeNet

conv1 = 11x11x3x96 filtres



ResNet

conv1 = 7x7x3x64 filtres

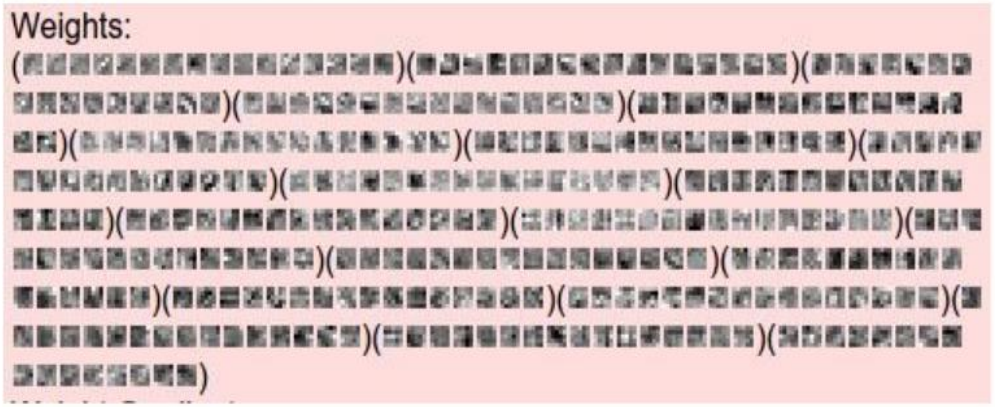
Brachmann, C. Redies, “Using Convolutional Neural Network Filters to Measure Left-Right Mirror Symmetry in Images”, Symmetry, 12(8), 2016
Yani Ioannou, “Structural Priors in Deep Neural Networks”, 2017

Visualiser les **autres couches n'est pas très intéressant**

Filtre couche 2
combine les 16 cartes d'activation de la couche précédente

Interprétation difficile

Couche 1
16 filtres de taille
7x7x3



Couche 2
20 filtres de taille
7x7x16

Couche 3
20 filtres de taille
7x7x20

cs231n.github.io
cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html

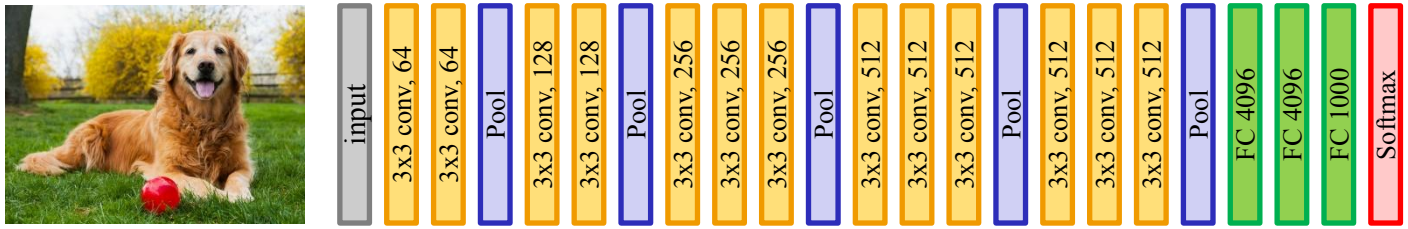
On tire rarement grand-chose à visualiser des filtres. Il faut donc une autre solution:

Options restantes:

1. Visualiser les **filtres**
2. Visualiser les **cartes d'activation** (neurones en sortie d'une couche)
3. **Construire/sélectionner une image** qui excite certains neurones

Visualiser les neurones de l'avant-dernière couche.

VGG16



Prédiction de 1000 valeurs

4096 neurones
(1 image = 1 point dans un espace à 4096 dim)

Ce qu'on peut faire:

1. **Projeter** N images vers l'espace à 4096 dimensions
2. Visualiser les **plus proches voisins**
3. Visualiser l'espace avec un algo de **réduction de la dimensionnalité**.

AlexNet

6 plus proches voisins de 5 images de référence

Distance euclidienne dans l'espace à 4096 dim

Conclusion : l'espace semble être **localement linéaire**

avec une **organisation sémantiques**

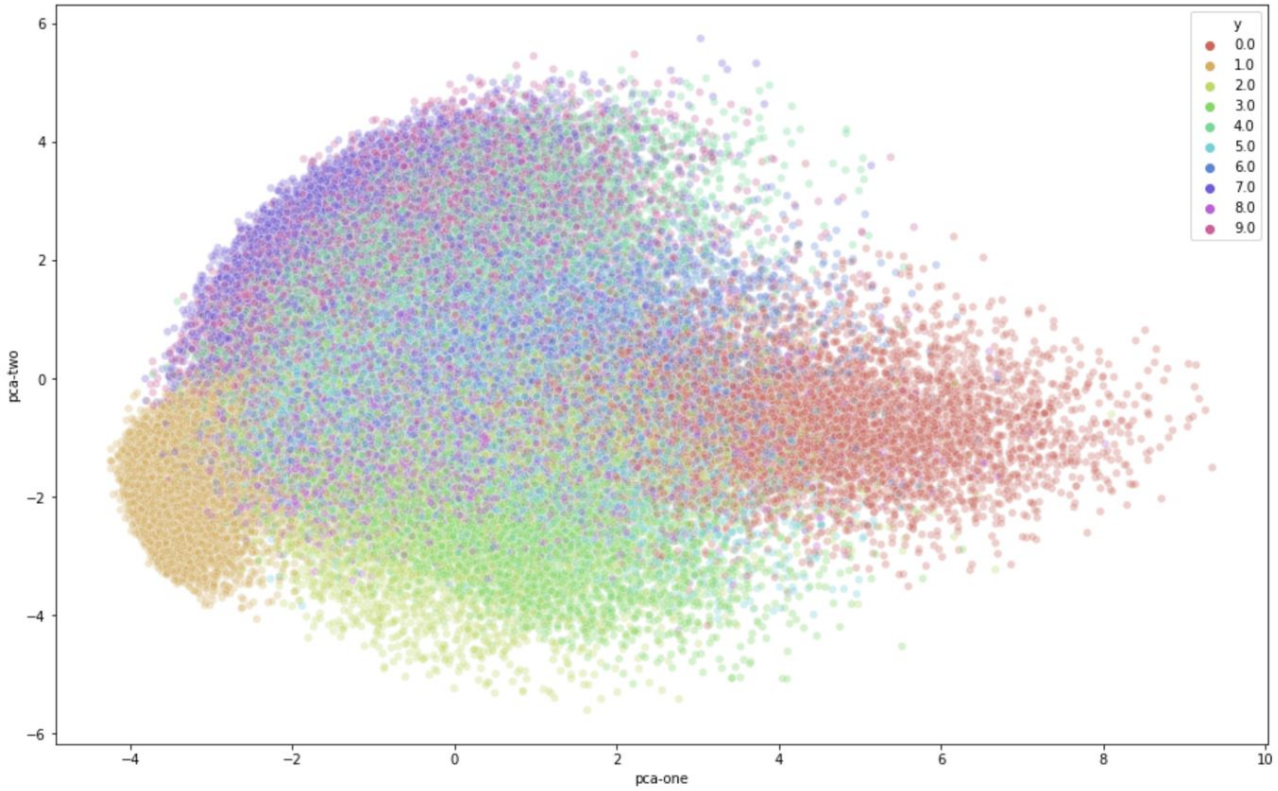
Images de référence

6 plus proches voisins

Krizhevsky et al, "ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012.

Visualiser l'espace à 4096 dim

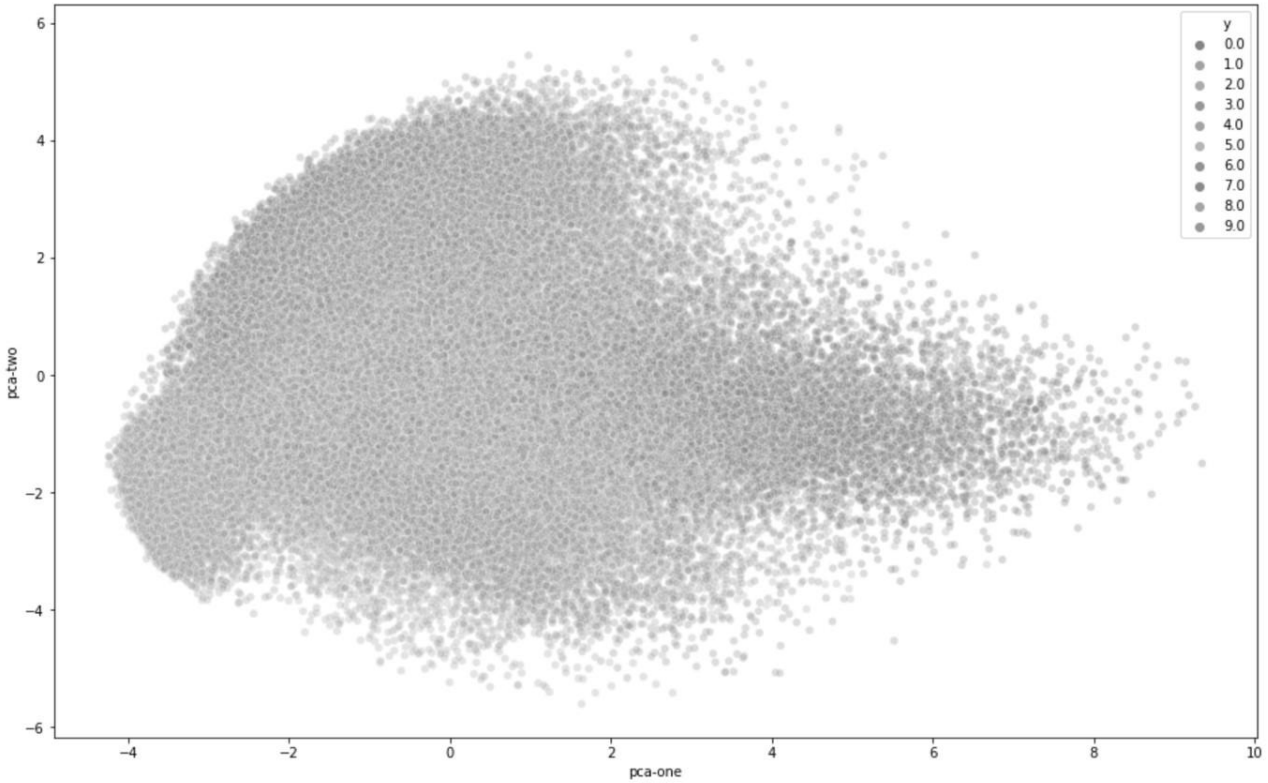
MNIST



https://towardsdatascience.com/visualising-high-dimensional-datasets-using-pca-and-t-sne-in-python-8ef87e7915b

Visualiser l'espace à 4096 dim

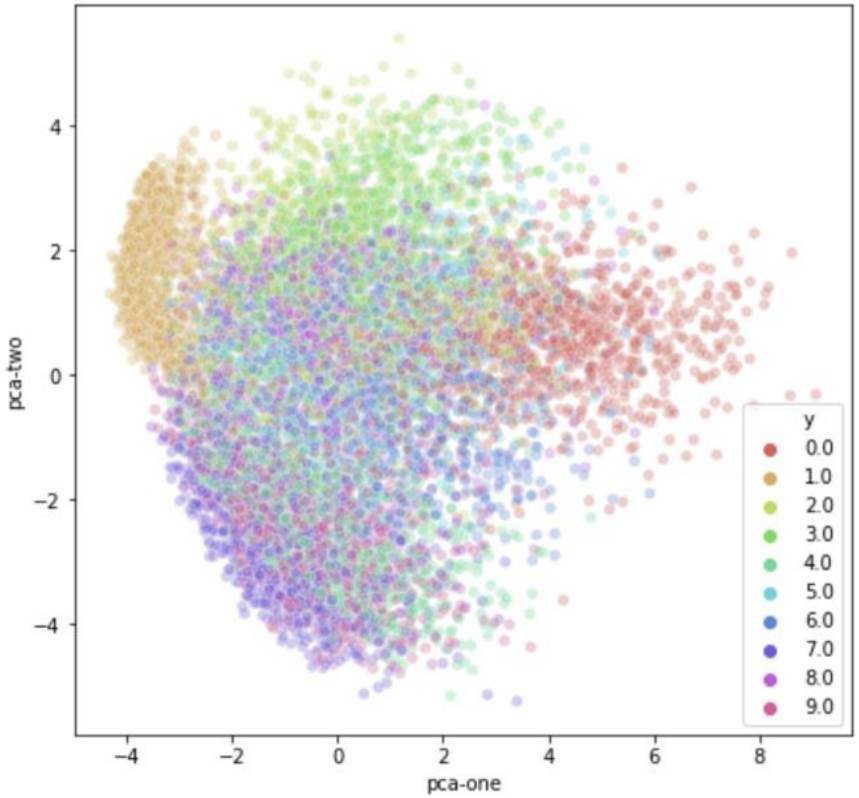
MNIST

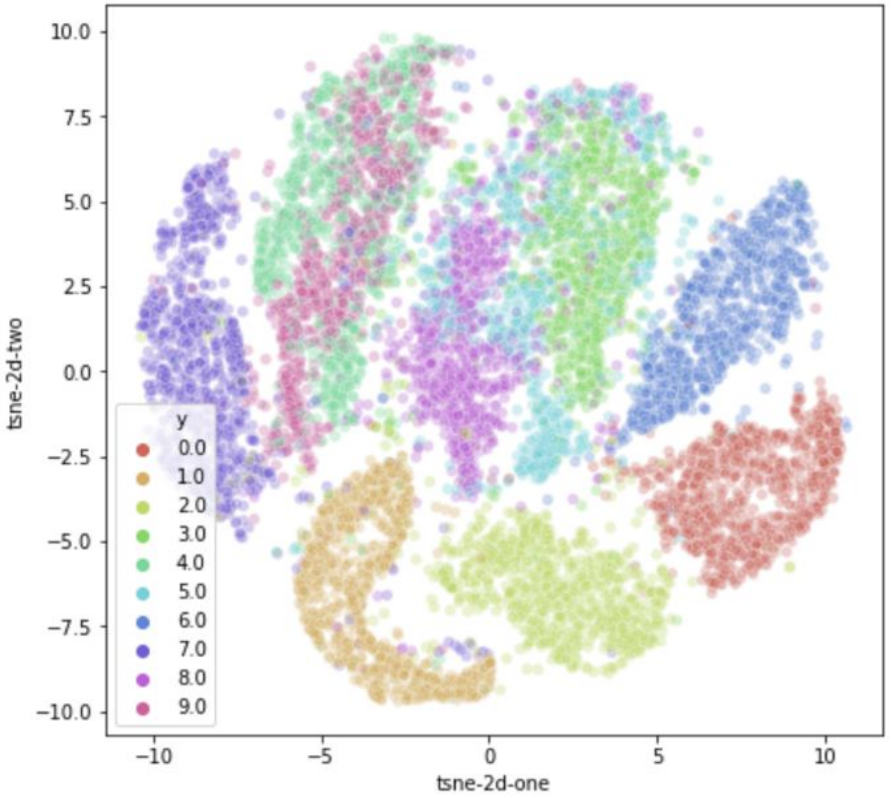


https://towardsdatascience.com/visualising-high-dimensional-datasets-using-pca-and-t-sne-in-python-8ef87e7915b

Visualiser l'espace à 4096 dim

MNIST

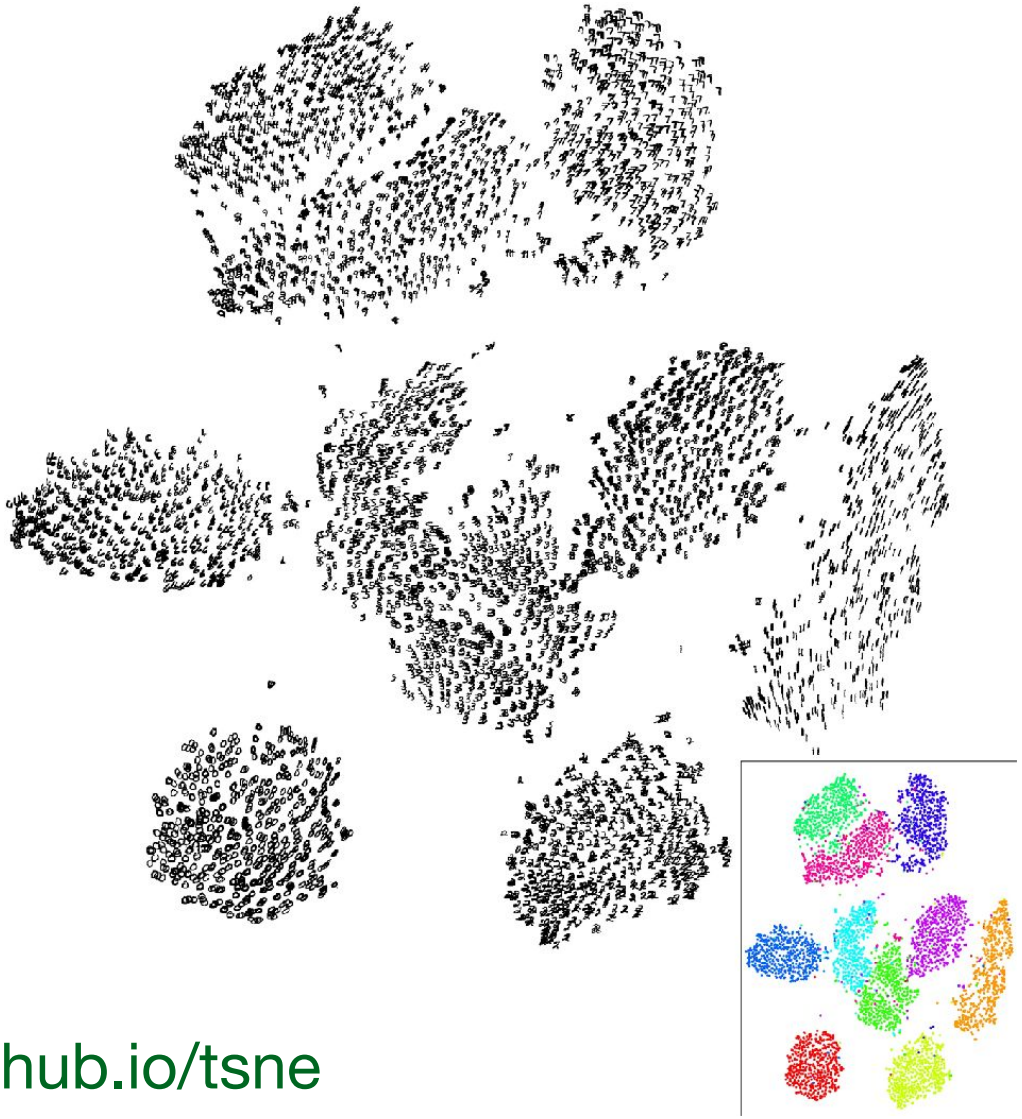




https://towardsdatascience.com/visualising-high-dimensional-datasets-using-pca-and-t-sne-in-python-8ef87e7915b

