Classification des images histopathologiques de cancer:Using DeepLearning CNN

Réaliser par:

- EL MOUKHTAR ABDELMALEK

Dataset

Le cancer du sein est la forme la plus courante de cancer chez les femmes, et le carcinome canalaire invasif (IDC) est la forme la plus courante de cancer du sein. L'identification et la catégorisation précises des sous-types de cancer du sein est une tâche clinique importante, et des méthodes automatisées peuvent être utilisées pour gagner du temps et réduire les erreurs.

1. Install the Kaggle library

```
! pip install -q kaggle
```

```
from google.colab import files
uploaded = files.upload()
```

6

3. Make a directory named kaggle

```
! mkdir ~/.kaggle
```

4. Copy the “kaggle.json” into this new directory

```
! cp kaggle.json ~/.kaggle/

```

5. Allocate the required permission for this file.

```
! chmod 600 ~/.kaggle/kaggle.json
```

6-Download datasets

```
#downlod dataset
! kaggle datasets download -d paultimothymooney/breast-histopathology-images
```

```
Downloading breast-histopathology-images.zip to /content
100% 3.10G/3.10G [00:49<00:00, 96.7MB/s]
100% 3.10G/3.10G [00:49<00:00, 67.6MB/s]
```

7-Extract the data:

```
! unzip breast-histopathology-images.zip -d breast_cancer_images
```

Le flux de sortie a été tronqué et ne contient que les 5000 dernières lignes.

```
  inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2351_
  inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2351_
  inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2351_
  inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2351_
  inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2351_
  inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2351_
  inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2351_
  inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2351_
  inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2351_
  inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2351_
  inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2351_
  inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2351_
  inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2351_
  inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2351_
  inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2401_
  inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2401_
  inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2401_
  inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2401_
  inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2401_
  inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2401_
  inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2401_
  inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2401_
  inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2401_
  inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2401_
  inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2401_
  inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2401_
```

```
inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2401_
inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2401_
inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2401_
inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2401_
inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2401_
inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2401_
inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2451_
inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2451_
inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2451_
inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2451_
inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2451_
inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2451_
inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2451_
inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2451_
inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2451_
inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2451_
inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2501_
inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2501_
inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2501_
inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2501_
inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2501_
inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2501_
inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2501_
inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x2501_
inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x251_y
inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x251_y
inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x251_y
inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x251_y
inflating: breast_cancer_images/IDC_regular_ps50_idx5/9346/0/9346_idx5_x251_y
```

```
import pandas as pd
import numpy as np
import os
from glob import glob
import random
import matplotlib.pylab as plt
import keras.backend as K
from sklearn.model_selection import train_test_split
import tensorflow as tf
import keras
from keras.utils.np_utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, MaxPool2D, Max
%matplotlib inline
```

```
try:
    tpu = tf.distribute.cluster_resolver.TPUClusterResolver()
    print('Running on TPU ', tpu.master())
except ValueError:
    tpu = None

if tpu:
    tf.config.experimental_connect_to_cluster(tpu)
    tf.tpu.experimental.initialize_tpu_system(tpu)
```

```
    strategy = tf.distribute.experimental.TPUStrategy(tpu)
else:
    strategy = tf.distribute.get_strategy() # default distribution strategy in Tens

print("REPLICAS: ", strategy.num_replicas_in_sync)
```

```
Running on TPU  grpc://10.8.46.130:8470
INFO:tensorflow:Deallocate tpu buffers before initializing tpu system.
INFO:tensorflow:Deallocate tpu buffers before initializing tpu system.
INFO:tensorflow:Initializing the TPU system: grpc://10.8.46.130:8470
INFO:tensorflow:Initializing the TPU system: grpc://10.8.46.130:8470
INFO:tensorflow:Finished initializing TPU system.
INFO:tensorflow:Finished initializing TPU system.
WARNING:absl:`tf.distribute.experimental.TPUStrategy` is deprecated, please use
INFO:tensorflow:Found TPU system:
INFO:tensorflow:Found TPU system:
INFO:tensorflow:*** Num TPU Cores: 8
INFO:tensorflow:*** Num TPU Cores: 8
INFO:tensorflow:*** Num TPU Workers: 1
INFO:tensorflow:*** Num TPU Workers: 1
INFO:tensorflow:*** Num TPU Cores Per Worker: 8
INFO:tensorflow:*** Num TPU Cores Per Worker: 8
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:localhost/replica:0/
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:localhost/replica:0/
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/tas
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/tas
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/tas
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/tas
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/tas
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/tas
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/tas
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/tas
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/tas
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/tas
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/tas
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/tas
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/tas
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/tas
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/tas
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/tas
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/tas
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/tas
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/tas
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/tas
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/tas
REPLICAS:  8
```

```
import numpy as np
import glob
import random
import warnings
warnings.filterwarnings(action = 'ignore')
import matplotlib.pyplot as plt

from PIL import Image
```

```

random.seed(98)
np.random.seed(98)
```

```
breast_img = glob.glob('/breast-histopathology-images/IDC_regular_ps50_idx5/**/*.png

for imgname in breast_img[:3]:
    print(imgname)

