

Rapport N°6

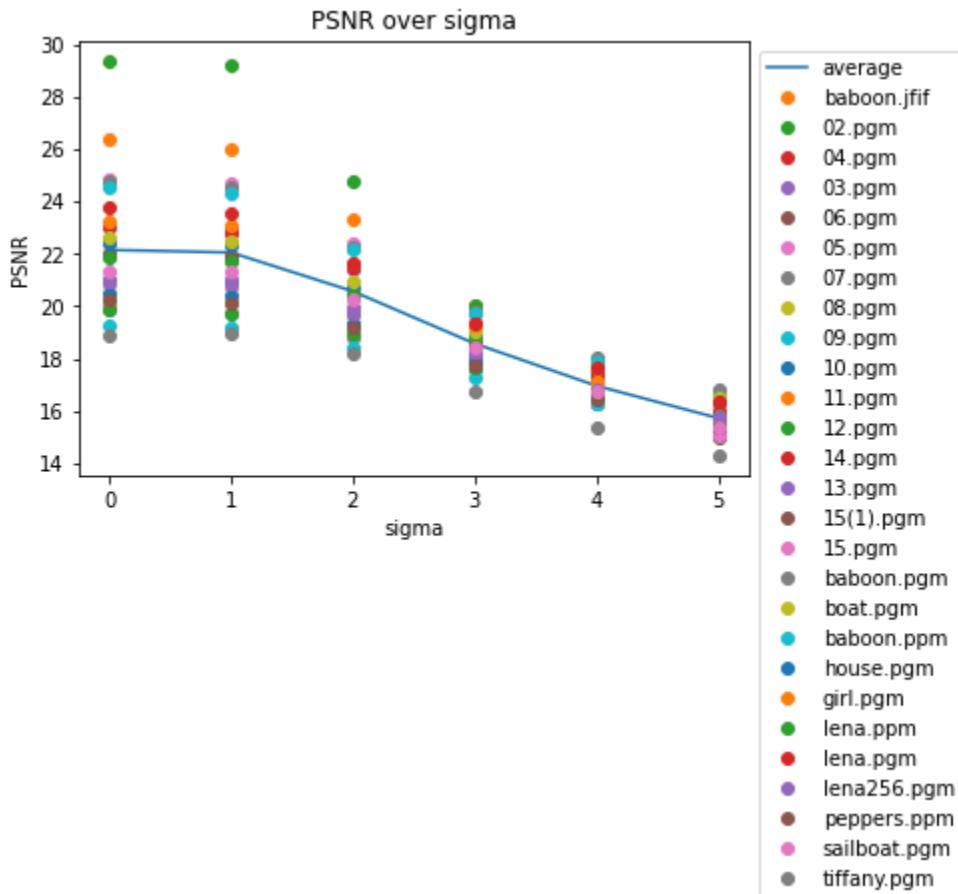
Semaine : 28/11/2022

Ce compte rendu est le sixième dans le cadre du projet image du premier semestre de Master 2 IMAGINE. Il comprend nos avancées dans la recherche et l'implémentation de l'environnement de recherche.

Taches effectuées

Au niveau des CNN, nous avons essayé d'évaluer la qualité du résultat des différents débruitageurs. Ces réseaux ont été entraînés sur un bruit gaussien ayant un écart-type (sigma) à 10% de la valeur maximale. On souhaite également analyser la qualité des réseaux en fonction du paramètre sigma.

Premièrement, on évalue le CNN utilisant des blocs en niveau de gris pour débruiter une image. Pour cela, on bruite une image avec un sigma, puis on débruit en utilisant le CNN et enfin on calcule le PSNR entre l'image d'origine et l'image débruité. On fait ce traitement pour chacune des images :

