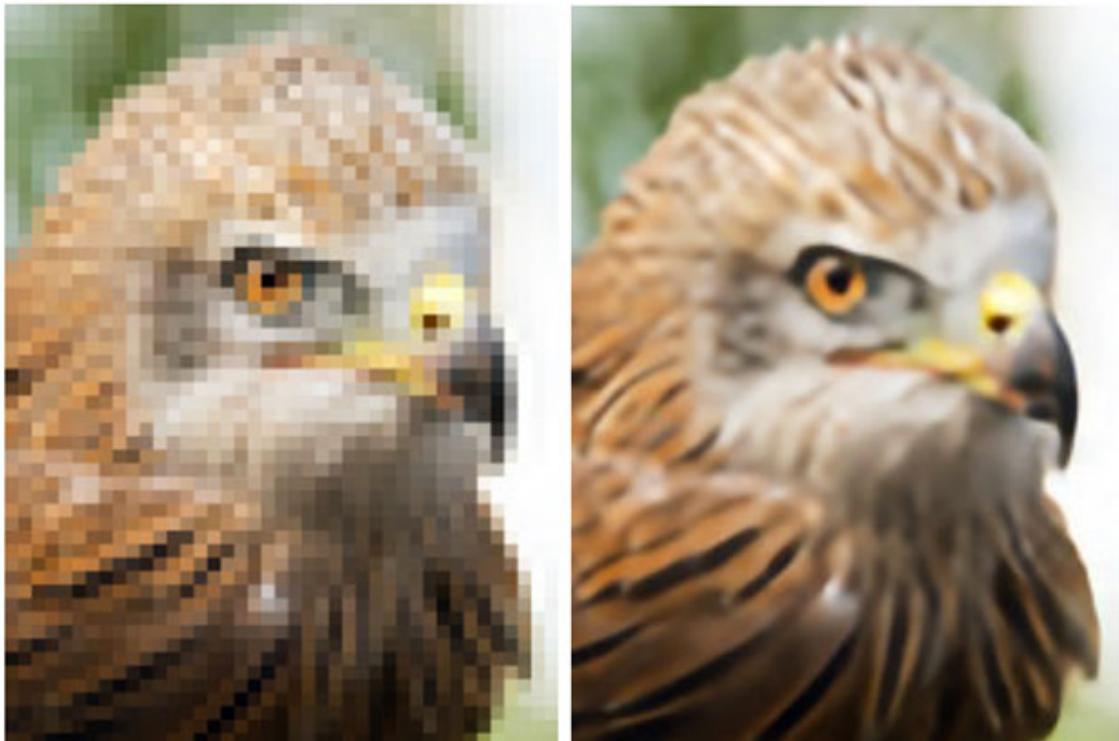


IA Python : Augmentation de la définition d'images



Sommaire

Préambule	2
Les Datasets	2
Créer un modèle et l'entraîner	3
Analyser les résultats	6
Premier essai	6
Deuxième essai	7
Troisième essai	7
Liens	9

Préambule

Le but de ce projet est d'apprendre le fonctionnement des IAs et d'en créer une en python. Au tout départ nous avions choisis un sujet plutôt large : "L'upscale d'image". En effet, il était question de créer une IA permettant d'augmenter la résolution (qualité) et la définition (taille pixel par pixel) d'une image à partir d'une image de mauvaise qualité. En effet, ce n'est pas la première IA du genre, il existe d'ailleurs déjà des IA publiques qui permettent de le faire. On peut d'ailleurs citer [Waifu2x](#) ou bien [Gigapixel AI](#). Ce procédé est utilisé dans de nombreux domaines, que ce soit sur les télévision haute résolution pour améliorer la qualité d'une source vidéo, ou bien dans nos téléphone pour améliorer la qualité de nos photos.

Au vu de l'ampleur du sujet et du temps limité du projet, nous avons dû limiter le sujet en se concentrant sur la création d'une IA permettant d'améliorer la résolution d'une image, sans en changer sa définition. Le programme utilise les modules : Numpy, Cv2, Tensorflow (en remplacement de sklearn) et Matplot.

