AVANTIX

Avancement Simulateur IQ

AVANTIX

Stages

10/01/2024

Sujets

- Computer Vision : Trouver des architectures concurrentes à YOLO. Framework mmrotate
- Feature extraction via IA : Remplacer la feature extraction TS via un conv1D multi-output
- CUDA/C++ : Implémenter nos briques logicielles en C++ → Peut être trop tard pour février ?

AVANTIX

Multi-STFT brevet

10/01/2024

AVANTIX

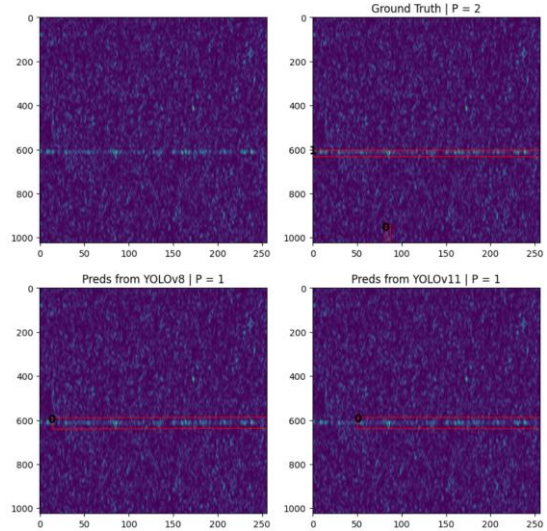
YOLOv11

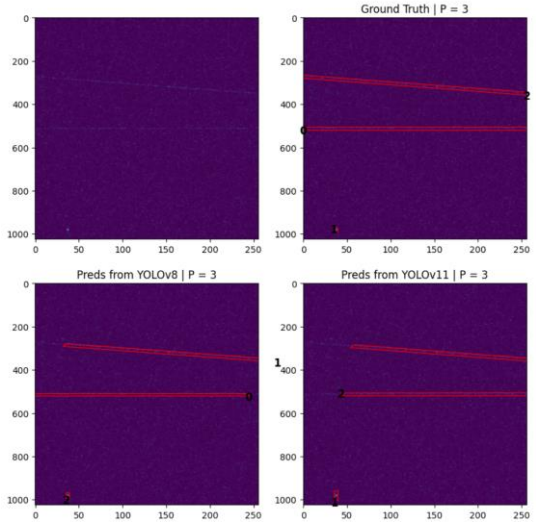
10/01/2024

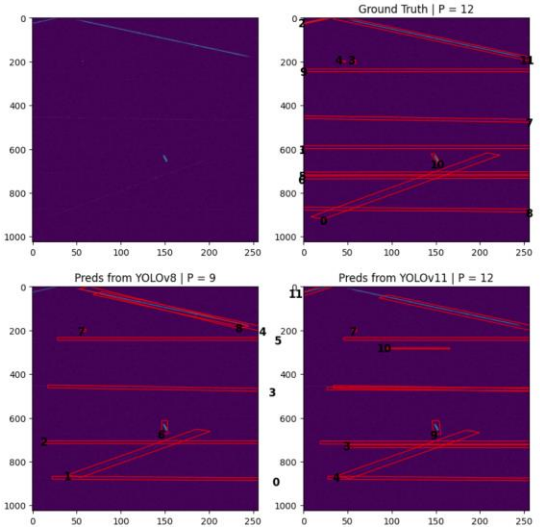
YOLOv11

- Architecture identique. Seuls certaines couches sont modifiées
- N’intègre pas la suppression du NMS présent dans YOLOv10
- Intègre l’option OBB
- Performance équivalente à v8 pour notre cas d’usage
- Gain de 25% en calcul :
  - YOLOv8 nano : 4.5 GFLOPS
  - YOLOv11 nano : 3.4 GFLOPS