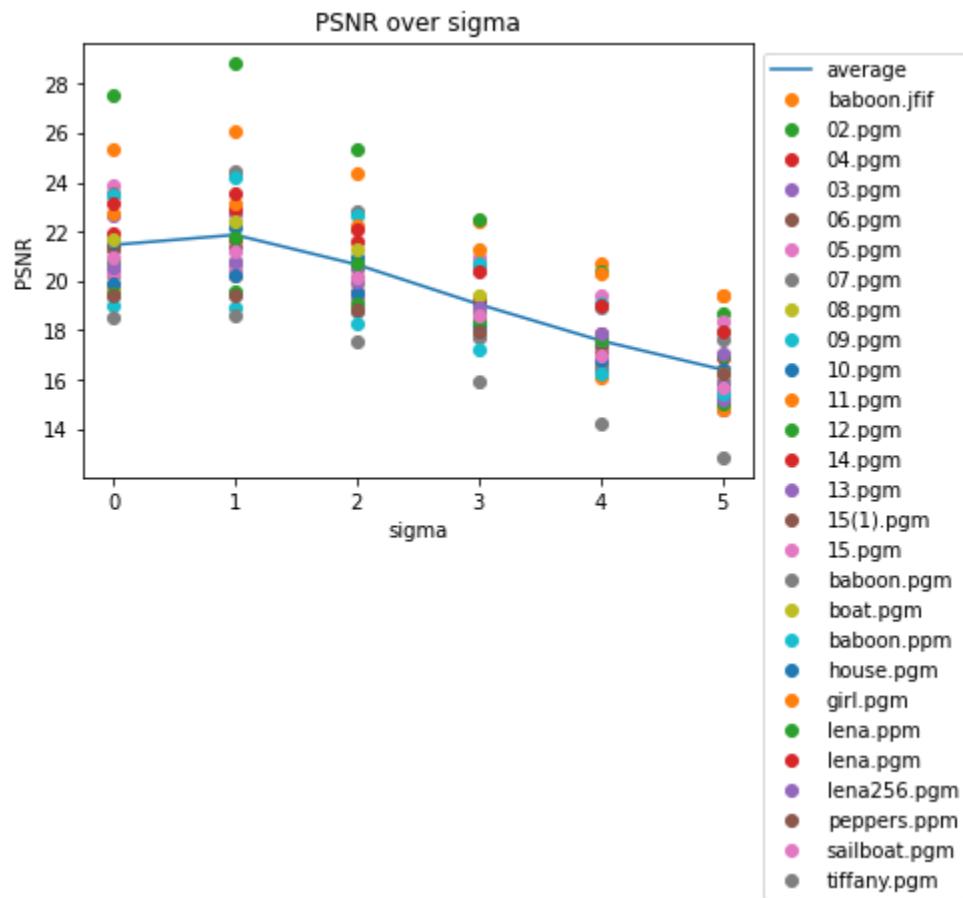


Graphique de l'évolution de la qualité en fonction de l'intensité du bruit.

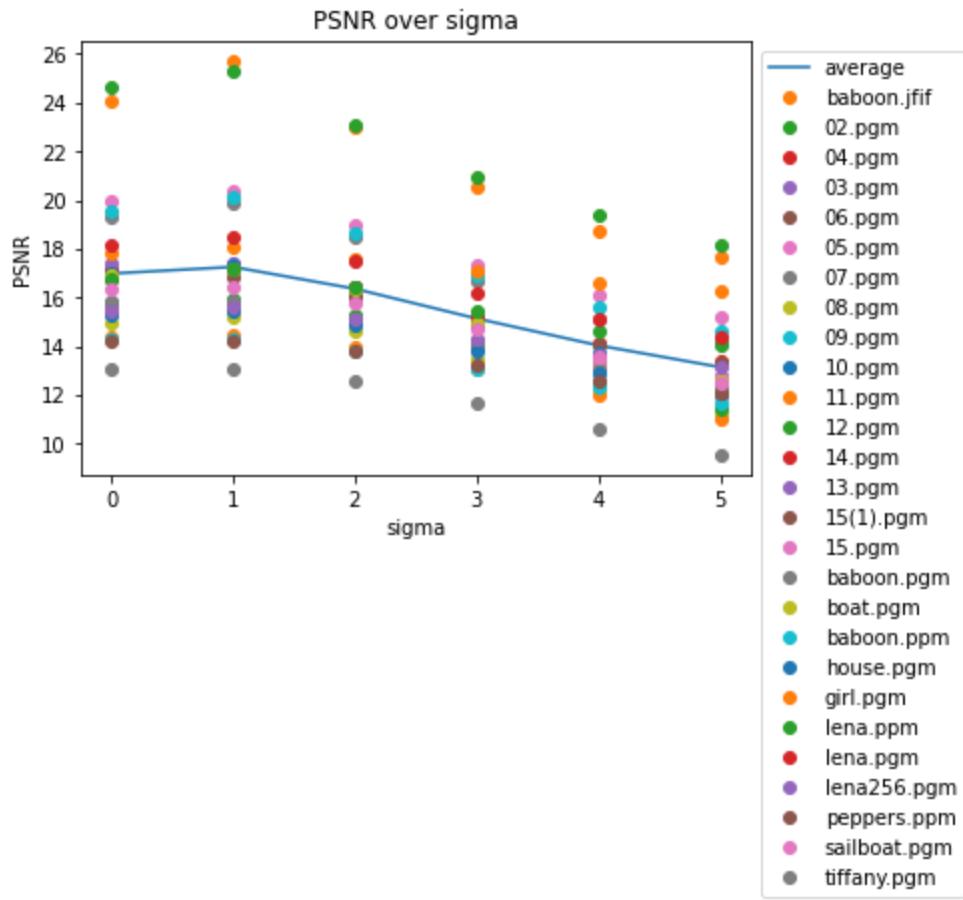
Le PSNR est exprimé en dB et l'intensité correspond à une multiplication du bruit gaussien, exprimé en dizaine de pourcent de la valeur maximale (sigma de 1 vaut $0.1 * 255 = 25.5$ et

sigma un de 5 vaut $0.5 * 255 = 127.5$.

On effectue le même traitement sur le CNN utilisant des blocs en couleurs.



Et sur le CNN utilisant des blocs en couleur de grande taille (64x64 à la place de 16x16).

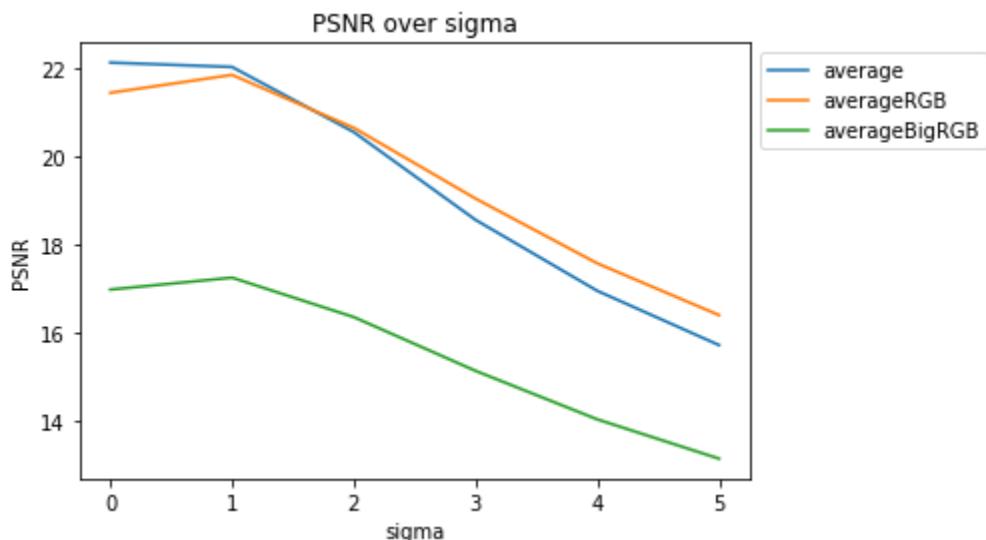


On peut ainsi comparer la qualité des différents CNN :

