Verrier Thomas
Bordes Alexandre

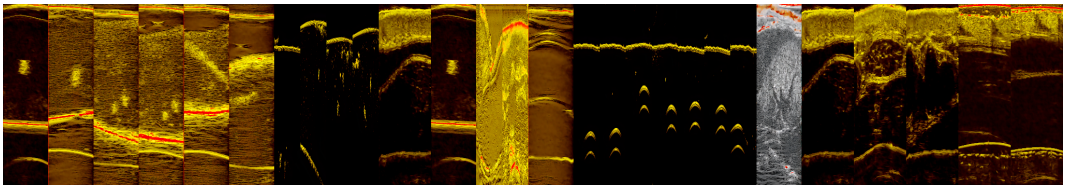
Imagerie Computationnelle: Réduction du bruit speckle des images ultrasonores par des techniques d'apprentissage profond

Traitement des données d'entrée :

Les données de test qu'on nous a fournies ont une taille qui n'est pas adaptée à l'entrée des réseaux de neurones. En ce qui concerne le format de l'image, nous avons donc dû rendre les images carrées ce qui évite que le modèle neuronal la déforme lors du pré-traitement des données.

Pour ce faire nous avons abordé deux techniques :

- Concaténer toutes les images en une longue "fresque" puis la découper en sous image de 1024px/1024px



Fresque de toute les images de test

- Concaténer la même images jusqu'à que sa taille soit de 1024px/1024px

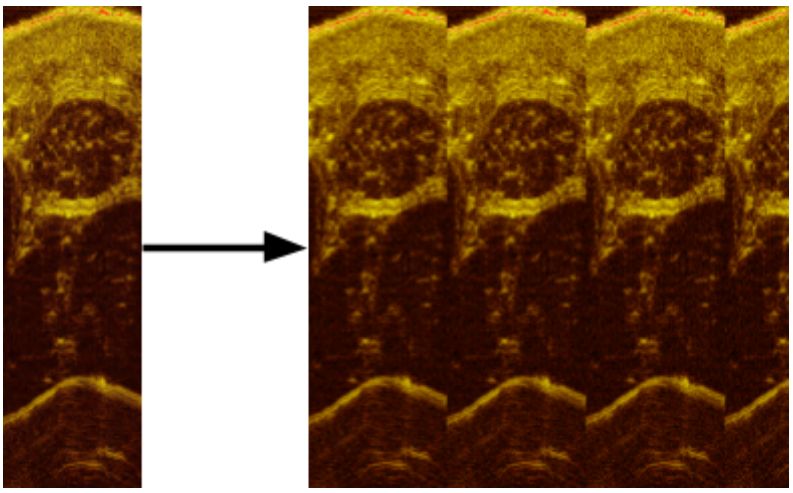


Image de test transformée en image carré

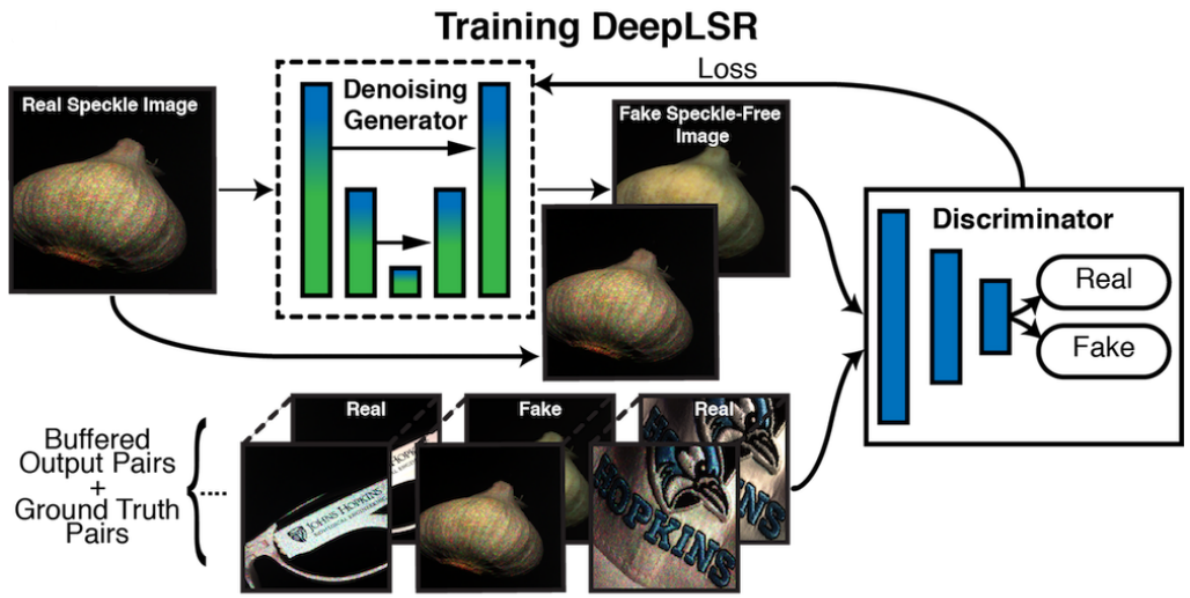
Finalement nous avons estimé que la deuxième technique était la plus cohérente car elle permet de garder le même contexte ainsi qu'une continuité dans l'image traitée par le réseau.

