

Comment mesurer la qualité d'une image ? Des métriques classiques aux métriques perceptuelles

Culture Sciences
de l'Ingénieur

Rémi KAZMIERCZAK

Édité le
03/10/2022

école
normale
supérieure
paris-saclay

Cette ressource est issue d'un stage de recherche à l'ENSTA de Rémi Kazmierczak, élève de 4^{ème} année au Département d'Enseignement et de Recherches Nikola Tesla de l'ENS Paris-Saclay. Ce texte a été relu par Thomas Rodet, enseignant chercheur au DER Nikola Tesla et au Laboratoire SATIE.

Dans beaucoup de tâches liées au traitement de l'image, la métrique de la qualité de l'image est le juge de paix de la qualité du travail fourni. Cependant, toutes les métriques ne se valent pas et il est important de bien cerner les limites de chaque opérateur.




*Figure 1 : À quel point mon image est détériorée ?
Une question simple d'apparence qui est en réalité plus complexe qu'il n'y paraît ...*

Dans cette ressource, nous dresserons un bref aperçu des principales métriques utilisées en traitement d'image, partant de la plus simple à la plus sophistiquée.

1 – Introduction

1.1 - Remarque préliminaire

Ce document est conçu pour être accompagné d'un script python, disponible en deux versions (en annexes de cette ressource [8]) :

- Un notebook collaborateur, qui est la solution la plus confortable, la partie importations de modules et de fichiers étant gérée automatiquement. Il s'agit d'un fichier similaire aux jupyter notebooks, qui consiste en une série de cellules de code à faire tourner en appuyant sur le bouton , le contenu du code est bien sûr modifiable.
- Un script python, qui a l'avantage de pouvoir tourner localement sur une machine, mais requiert de télécharger manuellement les fichiers et modules nécessaires.

Tourné tel quel, le script permet de retrouver les principaux résultats présentés dans cette ressource. Cependant, la philosophie de ce script est de manipuler les différents paramètres définis dans la ressource afin de visualiser de manière interactive leur influence.

1.2 - Représentation d'une image dans un ordinateur

Le traitement de l'image est la discipline qui consiste en l'étude des images numériques. Une image est codée dans un ordinateur sous la forme de pixels correspondant à des petits carrés de couleur d'intensité uniforme. Dans un ordinateur l'image est constituée d'un tableau bidimensionnel de pixel. Ces pixels sont codés sous la forme d'une valeur entière entre 0 et 255, soit $2^8=256$ donc un octet (la plus petite valeur adressable sur un ordinateur) dans le cas d'une image en niveau de gris. En ce qui concerne les images couleurs, chaque pixel est codé sur 3 canaux RGB (Red Green Blue), et chaque canal est codé par un entier entre 0 et 255.

Ainsi, on peut coder 16 millions de couleurs différentes $2^{(8 \times 3)} = 16,7 \times 10^6$. Mais cette représentation est très gourmande en place mémoire sur l'ordinateur. Par exemple, l'image de gauche de la Figure 1, image couleur de 237 x 400 pixels, occupe environ 300 kilos octet. Les images des appareils photos peuvent maintenant aisément capturer des images de 4 millions de pixels, c'est à dire qu'une image occupe 12 Mega Octets (12 millions d'octets) L'œil humain ne peut pas discriminer 16 millions de couleurs et stocker l'image brute prenant une place importante dans la mémoire de l'ordinateur, ces images sont généralement compressées avant d'être stockées (nous n'aborderons pas dans cette ressource la théorie de la compression des images). C'est pourquoi, lorsqu'on lit une image pour faire du traitement d'image (opération qui vise à modifier l'image ou en extraire de l'information), on utilise des programmes qui permettent de décoder l'image afin d'avoir accès au tableau tridimensionnel contenant les entiers codés sur 256 valeurs.

En guise d'exemple, on utilisera le module Pillow de python, qui est une librairie très utilisée pour traiter des images. C'est l'objet de la partie 1 du script fourni. Une fois l'image chargée, on peut la convertir en un objet de type np.ndarray.

Ainsi, pour une image en nuances de gris, on obtient :

```
Image nuances de gris :
[[ 96 116 117 ... 73 70 33]
 [ 93 107 111 ... 91 86 45]
 [ 90 98 109 ... 108 111 78]
 ...
 [240 241 243 ... 59 51 44]
 [236 238 241 ... 67 57 48]
 [235 236 239 ... 70 66 60]]
Taille de l'image :
(237, 400)
```




Figure 2 : Affichages des principales caractéristiques de l'image (Contenu et taille)

On remarque ainsi une caractéristique majeure de l'imagerie numérique. Une image équivaut à une série de valeurs correspondant à l'intensité du pixel associée (ici codée en 8 bits soit $2^8 = 256$ valeurs possibles). Plus le pixel associé est sombre, plus sa valeur est faible (comme dans le coin en bas à droite). Plus le pixel associé est clair, plus sa valeur est élevée, comme dans le coin en bas à gauche.

