HANDS ON AI – Certificat IA de l'UMONS

Initiation aux outils et frameworks

Fondements pratiques en vision par ordinateur sous Python

- ## Introduction :

L'objectif de cette séance est de se familiariser avec l'environnement **Jupyter Notebook** et le langage Python *via* la plateforme **Google Colab**. On commencera par la création d'un fichier **Jupyter Notebook** avec l'utilisation du processeur Graphique GPU sur le Cloud. Ensuite, nous verrons ensemble comment charger (upload) des fichiers dans **Google Colab** ainsi que les librairies nécessaires afin de mettre en œuvre des exemples de traitement d'images. A la fin de cet exercice, vous pourrez visualiser la représentation des caractéristiques d'une image via des convolutions avec la librairie **PyTorch**. Cette séance d'initiation est donc présentée en trois parties : 1. Préparation de l'environnement ; 2. Traitement d'images ; 3. Combinaison des différents filtres.

0. Préparation de l'environnement :

a. Se connecter : pour commencer, il faut utiliser un navigateur (de préférence Chrome) et lancer Google colab : https://colab.research.google.com. Il faudra ensuite se connecter avec votre adresse **Gmail.**

b. Charger le fichier de démarrage : charger le code de démarrage « **Initiation_HandsOnAI_Input_PARTIE1.ipynb** », disponible sur Moodle, depuis votre environnement Google Colab (Figure 1).

c. Vérifier le matériel réservé : vérifier que le GPU est bien sélectionné en allant sur : Exécution ->Modifier le type d'exécution. Sélectionnez le GPU si ce n'est pas encore fait.

d. Charger les librairies nécessaires (OpenCV, matplotlib, numpy, etc.)

e. Vérifier le chargement des librairies : par exemple, récupérer la version de la librairie OpenCV

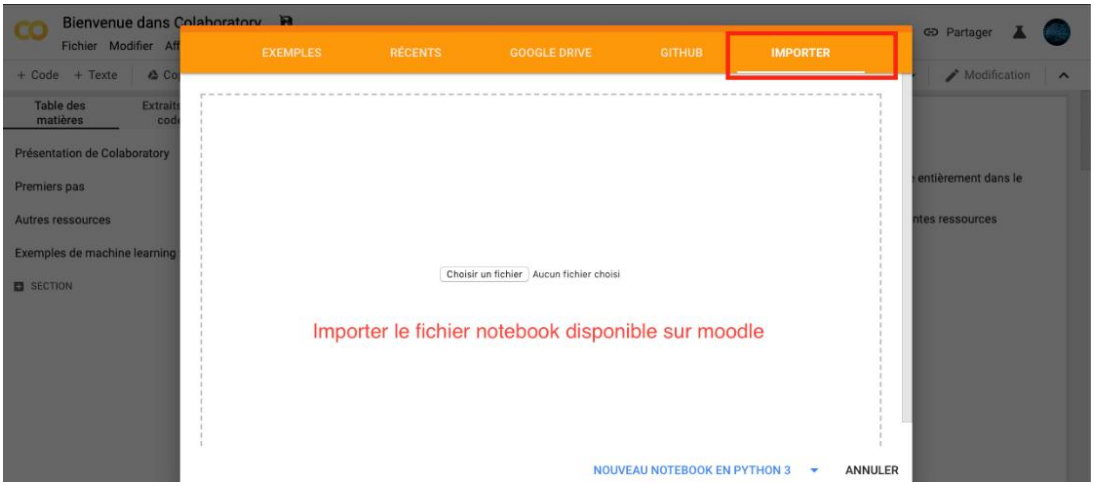

Figure 1: Importation d'un notebook avec Google colab

1. Traitement d'images avec Python :

Dans cette section, nous vous invitons à réaliser les dix (10) tâches décrites ci-dessous afin de prendre en main les différentes fonctions nécessaires pour réaliser un algorithme de traitement et extraction de caractéristiques d'images.