Les Datasets

Pour commencer, nous avons cherché des datasets déjà existant contenant des images en haute résolution avec leur version en basse résolution. Étant donné la nature de l'IA, il est essentiel d'avoir un nombre conséquent d'images pour l'entraîner, c'est pour cela que nous avons utilisé plusieurs datasets.

Pour éviter d'avoir à traiter un trop grand nombre d'images et pour réduire au maximum le temps nécessaire à l'entraînement de l'IA, nous avons utilisé 2025 photos (Haute/Basse résolution/définition) provenant de 3 datasets différents. Pour ce genre d'IA, il est normalement conseillé d'utiliser au minimum 10 000 photos pour obtenir de bon résultats. Nous avons donc 2 datasets qui proviennent de Kaggle et le dernier est issu des concours NTIRE et PIRM, organisés depuis quelque années qui ont pour sujet la restauration et l'amélioration d'image (les liens des datasets sont disponibles à la fin du rapport).

Une fois les datasets réunis, il a fallu d'abord les unifier avec la même structure de dossier, pour pouvoir ensuite les charger plus facilement dans notre code. Ensuite nous les avons chargés puis mis à la même résolution; 256x256 pour réduire la taille et le temps d'entraînement de l'IA; puis appliquer un filtre permettant de dégrader la qualité des images basses résolution des images. En effet, les datasets avaient déjà des images en basse définition mais tous n'avait pas la même différence entre les images en haute et basse résolution.

La dégradation de l'image consiste en l'application de deux filtres, du flou (blur) et du bruit (Salt & Pepper), l'intensité est faible pour éviter d'induire en erreur l'IA.

Une fois traités, les datasets ont été enregistrés en deux fichier numpy : high_x265.npy et low_x256.npy pour permettre un chargement plus rapide.