Pour ce qui est de l'image couleur, l'idée reste semblable, à ceci près qu'un pixel ne correspond pas à une unique valeur, mais à trois, comme illustré dans la figure 3. Il existe plusieurs façons d'encoder un pixel de couleur, mais une des plus répandues est le codage RGB (Rouge, Vert, Bleu).

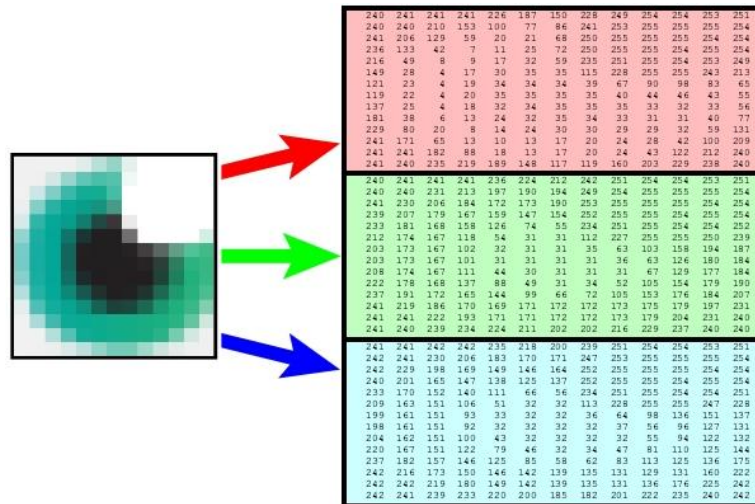


Figure 3 : Représentation sous forme numérique d'une image RGB : Chaque pixel correspond à une série de trois valeurs

Ainsi, on peut facilement représenter une image x numérique en tant qu'objet mathématique. Pour faciliter les calculs nous allons considérer que les pixels sont maintenant codés sur des valeurs réelles (c'est généralement ce qui est fait sous python où les fonctions de traitement d'images utilisent des tableaux de réels) :

$$x \in \mathbb{R}^{H \times W \times C}$$

- H : Nombre de pixels de hauteur de l'image (237 dans l'image représentant un burger)
- W : Nombre de pixels de largeur de l'image (400 dans l'image représentant un burger)
- C : Nombre de canaux (1 pour l'image en nuances de gris, 3 pour l'image couleur)

2 – Métriques de bases

Ces métriques sont basées sur la mesure de différence entre une image de référence et une image dégradée notée respectivement x et x_b . La première métrique utilisée est appelée SNR (Signal Noise Ratio). Dans ce cas, on fait le rapport entre l'énergie de l'image de référence (Signal) et l'énergie de la différence des deux images (Noise). Ce rapport étant dépendant de l'image de référence choisie afin de comparer les méthodes, la communauté a préféré le PSNR. Ce dernier compare la plus grande valeur de l'image à l'énergie de la différence normalisée. Enfin ces 15 dernières années a été introduit le SSIM afin d'avoir un critère qui tient compte des statistiques des images choisies et d'être le moins dépendant possible de l'image choisie.

2.1 - Définition des métriques les plus utilisées dans la communauté : PSNR et SSIM

Afin de mesurer rapidement le niveau de détérioration d'une image, des métriques se sont rapidement imposées. Nous exposons ici l'exemple de deux métriques très classiques.

Le **Peak Signal to Noise Ratio (PSNR)** est basé sur le calcul direct de l'écart pixel par pixel de chaque image. Ainsi, si l'on reprend les deux images x et x_b , le PSNR entre les deux images équivaut à :

$$PSNR(x, x_b) = \log_{20} \left(\frac{255^2}{\sqrt{MSE(x, x_b)}} \right)$$

Avec :

$$MSE(x, x_b) = \frac{1}{N} \sum_{i,j,c=0}^{H,W,C} (x^{i,j,c} - x_b^{i,j,c})^2$$

Cette mesure se concentrant sur des aspects locaux de l'image, on aime souvent allier cette métrique au **Structural SIMilarity (SSIM)** [1]. A contrario du PSNR, le SSIM se concentre plus sur des caractéristiques globales de l'image en considérant les valeurs statistiques de celle-ci :

$$SSIM(x, x_b) = \frac{(2\mu_{x_b}\mu_x + c_1)(2\sigma_{x_b}\sigma_x + c_2)(cov_{x_b x} + c_3)}{(\mu_{x_b}^2 + \mu_x^2 + c_1)(\sigma_{x_b}^2 + \sigma_x^2 + c_2)(\sigma_{x_b}\sigma_x + c_3)}$$

- μ_{image} : Espérance de l'image
- σ_{image} : Ecart type de l'image
- $cov_{image_1 image_2}$: Covariance entre deux images
- $c_1 = (k_1 L)^2, c_2 = (k_2 L)^2$ et $c_3 = \frac{c_2}{2}$: Constantes destinées à stabiliser la division. Valeurs par défaut : $k_1 = 0.01, k_2 = 0.03, L$ étant la valeur maximale d'un pixel (ici 255).

