



Projet Examen : WECCET 💰

Par ADAMA TRAORE

<https://github.com/adamstra/Weccet>

Introduction

Le projet **Weccet** vise à automatiser la détection et la reconnaissance des différentes pièces et billets de la monnaie Franc CFA à l'aide de techniques de vision par ordinateur. Ce projet trouve son utilité dans plusieurs domaines tels que les systèmes de comptage automatique d'argent, les distributeurs automatiques de billets, ou encore les applications bancaires pour la vérification des transactions en espèces.

Objectifs du Projet

L'objectif principal du projet **Weccet** est de développer un système capable de reconnaître et de différencier avec précision les pièces et les billets du CFA. Pour ce faire, le projet a été divisé en plusieurs étapes clés :

1. Collecte et annotation des données.
2. Entraînement d'un modèle de détection d'objets avec **YOLOv8**.

3. Développement d'une interface utilisateur avec [streamlit](#) pour l'interaction avec le modèle.

Collecte et Annotation des Données

Les images des différentes pièces et billets ont été collectées et téléchargées depuis google et certains images depuis mon kalper(portefeuille) et ensuite je les ai déposés sur la plateforme [Roboflow](#). Cette plateforme a été utilisée pour annoter les images, c'est-à-dire pour identifier et marquer les différentes pièces et billets présents sur chaque image. Les labels utilisés pour l'annotation sont les suivants :

Pieces

- **dërëm** = 5f
- **niaari dërëm** = 10f
- **juròom dërëm** = 25f
- **fukk dërëm** = 50f
- **niaari fukk dërëm** = 100f
- **nient fukk dërëm** = 200f
- **juròom fukk dërëm** = 250f

Billets

- **téeméer dërëm** = 500f
- **niaar téeméer dërëm** = 1000f
- **nient téeméer dërëm** = 2000f
- **junni dërëm** = 5000f
- **niaar junni dërëm** = 10000f





Entraînement du Modèle

Après l'annotation des données, un modèle de détection d'objets a été entraîné à l'aide de la bibliothèque [YOLO Ultralytics](#). YOLO (You Only Look Once) est une architecture de réseau de neurones convolutionnel spécialement conçue pour la détection rapide et précise d'objets dans des images. Le modèle a été configuré et entraîné sur un ensemble d'images annotées pour apprendre à reconnaître et classer les différentes pièces et billets.

Le contenu du dossier version

[YOLO](#) contient mes données (train, test, val) et un fichier data.yaml

```
train: ../train/images
val: ../valid/images
test: ../test/images

nc: 12
names: ['derem', 'fukk', 'junni', 'juroom derem',
        'juroom fukk', 'niaar junni', 'niaar teemeer', 'niaari fukk', 'nient fukk', 'nient teemeer', 't']

roboflow:
    workspace: adams-tra-xbf10
    project: weccet
    version: 2
    license: CC BY 4.0
    url: https://universe.roboflow.com/adams-tra-xbf10/weccet/d
```

▼ Etapes pour l'entraînement :

Dans ce projets on a utilisé google colabs pour avoir accès au GPU Tesla4 pour sa rapidité et sa puissance de calcul.

