

Nom Prénom 1 : .....

Nom Prénom 2 : .....

Nom Prénom 3 : .....

Nom Prénom 4 : .....

**Intelligence Artificielle pour l'Image**  
**Challenge IA & classification d'images**  
**Projet**  
**Python et OpenCV**  
**E5FI\_5I-IN7**

## 1. Introduction

### 1.1 Objectif

L'objectif de ce Projet est de développer un ensemble de classificateurs supervisés Machine Learning et Deep Learning :

- Chargement d'une base d'images,
- Entrainement d'un modèle de classification supervisée à l'aide d'une base d'apprentissage,
- Validation et test du modèle entraîné sur un nouveau jeu d'images,
- Evaluation des résultats obtenus et amélioration des performances si besoin.

*Recommandation : Fei-Fei Li Computer scientist TED video :*

[https://www.ted.com/talks/fei\\_fei\\_li\\_how\\_we\\_re\\_teaching\\_computers\\_to\\_understand\\_pictures?language](https://www.ted.com/talks/fei_fei_li_how_we_re_teaching_computers_to_understand_pictures?language)

Vous trouvez en ce qui suit des indications et exemples de codes.

### 1.2 Datasets

Trois bases d'images seront utilisées pour la réalisation de ce projet ;

- La base MNIST / DGITS :

```
from keras.datasets import mnist
# the data, shuffled and split between train and test sets
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

- Deux bases Kaggle\_ImageNet :
  - o La première contient 10000 images réparties sur deux classes « Dogs&Cats » :  
<https://www.kaggle.com/chetankv/dogs-cats-images>
  - o et la deuxième contient 25000 images réparties sur plusieurs classes « Intel Image Classification ; Image Scene Classification of Multiclass » :  
<https://www.kaggle.com/datasets/puneet6060/intel-image-classification>

Pour le chargement de ces bases, la commande `os.listdir` du package « os » sera utilisée

## 2. ML \_ Extraction du descripteur ORB

Tapez et exécutez le code suivant pour l'extraction du descripteur ORB :

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

```
orb=cv2.ORB_create()
im=cv2.imread( ???)
plt.imshow(im)

# Function for plotting keypoints def draw_keypoints(vis,
keypoints, color = (0, 255, 255)):    for kp in keypoints:
    x, y = kp.pt          plt.imshow(cv2.circle(vis,
(int(x), int(y)), 2, color))

# Plotting the keypoints kp =
orb.detect(im,None) kp, des =
orb.compute(im, kp)
img=draw_keypoints(im,kp)
```

Vous pouvez utiliser d'autres descripteurs de votre choix, que vous jugez plus adaptés pour une base d'images.

## 3. ML \_ Classification supervisée et non-supervisée

Voici des exemples de codes utiles pour la réalisation de votre projet :

### 3.1 SVM

```
# Construction du modèle from sklearn.svm import SVC #
Support Vector for Classification
```

```
#classifier = SVC(kernel = 'linear', random_state = 0) classifier
= SVC(kernel = 'rbf', random_state = 0) classifier.fit(X_train,
y_train)
```

```
# Faire de nouvelles prédictions y_pred
= classifier.predict(X_test)
```

## 3.2 Kmeans

```
import numpy as np import
cv2
```

```
#img = cv2.imread('OpenCV.PNG')
img = cv2.imread('lena.jpg') Z =
img.reshape((-1,3))
```

```
# convert to np.float32
Z = np.float32(Z)
```

```
# define criteria, number of clusters(K) and apply kmeans()
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0) K
= 10
ret,label,center=cv2.kmeans(Z,K,None,criteria,10,cv2.KMEANS_RANDOM_CENTERS)
```

```
# Now convert back into uint8, and make original image
center = np.uint8(center) res = center[label.flatten()]
res2 = res.reshape((img.shape))
```

```
cv2.imshow('Image',img)
cv2.imshow('res2',res2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Ce classifieur peut servir pour la construction du vocabulaire d'un sac de mot visuel

## 3.3 Sac de mots visuels modèle (BOW) « Bag of Word »

```
# Creating histogram of training image
from scipy.cluster.vq import vq
```

```
k=200
im_features=np.zeros((len(image_dataset),k),"float32") for
i in range(len(image_dataset)):
```

```

words,distance=vq(image_descriptor[i],voc) # Voc représente le dictionnaire préalablement #
construit
for w in words:
    im_features[i][w] += 1

```

## 4. DL \_ CNN « Convolutional\_Neural\_Networks »

```

# Importing libraries
from keras.models import Sequential
from keras.layers import Conv2D from
keras.layers import MaxPooling2D from
keras.layers import Flatten
from keras.layers import Dense

# Initialising the CNN
classifier = Sequential()

# Step 1 - Convolution
classifier.add(Conv2D(32, (3, 3), input_shape = (64, 64, 3), activation = 'relu'))

# Step 2 - Pooling
classifier.add(MaxPooling2D(pool_size = (2, 2)))

# Adding a second convolutional layer
classifier.add(Conv2D(32, (3, 3), activation = 'relu')) classifier.add(MaxPooling2D(pool_size
= (2, 2)))

# Step 3 - Flattening
classifier.add(Flatten())

# Step 4 - Full connection
classifier.add(Dense(units = 128, activation = 'relu')) classifier.add(Dense(units
= 1, activation = 'sigmoid'))

# Compiling the CNN
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

# .....

classifier.fit_generator(training_set,
steps_per_epoch = 8000,           epochs
= 25,
validation_data = test_set,
validation_steps = 2000)

# .....

```

## 5. Evaluation

```
# Accuracy accuracy=accuracy_score(true_classes,predict_classes)
print(accuracy)

# Matrice de confusion from sklearn.metrics
import confusion_matrix cm =
confusion_matrix(y_test, y_pred)
```