ÉCOLE CENTRALE DE LYON

MSO 3.4 - DEEP LEARNING

RAPPORT DE TD

TD4 - Style Transfer

Étudiant

Elion Mehdi

Enseignants

Liming Chen

Emmanuel Dellandréa

7 Mars 2019

Table des matières

Introduction

Le présent rapport a pour objet de présenter une technique d'utilisation du gradient d'images permettant d'altérer une image selon un certain style artistique : le transfert de style. Un notebook contenant les codes complétés sera joint à ce rapport pour le compte rendu final.

Notons que, dans le cadre de ce TD, nous utiliserons le réseau de neurones convolutionnel (CNN) pré-entraîné (sur la banque d'image *ImageNet*) *du* nom de *SqueezeNet* car il permet d'atteindre de bons résultats avec un nombre raisonnable de paramètres. Nous pourrons donc exécuter les différents calculs sur CPU.

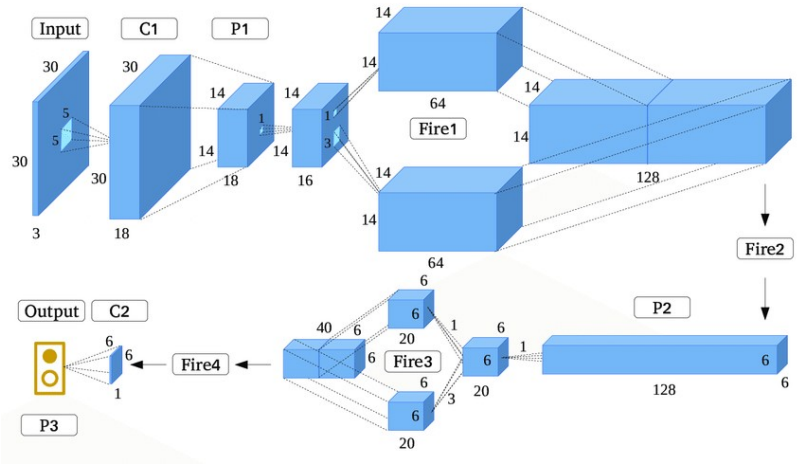
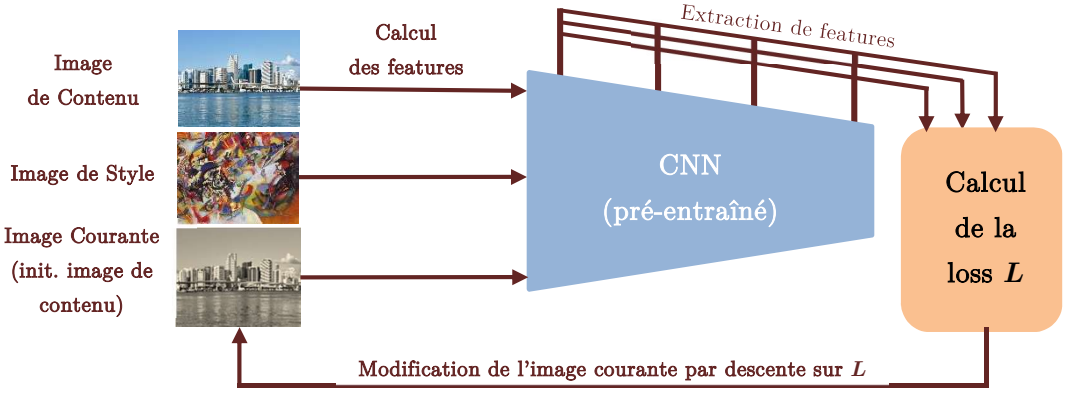


Figure 1 : Architecture de CNN SqueezeNet

Notons également que les paramètres du modèle de CNN pré-entraîné que l'on utilise ne changent dans aucune des trois techniques. Ce sont les valeurs des pixels de l'image en entrée du modèle qui sont modifiées.

1 Principe du Transfert de Style

L'idée générale du transfert de style revient à générer, à partir de deux images de référence, une nouvelle image qui reproduit le contenu de l'une (que l'on nomme alors image de contenu) et le style de l'autre (que l'on appelle alors image de style). Ici, l'image courante sera initialisée comme une copie de l'image de contenu.



