Introduction

Le but de ce TP est de se familiariser et de tester des méthodes basées sur les réseaux de neurones pour la restauration d'images. Nous explorons deux problèmes fondamentaux en restauration, le débruitage et le zoom (super-résolution mono-image).

Pour des raisons pratiques de puissance de calcul, un TP ne permet pas d'apprendre des réseaux. Nous nous concentrons sur l'usage est l'étude fine du fonctionnement des réseaux.

Mise en place informatique

Dans la section suivante vous trouverez le lien des données du TP pour ceux qui n'ont pas de compte à Télécom. Comme le TP nécessite plusieurs fichiers qui s'incluent les uns les autres il faut avoir une installation fonctionnelle de python et un éditeur de script python (pas un notebook). Par exemple spyder.

Pour visualiser les images, le TP utilise GIMP car les outils de visualisation sous python sont très mauvais avec les images. Suivant l'endroit où vous avez installé gimp et suivant votre système d'exploitation (linux, macOS, windows) vous devez modifier la fonction viewimage dans le fichier scratch.py pour mettre le bon chemin d'accès.

Vous devez installer opencv et un tensorflow 1.xx. (plus des packages standards).

Vous n'avez qu'à ouvrir le fichier scratch.py pour executer (et adapter) des commandes. Le fichier models.py contient la construction des réseaux et vous permet de comprendre leur architecture. Il est particulièrement utile pour répondre aux questions sur l'architecture ou pour extraire des réseaux d'autres calculs que le résultat final.

N'oubliez pas de mettre à jour la variable datadir suivant l'endroit où vous avez téléchargé des données sur votre PC.

En résumé vous avez besoin de :

— Un python raisonnablement récent.

— Une version tensorflow 1.xx (normalement cela existe toujours).

— Un éditeur python interactif tel que Spyder .

— le logiciel GIMP de traitement d'images (et savoir trouver son chemin dans votre disque dur)

Comment rendre le TP

Idéalement vous aurez fini le TP dans les trois heures et vous me rendez le TP sous forme papier (dans le monde avant covid...). Sinon, vous avez deux semaines pour rendre votre TP sur le répertoire suivant :

`RENDU` `https://bit.ly/3iI5rSg`

Le fichier doit être au format pdf et le nom du fichier doit commencer par votre nom de famille. Comme vous ne pouvez pas effacer, pour mettre une seconde version ajouter _V2 dans le nom pour rendre une autre version.

Pour ceux qui n'ont pas de compte Télécom les données sont là :

`DONNEES POUR CEUX SANS COMPTE  https://bit.ly/3aijwlJ`

1 DCT_denoiser

Dans cette première partie on étudie une méthode classique de débruitage. Elle se base sur un modèle probabiliste en rapport avec la parcimonie (sparsity). On considère d'abord une variable aléatoire réelle X dont la loi serait proportionnelle à $e^{-|X|}$ et une observation $Y = X + B$ où B est gaussienne de variance σ.

1. Montrer que si l'observation est connue, $(Y = y)$ la valeur de $X = x$ la plus probable est celle qui minimise
$$\underset{x}{argmin} \left(|x| + \frac{1}{2\sigma^2}(x-y)^2 \right)$$
et donner la formule donnant x en fonction de y (on pourra se contenter d'étudier le cas $y > 0$. Tracer cette fonction.
Ce modèle en dimension 1 se généralise en remplaçant $|X|$ par $\|X\|_1$ mais n'est pas réaliste avec les images **sauf** si on considère X comme la représentation de l'image dans une base orthonormée bien choisie (pas la base des pixels). Ici le choix s'est porté sur la base de la DCT.
2. En pensant à une décomposition sur une base orthonormée, décrire ce que le fait de réseau défini dans DCT_denoise et en quoi il diffère (légèrement) du résultat trouvé à la question précédente.
3. Est-ce que le seuil contenu dans le modèle pourrait être appris par une décente de gradient ? Expliquer comment on pourrait procéder pour l'apprendre sur une grande base de données ?
4. Donner le nombre approximatif d'opérations par pixel de ce modèle de débruitage (en fonction du paramètre N) ?

2 DnCNN

On aborde maintenant un débruiteur construit comme un réseau profond. Pour ce TP nous avons pris un seul réseau DnCNN qui a été entraîné avec un bruit $\sigma = 25$.

5. Décrire rapidement l'architecture de DnCNN. Combien de paramètres utilise-t-il (en ordre de grandeur) ?
6. Combien d'opération par pixel nécessite-t-il par pixel (en ordre de grandeur) ?
7. Comparer qualitativement DnCNN et le DCT_denoiser.

8. Les différentes sorties associées ont-elles des statistiques qui respectent les moyennes et écart-types "prescrits" dans les poids ? (utiliser la liste de variables H de l'objet débruiteur) Décrire brièvement l'algorithme que vous avez utilisé (ne faire qu'une image pour calculer les statistiques) ?
9. Pour un bruit de $\sigma = 5$ DnCNN reste-t-il plus efficace que le débruitage DCT (qui a l'avantage de pouvoir régler le niveau de bruit très facilement) ?

3 DRCN

10. Expliquer brièvement l'architecture de ce réseau de super-résolution.
11. Combien d'opérations sont nécessaires pour l'exécution ?
12. Comment expliquer la différence de résultat suivant la méthode de dézoom ?
13. Observer les différentes images intermédiaires produites par DRCN. Commentaires.