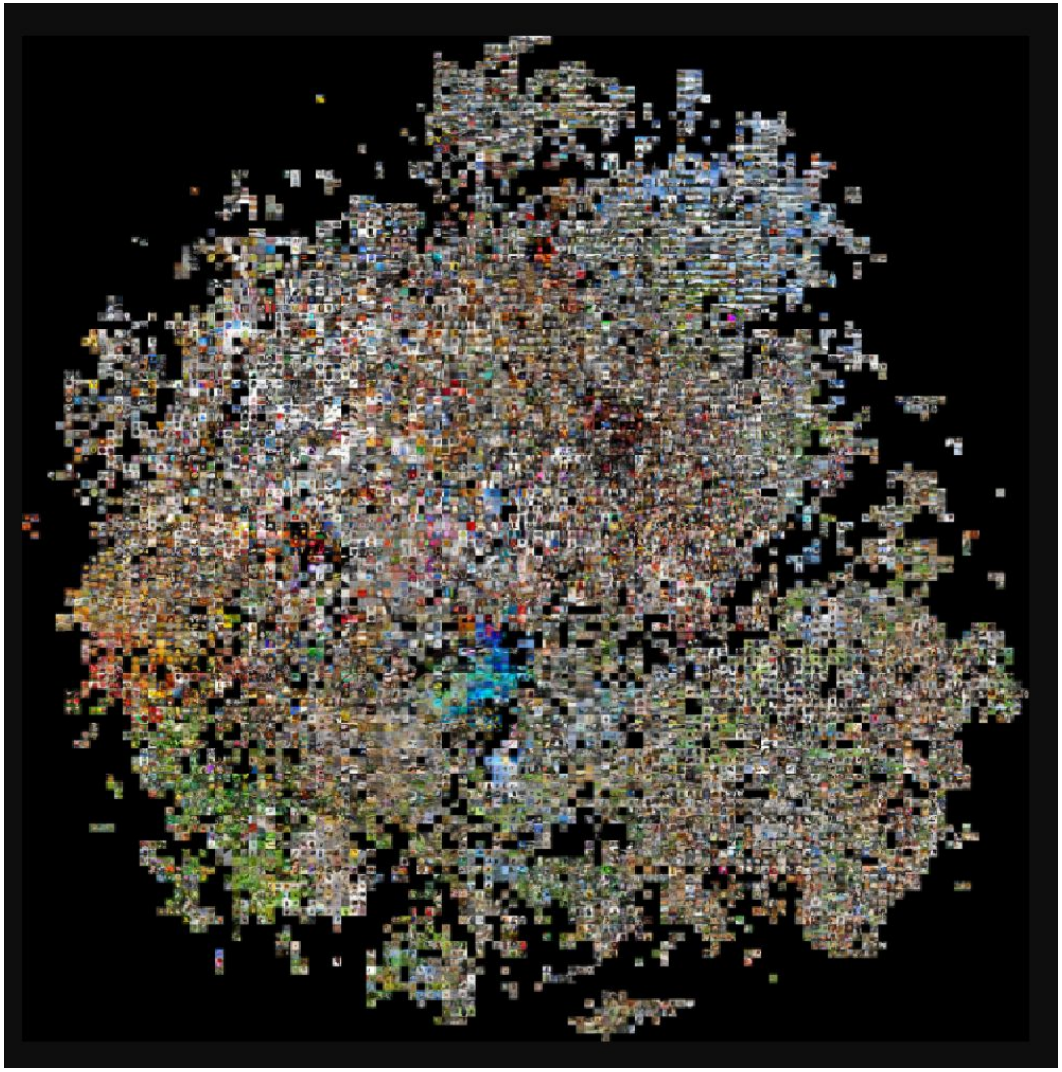
Visualiser l'espace à 4096 dim

MNIST



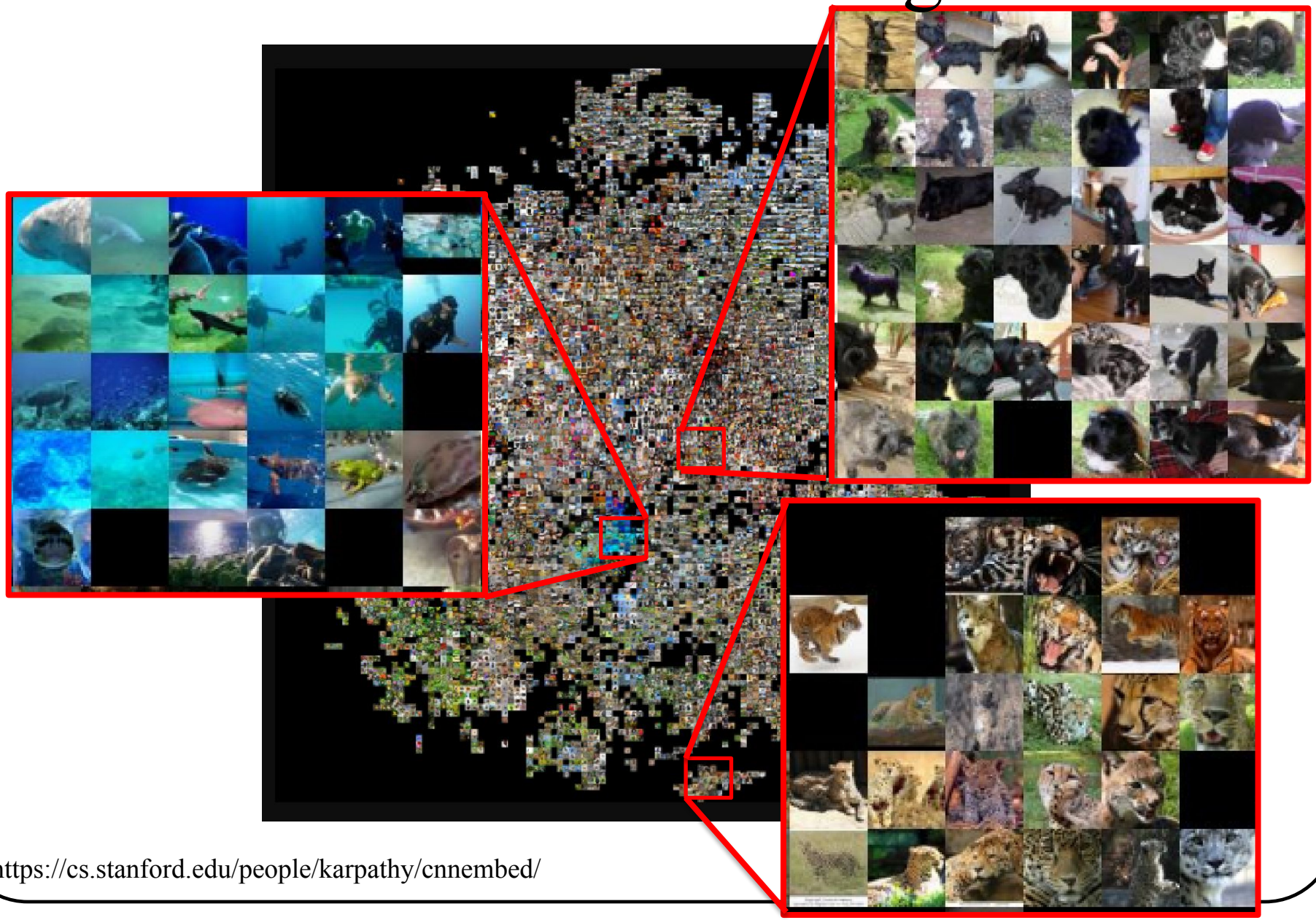
lvdmaaten.github.io/tsne

AlexNet + tsne + ImageNet



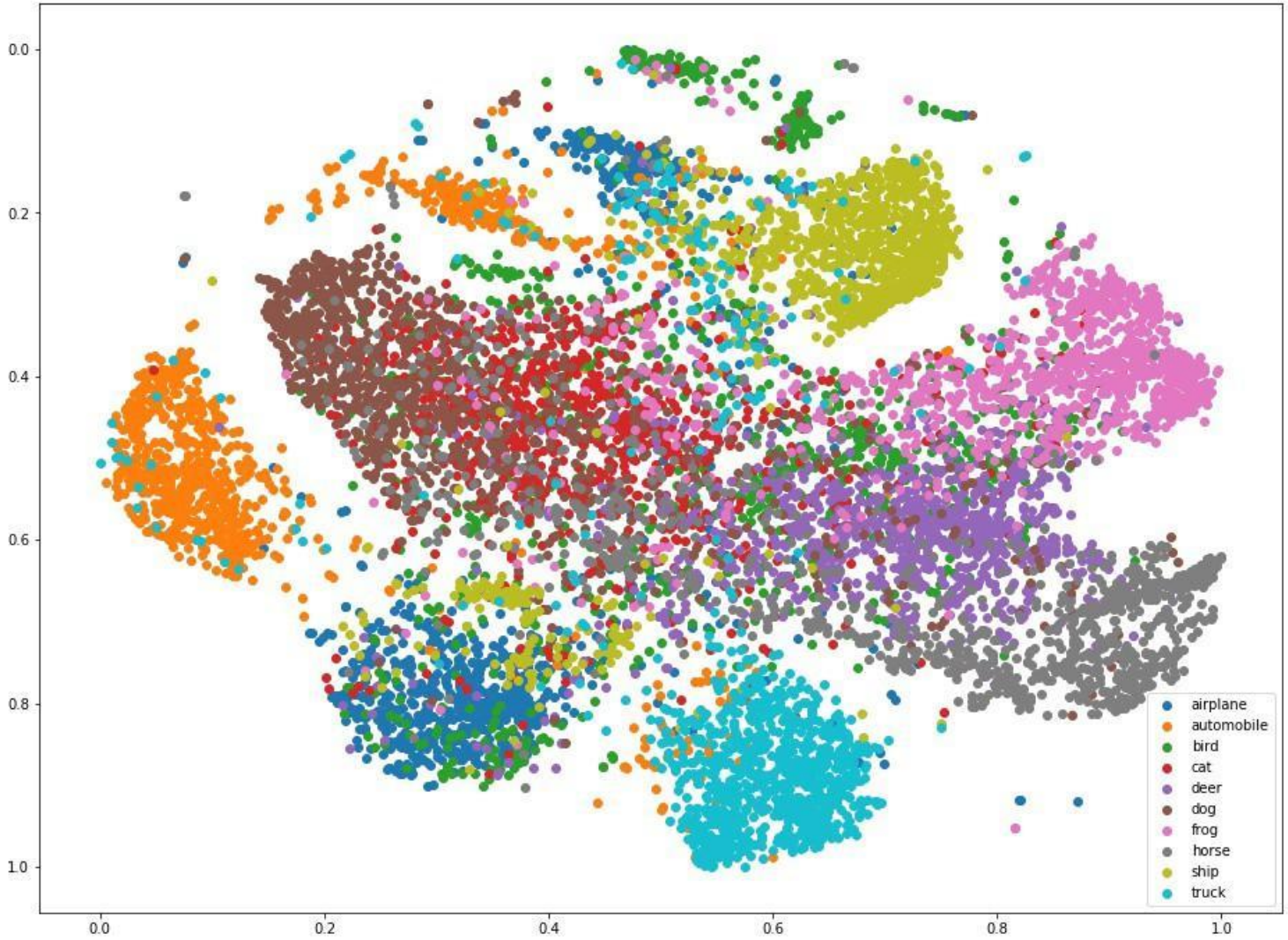
https://cs.stanford.edu/people/karpathy/cnnembed/

AlexNet + tsne + ImageNet

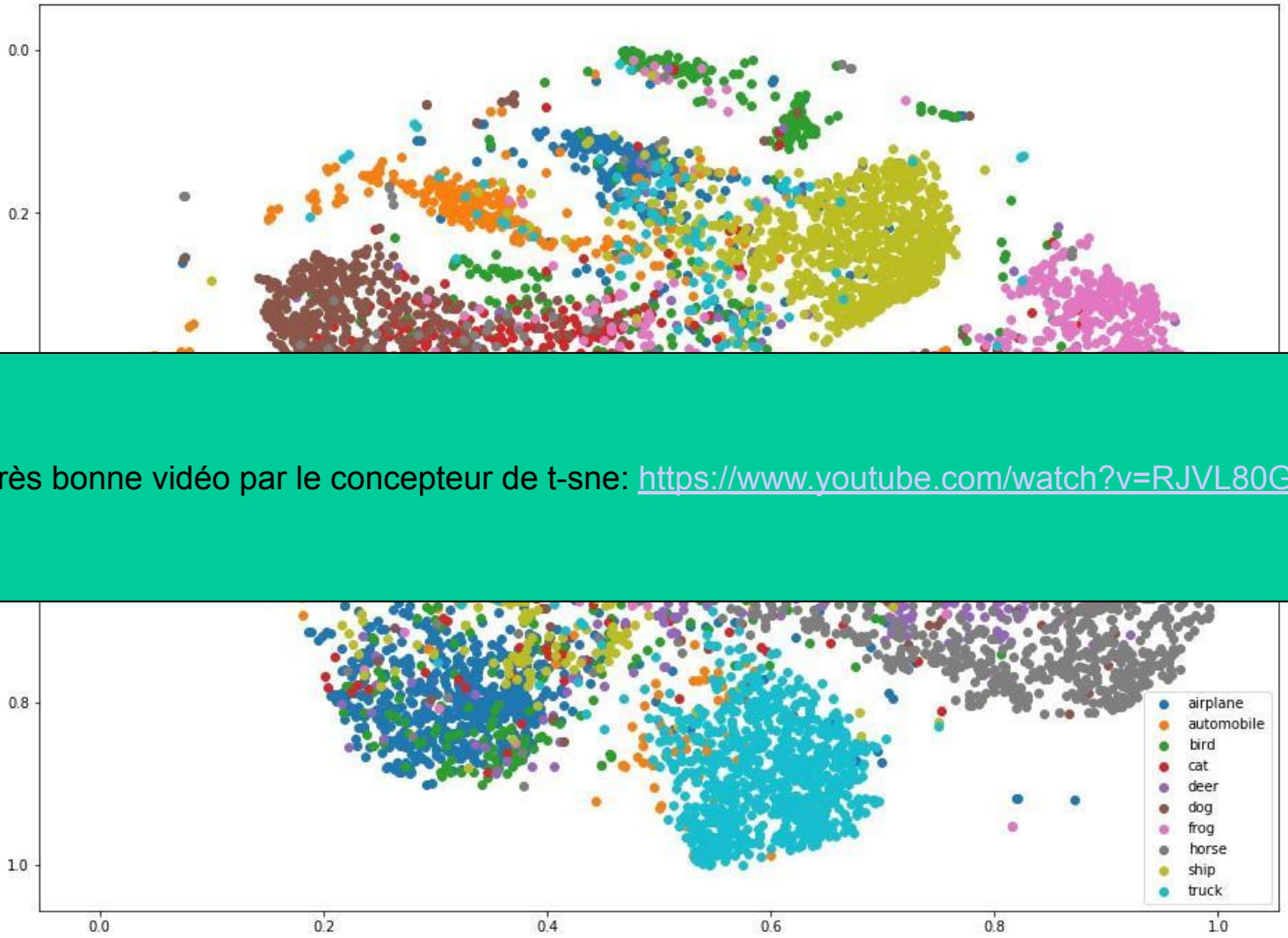


https://cs.stanford.edu/people/karpathy/cnnembed/

AlexNet + tsne + CIFAR10



AlexNet + tsne + CIFAR10



Très bonne vidéo par le concepteur de t-sne: https://www.youtube.com/watch?v=RJVL80Gg3lA

Visualiser les cartes d'activation (entre l'entrée et la sortie)

Ici la 151^{e} **carte d'activation** de la couche conv5 illustrées comme des images en niveau de gris de taille 13x13.

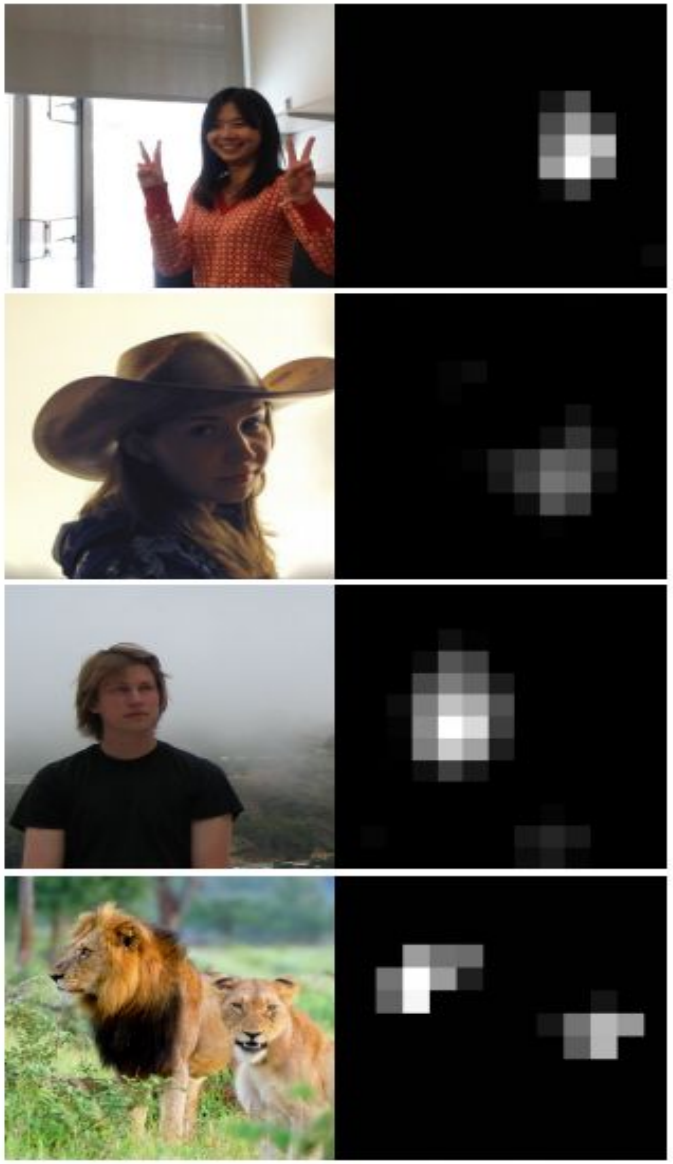


Figure 2. A view of the 13×13 activations of the 151st channel on the conv5 layer of a deep neural network trained on ImageNet, a

Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.

Visualiser les cartes d'activation (entre l'entrée et la sortie)

Ici la 151e **d'activatio**
couche cor
comme des
niveau de g
13x13.

Conclusion:

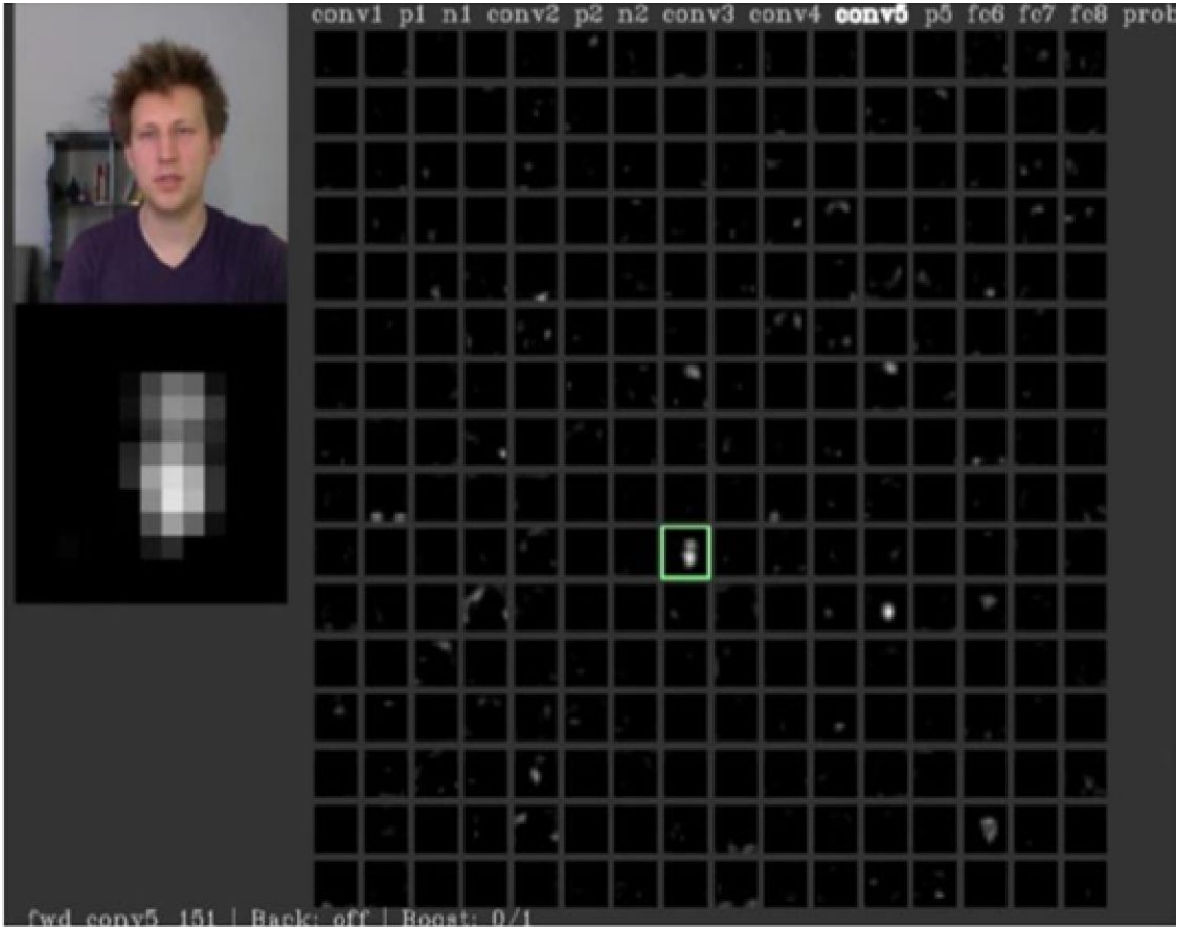
les neurones de cette carte d'activation se spécialisent dans la détection de visages

Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.

Figure 2. A view of the 13×13 activations of the 151st channel on the conv5 layer of a deep neural network trained on ImageNet, a

Visualiser les cartes d'activation

Ici les **256 cartes d'activation** de la couche conv5 illustrées comme des images en niveau de gris de taille 13x13.

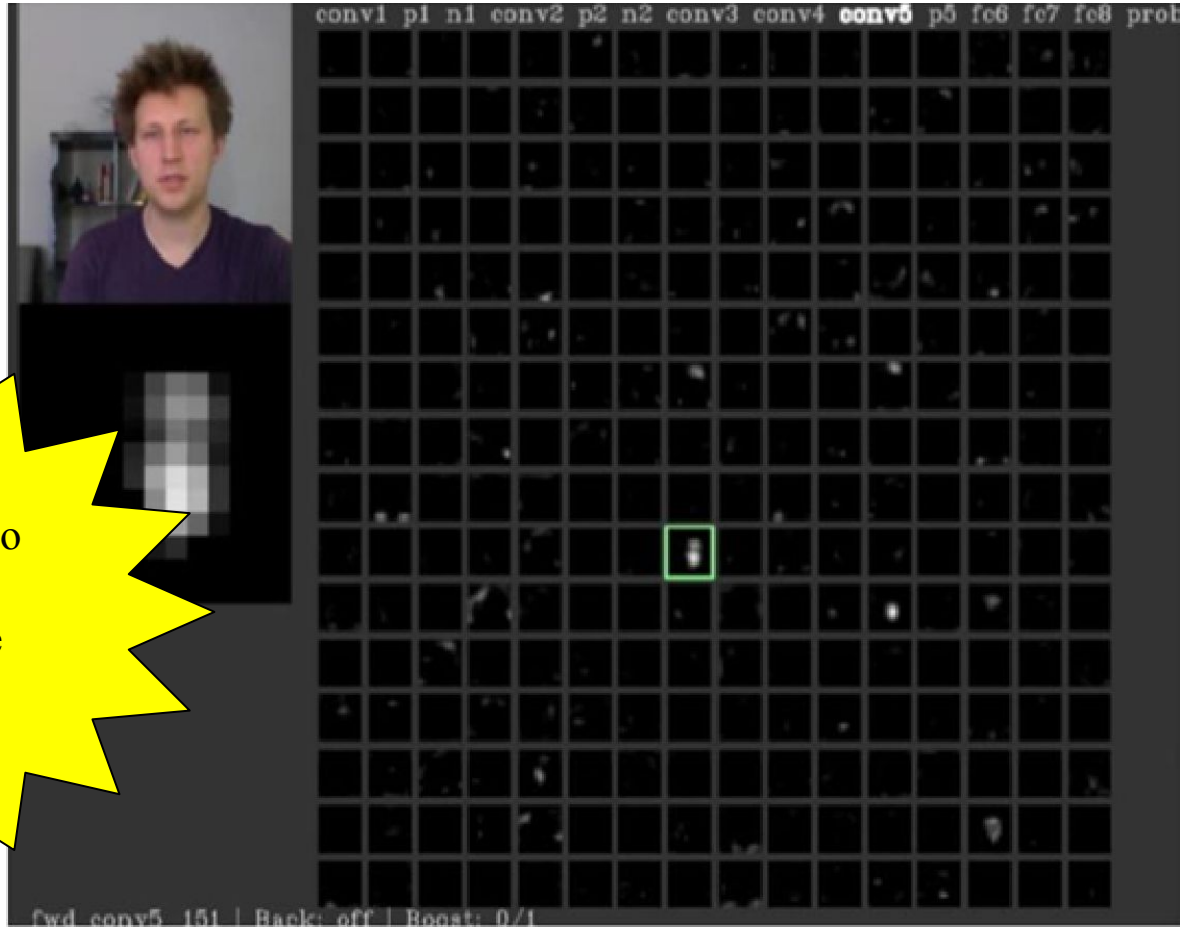


https://www.youtube.com/watch?v=AgkfIQ4IGaM
https://github.com/yosinski/deep-visualization-toolbox

Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.

Visualiser les cartes d'activation

Ici les **256 cartes d'activation** de la couche conv5 comme de niveau d 13x13.



Excellente vidéo sur la visualisation de filtres

https://www.youtube.com/watch?v=AgkfIQ4IGaM

https://github.com/yosinski/deep-visualization-toolbox

Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.

Visualisation par occultation

[Zeiler,Fergus 2014]

Comment identifier dans cette image ce qui permet au réseau de prédire la classe « chien »?

