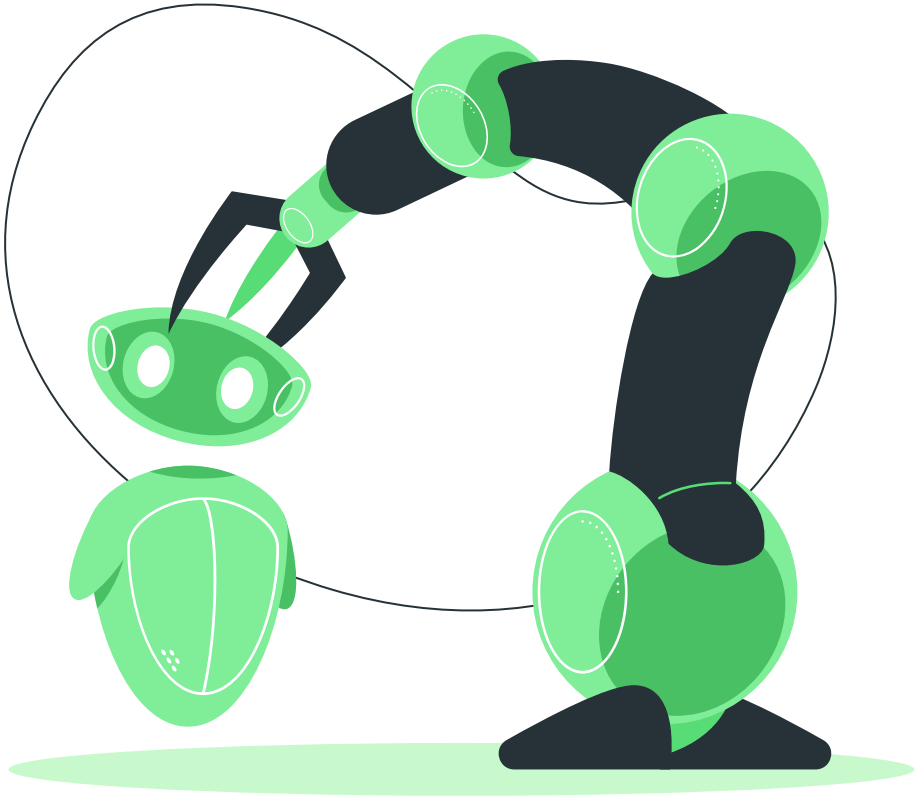
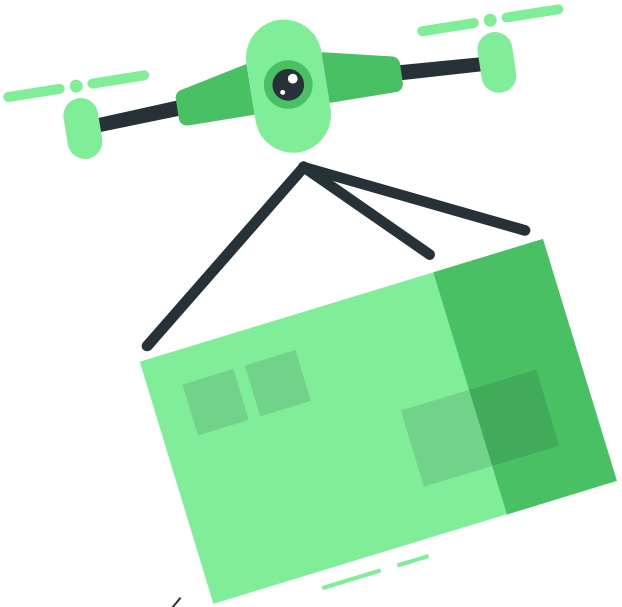
EdgeAI

Evan Galli
Jilian Lubrat
Eliot Menoret
Antoine-Marie Michelozzi



Sommaire

1. Contexte

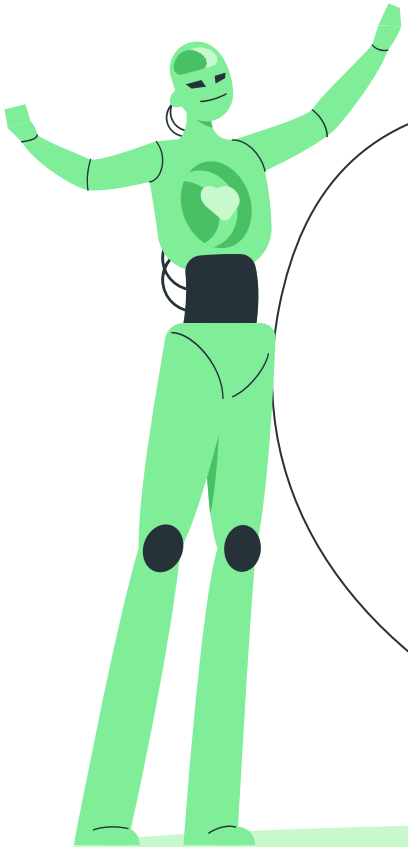


Contexte

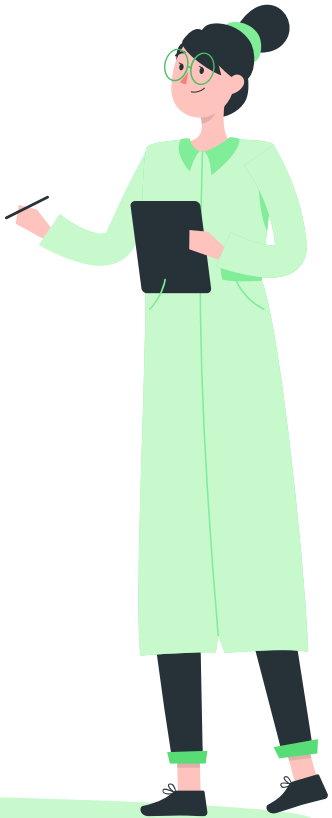
Projet de robot suiveur de personne via une caméra, nécessitant un modèle de segmentation

Objectif : Analyser les performances du modèle **YOLOv11n-seg** avec des ressources limitées

Enjeu de l'étude : Comparer les performances sur différents supports matériels de **différentes optimisations** du modèle afin de déterminer le meilleur **compromis** modèle + matériel



