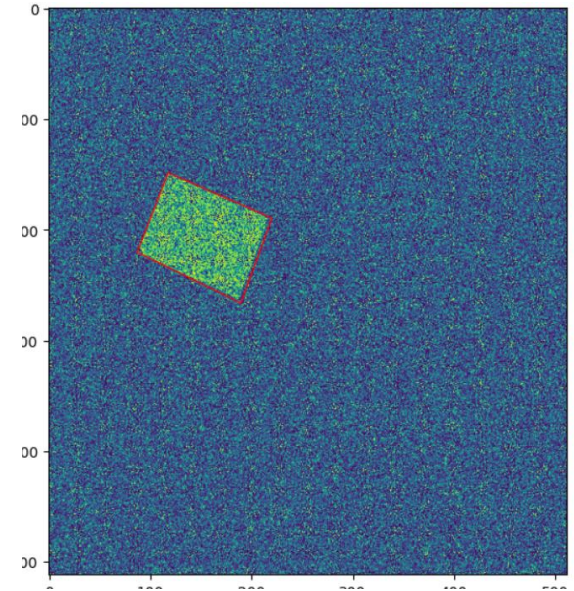
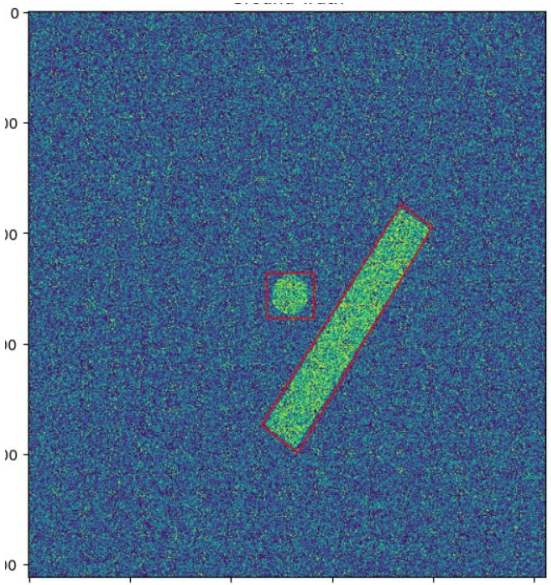
AVANTIX

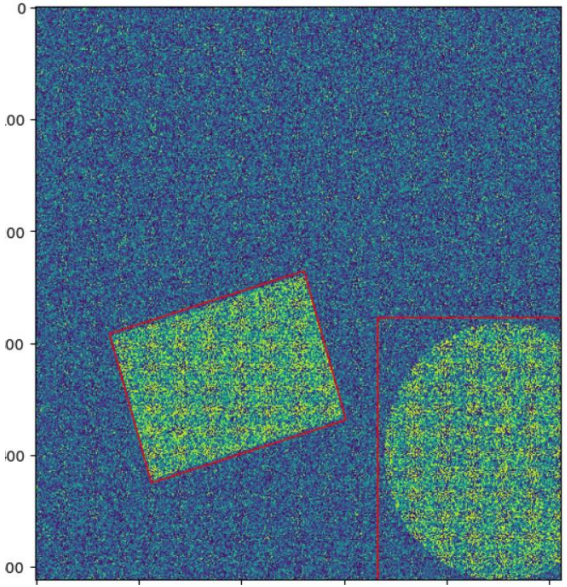
Avancement Simulateur IQ

Simple shape dataset

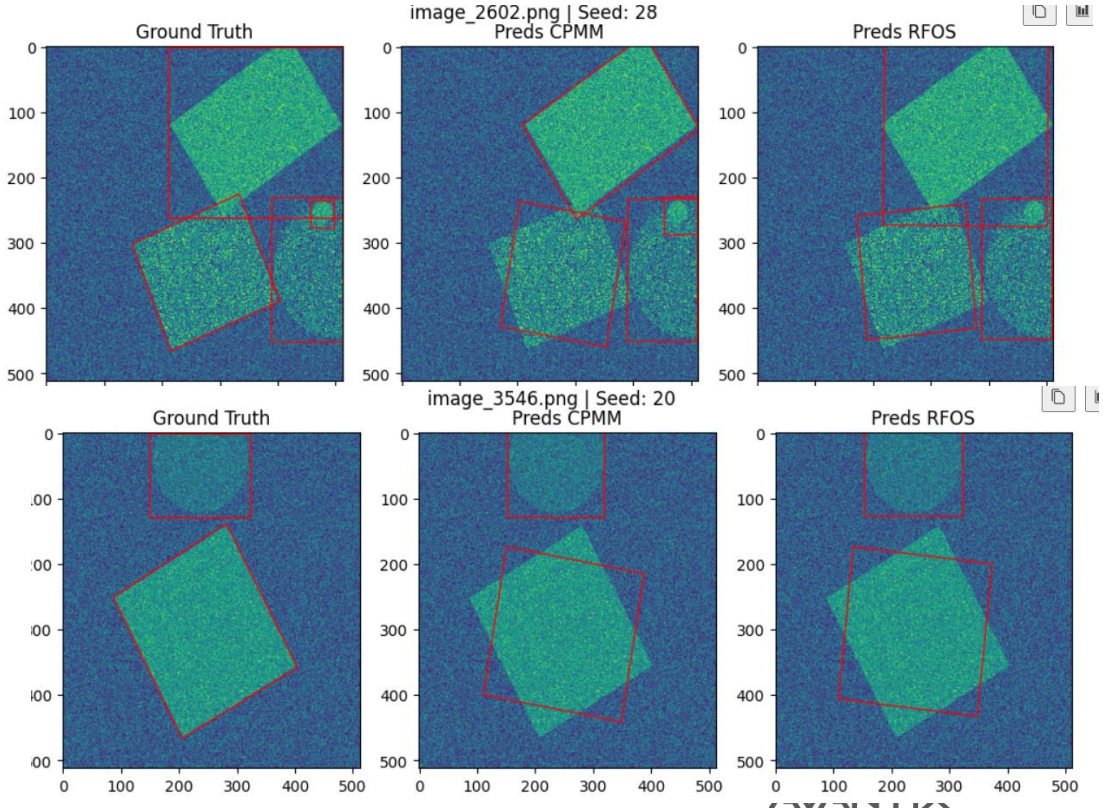
Création d'une BDD avec 2 formes simples : rectangle + cercle
BDD me permettant de plus facilement rétro-ingéniérer le YOLOv8
Tous les tests ont été réalisés sur un YOLOv8 : nano, P3->P5, HP par défaut