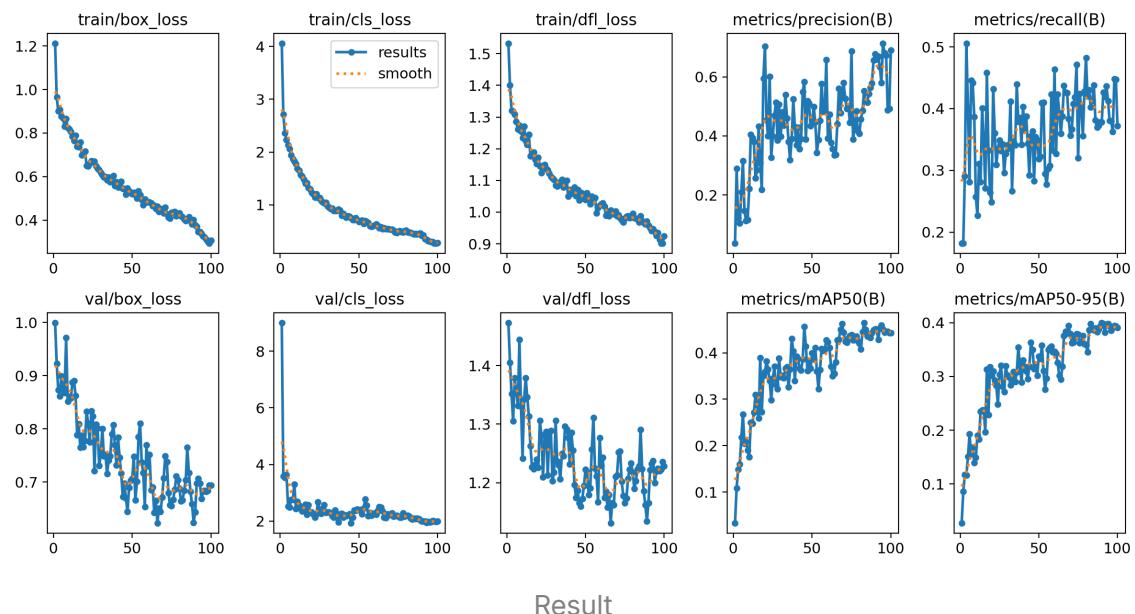
```
# Me connecter avec le Drive
from google.colab import drive
drive.mount('/content/drive')

# Entrainement Avec YOL0v8s
from ultralytics import YOLO

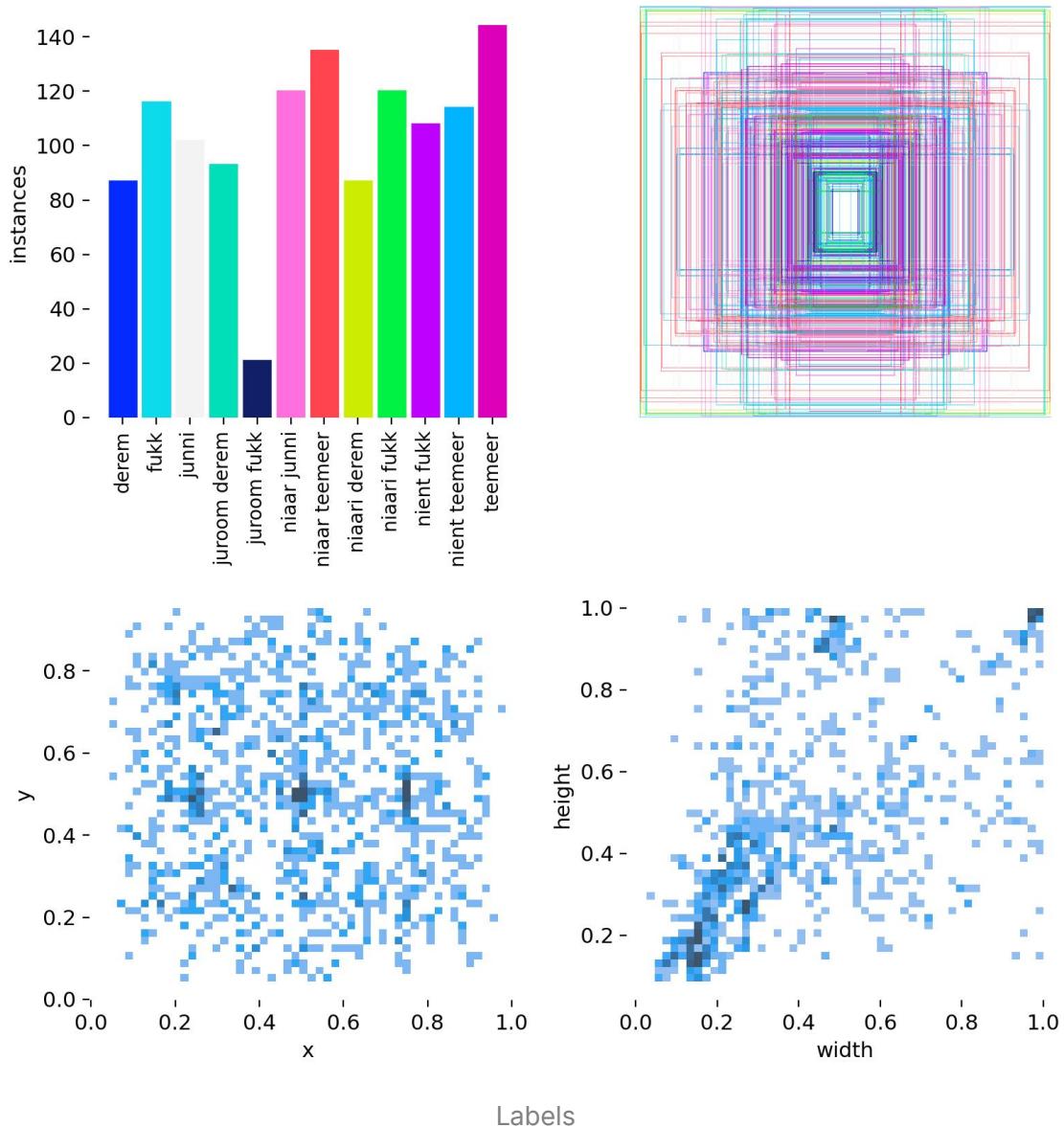
# Charger le modèle pré-entraîné (par exemple YOL0v8s)
model = YOLO('yolov8s.pt')

# Démarrer l'entraînement
model.train(data='/content/drive/MyDrive/Weccet/data.yaml'
            epochs=100, imgsz=640, batch=16, name='weccet-model')
```