2.2 - Illustration des métriques de base sur le cas du bruitage de l'image

Afin de tester nos métriques, on va volontairement bruite nos images, un bruit souvent utilisé est le bruit dit additif gaussien. Soit une image numérique sous forme de valeurs comme exprimé dans la partie précédente x , l'image bruitée est alors :

$$x_b = x + b_{gauss}$$

Avec :

$$b_{gauss} \sim \mathcal{N}(\mu, \sigma^2)$$

- μ : Espérance du bruit
- σ : Ecart type

La partie 2 du script consiste au codage et à l'affichage d'une image bruitée, ainsi, une image bruitée aura l'aspect ci-dessous :



Figure 4 : Image bruitée ($\mu = 0, \sigma = 50$)

Le script de la partie 3 calcule ces deux métriques dans plusieurs configurations. Calculons d'abord les valeurs pour un bruit gaussien de plus important :



Figure 5 : Image correspondant aux valeurs ; $\sigma = 10$, $PSNR = 28.26$, $SSIM = 0.805$



Figure 6 : Image correspondant aux valeurs ; $\sigma = 50$, $PSNR = 15.14$, $SSIM = 0.292$



Figure 7 : Image correspondant aux valeurs ; $\sigma = 100$, $PSNR = 10.45$, $SSIM = 0.134$

On remarque bien que plus le bruit est de variance élevée, plus l'image est illisible, et plus le SSIM et le PSNR est mauvais (un SSIM parfait vaut 1, et plus la valeur du PSNR est élevée plus la qualité est bonne en échelle logarithmique).

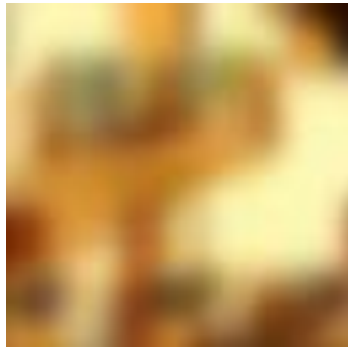
2.3 - Limites des métriques classiques

De manière générale, pour un même bruit ces métriques suffisent largement, ainsi, il n'est souvent pas nécessaire d'aller plus loin. Cependant, pour des cas plus complexes, utiliser de telles métriques ne suffit plus. C'est en particulier le cas lorsque la différence entre les images ne correspond pas à notre perception visuelle. L'exemple ci-dessous illustre cet effet. Cette fois-ci, nous nous intéressons au problème ci-dessous :



Figure 8 : Cette fois ci, l'image de référence est au milieu. À vue d'œil, quelle image semble entre Patch 0 et Patch 1 est la moins détériorée pour vous ?

Calculons à nouveau le PSNR et le SSIM, on trouve :



PSNR = 18.04, SSIM = 0.477



PSNR = 13.86, SSIM = 0.313

Figure 9 : Images correspondant aux valeurs :

Cette fois-ci la mesure va à l'encontre du jugement global : là où le regard humain a tendance à choisir la deuxième image, les métriques classiques ont tendance à choisir la première !

Une explication simple est que ces métriques se concentrent sur des caractéristiques de *bas niveau* tels que la luminance ou les différences entre les pixels, là où les humains sont plus sensibles aux détails de *haut niveau* telle la présence/clarté d'objets concrets comme une porte ou des fleurs.

Si on veut aller un peu plus loin dans l'explication, la perception humaine ajoute du sens à l'image : c'est à dire que l'on est très attaché à la reconnaissance de caractéristiques clés pour interpréter l'image. Dans l'image du patch 1 (Figure 8), on reconnaît par exemple la porte et le balcon même s'ils sont très déformés parce que l'on distingue mieux les bords, les motifs et la disposition des objets les uns par rapport aux autres. Dans cette image, la déformation engendre que les structures géométriques ne sont plus les mêmes, ce qui va conduire à une grande différence métrique bien que pour nous le sens est préservé. A contrario, la détérioration des textures, telle que celles du mur du patch 1, est moins perçue par l'œil humain, là où les critères mathématiques comme le PSNR et le SSIM vont signifier une erreur importante.

Pour mesurer la proximité entre une métrique et le jugement humain de manière plus rigoureuse, les auteurs de [2] ont créé une base de données. Cette base de données, nommée Berkeley-Adobe Perceptual Patch Similarity (BAPPS) dataset, consiste en une succession de tests tels que montrés sur la figure 8. Pour chacun de ces tests, on demande à un panel de testeurs de choisir quelle image est la plus proche de la référence entre l'image 1 et 2. Ainsi, le BAPPS dataset regroupe 28.8k jugements sur 9.6k images comportant une grande variété de bruits (colorisation, bruit gaussien, interpolations ...).