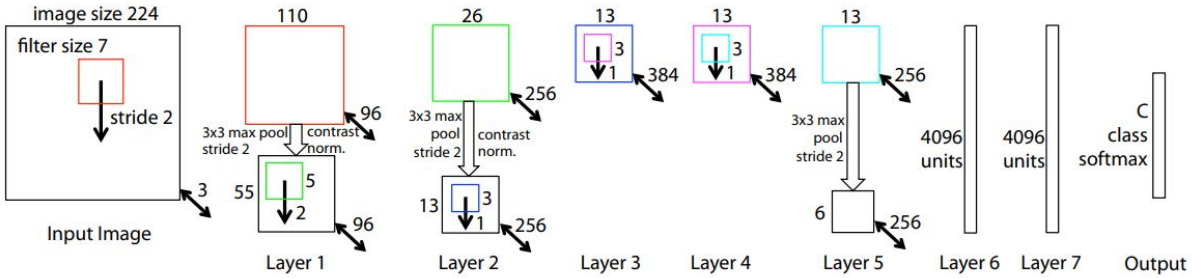
Zeiler and Fergus, “Visualizing and Understanding Convolutional Networks”, ECCV 2014

Visualisation par occultation

[Zeiler,Fergus 2014]

Solution : occulter des zones de l'image afin d'en constater l'impact

ZF-Net



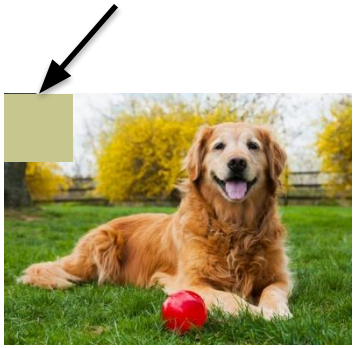
'chien .91'

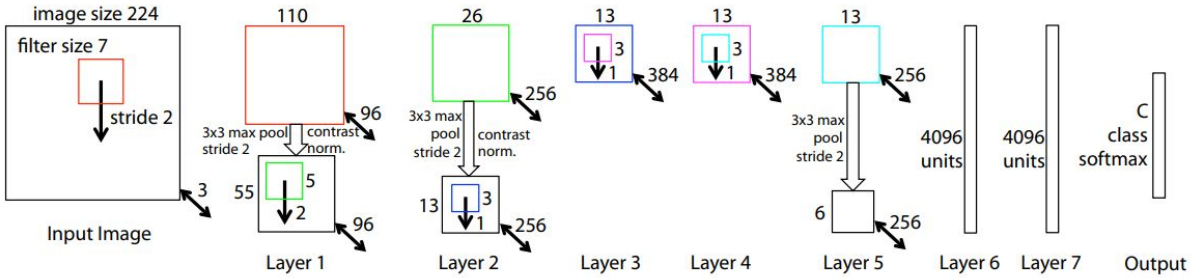
Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

Visualisation par occultation

[Zeiler,Fergus 2014]

Solution : occulter des zones de l'image afin d'en constater l'impact





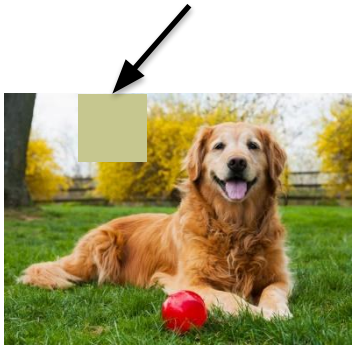
'chien .91'

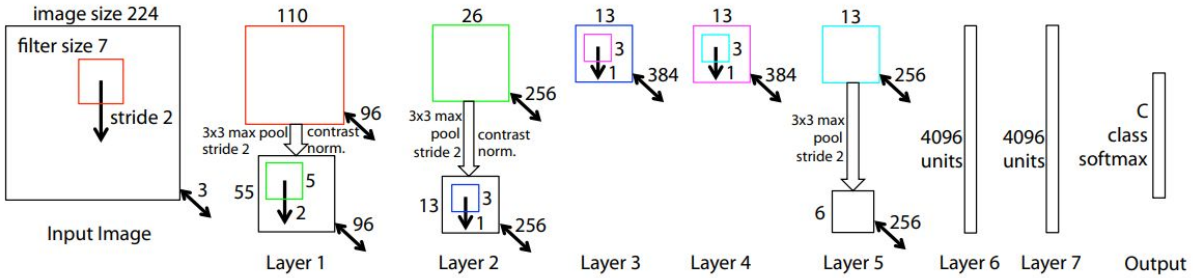
Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

Visualisation par occultation

[Zeiler,Fergus 2014]

Solution : occulter des zones de l'image afin d'en constater l'impact





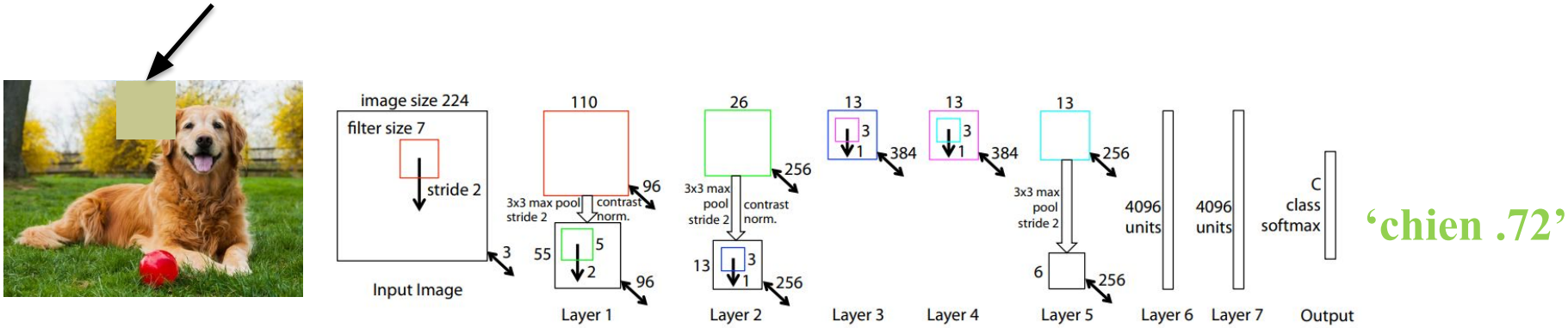
'chien .91'

Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

Visualisation par occultation

[Zeiler,Fergus 2014]

Solution : occulter des zones de l'image afin d'en constater l'impact

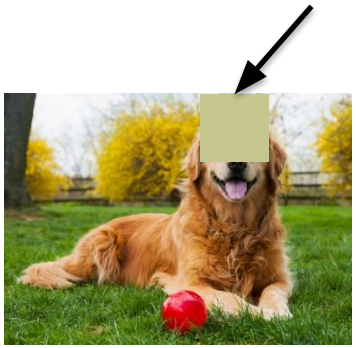


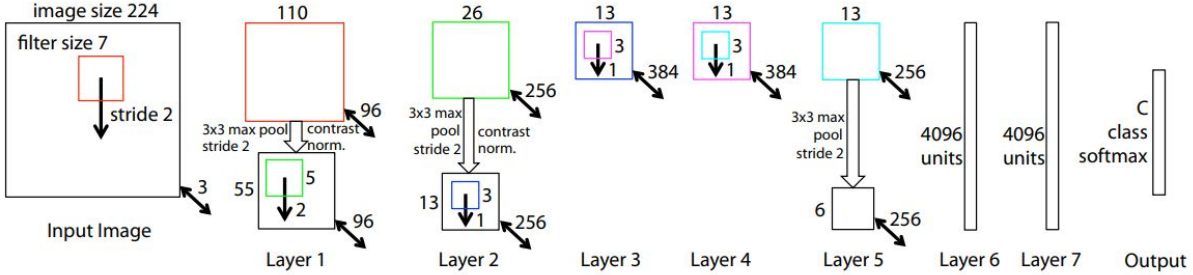
Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

Visualisation par occultation

[Zeiler,Fergus 2014]

Solution : occulter des zones de l'image afin d'en constater l'impact





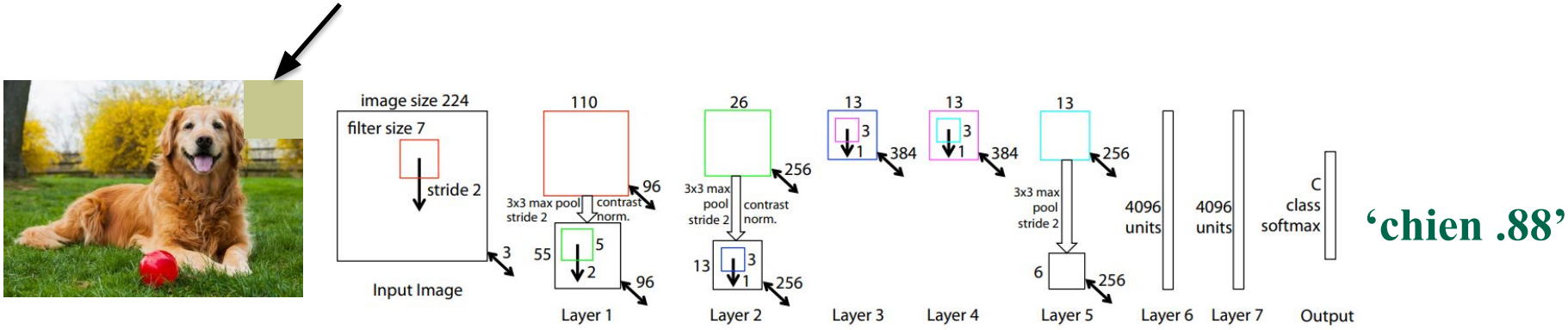
'chien .31'

Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

Visualisation par occultation

[Zeiler,Fergus 2014]

Solution : occulter des zones de l'image afin d'en constater l'impact



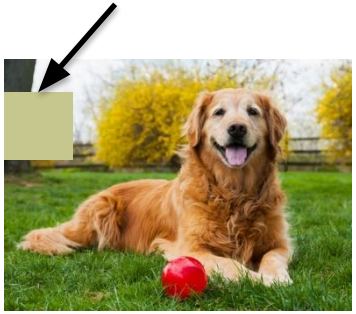
'chien .88'

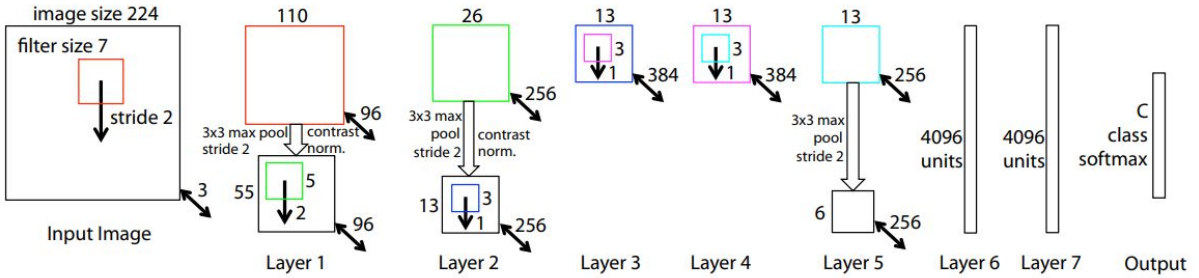
Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

Visualisation par occultation

[Zeiler,Fergus 2014]

Solution : occulter des zones de l'image afin d'en constater l'impact





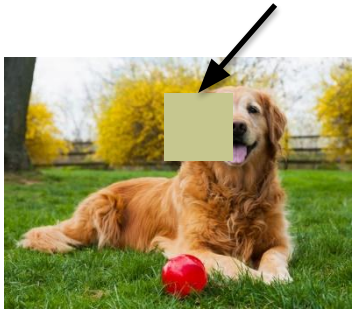
'chien .9'

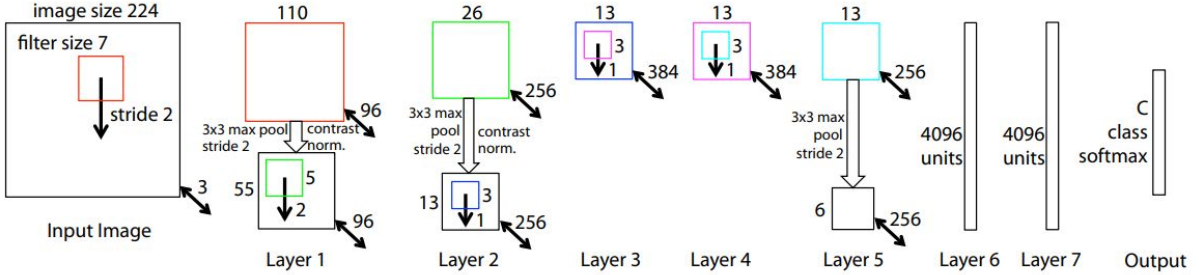
Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

Visualisation par occultation

[Zeiler,Fergus 2014]

Solution : occulter des zones de l'image afin d'en constater l'impact





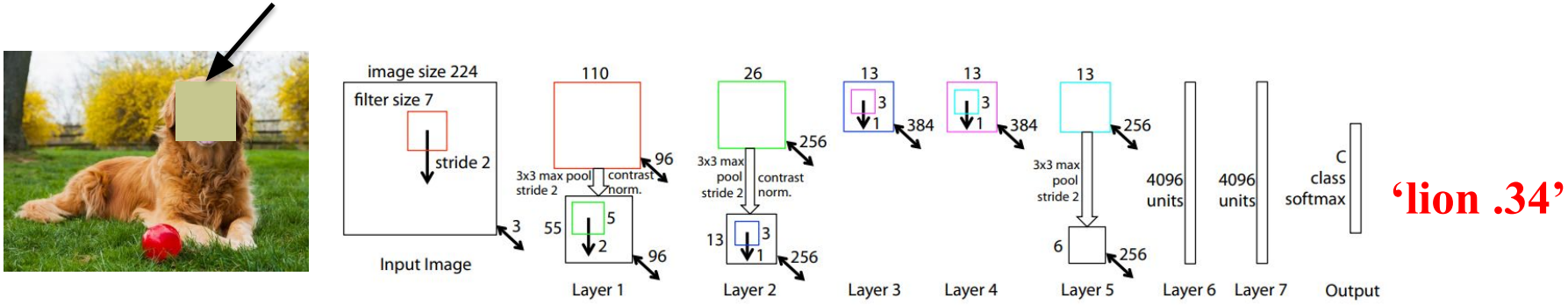
'chien .2'

Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

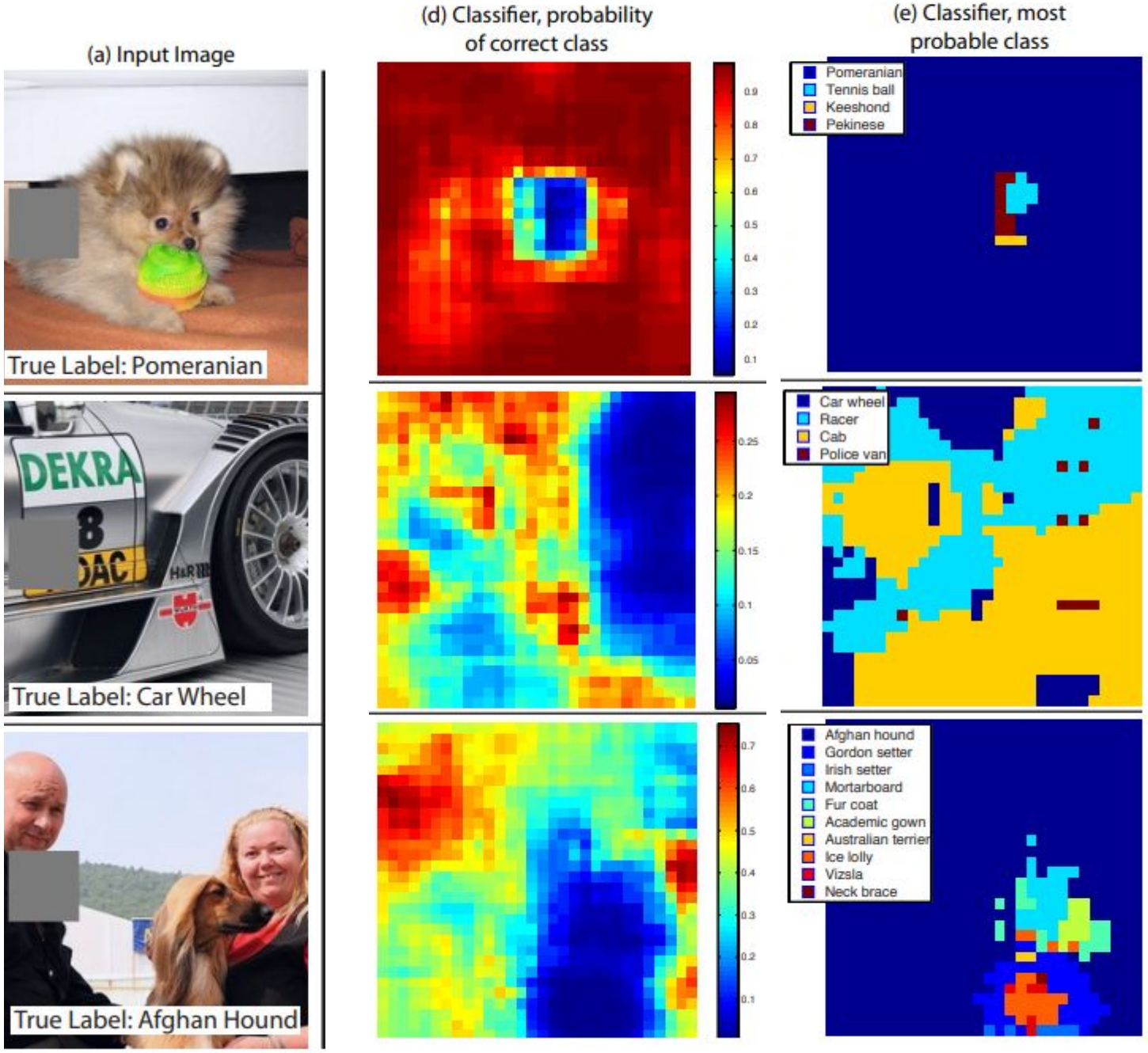
Visualisation par occultation

[Zeiler,Fergus 2014]

Solution : occulter des zones de l'image afin d'en constater l'impact

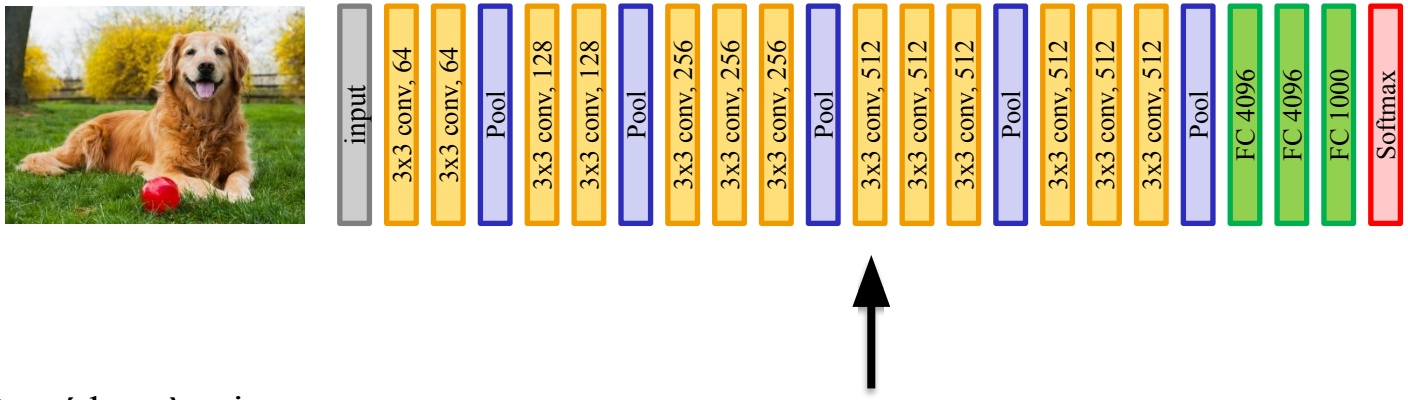


Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014



Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

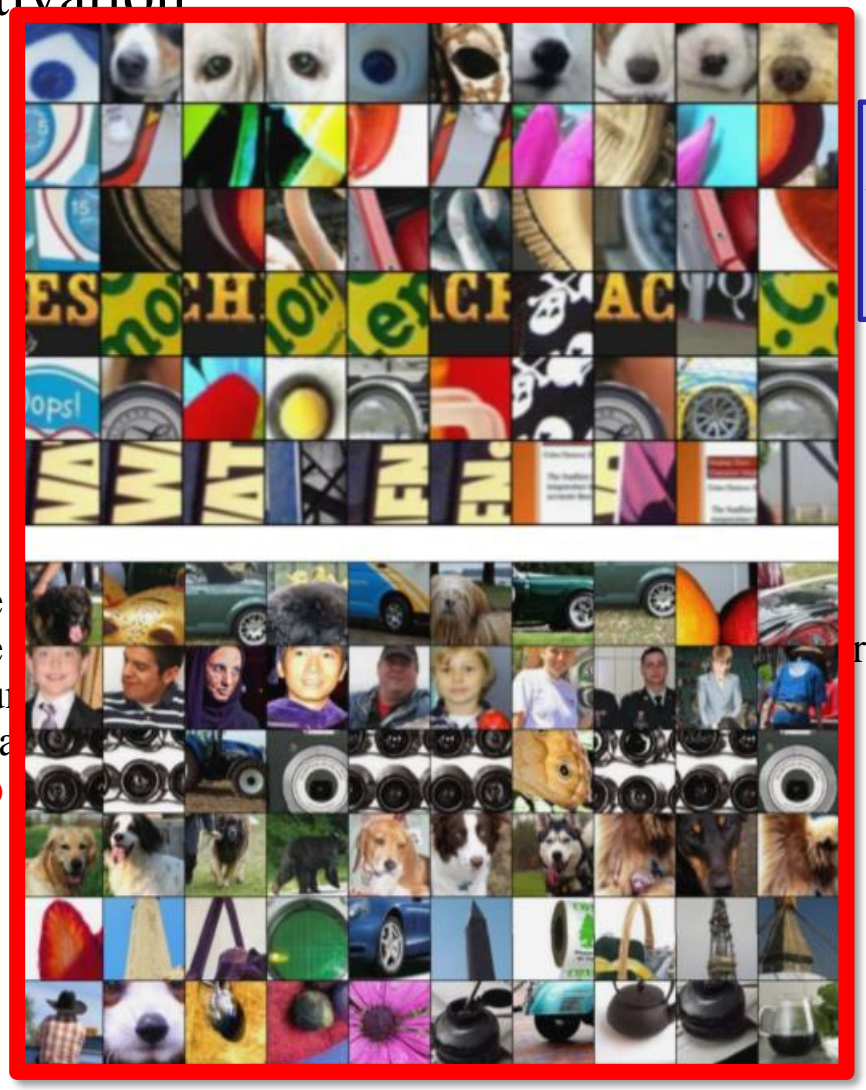
Identifier les zones de l'image qui activent au maximum les neurones d'une carte d'activation



Procédure à suivre :

- Sélectionner une **couche** (ex. conv8)
- Sélectionner une **carte d'activation** de cette couche (ex.: la 20^{e} carte sur 512)
- **Prop. avant** de plusieurs images
- Retenir les N images ayant provoqué une **activation maximale** dans cette carte
- **Crop du champ récepteur** des neurones maximalement activés

Identifier les zones de l'image qui activent au maximum les neurones d'une carte d'activation



Procédure à suivre :

- Sélectionner une
- Sélectionner une rte sur 512)
- **Prop. avant** pou
- Retenir les N ima
- **Crop du champ**

Springenberg et al, "Striving for Simplicity: The All Convolutional Net", ICLR Workshop 2015

Visualisation du « ZF-Net »

[Zeiler,Fergus 2014]

Même expérience mais avec le ZF-Net.

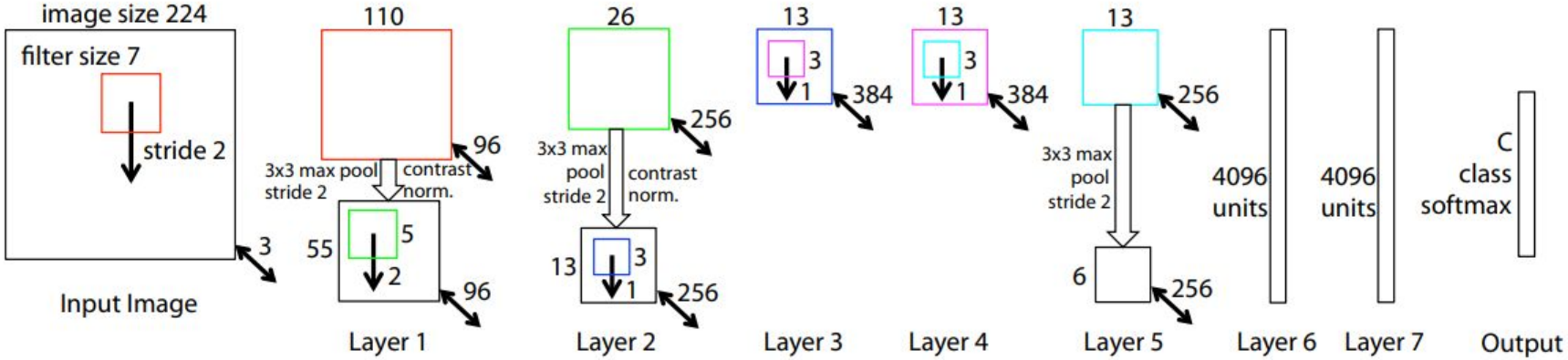


Figure 3. Architecture of our 8 layer convnet model. A 224 by 224 crop of an image (with 3 color planes) is presented as the input. This is convolved with 96 different 1st layer filters (red), each of size 7 by 7, using a stride of 2 in both x and y. The resulting feature maps are then: (i) passed through a rectified linear function (not shown), (ii) pooled (max within 3x3 regions, using stride 2) and (iii) contrast normalized across feature maps to give 96 different 55 by 55 element feature maps. Similar operations are repeated in layers 2,3,4,5. The last two layers are fully connected, taking features from the top convolutional layer as input in vector form ($6 \cdot 6 \cdot 256 = 9216$ dimensions). The final layer is a C-way softmax function, C being the number of classes. All filters and feature maps are square in shape.

