



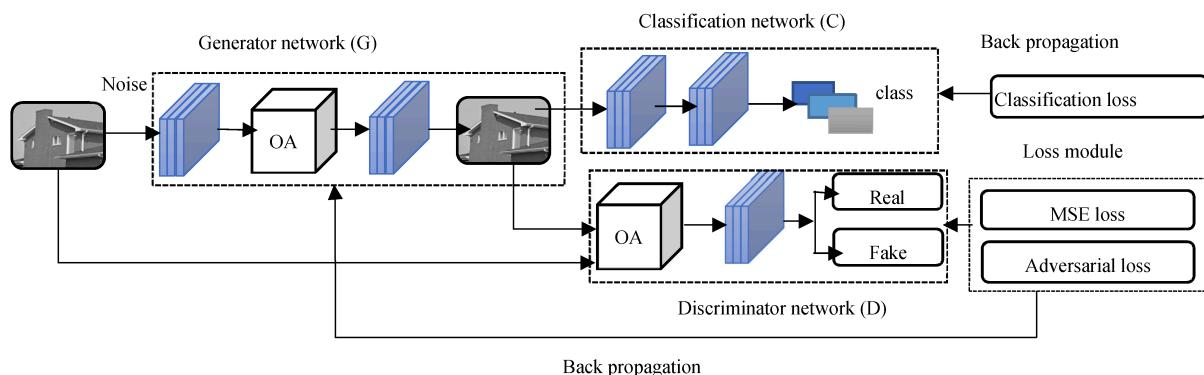
## Projet débruitage - Compte rendu semaine du 3/11

Viguier Killian

Wang Xihao

M2 IMAGINE  
Faculté des Sciences  
Université de Montpellier

9 Novembre 2025



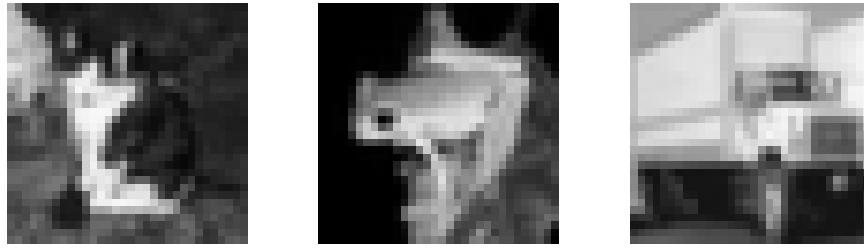
### Résumé

Dans ce premier compte rendu, nous verrons les tâches effectuées lors de la première semaine du projet image qui porte sur le débruitage d'images, puis nous nous pencherons sur les prochaines étapes prévues. Lien du GitHub : [https://github.com/VigKillian/denoiser\\_engine](https://github.com/VigKillian/denoiser_engine)

# 1 Base de données d'images

## 1.1 Base d'images CIFAR-10

Pour notre projet, nous avons décidé de prendre comme ensemble d'images CIFAR-10 qui est largement utilisé pour les algorithmes de deep learning. Cette collection comporte les classes : avions, voitures, oiseaux, chats, cerfs, chiens, grenouilles, chevaux, navires et camions, et nous avons sélectionné 100 images de taille 32x32 pour chacune des classes, revenant à 1000 images initiales.



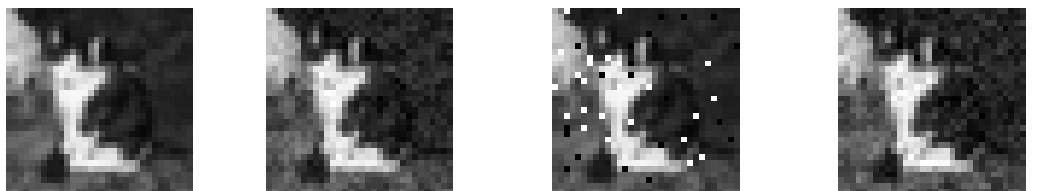
(a) Image de chat      (b) Image de grenouille      (c) Image de camion  
Fig. 1. – Exemples d'images de la collection d'images CIFAR-10

## 1.2 Bruitage d'images

Ensuite, nous avons implémenté 3 différents algorithmes de bruitage d'images pour les images précédemment récupérées :

- Un bruit aléatoire en chaque pixel (d'amplitude  $\pm 15$ ) suivant une loi uniforme.
- Un bruit gaussien défini par la fonction :  $p_G(z) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(z-\mu)^2}{2\sigma^2}\right]$
- Un bruit pepper&salt : chaque pixel a une chance (5%) de voir sa valeur changer à 0 ou 255.

Ces algorithmes de bruitages sont appliqués sur chacune des images de notre collection d'images initiales, ce qui nous donne au final 3000 images bruitées.



(a) Image initiale      (b) Bruit aléatoire      (c) Bruit Pepper&Salt      (d) Bruit gaussien  
Fig. 2. – Image de chat avec différents bruitages

## 2 Suite du projet

Nous nous baserons sur une base de code de débruitage d'images avec auto-encodeur (lien : <https://github.com/Research-and-Project/ConvDAE>) que nous allons modifier pour ajouter un GAN. Cette base de code utilise tensorflow, une bibliothèque de Python pour le Machine Learning.

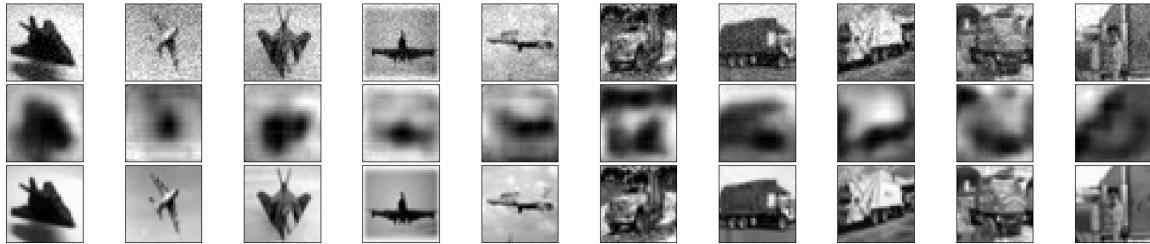


Fig. 3. – Exemple de résultat de la base de code : En haut l'image bruitée, en bas l'image initiale et au milieu l'image résultante du débruiteur.