3 – Pouvoir perceptuel des réseaux de neurones

Afin de gérer le problème soulevé dans la partie précédente, le domaine du traitement du signal a fourni un outil extrêmement puissant : les réseaux neuronaux profonds (deep learning) [6].

Le principe est le suivant : on crée une fonction $Res_{\theta}(x)$ en empilant plusieurs fonctions $l_{\theta_i}^{(i)}$ appelés couches, admettant en entrée la sortie de la couche précédente, et ayant pour paramètre θ_i . Le but étant de s'approcher au mieux d'une hypothétique fonction $Res_{idéal}$ qui correspond à la fonction réalisant les meilleures prédictions $Res_{\theta idéal}(x) = y$ correspondant à une tâche particulière. Les tâches sont associées à des domaines variées (classification, détection, synthèse, régression ...) :

$$Res_{\theta}(x) = l_{\theta_N}^{(N)} \left(\dots \left(l_{\theta_2}^{(2)} \left(l_{\theta_1}^{(1)}(x) \right) \right) \dots \right)$$

Pour ceci, on fait varier les paramètres θ grâce a un set de données d'entraînement $X_{data} = \{x_1 \dots x_N\}$. Il existe plusieurs variantes dans lesquelles on a plus ou moins d'informations sur les données d'entraînement : X_{data} seul (self supervised) X_{data} et sortie associée y (supervised), X_{data} en grande quantité et sortie associée y en petite quantité (semi supervised) ...

L'intérêt ici est de pouvoir empiler une grande quantité de couches de neurones afin de modéliser des fonctions complexes (Figure 10). Cela étant permis par les récentes conjonctures permettant l'accessibilité à des bases de données toujours plus grandes, l'augmentation importante des capacités de calcul et les nombreux travaux de recherches effectuées ces dernières années.

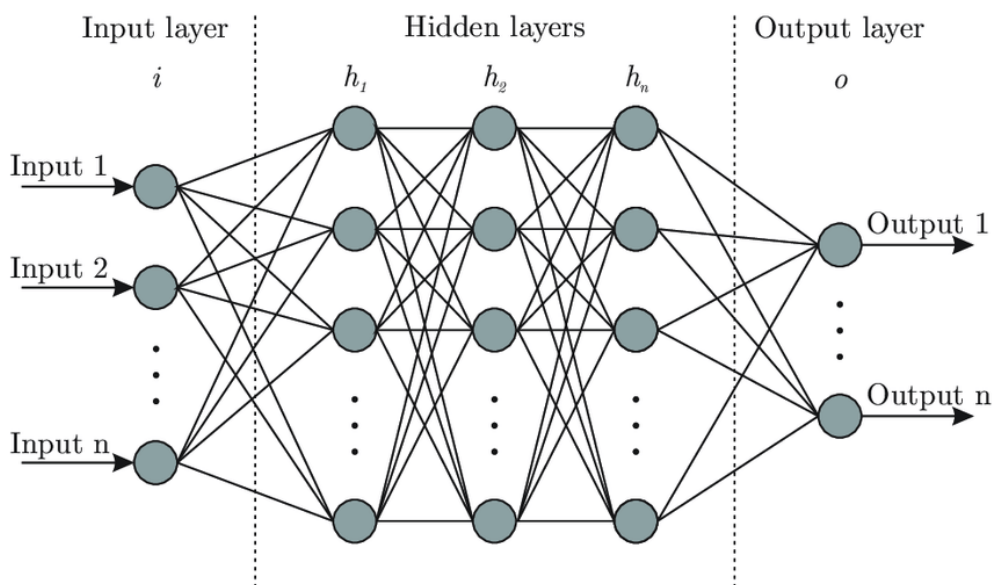


Figure 10 : Structure d'un réseau de neurones. On parle de deep learning lorsque le nombre de couches intermédiaires devient très important

Un point intéressant de ces réseaux est la manière dont ils traitent les données. La partie 4 du script fourni consiste en l'exploration d'un réseau pré-entraîné pour la classification. Entre autres, on observe le contenu de la sortie des couches intermédiaire (features maps) soit, si l'on veut par exemple la sortie du keme bloc $f_k(x)$ (k), on calculera la valeur :

$$f_k(x) = l_{\theta_k}^{(k)} \left(\dots \left(l_{\theta_2}^{(2)} \left(l_{\theta_1}^{(1)}(x) \right) \right) \dots \right)$$

Ainsi, si l'on observe le contenu des blocs :