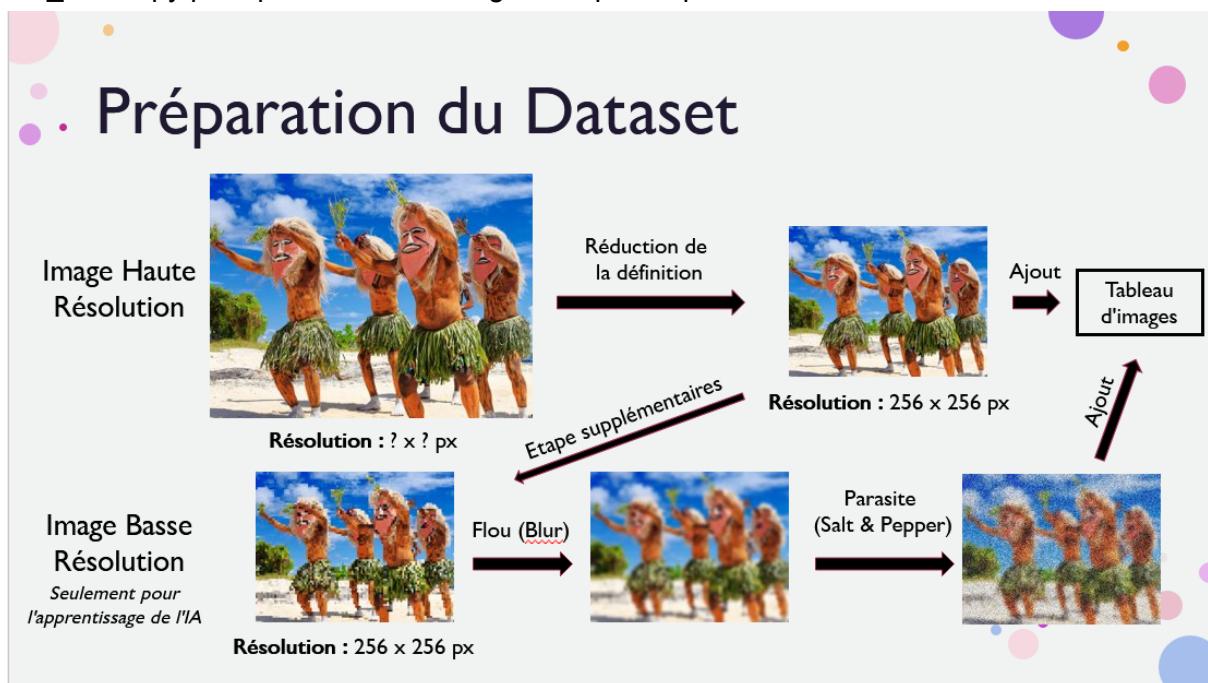


Schéma décrivant le processus de préparations des datasets

Créer un modèle et l'entraîner

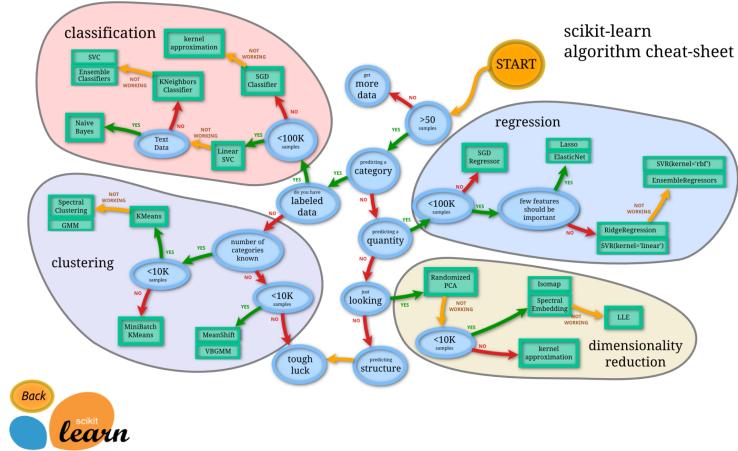
Choisir le bon algorithme n'est pas simple car il en existe énormément, il existe plusieurs grandes familles : Classification, Régression, Non supervisés,Réseau de Neurone...



Machine Learning algorithm classification. Interactive chart created by the [author](#).

Source : Edward Data Science

mentale présente sur la documentation de sklearn nous a permis de facilement nous orienter vers le bon modèle.



Source : [scikit-learn](#)

Un modèle Non Supervisé (Unsupervised), est un modèle qui va chercher à faire des corrélations entre des données par l'intermédiaire de fonction non linéaire.