2. Plateformes



Plateformes

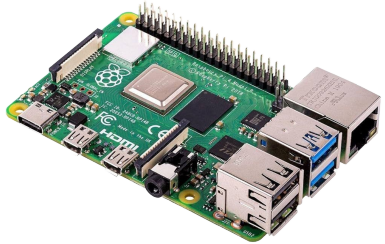
AMD Ryzen 5 8645HS

- 6 cœurs / 12 threads
- Fréquence : **4.3 GHz**
- RAM (sur laptop) **32 Go** DDR5

NVIDIA 4060 **Laptop**

- 3072 CUDA cores
- Fréquence GPU : **2.1 GHz**
- Mémoire : **8 Go** GDDR6

Plateformes



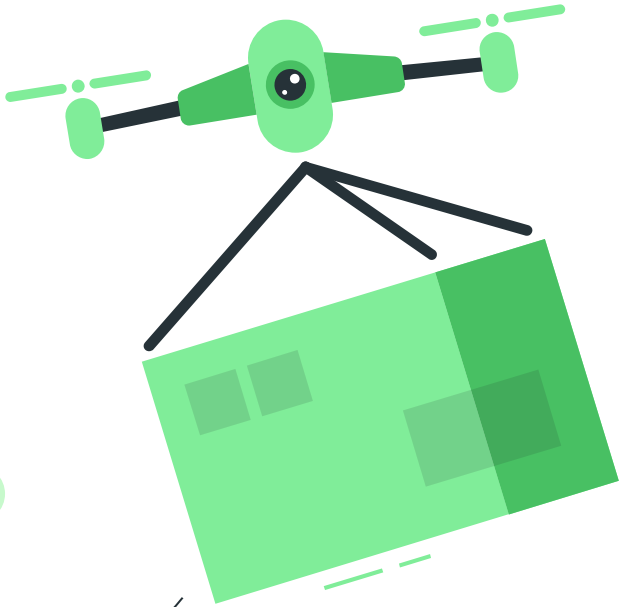
Raspberry Pi 4

- CPU Broadcom BCM2711
- 4 cœurs
- ARM Cortex-A72 **1.5 GHz**

Caméra Luxonis Oak-D Pro V2

- VPU Intel Movidius Myriad X
- 16 cœurs SHAVES
- Fréquence : **~700 MHz**

3. Protocole de benchmark



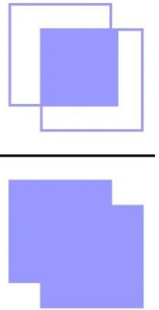
Métriques sélectionnées

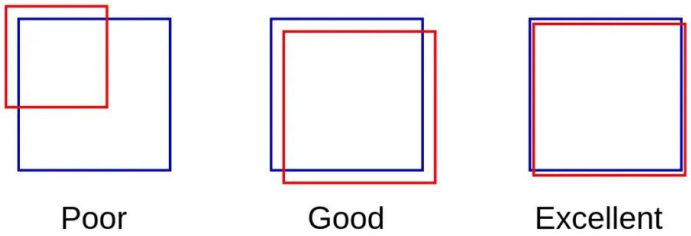
Performance Temporelle	Efficacité Énergétique	Qualité de Prédiction
Objectif : Maximiser les images par seconde pour un suivi de personne fluide et sans saccades.	**Objectif** : Minimiser la consommation énergétique pour utilisation embarquée.	**Objectif** : Maximiser la précision pour s'assurer que les personnes soient bien détectées

mAP (mean Average Precision)

Pour déterminer si vrai positif :

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$





Précision = TP / (TP + FP)

Rappel = TP / (TP + FN)