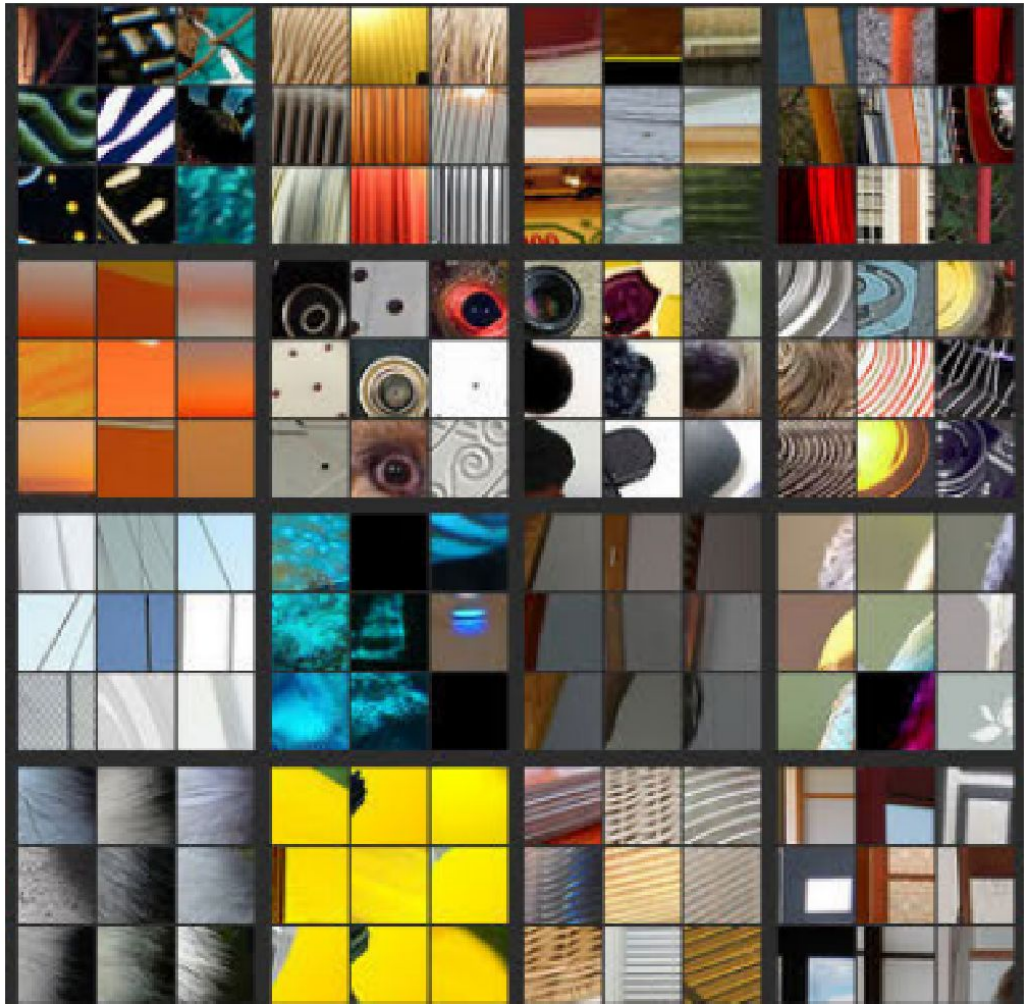
Zeiler and Fergus, “Visualizing and Understanding Convolutional Networks”, ECCV 2014

Visualisation du « ZF-Net »

[Zeiler,Fergus 2014]

Patches qui activent au maximum les couches 1 et 2

Couche 1



Couche 1

Zeiler and Fergus, “Visualizing and Understanding Convolutional Networks”, ECCV 2014

Visualisation du « ZF-Net »

[Zeiler,Fergus 2014]

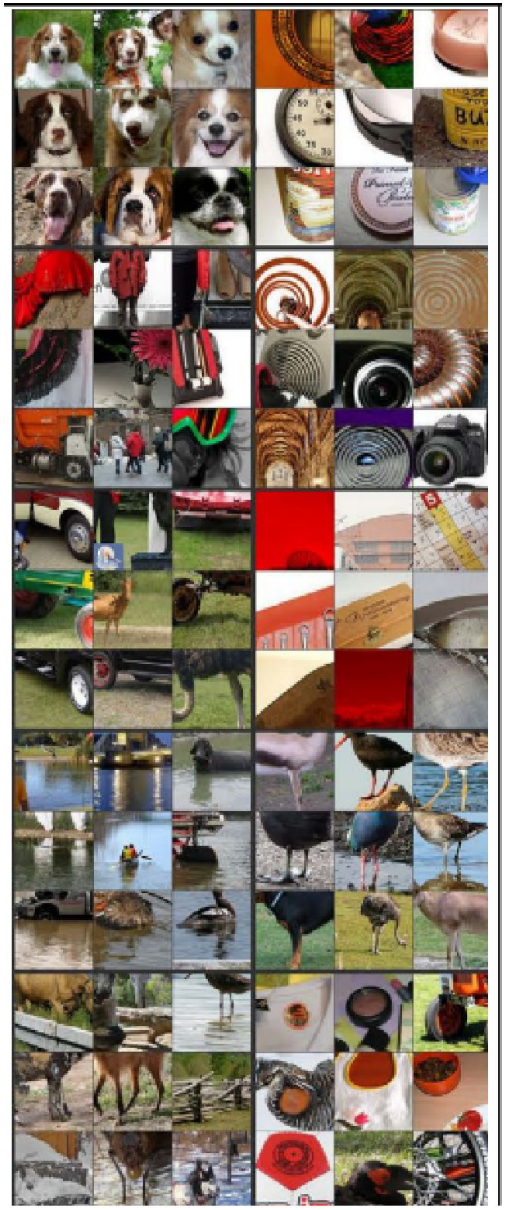
Patches qui activent au maximum la couche 3

Zeiler and Fergus, “Visualizing and Understanding Convolutional Networks”, ECCV 2014

Visualisation du « ZF-Net »

[Zeiler,Fergus 2014]

Patches qui activent au maximum les couches 4 et 5



Zeiler and Fergus, “Visualizing and Understanding Convolutional Networks”, ECCV 2014

Visualisation du « ZF-Net »

[Zeiler,Fergus 2014]

Patches qui activent au maximum les couches 4

Conclusion:

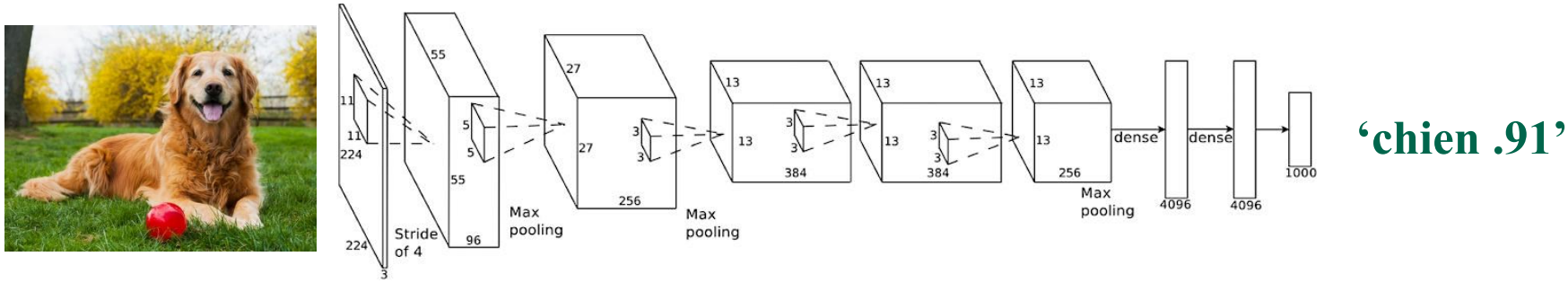
Plus les neurones sont profonds, plus ils détectent des représentations complexes

Zeiler and Fergus, “Visualizing and Understanding Convolutional Networks”, ECCV 2014

Visualisation par rétropropagation

[Simonyan et al. 2014], [Zeiler-Fergu 2014], [Springenberg et al., 2015]

Comment identifier dans cette image
ce qui permet au réseau de prédire la classe « chien »?



‘chien .91’

AlexNet

JT Springenberg, A Dosovitskiy, T Brox, M.Riedmiller, “Striving for simplicity: the all convolutional net”, ICLR 2015

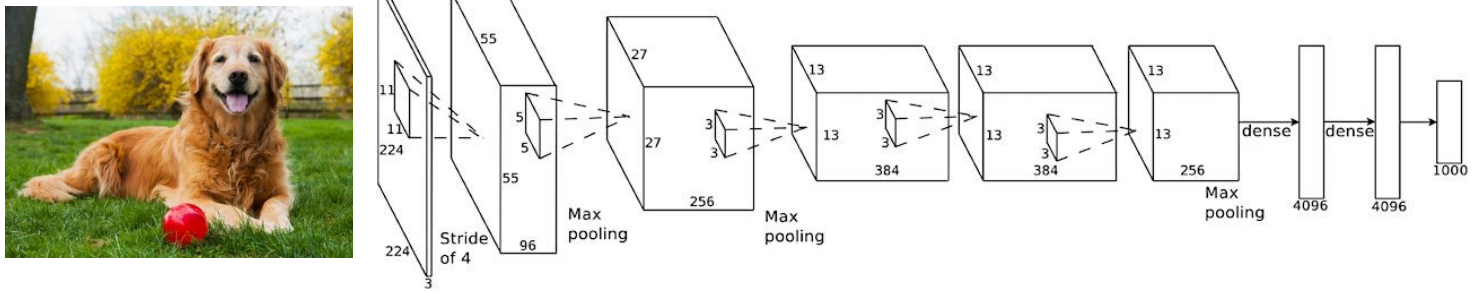
Simonyan, Vedaldi, and Zisserman, “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps”, ICLR Workshop 2014.

Zeiler and Fergus, “Visualizing and Understanding Convolutional Networks”, ECCV 2014

Visualisation par rétropropagation

[Simonyan et al. 2014], [Zeiler-Fergu 2014], [Springenberg et al., 2015]

Étape 1: pré-entrainer le réseau sur une grosse base de données (ex. ImageNet)



JT Springenberg, A Dosovitskiy, T Brox, M.Riedmiller, “Striving for simplicity: the all convolutional net”, ICLR 2015

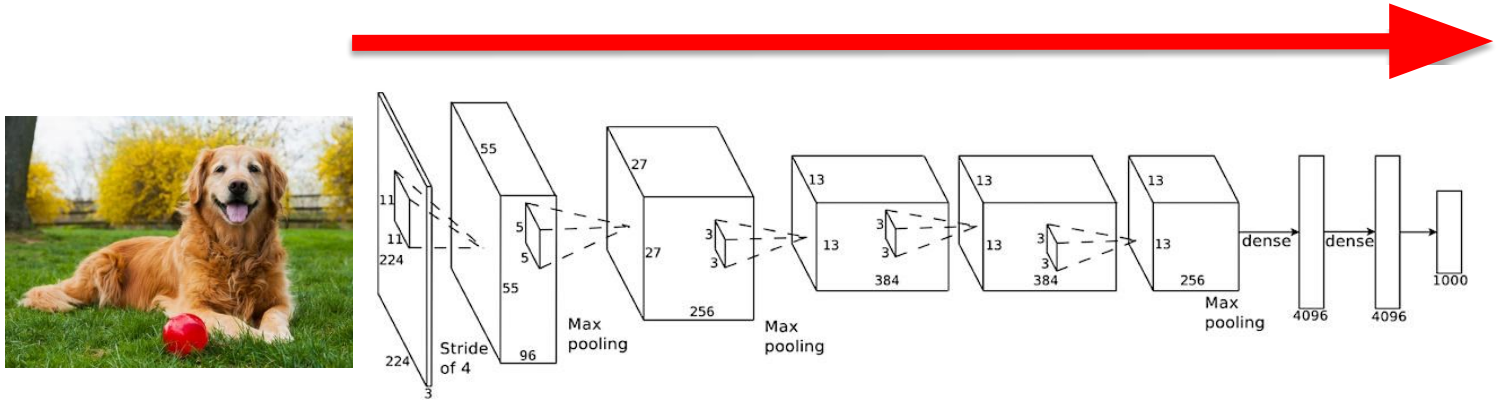
Simonyan, Vedaldi, and Zisserman, “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps”, ICLR Workshop 2014.

Zeiler and Fergus, “Visualizing and Understanding Convolutional Networks”, ECCV 2014

Visualisation par rétropropagation

[Simonyan et al. 2014], [Zeiler-Fergu 2014], [Springenberg et al., 2015]

Étape 2: prop. avant d'une image



JT Springenberg, A Dosovitskiy, T Brox, M.Riedmiller, "Striving for simplicity: the all convolutional net", ICLR 2015

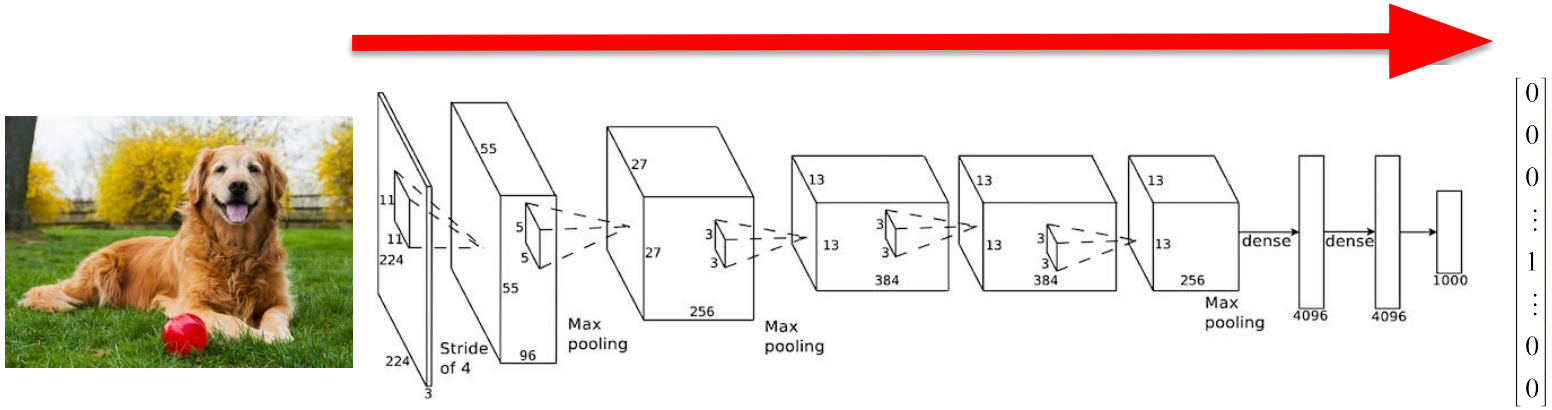
Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.

Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

Visualisation par rétropropagation

[Simonyan et al. 2014], [Zeiler-Fergu 2014], [Springenberg et al., 2015]

Étape 3: forcer le score du réseau à 1 pour la classe d'intérêt



JT Springenberg, A Dosovitskiy, T Brox, M.Riedmiller, "Striving for simplicity: the all convolutional net", ICLR 2015

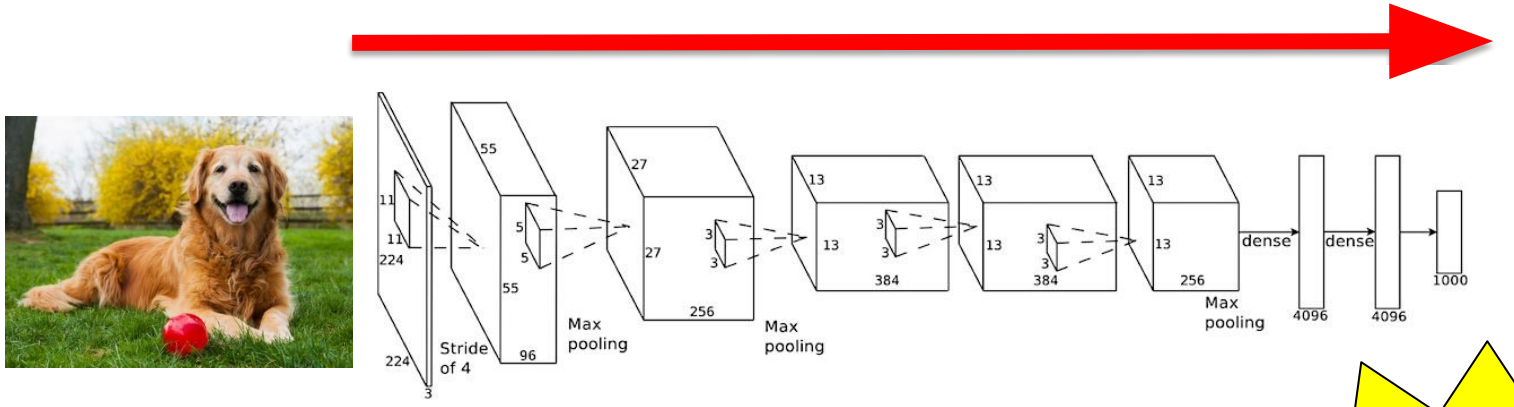
Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.

Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

Visualisation par rétropropagation

[Simonyan et al. 2014], [Zeiler-Fergu 2014], [Springenberg et al., 2015]

Étape 3: forcer le score du réseau à 1 pour la classe d'intérêt



C'est notre loss !

JT Springenberg, A Dosovitskiy, T Brox, M.Riedmiller, "Striving for simplicity: the all convolutional net", I

Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Mode

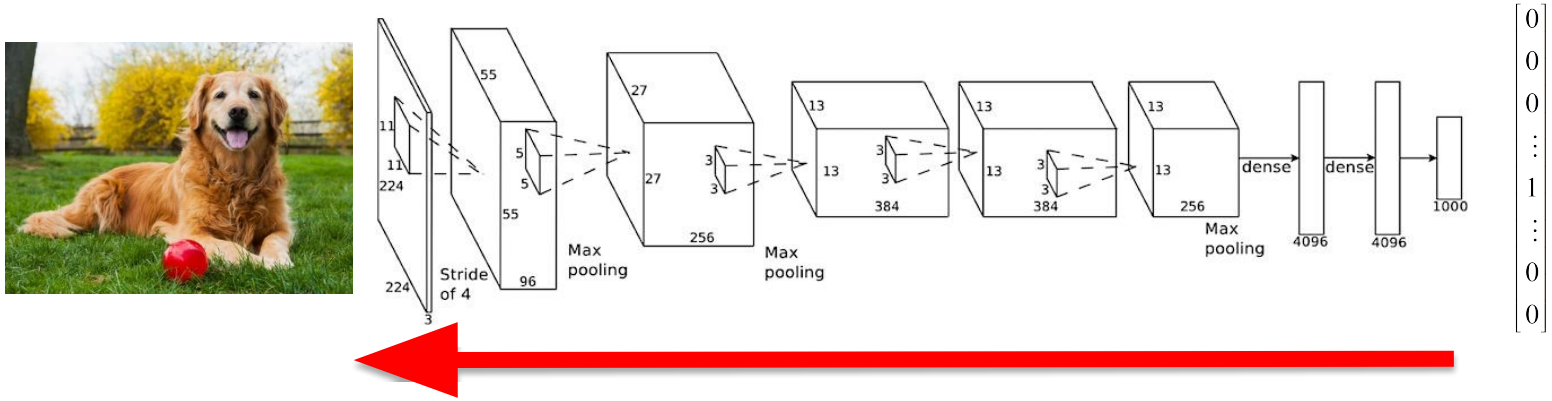
and Saliency Maps", ICLR Workshop 2014.

Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

Visualisation par rétropropagation

[Simonyan et al. 2014], [Zeiler-Fergu 2014], [Springenberg et al., 2015]

Étape 4: rétro-propagation du gradient jusqu'à l'entrée
(chaque pixel = gradient)



JT Springenberg, A Dosovitskiy, T Brox, M.Riedmiller, "Striving for simplicity: the all convolutional net", ICLR 2015

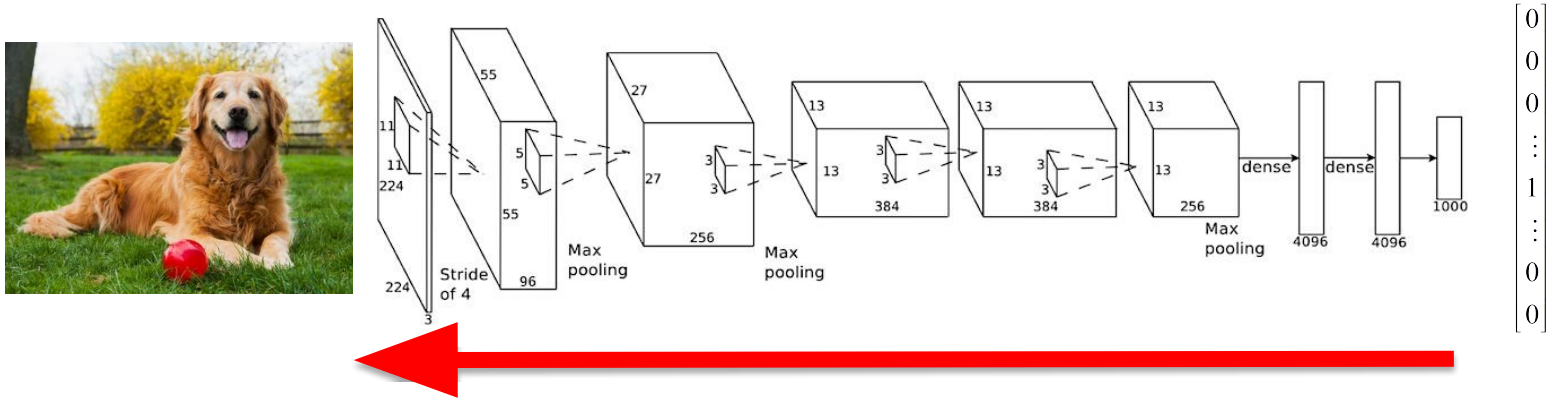
Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.

Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

Visualisation par rétropropagation

[Simonyan et al. 2014], [Zeiler-Fergu 2014], [Springenberg et al., 2015]

Étape 5: convertir les gradients en une image



JT Springenberg, A Dosovitskiy, T Brox, M.Riedmiller, “Striving for simplicity: the all convolutional net”, ICLR 2015

Simonyan, Vedaldi, and Zisserman, “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps”, ICLR Workshop 2014.