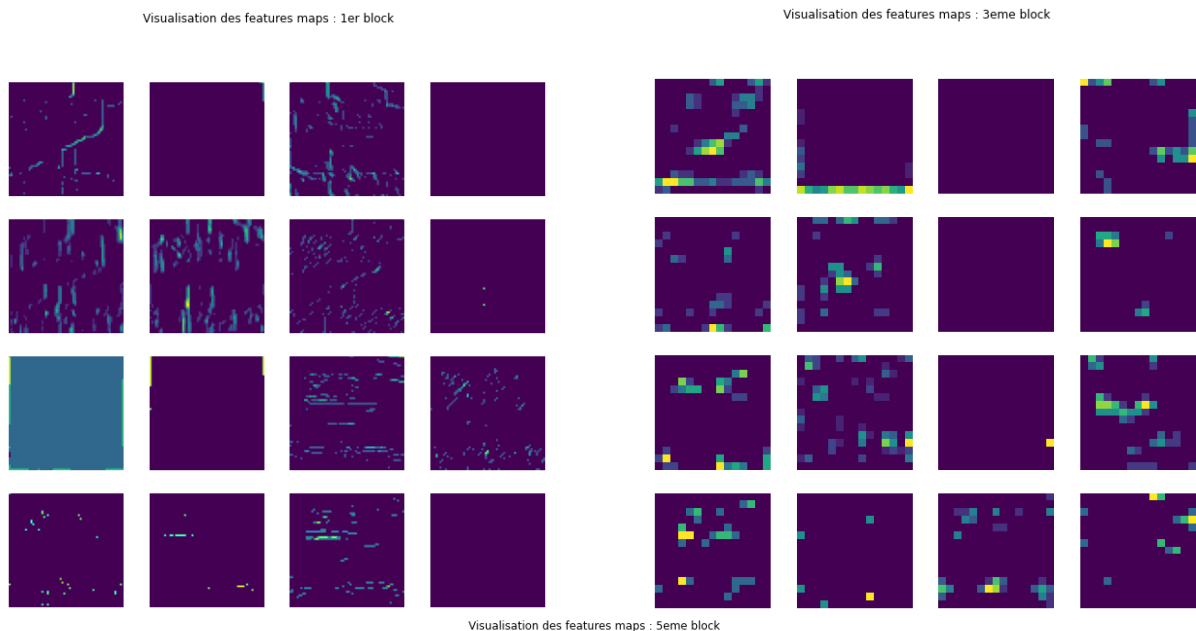


Figure 11 : Contenu des 16 premières couches pour différents blocs

On remarque ainsi une propriété intéressante des réseaux de neurones : plus on va profondément dans le réseau, plus les informations observées sont de *haut niveau*. En sortie du premier bloc, les informations extraites correspondent à des caractéristiques tels que les contours. Pour le 5ème bloc, les activations correspondent à des objets plus concrets tel que la présence de balcons.

Regardons ensuite pour les trois images de la figure 8, une feature map associée :

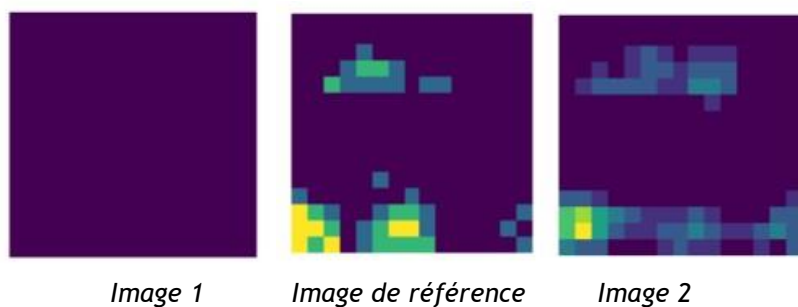


Figure 12 : La même feature map pour différentes images

On remarque ainsi l'utilité d'utiliser de telles méthodes pour la mesure de la qualité : là où aucune activation n'est remarquée sur l'image 1, on remarque que l'image 2 possède des activations plus

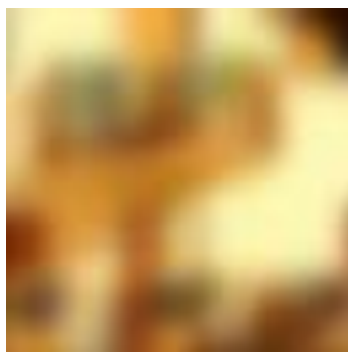
similaires à celles de l'image de référence, indiquant que les mêmes caractéristiques ont été détectées.

On propose alors, en se basant sur cette observation, d'introduire une mesure naïve de la qualité basée sur les sorties de ces blocs :

$$MSE_{DNN(x,x_b)} = \frac{1}{B_{max} - B_{min}} \sum_{k=B_{min}}^{B_{max}} \frac{1}{N_k} \sum_{i,j,c=0}^{H,W,C} (f_k(x^{i,j,c}) - f_k(x_b^{i,j,c}))^2$$

Cette mesure est assez similaire au MSE décrit précédemment, la différence étant que, au lieu de calculer l'écart quadratique moyen directement sur l'image, on prend en compte les sorties des derniers blocs d'un réseau de neurones pré-entraîné, la sortie du kème bloc étant notée f_k .

