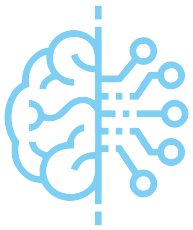

UNIVERSITÉ DE TOULON

Rapport du projet : Floutage d'entité dans des vidéos

CARON Théo
BRUN Adrien

Mention Intelligence Artificielle
Master Informatique DID

14/01/2024

Source Git :
https ://github.com/
TheoCARON1999/projetIA_floutage

Responsable module :
Patrick PERROT

Introduction

Ce rapport est le résultat du travail réalisé sur un projet du 1er semestre du Master informatique DID pour le module traitement du signal. Ce projet consiste en la réalisation ou l'utilisation d'un modèle d'apprentissage d'intelligence artificielle. Pour notre binôme en particulier le but étant la reconnaissance d'entité dans des images ou vidéos afin de réaliser un floutage automatique de l'entité désiré.

1 Le projet

Nous avons décidé de partir sur l'utilisation du YOLO version 8 fourni par Ultralytics, plusieurs raison justifie ce choix, efficace, rapide, simple d'utilisation grâce à leur API, elle bien documenté et très bien présenté sur leurs site web YOLOv8 et très important le projet est Open Source gitub et ouvert à la contribution. De plus s'éloignant d'une architecture unique, YOLOv8 prend en charge toutes les versions de YOLO, y compris celles de ses concurrents. Il permet notamment de sélection à partie de quelle probabilité on peut considérer que l'objet est présent lors de la prédiction.

Nous avons tenté à l'aide du site Roboflow de faire notre propre data set (figure 2) `https://universe.roboflow.com/work-ewkyd/foruni/dataset/2` pour entraîner le model à détecter des visages uniques, des plaques d'immatriculations ou des vitres de véhicules car le model de basse ne le permet pas mais le model crée est peu concluant car le jeu de donnée n'est pas assez conséquent.

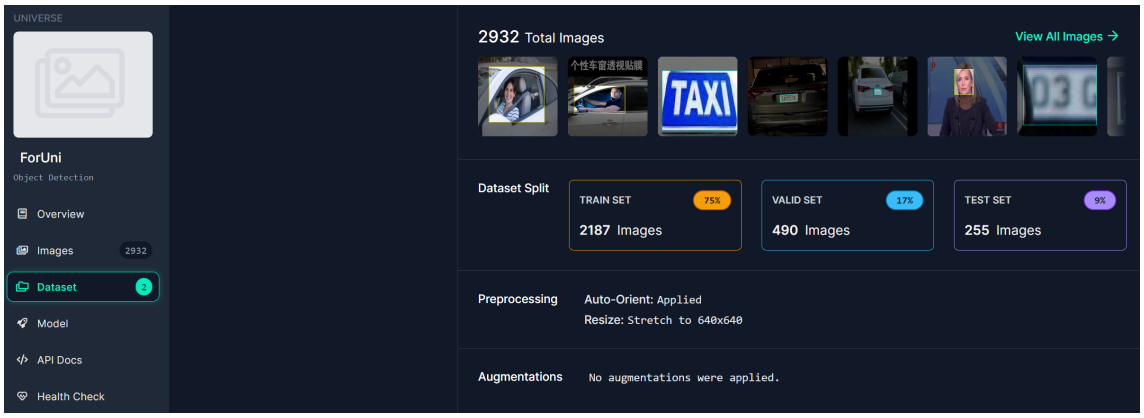

FIGURE 2 – notre data set

2 Principe de base

Si on veut flouter différente entité cela revient donc à faire de la localisation et classification d'objet :

Les algorithmes de classification d'images prédisent le type ou la classe d'un objet dans une image parmi un ensemble prédéfini de classes pour lesquelles l'algorithme a été formé. Habituellement, l'entrée est une image avec un seul objet, comme un chat. La sortie est une classe ou une étiquette représentant un objet particulier, souvent avec une probabilité de cette prédiction.

Les algorithmes de localisation d'objets localisent la présence d'un objet dans l'image et représentent son emplacement avec un cadre de délimitation. Ils prennent une image avec un ou plusieurs objets en entrée et affichent l'emplacement d'un ou plusieurs cadres de délimitation en utilisant leur position, leur hauteur et leur largeur.

3 Fonctionnement Model

Le model est entraîner à l'aide de donnée labelisé avec des boites définissants la zone de l'image ou ce situe l'objet et la classe à la quelle il appartient.

You Only Look Once fonctionne grâce à un CNN un réseau de neurone convolutif.
Cela se base donc sur un produit de convolution (figure 3) à l'aide de matrice qui permettent de détecter la variation rapide de coloration d'un pixel à l'autre pour faire de la détection de contour et donc de forme d'un objet.