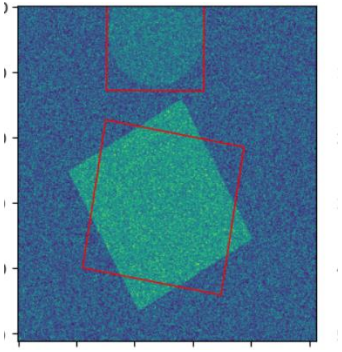
Cas bord d'images



Bug yolo

Bug yolo encore …
Il ne semble être présent que pour une orientation donnée

Bug YOLO orientation



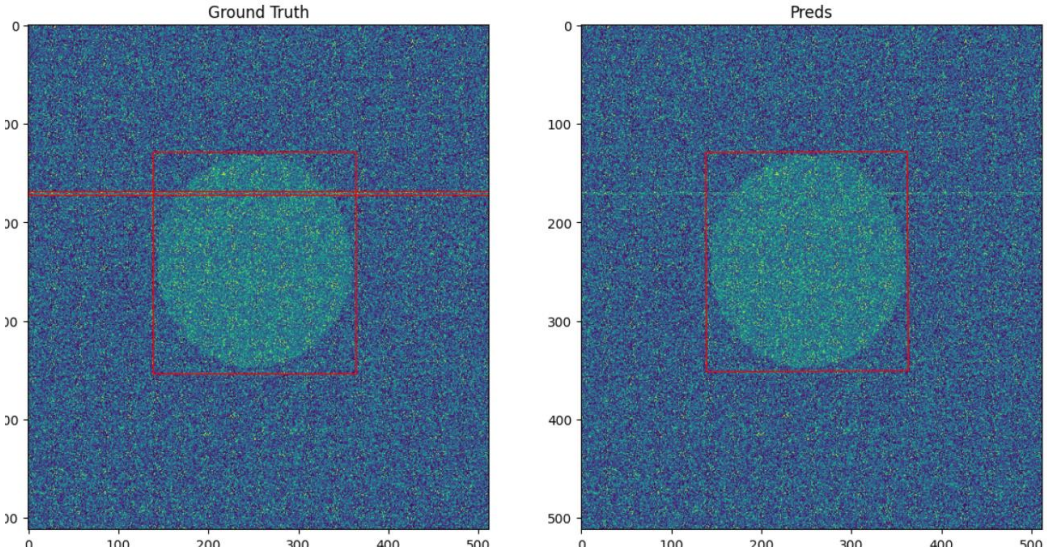
- Tous les bugs ont les coordonnées qui tournent dans le sens trigo mais ils existent des cas non bug qui tournent également dans le sens trigo
- Il y a plus de corrects que de bug pour l'orientation trigo
- Tous les bugs ont la première coordonné en BottomLeft (BL) ou BottomRight (BR)
- Possible que ça se résolve par HP tuning et/ou un apprentissage plus long avec un meilleur learning rate
- Vu des coquilles dans le framework sur conversion des coordonnées → J'ai fait remonter au lead dev

Non détection (ND) CW

Folder : cw_added

Génération d'une BDD avec des rectangles horizontaux, fins et longs.

Confirmation les CW ne sont pas détectés



ND CW

Ce qui a été testé :

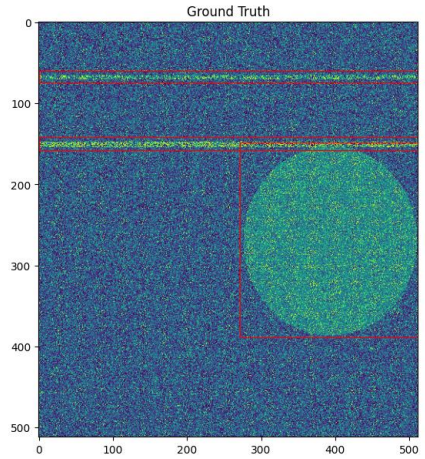
- **HP reg_max → pas d'influence sur les 3 valeurs testées : 16 (défaut), 1 et 256**
- **Split des CW → Fonctionne mais nécessite d'avoir 2 modèles**
- **Bbox plus large → Détection rognée → Solution retenue**

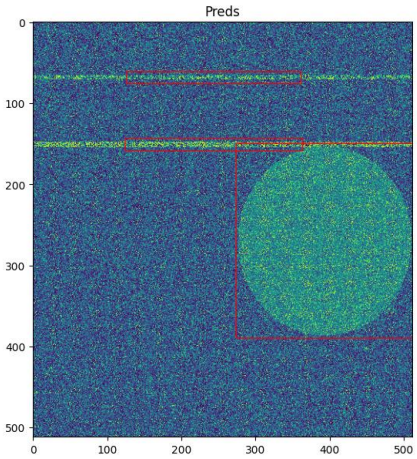
ND CW : Solution par élargissement BBOX

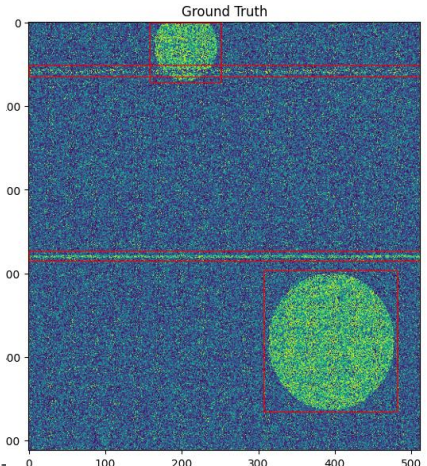
Folders : cw_enlarged_8_pix

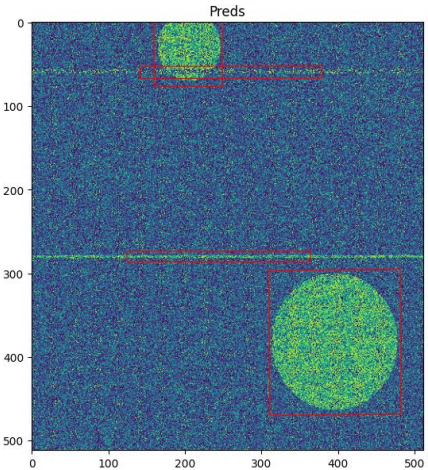
Essai élargissement de 8 pixels des BBOX des CW → Détection des CW mais fort rognage en largeur

Solution : Post traitement des BBOX fins et longs détectées afin de reconnaitre si CW ou non