Ainsi, pour $B_{min} = 3$ et $B_{max} = 5$, on trouve :



MSE_{DNN} = 1.672



MSE_{DNN} = 1.628

Figure 13 : Résultats pour les deux images

Cette fois-ci, on a bien une métrique favorisant l'aspect perceptuel de l'image ! De manière générale, les auteurs de [2] montrent, en testant sur l'intégralité de BAPPS dataset, que pour la grande majorité des cas, les méthodes basées sur les réseaux de neurones sont beaucoup plus performantes que les métriques classiques présentées dans la deuxième partie. Ces observations ont mené à la création de fonctions de pertes dites *perceptuelles* [4]

Cependant, on peut nuancer ces observations en soulignant quelques désavantages de ces méthodes, en utilisant un réseau de neurones. Au-delà du temps de calcul plus long, on perd le sens physique et des propriétés mathématiques présentes dans les métriques citées plus haut. Pour le cas du bruit gaussien par exemple, on peut relier la valeur du MSE à la variance du bruit. De plus, on peut s'interroger sur le fait de considérer la perception humaine comme un objectif : si cela est souhaitable dans des tâches où l'on veut générer des images les plus agréables possibles pour des observateurs humains, il est plus questionnable de considérer les performances sur le BAPPS dataset comme absolues.

3.1 - Aller plus loin dans les métriques perceptuelles : MRPL loss

Une autre critique des métriques perceptuelles est le manque d'étude des différentes composantes de ces métriques. Pour aller plus loin et dans l'objectif de proposer une métrique perceptuelle optimale, [3] une étude des différents paramètres influant les manières d'extraire les informations de réseaux pré-entraînés est proposée, incluant l'architecture du réseau, la procédure d'entraînement, l'opérateur de normalisation, l'utilisation de matrices de Gram [7], l'utilisation de plusieurs résolutions et enfin les blocs utilisés.

À partir de cette étude une métrique a été créée : le Multi-Resolution Perceptual Loss (MRPL), dont le principe est illustré dans la figure 13. Les détails sur le choix des différents paramètres sont détaillés dans [3].

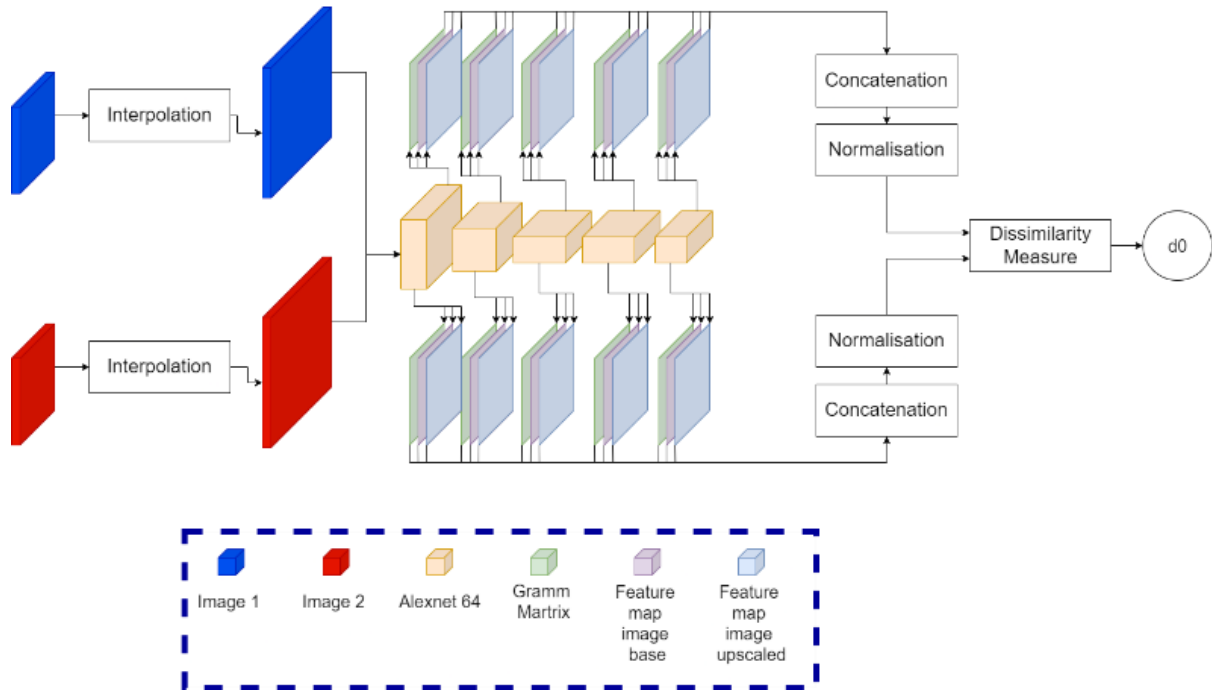


Figure 14 : Illustration de la MR-perceptual loss

3.2 - Extraction d'information du réseau

Un premier point important de la MRPL est la stratégie d'extraction des caractéristiques. En plus de récolter l'entrée brute, deux autres informations sont récoltées :