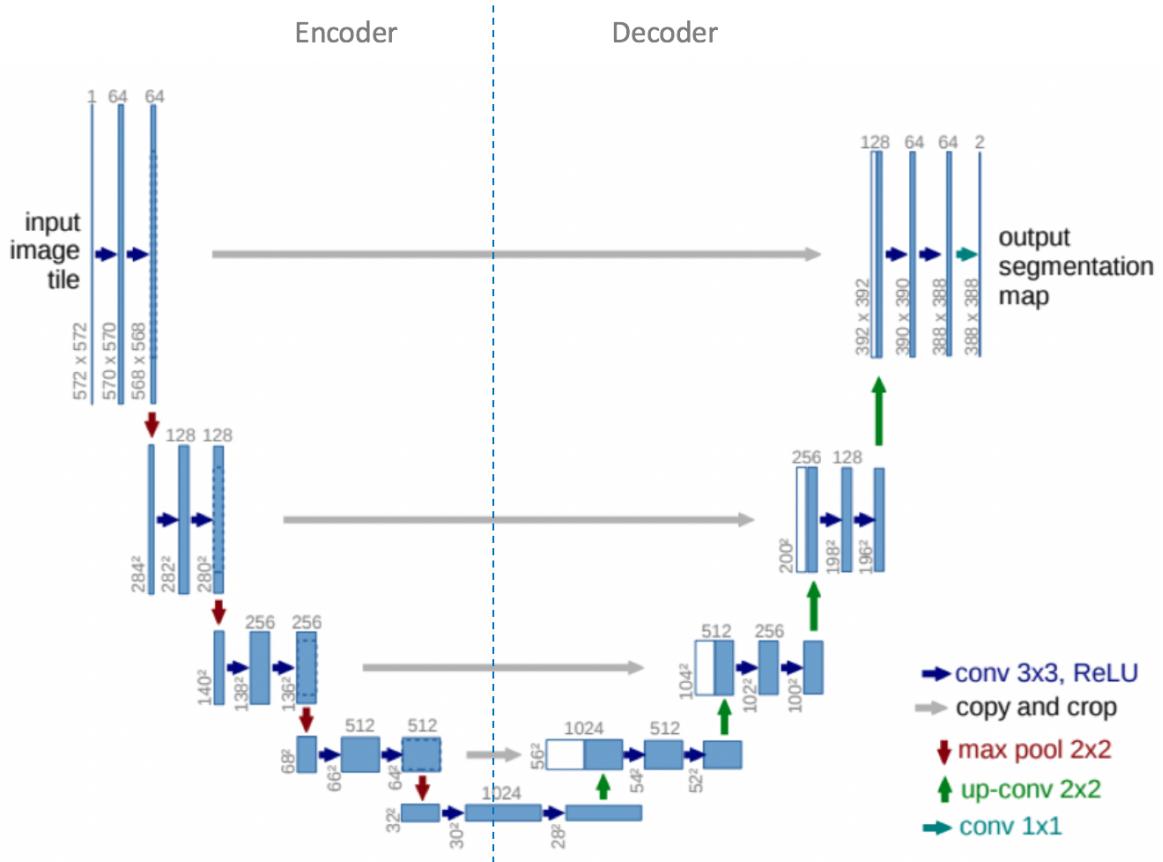
Le modèle Isomap est pareil c'est un algorithme de réduction de dimensionnalité non linéaire. Il consiste à créer un réseau de nombre fini de points (voisin) interconnecter entre eux, construit sous forme de graphe. Le but est de trouver le chemin le plus rapide entre chaque paire de points. De cette manière, il est possible de trouver une donnée proche de celle qui a été donnée.

Malheureusement après de nombreuses tentatives nous n'avons pas réussi à obtenir des résultats cohérent et utilisable (le code est toujours présent sur le notebook). En effet, on n'arrivait pas à obtenir un modèle fonctionnel avec Isomap, donc nous avons recherché un autre algorithme possible pour créer notre modèle.

Après de nombreuses recherches sur internet, notre deuxième choix de modèle s'est porté sur le modèle [U-Net](#) qui est de la famille des Neural Network (Réseau de neurones) et pour cela nous avons utilisé le module Tensorflow en remplacement de sklearn.

Le modèle est divisé en 2 parties différentes:

- L'encodeur qui a pour tâche de sous-échantillonner les images suite à l'application des blocs de convolution successifs, cela va avoir pour effet d'encoder l'image donnée à plusieurs niveaux différents et à la fin obtenir une image sémantique (simplifier/schématiser).
- Le décodeur lui, va être chargé de sur-échantillonner les images encodées et les concaténer à la suite des autres par des opérations de convolution. Le but est de projeter les caractéristiques spécifiques apprises par l'encodeur dans l'image pour avoir une image plus détaillée.



En tout, nous avons produit 3 versions de notre IA, la plupart des modifications ont d'ailleurs été apportées sur le dataset.

Analyser les résultats

Afin de mesurer les différences entre les images et ainsi mesurer les performances de l'IA nous avons utilisé une méthode appelée PSNR et qui est très utilisée pour contrôler la qualité des compresseurs d'images.