Pour ce faire, on extrait les features des images à l'aide du CNN pré-entraîné puis on calcul la valeur d'une fonction de perte, ou « loss », qui permet de traduire mathématiquement ces critères. Cette loss se décompose en trois termes :

- la **content loss** L_c qui permet de mesurer et pénaliser la différence de features entre l'image courante et l'image de contenu
- la **style loss** L_s qui permet de mesurer et pénaliser la différence de features entre l'image courante et l'image de contenu
- la **total variation loss** L_{tv} qui permet de mesurer et de pénaliser le manque de « continuité » dans les valeurs de pixels de l'image

Une fois que l'on a calculé cette loss, on en calcule le gradient par rapport aux pixels de l'image courante, puis on l'utilise pour modifier la valeur de ces pixels dans le but de minimiser cette loss : c'est la descente du gradient.

On réitère ensuite cette procédure un certain nombre de fois jusqu'à obtenir une image convenable. En pratique, on effectuera 200 itérations.

Dans les sous sections suivantes, nous détaillerons le calcul des différentes fonctions de perte.

1.1 Content Loss

Soit une image I de taille $C \times H \times W$ où

- C est le nombre de channels (ici 3 car on traite des images RGB)
- H est le nombre de pixels en hauteur
- W est le nombre de pixels en largeur

Après extraction passage de l'image I dans le CNN pré-entraîné, on obtient, en sortie de la couche ℓ et après redimensionnement, ses features F_I^ℓ sous la forme d'un vecteur de taille $N^\ell \times M^\ell$ où :

- N^ℓ est le nombre de channels C^ℓ de la couche ℓ du CNN
- $M^\ell = H^\ell \times W^\ell$ est le de la hauteur H^ℓ et de la largeur W^ℓ de la couche ℓ du CNN

Pour rappel, la content loss doit mesurer la différence entre les features F^ℓ de l'image courante et les features P^ℓ de l'image de contexte. Ici on utilisera la somme des différences au carré multipliée par le coefficient de contenu :

$$L_c = \omega_c \sum_{i=1}^{N^\ell} \sum_{j=1}^{M^\ell} (F_{ij}^\ell - P_{ij}^\ell)^2$$

1.2 Style Loss

La style loss a pour objectif de mesurer la différence de style entre l'image courante et l'image de style. Pour cela, on choisit de mesurer la corrélation entres les features de chaque couche du CNN à l'aide de la matrice de Gram.

Etant donné une feature map F^ℓ de taille $1 \times C_\ell \times M_\ell$, sa matrice de Gram est de taille $1 \times C_\ell \times C_\ell$ et est définie par :

$$\forall (i,j) \in [|1, C_\ell|]^2, \quad G_{ij}^\ell = \sum_{k=1}^{M_\ell} F_{ik}^\ell F_{jk}^\ell$$

Soit $\mathcal{L}$ l'ensemble des couches du CNN dont on extrait les feature maps. Pour $\ell \in \mathcal{L}$, on note :

- G^ℓ la matrice de Gram de l'image courante
- A^ℓ la matrice de Gram de l'image de style
- ω_ℓ le coefficient de style sur la couche ℓ

La style loss s'exprime ensuite comme somme des différences au carré des matrices de Gram de l'image courante et de l'image de style pour chaque couche de feature :

$$L_s = \sum_{\ell \in \mathcal{L}} \omega_\ell \sum_{(i,j) \in [|1, C_\ell|]^2} (G^\ell_{ij} - A^\ell_{ij})^2$$

1.3 Total Variation Loss

La total variation loss a pour objectif de mesurer et de pénaliser l'absence de continuité dans les valeurs de pixels de l'image courante afin de garantir de faibles variations de couleurs. Etant donné un coefficient de variation totale ω_{tv}, L_{tv} s'exprime comme la somme pour chaque couleur et pour chaque pixel de la différence au carré avec le pixel voisin, dans la direction horizontale et dans la direction verticale :