- Les matrices de Gramm f_k^{gramm} :

$$f_k^{gramm}[c_1, c_2](x_i) = \sum_{h,w} f_k(x_i)[h, w, c_1] f_k(x_i)[h, w, c_2]$$

Introduite dans [5], elle correspond à la matrice de covariance entre les descripteurs. Ainsi, partant du fait que chaque descripteur correspond à l'identification d'une caractéristique précise, cela permet de visualiser les interdépendances entre celles-ci. Par exemple, si les valeurs de la matrice de Gramm entre un descripteur extrayant la couleur bleue et celui extrayant des bords sont élevées, alors la corrélation entre présence de bords et présence de bleu est importante : on parle alors de style. Une manière concrète de visualiser cela est de construire une image ayant le contenu (matrices f_ω^b) d'une image A et les matrices de Gramm correspondant à celle d'une image B :

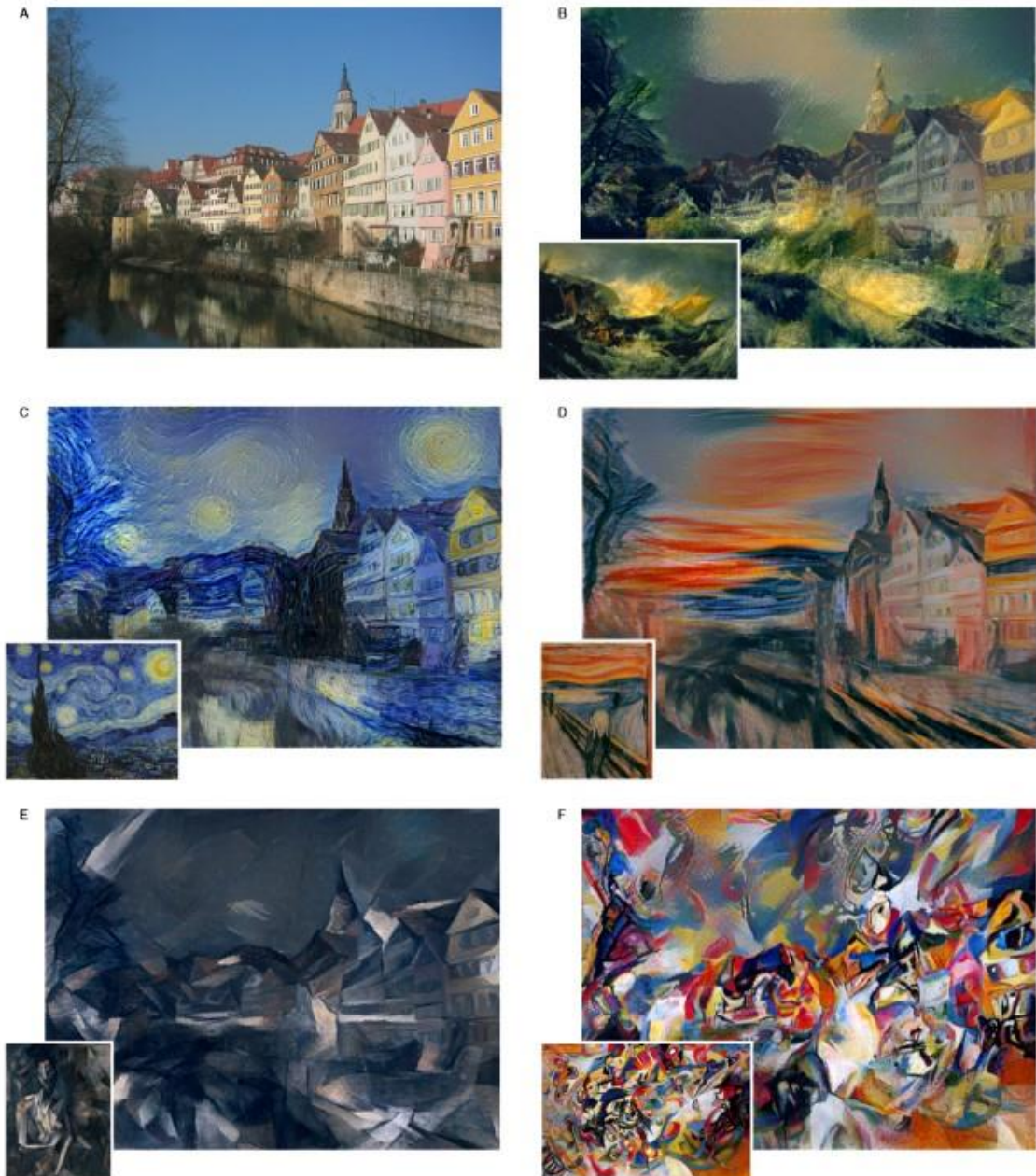


Figure 15 : Images obtenues en faisant correspondre de contenu de l'image A avec les matrices de Gramm de différents tableaux

- La sortie du réseau pour une entrée de résolution supérieure $f_k(x_{up})$: par interpolation, on augmente la résolution de l'image d'entrée par deux avant de l'introduire dans le réseau de neurones. Cela permet d'aider la détection des petits objets.