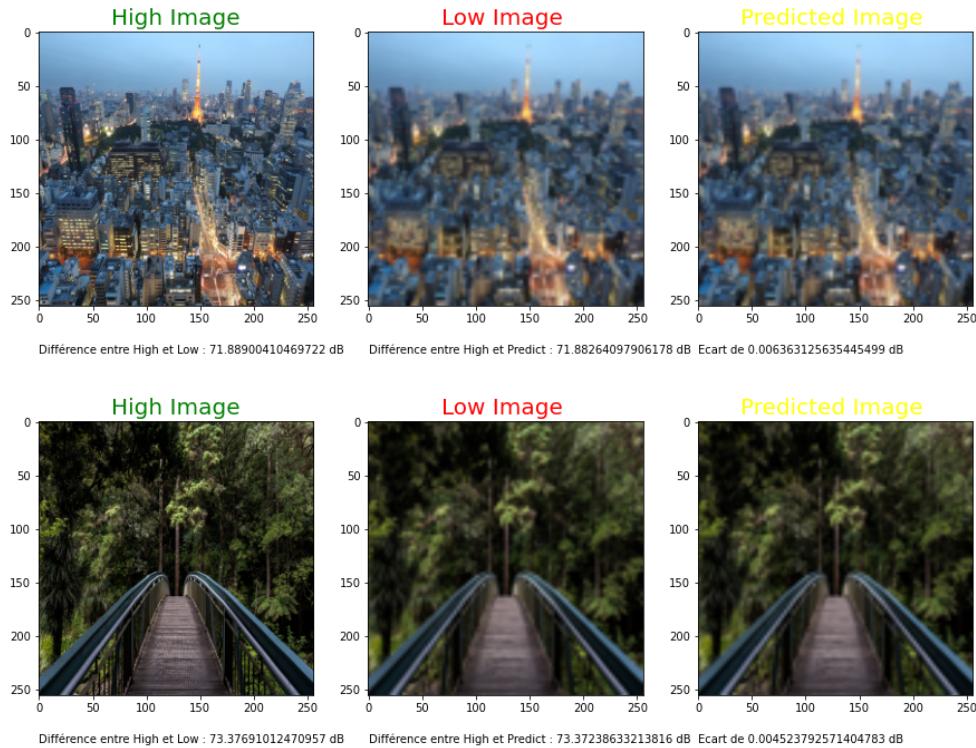
Le PSNR (Peak Signal to Noise Ratio) est une méthode d'évaluation et comparaison de modèles qui cherche à quel point une image (de base qualité si possible) change de l'image originale de haute qualité. Plus grande est la valeur PSNR, meilleure est la reconstruction de l'image. Cette méthode d'évaluation utilise une seconde valeur MSE (Mean Squared Error) qui calcule les erreurs moyens au carré

$$MSE = \frac{\sum_{M,N} [I_1(m,n) - I_2(m,n)]^2}{M * N}$$

$$PSNR = 10 \log_{10} \left(\frac{R^2}{MSE} \right)$$

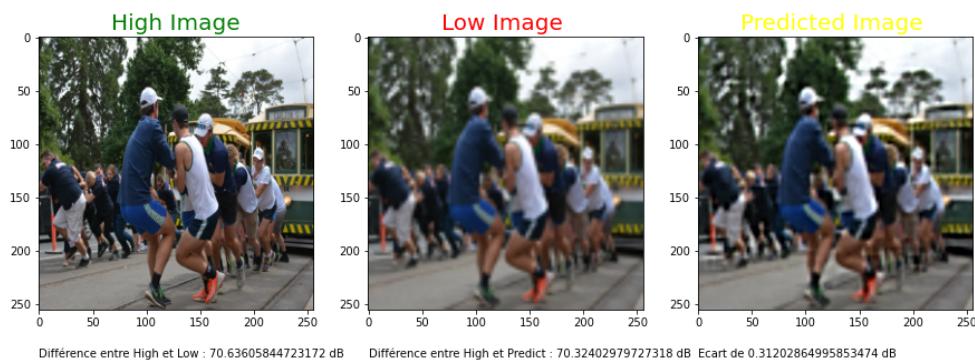
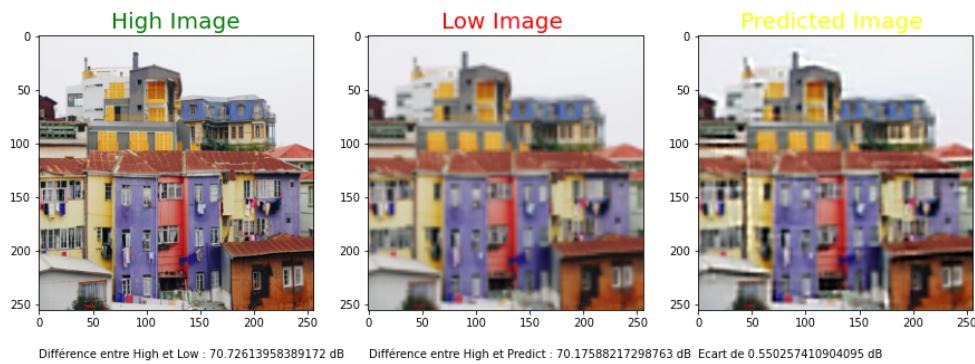
Premier essai

