



UNIVERSITÉ SIDI MOHAMED BEN ABDLLAH
FACULTÉ DES SCIENCES DHAR EL MAHRAZ DE FÈS



DÉPARTEMENT D'INFORMATIQUE



DATA MINING

Image Dataset Pre-Processing

Présenter par :
ABIBOU SOUKAYNA

Supervisé par :
RIFFI JAMAL

Année universitaire : 2020/2021

Filière MIDVI

I- Introduction :

Dans cet article, nous allons passer par les étapes de prétraitement d'image nécessaire avant de les transmettre à des modèles d'apprentissage automatique, ce processus de nettoyage et d'encodage est appelé *prétraitement d'image*.

Qu'est-ce que le traitement d'image?

Le traitement d'image est divisé en traitement d'image analogique et traitement d'image numérique.

Le traitement d'image numérique est l'utilisation d'algorithmes informatiques pour effectuer un traitement d'image sur des images numériques. Il permet d'appliquer une gamme beaucoup plus large d'algorithmes aux données d'entrée – le but du traitement d'image numérique est d'améliorer les données d'image (caractéristiques) en supprimant les distorsions indésirables et / ou en améliorant certaines caractéristiques importantes de l'image afin que nos modèles d'IA puissent bénéficier de ces données améliorées pour travailler.

Dans ce noyau, nous allons voir quelques étapes et techniques de base de nettoyage des images pour encoder des données images.

NB : Nous travaillerons sur l'environnement Google colab.

II- Prétraitement d'image :

1. Importation de bibliothèques:



```
+ Code + Texte
RAM
Disque
Modification
^

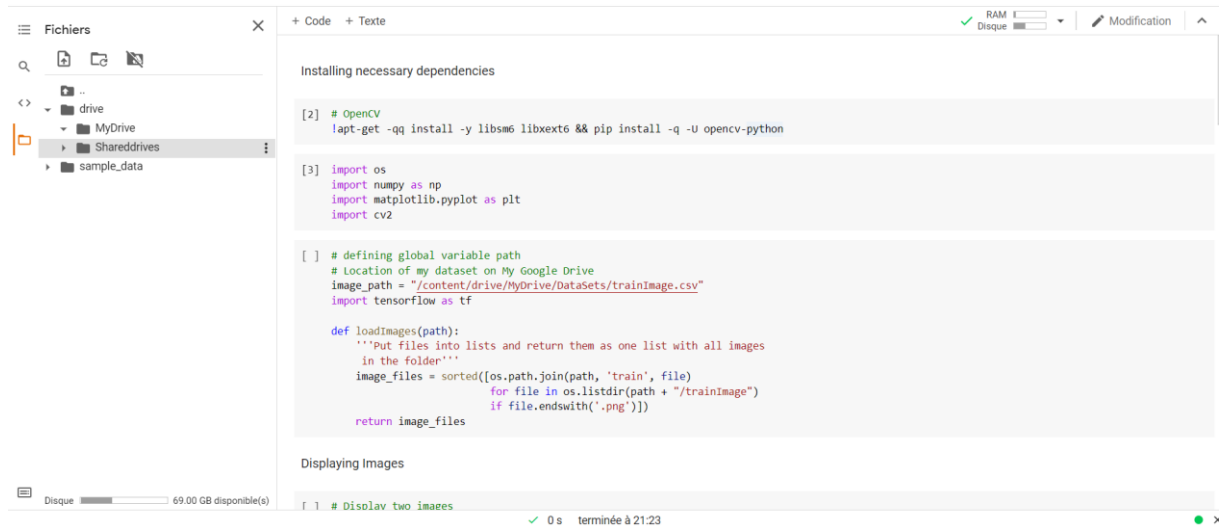
Installing necessary dependencies

[2] # OpenCV
lapt-get -qq install -y libsm6 libxext6 && pip install -q -U opencv-python

import os
import numpy as np
import matplotlib.pyplot as plt
import cv2
```

2. Télécharger les données :

La première étape consiste à télécharger notre ensemble de données dans cet environnement Google colab.



The screenshot shows a Google Colab interface. On the left, a file explorer shows a directory structure with 'drive', 'MyDrive', 'Shared drives', and 'sample_data'. The main area contains a code editor with the following code:

```
Installing necessary dependencies

[2] # OpenCV
!apt-get -qq install -y libsm6 libxext6 && pip install -q -U opencv-python

[3] import os
import numpy as np
import matplotlib.pyplot as plt
import cv2

[ ] # defining global variable path
# Location of my dataset on My Google Drive
image_path = "/content/drive/MyDrive/Datasets/trainImage.csv"
import tensorflow as tf

def loadImages(path):
    '''Put files into lists and return them as one list with all images
    in the folder'''
    image_files = sorted([os.path.join(path, 'train', file)
                          for file in os.listdir(path + "/trainImage")
                          if file.endswith('.png')])
    return image_files

Displaying Images

[ ] # Display two images
```

The bottom status bar indicates '0 s terminée à 21:23'.

3. Nettoyage du data :

Nous allons voir :

- Lire l'image
- Redimensionner l'image
- Supprimer le bruit (Denoise)
- Segmentation
- Morphologie (lissage des bords)

Étape 1: Lisez les images.