AP = Aire sous la courbe Précision-Rappel

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k$$

Calculé pour différent seuil de l'IoU [0.5;0.95] puis moyenné

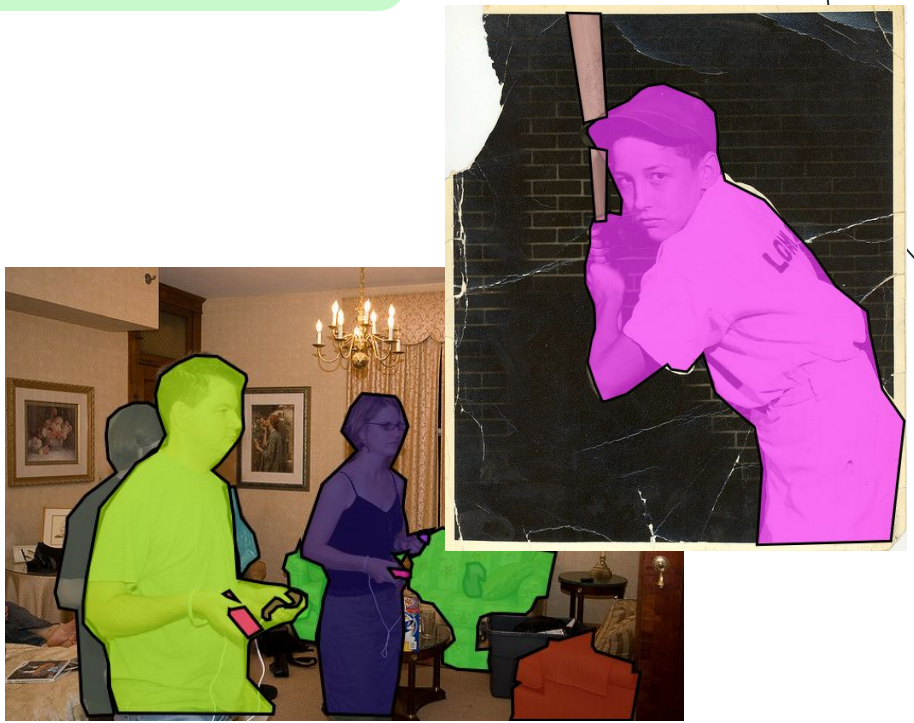
Pipeline benchmark

Dataset

- Utilisation du dataset **coco-person** sur **Kaggle**

Objectif

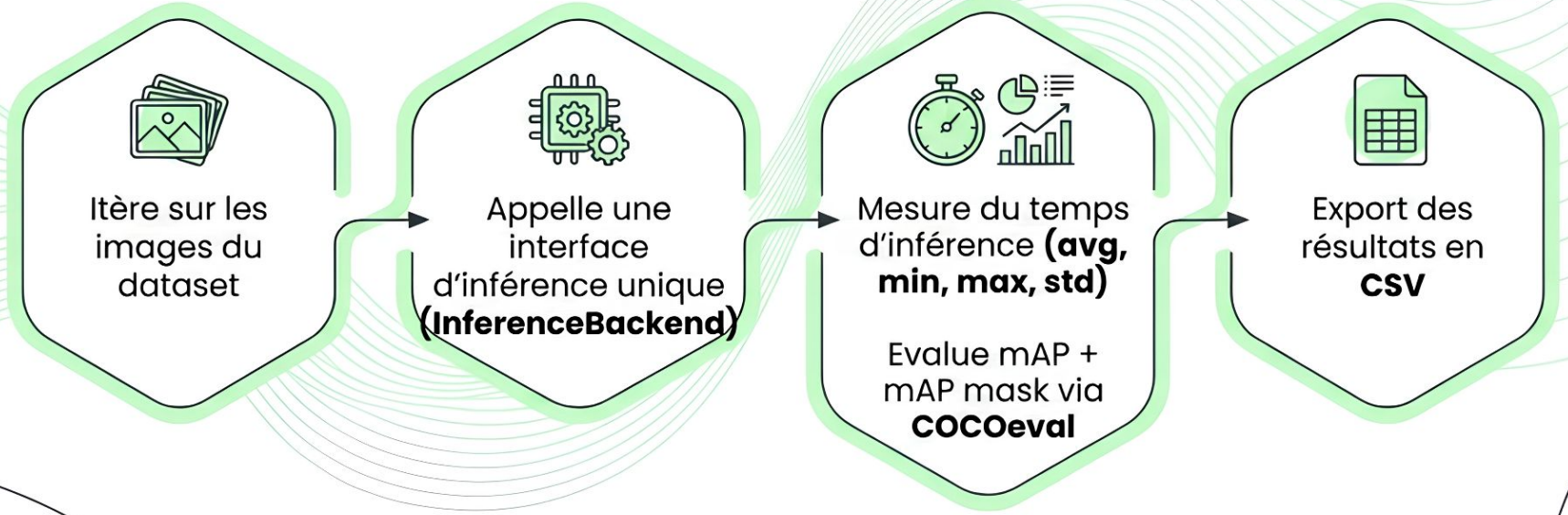
- Comparer les performances uniquement sur la classe **person**



Images extraites du dataset coco-person

Pipeline benchmark

Fonctionnement Pipeline



Inference Backend

Role

- Classe abstraite **standardisant l'inférence**
- Garantit des résultats comparables, indépendamment du **matériel** ou du **framework**

Pré-traitement

- Redimensionnement
- Letterboxing

Post-traitement

- Décodage YOLO
- Application des **masques** et **bounding boxes**
- **Retrait du letterboxing** pour reprojection sur l'image originale

Implémentations

OnnxBackend

- Sert à exécuter l'inférence YOLO sur **PC/RPi** via **ONNX Runtime**
- Permet d'évaluer **CPU** et **GPU** avec la même interface

OakDBackend

- Sert à exécuter l'inférence sur l'**OAK-D Pro** (NPU intégré)
- **Envoie l'image** à la caméra avec la lib **Depthai**, récupère les sorties du modèle

Consommation

Méthodologie de mesure

- Mesure basée sur le **delta de consommation** entre l'état idle et l'inférence
- Permet une comparaison **équitable** avec la caméra embarquée, qui n'exécute pas de système d'exploitation

Environnement contrôlé

- **Nettoyage** du système hôte
- Arrêt des processus et services **non essentiels**
- Réduction maximale des tâches en arrière-plan pour limiter le **bruit de mesure**

Consommation

GPU (NVIDIA)

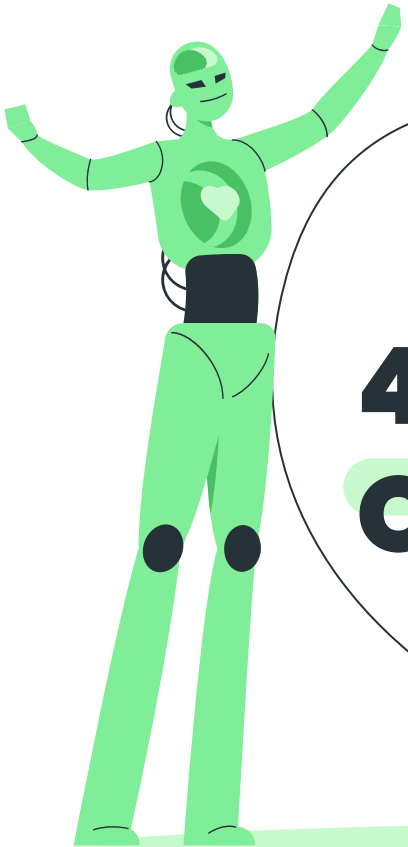
- Utilisation de **pynvml** wrapper de **NVML**, lib officielle NVIDIA pour monitorer et gérer les GPU

CPU

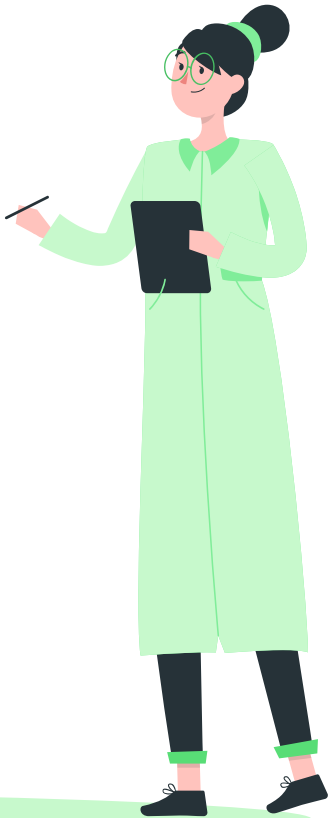
- Enregistrement des consommations via le logiciel **HWiNFO** (via export CSV)

Raspberry/Oak-d Pro

- Enregistrement vidéo du dongle USB mesurant la consommation
- Récupération d'une image par seconde
- Envois à **Gemini** des images pour récupérer les valeurs