Lors du premier essai, l'IA n'a modifié quasiment aucune image. Perplexe, nous sommes vite arrivés à la conclusion qu'il s'agissait d'un problème lié aux images d'entraînement. Une grande partie des images ne possédaient que très peu de différences entre les images en haute et les images basse définition. En effet, l'écart entre le PSNR entre les images à haute et basse définition et le PSNR entre les images à haute définition et celles prédites sont quasiment nul.



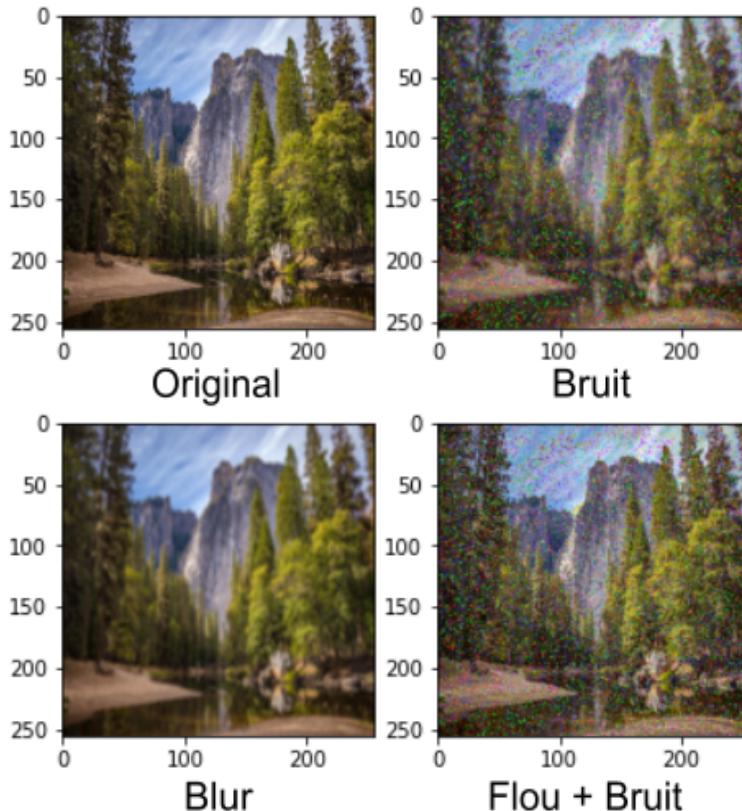
Deuxième essai

Pour résoudre ce problème, nous avons appliqué des filtres supplémentaires pour diminuer la qualité des images d'entraînement. Nous avons donc flouté les images de basses qualités pour simuler une baisse de qualité visuel. Ceci a eu un effet positif sur notre projet car bien que les améliorations étaient loin d'être parfaites, on a pu observer une nette amélioration.

