M1, Intelligent Processing Systems

Mini projet open source

CARTOONIFIER une image avec OpenCV

Auteurs :

Abouche HIND

Superviseur : Dr. ABDELLAH IDRISSI

10 octobre 2020

Promotion 2019 - 2021

Sommaire

1 Introduction

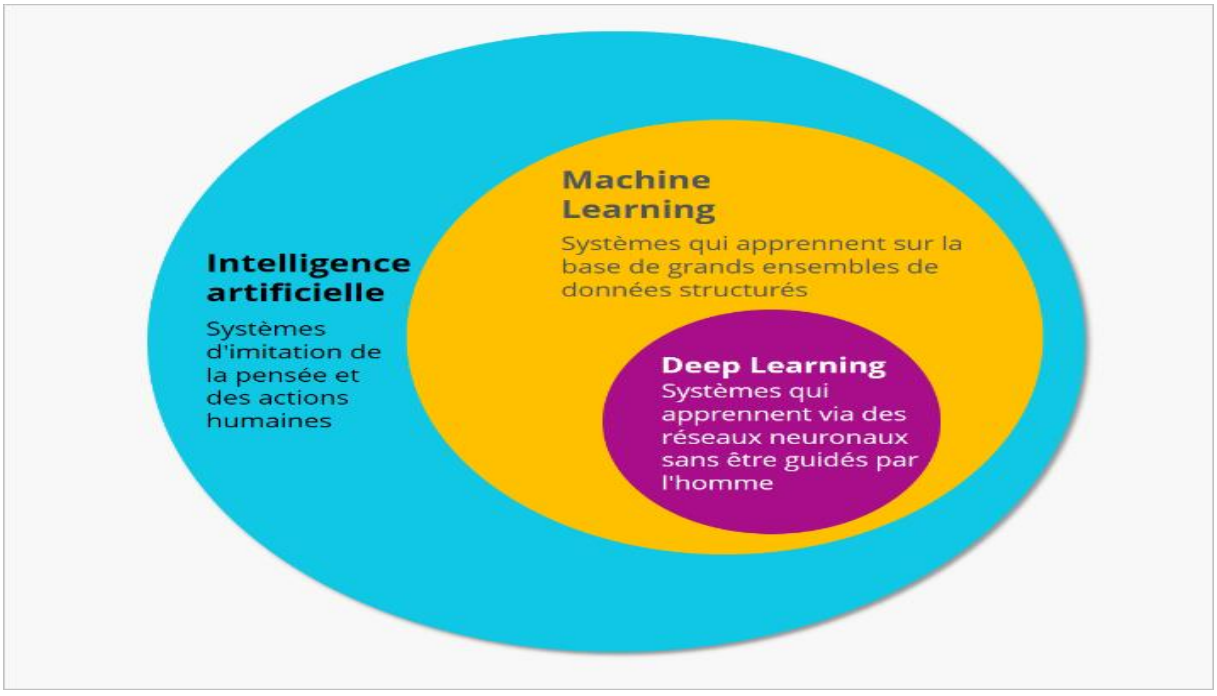
1.1 Introduction générale :

L'intelligence artificielle permet aux machines d'apprendre par « expérience », de s'adapter aux nouvelles données et d'effectuer des tâches comme le ferait un être humain de manière que l'on qualifierait d'« intelligente ».

Le machine learning est un élément essentiel de l'IA. C'est la science dotant les ordinateurs de la possibilité d'agir sans être explicitement programmés.

Les deux concepts semblent très similaires, mais en quoi diffèrent-ils ? Fondamentalement, l'IA est l'intelligence, celle qui dote les machines de leur apparente intelligence, alors que le machine learning consiste en l'application de méthodes de calcul qui la soutiennent. En d'autres termes, l'IA est la science et le machine learning, c'est le logiciel et les algorithmes qui rendent les machines intelligentes.

Le Deep Learning, ou apprentissage profond, est l'une des principales technologies de Machine Learning et d'intelligence artificielle.

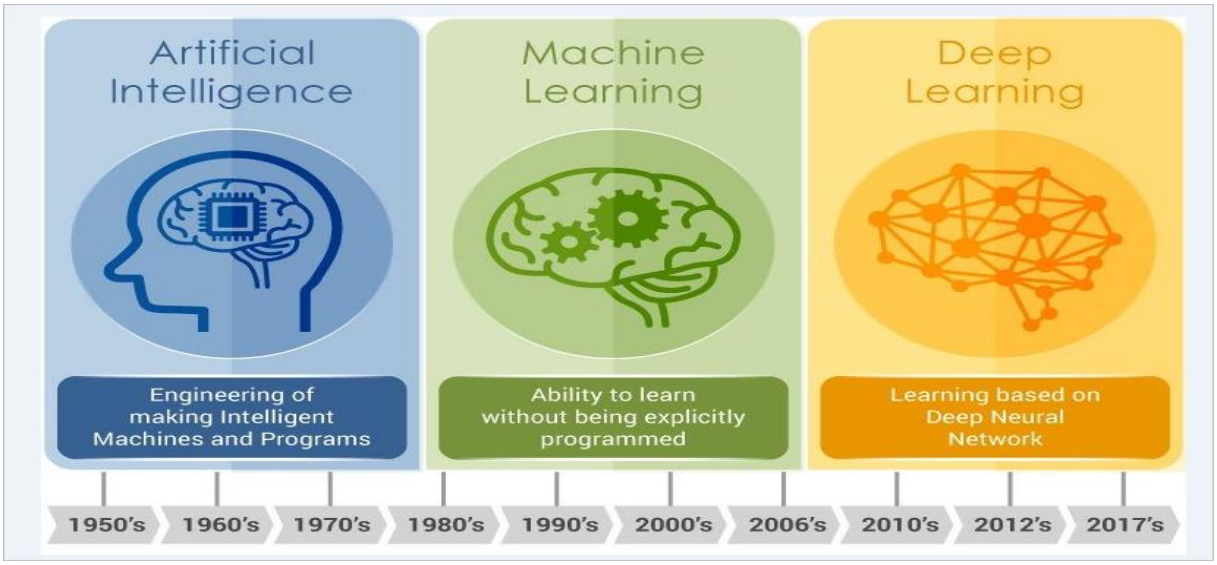


Le Deep Learning une forme d'intelligence artificielle, dérivée du Machine Learning, Pour comprendre ce qu'est le Deep Learning, il convient donc de comprendre ce qu'est le Machine Learning.

