UNIVERSITE DE KINSHASA

FACULTE DE SCIENCES ET TECHNOLOGIES

MENTION MATHEMATIQUES, STATISTIQUE ET INFORMATIQUE

B.P 190 KINSHASA XI

SEMINAIRE DE VISUALISATION DES DONNEES

Traitement d'images : Utilisation de bibliothèques comme OpenCV pour le traitement d'images et la détection d'objets

Mise en place d'une micro application web de détection des objets dans une image téléchargée

APPRENANT

MWAMBA KANDE Franklin

Prof MASAKUNA Jordan

2023-2024

I. Introduction

La vision par ordinateur est un domaine de l'intelligence artificielle qui vise à permettre aux machines de comprendre et d'interpréter le monde visuel. Ce domaine, né dans les années 1960, a connu une évolution rapide, transformant des tâches de reconnaissance de formes simples en applications sophistiquées comme la reconnaissance faciale, la conduite autonome et la détection d'objets. Cette transformation a été rendue possible grâce aux progrès des algorithmes, de la puissance de calcul et des techniques d'apprentissage automatique.

Un aspect crucial de la vision par ordinateur est la détection d'objets, qui consiste à localiser et à identifier des objets dans une image ou une vidéo. Les premières méthodes de détection d'objets utilisaient des techniques basées sur les caractéristiques, comme les descripteurs de contours ou les histogrammes de gradients orientés (HOG). Cependant, ces approches étaient limitées en termes de précision et de généralisation.

Avec l'essor de l'apprentissage profond, de nouvelles méthodes de détection d'objets ont émergé, parmi lesquelles YOLO (You Only Look Once) est l'une des plus remarquables. Développé par Joseph Redmon et ses collaborateurs, YOLO a introduit une approche novatrice où l'image entière est traitée en une seule étape par un réseau neuronal convolutionnel (CNN). Contrairement aux méthodes traditionnelles qui décomposaient l'image en régions d'intérêt (RoI), YOLO divise l'image en une grille et fait des prédictions pour chaque cellule de la grille en une seule évaluation, permettant ainsi une détection d'objets en temps réel.

YOLO a évolué au fil du temps, avec des versions successives comme YOLOv2, YOLOv3, et plus récemment YOLOv4 et YOLOv5, chacune apportant des améliorations en termes de précision et de vitesse. Ces versions ont intégré des techniques avancées telles que les réseaux résiduels, les ancres (anchors) et les mécanismes d'attention, rendant YOLO encore plus efficace pour une variété de tâches de détection d'objets.

Pour illustrer les capacités de YOLO et l'intégration de la vision par ordinateur dans une application web, nous avons développé une application avec Flask, un micro-framework web en Python, et YOLO pour la détection d'objets. OpenCV (Open Source Computer Vision Library), une bibliothèque de traitement d'images, a également été utilisée pour manipuler et traiter les images dans notre application. OpenCV, créée par Intel en 1999, est une bibliothèque open-source offrant plus de 2500 algorithmes optimisés couvrant une large gamme de tâches en vision par ordinateur. Elle est largement utilisée dans l'industrie, la recherche académique et les projets personnels.

L'application permet aux utilisateurs de télécharger une image, qui est ensuite analysée par YOLO pour détecter et annoter les objets présents. Le résultat est une image annotée retournée à l'utilisateur via l'interface web. Cette application démontre non seulement l'efficacité de YOLO pour la détection d'objets en temps réel, mais aussi comment les frameworks web peuvent être utilisés pour créer des interfaces interactives et conviviales pour les utilisateurs finaux.

II. Cadre théorique

II.1. Vision par ordinateur

La vision par ordinateur est un domaine de l'intelligence artificielle qui vise à permettre aux ordinateurs de comprendre et d'interpréter des images et des vidéos, imitant ainsi la capacité visuelle humaine. Elle englobe diverses techniques et algorithmes pour l'acquisition, le traitement, l'analyse et la compréhension d'images numériques. Les applications de la vision par ordinateur sont nombreuses et variées, allant de la reconnaissance faciale et la détection d'objets à la surveillance, la navigation autonome des véhicules, et la réalité augmentée. Les principales étapes du processus de vision par ordinateur incluent la capture de l'image, le prétraitement (comme la réduction de bruit et l'amélioration de l'image), la segmentation (séparation des objets de l'arrière-plan), et l'analyse (reconnaissance des objets et interprétation de la scène). Les progrès récents en apprentissage profond, notamment les réseaux neuronaux convolutionnels (CNN), ont considérablement amélioré les capacités de la vision par ordinateur, permettant des performances quasi-humaines dans certaines tâches.