YOLOv11 vs YOLOv8







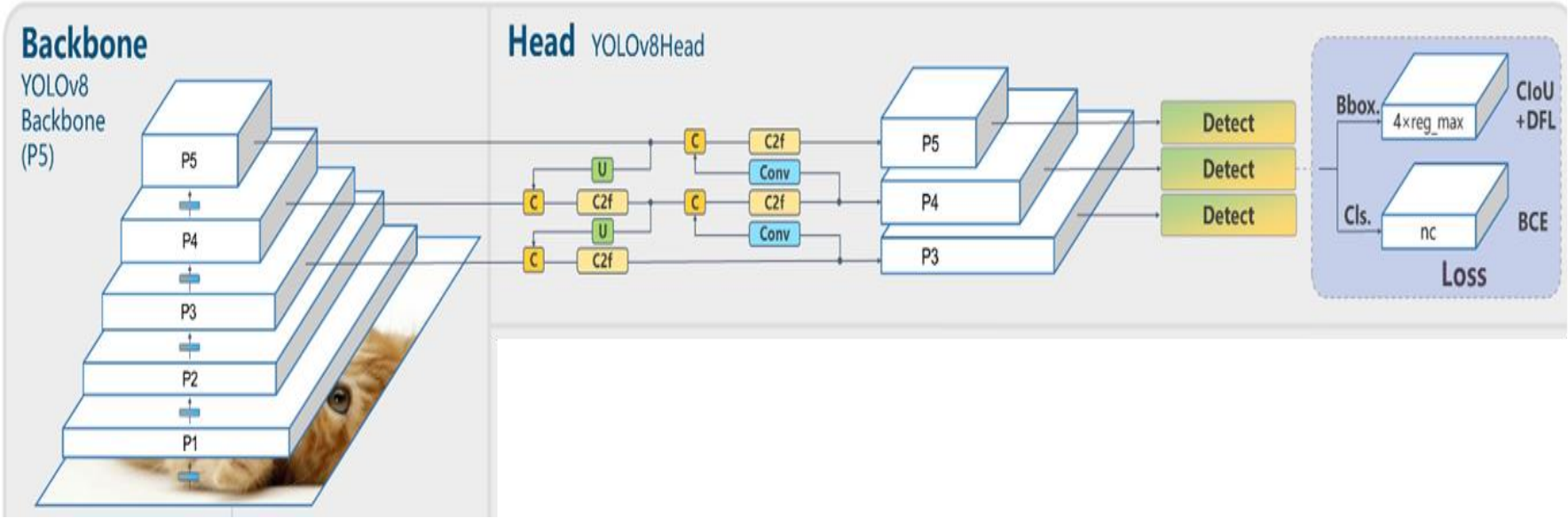
AVANTIX

Temps contraint : Inférence

10/01/2024

Revue architecture YOLOv8

P2P5 : architecture de référence



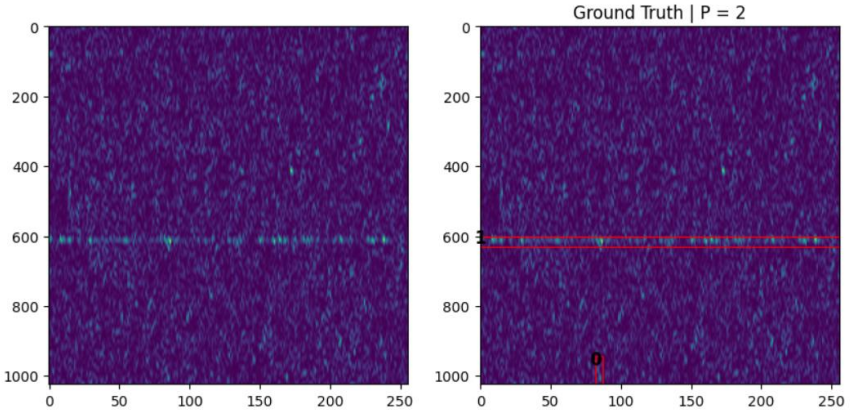
Optimisation via nombre de paramètres

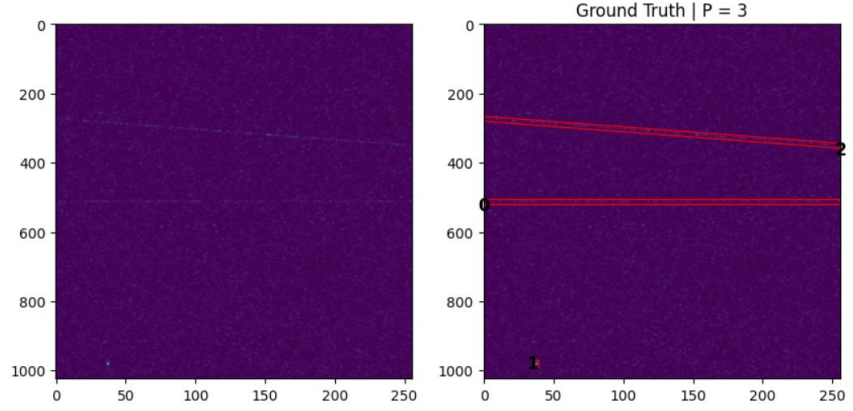
Idée : Conserver l'architecture de référence en réduisant le nombre de paramètres (surtout le nombre de channels)