```

```
imagePatches = glob.glob("breast_cancer_images/**/*.png", recursive=True)
for filename in imagePatches[0:10]:
    print(filename)

```

```
breast_cancer_images/13458/0/13458_idx5_x851_y1001_class0.png
breast_cancer_images/13458/0/13458_idx5_x801_y1001_class0.png
breast_cancer_images/13458/0/13458_idx5_x151_y1001_class0.png
breast_cancer_images/13458/0/13458_idx5_x1101_y1351_class0.png
breast_cancer_images/13458/0/13458_idx5_x1301_y1351_class0.png
breast_cancer_images/13458/0/13458_idx5_x501_y1051_class0.png
breast_cancer_images/13458/0/13458_idx5_x251_y851_class0.png
breast_cancer_images/13458/0/13458_idx5_x801_y901_class0.png
breast_cancer_images/13458/0/13458_idx5_x1101_y1201_class0.png
breast_cancer_images/13458/0/13458_idx5_x1301_y1401_class0.png
```

```
non_img = []
can_img = []

for img in imagePatches:
    if img[-5] == '0' :
        non_img.append(img)

    elif img[-5] == '1' :
        can_img.append(img)
```

```
non_num = len(non_img)
can_num = len(can_img)

total_img_num = non_num + can_num

print('Number of Images in IDC (-): {}' .format(non_num))
print('Number of Images in IDC (+) : {}' .format(can_num))
print('Total Number of Images : {}' .format(total_img_num))
```

```
Number of Images in IDC (-): 397476
Number of Images in IDC (+) : 157572
Total Number of Images : 555048
```

```
from keras.preprocessing import image
```

```

plt.figure(figsize = (15, 15))

some_non = np.random.randint(0, len(non_img), 18)
some_can = np.random.randint(0, len(can_img), 18)

s = 0
for num in some_non:

        img = image.load_img((non_img[num]), target_size=(100, 100))
        img = image.img_to_array(img)

        plt.subplot(6, 6, 2*s+1)
        plt.axis('off')
        plt.title('IDC (-)')
        plt.imshow(img.astype('uint8'))
        s += 1

s = 1
for num in some_can:

        img = image.load_img((can_img[num]), target_size=(100, 100))
        img = image.img_to_array(img)

        plt.subplot(6, 6, 2*s)
        plt.axis('off')
        plt.title('IDC (+)')
        plt.imshow(img.astype('uint8'))
        s += 1
```

```
from matplotlib.image import imread
import cv2

some_non_img = random.sample(non_img, len(can_img))
some_can_img = random.sample(can_img, len(can_img))

non_img_arr = []
can_img_arr = []

for img in some_non_img:

    n_img = cv2.imread(img, cv2.IMREAD_COLOR)
    n_img_size = cv2.resize(n_img, (50, 50), interpolation = cv2.INTER_LINEAR)
    non_img_arr.append([n_img_size, 0])

for img in some_can_img:

    c_img = cv2.imread(img, cv2.IMREAD_COLOR)
    c_img_size = cv2.resize(c_img, (50, 50), interpolation = cv2.INTER_LINEAR)
    can_img_arr.append([c_img_size, 1])