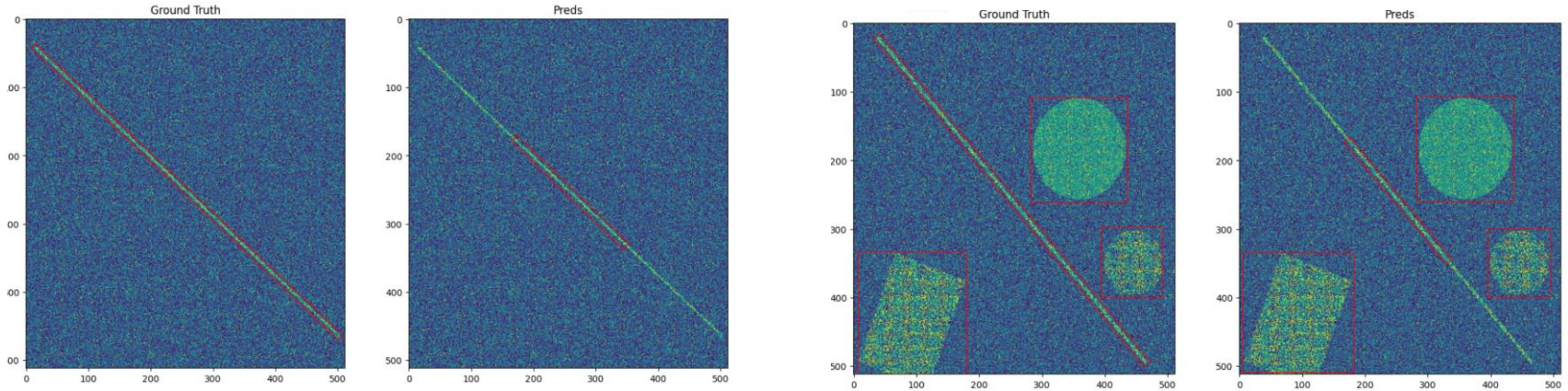




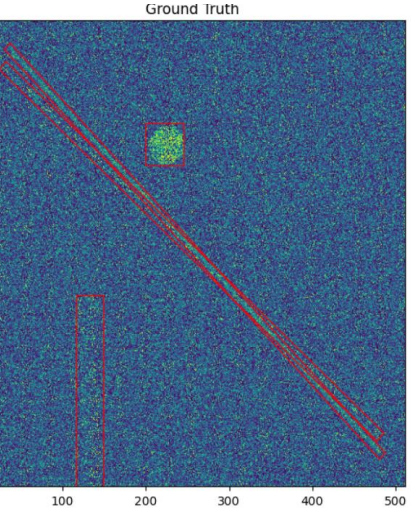
FMCW

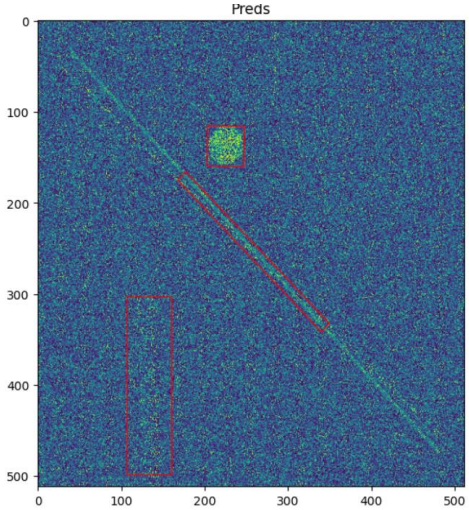
Folder : cw_&_fmcw

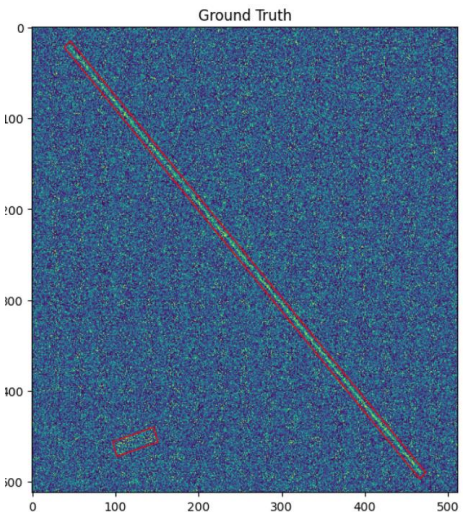
Vérification du comportement de YOLO avec des FMCW → Analogue au CW

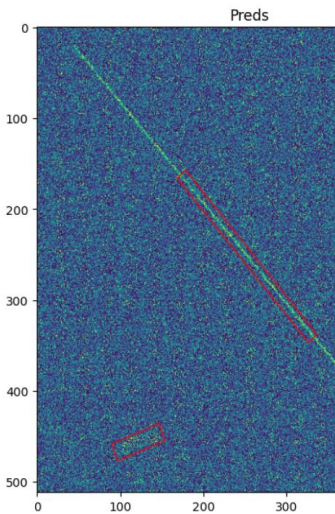


FMCW

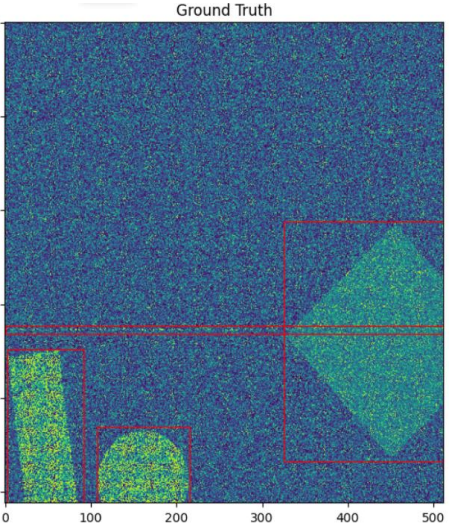

Ground Truth
100
200
300
400
500

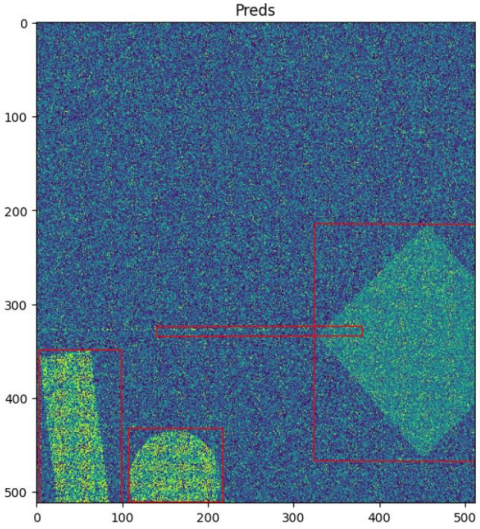

Preds
0
100
200
300
400
500
0
100
200
300
400
500

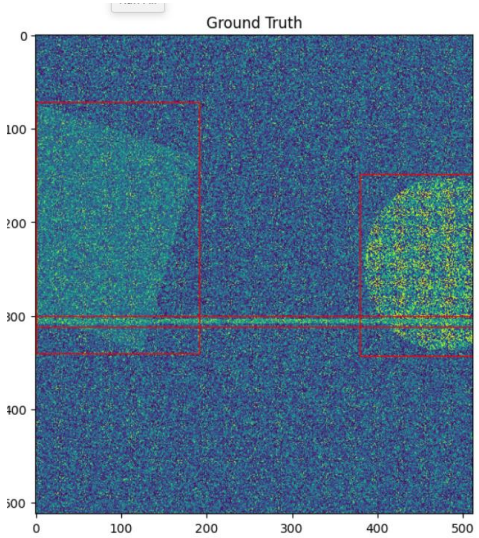

Ground Truth
0
100
200
300
400
500
0
100
200
300
400
500

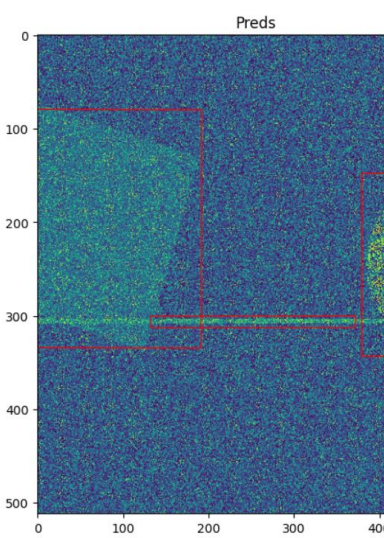

Preds
0
100
200
300
400
500
0
100
200
300

FMCW

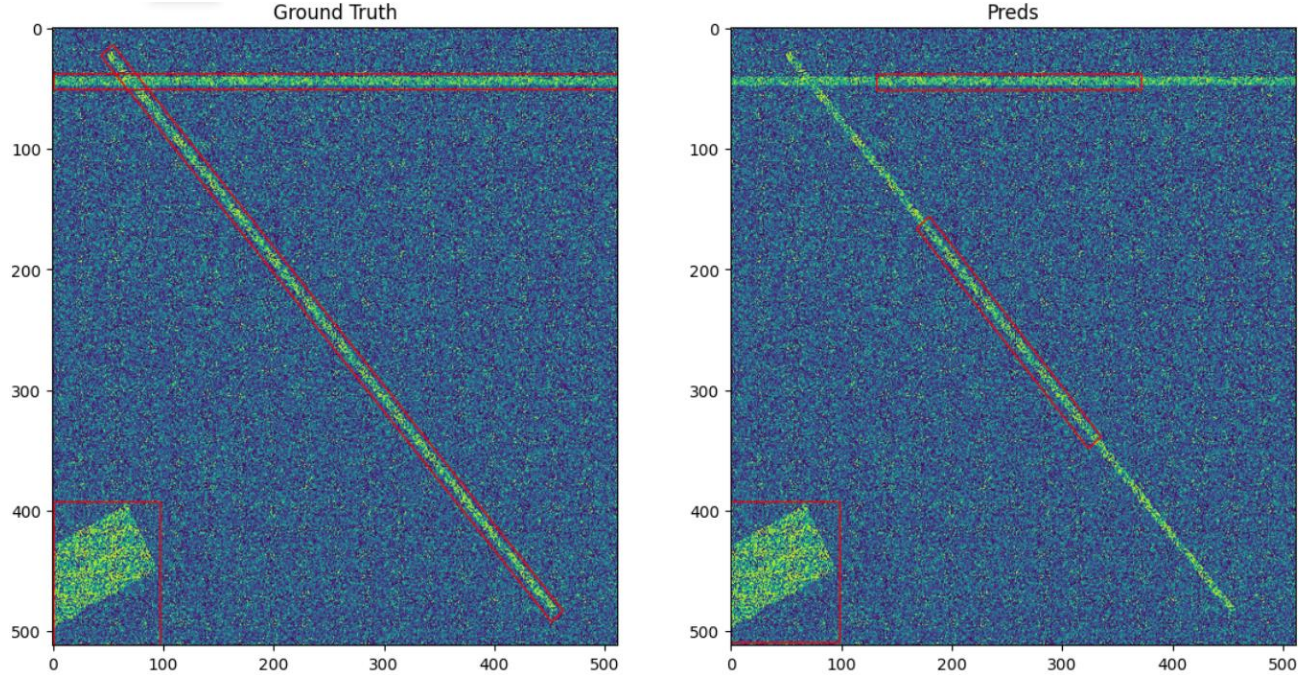

Ground Truth
0
100
200
300
400
500


Preds
0
100
200
300
400
500


Ground Truth
0
100
200
300
400
500


Preds
0
100
200
300
400
500

FMCW



Petits + gros objets

Folder : all_scales_shapes_large_p3_p5 & all_scales_shapes_nano_p2_p5**

Génération d'une très grande BDD en piochant dans toutes les configurations d'objets possibles afin de vérifier la robustesse de YOLO.

