UNIVERSITÉ
DE REIMS
CHAMPAGNE-ARDENNE

Compte-rendu

Réalisé par :

MOALI Sarah

J’ai utilisé Keras pour construire et entraîner le modèle de segmentation U-NET.

```
!pip install --upgrade git+https://github.com/divamgupta/image-segmentation-keras
```

Préparation de la dataset :

La première étape de la formation du modèle de segmentation consiste à préparer l'ensemble de données.

On a besoin des images RGB d'entrée et des images de segmentation en générant manuellement les masques de segmentation de la vérité terrain à l’aide de l’outil Labelme.

J’ai dû trier la dataset et prendre que quelques images et leurs masks.

- **Redimensionnement de la taille des images :**

La taille de l'image d'entrée et de l'image de segmentation doit être la même

```
from PIL import Image
import os


SIZE_X = 400 #Resize images (height  = X, width = Y)
SIZE_Y = 400
l_mask=sorted(os.listdir('/content/drive/MyDrive/test_mask'))

for i in l_mask:

        img='/content/drive/MyDrive/test_mask/'+i
        image = Image.open(img)
        new_image= image.resize((SIZE_X, SIZE_Y))
        os.remove(img)
        new_image.save(img)


#redimensionnement des images de train

l_images=sorted(os.listdir('/content/drive/MyDrive/images'))

for i in l_images:

        img='/content/drive/MyDrive/images/'+i
        image = Image.open(img)
        new_image= image.resize((SIZE_X, SIZE_Y))
        os.remove(img)
        new_image.save(img)
```

```
#redimensionnement des images de test
l_image_t=sorted(os.listdir('/content/drive/MyDrive/test_images'))

for i in l_image_t:

        img='/content/drive/MyDrive/test_images/'+i
        image = Image.open(img)
        new_image= image.resize((SIZE_X, SIZE_Y))
        os.remove(img)
        new_image.save(img)



l_maskT=sorted(os.listdir('/content/drive/MyDrive/masks'))

for i in l_maskT:

        img='/content/drive/MyDrive/masks/'+i
        image = Image.open(img)
        new_image= image.resize((SIZE_X, SIZE_Y))
        os.remove(img)
        new_image.save(img)
```

- **La mise en mode gris des masks :**

Chaque pixel aura une valeur entre 0 et n, correspondant au nombre de classe total présent au sein de notre dataset

On a 3 classes une classe pour la feuille une autre pour la maladie et la troisième classe est pour le background.

```
# 3 pixels chaque pixel représente une classe
import numpy as np
from cv2 import imread, imwrite
import os
file=sorted(os.listdir('/content/drive/MyDrive/test_mask/'))

for image in file:
        s="/content/drive/MyDrive/test_mask/"+image
        #charger une image en mode gris
        img = imread(s, 0)
        v = np.zeros((img.shape[0], img.shape[1], 3))
        for (i, val) in enumerate(np.unique(img)):
            v[:,:,:][img == val] = i
        path_des='/content/drive/MyDrive/test_mask/'+image

        imwrite(path_des, v)
```

J’ai choisi le modèle de segmentation U-NET :

Je voulais faire une comparaison entre un modèle unet avec VGG16 comme encodeur et un modèle unet avec RestNet50 comme encodeur mais je n’avais pas eu assez de temps vu que ça consomme de la mémoire GPU et le modèle prend également plus de temps à s'entraîner.

- **Création du modèle :**

J’ai importé le modèle U-net prêts à l'emploi à partir de keras_segmentation.

```
from keras_segmentation.models.unet import unet
model = unet(n_classes=3 ,input_height=384, input_width=384)
```

- **Entrainement du modèle :**

Après avoir préparé l'ensemble de données et construit le modèle, nous devons entraîner le modèle.

```
model.train(
    train_images =  "/content/drive/MyDrive/data/images/",
    train_annotations = "/content/drive/MyDrive/data/masksgrey/",
    epochs=10)
```

```
Verifying training dataset
100%|██████████| 307/307 [03:54<00:00,  1.31it/s]
Dataset verified!
Epoch 1/10
512/512 [==============================] - 3116s 6s/step - loss: 0.9923 - accuracy: 0.5609
Epoch 2/10
512/512 [==============================] - 3118s 6s/step - loss: 0.8276 - accuracy: 0.6346
Epoch 3/10
512/512 [==============================] - 3096s 6s/step - loss: 0.7852 - accuracy: 0.6572
Epoch 4/10
512/512 [==============================] - 3091s 6s/step - loss: 0.7602 - accuracy: 0.6775
Epoch 5/10
512/512 [==============================] - 3103s 6s/step - loss: 0.7268 - accuracy: 0.6952
Epoch 6/10
512/512 [==============================] - 3094s 6s/step - loss: 0.6907 - accuracy: 0.7127
Epoch 7/10
512/512 [==============================] - 3097s 6s/step - loss: 0.6953 - accuracy: 0.7152
Epoch 8/10
512/512 [==============================] - 3105s 6s/step - loss: 0.6655 - accuracy: 0.7279
Epoch 9/10
512/512 [==============================] - 3144s 6s/step - loss: 0.6368 - accuracy: 0.7414
Epoch 10/10
512/512 [==============================] - 3150s 6s/step - loss: 0.6453 - accuracy: 0.7407
```