Le résultat de notre model :



La fréquence de classe présent dans notre base de train

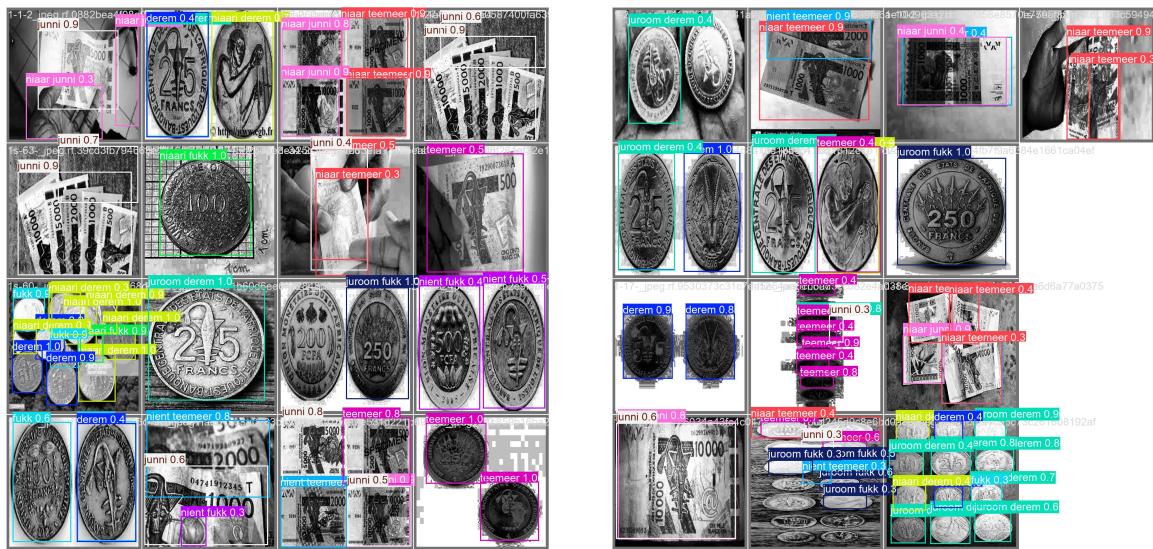


Évaluation avec des données de test

```
# Évaluation sur le jeu de validation
results = model.val()

# Évaluation sur le jeu de test
results = model.val(data='/content/drive/MyDrive/Weccet/da
                           split='test')
```