En bleu : le CNN utilisant des blocs (16x16) en nuance de gris

En orange : le CNN utilisant des blocs (16x16) en couleur

En vert : le CNN utilisant des blocs de grande taille (64x64) en couleur.



On remarque que la troisième méthode est celle qui produit la moins bonne qualité. De plus, la première méthode permet d'avoir de meilleurs résultats pour un petit sigma, mais la

deuxième donne de meilleurs résultats pour un grand sigma.

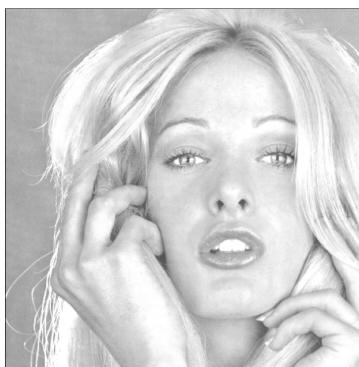
On peut également faire un comparatif visuel. En effet, certaines images sont, pour les trois méthodes, de mauvaise qualité. Ce sont donc des cas difficiles. Par exemple, l'image "tiffany.pgm" possède toujours un faible PSNR :

À gauche : l'image originale, au milieu : l'image bruitée avec un sigma de 0.1, à droite, l'image débruitée.

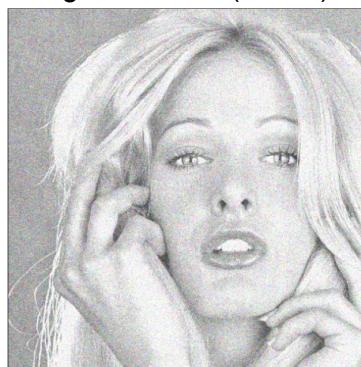
En haut: le CNN utilisant des blocs (16x16) en nuance de gris

Au milieu: le CNN utilisant des blocs (16x16) en couleur

En bas: le CNN utilisant des blocs de grande taille (64x64) en couleur.