Le concept de Machine Learning date du milieu du 20ème siècle. Dans les années 1950, le mathématicien britannique Alan Turing imagine une machine capable d'apprendre, une « Learning Machine ». Au cours des décennies suivantes, différentes techniques de Machine Learning ont été développées pour créer des algorithmes capables d'apprendre et de s'améliorer de manière autonome.

Le Machine learning est la technologie la plus ancienne et la plus simple. Elle s'appuie sur un algorithme qui adapte lui-même le système à partir des retours faits par l'humain. La mise en place de cette technologie implique l'existence de données organisées. Le système est ensuite alimenté par des données structurées et catégorisées lui permettant de comprendre comment classer de nouvelles données similaires. En fonction de ce classement, le système exécute ensuite les actions programmées. Il sait par exemple identifier si une photo montre un chien ou un chat et classer le document dans le dossier correspondant.

Si le Machine Learning (ML) et le Deep Learning (DL) sont des Intelligences Artificielles, l'inverse n'est pas vrai. Par exemple, les graphiques de connaissances ou les moteurs de règles sont des Intelligences Artificielles mais ne relèvent pas du ML ni du DL.



1.2 Objectif du projet

Idée de projet: Transformez les images en dessin animé. Oui, l'objectif de ce projet d'apprentissage automatique est de CARTOONIFIER les images. Ainsi, nous allons créer une application python qui transformera une image en son dessin animé à l'aide d’OpenCV.

2 APERÇU DES TECHNOLOGIES UTILISÉES

2.1 Différence entre Deep Learning et computer vision

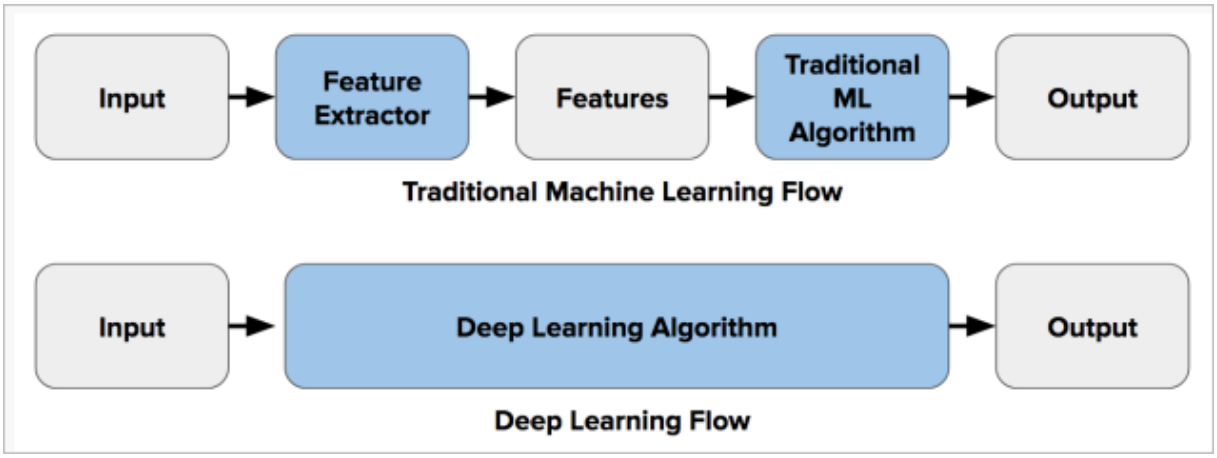
La vision par ordinateur est le domaine d'étude qui permet aux ordinateurs de voir et d'identifier des images et des vidéos numériques comme le ferait un humain. Les défis auxquels il est confronté découlent en grande partie de la compréhension limitée de la vision biologique. La vision par ordinateur implique l'acquisition, le traitement, l'analyse et la compréhension d'images numériques pour extraire des données de haute dimension du monde réel afin de générer des informations symboliques ou numériques qui peuvent ensuite être utilisées pour prendre des décisions. Le processus comprend souvent des pratiques telles que la reconnaissance d'objets, le suivi vidéo, l'estimation de mouvement et la restauration d'image

La vision par ordinateur est devenue l'un des domaines de recherche vitaux et les applications commerciales liées à l'utilisation des méthodologies de vision par ordinateur deviennent une part considérable de l'industrie. La précision et la vitesse de traitement et d'identification des images capturées à partir de caméras se sont développées au fil des décennies. Étant le garçon bien connu de la ville, Deep Learning joue un rôle majeur en tant qu'outil de vision par ordinateur

La différence est que les systèmes de vision traditionnels impliquent un humain disant à une machine ce qui devrait être là par rapport à un algorithme de Deep Learning extrayant automatiquement les caractéristiques de ce qui est là.

Deep Learning qui est un sous-ensemble de l'apprentissage automatique, a montré un gain significatif de performances et de

précision dans le domaine de la vision par ordinateur.



Exemple pour bien comprendre la différence : « If you want to teach a [deep] neural network to recognize a cat, for instance, you don’t tell it to look for whiskers, ears, fur, and eyes. You simply show it thousands and thousands of photos of cats, and eventually it works things out. If it keeps misclassifying foxes as cats, you don’t rewrite the code. You just keep coaching it ».

2.2 Opencv

OpenCV (Open Source Computer Vision Library) est une bibliothèque de logiciels et de logiciels d'apprentissage automatique. Il a été conçu pour divers objectifs tels que l'apprentissage automatique, la vision par ordinateur, l'algorithme, les opérations mathématiques, la capture vidéo, le traitement d'images, etc.

La bibliothèque contient plus de 2500 algorithmes optimisés, qui ont une excellente précision en termes de performances et de vitesse. Ces algorithmes peuvent être utilisés pour détecter et reconnaître des visages, identifier des objets, classer des actions humaines dans des vidéos, suivre des mouvements de caméra, etc.