4. Optimisations



Quantization

Principe

- Réduction de la **précision** des poids du modèle

Objectifs

- **Accélérer** l'inférence
- Réduire la consommation
- **Diminution** l'empreinte mémoire

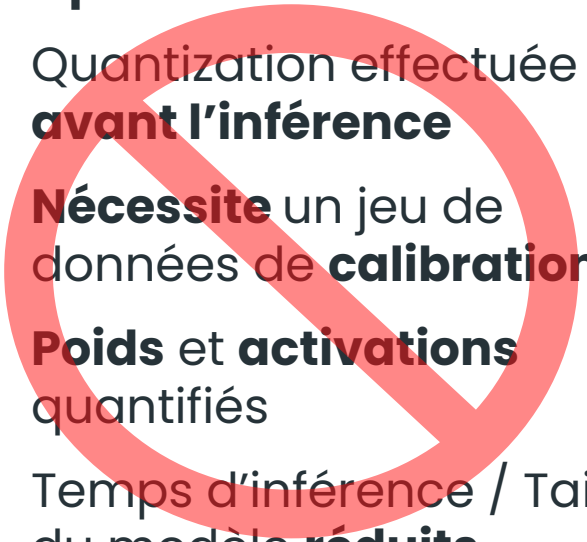
Formats utilisés

- **FP16 :** supporté sur toutes les plateformes
- **INT8 :** Raspberry Pi, CPU & GPU (non supporté sur **Oak-D Pro V2**)
- En dessous perte de précision trop importante

Quantization

Statique

- Quantization effectuée **avant l'inférence**
- **Nécessite** un jeu de données de **calibration**
- **Poids** et **activations** quantifiés
- Temps d'inférence / Taille du modèle **réduits**



Dynamique

- **Poids** quantifiés hors ligne
- **Activations** quantifiées à l'exécution
- **Réduction** taille moins importantes que **statique**
- Gains matériels **limités** (scale pour les activations)
- **Plus simple:** Pas de ré-entraînement

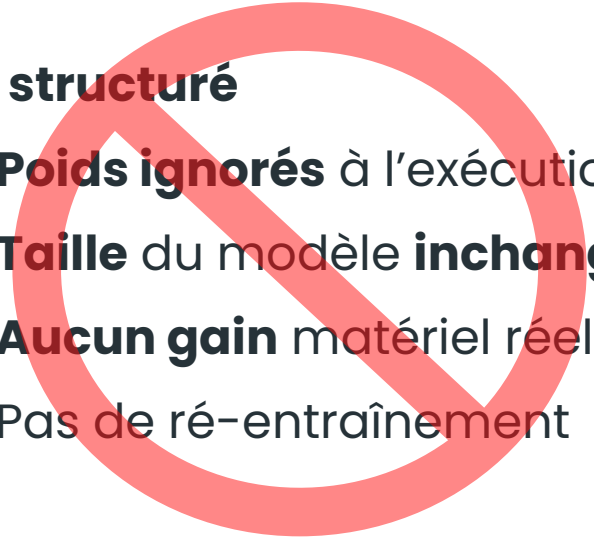
Pruning

Structuré

- **Suppression définitive** de poids ou de canaux
- Modèle **plus léger**
- **Accélération réelle** à l'inférence
- **Ré-entraînement** nécessaire

Non structuré

- **Poids ignorés** à l'exécution
- **Taille** du modèle **inchangée**
- **Aucun gain** matériel réel
- Pas de ré-entraînement



Pruning structuré

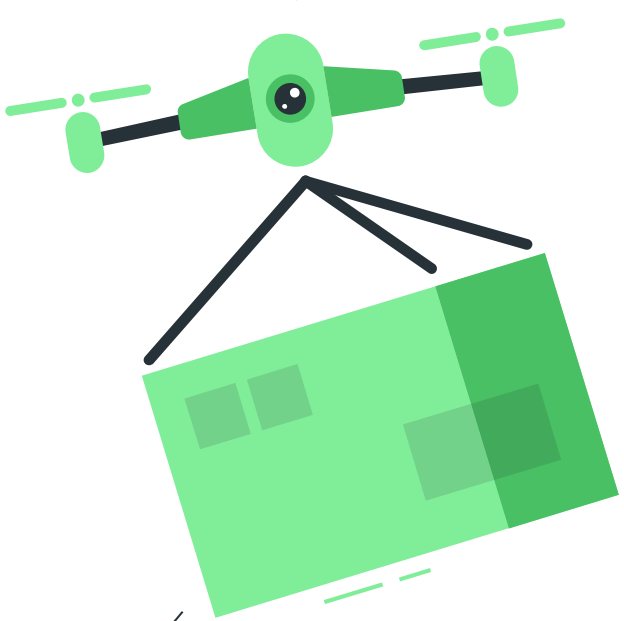
- Utilisation du projet YOLO-Pruning-RKNN par heyongxin233
- Pruning structurel qui supprime des canaux ou des couches entières. Permet d'accélérer réellement l'inférence.
- Adaptation du projet pour utiliser le dataset **coco-person** durant le réentrainement