tiffany.pgm



tiffany.pgm/noised



tiffany.pgm/denoised



tiffany.pgm



tiffany.pgm/noised



tiffany.pgm/denoised



tiffany.pgm

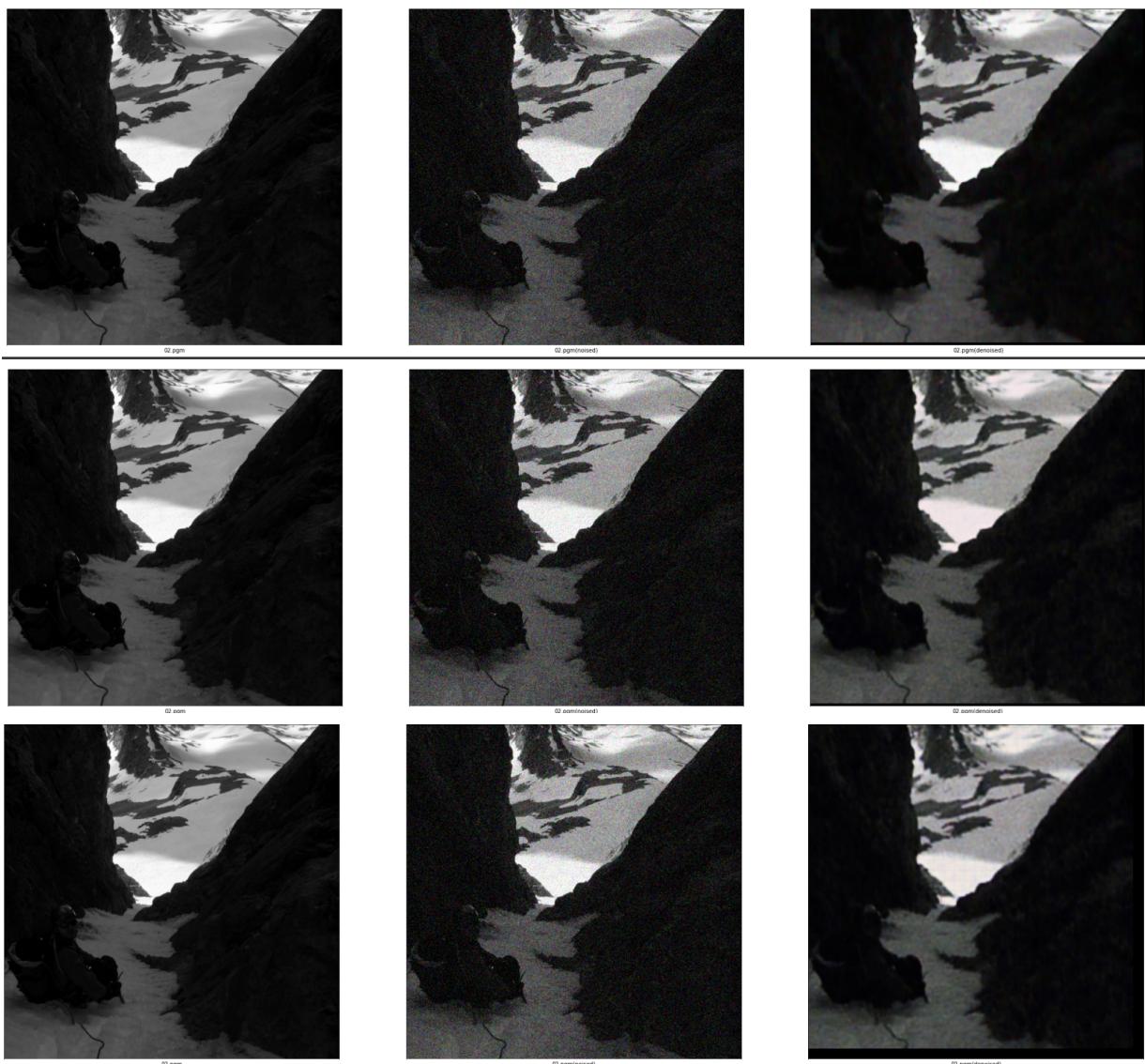


tiffany.pgm/noised



tiffany.pgm/denoised

Et on peut faire un comparatif d'un cas facile : "02.pgm" :

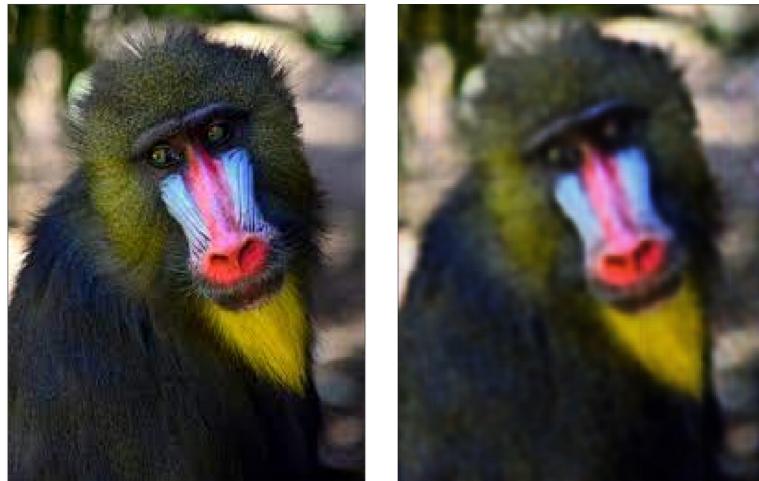


Entretien avec monsieur Puech :

Lors de notre entretien avec monsieur Puech, nous avons pu échanger autour des différents résultats de débruitage que nous avons obtenus, et notamment d'une possible stratégie de mise en place d'une technique de débruitage en particulier, que nous allons présenter ci-dessous :

- Dans un premier temps, nous avons remarqué que les auto-encodeurs simples (pas variationnels), sont une première bonne approche pour le débruitage d'images bruitées. Ils permettent, par des étapes de convolutions, de générer des images moins bruitées, mais pas tout à fait proche de la vraie image d'origine. En effet, on a pu constater dans les résultats précédents que l'on arrive à globalement reconstruire l'image d'origine (présence des bonnes couleurs, du sujet), mais que l'on obtient un résultat flou ; on a du mal, au moment de la reconstruction, à préserver des

informations de texture notamment.

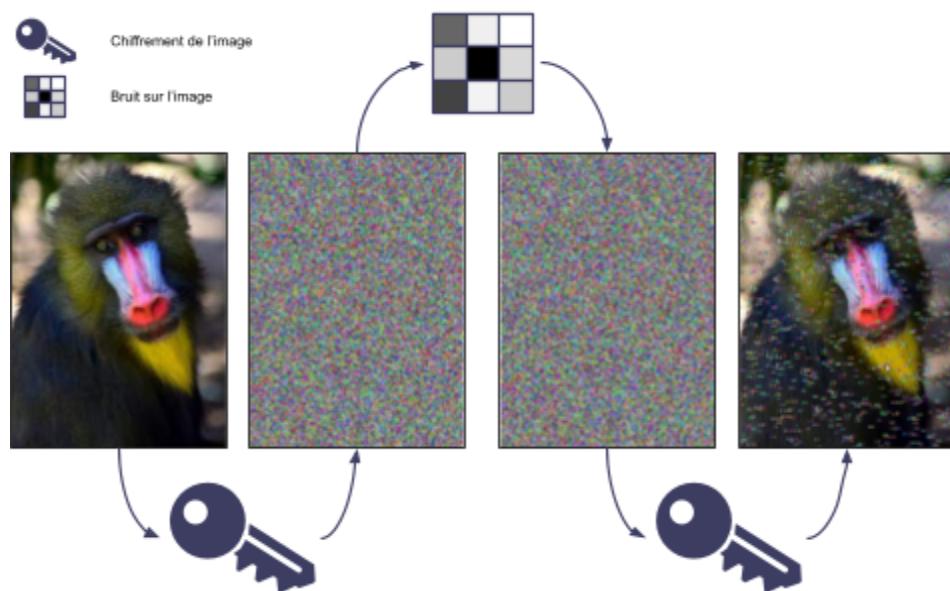


Résultat pour des blocs de taille 64x64, avec un padding de 16.