Zeiler and Fergus, “Visualizing and Understanding Convolutional Networks”, ECCV 2014

Approche par rétropropagation

```
def generate_grad_saliancy (self, input_image, target_class):

    model_output = self.model(input_image)

    # Init gradients à zero
    self.model.zero_grad()

    # one_hot = 00000100000, 1 sur la classe cible
    one_hot_output = torch.FloatTensor(1, model_output.size()[-1]).zero_()
    one_hot_output[0][target_class] = 1

     # Backward pass
     model_output.backward(gradient=one_hot_output)

     # [0] pour éliminer la première dimension (1,3,224,224)
     gradients_saliancy = self.gradients.data.numpy()[0]

     return gradients_saliancy
```

Rétropropagation à partir du score du réseau

[Simonyan et al. 2014]

Gradient en valeur absolue

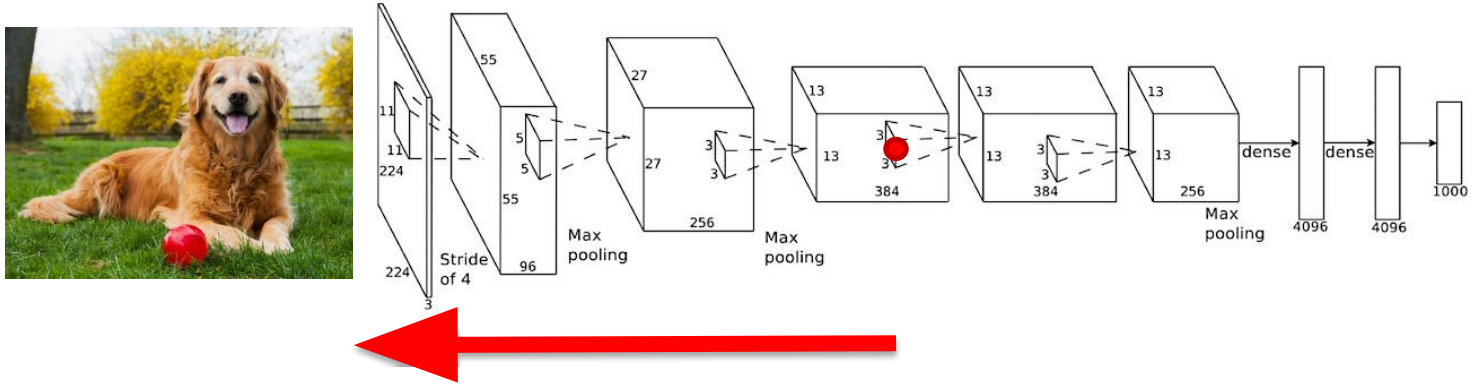
Simonyan, Vedaldi, and Zisserman, “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps”, ICLR Workshop 2014.

Visualisation par rétropropagation

[Simonyan et al. 2014], [Zeiler-Fergu 2014], [Springenberg et al., 2015]

On peut faire la même chose pour **1 neurone** :

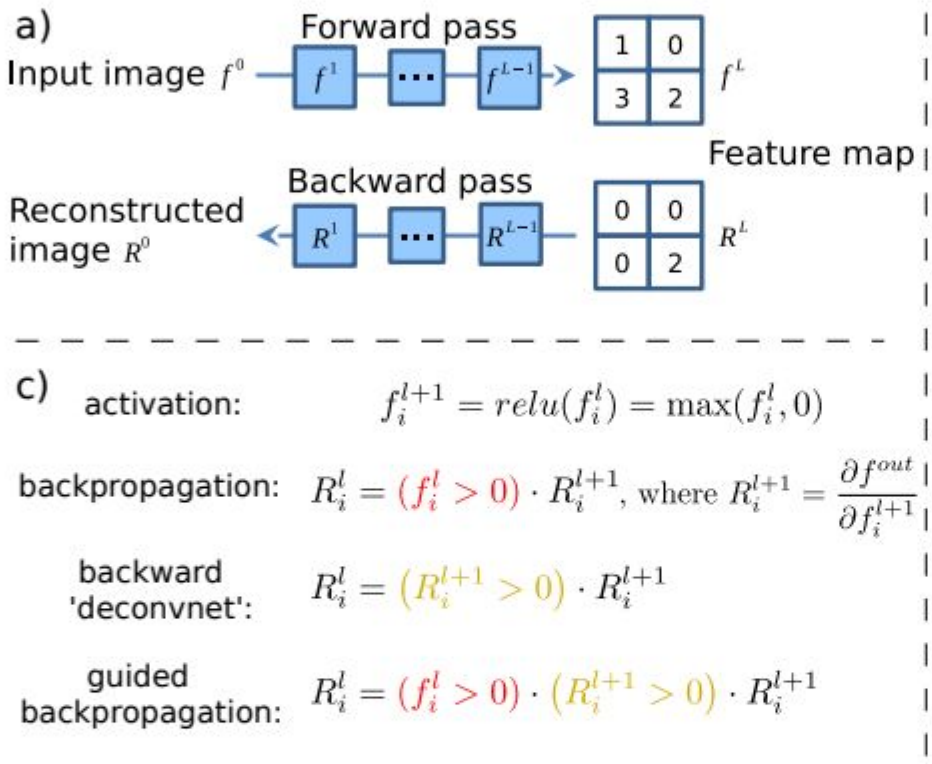
- Propagation avant jusqu'à la couche X
- Forcer à 0 la sortie de tous les neurones de la couche X
- Mettre à 1 la sortie du neurone d'intérêt
- Propager le gradient vers l'image d'entrée.

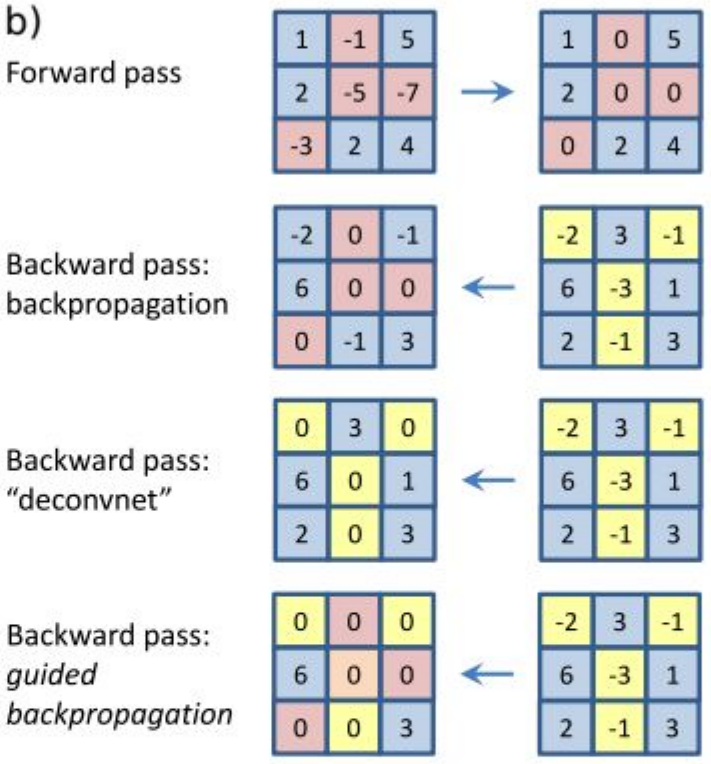


Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.

NOTE: 3 façons de rétropropager le gradient

[Simonyan et al. 2014], [Zeiler-Fergus 2014], [Springenberg et al., 2015]

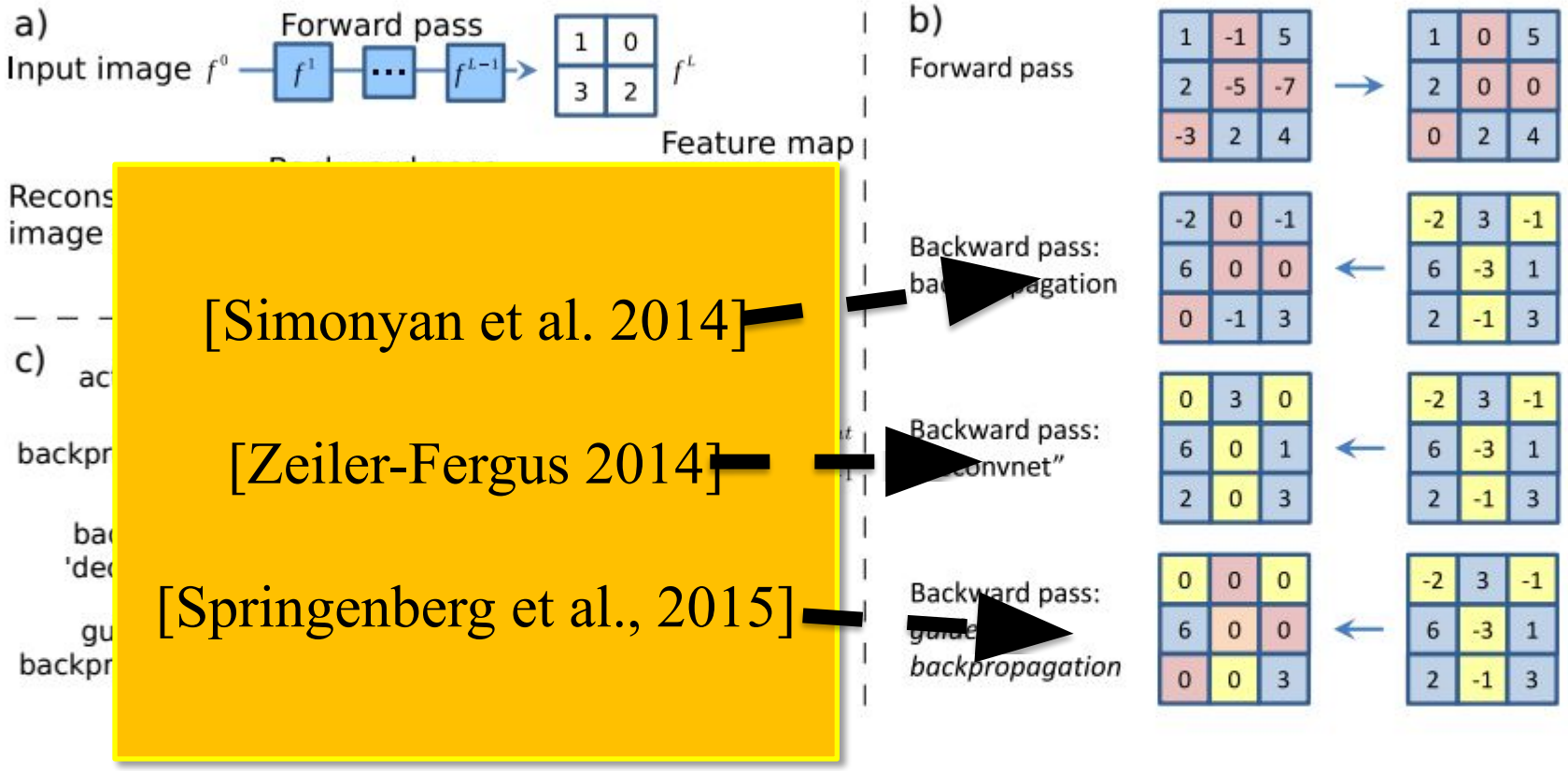




JT Springenberg, A Dosovitskiy, T Brox, M.Riedmiller, "Striving for simplicity: the all convolutional net", ICLR 2015

NOTE: 3 façons de rétropopager le gradient

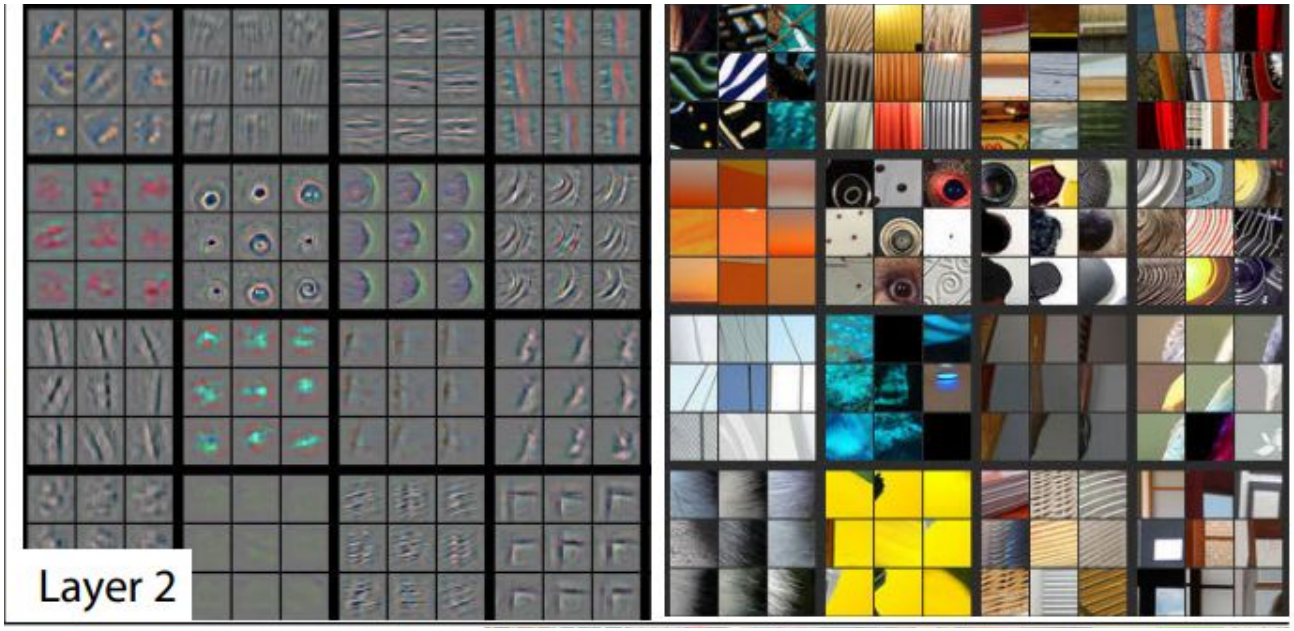
[Simonyan et al. 2014], [Zeiler-Fergus 2014], [Springenberg et al., 2015]

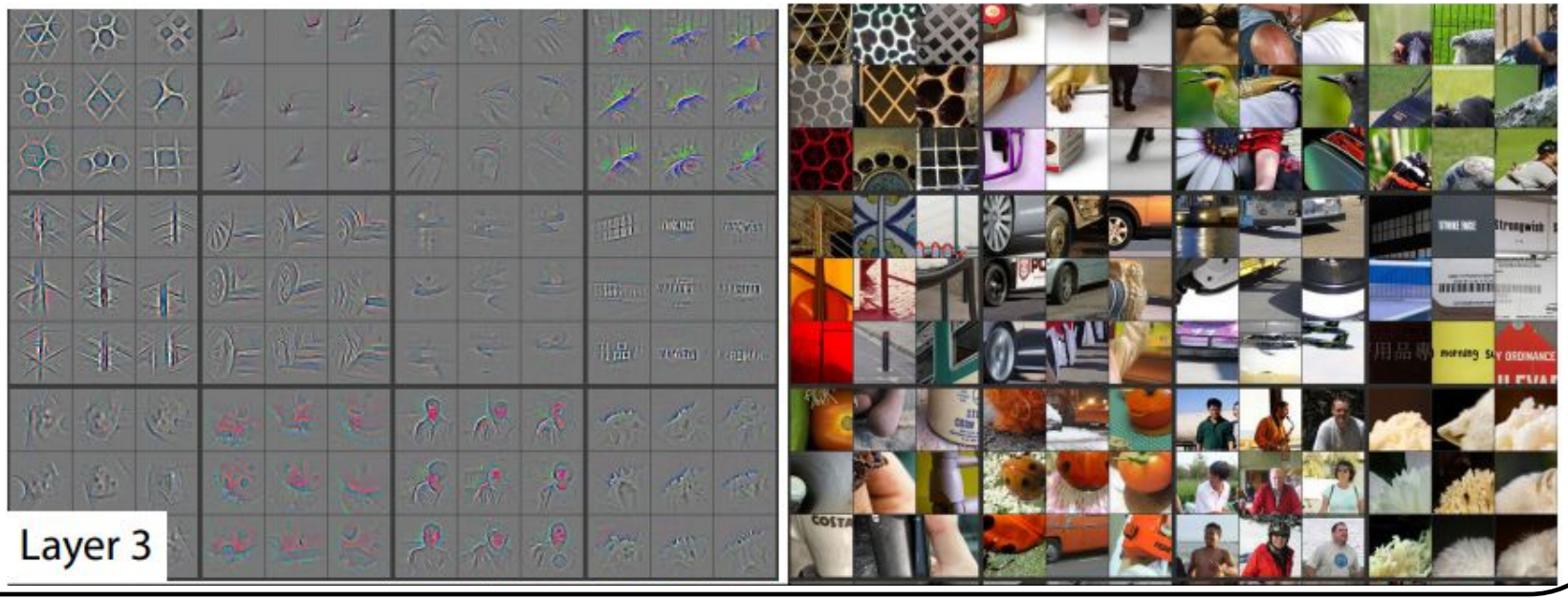


JT Springenberg, A Dosovitskiy, T Brox, M.Riedmiller, "Striving for simplicity: the all convolutional net", ICLR 2015

« Deconv net »

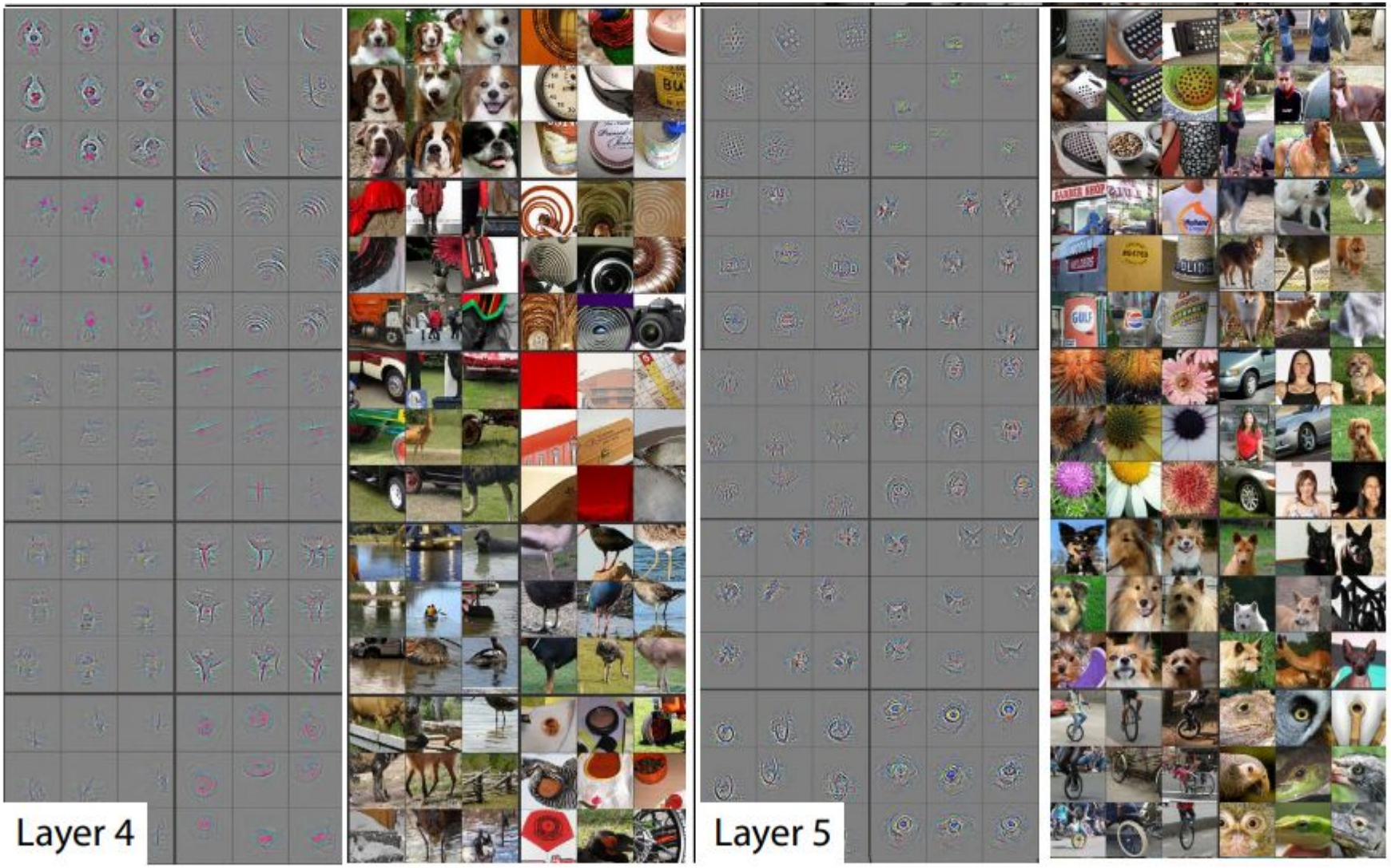
[Zeiler-Fergus 2014]





« Deconv net »

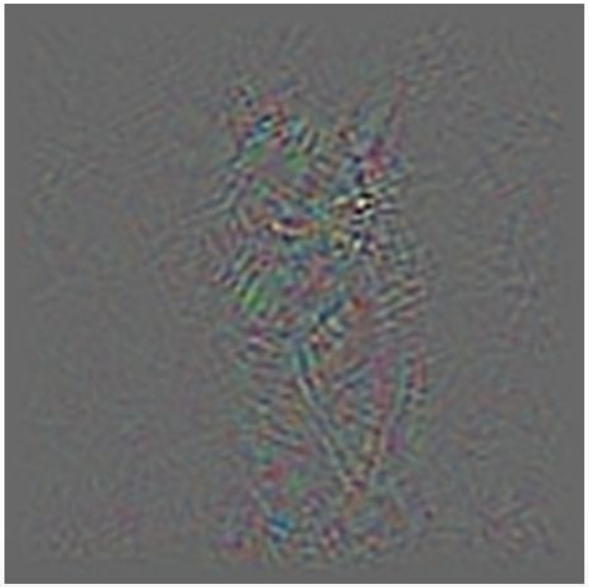
[Zeiler-Fergus 2014]



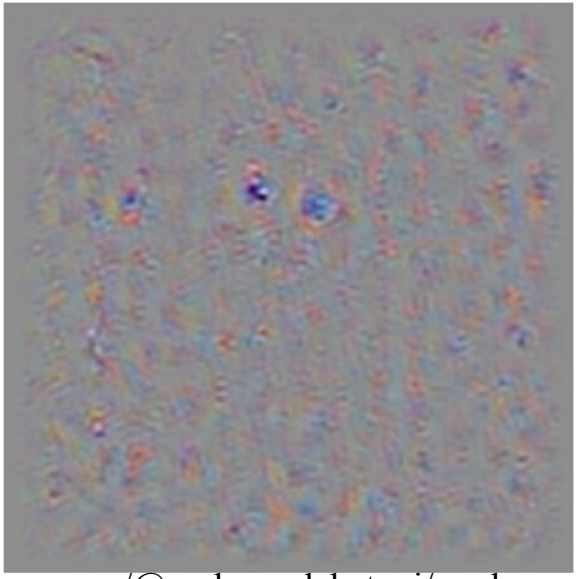
L’approche « *guided backprop* » est plus souvent utilisée car les résultats sont plus saillants et moins bruités

Input image

Backpropagation



Deconvolution



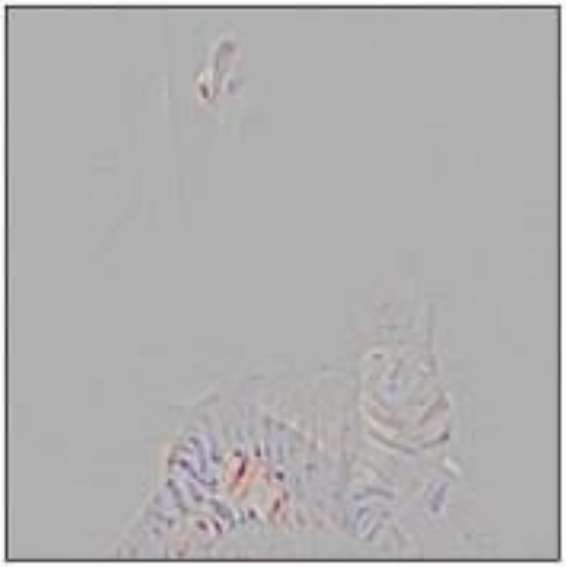
Guided Backprop

https://medium.com/@mohamedchetoui/grad-cam-gradient-weighted-class-activation-mapping-ffd72742243a