2.3 Les package importées

a) Package cv2 :

Importé pour utiliser OpenCV pour le traitement d'image

b) Package easygui :

EasyGUI est un module pour la programmation GUI très simple et très facile en Python. EasyGUI est différent des autres générateurs GUI en ce que EasyGUI n'est PAS piloté par les événements. Au lieu de cela, toutes les interactions GUI sont appelées par de simples appels de fonction.

c) Package numpy :

NumPy est la bibliothèque la plus populaire de calcul scientifique en Python. Ainsi, elle permet d'effectuer les calculs scientifiques de base et de manipuler assez facilement les tableaux multidimensionnels

d) Package matplotlib :

Elle s'agit sûrement de l'une des bibliothèques python les plus utilisées pour représenter des graphiques en 2D. Elle permet de produire une grande variété de graphiques et ils sont de grande qualité

e) Package os :

Ce module fournit une façon portable d'utiliser les fonctionnalités dépendantes du système d'exploitation. Si vous voulez simplement lire ou écrire un fichier, voir **open()**, si vous voulez manipuler les chemins de fichiers, voir le module **os.path** etc.

f) Package tkinter :

Le paquet tkinter (« interface Tk ») est l'interface Python standard de la boîte à outils d'IUG Tk. Tk et tkinter sont disponibles sur la plupart des plates-formes Unix, ainsi que sur les systèmes Windows (Tk lui-même ne fait pas partie de Python ; il est maintenu par ActiveState).

2.4 Caricaturer une image

Au cours des dernières années, un logiciel de dessinateur professionnel est apparu partout. Pour convertir une image en dessin animé, plusieurs transformations sont effectuées :

a) L’image est convertie en image en niveaux de gris. Oui, semblable aux photos de l’ancien jour.
b) Ensuite, l'image en niveaux de gris est lissée
c) Extraire les bords de l'image.
d) Enfin, nous formons une image couleur et la masquons avec des bords. Cela crée une belle image de dessin animé avec des bords et une couleur allégée de l'image originale.

Nous allons bien expliquer ces étapes dans le chapitre de l’implémentation.

3 Implémentation du code en python avec explication

3.1 Introduction

Votre enfance vous manque? Oui, tout le monde le fait. Alors, aujourd'hui, nous allons donner à nos images des effets caricaturaux

3.2 Étapes pour développer le dessin animé d'une image

Figure : image originale pour cartonnier

a) Étape 1: importation des modules requis

Nous importerons les modules suivants:

```
import cv2 #importé pour utiliser OpenCV pour le traitement d'image
import easygui #importé pour ouvrir une boîte de fichiers.
#Cela nous permet de sélectionner n'importe quel fichier de notre système.
import numpy as np  #les images sont stockées et traitées sous forme de nombres. Ceux-ci sont considérés comme des tablea
# Nous utilisons NumPy pour traiter les tableaux.
import imageio #Utilisé pour lire l'image qui est choisi par boîte de fichier en utilisant un chemin.

import sys
import matplotlib.pyplot as plt #Cette bibliothèque est utilisée pour la visualisation et le traçage.
import os #pour l'interaction avec le système d'exploitation.
#Ici, pour lire le chemin et enregistrer les images sur ce chemin
import tkinter as tk
from tkinter import filedialog
from tkinter import *
from PIL import ImageTk, Image
```

b) Étape 2: Création d'une boîte de fichiers pour choisir un image particulière :

Dans cette étape, nous allons créer la fenêtre principale de notre application, où les boutons, les étiquettes et les images résideront

```
def upload():
    ImagePath=easygui.fileopenbox()
    cartoonify(ImagePath)
```

Le code ci-dessus ouvre la boîte de fichier, c'est-à-dire la boîte de dialogue pour choisir l'image de l'appareil, qui s'ouvre chaque fois que nous exécutons le code.

fileopenbox () est la méthode du module easyGUI qui renvoie le chemin du fichier choisi sous forme de chaîne de caractéres.

- Maintenant, toutes les opérations se feront sur le clic du bouton, donc toutes les étapes ci-dessous font partie de la fonction cartoonify (ImagePath) :

```
def cartoonify(ImagePath):
```

c) Étape 3: Comment une image est-elle stockée ? :

Pour un ordinateur, tout n'est que des chiffres. Ainsi, dans le code ci-dessous, nous convertirons notre image en un tableau numpy

```
# lire l'image
originalmage = cv2.imread(ImagePath)
originalmage = cv2.cvtColor(originalmage, cv2.COLOR_BGR2RGB)
# l'image est stockée sous forme de nombres

# confirmer que l'image est choisie
if originalmage is None:
    print("Can not find any image. Choose appropriate file")
    sys.exit()

ReSized1 = cv2.resize(originalmage, (960, 540))
plt.imshow(ReSized1, cmap='gray')
```

Imread est une méthode en cv2 qui permet de stocker des images sous forme de nombres. Cela nous aide à effectuer des opérations en fonction de nos besoins. L'image est lue comme un tableau numpy, dans lequel les valeurs de cellule représentent les valeurs R, V et B d'un pixel.

Nous redimensionnons l'image après chaque transformation pour afficher enfin toutes les images à une échelle similaire

d) Étape 4: Transformer une image en niveaux de gris :

- Code :