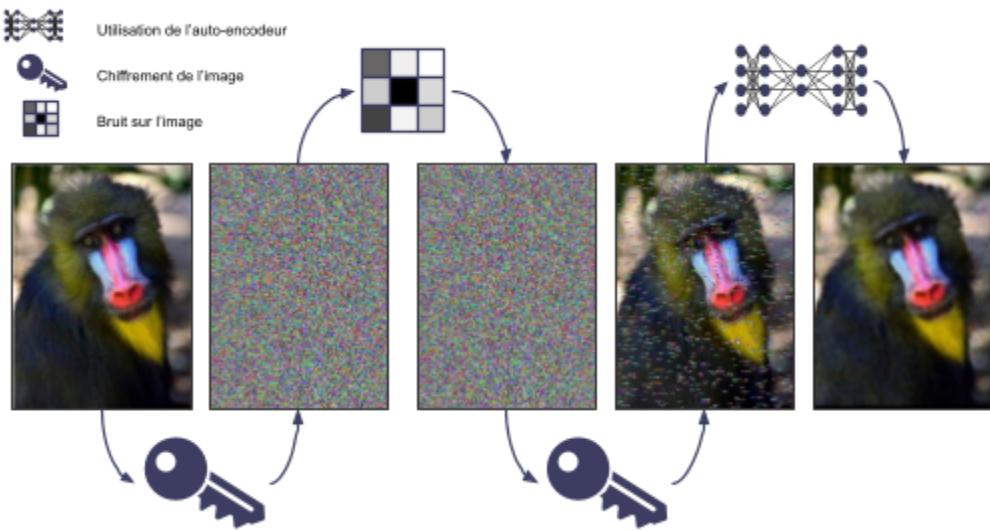
On remarque que certains détails importants sur cette image, comme l'intérieur des yeux, sont complètement effacés.

Nous allons donc nous baser sur ces premiers résultats afin de concevoir une stratégie permettant de reconstruire au mieux cette dernière information manquante, celle de la texture.

- Ainsi, si on voulait chercher à formaliser notre problème sous la forme d'un schéma, nous pourrions par exemple construire le dessin suivant :

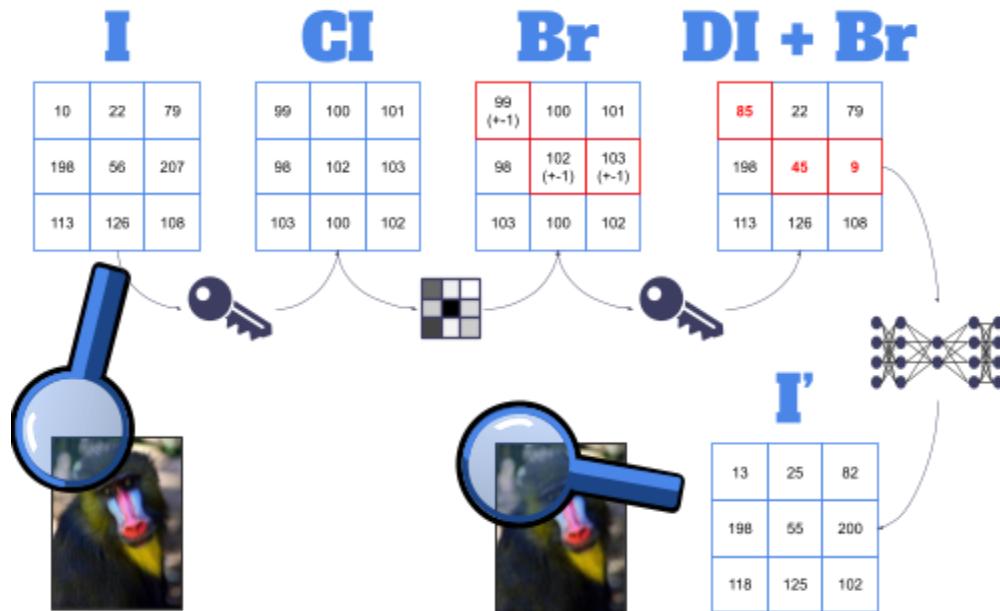


On se rend compte, sur la dernière image, qu'il y a beaucoup plus de bruits qu'au départ. C'est ce bruit que l'on va chercher à éliminer. On peut également chercher à schématiser l'utilisation d'un auto-encodeur comme suit :