Instruction 1.1. **Chargement des librairies**

Instruction 1.2. **Charger une image couleur de votre choix sur l'environnement Google colab :** en utilisant soit soit la commande « **files.upload()** » ou la commande **wget** pour récupérer une image depuis internet

Instruction 1.3. **Vérifier le bon chargement de l'image :** à l'aide de la commande : ls -l

Question 1.1. **Afficher l'image chargée :** en utilisant soit IPython ou OpenCV

Question 1.2. **Compléter l'affichage via un autre moyen**

Instruction 1.4. **Convertir l'image en noir et blanc (niveau de gris) :** en utilisant la fonction OpenCV « **cv2.cvtColor** », enregistrer l'image : en utilisant la fonction OpenCV « **imwrite** » et afficher

Question 1.3. **Vérifier si l'image est bien enregistrée**

Instruction 1.5. **Tester la convolution d'images avec le filtre gaussien :** en utilisant différentes tailles de filtres

Question 1.4. **Enregistrer l'image et vérifier si l'image est bien enregistrée**

Question 1.5. **A partir de la même image, extraire différentes caractéristiques avec les filtres suivants :**

- Filtre gaussien (GaussienBlur) avec une taille de filtre (3,3) (Fig. 2)
- Filtre gaussien (GaussienBlur) avec une taille de filtre (15,15)
- Filtre Médian avec une taille de filtre (3,3)
- Filtre bilatéral avec une taille de filtre (3,3)
- Filtre de Sobel (horizontal)
- Filtre de Sobel (vertical)
- Filtre Canny pour la détection des contours

- Filtre Canny avec de nouveaux paramètres

Question 1.6. **Calculer les filtres « horizontal et vertical » de Prewitt :** l'aide des valeurs du filtre (Fig. 3)

Instruction 1.7. **Afficher toutes les images extraites.**

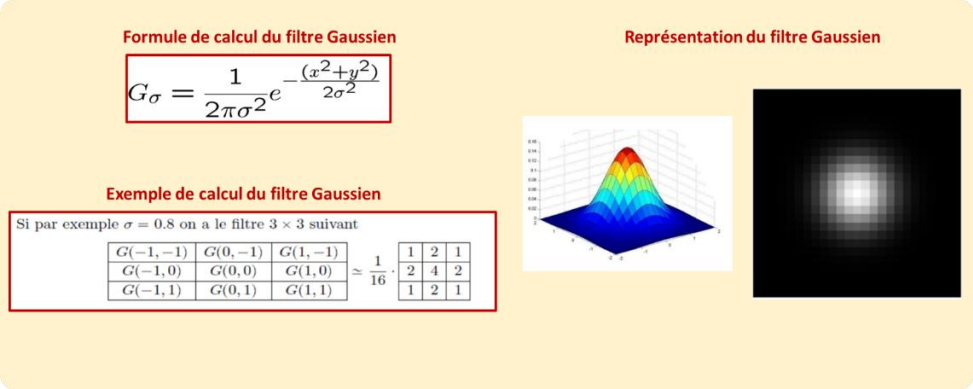

Figure 2: calcul et représentation du filtre gaussien

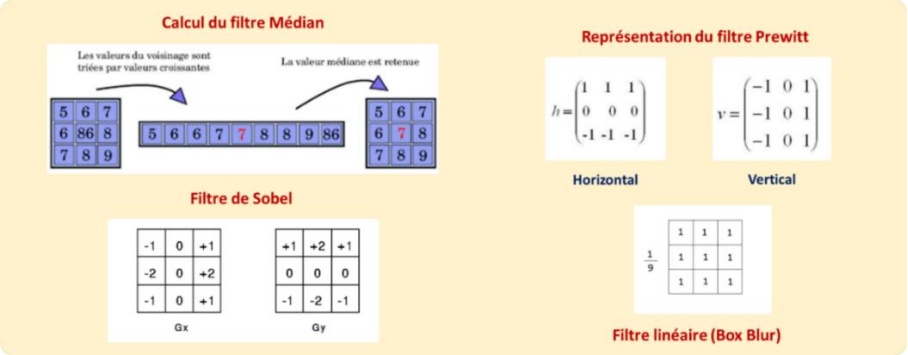

Figure 3: calcul et représentation des filtre Médian, Sobel, Prewitt et linéaire

- **Exercice N° 01 :** développer un algorithme de traitement d'images permettant de :
  a. Charger une image depuis ce lien avant de la convertir en noir et blanc ;
  b. Réduire les bruits de l'image en utilisant le filtre Gaussien ;
  c. Appliquer le filtre de Sobel (horizontalement (Gx) puis verticalement (Gy)) ;
  d. Calculer le Gardient de Sobel résultant (G) avec la formule : $\mathbf{G} = \sqrt{\mathbf{G}_x{}^2 + \mathbf{G}_y{}^2}$
  e. Appliquer le filtre de détection de contours de Canny
  f. Afficher les images résultantes sous le même plot (Fi. 4).