Optimisations par plateformes

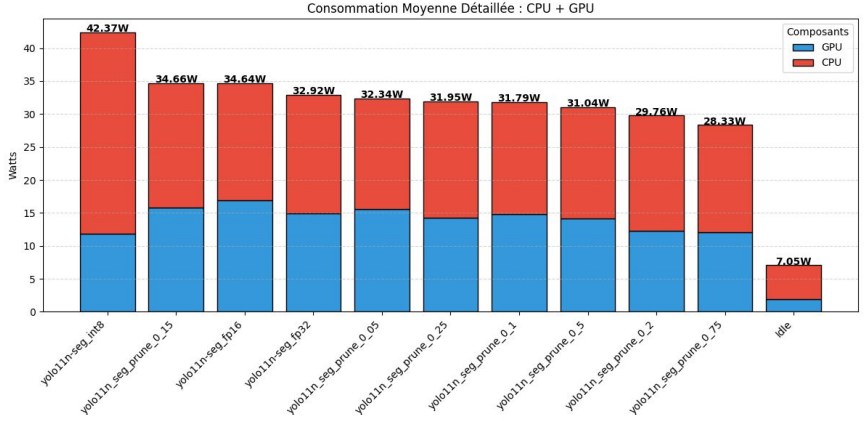
OAK-D Pro	Raspberry Pi 4	PC

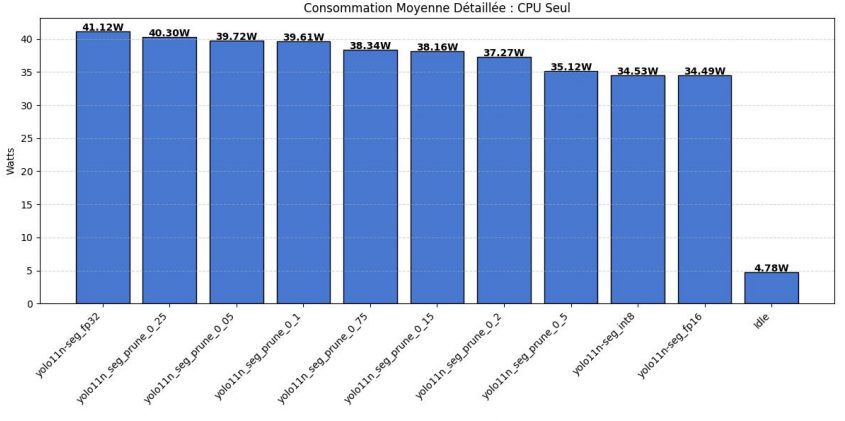
• Allocation de 8 SHAVEs (conseillé par la documentation officielle)	• Paramétrage de ONNX Runtime :
	  ○ Graph Optimization
	  ○ Multi-threading
	  ○ Gestion RAM
	  ○ Mode Séquentiel
		• Utilisation du GPU pour paralléliser les calculs dans le cas du benchmark CPU + GPU