II.2. OpenCV

OpenCV (Open Source Computer Vision Library) est une bibliothèque open-source de traitement d'images et de vision par ordinateur, initialement développée par Intel en 1999. Elle offre plus de 2500 algorithmes optimisés couvrant une vaste gamme de tâches de vision par ordinateur, telles que la détection et la reconnaissance de visages, la segmentation d'objets, la reconstruction 3D, la recherche d'images similaires, et bien plus encore. OpenCV est largement utilisée dans l'industrie et la recherche pour développer des applications interactives en temps réel. Elle est compatible avec plusieurs langages de programmation, dont Python, C++, Java et MATLAB, et peut être exécutée sur diverses plateformes, y compris Windows, Linux, Android et MacOS. OpenCV facilite le développement de solutions de vision par ordinateur en fournissant des outils puissants pour le traitement d'images, la détection de caractéristiques, et la reconnaissance d'objets, rendant ces technologies accessibles même aux développeurs non experts.

II.3. OpenCV-Python

OpenCV-Python est l'interface Python de la bibliothèque OpenCV, qui permet d'intégrer facilement les puissantes fonctionnalités de traitement d'images et de vision par ordinateur d'OpenCV dans les programmes Python. Cette interface simplifie l'utilisation des fonctions OpenCV grâce à la syntaxe simple et expressive de Python, ce qui en fait un choix populaire pour les développeurs et les chercheurs. OpenCV-Python prend en charge une vaste gamme de fonctions, allant de la manipulation d'images de base (comme le redimensionnement, le

recadrage et la conversion de formats) aux algorithmes avancés de détection et de reconnaissance d'objets. En outre, OpenCV-Python s'intègre bien avec d'autres bibliothèques Python populaires, telles que NumPy pour le traitement efficace des matrices et Matplotlib pour la visualisation des données. Cette combinaison de simplicité et de puissance fait d'OpenCV-Python un outil indispensable pour les projets de vision par ordinateur en Python.

II.4. YOLO

YOLO (You Only Look Once) est une famille d'algorithmes de détection d'objets en temps réel qui se distingue par sa rapidité et sa précision. Développé par Joseph Redmon et ses collaborateurs, YOLO adopte une approche unique en divisant l'image en une grille et en effectuant une seule évaluation pour prédire simultanément les classes d'objets et leurs positions dans l'image. Contrairement aux méthodes traditionnelles qui analysent l'image par segments, YOLO traite l'image entière en une seule fois, ce qui permet une détection d'objets extrêmement rapide. Depuis sa première version, YOLO a connu plusieurs itérations, chaque nouvelle version améliorant la précision et l'efficacité de l'algorithme. YOLOv3 et YOLOv4 ont introduit des techniques avancées comme les réseaux résiduels et les ancres (anchors), tandis que les versions , développé par l'équipe de Ultralytics, a encore optimisé la vitesse et la précision, rendant YOLO idéal pour des applications en temps réel comme la surveillance, la robotique, et la conduite autonome.

III. Implementation Pratique

III.1. Description

Dans cette étude de cas, nous avons développé une application web avec Flask permettant de détecter des objets dans une image en utilisant la bibliothèque YOLO. Le code ci-dessous montre comment cette application fonctionne :

```
from flask import Flask, request, render_template, jsonify
from settings import *
import base64
from ultralytics import YOLO
import cv2
import numpy as np

app = Flask(__name__)


@app.route("/")
def form_upload():
    # detect_object()
    return render_template('upload_img.html')

@app.route("/img_b64", methods=['POST'])
def img_b64():
    if request.method == 'POST':
        img_upload = request.files['file']
        img_b64 = detect_object(img_upload.read())

        return jsonify({"img_b64": img_b64}), 200, {'Content-Type':
'application/json; charset=utf-8'}
    return jsonify({"error": "Method not allowed"}), 405

def detect_object(image_file):
    image = cv2.imdecode(np.frombuffer(image_file, np.uint8), cv2.IMREAD_COLOR)

    # Charger le modèle YOLOv8 pré-entraîné
    model = YOLO('yolov8n.pt')

    # Charger une image à partir d'un fichier
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    # Effectuer la détection d'objets sur l'image
    results = model(image_rgb)
```

```
    # Vous pouvez aussi afficher l'image avec matplotlib
    annotated_image = results[0].plot()  # Affiche la première image avec les
résultats

    # Convertir l'image en format PNG et l'encoder en base64
    _, buffer = cv2.imencode('.png', annotated_image)
    image_b64 = base64.b64encode(buffer).decode('utf-8')

    return image_b64


if __name__ == "__main__":
    app.run(debug=DEBUG, host=HOST, port=PORT)
```

Ce code met en place une application Flask avec deux routes principales :

- /: Affiche le formulaire d'upload d'image.
- /img_b64: Reçoit l'image uploadée, effectue la détection d'objets en utilisant YOLO, et renvoie l'image annotée encodée en base64.