```

```
X = []
y = []

breast_img_arr = np.concatenate((non_img_arr, can_img_arr))
random.shuffle(breast_img_arr)

for feature, label in breast_img_arr:
    X.append(feature)
```

```
    y.append(label)

X = np.array(X)
y = np.array(y)

print('X shape : {}' .format(X.shape))
```

```
X shape : (315144, 50, 50, 3)
```

```
from sklearn.model_selection import train_test_split
from keras.utils.np_utils import to_categorical

X_train, X_predict, y_train, y_true = train_test_split(X, y, test_size = 0.3, random

rate = 0.5
num = int(X.shape[0] * rate)

X_test = X_train[num:]
X_train = X_train[:num]

y_test = y_train[num:]
y_train = y_train[:num]

y_train = to_categorical(y_train, 2)
y_test = to_categorical(y_test, 2)
y_true = to_categorical(y_true, 2)

print('X_train shape : {}' .format(X_train.shape))
print('X_test shape : {}' .format(X_test.shape))
print('X_predict shape : {}' .format(X_predict.shape))
print('y_train shape : {}' .format(y_train.shape))
print('y_test shape : {}' .format(y_test.shape))
print('y_true shape : {}' .format(y_true.shape))
```

```
X_train shape : (157572, 50, 50, 3)
X_test shape : (63028, 50, 50, 3)
X_predict shape : (94544, 50, 50, 3)
y_train shape : (157572, 2)
y_test shape : (63028, 2)
y_true shape : (94544, 2)
```

```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense

model = Sequential()

model.add(Conv2D(32, (3, 3), padding = 'same', activation = 'relu', input_shape = (5
model.add(MaxPooling2D(2, 2))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), padding = 'same', activation = 'relu', input_shape = (5
model.add(MaxPooling2D(2, 2))
```

```
model.add(Dropout(0.25))

model.add(Conv2D(128, (3, 3), padding = 'same', activation = 'relu', input_shape = (
model.add(MaxPooling2D(2, 2))
model.add(Dropout(0.25))

model.add(Conv2D(128, (3, 3), padding = 'same', activation = 'relu', input_shape = (
model.add(MaxPooling2D(2, 2))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(128, activation = 'relu'))

model.add(Dropout(0.5))
model.add(Dense(2, activation = 'sigmoid'))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 50, 50, 32)        896       
                                                                 
 max_pooling2d (MaxPooling2D  (None, 25, 25, 32)       0         
 )                                                               
                                                                 
 dropout (Dropout)           (None, 25, 25, 32)        0         
                                                                 
 conv2d_1 (Conv2D)           (None, 25, 25, 64)        18496     
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 12, 12, 64)       0         
 2D)                                                             
                                                                 
 dropout_1 (Dropout)         (None, 12, 12, 64)        0         
                                                                 
 conv2d_2 (Conv2D)           (None, 12, 12, 128)       73856     
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 6, 6, 128)        0         
 2D)                                                             
                                                                 
 dropout_2 (Dropout)         (None, 6, 6, 128)         0         
                                                                 
 conv2d_3 (Conv2D)           (None, 6, 6, 128)         147584    
                                                                 
 max_pooling2d_3 (MaxPooling  (None, 3, 3, 128)        0         
 2D)                                                             
                                                                 
 dropout_3 (Dropout)         (None, 3, 3, 128)         0         
                                                                 
 flatten (Flatten)           (None, 1152)              0         
                                                                 
 dense (Dense)               (None, 128)               147584    
                                                                 
 dropout_4 (Dropout)         (None, 128)               0         
                                                                 
 dense_1 (Dense)             (None, 2)                 258       
```

```
=================================================================
Total params: 388,674
Trainable params: 388,674
Non-trainable params: 0
_________________________________________________________________
```

```
#from keras.optimizers import Adam
from tensorflow.keras.optimizers import Adam


Adam = Adam(learning_rate = 0.0001)
model.compile(loss = 'binary_crossentropy', optimizer = Adam, metrics = ['accuracy']
```

```
history = model.fit(X_train, y_train, validation_data = (X_test, y_test), epochs = 1
```

```
Epoch 1/15
3152/3152 [==============================] - 127s 40ms/step - loss: 0.6002 - accu
Epoch 2/15
3152/3152 [==============================] - 116s 37ms/step - loss: 0.3139 - accu
Epoch 3/15
3152/3152 [==============================] - 115s 37ms/step - loss: 0.2935 - accu
Epoch 4/15
3152/3152 [==============================] - 114s 36ms/step - loss: 0.2818 - accu
Epoch 5/15
3152/3152 [==============================] - 115s 37ms/step - loss: 0.2728 - accu
Epoch 6/15
3152/3152 [==============================] - 114s 36ms/step - loss: 0.2657 - accu
Epoch 7/15
3152/3152 [==============================] - 115s 37ms/step - loss: 0.2611 - accu
Epoch 8/15
3152/3152 [==============================] - 115s 37ms/step - loss: 0.2580 - accu
Epoch 9/15
3152/3152 [==============================] - 113s 36ms/step - loss: 0.2545 - accu
Epoch 10/15
3152/3152 [==============================] - 113s 36ms/step - loss: 0.2515 - accu
Epoch 11/15
3152/3152 [==============================] - 114s 36ms/step - loss: 0.2485 - accu
Epoch 12/15
3152/3152 [==============================] - 115s 37ms/step - loss: 0.2458 - accu
Epoch 13/15
3152/3152 [==============================] - 114s 36ms/step - loss: 0.2433 - accu
Epoch 14/15
3152/3152 [==============================] - 114s 36ms/step - loss: 0.2406 - accu
Epoch 15/15
3152/3152 [==============================] - 113s 36ms/step - loss: 0.2385 - accu
```

```
model.save("model_TP.h5")
```

```
P = model.predict(X_predict)

true = 0
for i in range(X_predict.shape[0]):
    if(np.argmax(P[i]) == np.argmax(y_true[i])):
```

```
        true = true + 1

pre_accuracy = 100 * float(true/X_predict.shape[0])
print('Predict Accuracy: {}' .format(pre_accuracy))
```

```
Predict Accuracy: 90.53245049923845
```

```
result = model.evaluate(X_test, y_test, batch_size = 50)
print('Test Loss, Test Accuracy :', result)
```

```
1261/1261 [==============================] - 19s 15ms/step - loss: 0.2761 - accur
Test Loss, Test Accuracy : [0.2760542035102844, 0.9058672189712524]
```

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

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.xlabel('epoch')
plt.ylabel('loss')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```