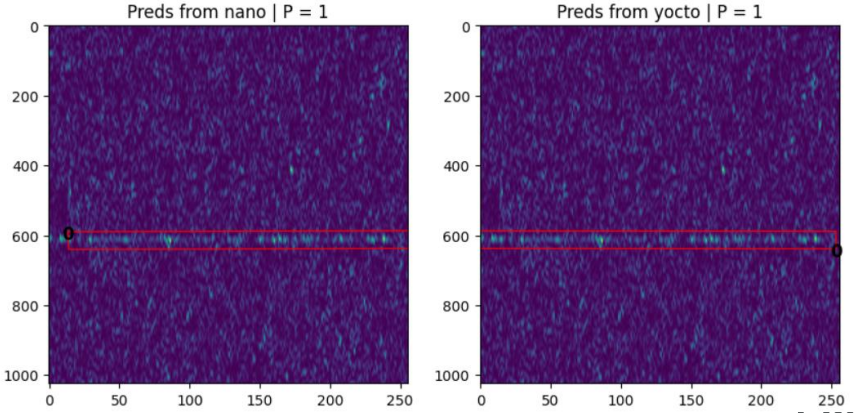
YOLO version	Architecture	Diff par rapport à ref	Taille	GFLOPs	Inference per img (µs)	Ratio to Real Time	fitness	Deployable ?
8	P2P5 (référence)	NA	nano	4,5	47,9	221	0,51	Y
8	P2P5 (référence)	NA	pico	4				
8	P2P5 (référence)	NA	femto	1,1				
8	P2P5 (référence)	NA	atto	0,8				
8	P2P5 (référence)	NA	zepto	0,3				
8	**P2P5 (référence)**	**NA**	**yocto**	**0,1**	**17,6**	**81**	**0,394**	**~**

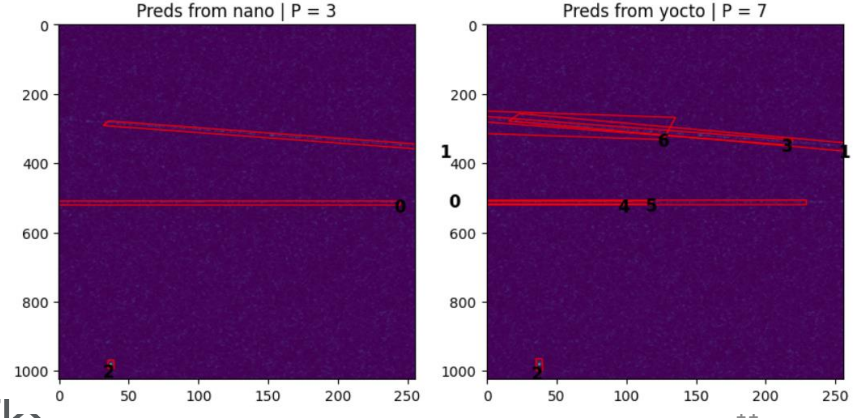
Il n'est pas intéressant d'aller plus bas car on voit que pour un facteur 45 gagné en GFLOPS on ne gagne qu'un facteur 2,7 en inférence