Le template HTML pour le formulaire d'upload d'image permet à l'utilisateur de télécharger une image et de visualiser les résultats après la détection :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <script class="jsbin" src="../static/js/jquery-3.2.1.min.js"></script>
    <script src="../static/js/script.js"></script>
    <link rel="stylesheet" href="../static/css/style.css">
</head>
<body>
    <div class="file-upload">
    <button class="file-upload-btn" type="button" id="file-input"
onclick="$('.file-upload-input').trigger( 'click' )">Importer l'image</button>

    <div class="image-upload-wrap">
        <input class="file-upload-input" type='file' onchange="readURL(this);"
accept="image/*" />
        <div class="drag-text">
        <h3>Veuillez glisser déposer ou sélectionner l'image</h3>
```

```
        </div>
    </div>
    <div class="file-upload-content">
        <img class="file-upload-image" src="#" alt="your image" />
        <div class="image-title-wrap">
        <button type="button" onclick="removeUpload()" class="remove-
image">Supprimer <span class="image-title">Importer Image</span></button>
        </div>
    </div>
    <br>
    <button type="button" class="btn-detect" onclick="detect_objects()">Detecter
les objets</button>
    <img src="" class="img-detect" alt="">
    </div>
</body>
</html>
```

Le script JavaScript gère l'affichage de l'image uploadée et envoie l'image au serveur pour détection d'objets :

```
function readURL(input) {
    if (input.files && input.files[0]) {
        var reader = new FileReader();

        reader.onload = function(e) {
            $('.image-upload-wrap').hide();
            $('.file-upload-image').attr('src', e.target.result);
            $('.file-upload-content').show();
            $('.image-title').html(input.files[0].name);
        };

        reader.readAsDataURL(input.files[0]);
    } else {
        removeUpload();
    }
}

function removeUpload() {
    $('.file-upload-input').replaceWith($('.file-upload-input').clone());
    $('.file-upload-content').hide();
    $('.image-upload-wrap').show();
    $('.img-detect').attr("src", "");
}

$('.image-upload-wrap').on('dragover', function () {
```

```
    $('.image-upload-wrap').addClass('image-dropping');
});

$('.image-upload-wrap').on('dragleave', function () {
    $('.image-upload-wrap').removeClass('image-dropping');
});

function detect_objects() {
    var formData = new FormData();
    var fileInput = document.querySelector('.file-upload-input');
    var file = fileInput.files[0];
    formData.append('file', file);