Toujours ayant au maximum 4 objets par image

~~YOLOv8 large + HP tuning + P3 -> P5~~ : Abandonné car trop long
YOLOv8 nano + HP tuning + P2 -> P5

Grace aux 2 runs précédents, on remarque que même sur un problème aussi simple la version large surpasse la version nano

Large P3P5

epoch,	train/box_loss,	val/box_loss,
1,	0.93027,	0.83108,
2,	0.73255,	0.74052,
3,	0.67391,	0.67554,
4,	0.62643,	0.64015,
5,	0.60052,	0.64003,
6,	0.58303,	0.6067,
7,	0.56703,	0.6074,
8,	0.55627,	0.57276,

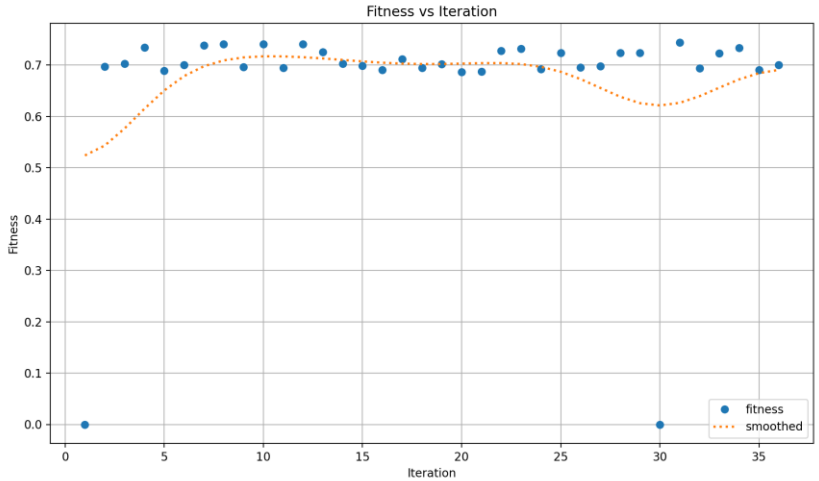
Nano P2P5

epoch,	train/box_loss,	val/box_loss,
27,	0.66663,	0.77157,
28,	0.66423,	0.77055,
29,	0.66029,	0.77021,
30,	0.65811,	0.7696,
31,	0.65383,	0.7694,
32,	0.65235,	0.76885,
33,	0.65089,	0.76854,
34,	0.64713,	0.768,
35,	0.64219,	0.76745,

Petits + gros objets : HP tuning

Folder : all_scales_shapes_large_p3_p5 & all_scales_shapes_nano_p2_p5**

Après avoir étudié les HP de YOLO, j'ai lancé sur les HP que je considère pertinent et ce qu'on remarque c'est que le choix des HP non fixés n'a pas d'incidence sur les performances du modèle.



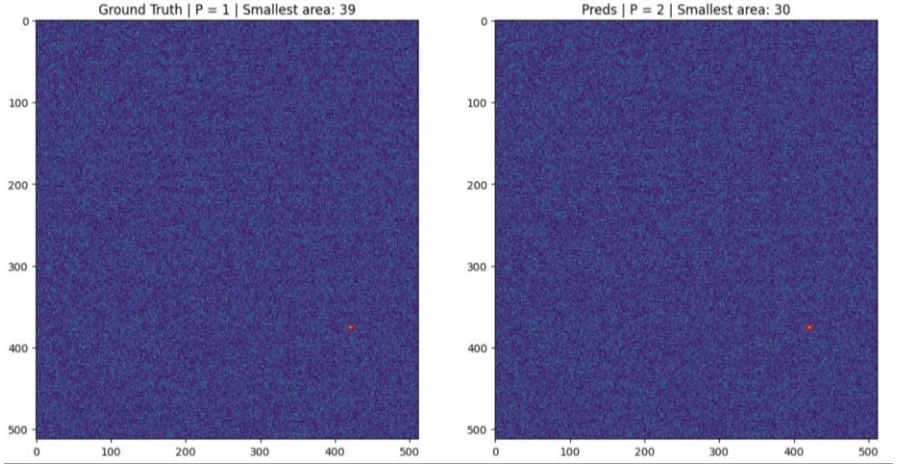
Petits + gros objets : Stats

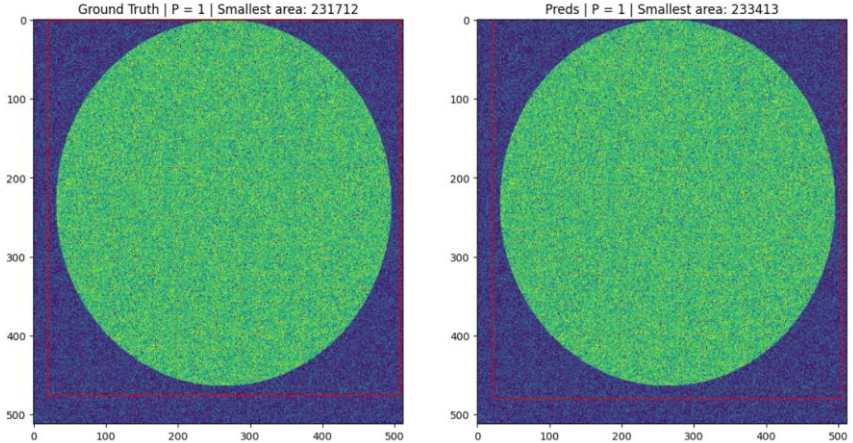
Folder : all_scales_shapes_large_p3_p5 & all_scales_shapes_nano_p2_p5**

Plus petit objet détecté : 30 pixels (vs plus petit objet présent : 2 pixels)

Plus gros objet détecté : 233413 pixels (quasiment 512x512 = 262144 pixels)

Conclusion : Il ne faut pas compter sur la fft optimale pour détecter les signaux à très bas SNR optimal. D'ailleurs pour ces cas, même avec une méthode par seuil il ne serait pas possible de les différencier du bruit.



