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

Le TP utilise python (anaconda3) et le module de réseaux de neurones **tensorflow**. Il est développé par Google et a l'avantage d'être grandement optimisé pour beaucoup d'architectures. Son autre avantage est que l'utilisation ou non de la GPU est transparente et ne demande pas de changer le programme pour passer d'une architecture à l'autre.

Pour gagner du temps et de l'espace disque, vous allez utiliser mon installation de **tensorflow** [1]. Pour cela, si vous possédez déjà un répertoire ".conda" il faut changer son nom et créer un lien symbolique vers mon ".conda". Exécutez les commandes suivantes dans une nouvelle fenêtre de terminal (la première seulement si vous avez déjà un ".conda" [2]) :

```
mv .conda .conda_bak  (voir avertissement)
ln -s ~ladjal/.conda .conda
mkdir TP1_DELIRES
cd TP1_DELIRES
cp  ~ladjal/TP1_DELIRES/*.py .
ln -s ~ladjal/TP1_DELIRES/data TP1_DELIRES/data
```

Ce qui précède ne doit être fait qu'une seule fois. Appelez l'encadrant au moindre problème d'ordre informatique.

Ensuite, pour lancer l'environment de travail depuis un terminal

```
export PATH=/cal/softs/anaconda/anaconda3/bin:$PATH
cd TP1_DELIRES/
source activate tf_env
spyder
```

Ce TP est centré sur l'analyse de programmes déjà écrits. Vous travaillerez sous spyder essentiellement en sélectionnant une zone de programme et par clique-droit -> exécuter la cellule. Cela permet une souplesse d'utilisation et de comprendre le code à mesure que l'on progresse dans le TP.

Nous supposons une certaine familiarité avec python/tensorflow, mais cela n'est pas l'objet du TP. Si vous ne comprenez pas une construction python demandez des éclaircissements.

1. Les instructions sont valables si vous êtes sur des machines de la DSI de Télécom Paristech
2. Dans ce cas pour revenir à votre installation originale il faut faire : cd ; rm .conda ; mv .conda_bak .conda

Comment rendre le TP

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

`RENDU https://bit.ly/2MIRNOH`

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/2CTwnKm`

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 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 come un réseau profondprofond. 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 gradeur) ?
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 diff'rentes sorties associés 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 et DCSCN

10. Expliquer brièvement les architectures des deux réseaux de super-résolution.
11. Combien d'opérations sont nécessaires pour chacun d'eux ?
12. Comment expliquer la différence de résultat suivant la méthode de dézoom ?
13. Observer les différentes images intermédiaires produites par DRCN. Commentaires.

4 Un entraînement

Dans le fichier DCSCN_entrainement vous trouverez un réseau très proche du DCSCN déjà vu.

14. Expliquer les principales différences entre ce réseau et le réseau DCSCN déjà vu.
15. **Les questions suivantes sont optionnelles** Que fait la procédure en fin de fichier ? Fonctionne-t-elle tel quel ?
16. Qu'avez-vous modifié pour faire ne plus avoir d'explosion numérique ?
17. Comment modifier DRCN pour faire la même chose. (note technique, comme DRCN est beaucoup plus lent un entrainement pour trouver le noyau n'est pas envisageable, encore désolé).