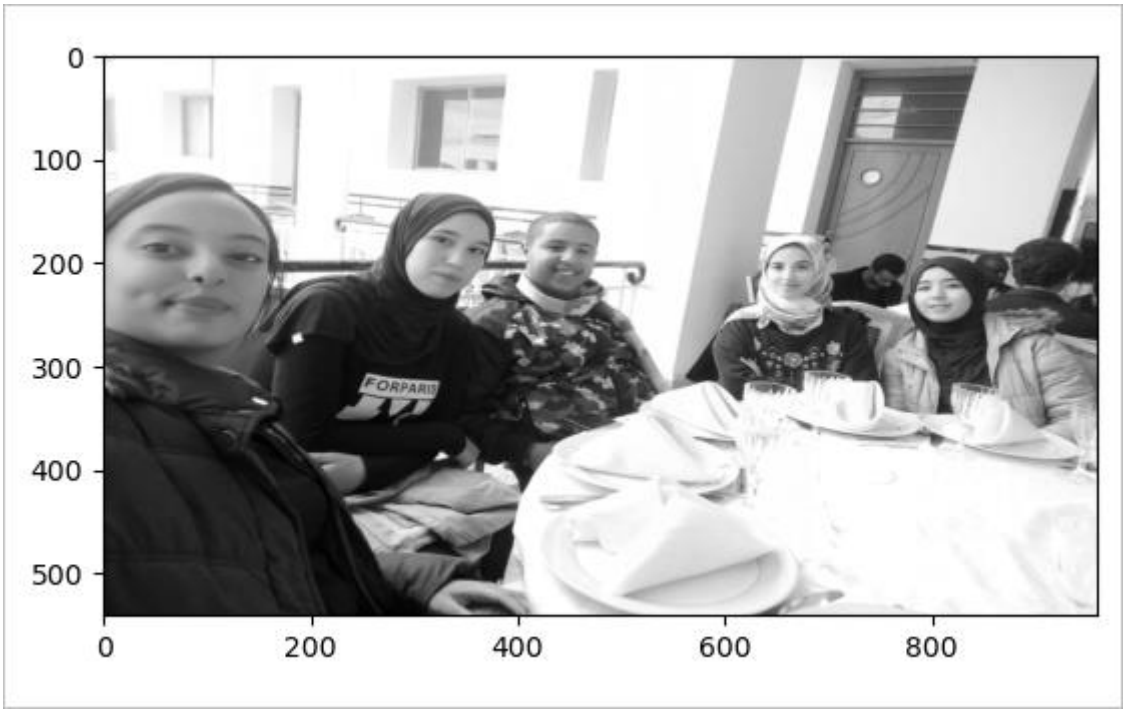
```
#conversion d'une image en niveaux de gris
grayScaleImage= cv2.cvtColor(originalmage, cv2.COLOR_BGR2GRAY)
ReSized2 = cv2.resize(grayScaleImage, (960, 540))
plt.imshow(ReSized2, cmap='gray')
```

- Explication :
  cvtColor (image, flag) est une méthode en cv2 qui est utilisée pour transformer une image dans l'espace colorimétrique mentionné comme «flag». Ici, notre première étape consiste à convertir l'image en niveaux de gris. Ainsi, nous utilisons le flag BGR2GRAY. Cela renvoie l'image en niveaux de gris. Une image en niveaux de gris est stockée en tant que grayScaleImage.

  Après chaque transformation, nous redimensionnons l'image résultante à l'aide de la méthode resize () dans cv2 et l'affiche à l'aide de la méthode imshow (). Ceci est fait pour

obtenir des informations plus claires sur chaque étape de transformation

- Output de ce code :



e) Étape 5: Lissage d'une image en niveaux de gris :

- Code :

```
#application d'un flou médian pour lisser une image
smoothGrayScale = cv2.medianBlur(grayScaleImage, 5)
ReSized3 = cv2.resize(smoothGrayScale, (960, 540))
plt.imshow(ReSized3, cmap='gray')
```

- Explication :

Pour lisser une image, nous appliquons simplement un effet de flou. Ceci est fait en utilisant la fonction medianBlur (). Ici, le pixel central se voit attribuer une valeur moyenne de tous les pixels qui tombent sous le noyau. À son tour, créant un effet de flou.

- Output :

f) Étape 6: Récupération des bords d'une image :

- Code :

```
#récupération des bords pour un effet de dessin animé en utilisant la technique de seuillage
getEdge = cv2.adaptiveThreshold(smoothGrayScale, 255,
    cv2.ADAPTIVE_THRESH_MEAN_C,
    cv2.THRESH_BINARY, 9, 9)

ReSized4 = cv2.resize(getEdge, (960, 540))
plt.imshow(ReSized4, cmap='gray')
```

- Explication :

L'effet de dessin animé a deux spécialités:

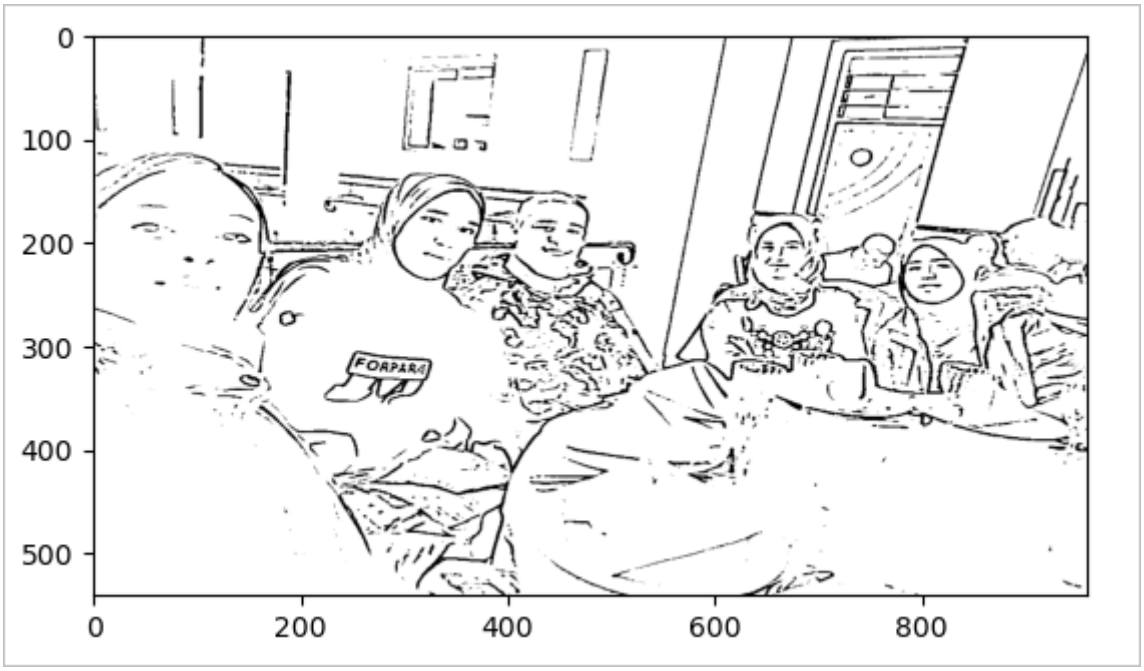
1. Bords en surbrillance
2. Couleurs lisses

Dans cette étape, nous travaillerons sur la première spécialité (Bords en surbrillance).