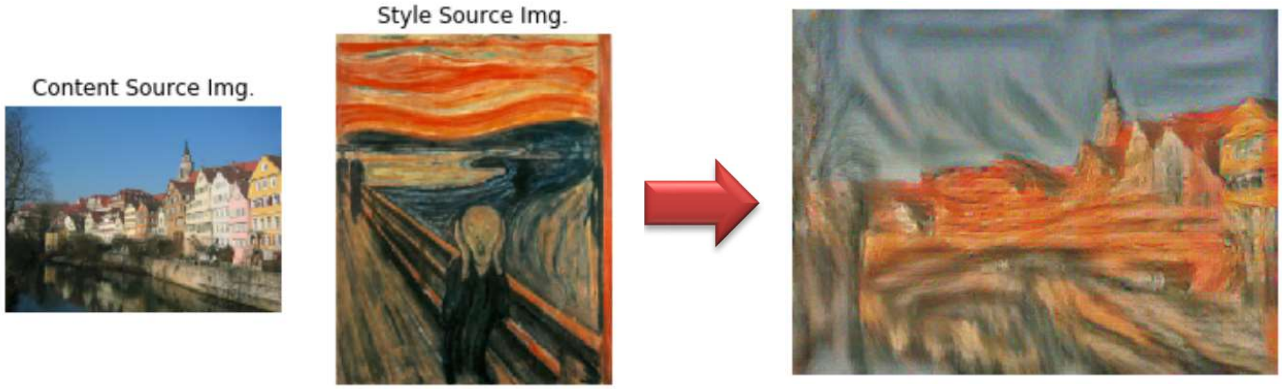
$$L_{tv} = \omega_{tv} \sum_{c=1}^{3} \sum_{i=1}^{H-1} \sum_{j=1}^{W-1} ((x_{i,j+1,c} - x_{i,j,c})^2 + (x_{i+1,j,c} - x_{i,j,c})^2)$$

2 Transfert de Style : Expérimentations

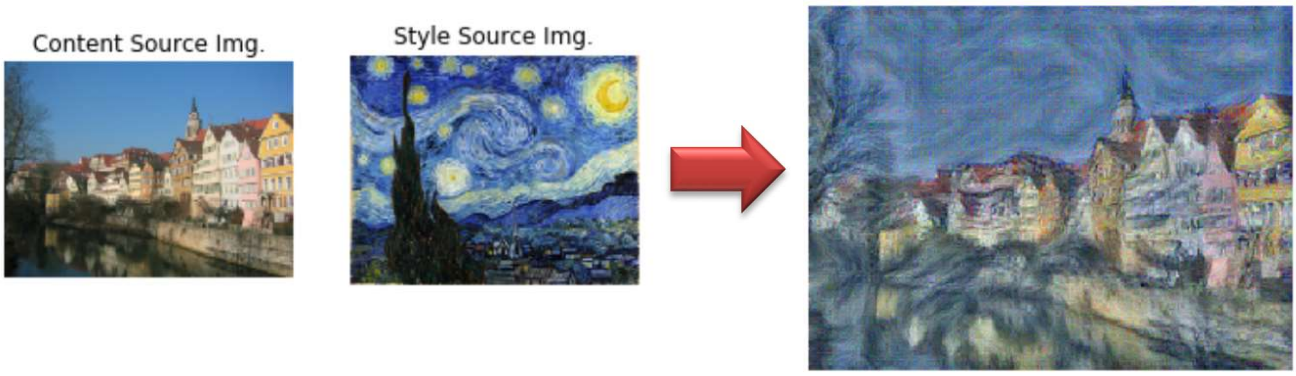
2.1 Essais et Résultats

Dans cette section, nous allons montrer quelques résultats préliminaires obtenus à l'aide du code complété sur le notebook. Notons que les résultats fournis sont obtenus à l'issue de 200 itérations. Les paramètres (poids, couches de CNN...) utilisés pour obtenir ces résultats sont fournis dans le code.

Contenu « Tubingen » / Style « Scream »