    $.ajax({
        type: "POST",
        url: "/img_b64",
        data: formData,
        processData: false,
        contentType: false,
        dataType: "json",
        success: function(response) {
            console.log(response.img_b64);
            $('.img-detect').attr("src", "" +
response.img_b64);
        },
        error: function(xhr, status, error) {
            console.error("Error:", status, error);
        }
    });
}
```

III.2. Interface graphique démonstrative

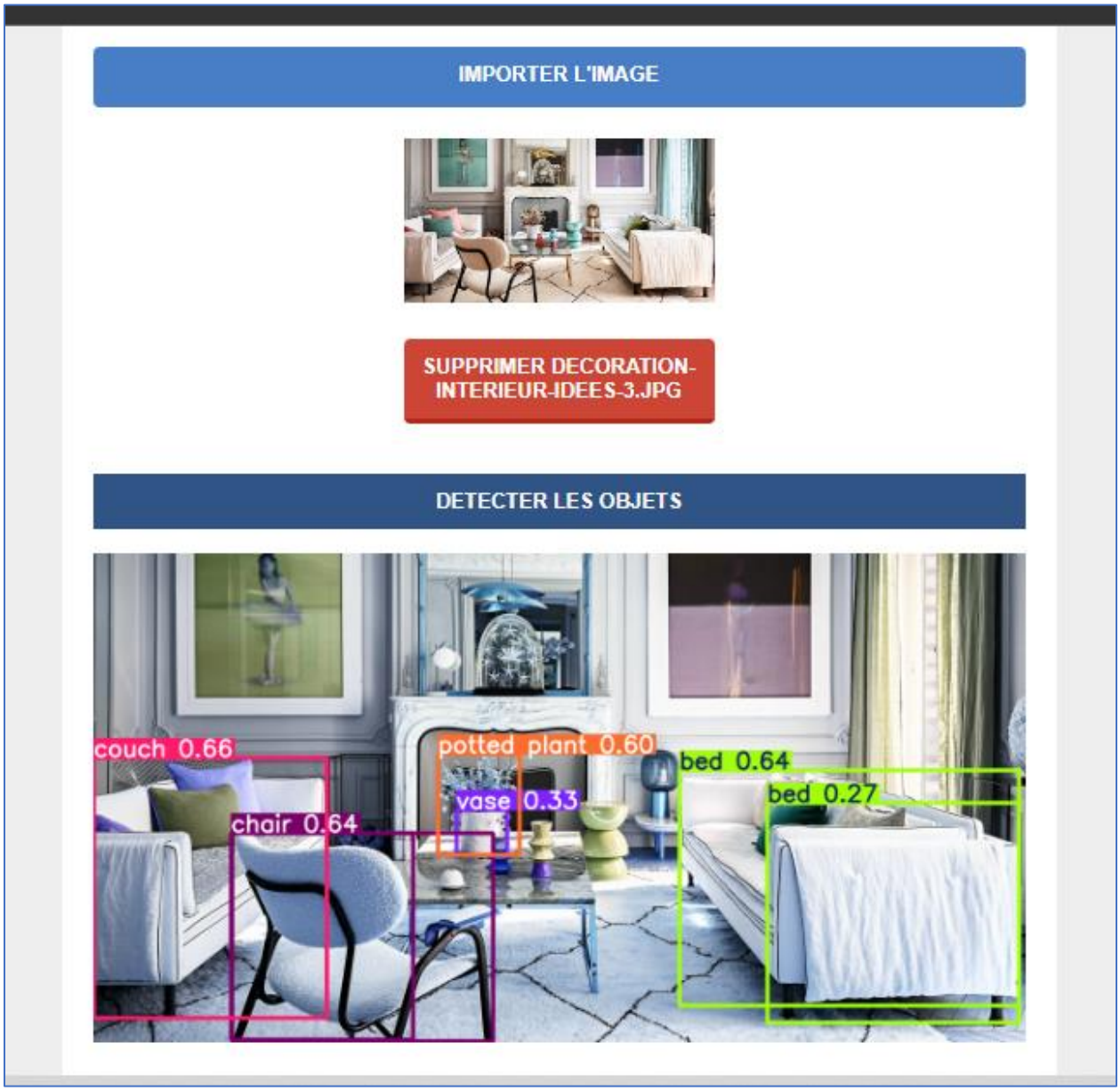

Fig.1 : Illustration de l'application

IV. Discussion

L'application développée démontre l'efficacité de la combinaison de Flask pour le développement web et YOLO pour la détection d'objets. L'utilisation de Flask permet de créer une interface utilisateur simple pour télécharger des images, tandis que YOLO fournit une détection rapide et précise des objets.

Cependant, il y a des limites à prendre en compte :

Performance : YOLO nécessite des ressources de calcul considérables, en particulier pour des modèles plus complexes ou pour le traitement de grandes quantités d'images en temps réel.

Précision : Bien que YOLO soit précis, il peut parfois manquer de détecter certains objets ou confondre des objets similaires.

Flexibilité : La personnalisation de YOLO pour des cas d'utilisation spécifiques peut nécessiter des efforts supplémentaires en termes de réglage des hyperparamètres et d'entraînement sur des ensembles de données spécialisés.

V. Conclusion

En conclusion, l'application développée démontre efficacement l'intégration de la vision par ordinateur dans une interface web, en utilisant Flask et la bibliothèque YOLO pour la détection d'objets. Cette application permet aux utilisateurs de télécharger des images et d'obtenir des annotations en temps réel, illustrant ainsi le potentiel de la vision par ordinateur pour des tâches interactives et pratiques.

L'utilisation de Flask facilite la création d'une interface utilisateur conviviale, tandis que YOLO, avec sa capacité à détecter rapidement et précisément des objets, offre une solution puissante pour la reconnaissance d'objets. La combinaison de ces technologies montre comment des frameworks de développement web et des modèles d'apprentissage profond peuvent être utilisés ensemble pour créer des applications innovantes et utiles.

Cependant, cette application n'est pas sans limites. YOLO, bien que performant, requiert des ressources de calcul importantes, ce qui peut poser des problèmes de performance, notamment pour le traitement de grandes quantités d'images ou pour des applications en temps réel exigeantes. De plus, bien que précis, YOLO peut parfois manquer certains objets ou confondre des objets similaires, ce qui peut affecter la fiabilité des résultats dans des contextes critiques. La personnalisation du modèle pour des cas d'utilisation spécifiques peut également nécessiter un ajustement minutieux des hyperparamètres et un entraînement sur des ensembles de données adaptés.

Malgré ces défis, l'application développée met en lumière les avancées et le potentiel de la vision par ordinateur. Elle ouvre la voie à des améliorations futures, telles que l'optimisation des performances, l'augmentation de la précision et la personnalisation pour des applications spécifiques. En continuant à explorer et à développer ces technologies, il est possible de créer des solutions encore plus robustes et efficaces, adaptées à un large éventail de domaines et de besoins.

Ainsi, cette application représente une étape importante dans l'utilisation pratique de la vision par ordinateur, démontrant ses capacités tout en mettant en évidence les domaines nécessitant des recherches et des améliorations continues.