Nous allons essayer de récupérer les bords et de les mettre en évidence. Ceci est atteint par la technique de seuillage adaptatif. La valeur de seuil est la moyenne de la zone de valeurs de pixels de voisinage moins la constante C. C est une constante qui est soustraite de la somme moyenne ou pondérée des pixels de voisinage.

Thresh_binary est le type de seuil appliqué et les paramètres restants déterminent la taille du bloc.

- Output :



g) Étape 7: Préparation d'une image de masque :

- Code :

```
#application d'un filtre bilatéral pour éliminer le bruit et gardez les bords nets au besoin
colorImage = cv2.bilateralFilter(originalmage, 9, 300, 300)
ReSized5 = cv2.resize(colorImage, (960, 540))
plt.imshow(ReSized5, cmap='gray')
```

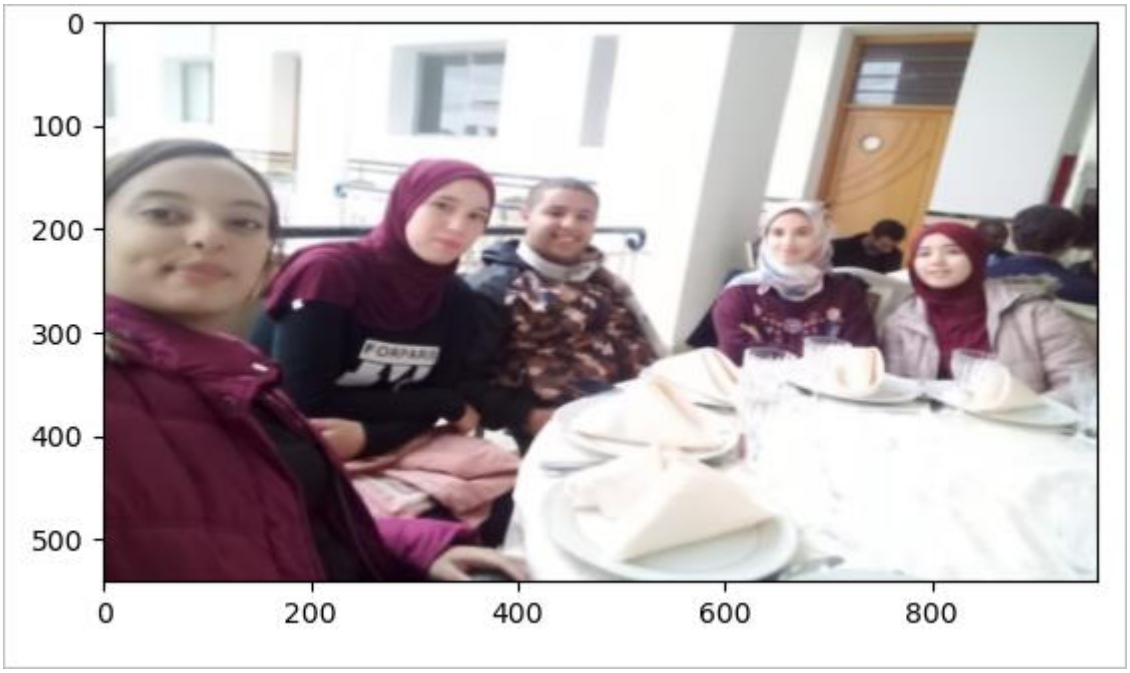
- Explication :

Dans le code ci-dessus, nous travaillons enfin sur la deuxième spécialité. Nous préparons une image couleur éclaircie que nous masquons avec des bords à la fin pour produire une image de dessin animé. Nous utilisons bilatéralFilter qui supprime le bruit. Cela peut être considéré comme un lissage d'une image dans une certaine mesure

Le troisième paramètre est le diamètre du voisinage des pixels. Le quatrième et le cinquième paramètre définissent signmaColor et sigmaSpace. Ces paramètres sont utilisés pour donner un effet sigma

Il est similaire à BEAUTIFY ou AI effect dans les caméras des téléphones mobiles modernes.

- Output du code ci-dessus:



h) Étape 8: Donner un effet de dessin animé

- Code :

```
#masquage de l'image bordée avec notre image "BEAUTIFY"
cartoonImage = cv2.bitwise_and(colorImage, colorImage, mask=getEdge)

ReSized6 = cv2.resize(cartoonImage, (960, 540))
plt.imshow(ReSized6, cmap='gray')
```

- Explication :

Alors, combinons les deux spécialités. Cela sera fait en utilisant MASKING. Nous effectuons au niveau du bit et sur deux images pour les masquer.

Les images ne sont que des chiffres, c'est ainsi que nous masquons une image bordée sur notre image "BEAUTIFY".