nano vs yocto

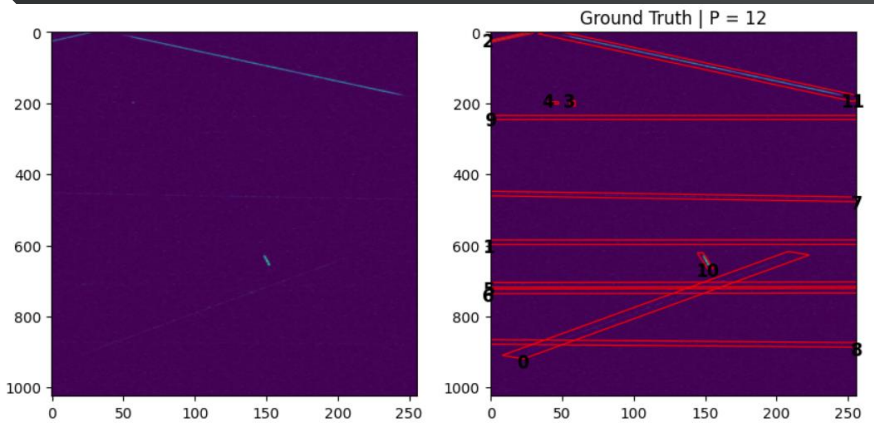


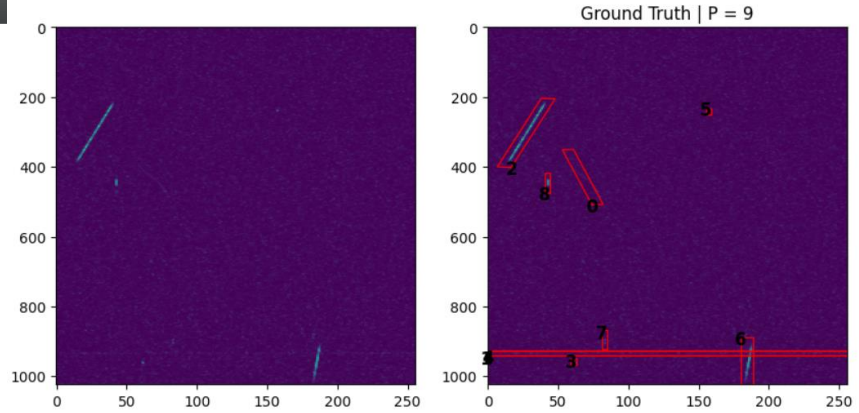


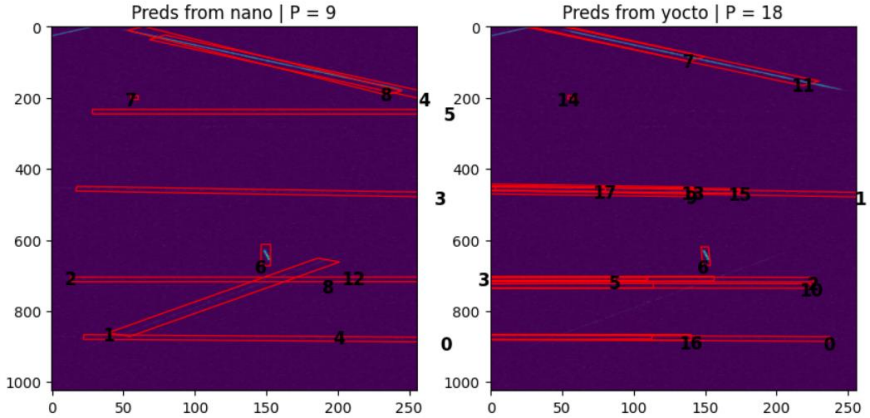


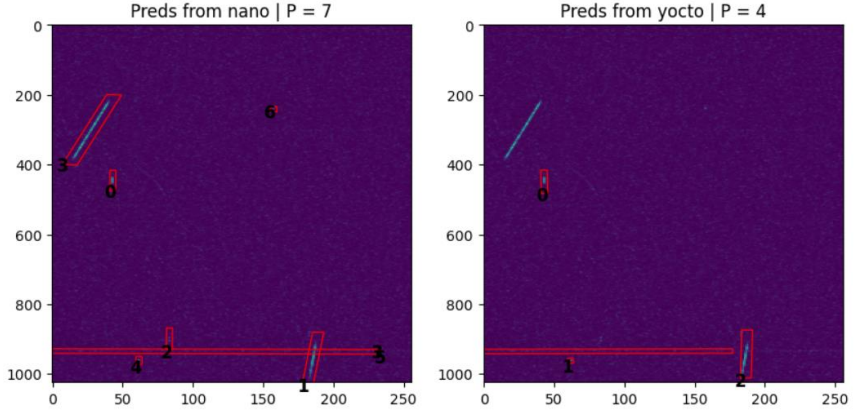