Résultat à la sortie :



Développement de l'Interface Utilisateur

Pour rendre le modèle accessible et utilisable par des utilisateurs finaux, une interface utilisateur a été développée avec Streamlit. Cette interface permet à l'utilisateur de télécharger une image contenant des pièces ou des billets de CFA, et le modèle détecte et affiche les valeurs reconnues.

OpenCV est un outil pour

le traitement d'images et la réalisation de tâches de vision par ordinateur. Il s'agit d'une bibliothèque open source qui peut être utilisée pour effectuer des tâches telles que la détection de visages, le suivi des objections, la détection de points de repère et bien plus encore.

Streamlit a été choisi pour sa simplicité et son efficacité à créer des applications web interactives pour les projets de machine learning.

▼ Le Code de notre Interface

```
import streamlit as st
import cv2
import numpy as np
```

```

from PIL import Image
from ultralytics import YOLO

# Charger le modèle Weccet fait avec YOLO
model = YOLO("best.pt")

# Partie OpenCV pour afficher les binding Box(bbox)
def run_inference(image: np.ndarray):
    st.write("Running inference...")

    if image.dtype != np.uint8:
        image = image.astype(np.uint8)

    # Dimensions originales de l'image
    original_height, original_width = image.shape[:2]

    # Redimensionner l'image pour le modèle
    image_resized = cv2.resize(
        image, (640, 640)
    )
    results = model(image_resized)

    # Extraire les boîtes englobantes, labels, et scores
    boxes = results[
        0
    ].boxes.xyxy.numpy()
    labels = results[0].boxes.cls.numpy()
    scores = results[0].boxes.conf.numpy()

    # Ajuster les coordonnées des boîtes à l'image originale
    scale_x = original_width / 640
    scale_y = original_height / 640
    boxes[:, [0, 2]] *= scale_x
    boxes[:, [1, 3]] *= scale_y

    return boxes, labels, scores

st.title("WECCET INTERFACE")

```

```

uploaded_image = st.file_uploader("Téléchargez une image",
webcam_image = st.camera_input("Ou capturez une image via la caméra")

image = None

if uploaded_image is not None:
    st.write("Image uploaded.")
    image = Image.open(uploaded_image).convert(
        "RGB"
)
elif webcam_image is not None:
    st.write("Image captured from webcam.")
    image = Image.open(webcam_image).convert("RGB")
else:
    st.write("Using default image.")
    image = Image.open("images/img.jpeg").convert(
        "RGB"
)

if image is not None:
    st.image(image, caption="Image originale", use_column_index=0)
    image_np = np.array(image)

    boxes, labels, scores = run_inference(image_np)

    for box, label, score in zip(boxes, labels, scores):
        x1, y1, x2, y2 = map(int, box)
        cv2.rectangle(image_np, (x1, y1), (x2, y2), (0, 255, 0), 2)
        cv2.putText(
            image_np,
            f"{model.names[int(label)]}: {score:.2f}",
            (x1, y1 - 10),
            cv2.FONT_HERSHEY_SIMPLEX,
            0.5,
            (0, 255, 0),
            2,
)

```

```
)  
  
    st.image(image_np, caption="Image avec détection", use  
else:  
    st.write(  
        "Aucune image n'a été chargée ou capturée."  
    )
```

Voici à quoi ressemble notre interface où on peut charger une image à partir de notre ordinateur et après automatiquement on aura notre image avec le nom de chaque pièce.