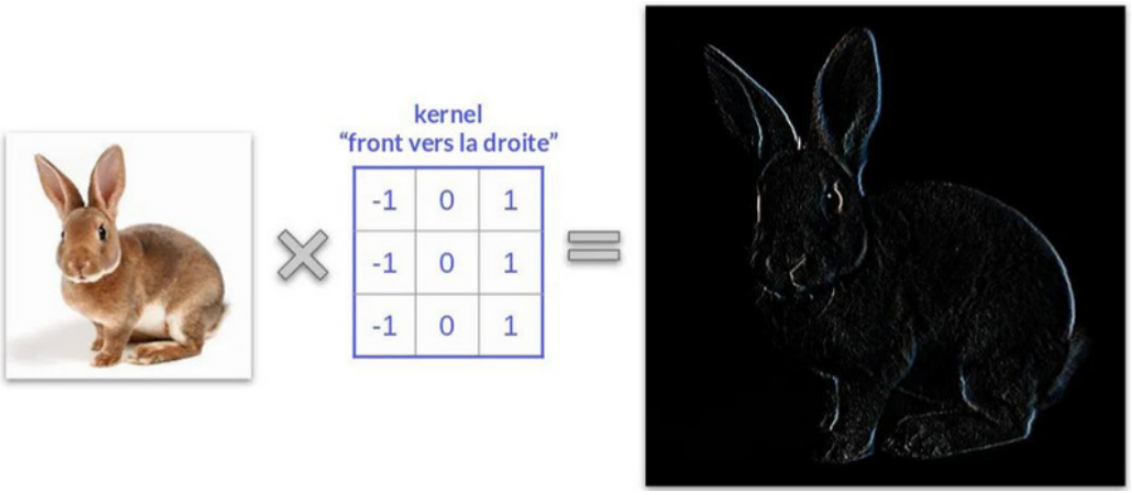

FIGURE 3 – produit de convolution

Plutôt que de travail sur un pipeline complexe qui demanderai un entrainement séparé de chaque composante comme les R-CNN (Region-based Convolutional Neural Network), YOLO voit la détection d'objets comme un problème de régression unique. Le système divise l'image d'entrée en une grille S × S. Chaque case de la

grille (figure 4) ne peut détecter qu'un seul objet donc si le centre d'un objet tombe dans une case de la grille, cette case est chargée de détecter cet objet, chaque case possède un vecteur avec la probabilité d'apparitions d'un objet, les coordonnées de son centre, sa largeur, sa hauteur et la classe à là qu'elle il appartient.

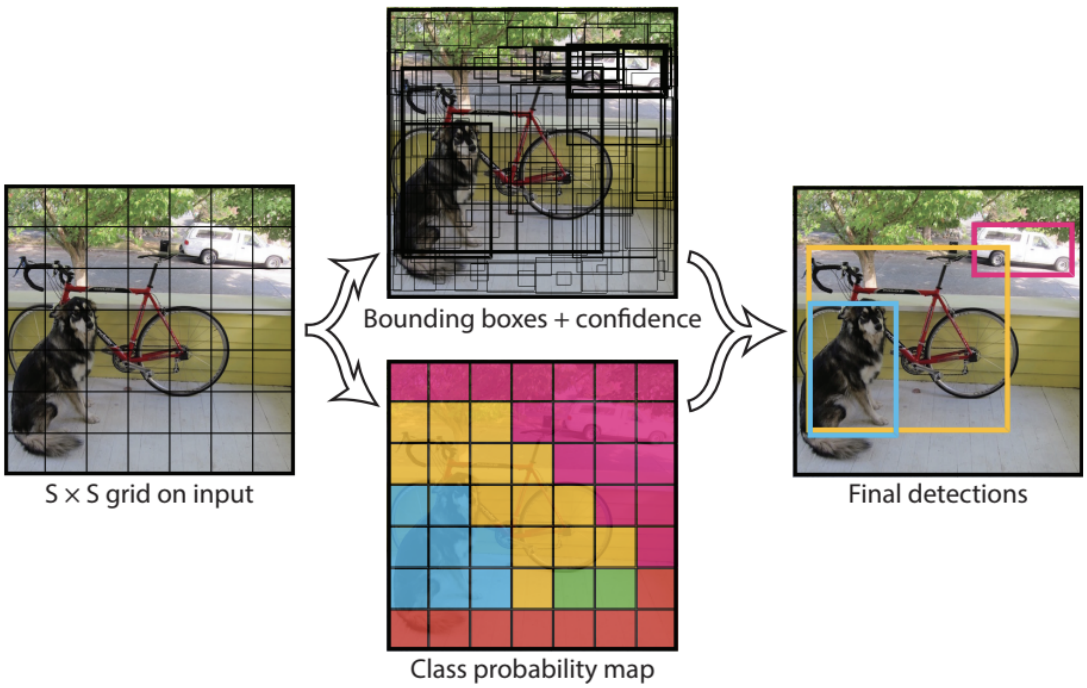

FIGURE 4 – grilles de cases de YOLO

Les couches convolutives initiales du réseau extraient les caractéristiques (features) de l'image tandis que les couches entièrement connectées prédisent les probabilités et les coordonnées de sortie.