nano vs yocto

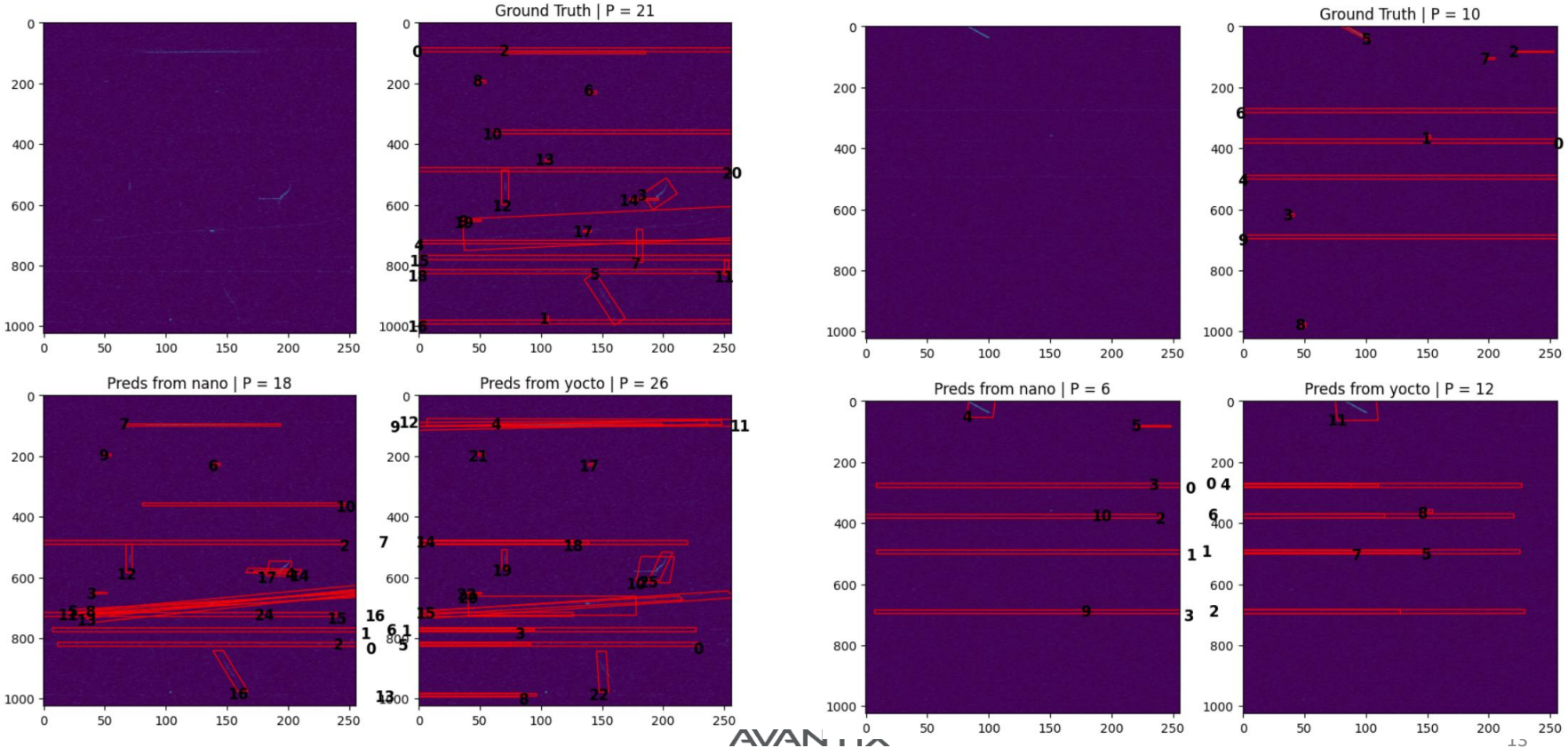




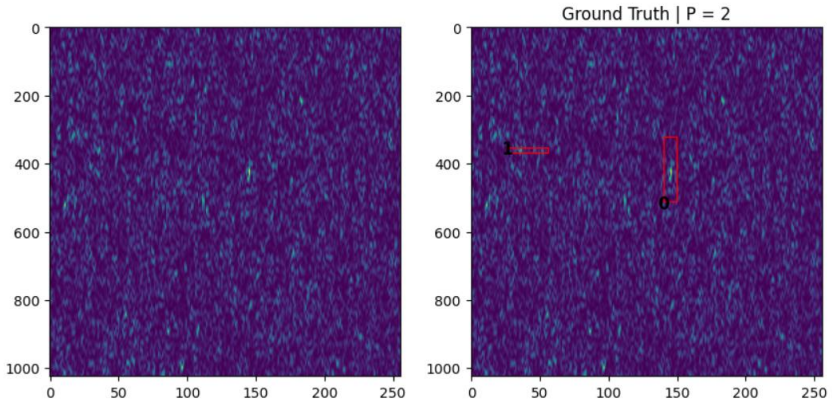


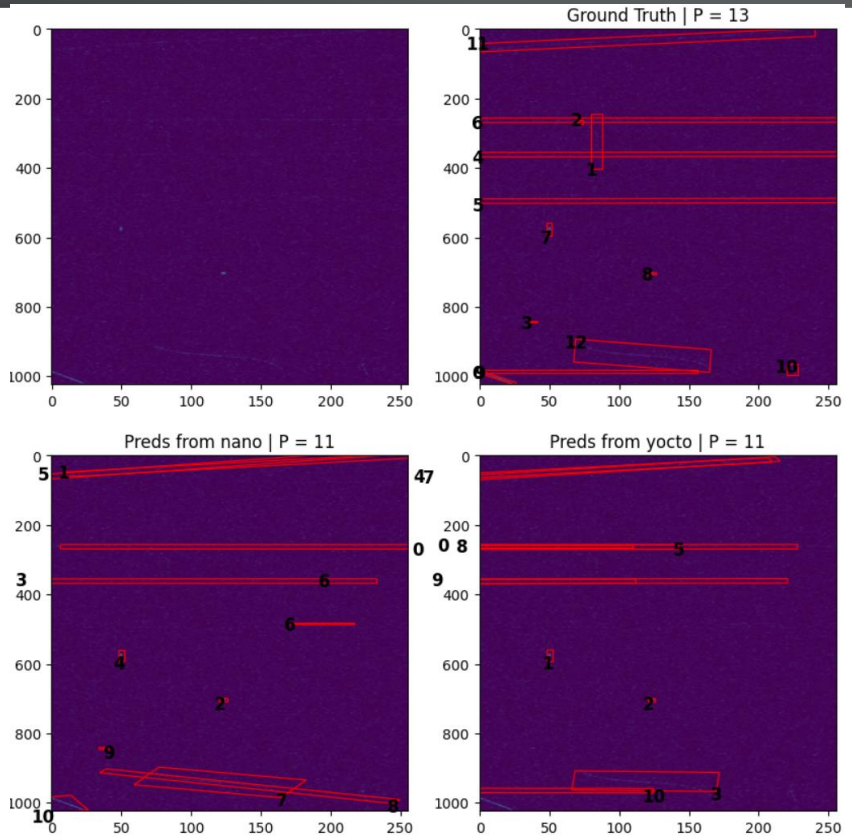


nano vs yocto