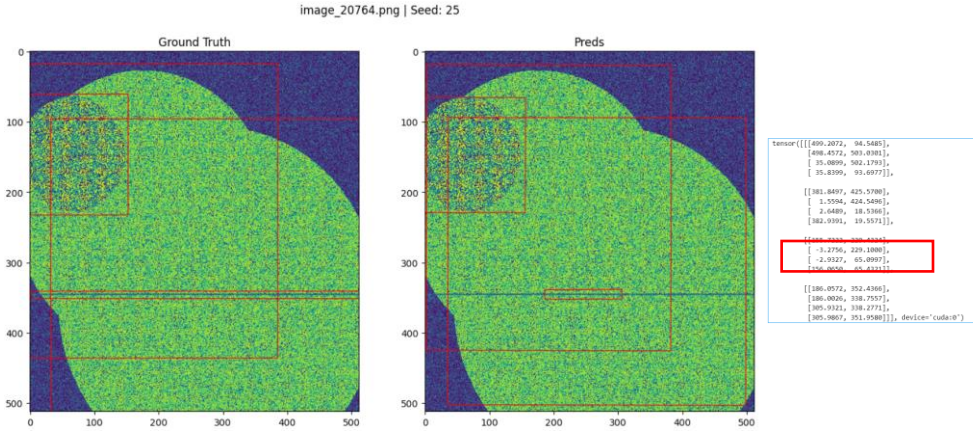


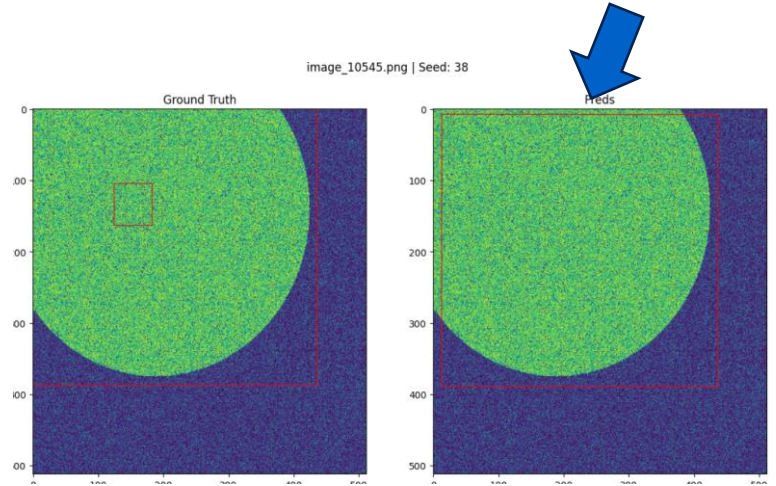
Gestion des coordonnées à l'extérieur de l'image

Folder : all_shape_p2_p5_outside_coord

Après avoir commenté les parties qui exclus les coordonnées à l'extérieur :

- L'apprentissage se passe sans bug
- On peut même voir des prédictions de coordonnées négatives (voir fig à gauche)
- Cependant pour certains gros objets, le modèle est réticent à avoir des coordonnées extrêmes (voir fig à droite)





Labelizer

Modification de labelizer au vu des résultats sur simples shapes :

- Augmenter la hauteur des bboxes des CW et FMCW. Par défaut à 8 pixels
- Borne inférieure sur w et h. Par défaut à 4 pixels
- Coords tournent dans le sens CW
- Borner les coords à 1,05x la définition de l'image → Trouver barycentre (x,y) et réduire w & h en conservant l'angle

BT1.0.2

Paramètres :

nperseg = [128, 256, 512]
noverlap = nperseg / 2
nfft = 512

fc_rec = 2.5 GHz
fs_rec = [4, 20, 100, 500, 1e3, 4e3] MS/s
dt_rec = 513 * nperseg / (2*fs_rec)
sigma_noise = 1mV
sig_types = [[‘radar’], [‘radar’, ‘cellular’]]
snr_db_opt_radar : friend / no friend
snr_db_opt_com = 35 dB
snr_db_eff = -2 dB

100k enregistrements

DI (ns)	Sans Modulation	Modulations de Fréquence (MHz/µs)		Modulations de Phase Phase / Nombre de moments
10-100	Oui	-	-	-
100-1_000	Oui	-	-	-
1_000-10_000	Oui	20	100	180° 7, 8
10_000-100_000	Oui	2	10	180° 7, 8, 13, 25, 33, 39 60° 25, 30 45° 8, 35, 70
100_000-1_000_000	Oui	10^{-3}	1	180° 7, 8, 13, 25, 33, 39 60° 25, 30 45° 8, 35, 70
1_000_000-10_000_000	Non	0,001	0,1	-
10_000_000-100_000_000	Non	0,0001	0,01	-
100_000_000-inf	Non	0,00001	0,001	-
CW	Oui	-	-	-

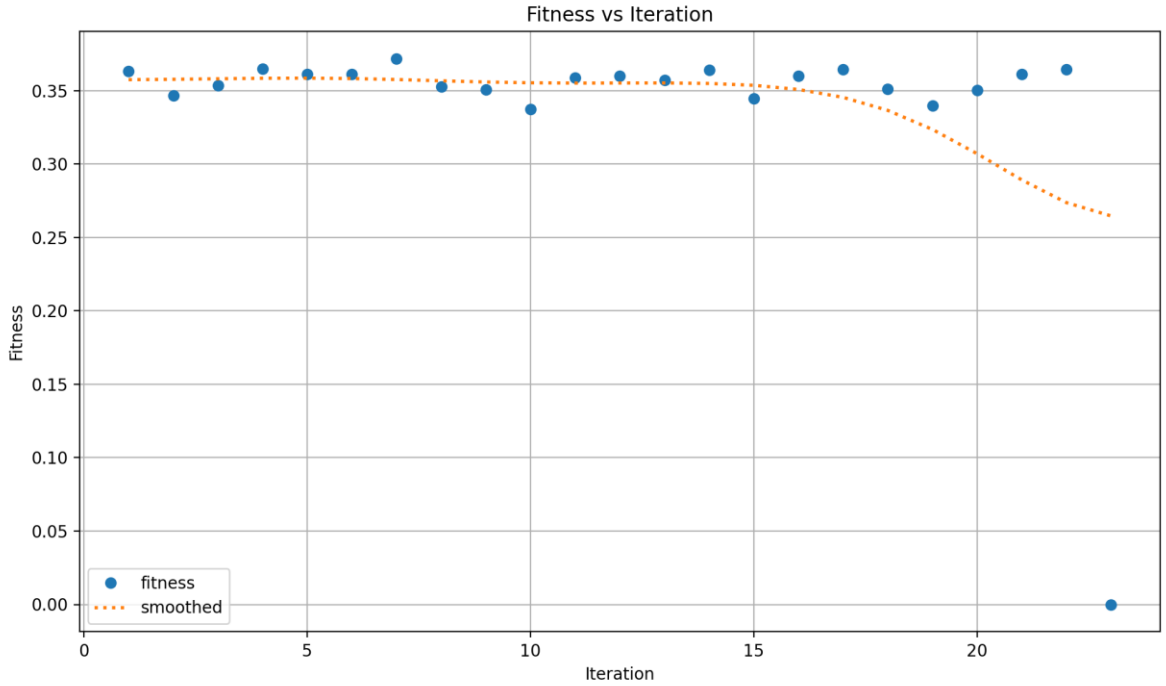
Meta-formes d’ondes

4 classes

- Petits objets (< 20 pixels)
- NO-MOD + CW + P1 + Frank (qq fois doucle chirp) + P4 + Biphasique
- FMOD + FMCW + P2(ressemble au symbole %)
- P3 (double chirp)

HP tuning sur BT102

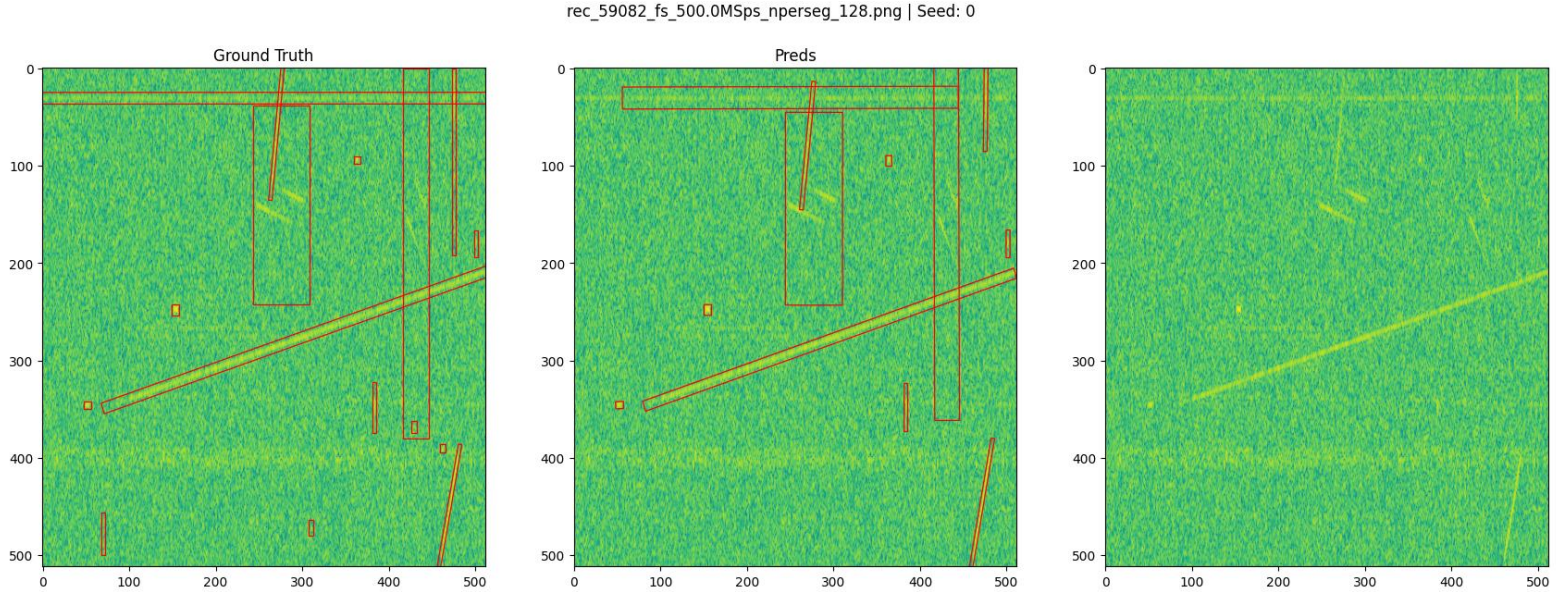
Comme pour les formes simples → aucun impact