De plus nous avons testé des images trouvées nous même sur lesquelles on a ajouté du speckle noise. Cela nous permettra ensuite de faire des calculs de métrique type MSE et PSNR. Le dataset viens d'ici : ultrasound images

Test des réseaux de neurones:

Deep Laser Speckle Reduction (DLSR) :
papier

Architecture du réseau :
Le réseau est DLSR est un générative adversarial network **(GAN)**, il est basé sur un Générateur qui va créer une image et un Discriminateur qui va juger l'image créée par le Générateur. Ainsi le but du générateur est de tromper le discriminateur pour qu'il pense que l'image créée est une vraie image.



Architecture du modèle DeepLSR
source : https://arxiv.org/pdf/1810.10039.pdf

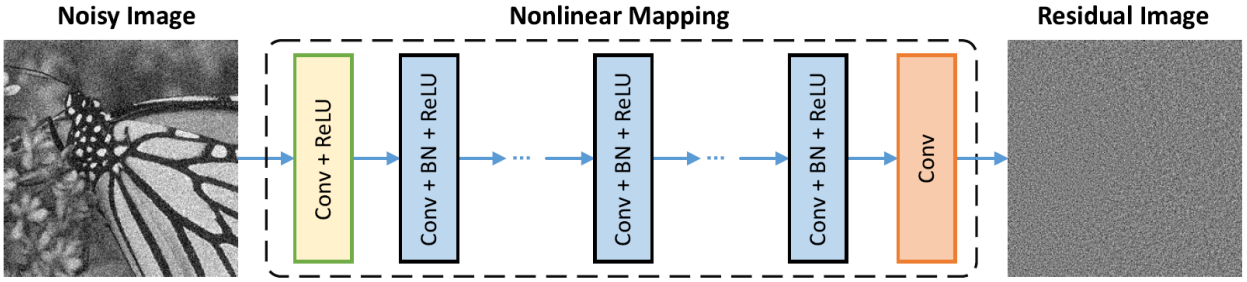
Entrainement du réseau :
Pour entraîner le réseau nous avons dû faire du traitement de données. Le réseau prend en entrée des images de taille 1024px/2048px qui correspondent à deux images concaténées. La première est l'image bruitée et la seconde est l'image débruitée. Pour ce faire, on a pris un dataset ultrasound images (800 images) puis on a appliqué un speckle noise sur la totalité,on a ensuite concaténé l'image bruitée et débruitée. Une fois le jeu de données obtenu nous avons entraîné le modèle neuronal ***.pynb/DLSR.ipynb*** sur 400 époques.

Test du réseau neuronal:
Pour tester le réseau neuronal, il faut un jeu de données et les poids du modèle. On place les poids dans le fichier checkpoints. Et on exécute le fichier test.py via ***.pynb/DLSR.ipynb***

Denoising Convolutional Neural Networks (DnCNNs)
papier

Architecture du réseau :

L'architecture est une suite de couches de convolution/bach normalisation, ainsi le but est d'extraire le bruit. points



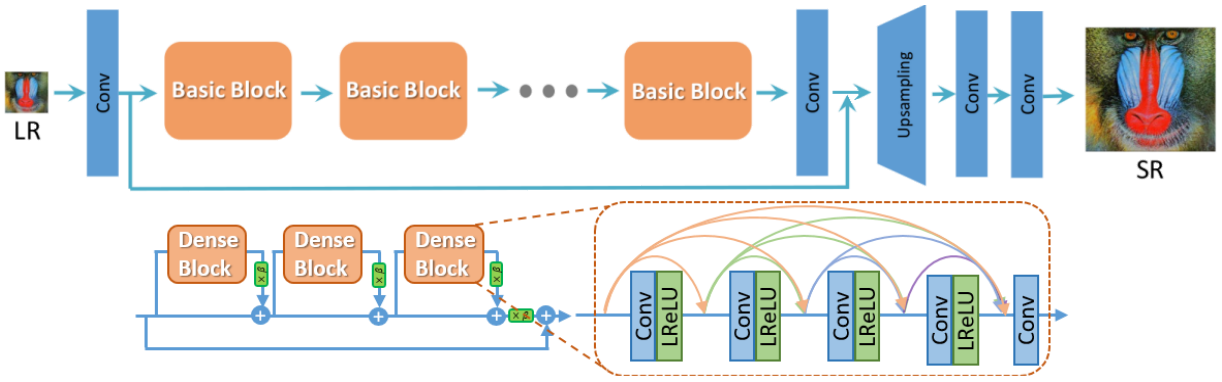
Architecture du modèle DnCNN
Source : https://github.com/cszn/KAIR#network-architectures

Test du réseau neuronal:
Pour tester le réseau neuronal, il faut un jeu de données et les poids du modèle. On place les poids, fournis par les auteurs, dans le fichier checkpoints. On exécute le fichier *main_test_dncnn.py* en spécifiant le modèle et le fait que du bruit est déjà présent.