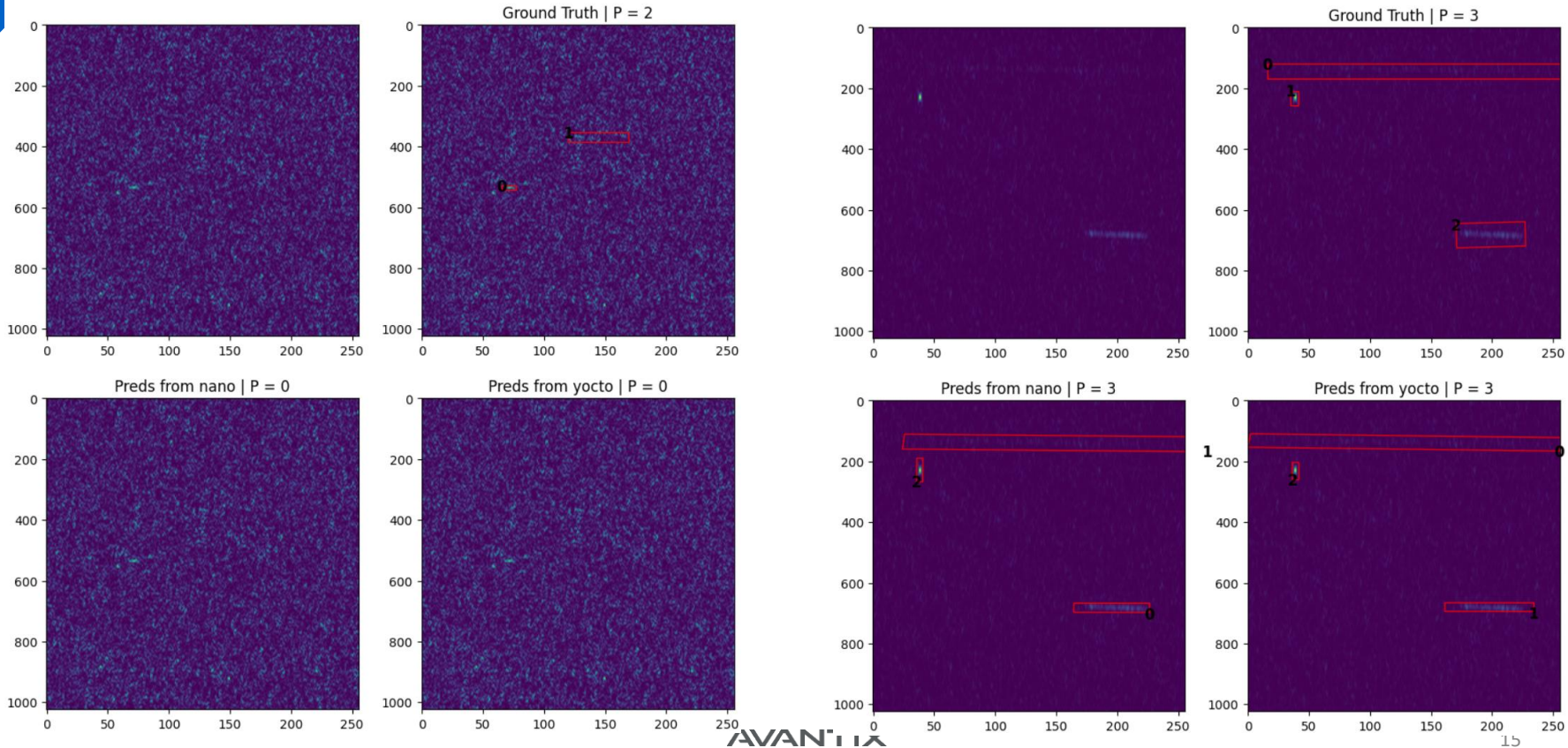


nano vs yocto

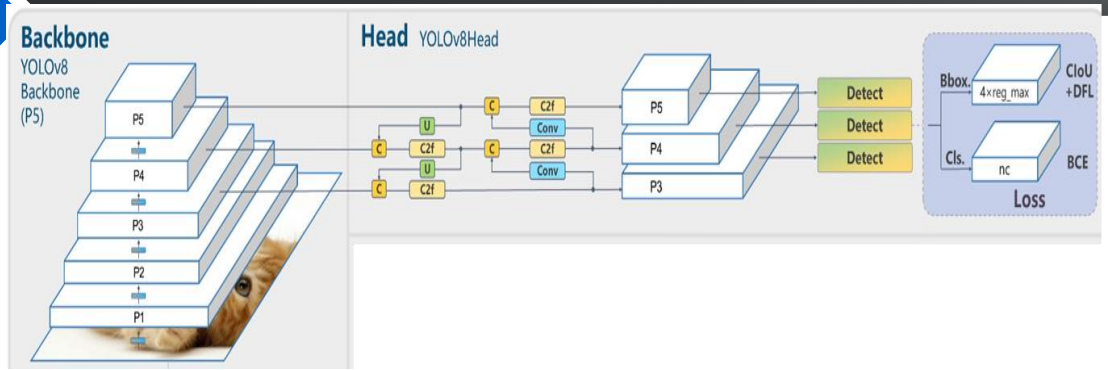




nano vs yocto



Optimisation via réarchitecture



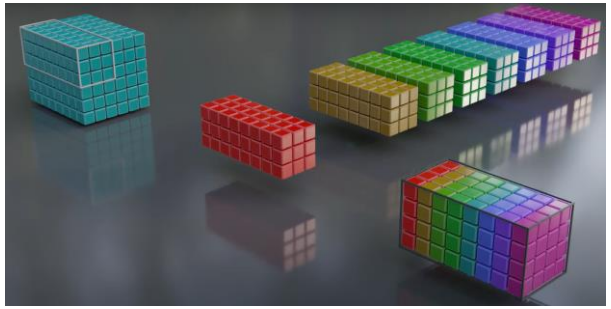
Idée :

1. Evaluer l'utilisation d'une seule couche « detect »
2. Supprimer, remplacer ou réduire en taille les couches les plus complexes du modèle