Figure 4: Exemple de résultat de l'exercice 01

2. Combinaison de caractéristiques (Partie II) :

Pour cette partie, veuillez charger le fichier de démarrage « **Initiation_HandsOnAI_Input_PARTIE2.ipynb** », disponible sur Moodle, depuis votre environnement Google Colab.

Durant cette partie, vous pouvez appliquer des transformations qui seront utilisées dans la suite du cours comme les fonctions « **MaxPooling »** et « **Flatten »** en illustrant leur effet. Le MaxPooling permet de réduire la dimension de la carte de caractéristiques en prenant la valeur maximum du filtre choisi comme l'illustre la Figure 5. Le Flatten permet de linéariser en 1D (fusionner) les cartes de caractéristiques en un vecteur.

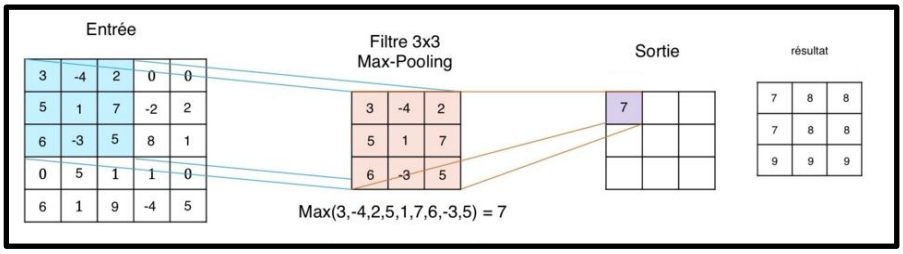

Figure 5: Illustration du Max Pooling

Nous vous proposons de suivre les étapes suivantes :

Instruction 2.1. **Importer les librairies**

Question 2.1. **Appliquer les différents filtres sur une image en niveau de gris**

Instruction 2.2. **Lire et exécuter les fonctions pour transformer en Tensor et en Numpy**

Question 2.2. **Compléter pour créer le dictionnaire des filtres avec leur titre et construire la liste des cartes de caractéristiques**

Instruction 2.3. **Afficher les cartes de caractéristiques**

Question 2.3. **Appliquer un MaxPooling à chaque carte de caractéristiques et afficher les cartes réduites**

```
pool = nn.MaxPool2d(kernel_size=2, stride=2)
```

Figure 6 : Résultat attendu pour le Maxpooling

Question 2.4. **Concaténation des cartes de caractéristiques**

```
Stack_viz = torch.cat(ensemble_a_concatener, dim = dimension_pour_concatener)
```

Question 2.5. **Aplatir** pour avoir un tensor 1D et affichage des caractéristiques du vecteur

```
Fusion_vector = stack_viz.flattent(start_dim=1).suqeeze(0).cpu().numpy()
```

Instruction 2.4. **Afficher l'ensemble des résultats avec les liens** :