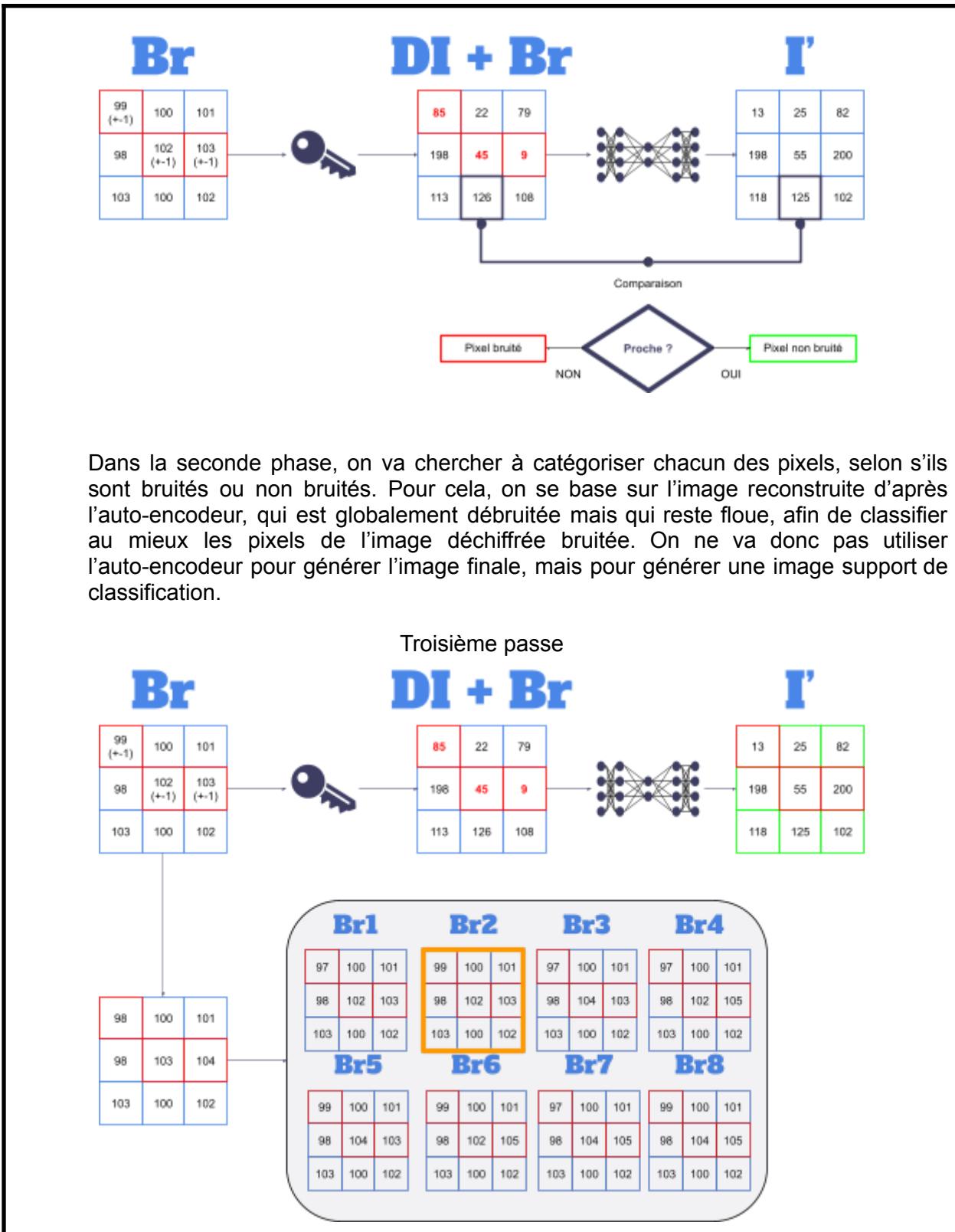
Donc, en utilisant les résultats de notre auto-encodeur, on peut par exemple mettre en place une approche itérative de résolution de ce bruit. Pour simplifier la modélisation de notre approche, nous allons nous baser sur des valeurs hypothétiques d'une petite portion de pixels, mais le principe reste identique pour la totalité de l'image.