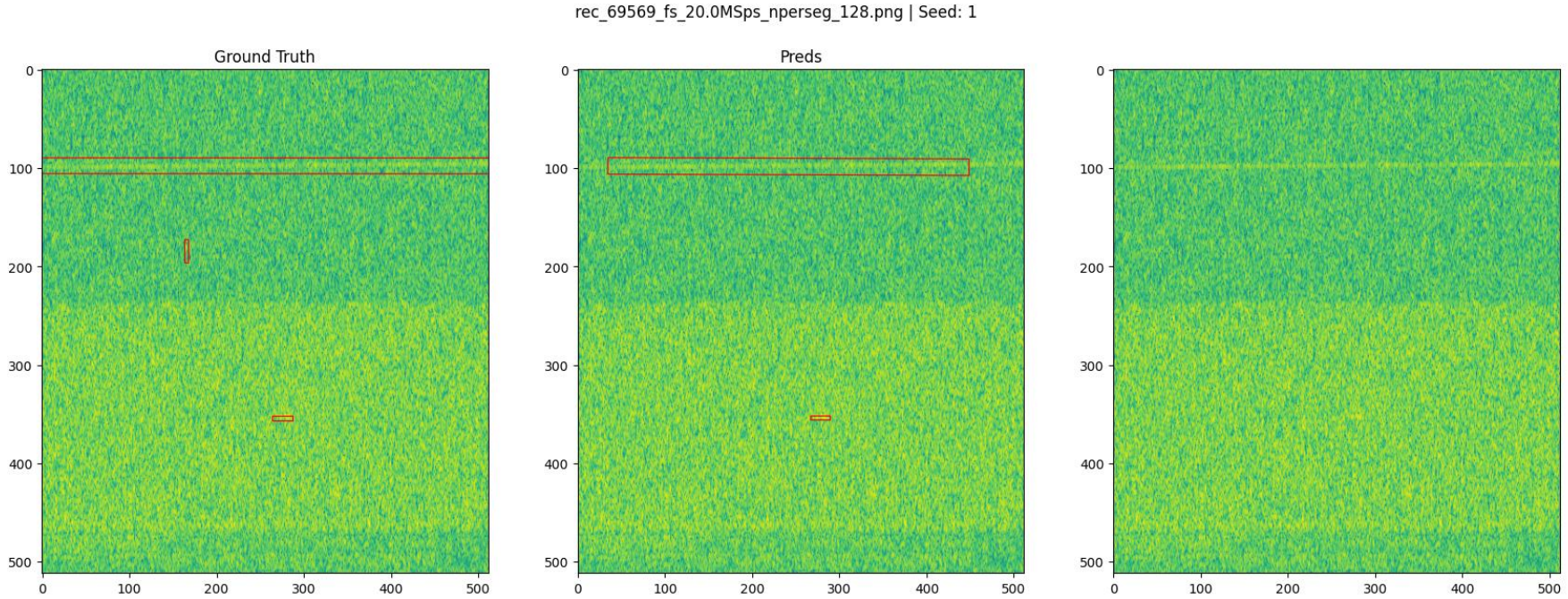
Résultats BT102

- ND de signaux très peu énergétiques
- Détecte des signaux dans de l'intérférence COM
- Cas de multi détection
- CW traversé par impulsions = ND des 2
- Chirp avec mauvais angle et multi détection en présence d'interférences
- Aucun FP si on ne considère pas les multi-détections
- ND de certains FMCW
- Cas complexe quasi parfait