Ainsi, on utilise ici, non pas la sortie brute de chaque bloc, mais la concaténation de différentes informations $f_{concat}(x)$:

$$f_{concat}(x) = \{f_5(x), f_2^{gramm}(x), \dots, f_5^{gramm}(x), f_2(x_{up}), \dots, f_5(x_{up})\}$$

Remarquons que tous les blocs ne sont pas pris en compte : comme indiqué dans la deuxième partie, on omet volontairement les blocs peu profonds.

3.3 - Opérateur de dissimilarité

Un autre point important est l'opérateur de dissimilarité, à partir de f_{concat} , il a été remarqué qu'utiliser la fonction sigmoïde et l'entropie binaire croisée offre de meilleurs résultats ainsi on définit La métrique finale :

$$MRPL(x, x_b) = BCE(\sigma(f_{concat}(x)), \sigma(f_{concat}(x_b)))$$

Avec :

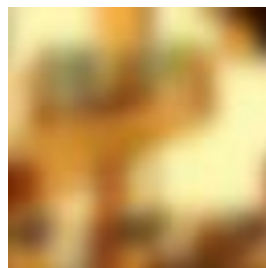
- BCE, l'entropie binaire croisée :

$$BCE(x, y) = -\frac{1}{NWC} \sum_{i,j,c=0}^{H,W,C} y_{i,j,c} \log x_{i,j,c} + (1 - y_{i,j,c}) \log(1 - x_{i,j,c})$$

- σ , la fonction sigmoïde (ici appliquée sur chaque élément) :

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

La partie 5 du script consiste au calcul du MRPL pour l'exemple du BAPPS dataset. Ainsi, on trouve :



MRPL = 1458



MRPL = 1128

Figure 16 : Résultats pour les deux images

On remarque que le jugement est bien similaire à celui de la métrique MSEDNN, avec un écart relatif encore plus grand. De manière générale les performances de la MR-perceptual loss sont significativement supérieures aux autres métriques similaires sur le BAPPS dataset

4 – Conclusion

La notion de qualité d'une image est assez subjective. En fonction des caractéristiques (taux de bruit, reconnaissance des objets ...), de la rapidité de calcul et des propriétés de l'opérateur, la méthode de mesure peut changer drastiquement.

Pour ce qui est de tâches simples visant à mesurer l'importance d'un bruit en particulier, les métriques classiques (Partie 2), sont adaptées. Cependant, lorsque la tâche consiste à mesurer des distorsions de plusieurs types et lorsque l'on veut mesurer une perception plus proche de l'œil humain, des méthodes basées sur des réseaux neuronaux profonds sont plus adaptés.

Références :

[1]: Zhou Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," in IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600-612, April 2004, doi: 10.1109/TIP.2003.819861.

[2]: R. Zhang, P. Isola, A. Efros, E. Shechtman and O. Wang, "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric," in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 2018 pp. 586-595.

[3]: Rémi Kazmierczak and Gianni Franchi and Nacim Belkhir and Antoine Manzanera and David Filliat (2022). A study of deep perceptual metrics for image quality assessment

[4]: Johnson, Justin & Alahi, Alexandre & Li, Fei-Fei. (2016). Perceptual Losses for Real-Time Style Transfer and Super-Resolution.

[5]: Gatys, Leon & Ecker, Alexander & Bethge, Matthias. (2015). A Neural Algorithm of Artistic Style. arXiv. 10.1167/16.12.326.

[6]: Deep Learning (Ian J. Goodfellow, Yoshua Bengio and Aaron Courville), MIT Press, 2016.

[7]: Conway JH, Sloane NJA (1998) Sphere Packings, Lattices and Groups, 3rd edn. Springer-Verlag, New York

[8]: Annexes Script :

- Par script python : Comment mesurer la qualité d'une image ? Des métriques classiques aux métriques perceptuelles, R. Kazmierczak, https://eduscol.education.fr/sti/si-ens-paris-saclay/ressources_pedagogiques/comment-mesurer-la-qualite-dune-image
- Par notebook collab : <https://colab.research.google.com/drive/1bsv1e1XYxY2kvKQxiw0LlLxut8hy0ja?usp=sharing>