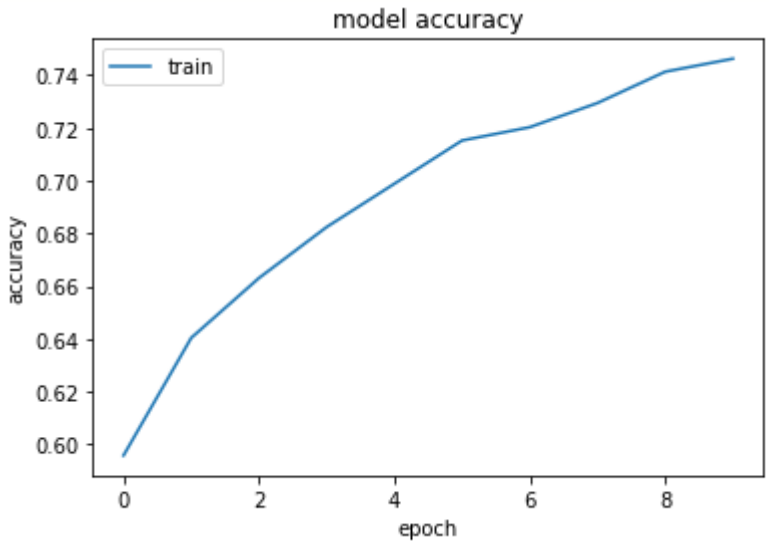
- **Création du modèle unet avec l’encodeur VGG16 en l’important de keras_segmentation:**

```
from keras_segmentation.models.unet import vgg_unet
model = vgg_unet(n_classes=3 , input_height=384, input_width=384)
```

J’ai utilisé l'enregistrement **history** pour obtenir des graphiques des mesures de précision et de perte.

Le code suivant tracera la précision à chaque époque

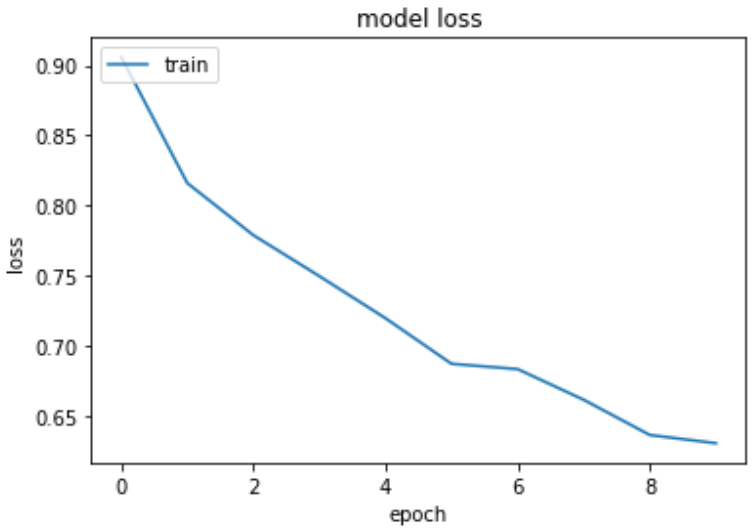
```
print(model.history.history.keys())
plt.plot(model.history.history['accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



Comme on peut le voir sur le diagramme, la précision augmente à l’augmentation des epochs, indiquant que le modèle a une bonne prédiction.

Le code suivant tracera la perte à chaque époque

```
# summarize history for loss
plt.plot(model.history.history['loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



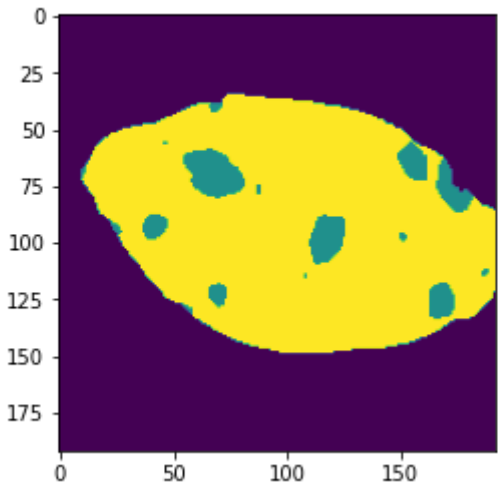
Comme on peut le voir sur le diagramme de perte sur les données d'entrainement, la perte sur l'ensemble d'apprentissage diminue avec l'augmentation des epochs. Si on augmente le nombre d'epoch on aura une valeur de loss qui s'approche à zéro.

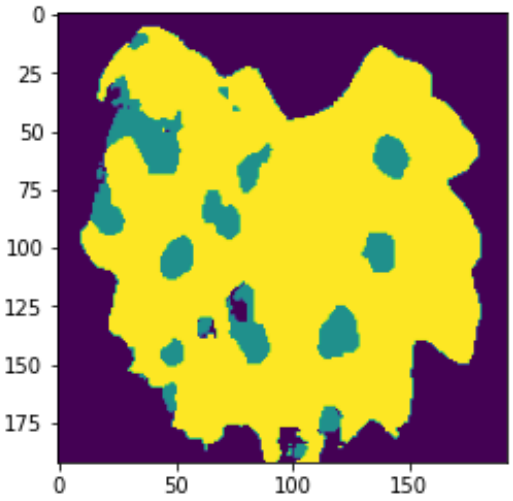
```
#voir la sortie du modèle sur une nouvelle image qui n'est pas présente
 dans l'ensemble d'apprentissage

out = model.predict_segmentation(
    inp="/content/drive/MyDrive/data/image.jpg",
    out_fname="/content/drive/MyDrive/data/out_image.png"
)

import matplotlib.pyplot as plt
plt.imshow(out)
```

```
<matplotlib.image.AxesImage at 0x7f8ecae3ec10>
```





Conclusion :

On remarque que le modèle est performant, il a bien généralisé sur les données test avec seulement 10 epochs (il a bien prédit le mask de la nouvelle image qui n'est pas présente dans l'ensemble d'apprentissage).