Guided backprop

[Springenberg et al., 2015]

Guided Grad-CAM for "Cat"

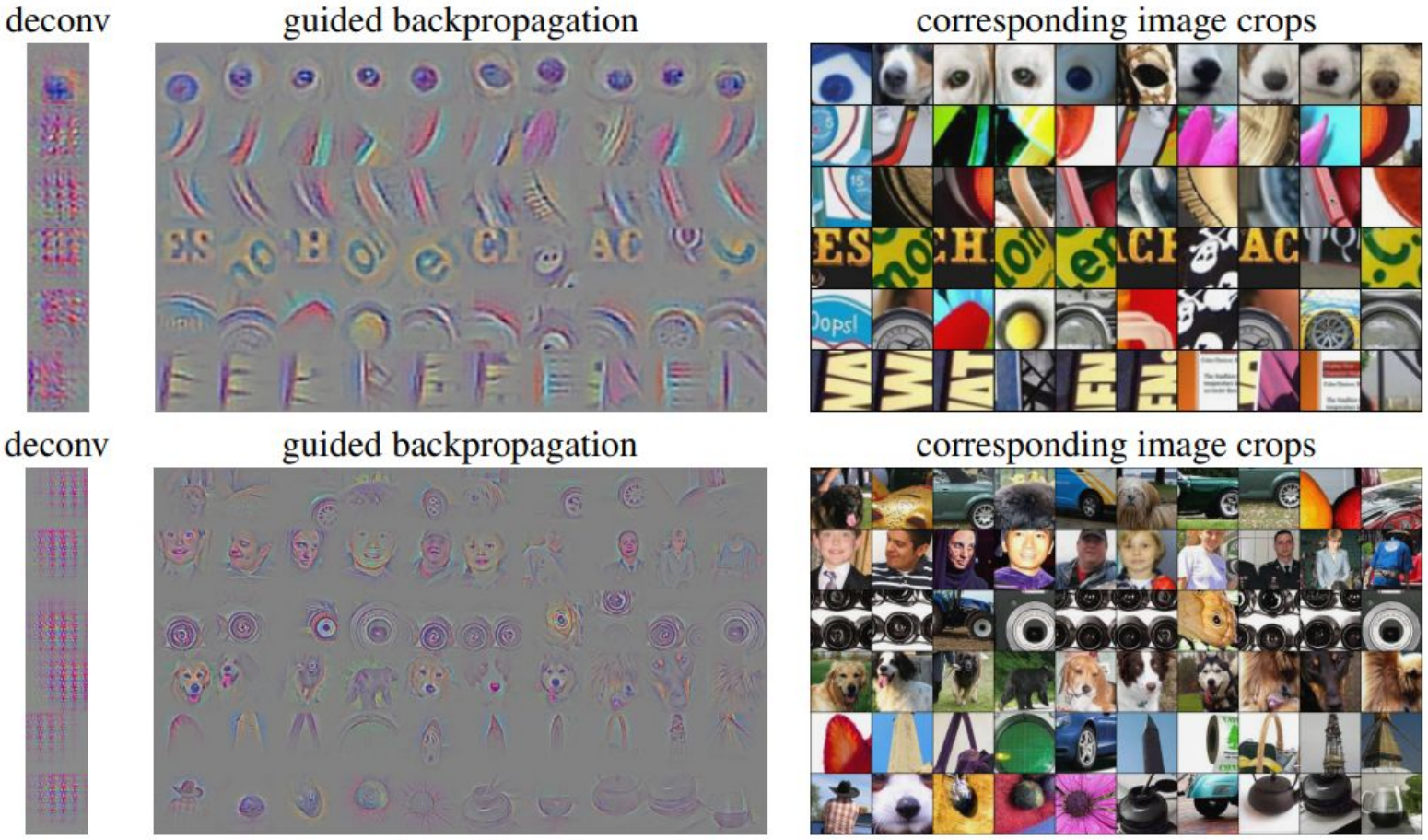


Guided Grad-CAM for "Dog"

https://medium.com/@mohamedchetoui/grad-cam-gradient-weighted-class-activation-mapping-ffd72742243a

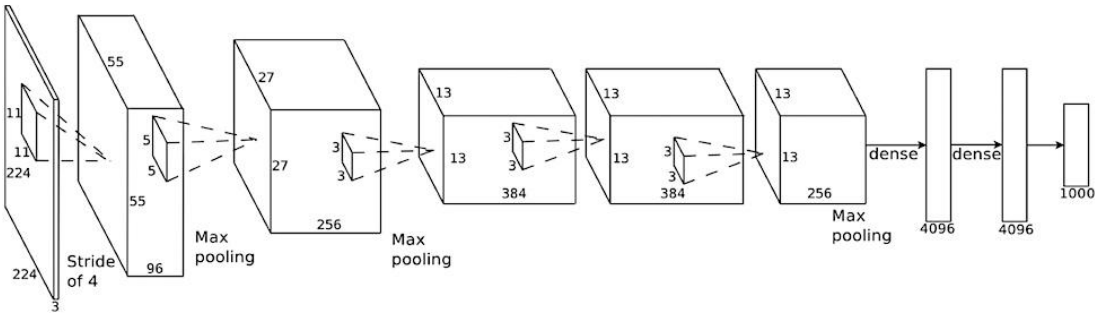
Guided backprop

[Springenberg et al., 2015]



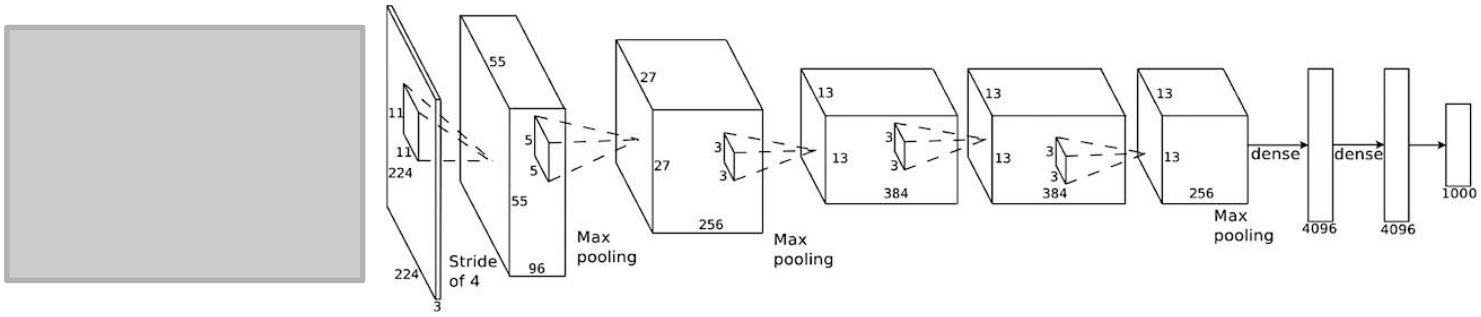
On peut « fabriquer » une image qui activera maximalement un neurone

Étape 1: Préentraîner le réseau avec une grosse BD (ex. ImageNet)



On peut « fabriquer » une image qui activera maximalement un neurone

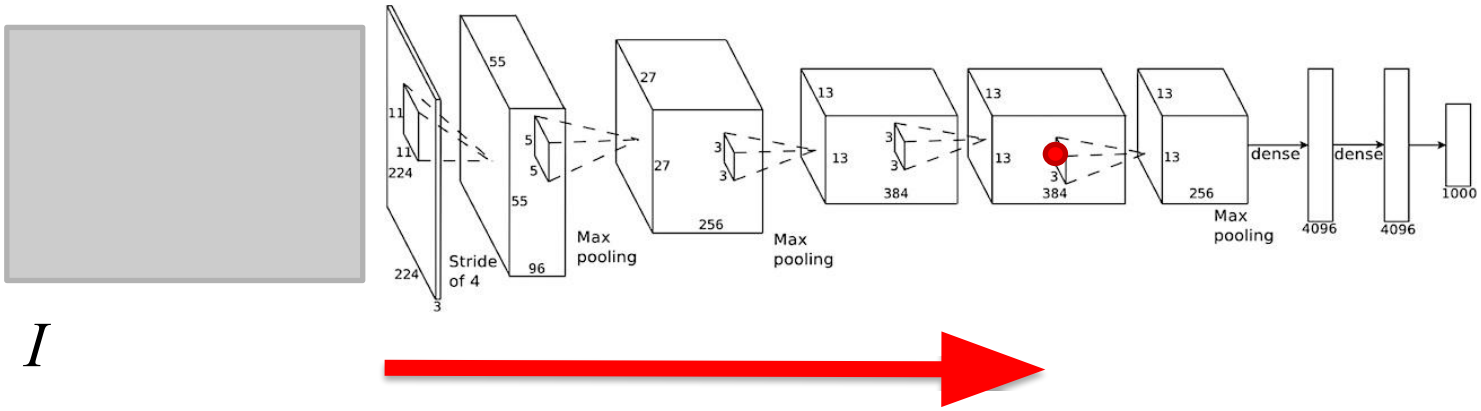
Étape 2: Initialiser une image avec des valeurs 0



I

On peut « fabriquer » une image qui activera maximalement un neurone

Étape 2: propagation avant jusqu'au neurone d'intérêt

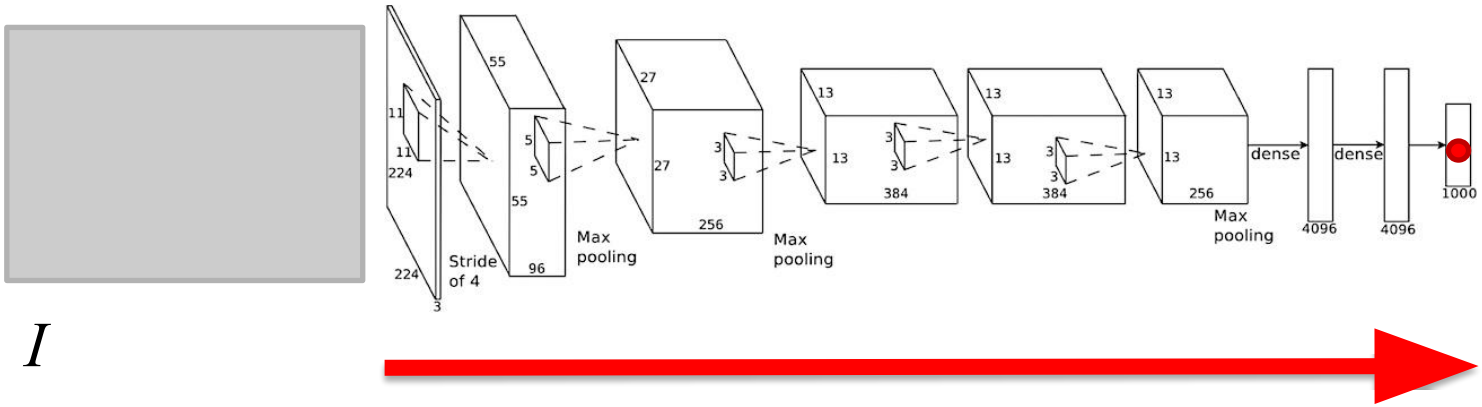


I

Peut être un neurone d'une couche intermédiaire

On peut « fabriquer » une image qui activera maximalement un neurone

Étape 2: propagation avant jusqu'au neurone d'intérêt

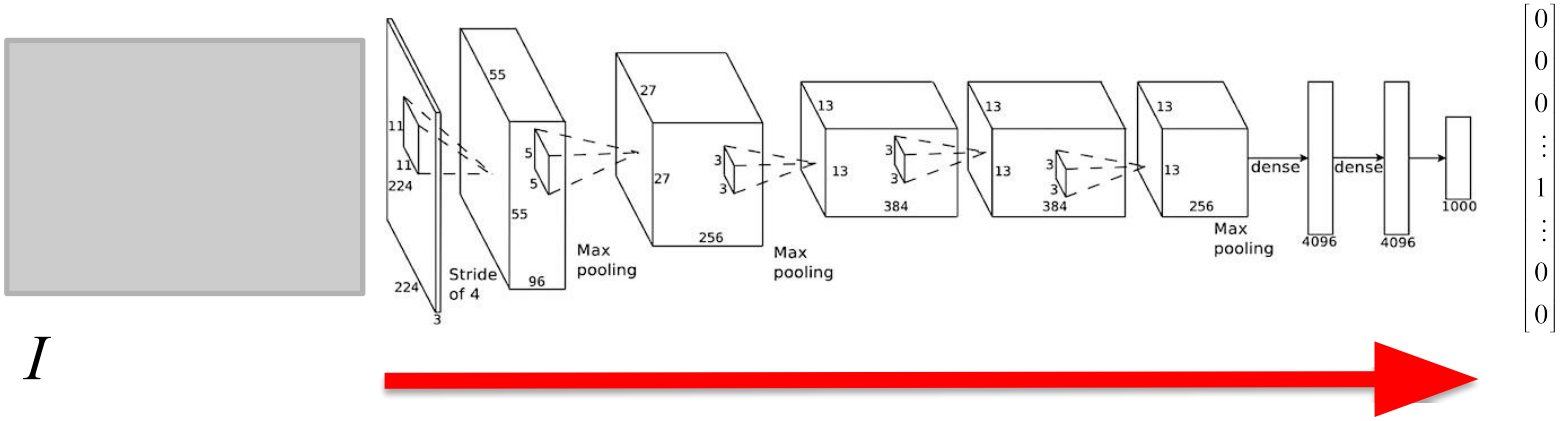


I

Ou encore un neurone de sortie

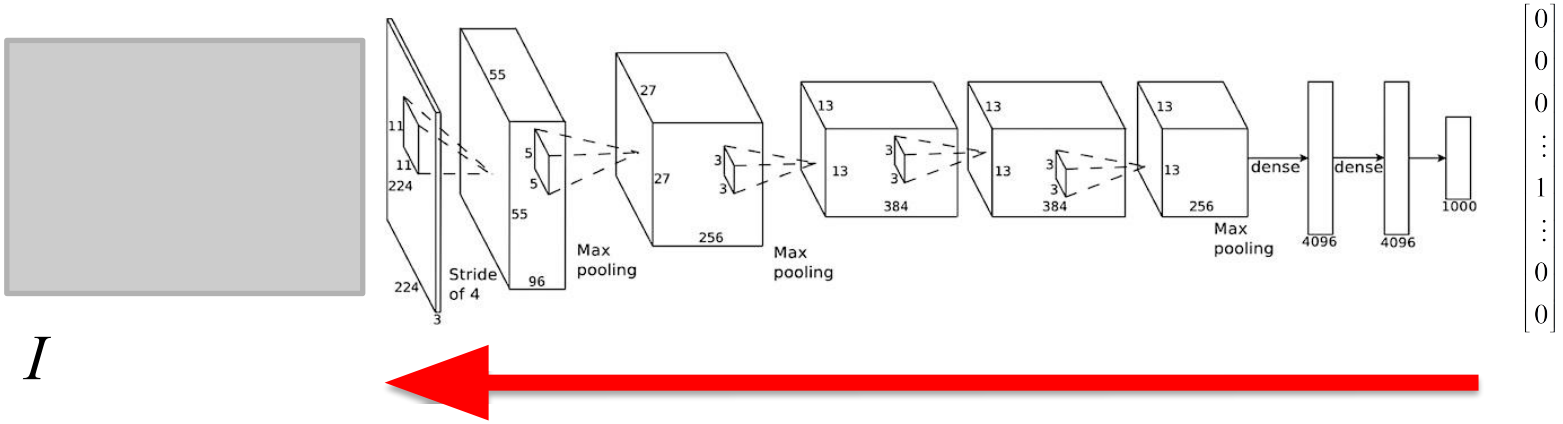
On peut « fabriquer » une image qui activera maximalement un neurone

Étape 3: forcer le neurone d'intérêt à 1 et les autres à 0



On peut « fabriquer » une image qui activera maximalement un neurone

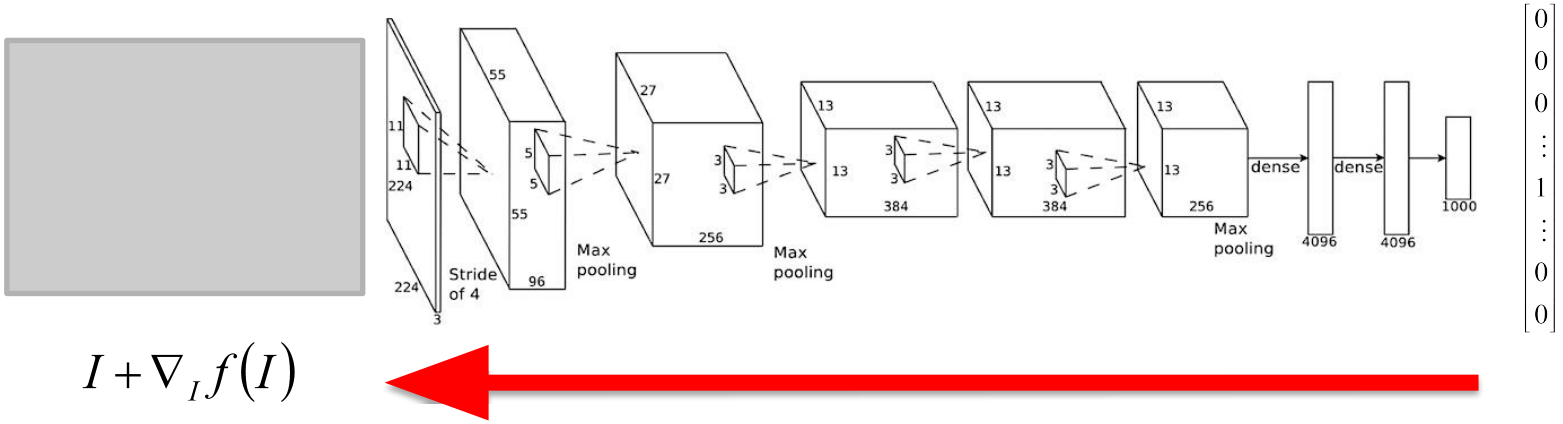
Étape 4: rétro propager le gradient jusqu'à l'image *I*



I

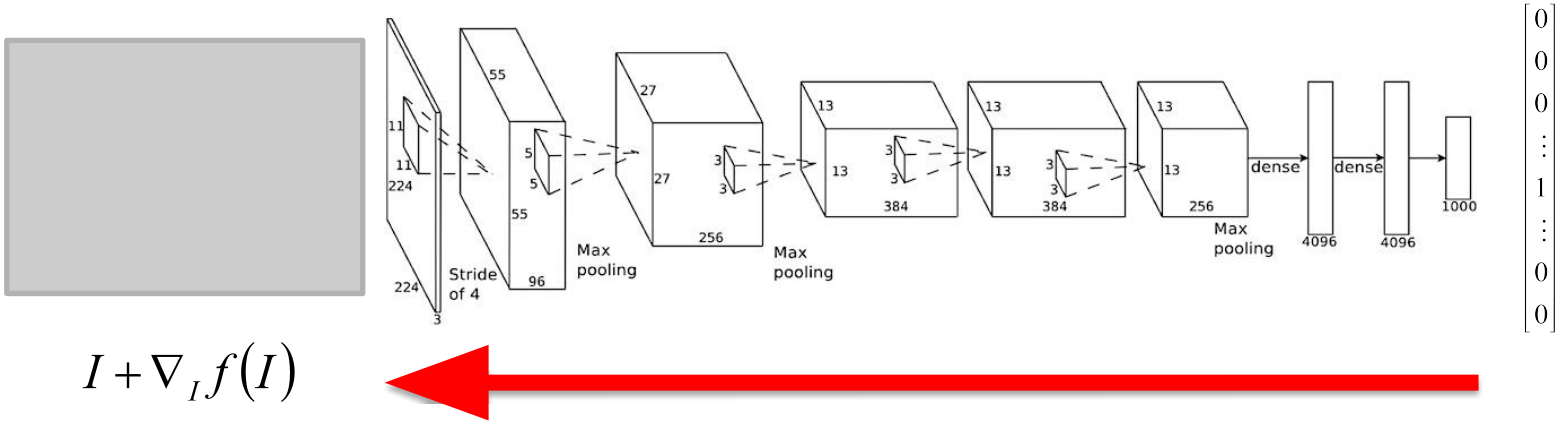
On peut « fabriquer » une image qui activera maximalement un neurone

Étape 5: ajouter le gradient à l'image I : **ascension du gradient**



On peut « fabriquer » une image qui activera maximalement un neurone

Étape 5: ajouter le gradient à l'image I : **ascension du gradient**



$$I^* = \arg\max f(I)$$

On peut « fabriquer » une image qui activera maximalement un neurone

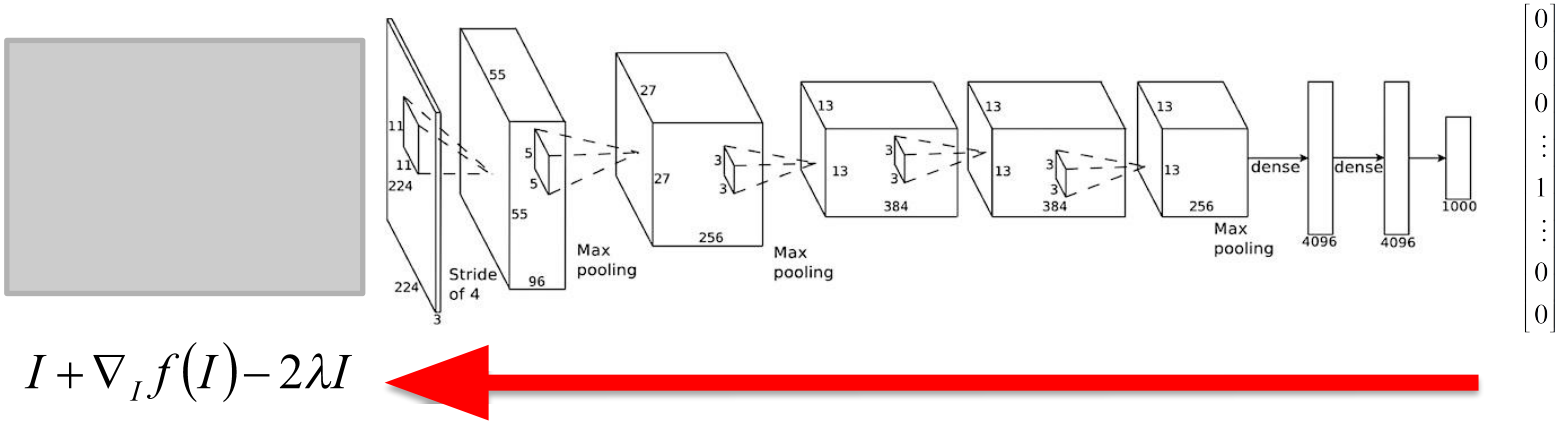
Étape 5: ajouter le gradient à l'image *I* : **ascension du gradient**