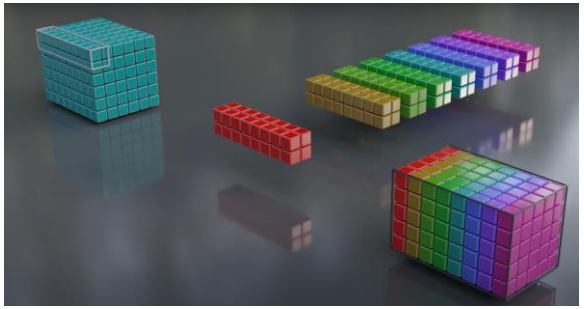
Optimisation via réarchitecture

Kernel size

3x3

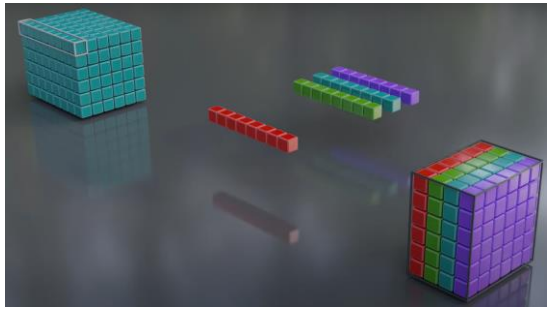


2x2

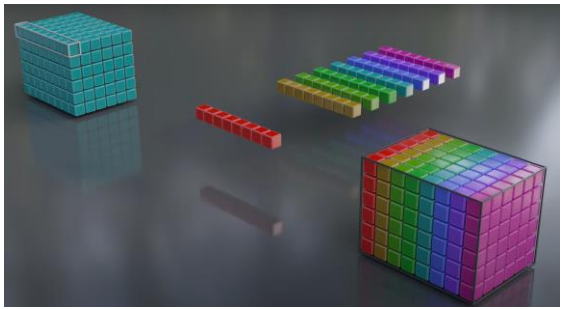


Num filters

Num filters = 3



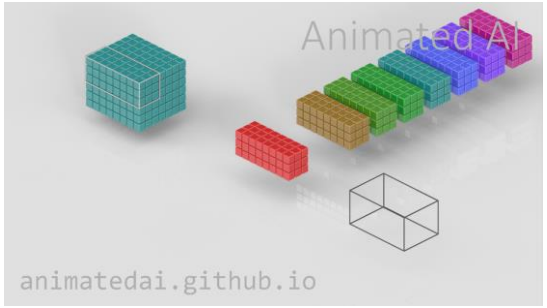
Num filters = 8



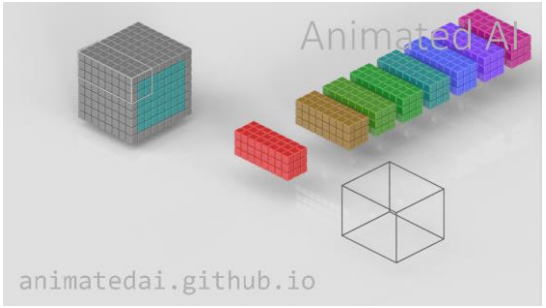
Optimisation via réarchitecture

Padding

Sans

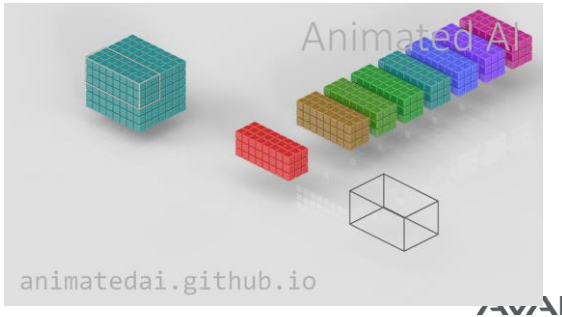


Avec

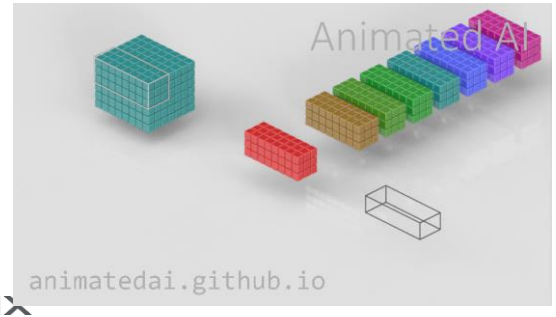


Stride

Stride = 1



Stride = 2



Formules CNN

$$H_{out} = floor\left(\frac{H_{in} + 2 \times padding\ - kernel_{size}}{stride} + 1\right)$$

$$W_{out} = floor\left(\frac{W_{in} + 2 \times padding\ - kernel_{size}}{stride} + 1\right)$$

Contraintes pour optimiser l'inférence :

- Avoir un réseau peu profond
- Eviter d'avoir des colonnes ou lignes inutilisées
- Eviter le padding
- $kernel_{size} = stride$ afin de ne visiter les éléments des tensors qu'une seule fois
- Idéalement avoir $kernel_{size}$ impair
- Grande receptive field car besoin de contexte pour distinguer signal du bruit

AVANTIX

Temps contraint : Inférence via TensorRT

10/01/2024

TensorRT

- Modèle nano crash à la compilation pour cause de VRAM insuffisante
- Modèle yocto compile mais crash à l'inférence

Semble être causé par un missmatch des versions de PyTorch, du driver NVIDIA et de CUDA → En StandBy

AVANTIX

MERCI