BSRGAN
papier

Architecture du réseau :

Le modèle BSRGAN est une adaptation du modèle ESRGAN. Son fonctionnement est basé sur un GAN. Le générateur crée des Super Résolution image et le discriminateur estime l'image est fausse où réelle. BSRGAN a surtout changé les données utilisées ainsi que la fonction d'optimisation et quelques hyperparamètre.

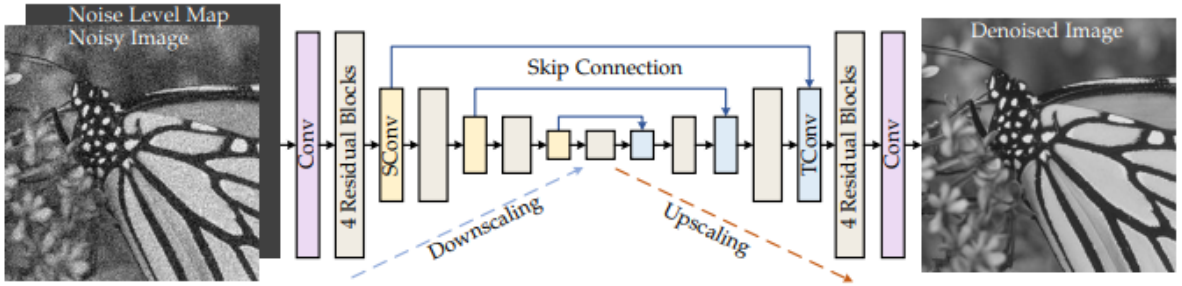


Architecture du modèle ESRGAN
source : https://arxiv.org/abs/1809.00219

Test du réseau :
La méthodologie de test est similaire au réseau précédent (extrait du même répertoire GitHub). Une fois les poids du modèle téléchargés, on lance l'exécution de notre modèle sur notre ensemble d'images.

Deep Plug-and-Play Image Restoration (DPIR) :
papier

L'architecture du modèle est un **encodeur/décodeur**, qui intègre un deep convolutional neural network (**DCNN)**. Il combine **resNet** (il ajoute en plus sur le resNet des skip connection) ainsi que **DenseNet.**



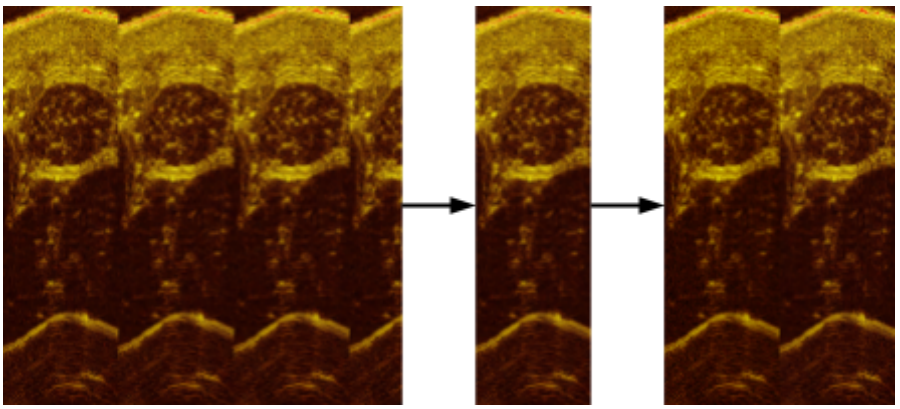
Architecture du réseau DPIR
source : https://arxiv.org/pdf/2008.13751.pdf

Test du réseau neuronal:
Pour tester le réseau neuronal, il faut un jeu de données et les poids du modèle. On place les poids, fournis par les auteurs, dans le fichier checkpoints. On exécute le fichier *main_test_dpir.py* en spécifiant le modèle et le fait que du bruit est déjà présent.

Traitement des données de sortie:

En sortie des réseaux nous obtenons des images carrés correspondant aux entrées débruitées.
Il faut réaliser l'opération inverse à la concaténation de la partie sur le pré-traitement pour récupérer l'image au format de base. Pour se faire, on se base sur le fichier data.json.

Une fois l'image récupérée, on réalise via le script `utils/un_split.py` la concaténation entre l'image d'origine bruitée est l'image nettoyée. Cela nous permet de comparer visuellement le traitement appliqué par le modèle.



Chaîne de traitement de l'image en sortie des modèles

A gauche l'image en sortie du modèle
Au centre l'image de base débruitée
A droite l'image de base bruitée et collé à l'image débruitée