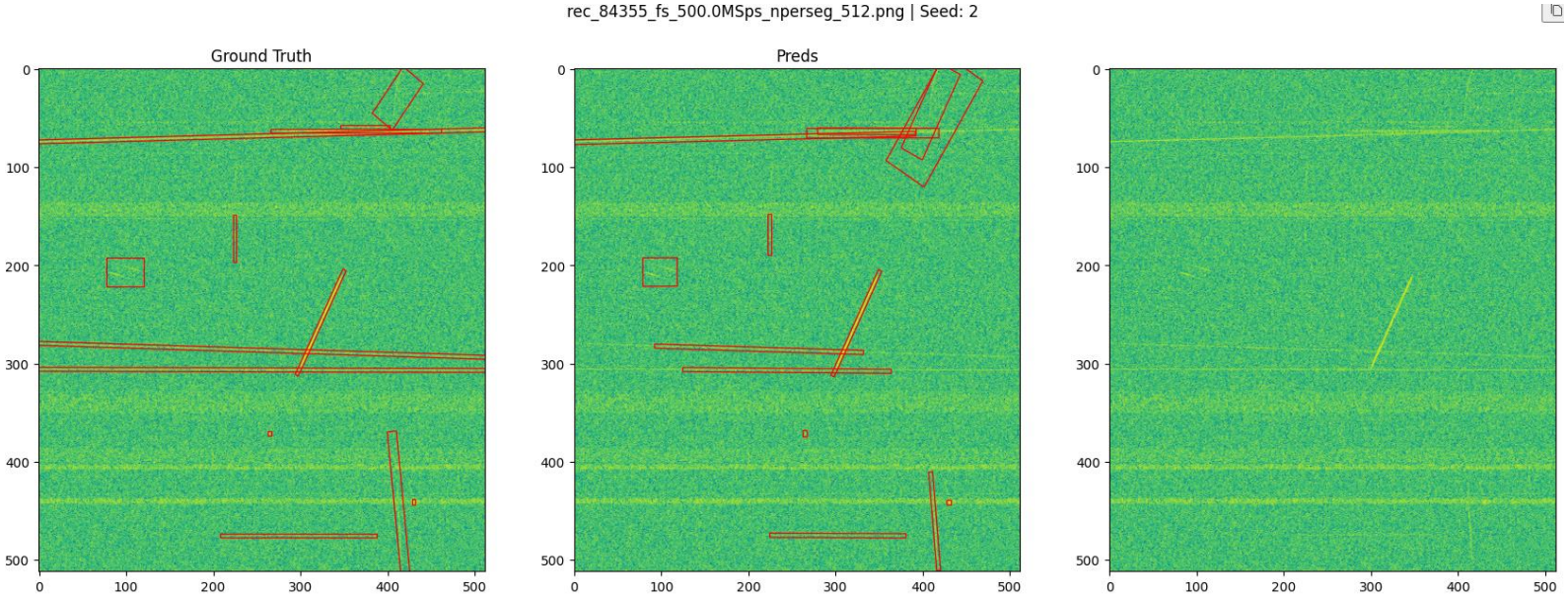
Résultats BT102 : ND peu énergétique



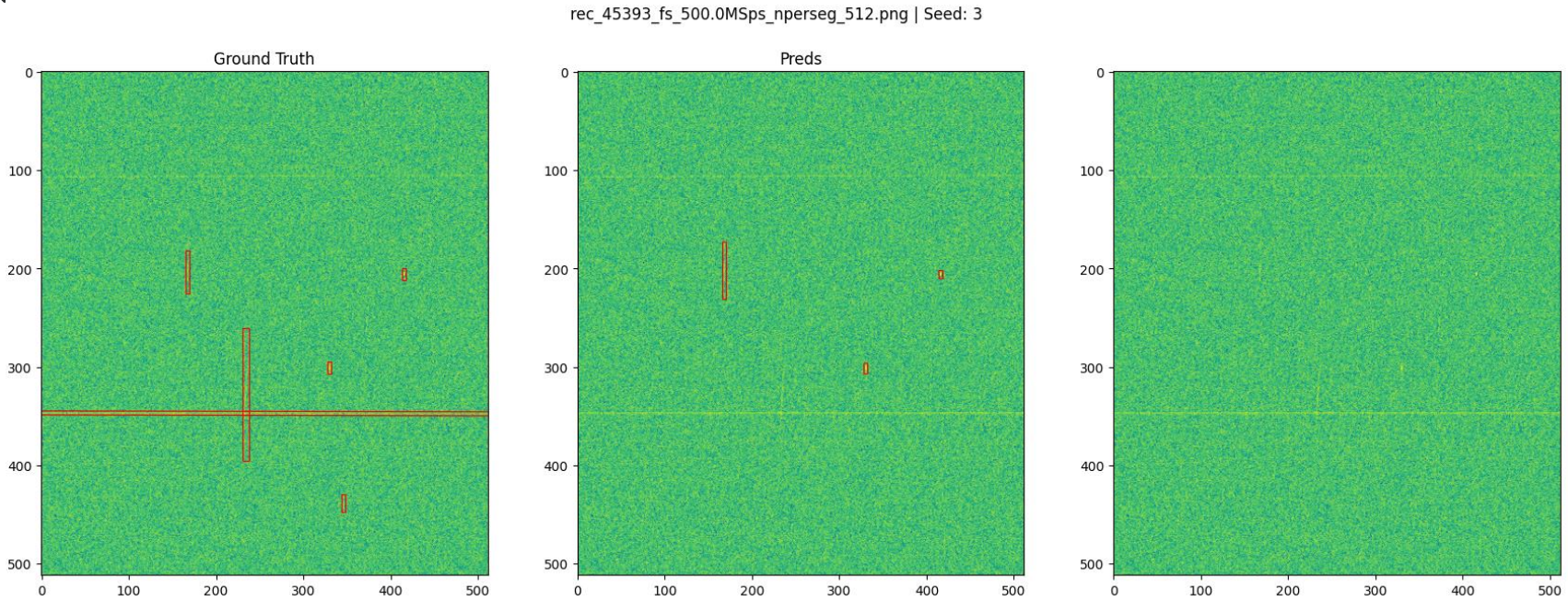
Résultats BT102 : intérférence COM



Résultats BT102 : multi-détection

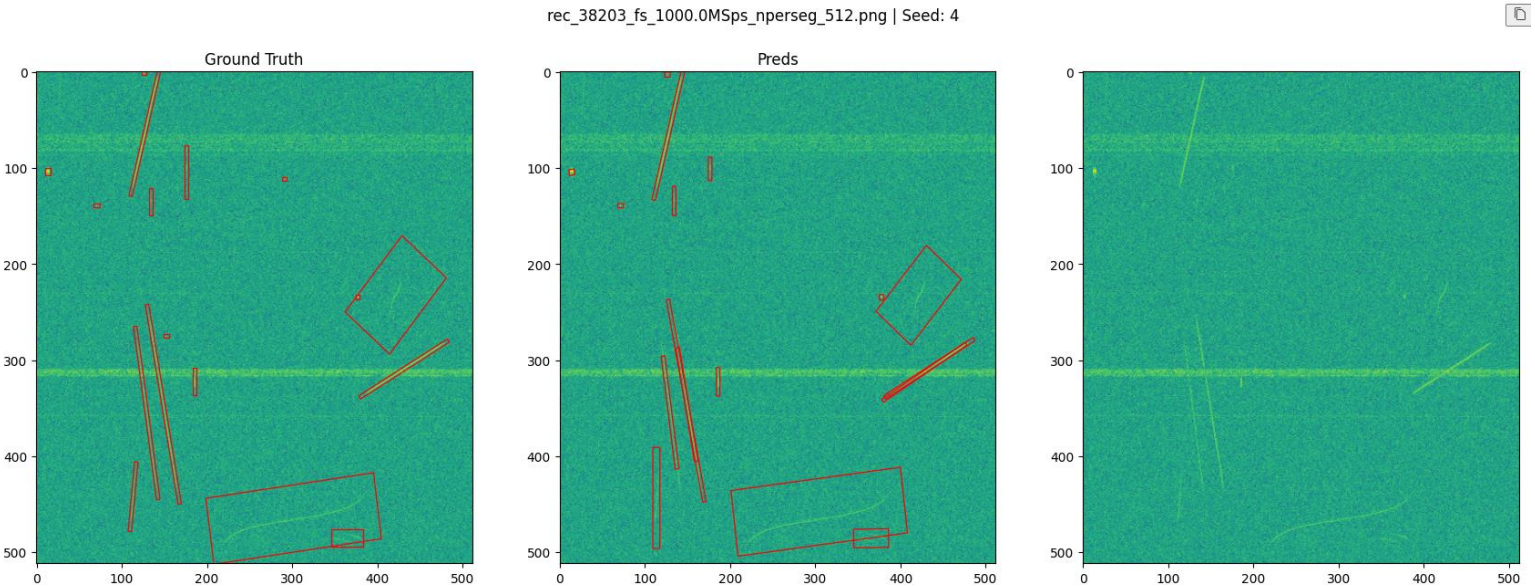


Résultats BT102 : ND CW + impulsion

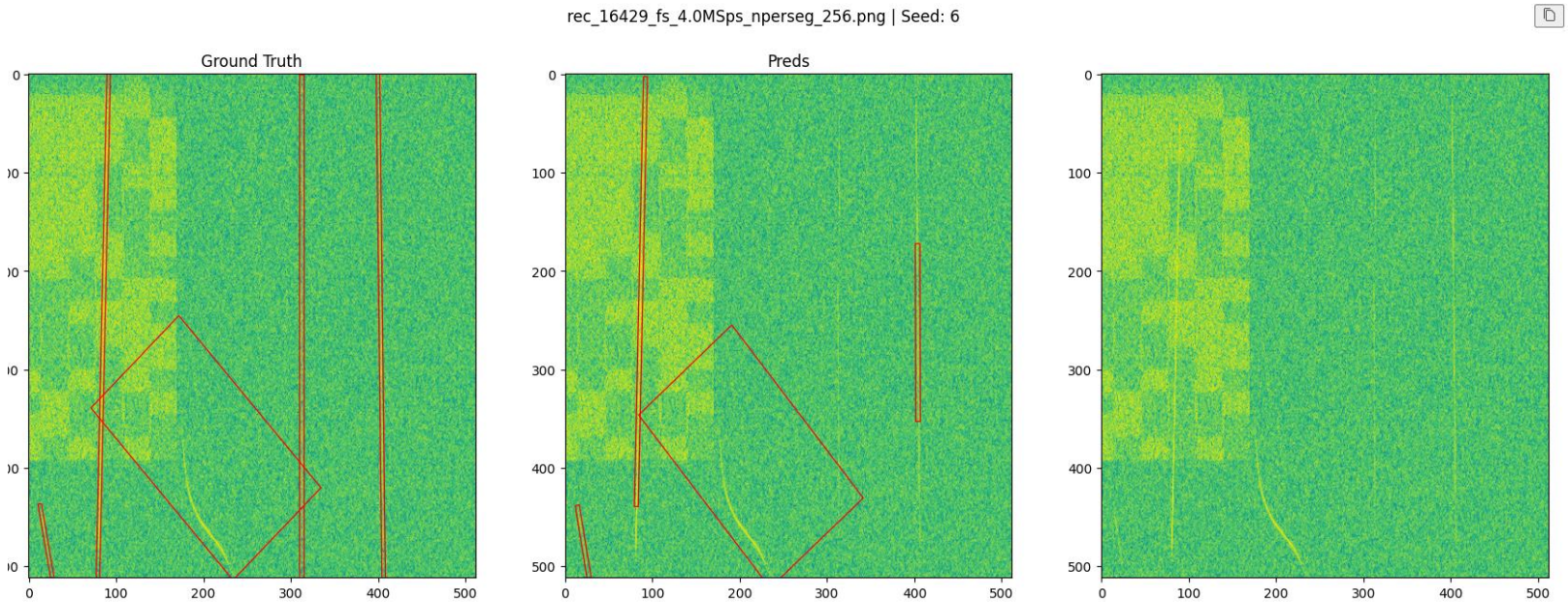


Lorsqu'une impulsion traverse un CW, le CW est souvent ND

Résultats BT102 : Chirp angle erroné + multi détection