- Output du code ci-dessus :

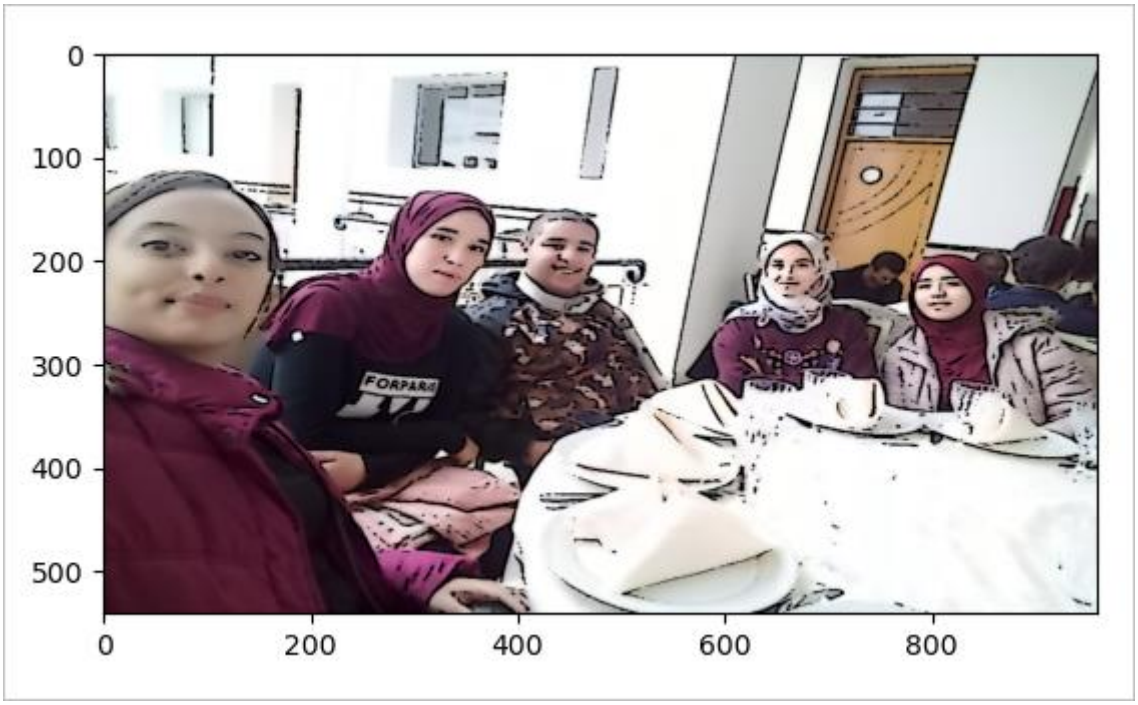


Figure : Résultat final

i) Étape 9: Tracer toutes les transitions ensemble

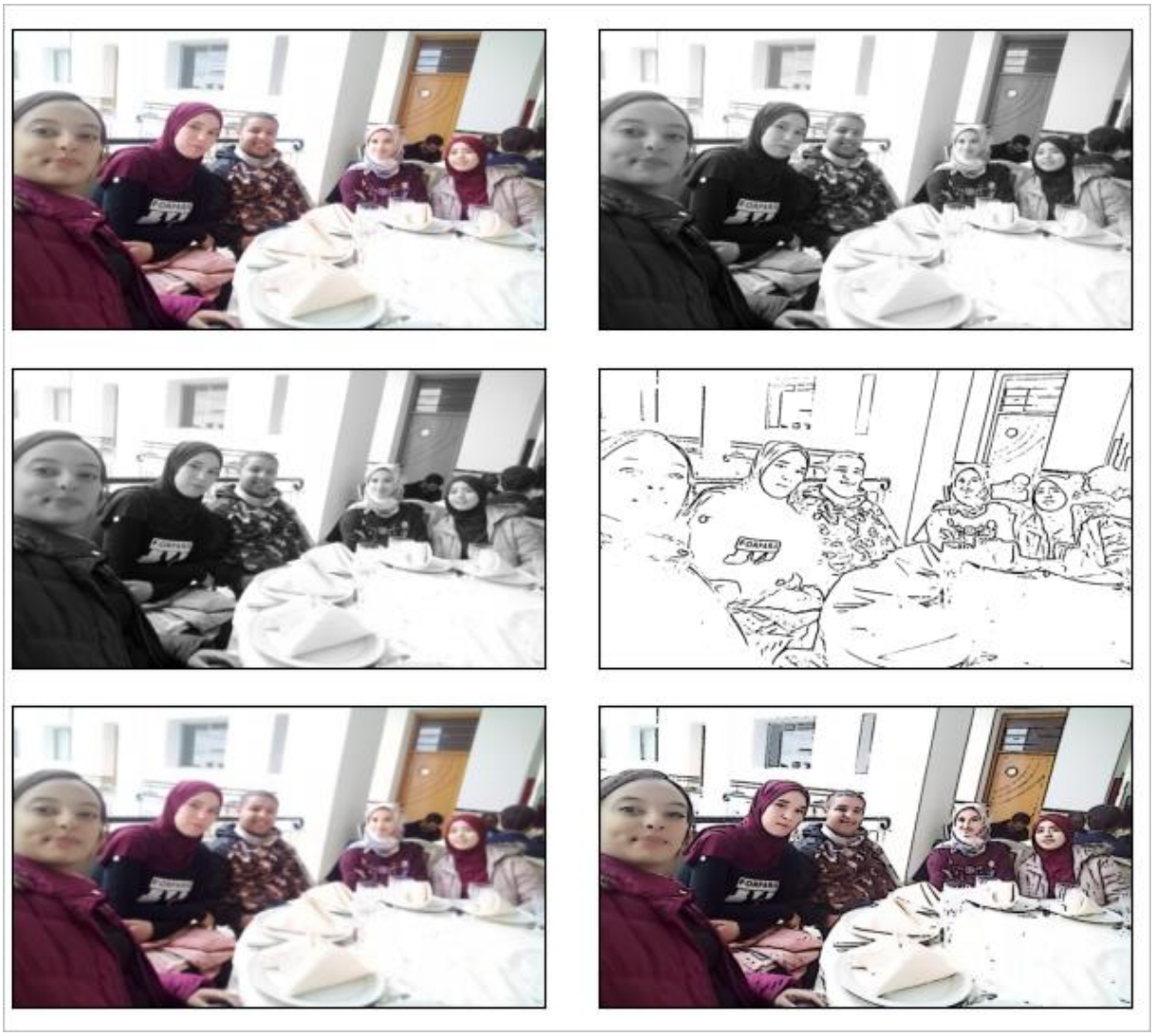
- Code :

```
# Tracer toute la transition
images=[ReSized1, ReSized2, ReSized3, ReSized4, ReSized5, ReSized6]

fig, axes = plt.subplots(3,2, figsize=(8,8), subplot_kw={'xticks':[], 'yticks':[]}, gridspec_kw=dict(hspace=0.1, wspace=0
for i, ax in enumerate(axes.flat):
    ax.imshow(images[i], cmap='gray')

plt.show()
```

- Explication :

Pour tracer toutes les images, nous faisons d'abord une liste de toutes les images. La liste ici est nommée «images» et contient toutes les images redimensionnées. Maintenant, nous créons des axes comme subl = plot dans un tracé et affichons une image dans chaque bloc sur l'axe en utilisant la méthode imshow ().