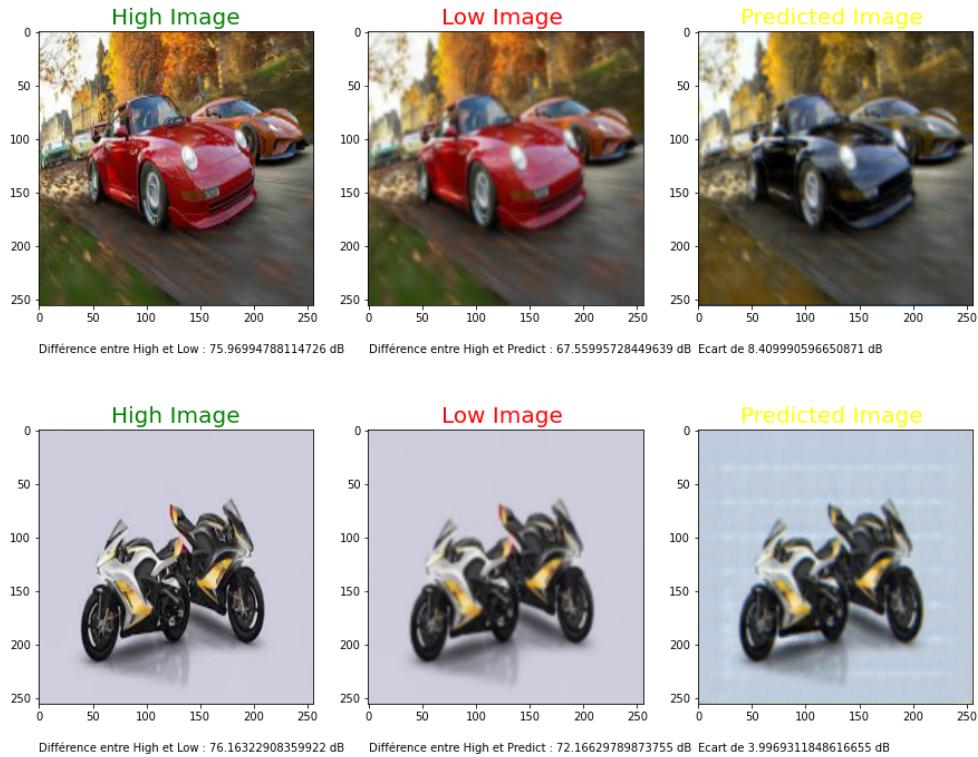


Troisième essai

Cette amélioration n'était pas suffisante donc nous avons créé une troisième version, le dataset d'entraînement de cette version est encore plus flou que celui de la deuxième version, et en plus nous avons ajouté un effet de bruit visuel.



On applique d'abord à l'image des parasites et ensuite on applique un flou
Malheureusement on remarque quelques problèmes :



Le troisième version de l'IA a tendance à changer trop les couleurs et on observe des artefacts en fond. On observe les mêmes effets mais plus atténusés que lors de nos tests, l'ia avait tendance à appliquer un filtre vert sur les images.

Les parasites, on s'en doute trop dénaturer l'image et l'IA a dévié de son but principal. On peut sans trop se tromper dire que la deuxième version est la plus réussie, si on se fie aux résultats. Le but initial était d'améliorer la qualité de l'image, au vu des résultats on voit que l'IA arrive à augmenter légèrement les détails.

Cependant, elle est perfectible sur quelques points, pour obtenir de meilleurs résultats, on peut considérablement augmenter le nombre d'images d'entraînement, on peut aussi mettre des images de plus grande définition et résolution, afin d'avoir le plus de détails possible sur les images haute et basse résolution. On peut aussi essayer d'utiliser des filtres moins destructeurs pour réduire la qualité.

Liens

Dépôt Gitlab :

[Dépôt Gitlab de l'IA](#) :

https://gitlab.iut-clermont.uca.fr/pomezquita/imageupscale_ia

[Dépôt Gitlab des datasets](#) :

https://gitlab.iut-clermont.uca.fr/lublouin1/imageupscale_ia_dataset

Source des Datasets :

[DIV2K](#) : <https://data.vision.ee.ethz.ch/cvl/DIV2K/>

[Kaggle image-super-resolution](#) :

<https://www.kaggle.com/adityachandrasekhar/image-super-resolution>

[Kaggle super-image resolution](#) :

<https://www.kaggle.com/akhileshdkapse/super-image-resolution>