Afin de s'assurer que l'image produite soit lisse,
on rajoute un terme de régularisation, souvent de type L2

$$I^* = \arg\max f(I) - \lambda \|I\|^2$$

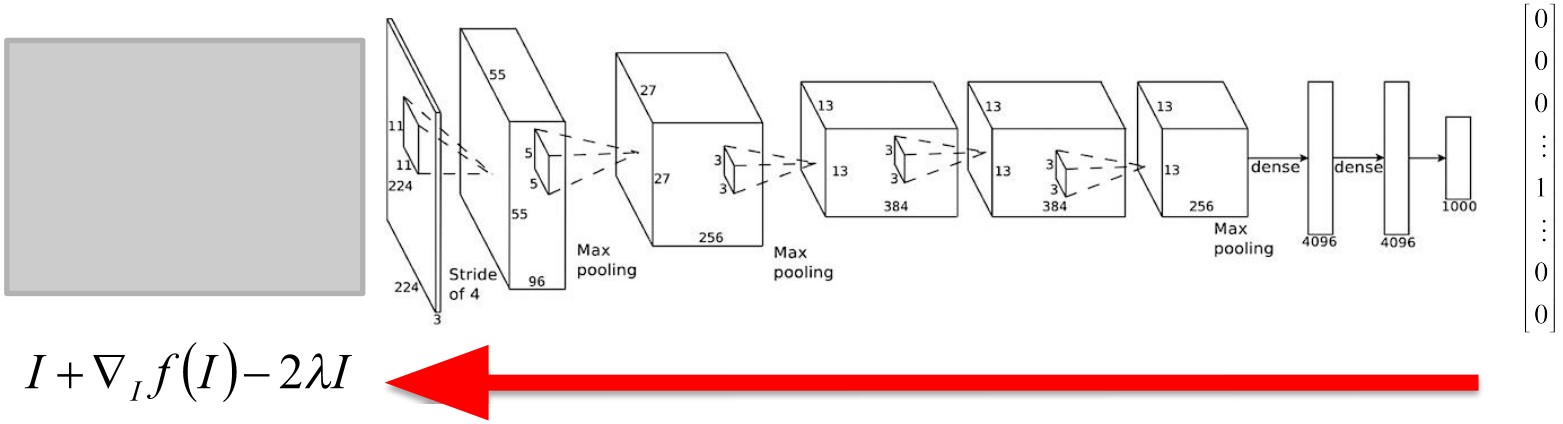
On peut « fabriquer » une image qui activera maximalement un neurone

Étape 5: ajouter le gradient à l'image I : **ascension du gradient**



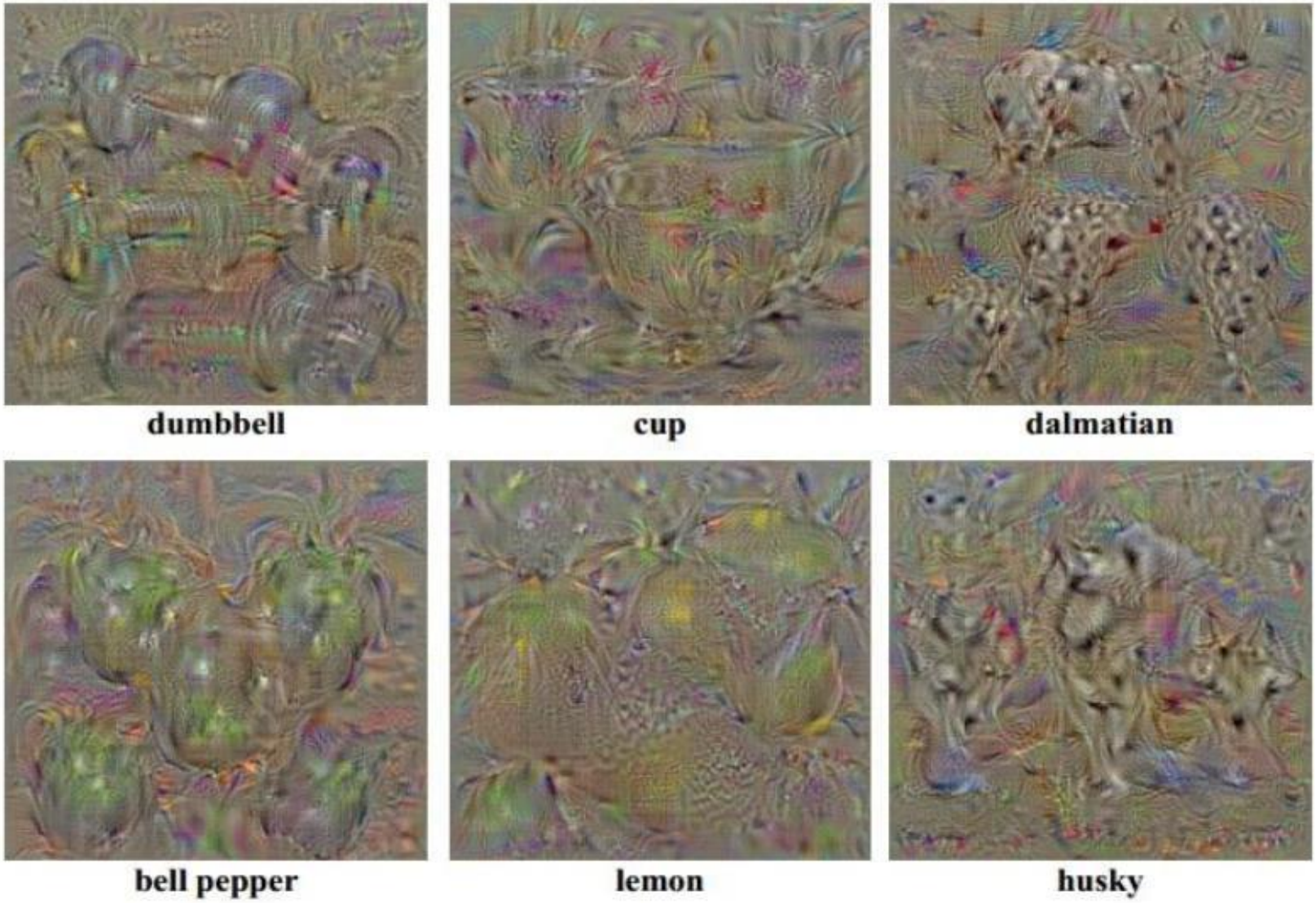
$$I^* = \arg\max f(I) - \lambda \|I\|^2$$

On peut « fabriquer » une image qui activera maximalement un neurone



NOTE IMPORTANTE: pour cette opération, les poids du réseau **sont gelés**

seule l'image d'entrée est modifiée

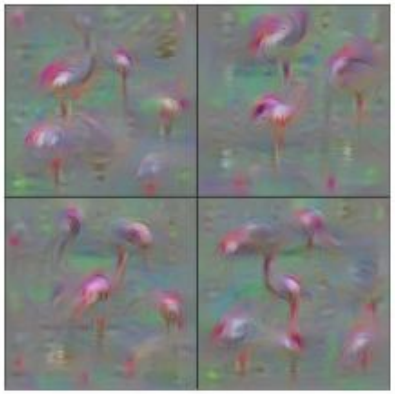


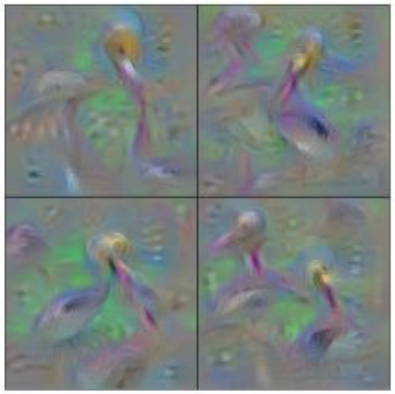
Simonyan, Vedaldi, and Zisserman, “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps”, ICLR Workshop 2014.

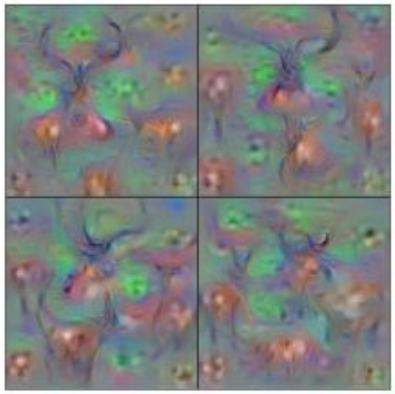
On peut améliorer les résultats. À chaque itération:

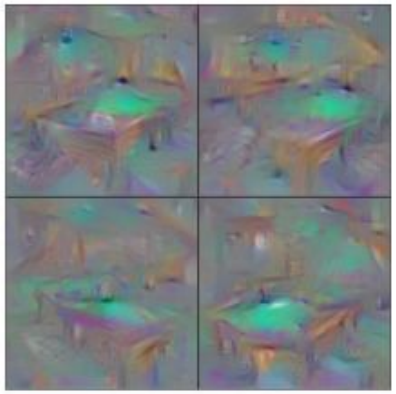
(1) Ajouter un **flou gaussien** à l'image à chaque itération

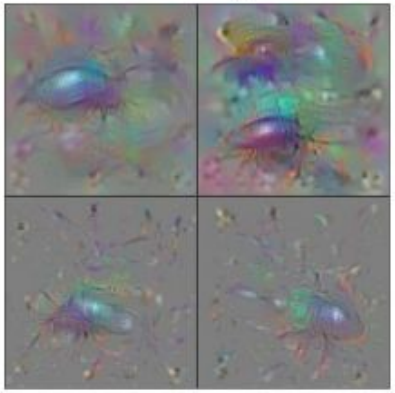
(2) **Forcer à 0** les pixels ayant une petite valeur

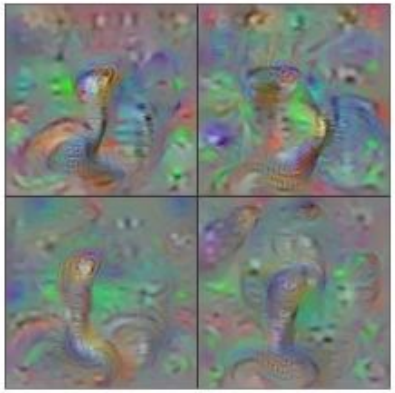

Flamingo

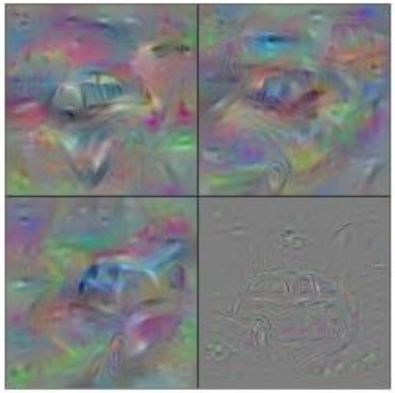

Pelican

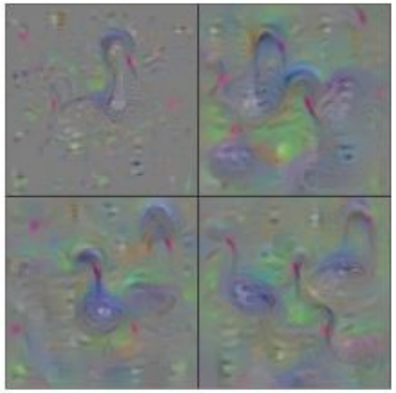

Hartebeest

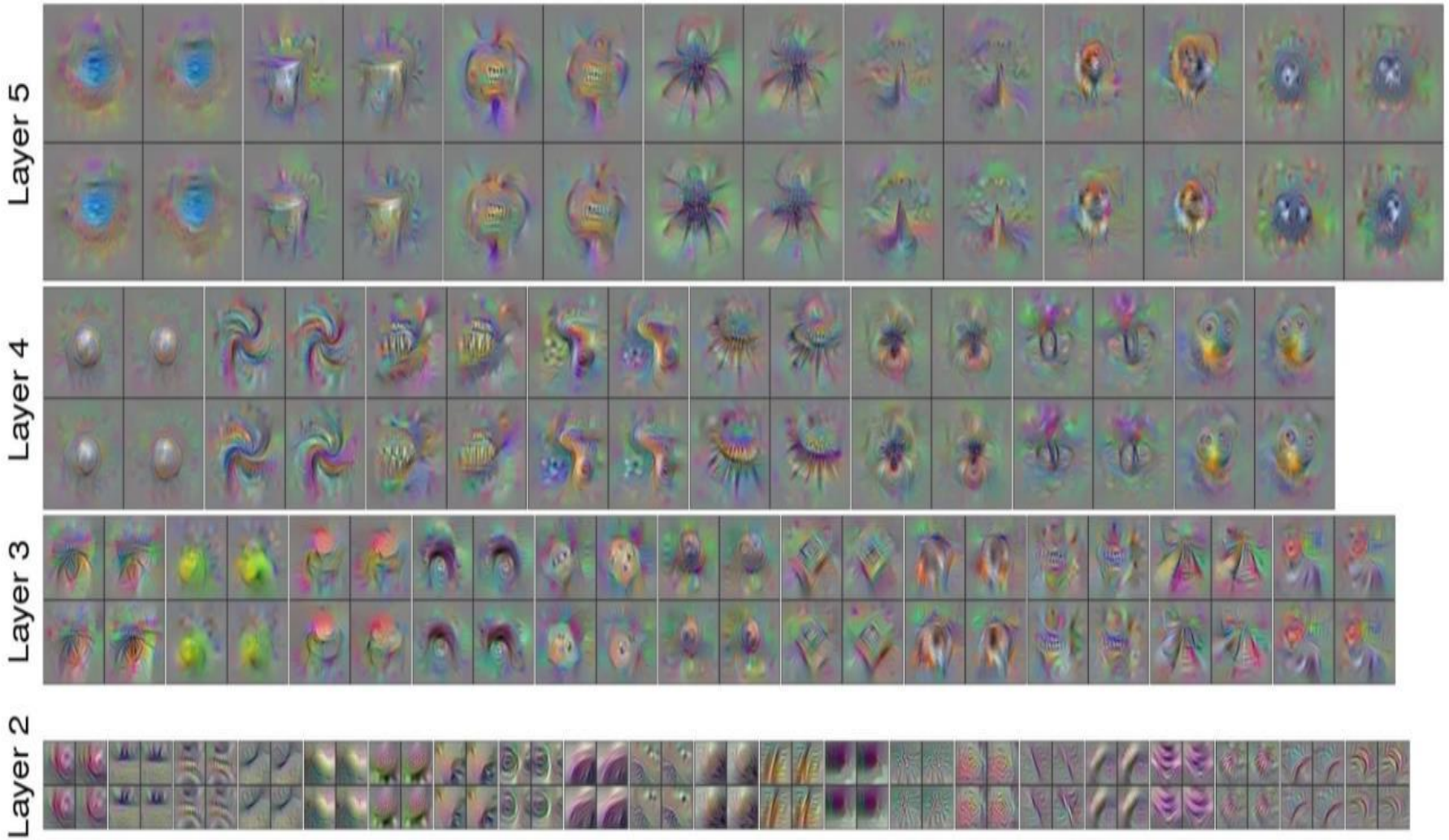

Billiard Table


Ground Beetle


Indian Cobra


Station Wagon

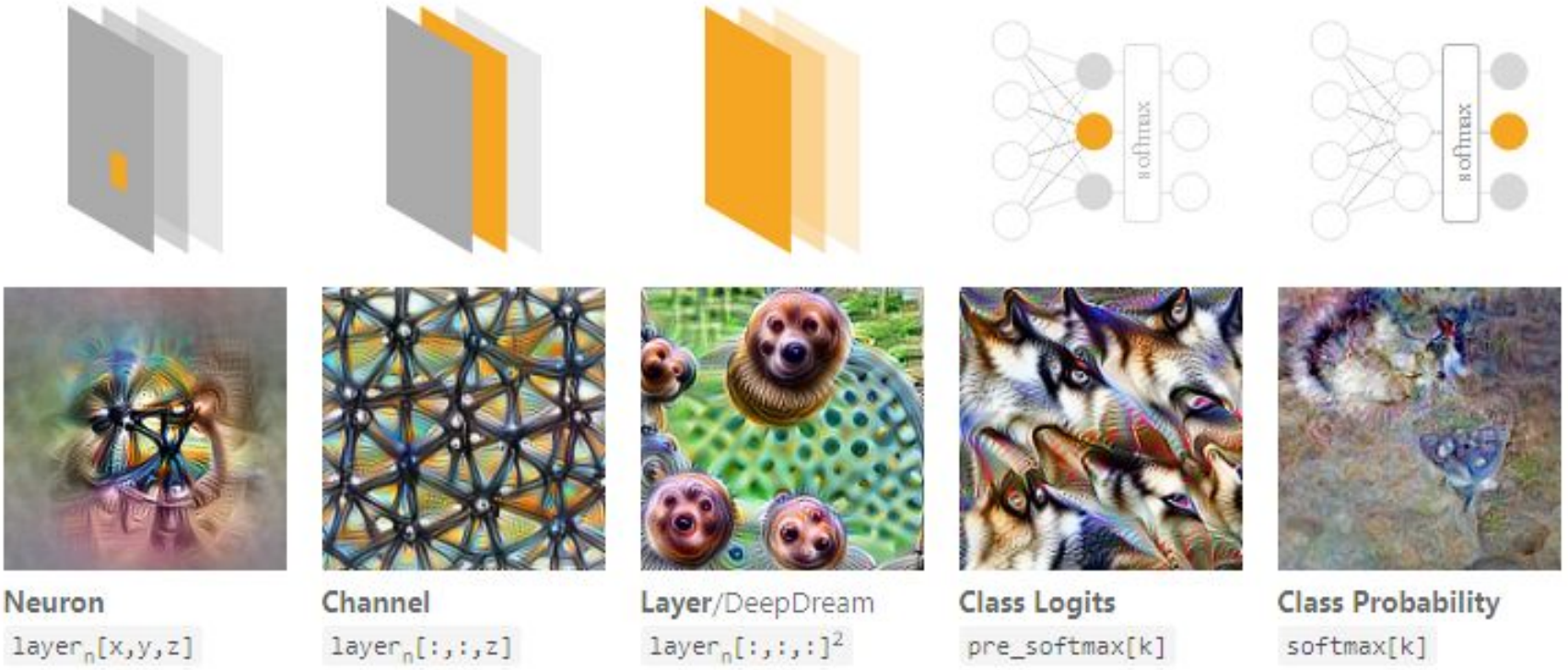

Black Swan



Yosinski et al, “Understanding Neural Networks Through Deep Visualization”, ICML DL Workshop 2014.

Et la recherche continue!

La visualisation de réseaux de neurones est un sujet de recherche actif!



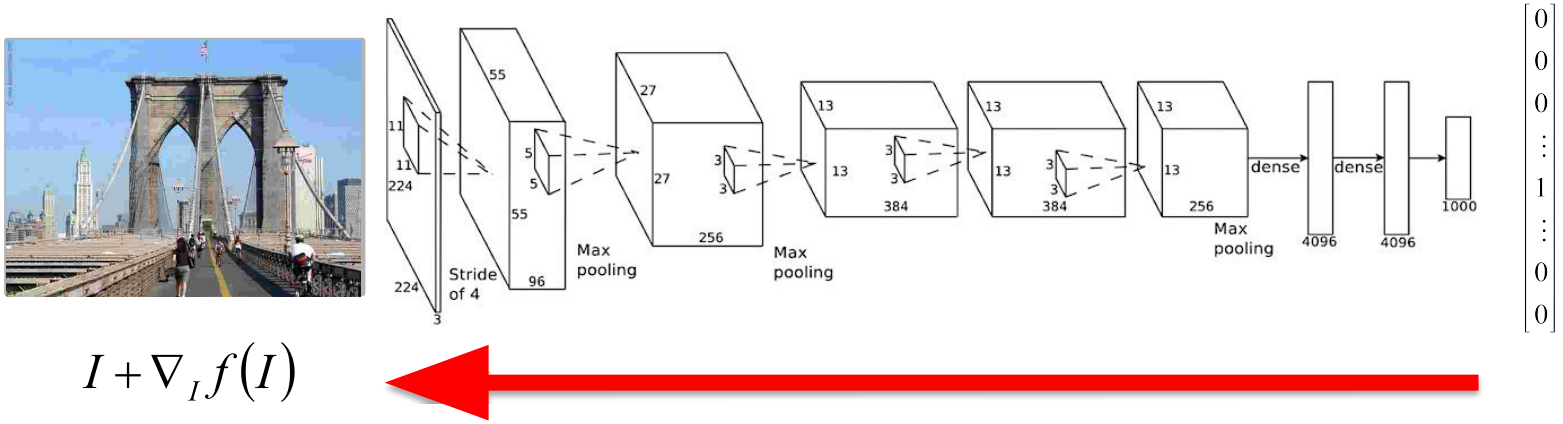
https://distill.pub/2017/feature-visualization/

Et la recherche continue!