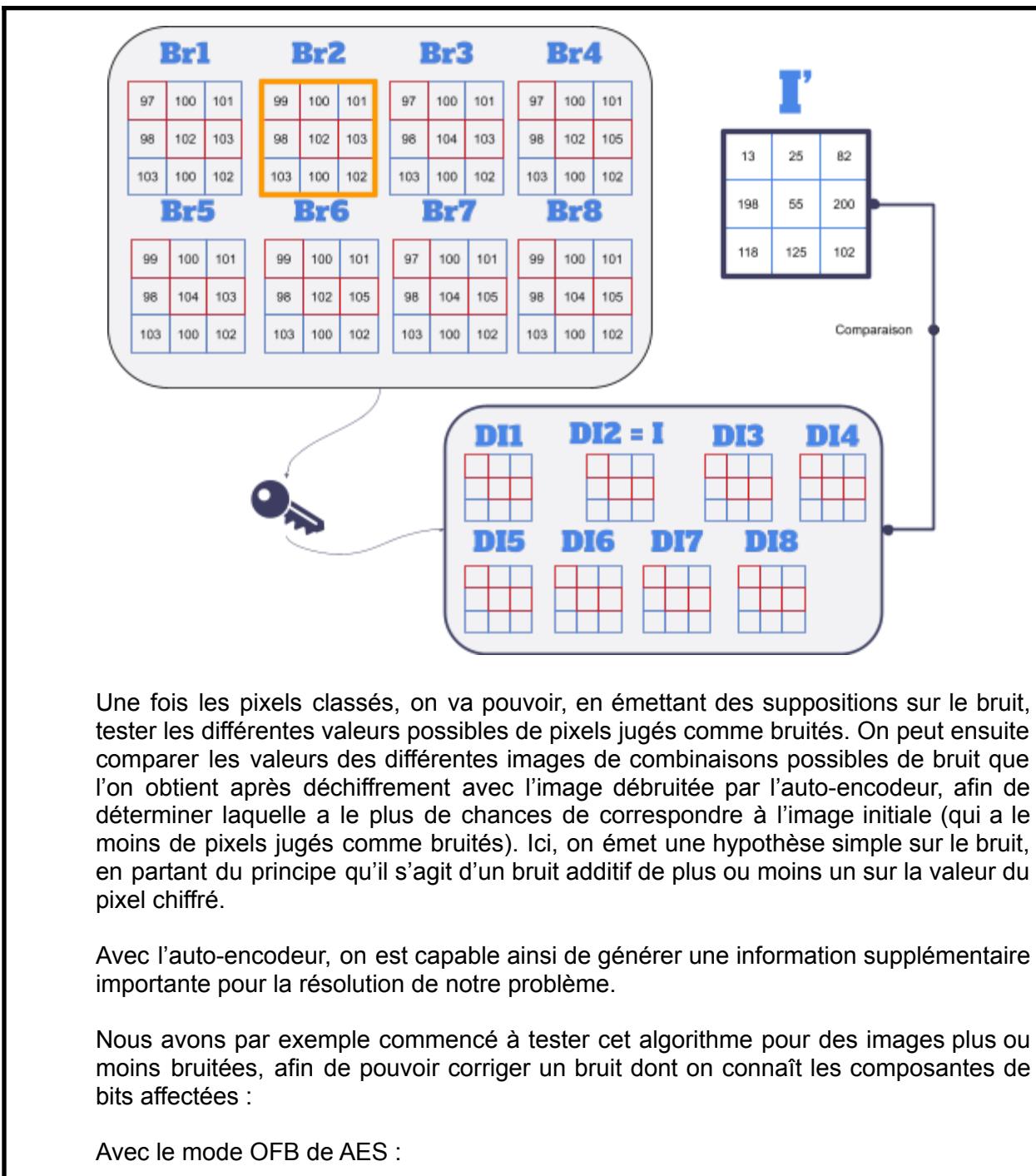
Première passe

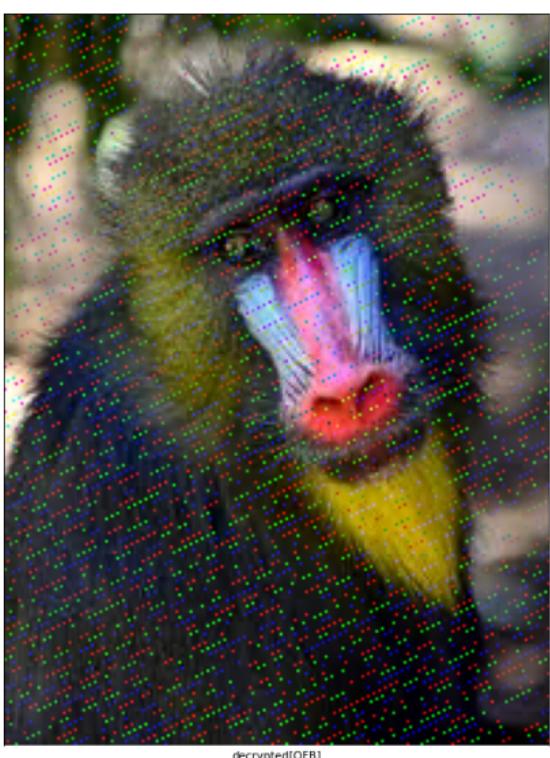
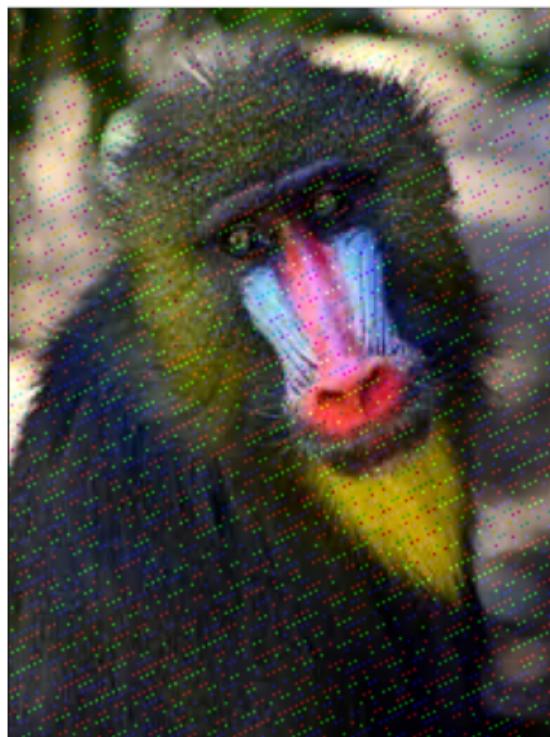


Dans un premier temps, on va donc intégrer l'auto-encodeur pour un débruitage un peu naïf de l'image. Comme vu plus haut, ce débruitage va nous permettre d'obtenir une image proche de l'image d'origine.