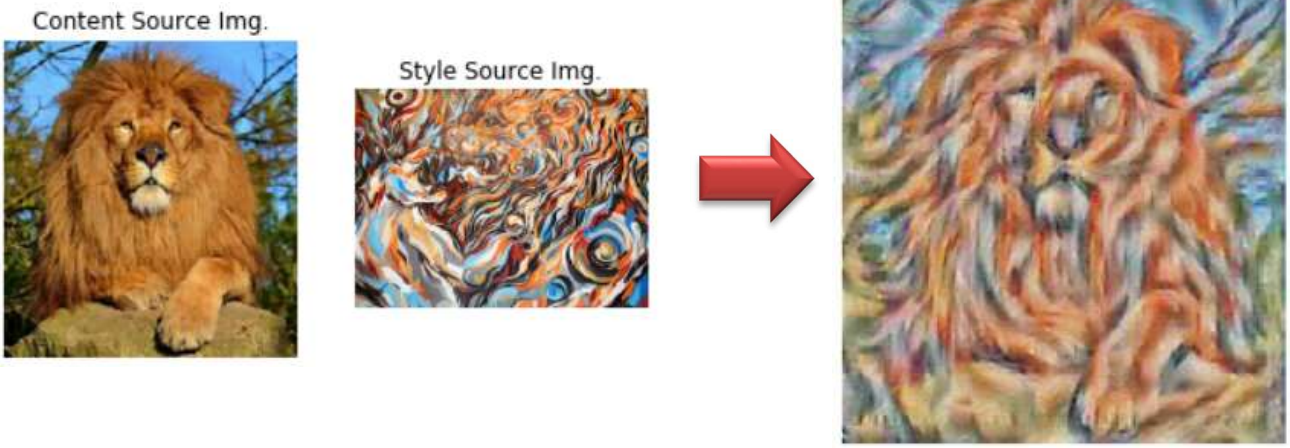
Contenu « Tubigen » / Style « Starry night »



Contenu « Mont Fuji » / Style « Estampe Japonaise »

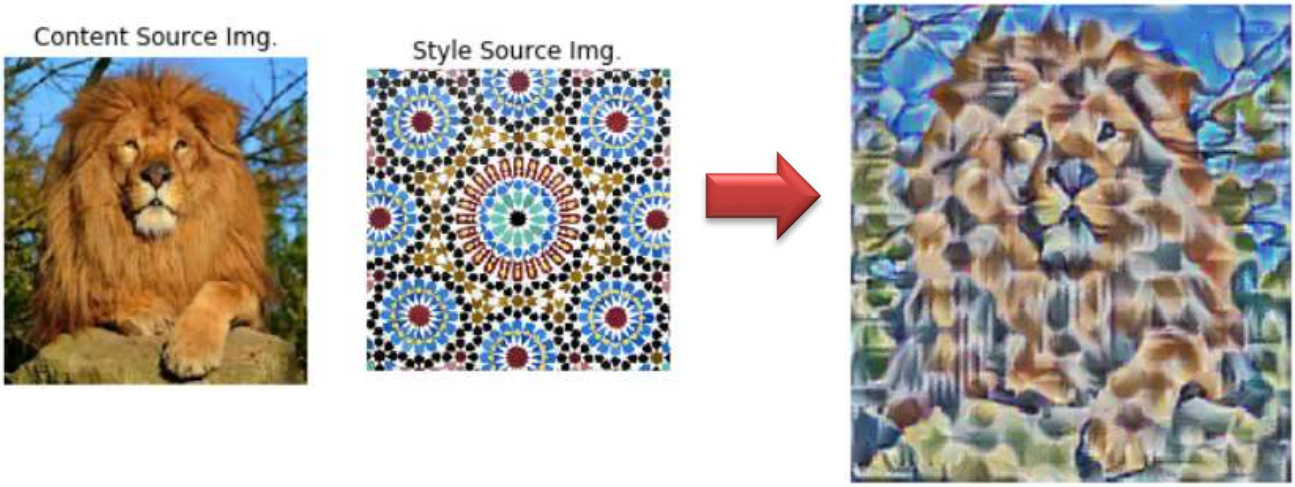
Contenu « Lion » / Style « Muse »

Contenu « Lion » / Style « Fire »



Contenu « Lion » / Style « Waves »

Contenu « Lion » / Style « Zellige »



Commentaires généraux

Sur tous les essais précédents, on arrive à générer des images dont le contenu rappelle celui de l'image de contenu et dont le style évoque celui de l'image de style. La procédure que l'on a implémentée et les fonctions de perte que l'on a définies fonctionnent donc convenablement. Toutefois, on peut d'ores et déjà émettre quelques commentaires à partir des résultats obtenus.

On constate que l'algorithme conserve le contenu dans tous les cas. En effet, bien que le style ait changé, on reconnait toujours le lion, le mont Fuji ou encore les bâtiments au bord du cours d'eau.

En revanche, l'imitation du style dans l'image générée est beaucoup plus flagrante lorsque ce style consiste en un jeu de teinte et de formes géométriques. En effet, dans le cas du lion, on retrouve très facilement les formes géométriques et les alternances de couleurs de la mosaïque « zellige », de même que l'on retrouve les ondulations et les alternances de couleurs des styles « wave » et « fire », ainsi que les couleurs et les formes polygonales du style « muse ».