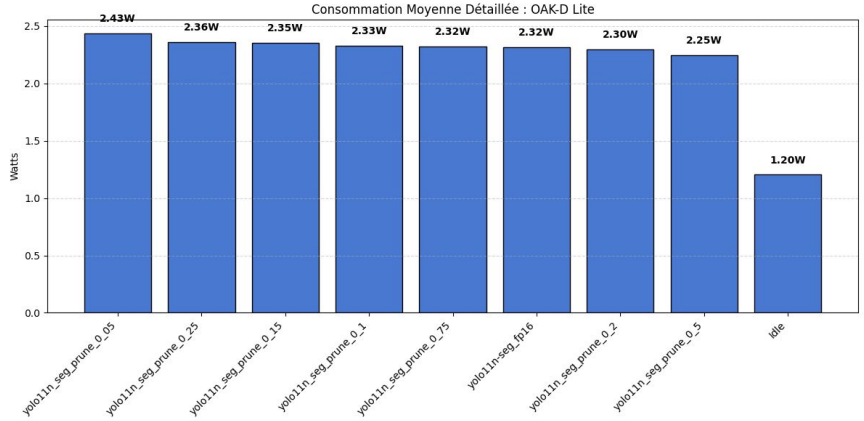
5. Résultats

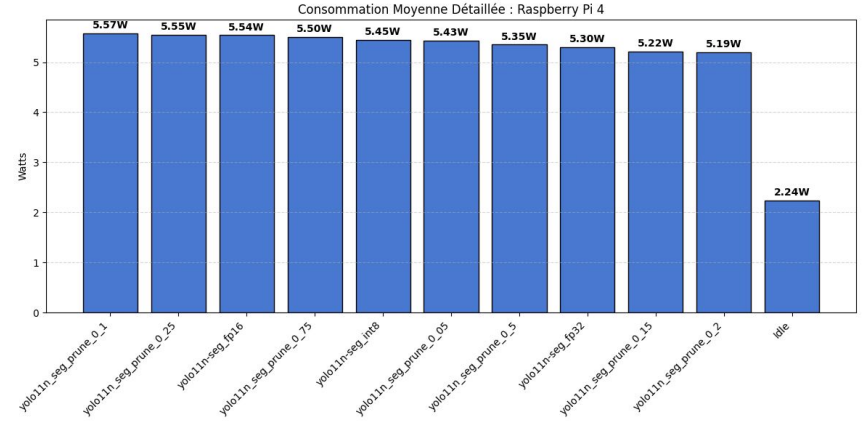


Résultats

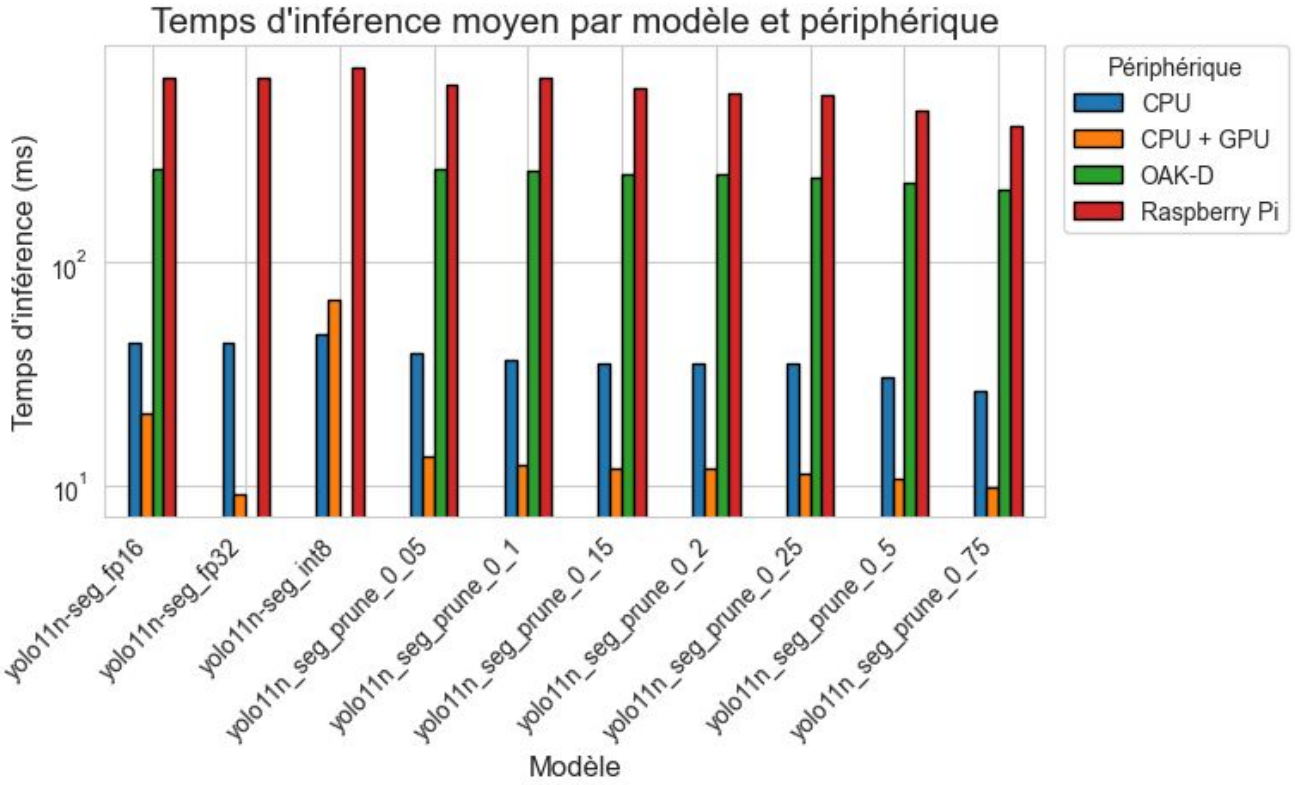

Consommation Moyenne Détaillée : CPU + GPU
Composants
GPU
CPU
Watts
42.37W
34.66W
34.64W
32.92W
32.34W
31.95W
31.79W
31.04W
29.76W
28.33W
7.05W
yolo11n-seg_int8
yolo11n_seg_prune_0_15
yolo11n-seg_fp16
yolo11n-seg_fp32
yolo11n_seg_prune_0_05
yolo11n_seg_prune_0_25
yolo11n_seg_prune_0_1
yolo11n_seg_prune_0_5
yolo11n_seg_prune_0_2
yolo11n_seg_prune_0_75
idle


Consommation Moyenne Détaillée : CPU Seul
Watts
41.12W
40.30W
39.72W
39.61W
38.34W
38.16W
37.27W
35.12W
34.53W
34.49W
4.78W
yolo11n-seg_fp32
yolo11n_seg_prune_0_25
yolo11n_seg_prune_0_05
yolo11n_seg_prune_0_1
yolo11n_seg_prune_0_75
yolo11n_seg_prune_0_15
yolo11n_seg_prune_0_2
yolo11n_seg_prune_0_5
yolo11n-seg_int8
yolo11n-seg_fp16
idle

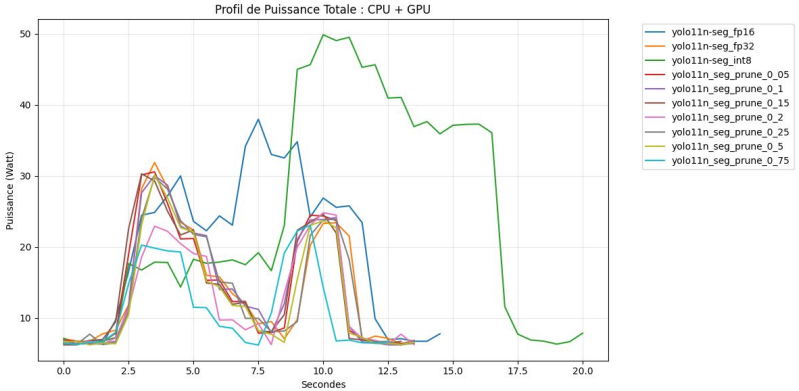

Consommation Moyenne Détaillée : OAK-D Lite
Watts
2.43W
2.36W
2.35W
2.33W
2.32W
2.32W
2.30W
2.25W
1.20W
yolo11n_seg_prune_0_05
yolo11n_seg_prune_0_25
yolo11n_seg_prune_0_15
yolo11n_seg_prune_0_1
yolo11n_seg_prune_0_75
yolo11n-seg_fp16
yolo11n_seg_prune_0_2
yolo11n_seg_prune_0_5
idle

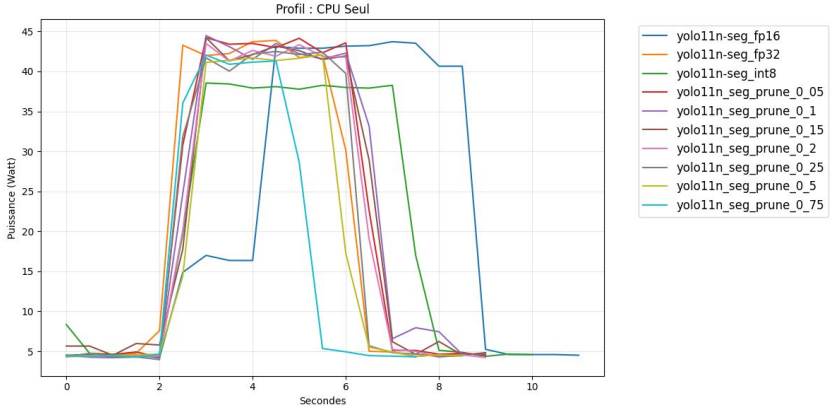

Consommation Moyenne Détaillée : Raspberry Pi 4
Watts
5.57W
5.55W
5.54W
5.50W
5.45W
5.43W
5.35W
5.30W
5.22W
5.19W
2.24W
yolo11n_seg_prune_0_1
yolo11n_seg_prune_0_25
yolo11n-seg_fp16
yolo11n_seg_prune_0_75
yolo11n-seg_int8
yolo11n_seg_prune_0_05
yolo11n_seg_prune_0_5
yolo11n-seg_fp32
yolo11n_seg_prune_0_15
yolo11n_seg_prune_0_2
idle