La visualisation de réseaux de neurones est un sujet de recherche actif!

https://distill.pub/2017/feature-visualization/

Deep dream : même opération mais partant d'une image réelle



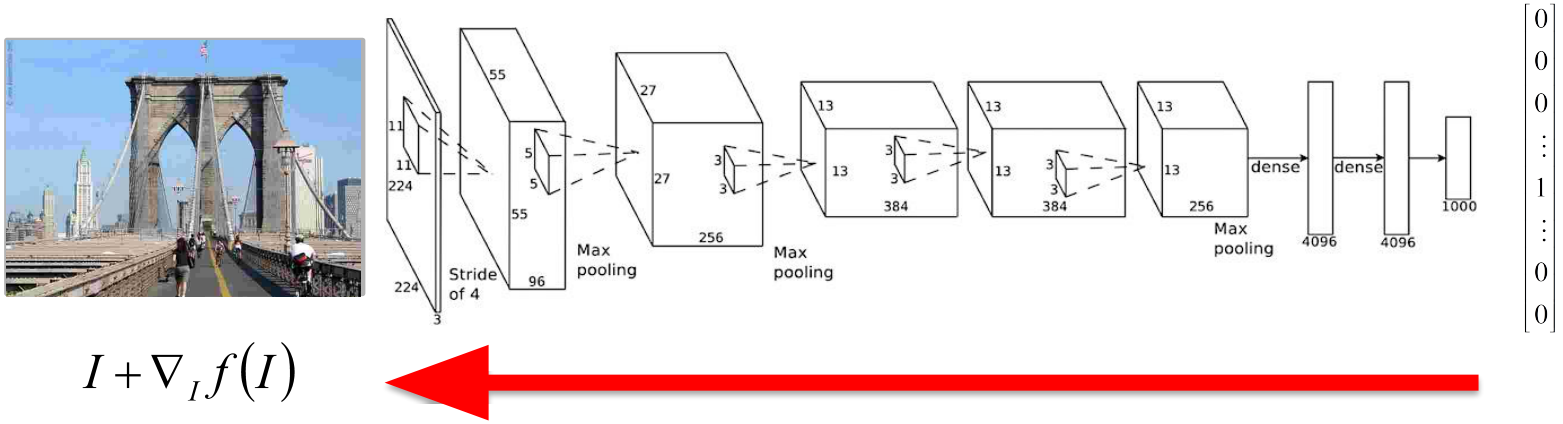
$I + \nabla_I f(I)$

NOTE IMPORTANTE: pour cette opération, les poids du réseau **sont gelés**

seule l'image d'entrée est modifiée

Deep dream : même opération mais partant d'une image réelle

1. Propager une image dans le réseau



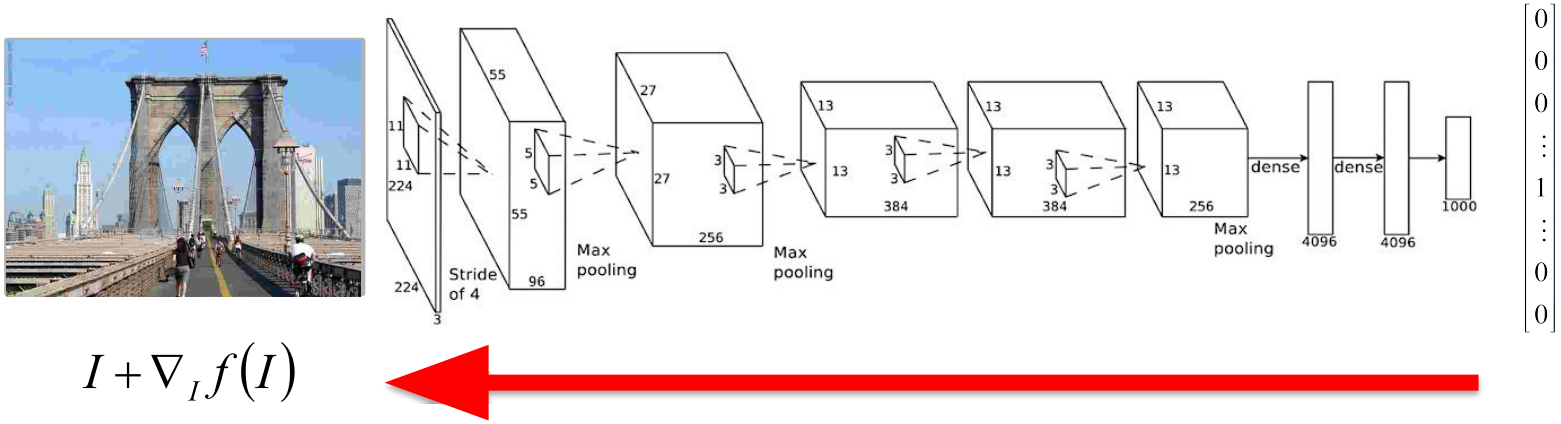
$I + \nabla_I f(I)$

NOTE IMPORTANTE: pour cette opération, les poids du réseau **sont gelés**

seule l'image d'entrée est modifiée

Deep dream : même opération mais partant d'une image réelle

1. Propager une image dans le réseau
2. Sélectionner une couche du réseau

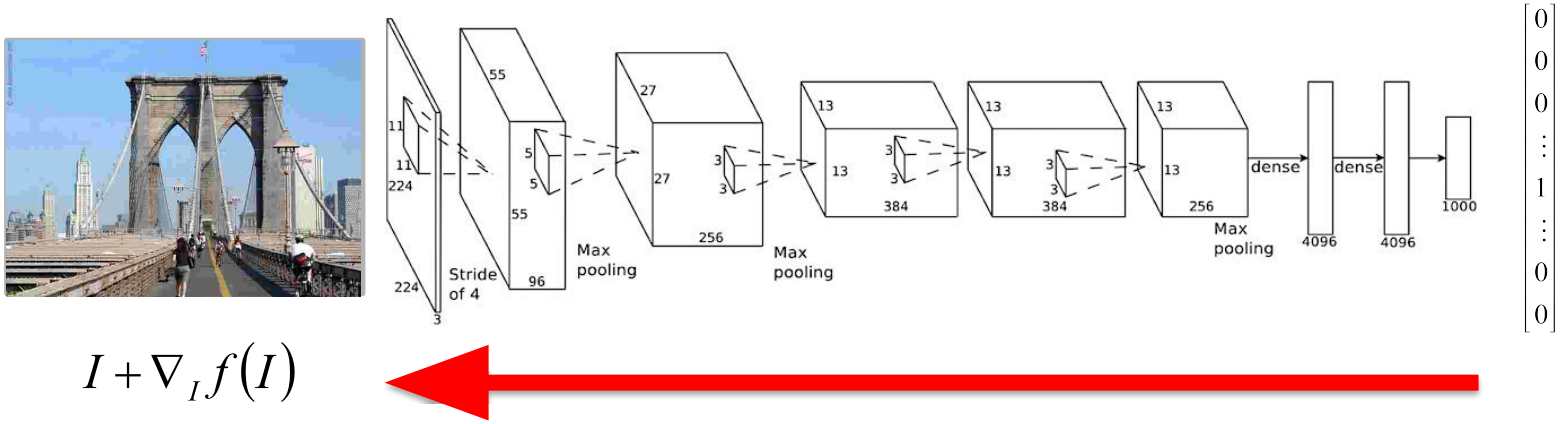


NOTE IMPORTANTE: pour cette opération, les poids du réseau **sont gelés**

seule l'image d'entrée est modifiée

Deep dream : même opération mais partant d'une image réelle

1. Propager une image dans le réseau
2. Sélectionner une couche du réseau
3. Rendre le gradient équivalent à son activation

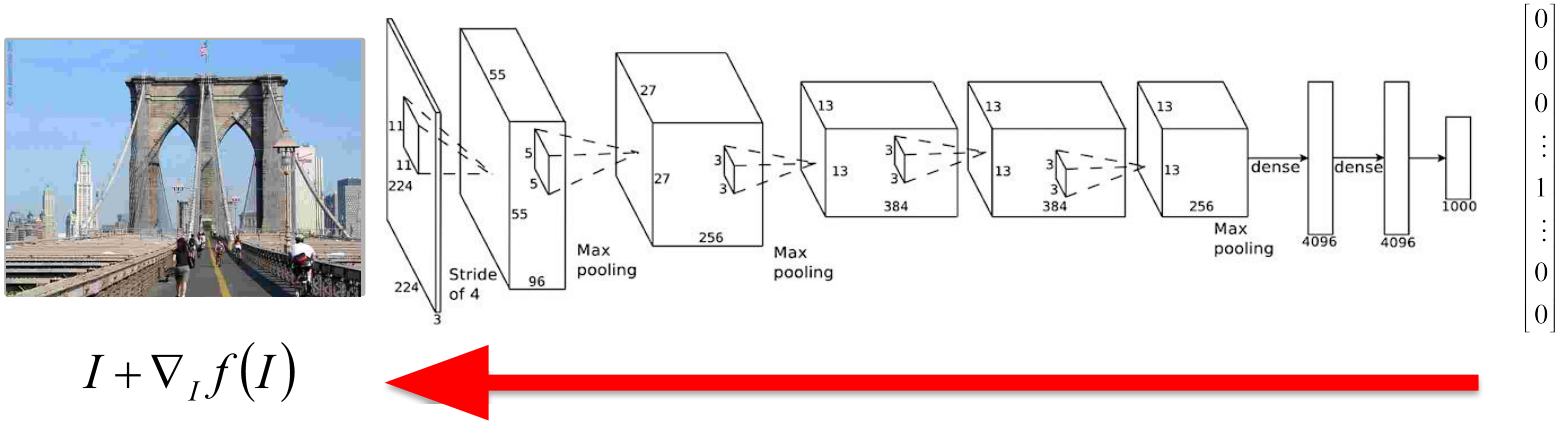


NOTE IMPORTANTE: pour cette opération, les poids du réseau **sont gelés**

seule l'image d'entrée est modifiée

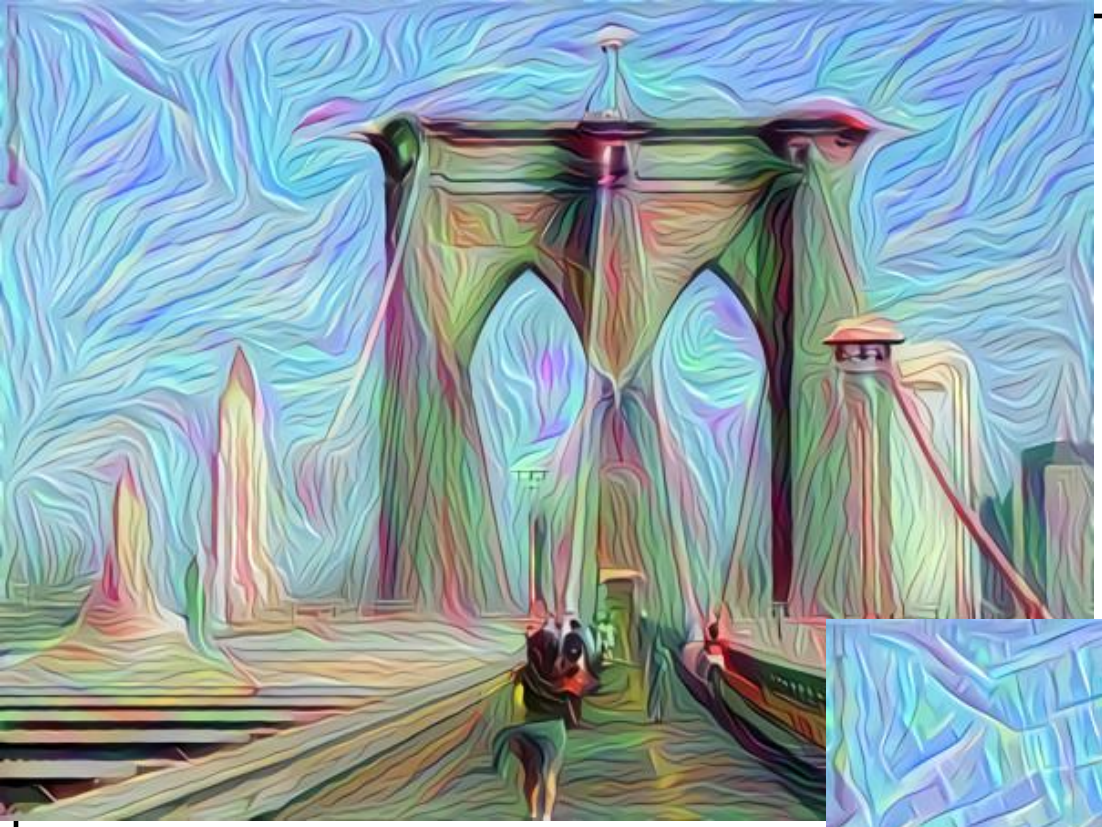
Deep dream : même opération mais partant d'une image réelle

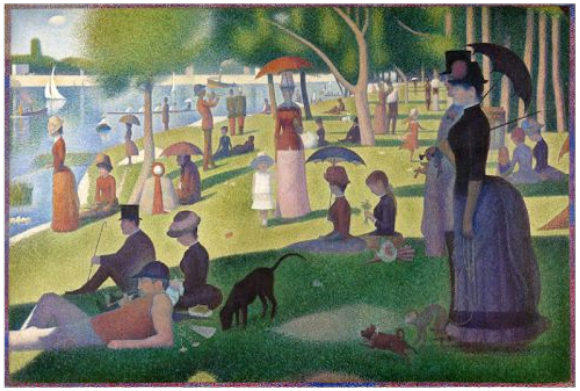
1. Propager une image dans le réseau
2. Sélectionner une couche du réseau
3. Rendre le gradient équivalent à son activation
4. Rétropropager dans l'image

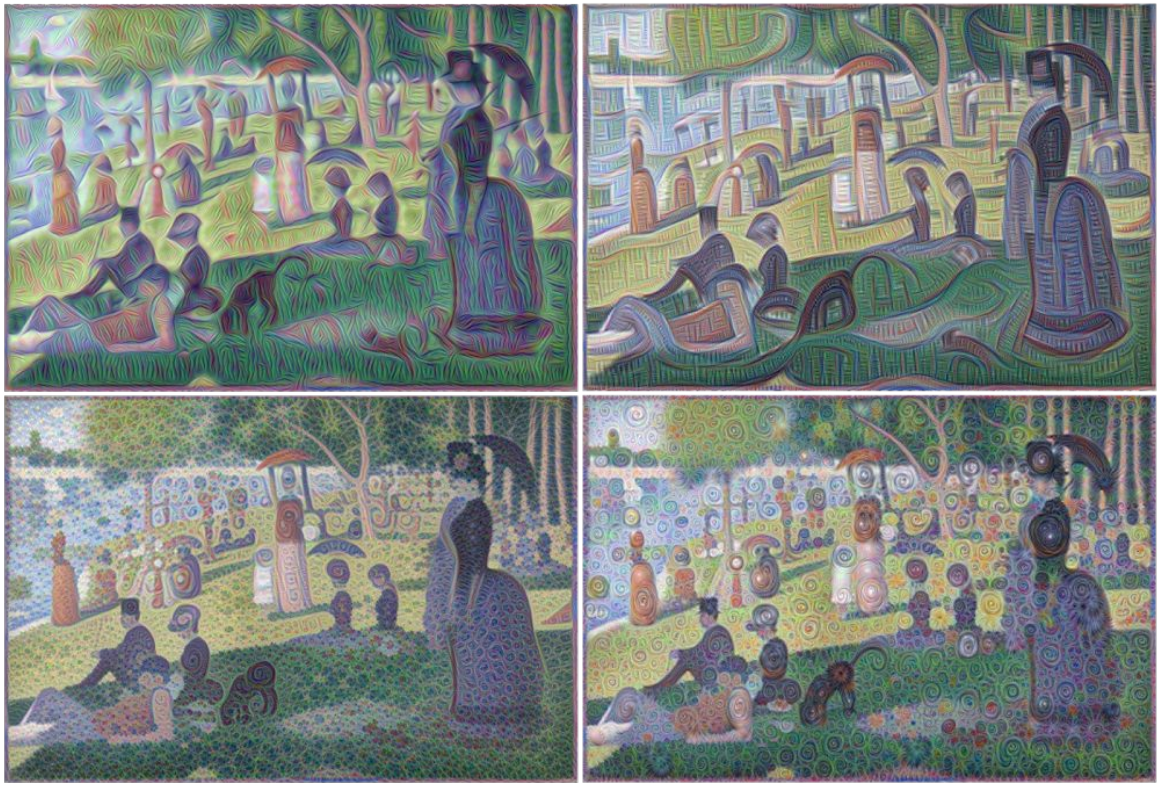


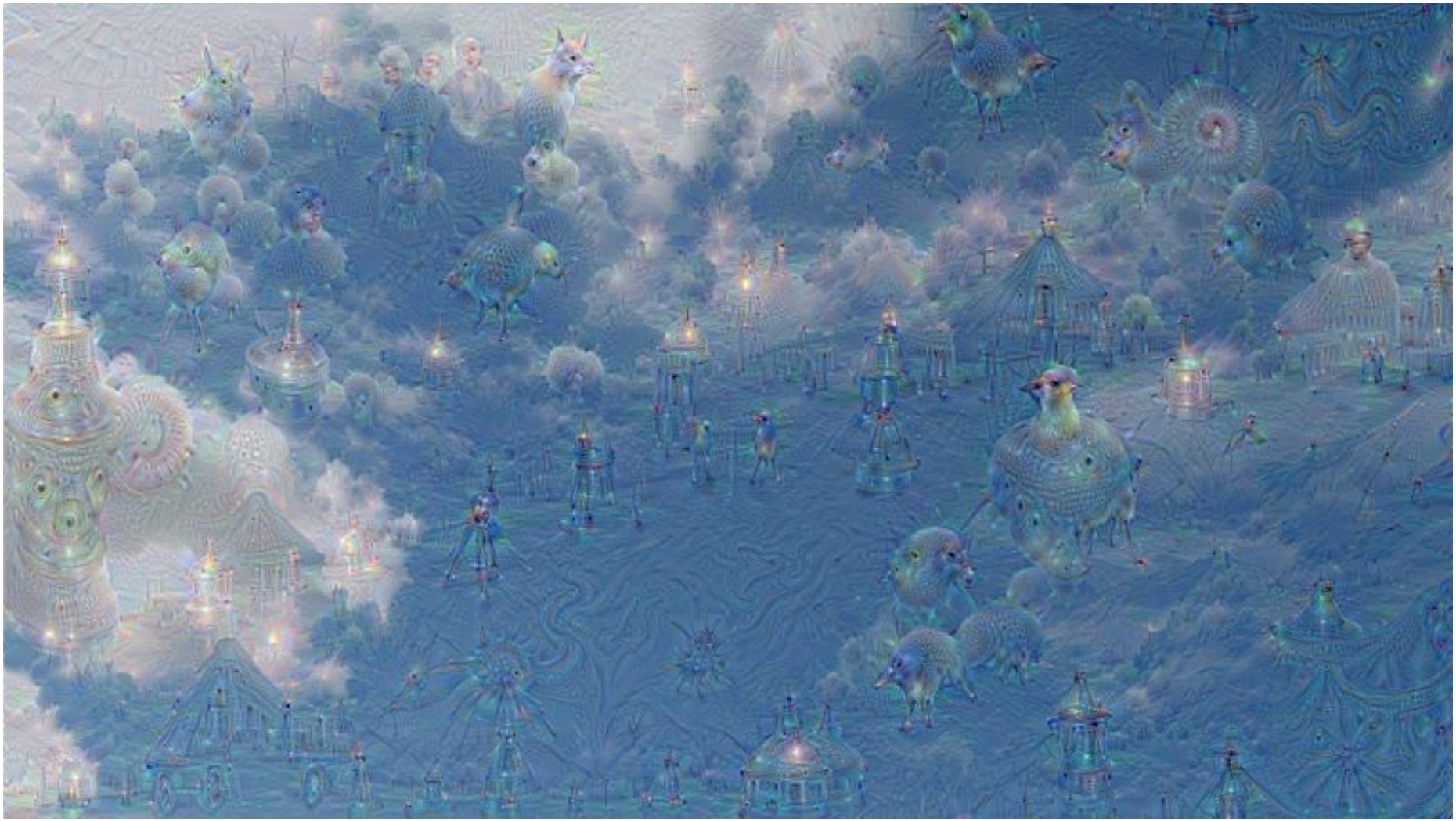
NOTE IMPORTANTE: pour cette opération, les poids du réseau **sont gelés**

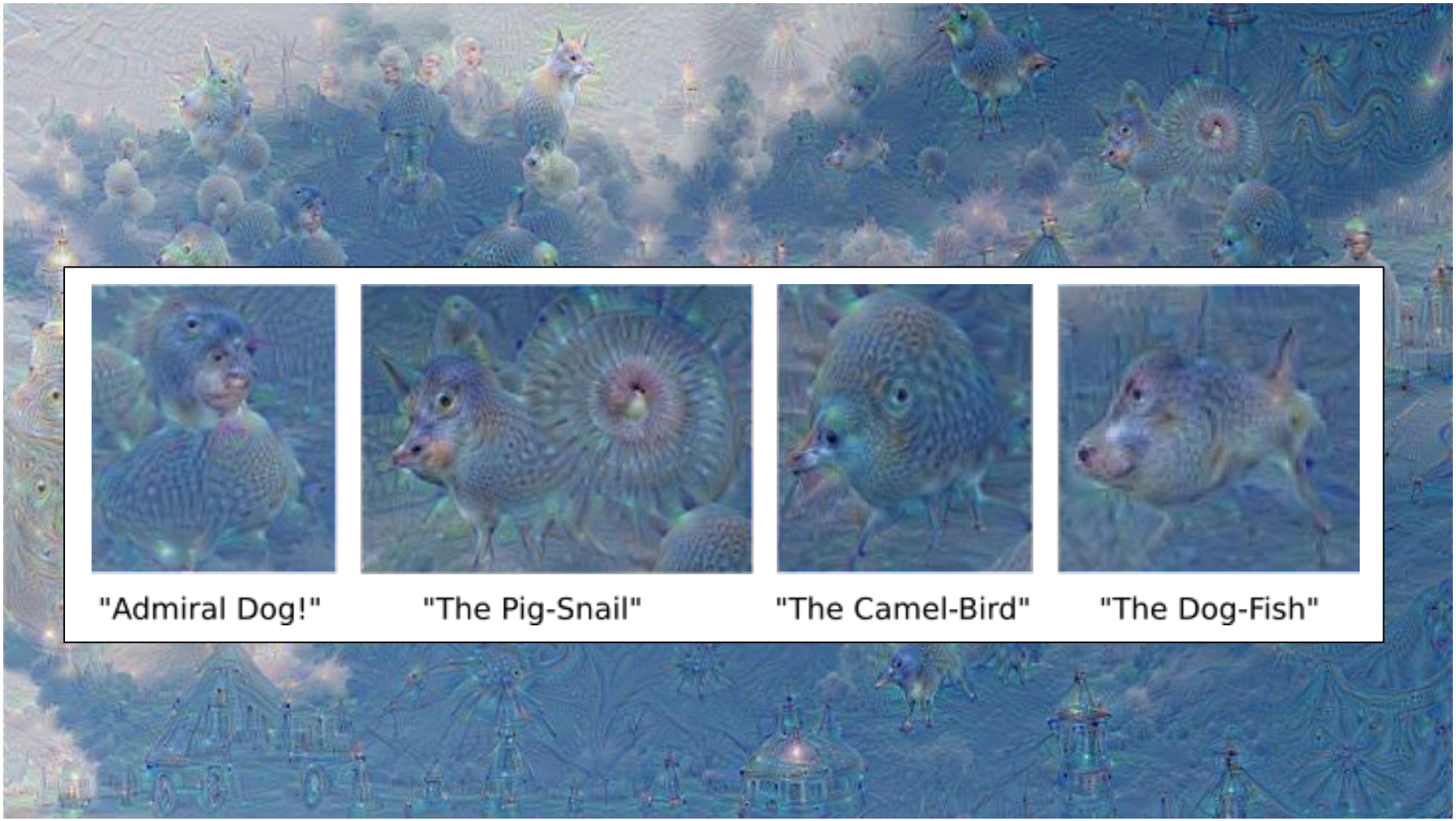
seule l'image d'entrée est modifiée



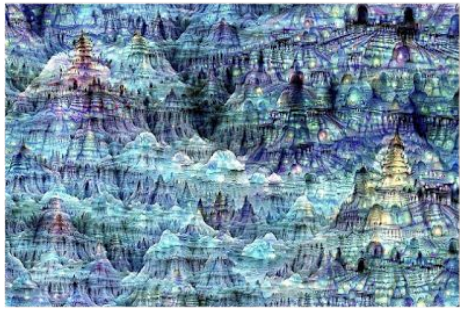
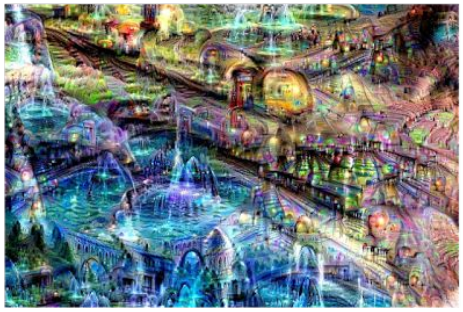


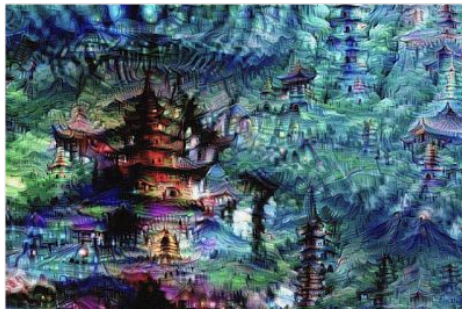




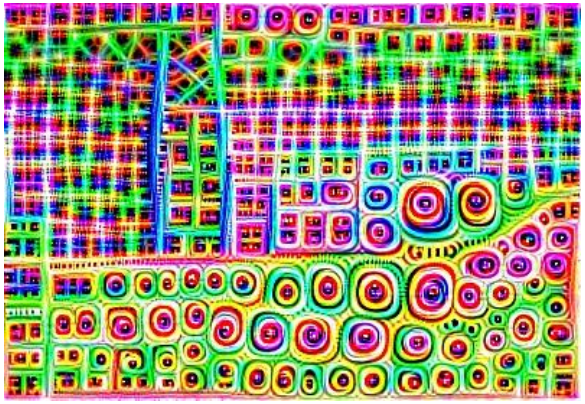

"Admiral Dog!"
"The Pig-Snail"
"The Camel-Bird"
"The Dog-Fish"





deepdreamgenerator.com/



deepscopeapp.com/

Conclusion

- Les CNNs ne sont pas *si* des boîtes fermées que ça
- Plusieurs techniques sont disponibles pour se renseigner sur les caractéristiques apprises par un CNN
  - Projection/Clustering
  - Occlusion
  - Rétropropagation
- Il faut sonder le réseau plutôt qu'obtenir une réponse claire
- On peut s'amuser avec certaines techniques