En revanche, lorsque le style à copier n'a pas de caractéristiques géométriques particulières, le résultat est moins flagrant. Par exemple, le style des estampes japonaises est plus subtil : l'algorithme parvient à copier les teintes de couleurs mais le style on ne retrouve pas le style artistique. On retrouve plutôt un vague aspect texturé de papier. On peut supposer que cela est dû au modèle de CNN dont les features rendent mieux compte de certaines formes que d'autres.

Enfin, en zoomant sur certaines images, on peut observer des discontinuités de couleur sur certains pixels. Cela est certainement dû à un poids trop élevé sur L_{tv} ou bien à des poids trop faibles sur L_c ou L_s. C'est ce que nous allons vérifier dans la section suivante.

2.2 Influence paramètres de la loss

Dans cette section, nous allons nous intéresser à l'influence des différents paramètres de l'algorithme, et en particulier à l'influence des termes de la loss sur le résultat. Pour cela, nous allons prendre comme référence l'image du lion avec le style « wave » dont les paramètres (de référence) sont donnés ci-dessous :

```
# lion + waves
params3 = {
    'content_image' : 'styles/lion.jpg',
    'style_image' : 'styles/lines.jpg',
    'image_size' : 192,
    'style_size' : 192*3,
    'content_layer' : 3,
    'content_weight' : 7e-2,
    'style_layers' : [1, 4, 5, 7],
    'style_weights' : [450000, 2000, 20, 4],
    'tv_weight' : 5e-2
}

style_transfer(**params3)
```

Le résultat obtenu avec ces paramètres est rappelé ci-dessous :

Influence de L_{tv}

Notons que sur l'image de référence ci-dessus à droite, certaines zones paraissent pixélisées. Comme nous l'avion suggéré précédemment, cela est très certainement en partie dû à un faible poids ω_{tv} sur la total variation loss L_{tv}.

En effet, comme on peut l'observer sur l'image ci-dessous en multipliant ω_{tv} par 100 par rapport à la valeur de référence, lorsque l'on augmente ω_{tv}, on obtient une image plus « douce » et moins pixélisée mais un peu plus floue en contrepartie.

Au contraire, lorsque l'on diminue ω_{tv}, on obtient une image très pixélisée voir striée dans les directions horizontale et verticale, d'où l'intérêt d'implémenter une total variation loss qui pénalise les variations de pixels dans ces directions. A titre d'exemple, l'image ci-dessous est obtenue avec $\omega_{tv} = 0$, tout autre paramètre demeurant inchangé.

Influence de L_c

La content loss a pour objectif de pénaliser l'écart au contenu de l'image de contenu. En effet, lorsque l'on annule ω_c, on obtient une image qui copie le style en question sans que le lion n'apparaisse nulle part. En revanche, lorsque l'on augmente w_c, le style s'estompe au profit du contenu de l'image de contenu. A titre d'exemple, les images ci-dessous permettent d'illustrer cela. Celle de gauche est obtenue en annulant ω_c et celle de droite est obtenue en multipliant ω_c par 5 par rapport à la valeur de référence.

$\omega_c = 0$ $\omega_c = 5\omega_{c0}$

Influence de L_s

Les poids utilisés dans la style loss portent sur un certain nombre (ici 4) de features maps issues du CNN. Chaque feature map a son propre poids et permet de générer certains motifs ou composantes du style en question, comme le montrent les images ci-dessous. Pour chacune de ces images, on ne conserve que l'un de poids w_l et on annule les autres.

