ROYAUME DU MAROC
UNIVERSITÉ ABDELMALEK ESSAÂDI
FACULTÉ DES SCIENCES ET TECHNIQUES
TANGER

# Analysis, Mining and Indexing in big multimedia systems

## Devoir 2 : Création d'un système d'indexation et de recherche d'images par le contenu.

**Réalisé par :**

**Ridouan EL AZZOUZI**

**Soufiane ALOUI EL IDRISSI**

**Encadré par :**

**Pr. M'hamed AIT KBIR**

# Table des matières

# Introduction

Dans ce devoir on va créer un system de détections des objets dans une image et leurs classifications avec un modèle pré-entrainé de l'algorithme YOLO3 puis on va faire une recherche aux images similaires à l'image requête d'après les objets qui il contienne, on utilisant la distances de hamming .

# Détection d’objets et classification

Qu'est-ce que la détection d'objets ?

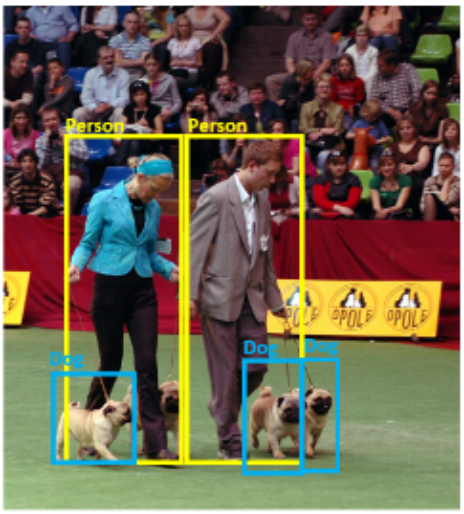
La détection d'objets est un problème de vision par ordinateur. Bien qu'étroitement liée à la classification d'images, la détection d'objets effectue une classification d'images à une échelle plus granulaire. La détection d'objets localise et catégorise les entités dans les images. Les modèles de détection d'objets sont généralement formés à l'aide de l'apprentissage en profondeur et des réseaux de neurones. Voir Apprentissage profond vs apprentissage automatique pour plus d'informations.

On utilise la détection d'objets lorsque les images contiennent plusieurs objets de types différents.

Image Classification

{Dog}

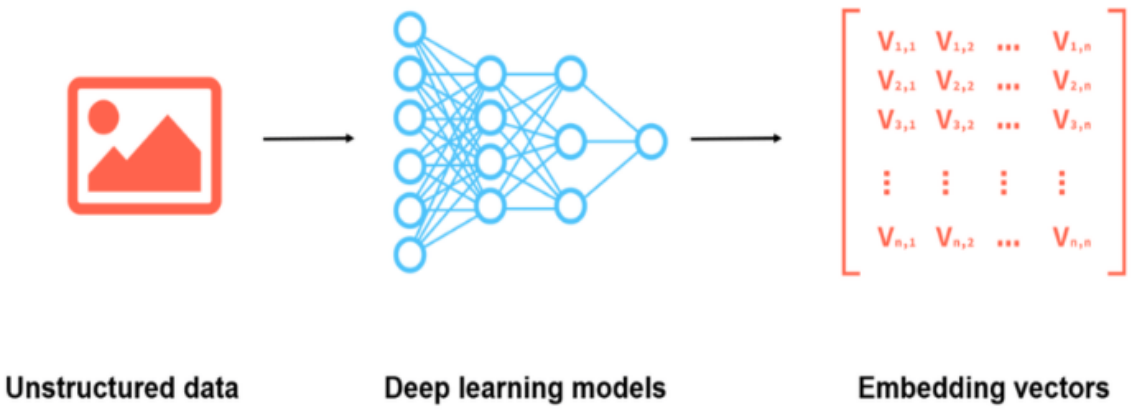
Object Detection




{Dog, Dog, Dog, Person, Person}

## Extraction de caractéristiques

L'extraction de caractéristiques fait référence à la conversion de données non structurées, difficiles à reconnaître pour les machines, en vecteurs de caractéristiques. Par exemple, les images peuvent être converties en vecteurs de caractéristiques multidimensionnels à l'aide de modèles d'apprentissage en profondeur. Actuellement, les modèles d'IA de reconnaissance d'images les plus populaires incluent VGG, GNN et ResNet. Cette rubrique explique comment extraire des fonctionnalités d'objets détectés à l'aide de ResNet-50.