Dans cette étape, nous stockons le chemin de notre donnée des images dans une variable puis nous avons créé une fonction pour charger des dossiers contenant des images dans des tableaux.

```
# defining global variable path
# Location of my dataset on My Google Drive
image_path = "/content/drive/MyDrive/DataSets/trainImage.csv"
import tensorflow as tf

def loadImages(path):
    '''Put files into lists and return them as one list with all images
    in the folder'''
    image_files = sorted([os.path.join(path, 'train', file)
                          for file in os.listdir(path + "/trainImage")
                          if file.endswith('.png')])
    return image_files
```

Étape 2: Redimensionner l'image.

Dans cette étape afin de visualiser le changement, nous allons créer deux fonctions pour afficher les images, la première étant une pour afficher une image et la seconde pour deux images. Après cela, nous créons une fonction appelée traitement qui ne reçoit que les images en tant que paramètre.

Pourquoi redimensionnons-nous notre image pendant la phase de prétraitement?

Certaines images capturées par une caméra et transmises à notre algorithme d'IA varient en taille, par conséquent, nous devons établir une taille de base pour toutes les images introduites dans nos algorithmes d'IA.

A screenshot of a code editor window. The window has a title bar with '+ Code' and '+ Texte' on the left, and 'RAM', 'Disque', 'Modification', and a refresh icon on the right. The code is written in Python and is as follows:

```
# Preprocessing
def processing(data):

    # Reading 3 images to work
    img = [cv2.imread(i, cv2.IMREAD_UNCHANGED) for i in data[:3]]
    try:
        print('Original size',img[0].shape)
    except AttributeError:
        print("shape not found")

    # -----
    # setting dim of the resize
    height = 220
    width = 220
    dim = (width, height)
    res_img = []
    for i in range(len(img)):
        res = cv2.resize(img[i], dim, interpolation=cv2.INTER_LINEAR)
        res_img.append(res)

    # Checking the size
    try:
        print('RESIZED', res_img[1].shape)
    except AttributeError:
        print("shape not found")

    # Visualizing one of the images in the array
    original = res_img[1]
    display_one(original)
```

- Taille d'origine (360, 480, 3) – (largeur, hauteur, nombre de canaux RVB)
- Redimensionné (220, 220, 3)

Étape 3: Supprimez le bruit (Denoise).

Pourtant, à l'intérieur de la fonction Processing (), nous ajoutons ce code pour lisser notre image afin d'éliminer les bruits indésirables. Nous faisons cela en utilisant le **flou gaussien**.

Le **flou gaussien** (également appelé lissage **gaussien**) est le résultat du **flou d'**une image par une fonction **gaussienne**. C'est un effet largement utilisé dans les logiciels graphiques, généralement pour réduire le bruit de l'image .

Le lissage gaussien est également utilisé comme étape de prétraitement dans **les** algorithmes de **vision par ordinateur** afin d'améliorer les structures d'images à différentes échelles.

```

# -----
# Remove noise
# Using Gaussian Blur
no_noise = []
for i in range(len(res_img)):
    blur = cv2.GaussianBlur(res_img[i], (5, 5), 0)
    no_noise.append(blur)

image = no_noise[1]
display(original, image, 'Original', 'Blured')
#-----

```

Étape 4: Segmentation et morphologie.

Dans cette étape, nous allons segmenter l'image, séparant l'arrière-plan des objets de premier plan et nous allons améliorer encore notre segmentation avec plus de suppression du bruit.

```

# Segmentation
gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
ret, thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)

# Displaying segmented images
display(original, thresh, 'Original', 'Segmented')

```

Nous pouvons voir que l'image ci-dessus a besoin d'être améliorée, par conséquent, nous appliquons un autre flou pour améliorer l'apparence avec le code suivant.

```

# Further noise removal (Morphology)
kernel = np.ones((3, 3), np.uint8)
opening = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel, iterations=2)

# sure background area
sure_bg = cv2.dilate(opening, kernel, iterations=3)

# Finding sure foreground area
dist_transform = cv2.distanceTransform(opening, cv2.DIST_L2, 5)
ret, sure_fg = cv2.threshold(dist_transform, 0.7 * dist_transform.max(), 255, 0)

# Finding unknown region
sure_fg = np.uint8(sure_fg)
unknown = cv2.subtract(sure_bg, sure_fg)

#Displaying segmented back ground
display(original, sure_bg, 'Original', 'Segmented Background')

```

Maintenant, nous séparons différents objets de l'image avec des marqueurs.

```

# Marker labelling
ret, markers = cv2.connectedComponents(sure_fg)

# Add one to all labels so that sure background is not 0, but 1
markers = markers + 1

# Now, mark the region of unknown with zero
markers[unknown == 255] = 0

markers = cv2.watershed(image, markers)
image[markers == -1] = [255, 0, 0]

# Displaying markers on the image
display(original, markers, 'Original', 'Marked')

```

➤ Sortie finale :

+ Code + Text

✓ RAM
Disque

Modification

Main Function the heart of the program

▶

```
def main():  
    # calling global variable  
    global image_path  
    '''The var Dataset is a list with all images in the folder '''  
    dataset = loadImages(image_path)  
    print('number of FILES in dir', len(dataset))  
    print("-----")  
    #print(cv2.imread(dataset[0]).shape)  
    print("List of files the first 3 in the folder:\n",dataset[:3])  
    print("-----")  
  
    # sending all the images to pre-processing  
    processing(dataset)  
  
main()
```


Webographie

➤ Code :

<https://colab.research.google.com/drive/1G66Tee6UgPbQSkU88IZY9mlF3qd5Kpt#scrollTo=Q7eL5692QNjv>