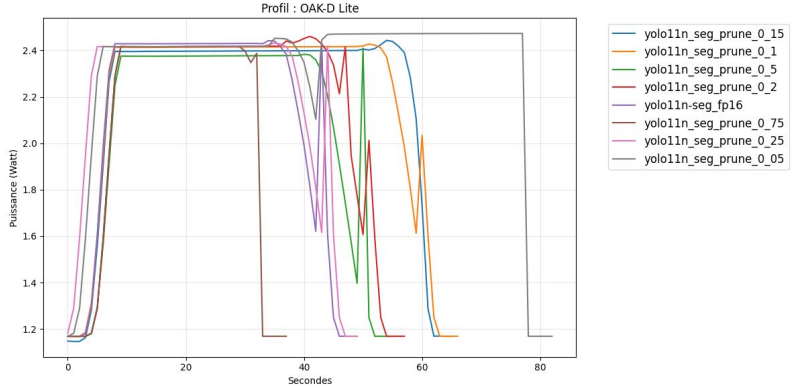
Résultats

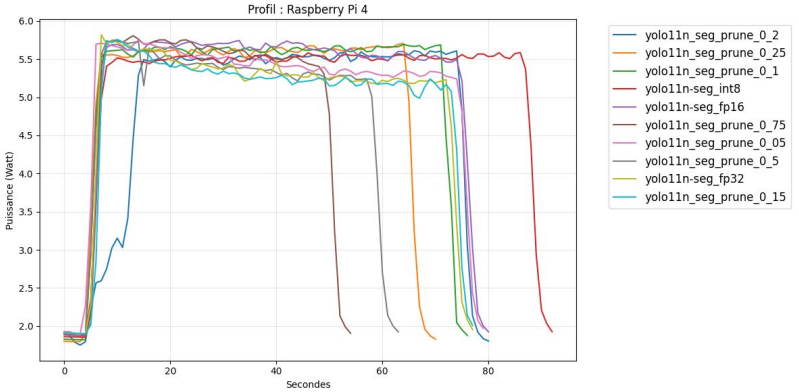

Temps d'inférence moyen par modèle et périphérique
Temps d'inférence (ms)
10^2
10^1
Périphérique
CPU
CPU + GPU
OAK-D
Raspberry Pi
yolo11n-seg_fp16
yolo11n-seg_fp32
yolo11n-seg_int8
yolo11n_seg_prune_0_05
yolo11n_seg_prune_0_1
yolo11n_seg_prune_0_15
yolo11n_seg_prune_0_2
yolo11n_seg_prune_0_25
yolo11n_seg_prune_0_5
yolo11n_seg_prune_0_75
Modèle

Résultats

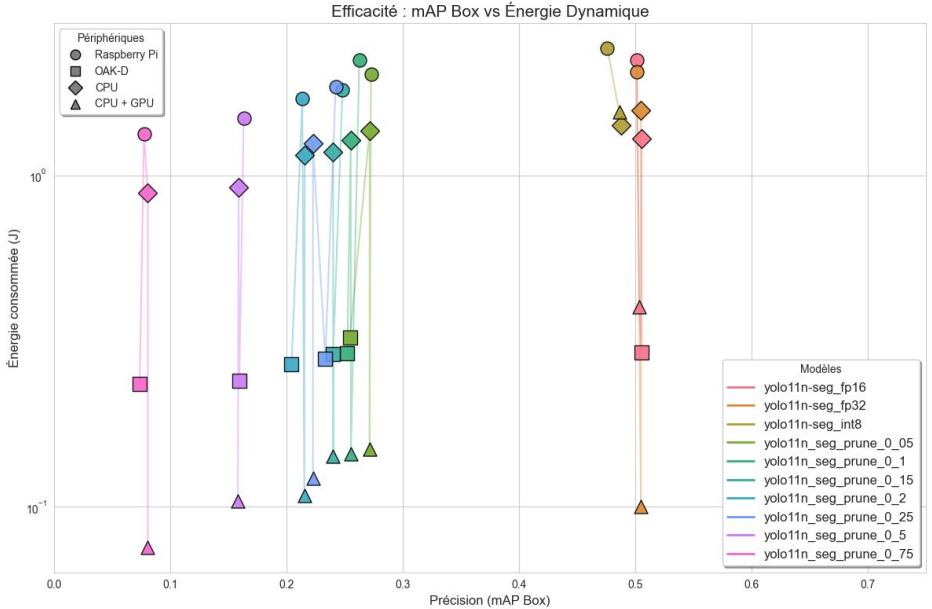

Profil de Puissance Totale : CPU + GPU
Puissance (Watt)
Secondes
yolo11n-seg_fp16
yolo11n-seg_fp32
yolo11n-seg_int8
yolo11n_seg_prune_0_05
yolo11n_seg_prune_0_1
yolo11n_seg_prune_0_15
yolo11n_seg_prune_0_2
yolo11n_seg_prune_0_25
yolo11n_seg_prune_0_5
yolo11n_seg_prune_0_75

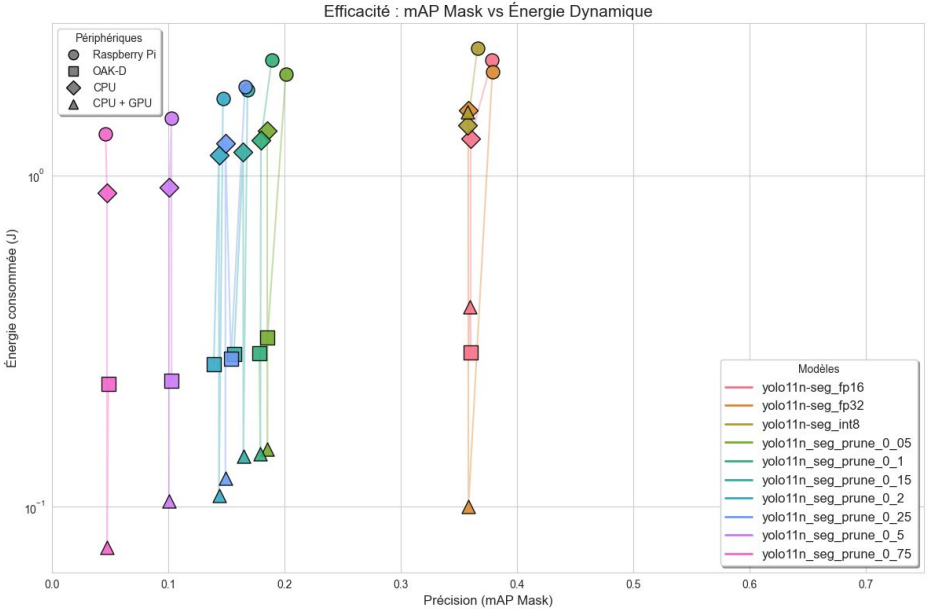

Profil : CPU Seul
Puissance (Watt)
Secondes
yolo11n-seg_fp16
yolo11n-seg_fp32
yolo11n-seg_int8
yolo11n_seg_prune_0_05
yolo11n_seg_prune_0_1
yolo11n_seg_prune_0_15
yolo11n_seg_prune_0_2
yolo11n_seg_prune_0_25
yolo11n_seg_prune_0_5
yolo11n_seg_prune_0_75