## Analyse vectorielle

Les vecteurs de caractéristiques extraits sont comparés aux vecteurs de bibliothèque et les informations correspondantes aux vecteurs les plus similaires sont renvoyées. Pour les jeux de données vectorielles de caractéristiques à grande échelle, le calcul est un énorme défi.

# La technologie utilisée

## OpenCV


Open Source Computer Vision Library (OpenCV) est une bibliothèque de vision par ordinateur multiplateforme, qui fournit de nombreux algorithmes universels pour le traitement d'images et la vision par ordinateur. OpenCV est couramment utilisé dans le domaine de la vision par ordinateur.

## YOLOv3


You Only Look Once, Version 3 (YOLOv3 ) est un algorithme de détection d'objets en une étape proposé ces dernières années. Comparé aux algorithmes traditionnels de détection d'objets avec la même précision, YOLOv3 est deux fois plus rapide. YOLOv3 mentionné dans ce sujet est la version améliorée de PaddlePaddle . Il utilise plusieurs méthodes d'optimisation avec une vitesse d'inférence plus élevée.

## ResNet-50

ResNet est le lauréat de l'ILSVRC 2015 en classification d'images en raison de sa simplicité et de sa praticité. À la base de nombreuses méthodes d'analyse d'images, ResNet s'avère être un modèle populaire spécialisé dans la détection, la segmentation et la reconnaissance d'images.

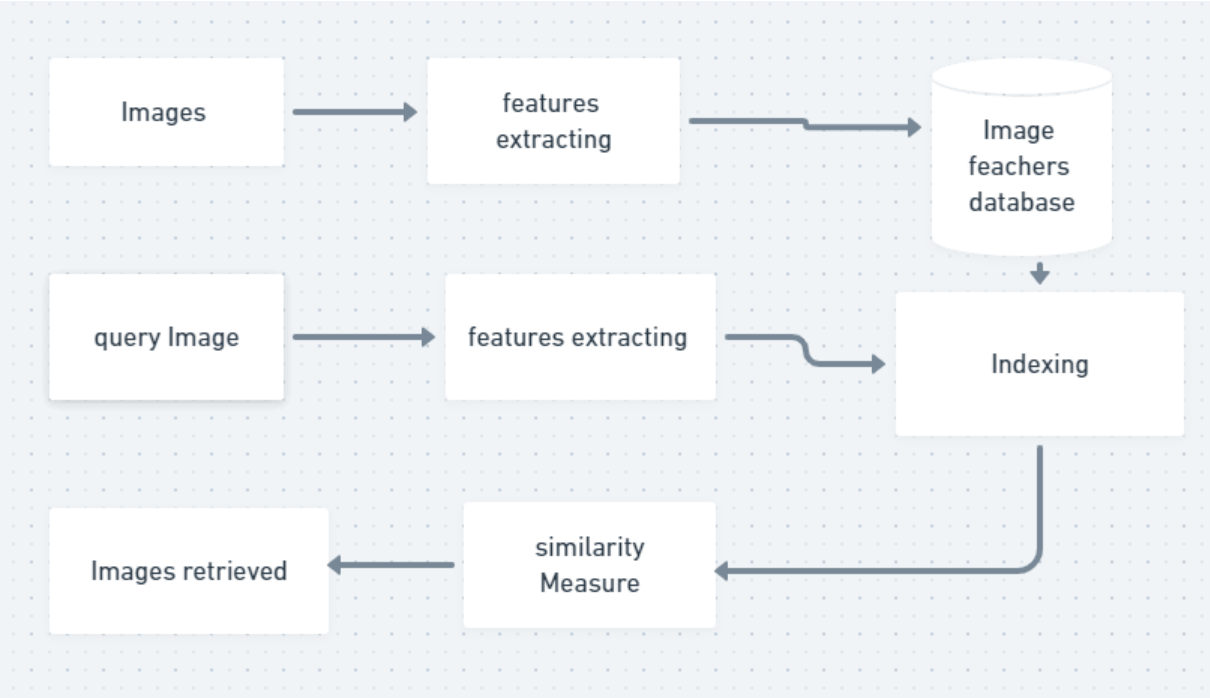
## Flask


Flask est un micro framework open-source de développement web en Python. Il est classé comme microframework car il est très léger. Flask a pour objectif de garder un noyau simple mais extensible

## Angular

Angular est un framework côté client, open source, basé sur TypeScript, et co-dirigé par l'équipe du projet « Angular » à Google et par une communauté de particuliers et de sociétés. Angular est une réécriture complète d'AngularJS, cadriciel construit par la même équipe.

