# Le Debruiteur - Guide utilisateur


- Jonas Freiburghaus
- Romain Capocasale
- He-Arc, INF3dlm-a
- Image Processing course
- 2019-2020


Nous avons respecté au mieux la norme **PEP8** en ce qui concerne la nomenclature et la documentation.
La conclusion se trouve dans le notebook 'Statistics.ipynb'.

## Liste des packages utilisé

- opencv-python
- numpy
- tensorflow
- matplotlib
- pandas
- scikit-image
- Pillow
- tqdm

## Téléchargement des données

### Manuellement

Le téléchargement peut soit se faire manuellement à l'adresse suivante :

http://www.vision.caltech.edu/Image_Datasets/Caltech101/101_ObjectCategories.tar.gz

Il faut mettre le dossier téléchargé au même niveau dans l'arborescence que les notebooks et le nommer 'images'.

### Via le script

Ou au travers du script `download_caltech101.sh` avec la commande `./download_caltech101.sh`, il se peut qu'il faille donner les droits d'exécution avec la commande chmod.

## Installation

### Via docker

Dans le dossier du projet lancer la commande :

```
docker-compose up -d
```

Puis ouvrir un navigateur à l'adresse : `localhost:8888`

### Via l'ordinateur courant

Les différents packages nécaissaires seront installés dans un environnement virtuel. Tout d'abord, il faut avoir `Python` installé sur sa machine.

#### PIP

En suite, il faut s'assurer d'avoir PIP installé avec la commande : `$ pip --version`.

Si PIP n'est pas installé, il peut être installé via le script `get-pip.py` présent dans le répertoire `user_guide` avec la commande `$ python get-pip.py` ou via le site web officiel de PIP.

#### Virtualenv

Une fois PIP installé, veuillez exectuer la commande `$ pip install virtualenv` pour installer virtualenv.

Par la suite, un environnement virtual peut être créé avec la commande : `$ virtualenv [env_name]`.

Une fois l'environnement créé, l'environnement virtuel peut être lancé via la commande : `$ .[env_name]/Script/activate` ou si cela ne fonctionne pas avec : `$ source [env_name]/Scripts/activate`.

#### Requierments

Une fois l'environnement virtuel lancé, veuillez executer la commande : `$ pip install -r requierments.txt` pour installer les packages nécaissaires au projet. Ce fichier est présent dans le dossier `user_guide`.

#### Jupyter

Et finalement :

```
pip install jupyterlab
```

## Arborescence

- `debruiteur` : le dossier contient le package créé pour le projet
  - `generator` :
    - **datagenerator.py :** contient une classe fournissant les fonctionnalités équivalentes à un genrateur python mais pour les images. Cette classe est utile pour fournir des images lors de l'entraînement des réseaux de neurones et ne pas dépasser la taille de la mémoire disponible.
  - `metrics` :
    - **metrics.py :** contient des méthodes permettant de comparer des images entres elles selon les différentes métriques défini pour le projet.
  - `models` :
    - **autoencoder.py :** contient les méthodes permettant de consuire l'architecture des réseau de neurones de type autoencoder (Dense et Convolution).

        - **blocks.py :** contient les méthodes permettant de construire les différents blocs du réseau de neurones GAN.
        - **gan.py :** contient les méthodes nécaissaires à la construction du réseau de neurones Generative Adversarial Network.
    - `noise` :
        - **noise.py :** contient les différentes classes permettant d'ajouter du bruit sur les images.
        - **filters.py :** contient les différents filtres permettant de réduire le bruit sur les images.
    - `plots` :
        - **plots.py :** contient différentes méthodes utilitaires pour l'affichages des images.
    - `preprocessing` :
        - **preprocessor.py :** contient les différentes méthodes permettant de charger les images, de les redimmensionner et d'y ajouter le bruit. Contient également les méthodes permettant de créer les différents dataframe avec le chemin des images.
    - `statistics` :
        - **statistics.py :** contient les méthodes permettant de calculer les différentes statistiques sur les méthodes de réduction de bruits.
    - `utils` :
        - **utils.py :** contient différentes méthodes utilitaires au projet.
- `images` : contient les images initiales.
- `resized_images` : contient les images redimmensionné (ce dossier se crée après avoir exécuté la cellule en question)
- `noised_images` : contient les images redimmensionné (ce dossier se crée après avoir exectué la cellule en question)
- `saved_models` : contient les modèles entrainés
- `user_guide` : contient le guide utilisateur
- `NoiseType.ipynb` : notebook jupyter qui présente les différents types de bruits appliqués aux images utilisées pour le bruit. Le notebook contient également des explications sur l'origine de ces types de bruits.
- `Filters.ipynb` : notebook jupyter qui présente les différents filtres appliqués sur des images bruitées.
- `NeuralNetworkTraining.ipynb` : notebook jupyter pour l'entraînement des modèles. Il n'est pas nécaissaire d'entraîner les modèles à chaque fois. Une fois qu'un modèle est entrainé, il est enregistré dans `saved_models` et peut être rechargé à tout moment.
- `NoiseReductionNeuralNetwork.ipynb` : notebook jupyter qui présente les résultats des réseaux de neuronnes préalablement entrainés sur différentes images bruitées.
- `Statistics.ipynb` : notebook jupyter qui compare les différentes techniques (filtres et réseaux de neuronnes) utilisé pour ce projet sur les différents types de bruits. Le notebook contient également différentes statistiques sur ces méthodes ainsi que la conclusion de notre projet.
- `denoise.py` : Script utilitaire pour débruiter une image.
- `Dockerfile` : Image docker avec les packages nécessaires pour lancer le projet
- `docker-compose.yml` : Facilite la mise en place de l'image docker

## Exécution des notebooks

Les notebook peuvent être exectués avec la commande : `$ jupyter notebook`. Une fois le notebook en question choisi, il faut se rendre sous l'onglet `Noyau` -> `Changer de noyau` et verifier que l'environnement créé précédemment est bien activé pour le notebook.

## Exécution du script utilitaire

```
python denoise.py -i <input_img_path> -o <output_img_path> -m <model name> -p
<post processing filter>
```

Exemple :

```
python denoise.py -i test.png -o denoised.png -m 0 -p 2
```

Pour obtenir plus d'information sur les différentes options :

```
python denoise.py -h
```

## Run unit test

First make sure there is an image in test_image dir and then run.

```
python -m unit_test TestDenoiseMethods
```