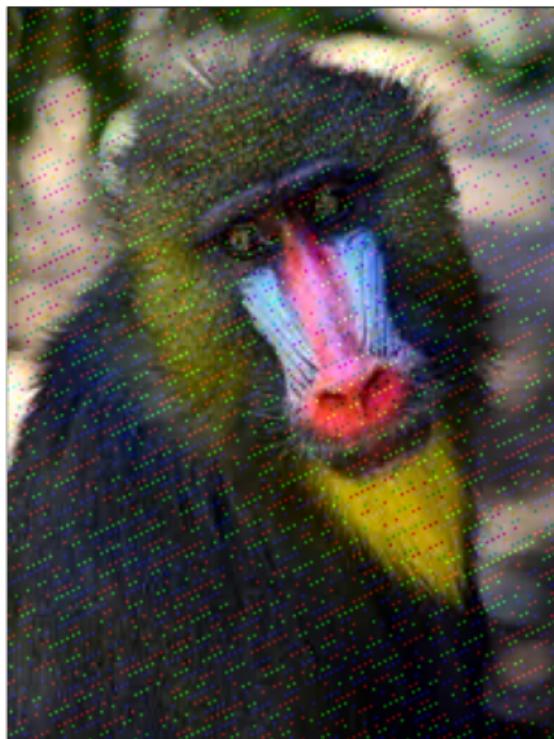
Seconde passe







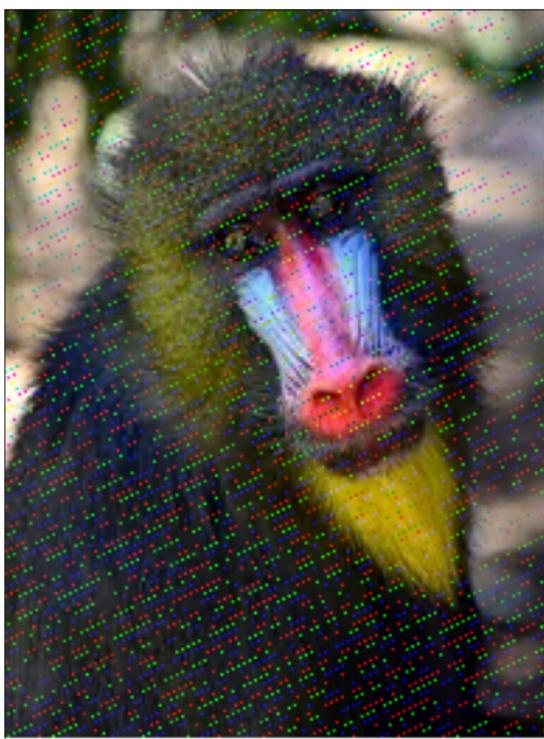
Avec le mode CTR de AES :



decrypted[CTR]



denoised[Autoencoder_selection]



decrypted[CTR]



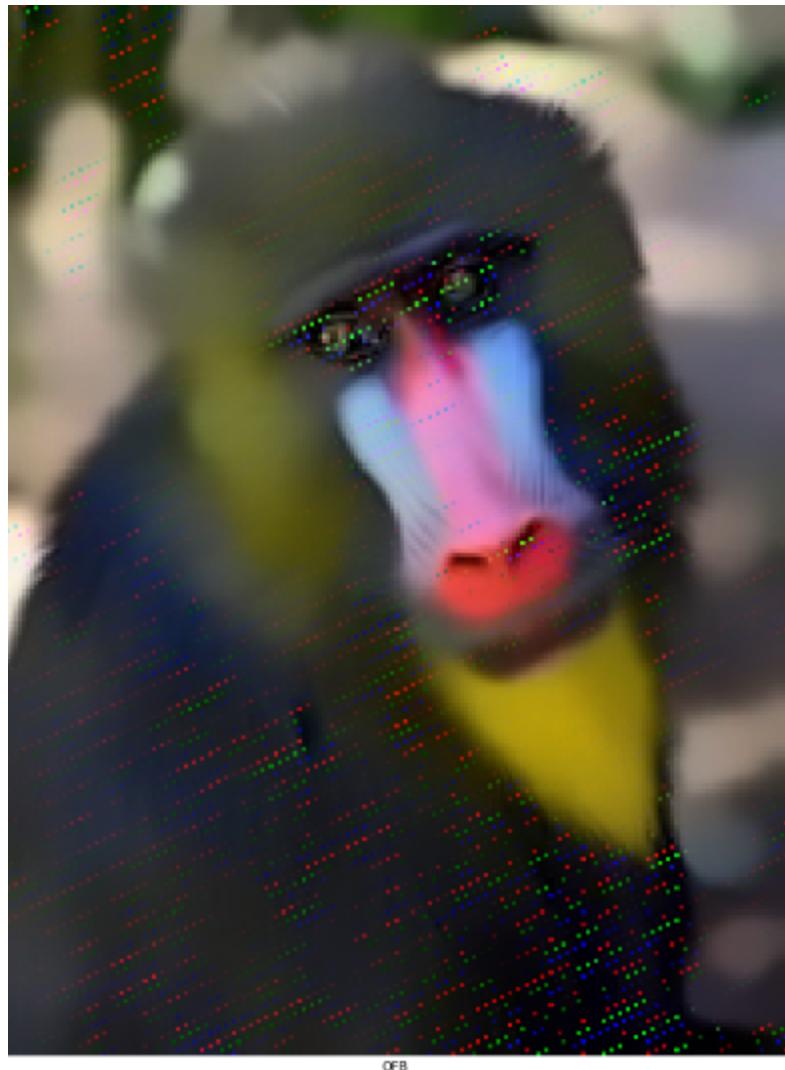
denoised[Autoencoder_selection]

On remarque que cet algorithme est prometteur dans son utilisation, puisque pour des

modes de chiffrement ne propageant pas le bruit à l'intérieur du bloc chiffré (image de gauche), une bonne partie du bruit reste enlevable. Cependant, pour des modes de chiffrement comme ECB ou CFB, il nous faut opérer une autre approche. Une manière de faire peut alors être de se servir du nombre de pixels jugés comme bruités au sein d'un même bloc, afin de déterminer si le bloc en question est originellement bruité, pour le cas échéant le corriger en testant les combinaisons de valeurs possibles par superposition sur le bruit au sein d'un bloc.

Nous avons également vu une méthode de débruitage d'OpenCV se basant sur une application de pattern dans l'image afin de débruiter plus facilement celle-ci (`fastNIMeansDenoisingColored()`). En effet, si on prend une image avec des motifs qui se répètent par exemple, il devient alors plus facile de débruiter une zone possédant un de ces motifs répétitifs mais bruité, dès lors que l'on sait à quoi ressemble ce motif en clair. Cette technique est également employée pour le débruitage de vidéos, où on peut se servir d'un même élément à des frames différentes pour pouvoir le débruiter dans la vidéo.





On remarque que pour des images fortement bruitées présentées plus haut (respectivement la deuxième image de gauche de CTR, et la deuxième image de gauche de OFB), ce type de débruitage avec l'image fournie, a du mal à obtenir de bons résultats. De plus, on ne se sert pas du tout des caractéristiques de l'algorithme de chiffrement utilisé pour nous aider à mieux débruiter.

Reste à faire

- Préparer le poster
- Préparer l'oral
- Préparer plus de scénarios de test de notre algorithme.