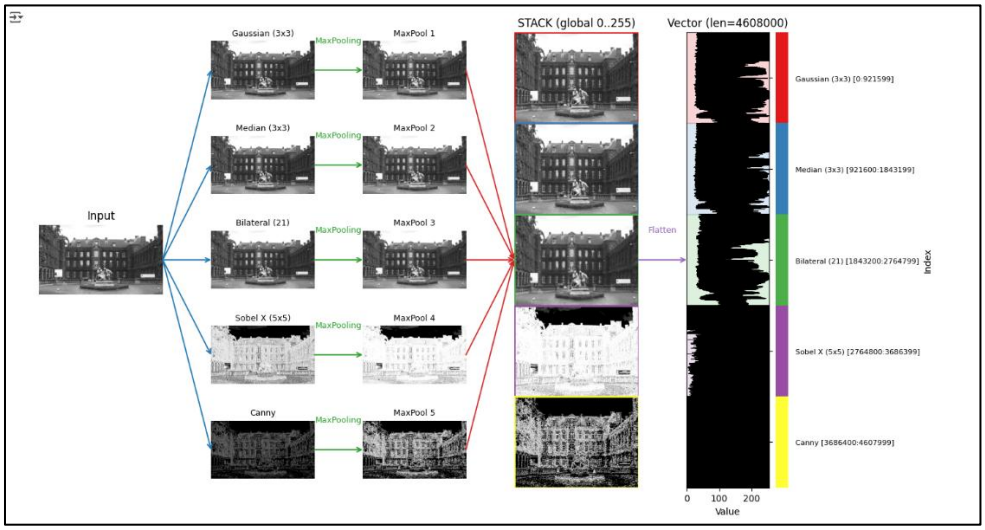

Figure 7 : Visualisation complète

Instruction 2.5. **Appliquer la même chose avec les couches convolutionnelles « conv2D » :**

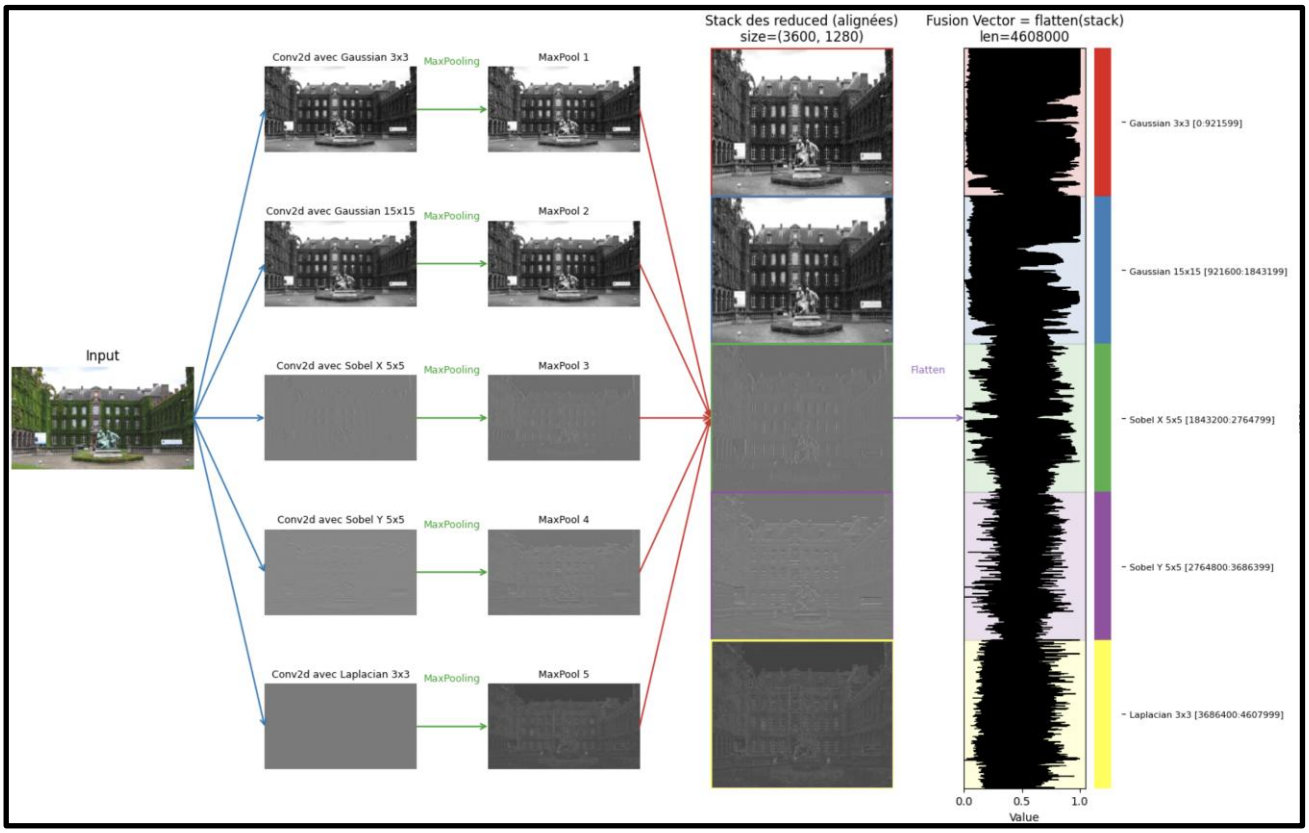

Figure 8 : Visualisation avec les Conv2D