$= 1$ $= 4$ $= 5$ $= 7$

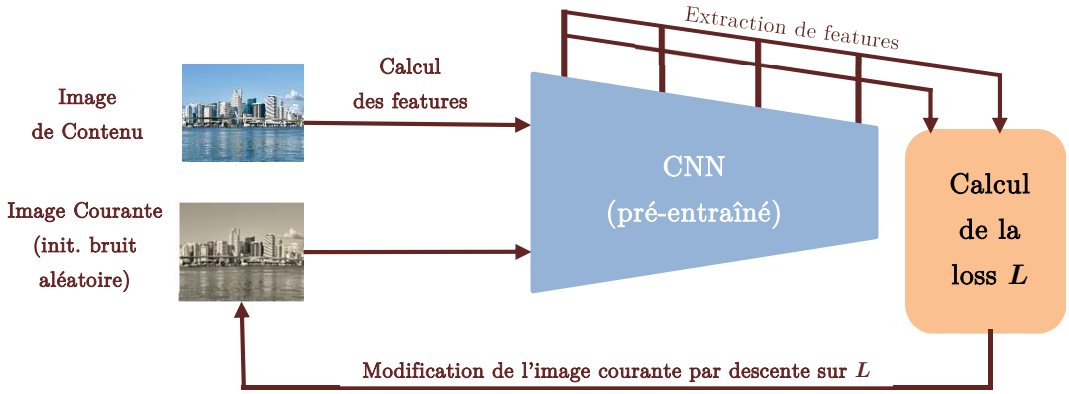
3 Feature Inversion

3.1 Définition

Le principe de l'inversion de features consiste, à partir d'une image initiale composé d'un bruit aléatoire, à reconstruire l'image de contenu à partir de ses features issues du CNN. Cela permet en outre de visualiser et de comprendre les features que le CNN est entraîné à reconnaitre.

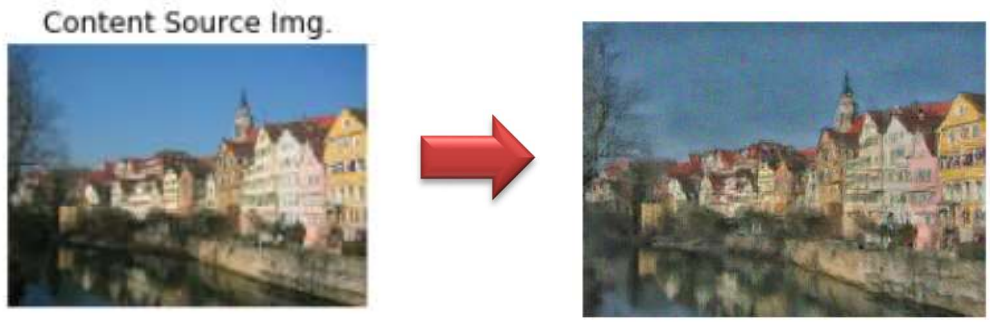
3.2 Principe

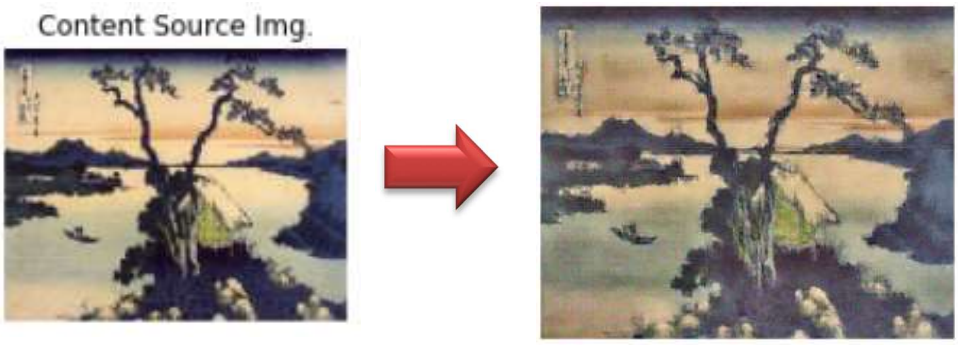
Le principe est sensiblement le même que pour le transfert de style à la différence près que l'on initialise l'image avec un bruit aléatoire et que l'on annule les coefficients de style.



3.3 Résultats

Voici le genre de résultats que l'on obtient avec cette technique.





Dans les deux cas, on reconnait dans l'image générer le contenu de l'image de référence, qu'il s'agisse des couleurs, des formes ou des objets contenus dans l'image. On peut donc conclure que les features que le CNN est entraîné à reconnaître sont pertinentes. On remarque toutefois que les images générées ont un aspect globalement plus « flou » que l'image d'origine. Cela est dû au fait que la loss porte sur un écart de features et non de pixels : on retrouve donc le contenu « sémantique » mais pas nécessairement avec exactitude.

Conclusion

À travers des techniques de génération d'images réalises à l'aide d'un CNN pré-entraîné et d'une descente de gradient sur des pixels, ce TD a permis d'explorer la sémantique des features reconnues par le CNN et divers moyen de les exploiter dans une fonction de perte afin d'atteindre un objectif précis.