- Output :



j) Étape 10: Fonctionnellement du bouton Enregistrer

- Code :

```
def save(ReSized6, ImagePath):
    #enregistrement d'une image en utilisant imwrite ()
    newName="Image_caricaturée"
    path1 = os.path.dirname(ImagePath)
    extension=os.path.splitext(ImagePath)[1]
    path = os.path.join(path1, newName+extension)
    cv2.imwrite(path, cv2.cvtColor(ReSized6, cv2.COLOR_RGB2BGR))
    I= "Image enregistrée par nom " + newName +" dans "+ path
    tk.messagebox.showinfo(title='Abouche Hind', message=I)
```

- Explication :

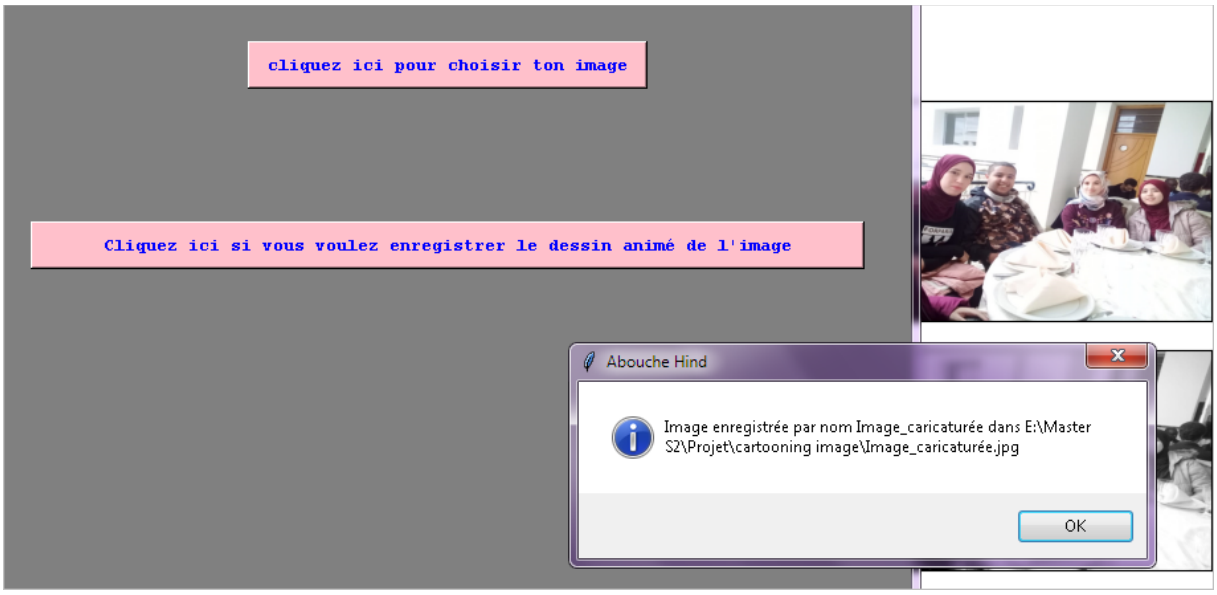
Ici, l'idée est de sauvegarder l'image résultante. Pour cela, nous prenons l'ancien chemin, et changeons simplement la queue (nom de l'ancien fichier) en un nouveau nom et stockons l'image caricaturée avec un nouveau nom dans le même dossier en ajoutant le nouveau nom à la partie tête du fichier

Pour cela, nous extrayons la partie head du chemin du fichier par la méthode os.path.dirname (). De même, os.path.splitext (ImagePath) [1] est utilisé pour extraire l'extension du fichier du chemin.

Ici, newName stocke «Image_caricaturée» comme nom d'un nouveau fichier. os.path.join (path1, newName + extension) rejoint la tête du chemin vers le nouveau nom et l'extension. Cela constitue le chemin complet du nouveau fichier.

La méthode imwrite () de cv2 est utilisée pour enregistrer le fichier au chemin mentionné. cv2.cvtColor (ReSized6, cv2.COLOR_RGB2BGR) est utilisé pour garantir qu'aucune couleur n'est extraite ou mise en surbrillance pendant que nous enregistrons notre image. Ainsi, l'utilisateur reçoit enfin la confirmation que l'image est enregistrée avec le nom et le chemin du fichier.

- Output :



k) Étape 11: Création de la fenêtre principale

- Code :

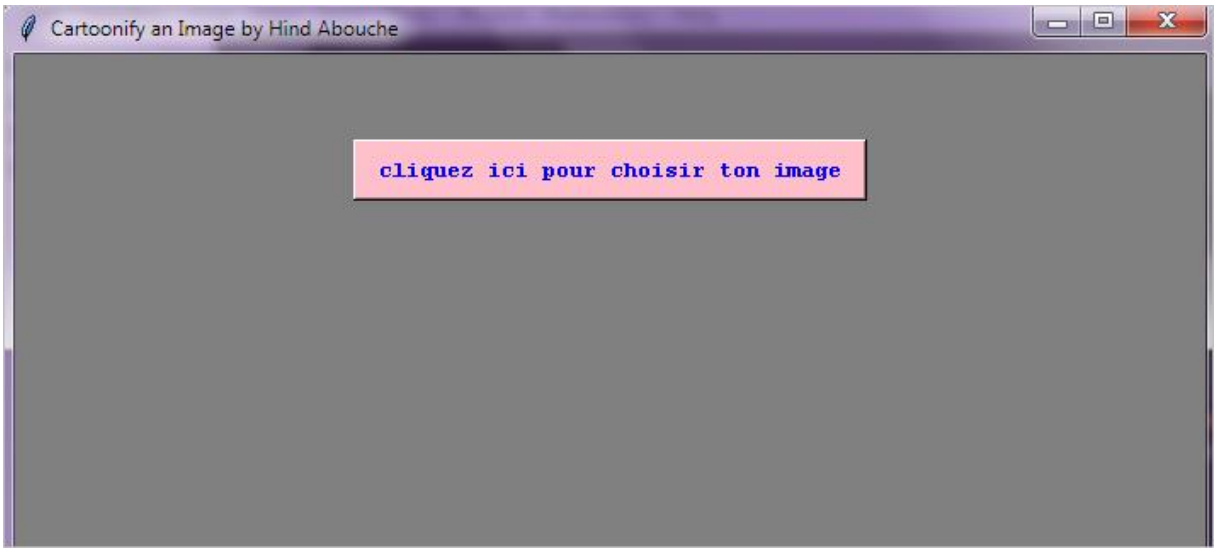
```
top=tk.Tk()
top.geometry('700x700')
top.title('Cartoonify an Image by Hind Abouche')
top.configure(background='gray')
label=Label(top,background='gray', font=('Courier New',40,'bold'))
```