Dans sa 1er version le réseau n'avait que 24 couches convolutives (9 pour la version légère) + 2 couches totalement connectés, dans sa 8ème version le réseau possède 53 couches convolutives et utilise des connexions partielles entre étages pour améliorer le flux d'informations entre les différentes couches.

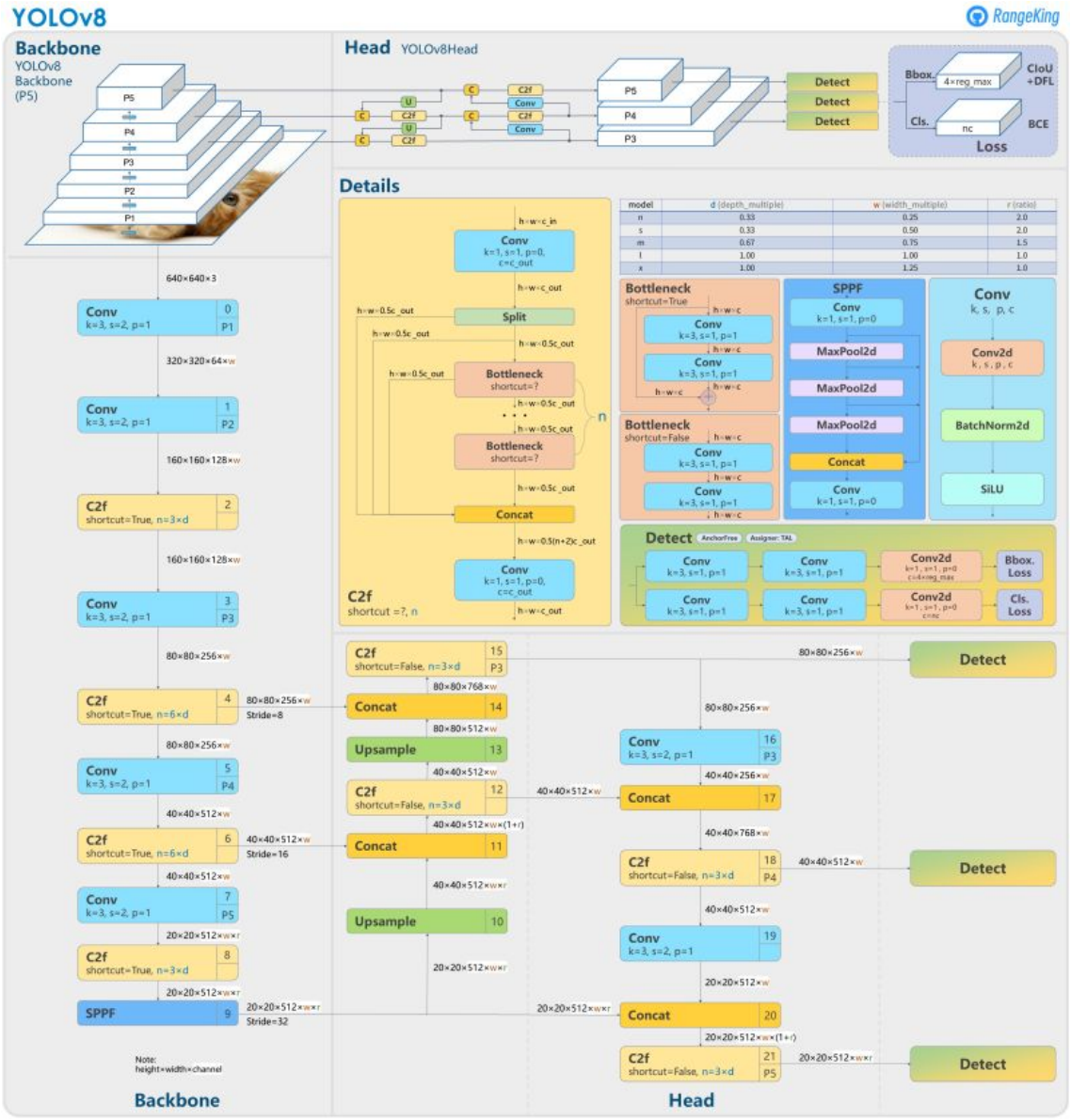

FIGURE 5 – Architecture YOLO v8

3.1 Bibliothèque

L'application est développée avec les bibliothèques suivantes :
— ultralytics
— tkinter
— PIL
— cv2
— numpy

3.2 Source

https://arxiv.org/abs/1506.02640
https://medium.com/betomorrow/les-r%C3%A9seaux-de-neurones-de-convolutions-pour-C3%A9ophytes-2b36a59cf648
https://medium.com/@VK_Venkatkumar/yolov8-architecture-cow-counter-with-region-b~:text=This%20architecture%20consists%20of%2053,series%20of%20fully%20connected%20layers.