# Architecture de l’application



# Distance de Hamming

**La distance de Hamming** est une notion mathématique, définie par Richard Hamming, et utilisée en informatique, en traitement du signal et dans les télécommunications. Elle joue un rôle important en théorie algébrique des codes correcteurs. Elle permet de quantifier la différence entre deux séquences de symboles. C'est une distance au sens mathématique du terme. À deux suites de symboles de même longueur, elle associe le nombre de positions où les deux suites diffèrent.

- Soit A un alphabet et F l'ensemble des suites de longueur n à valeur dans A. La distance de Hamming entre deux éléments a et b de F est le nombre d'éléments de l'ensemble des images de a qui diffèrent de celle de b.

Formellement, si d(.,.) désigne la distance de Hamming :

$$\forall a, b \in F \quad a = (a_i)_{i \in [0,n-1]} \; et \; b = (b_i)_{i \in [0,n-1]} \quad d(a,b) = \#\{i : a_i \neq b_i\}$$

La notation #E désigne le cardinal de l'ensemble E.

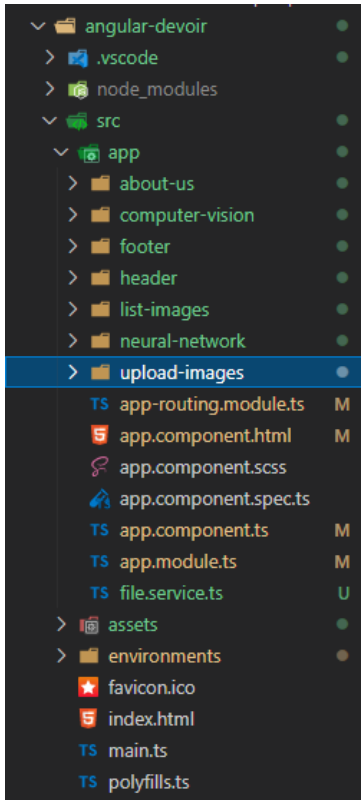
Un cas important dans la pratique est celui des symboles binaires. Autrement dit A= {0,1}, On peut alors écrire, si $\oplus$ désigne le ou exclusif.

$$d(a,b) = \sum_{i=0}^{n-1} (a_i \oplus b_i)$$

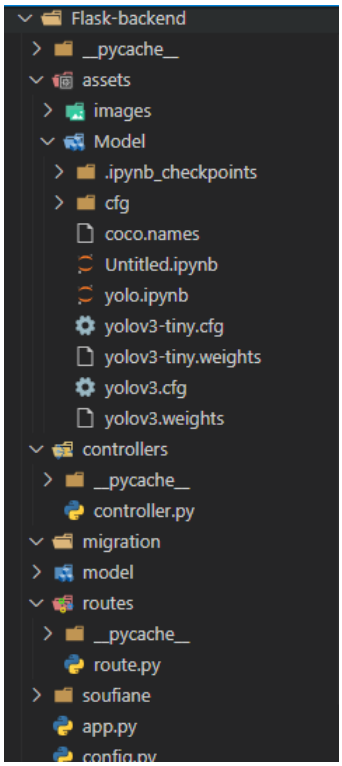
# Déploiement

## Architecture de l’application

- **Partie front-end**




- **Partie back-end**




Le fichier app.py

app.py

```python
from flask import Flask
from flask_cors import CORS
from flask_restful import Api
from routes.route import routes

app = Flask(__name__)
api = Api(app)
CORS(app)
app.register_blueprint(routes,url_prefix='')

if __name__ == '__main__':
    app.run(debug=True)
```