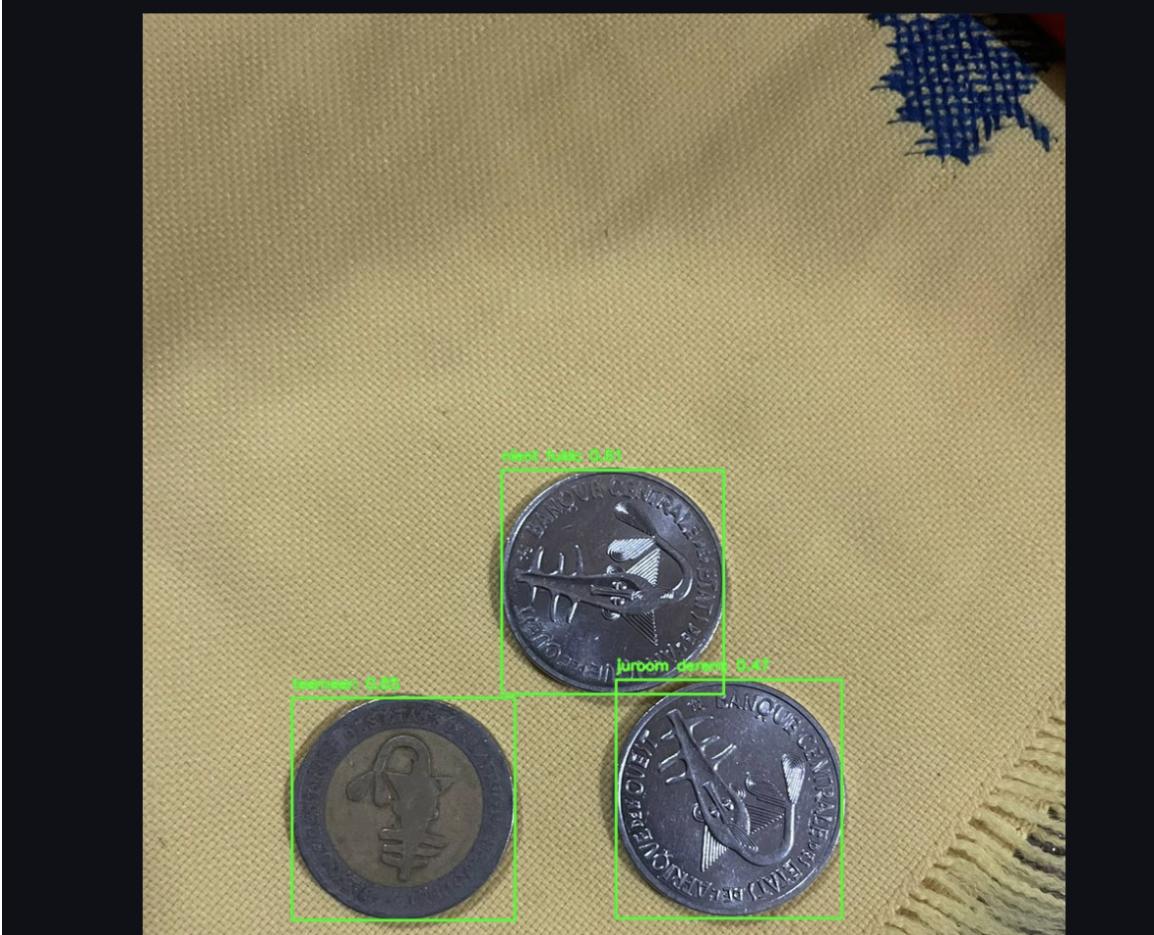
Notre Interface graphiques

Exemple de résultat:



Image originale

Running inference...



Résultats

Le modèle a montré une précision satisfaisante dans la détection et la reconnaissance des pièces et billets dans des conditions d'éclairage variées et sur des images de qualité différente. Les résultats sont prometteurs et mais avec plus de données on pourrait avoir de meilleurs résultats .

Conclusion et Perspectives

Le projet Weccet a démontré la faisabilité de l'utilisation de la vision par ordinateur pour la reconnaissance des pièces et billets du CFA. À l'avenir, l'optimisation du modèle pour une meilleure précision, l'intégration de nouvelles dénominations et l'adaptation à d'autres devises pourraient constituer des axes de développement intéressants.