l) Étape 12: Création un bouton pour choisir une image dans la fenêtre principale

- Code :

```
upload=Button(top,text="cliquez ici pour choisir ton image",command=upload,padx=10,pady=5)
upload.configure(background='pink', foreground='blue',font=('Courier New',10,'bold'))
upload.pack(side=TOP,pady=50)
```

- Output :

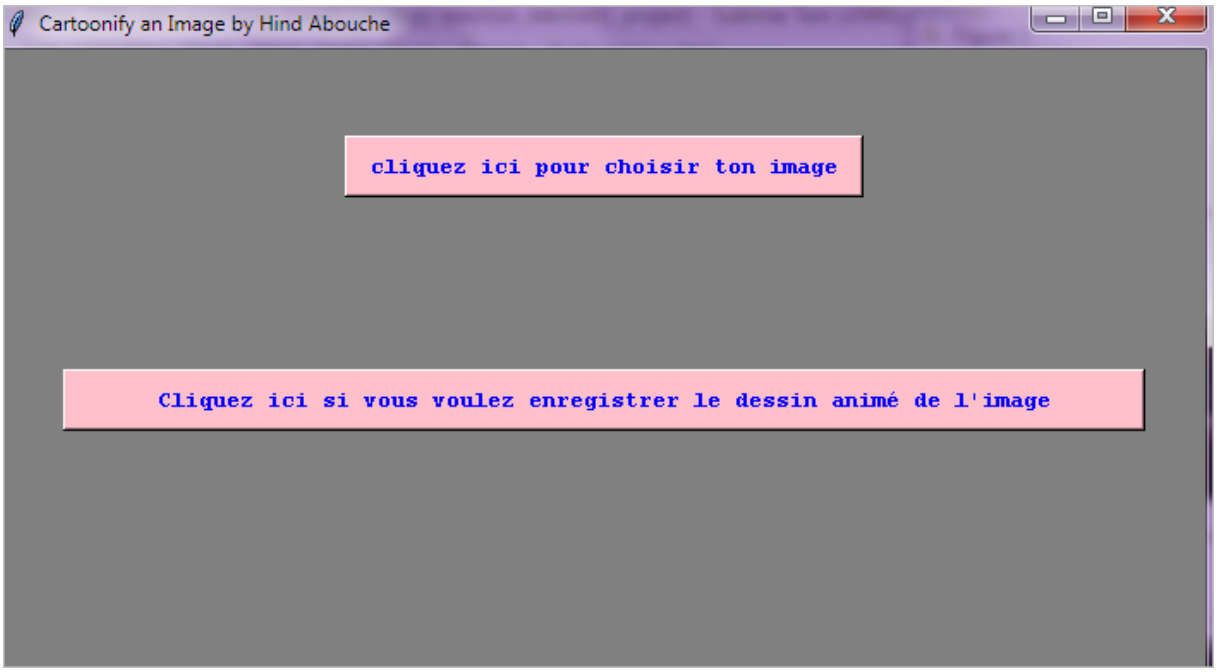


m) Étape 13: Créer un bouton Enregistrer dans la fenêtre principale

- Code :

```
save1=Button(top,text="Cliquez ici si vous voulez enregistrer le dessin animé de l'image",command=lambda: save(ReSize
save1.configure(background='pink', foreground='blue',font=('Courier New',10,'bold'))
save1.pack(side=TOP,pady=50)
```

- Output :



n) Étape 14: Fonction principale pour créer la fenêtre tkinter

```
top.mainloop()
```

3.3 Résultat final :

4 Conclusion

Deep Learning a ajouté un énorme coup de pouce au domaine déjà en développement rapide de la vision par ordinateur. Avec le Deep Learning, de nombreuses nouvelles applications des techniques de vision par ordinateur ont été introduites et font désormais partie de notre vie quotidienne. Il s'agit notamment de la reconnaissance faciale et de l'indexation, de la stylisation des photos ou de la vision industrielle dans les voitures autonomes

Dans ce mini-projet nous avons transformé les images en son dessin animé. Ainsi, nous avons créés une application python qui transforme une image en son dessin animé en utilisant OpenCV. C’est un projet qui reste tout de même basique par rapport aux différentes applications disponibles de nos jours. On trouve parmi ces applications « age and gender detection », « face recognition » etc.

Référence :

1. https://www.lebigdata.fr/deep-learning-definition
2. https://www.oracle.com/fr/artificial-intelligence/deep-learning-machine-learning-intelligence-artificielle.html
3. https://riptutorial.com/fr/opencv/example/23242/quoi-et-pourquoi-opencv-
4. https://www.coursera.org/learn/deep-learning-in-computer-vision
5. https://justaskthales.com/fr/quest-ce-que-lintelligence-artificielle-et-le-machine-learning/
6. https://naadispeaks.wordpress.com/2018/08/12/deep-learning-vs-traditional-computer-vision/#:~:text=The%20Deep%20Learning%20approach%20%E2%80%93&text=The%20main%20difference%20in%20deep,simply%20put%20in%20this%20way.