- L’extraction de vecteur caractéristique des images

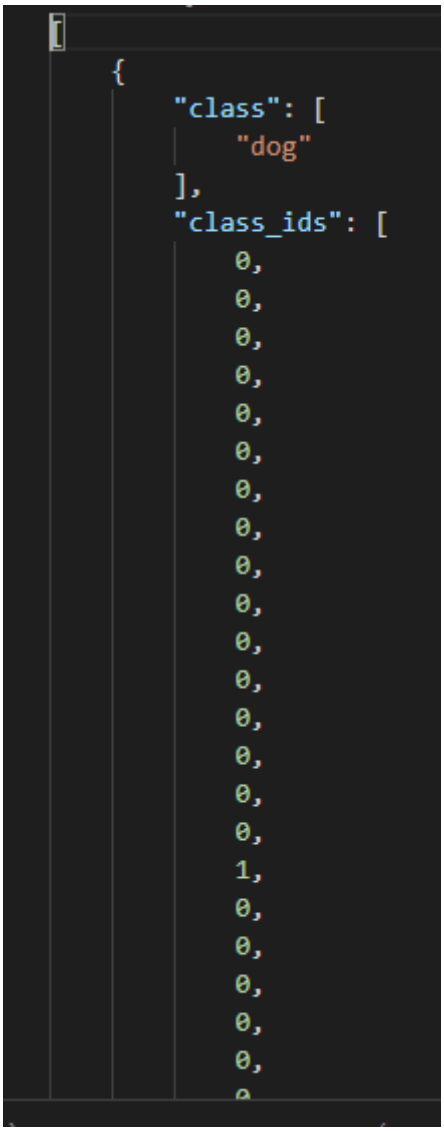
```python
import cv2
import numpy as np
import os
import json
import base64
from scipy.spatial import distance


class Extract:
    def __init__(self) -> None:
        self.path = "assets/dataSet/"

    def yoloModel(self, img1):
        net = cv2.dnn.readNet("assets/Model/yolov3.weights",
                              "assets/Model/yolov3.cfg")
        classes = []
        with open("assets/Model/coco.names", "r") as f:
            classes = [line.strip() for line in f.readlines()]

        layer_names = net.getLayerNames()
        output_layers = [layer_names[i-1]
                         for i in net.getUnconnectedOutLayers()]
        colors = np.random.uniform(0, 255, size=(len(classes), 3))
        img = cv2.imread(img1)
        img = cv2.resize(img, None, fx=0.4, fy=0.4)
        height, width, channels = img.shape
        blob = cv2.dnn.blobFromImage(
            img, 0.00392, (416, 416), (0, 0, 0), True, crop=False)
```

```
        img, 0.00392, (416, 416), (0, 0, 0), True, crop=False)
    net.setInput(blob)
    outs = net.forward(output_layers)
    # Showing informations on the screen
    class_ids = []
    confidences = []
    boxes = []
    for out in outs:
        for detection in out:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]
            if confidence > 0.5:
                # Object detected
                center_x = int(detection[0] * width)
                center_y = int(detection[1] * height)
                w = int(detection[2] * width)
                h = int(detection[3] * height)
                # Rectangle coordinates
                x = int(center_x - w / 2)
                y = int(center_y - h / 2)
                boxes.append([x, y, w, h])
                confidences.append(float(confidence))
                class_ids.append(class_id)
```

```
    indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)
    font = cv2.FONT_HERSHEY_PLAIN
    res = {
        'class': [],
        'class_ids': [],
        'src': ''
    }
    for i in range(len(boxes)):
        if i in indexes:
            # x, y, w, h = boxes[i]
            label = str(classes[class_ids[i]])
            # color = colors[i]
            # cv2.rectangle(img, (x, y), (x + w, y + h), color, 2)
            # cv2.putText(img, label, (x, y + 30), font, 2, color, 2)
            string = base64.b64encode(
                cv2.imencode('.jpg', img)[1]).decode()
            res['class'].append(label)
            res['class_ids'].append(class_ids[i])
            res['src'] = ''+string
    zeros = np.zeros(80,dtype=int)
    for n in res['class_ids']:
        zeros[n]=1
    res['class_ids']=list(zeros)
    return res
```

```
    def np_encoder(self, object):
        if isinstance(object, np.generic):
            return object.item()

    def hamming_distance(self, img1, img2):
        score = distance.hamming(img1, img2)
        return score

    def build_vector(self, img):
        pass


if __name__ == "__main__":
    extract = Extract()
    pathi = 'assets/images/'
    paths = []
    res = []
    with os.scandir(pathi) as entries:
        for entry in entries:
            paths.append(pathi+entry.name)

    for path in paths:
        res.append(extract.yoloModel(path))

    with open('fi.json', 'w') as f:
       json.dump(res,f,default=extract.np_encoder)
    # vec1 = np.zeros(80,dtype=int)
```