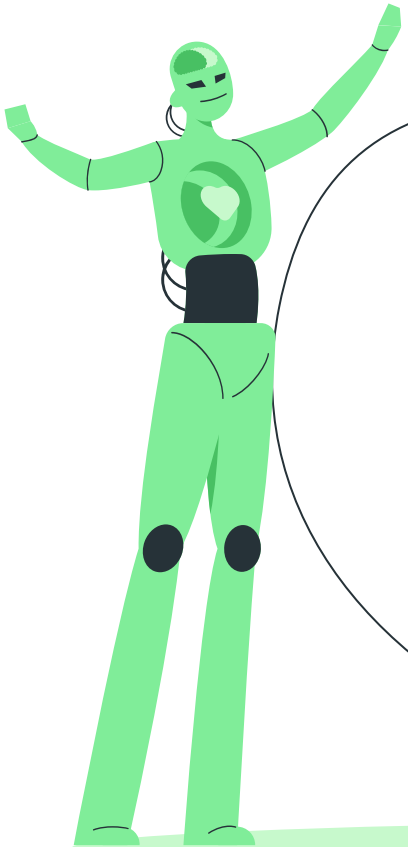

Profil : OAK-D Lite
Puissance (Watt)
Secondes
yolo11n_seg_prune_0_15
yolo11n_seg_prune_0_1
yolo11n_seg_prune_0_5
yolo11n_seg_prune_0_2
yolo11n-seg_fp16
yolo11n_seg_prune_0_75
yolo11n_seg_prune_0_25
yolo11n_seg_prune_0_05


Profil : Raspberry Pi 4
Puissance (Watt)
Secondes
yolo11n_seg_prune_0_2
yolo11n_seg_prune_0_25
yolo11n_seg_prune_0_1
yolo11n-seg_int8
yolo11n-seg_fp16
yolo11n_seg_prune_0_75
yolo11n_seg_prune_0_05
yolo11n_seg_prune_0_5
yolo11n-seg_fp32
yolo11n_seg_prune_0_15

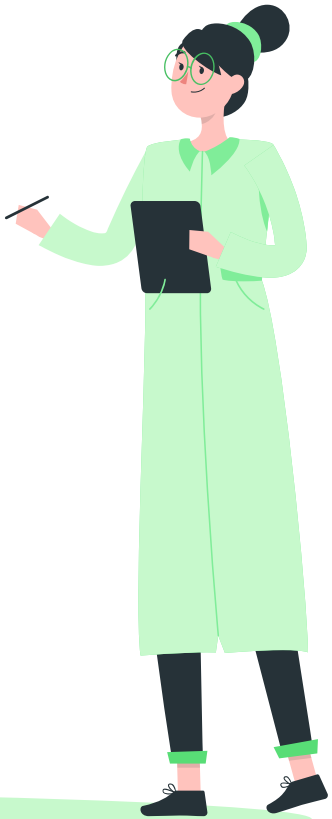
Résultats


Efficacité : mAP Box vs Énergie Dynamique
Périphériques
Raspberry Pi
OAK-D
CPU
CPU + GPU
Énergie consommée (J)
Précision (mAP Box)
Modèles
yolo11n-seg_fp16
yolo11n-seg_fp32
yolo11n-seg_int8
yolo11n_seg_prune_0_05
yolo11n_seg_prune_0_1
yolo11n_seg_prune_0_15
yolo11n_seg_prune_0_2
yolo11n_seg_prune_0_25
yolo11n_seg_prune_0_5
yolo11n_seg_prune_0_75


Efficacité : mAP Mask vs Énergie Dynamique
Périphériques
Raspberry Pi
OAK-D
CPU
CPU + GPU
Énergie consommée (J)
Précision (mAP Mask)
Modèles
yolo11n-seg_fp16
yolo11n-seg_fp32
yolo11n-seg_int8
yolo11n_seg_prune_0_05
yolo11n_seg_prune_0_1
yolo11n_seg_prune_0_15
yolo11n_seg_prune_0_2
yolo11n_seg_prune_0_25
yolo11n_seg_prune_0_5
yolo11n_seg_prune_0_75



6. Conclusion

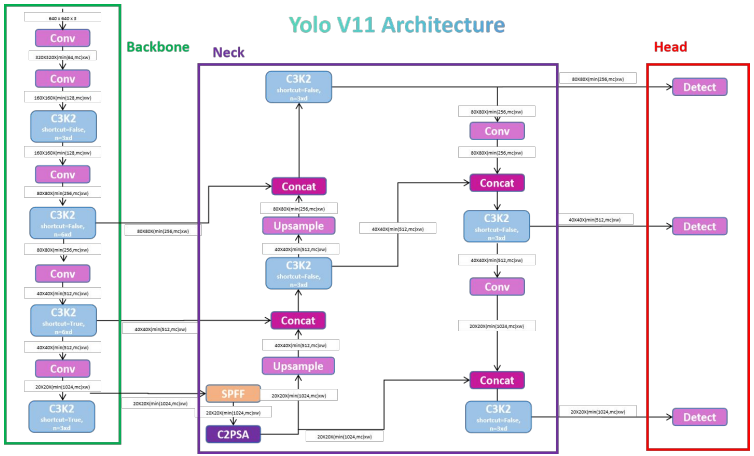


Choix Optimal : Device

- La caméra **OAK-D Pro** est le choix optimal
- Sa consommation énergétique est **légèrement inférieure** à CPU+GPU
- Le mAP (Mean Average Precision) obtenu est **similaire** sur les autres plateformes
- L'utilisation sur batterie permet une solution **embarquée**
- Meilleur compromis entre modèle et matériel

Choix Optimal : Modèle

- Le modèle en précision **FP16** est le meilleur choix à utiliser avec la caméra
- La consommation d'énergie est similaire aux autres modèles
- **mAP bien meilleur** que pour les autres modèles prunés



Merci

Avez-vous des questions ?