Université de Strasbourg
CNAM

TP – Implémentation d'une Pipeline simple en Python

L'objectif de cet exercice est de créer une première version d'une pipeline Python qui a pour objectif de traiter un dossiers contenant des images. La structure du projet pourra ressembler à celle décrite en figure 1.

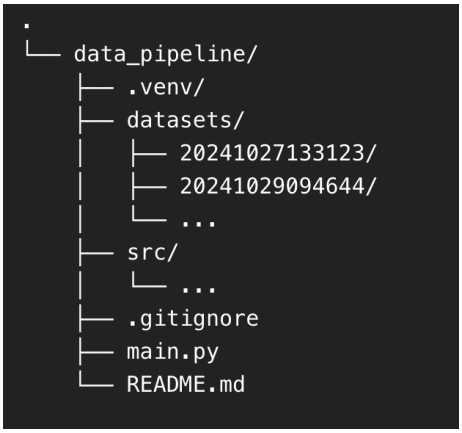

Figure 1 – Exemple de structure du projet

Quelques éléments à prendre en compte :

- Le fichier *main.py* est le point d'entrée (automatiser le lancement du fichier en créant une configuration PyCharm).
- Écrire une version minimale pour le *README.md*.
- Le placement de vos fichiers dans le dossier *src/* devra être cohérent.
- Les images à traiter (situées par exemple dans un dossier *input_ images/*) peuvent être localisées en dehors du projet. À défaut, penser à les exclure du versioning via le fichier *.gitignore*.

Les opérations à réaliser sont les suivantes :

- Créer une classe *ImageProcessor*, qui accepte dans son constructeur un chemin d'accès relatif qui pointe sur le dossier contenant les images à traiter.
- Exposer, via cette classe, une méthode *process_folder()* qui permet de parcourir le dossier d'images puis, pour chaque image :

1. Ouvre l'image.
2. Redimensionne l'image vers un format carré en utilisant une taille de destination fournie en paramètre de la fonction :
    - Seul un *int* est nécessaire (e.g. 640 si on souhaite que l'image de destination ait une dimension de 640*640).
    - L'opération peut être réalisée avec la methode *resize()* de Pillow.
    - Attention à gérer les cas où la largeur est supérieure, inférieure ou égale à la hauteur de l'image. L'important est de garder le même ratio entre la largeur et la hauteur afin que l'image ne soit pas déformée.

Figure 2 - Une image avant (2048*1536) et après (640x480) l'opération *resize()*

3. Appliquer une opération de remplissage (*padding*) sur l'image lorsque celle-ci n'est pas carrée. Cette opération consiste à ajouter des pixels sur les côtés les plus courts de l'image pour obtenir un format carré. Les pixels ajoutés peuvent en général être de couleur unie, on peut aussi utiliser un effet de flou/miroir des bords existants, permettant ainsi d'obtenir systématiquement des images aux dimensions égales en hauteur et en largeur.
    - Un exemple d'explication de padding peut être trouvé ici (section *Add padding to the rectangle to make it square).*
    - Le padding devra uniquement être ajouté à droite (si largeur $<$ hauteur) ou en bas (si largueur $>$ hauteur).
    - Le padding devra être de couleur unie avec un code couleur de (114, 114, 144).

Figure 3 – Exemple de padding à droite (couleur noir) lorsque largeur < hauteur

4. L'image qui résulte est sauvegardée dans un nouveau dossier identifié par un identifiant unique (e.g. date au format YmdHMS), lui-même placé dans un dossier *dataset* situé à la racine du projet. Un nouveau dossier est donc généré à chaque exécution de la pipeline.

- Enfin, appeler la pipeline via le fichier *main.py* en utilisant une configuration PyCharm.

Les points essentiels à prendre en compte :

- S'assurer qu'il n'y a pas d'erreur de linting.
- Ajouter tous les type hints.
- Utiliser un formatter (black, ruff, etc.).
- Ajouter toutes les docstring (quand nécessaire) et garder le format consistant.
- Ajouter et configurer le *pre-commit*.
- Utiliser les images fournies avec l'exercice (*input_ images.zip*).

Quelques pistes pour aider :

- Créer un identifiant unique avec le module *datetime* :

```
from datetime import datetime
datetime.now().strftime("%Y%m%d%H%M%S")
```

- Parcourir un dossier :

```
import os
urls = os.path.listdir(url)
```

- La manipulation des images se fera avec Numpy et Pillow.