- Le fichier **fi.json** contient tous les vecteurs caractéristiques générés



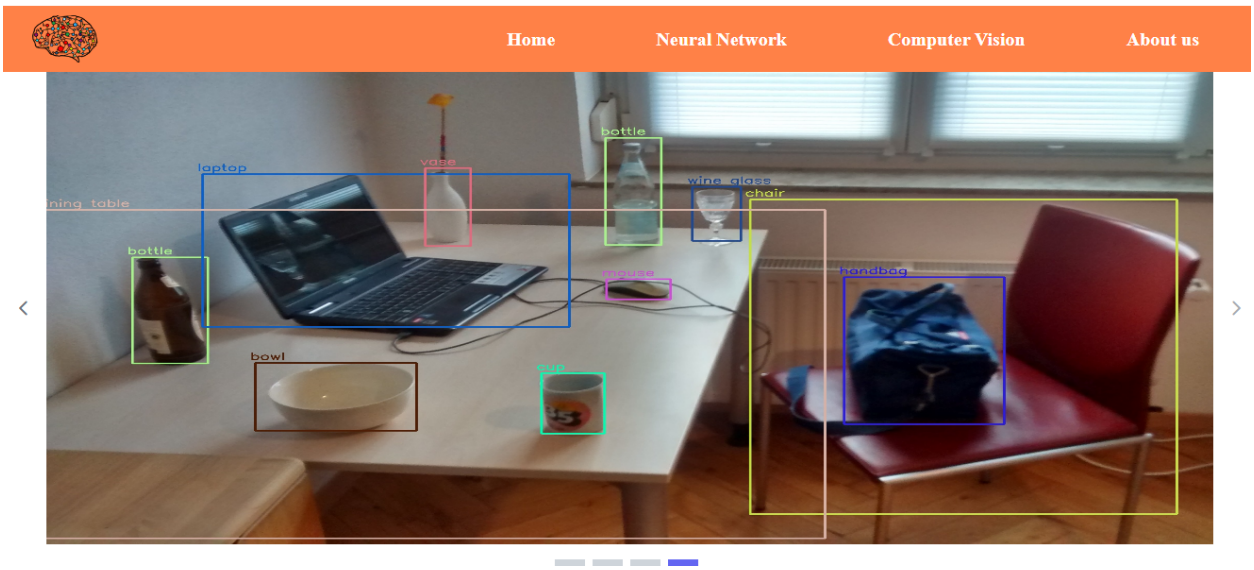

- La fonction qui calcule la distance de **Hamming** entre deux images

```
def hamming_distance(self, img1, img2):
    score = distance.hamming(img1, img2)
    return score
```

- La fonction **search()**

```
def search():
    ex = Extract()
    data = json.loads(request.data)
    vec = np.zeros(80,dtype=int)

    for d in data:
        vec[d]=1

    print(vec)
    with open('fi.json', 'r') as f:
        red = json.load(f)
    score = []
    img=list()
    for r in red:
        if ex.hamming_distance(vec,r['class_ids'])<=0.0125:
            img.append(r['src'])

    print(score)
    return {
        "images":img,

    }
```

L’interface de l’application



l’image d’entrée

La sortie

search

**Boxes : 4**
**Classes or object In the Image**

- cat
- person
- person
- dog

## Information

AI
Privacy and Policy
About us

## Contact us

ai@contact.com
+212 689-652792

## Newsletter

exemple@gmail.com
subscribe

all rights reserved-AI

zebra.jpeg

**Boxes : 1**
**Classes or object In the Image**

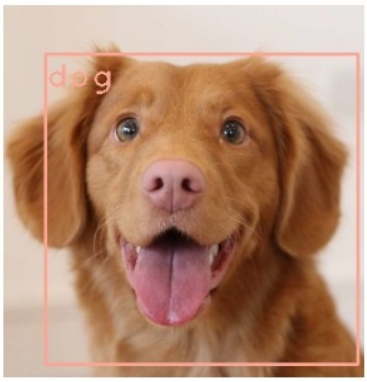
- zebra

cat2.jpeg


**Boxes : 1**
**Classes or object In the Image**

- cat




**Boxes : 1**
**Classes or object In the Image**

- dog