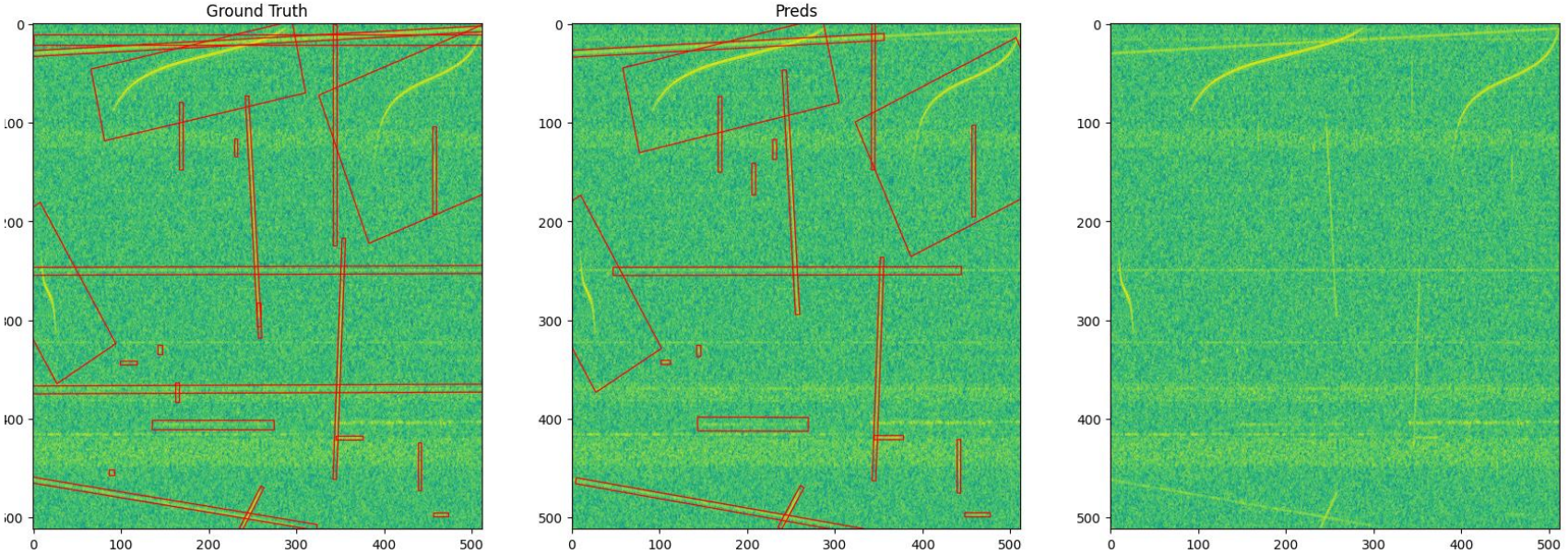


Résultats BT102 : FMCW ND



Résultats BT102 : Cas complexe quasi parfait

rec_58236_fs_500.0MSps_nperseg_256.png | Seed: 12

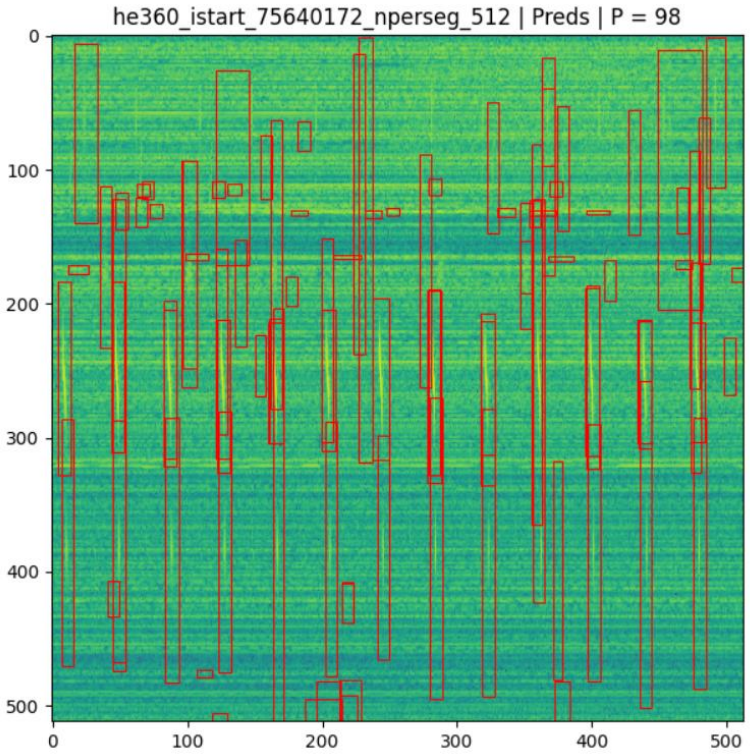


Résultats BT102 : Stats

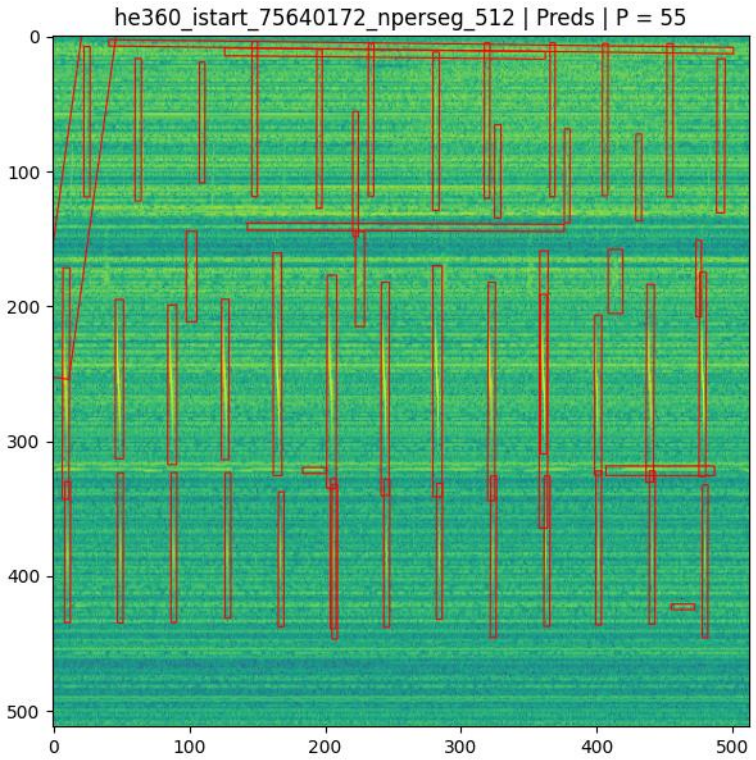
TODO

Résultats HE360 sur YOLOv8-OBB nano

Avant (14/02)



Aujourd'hui



Bug YOLOv8

Apprentissage instable : Bbox Loss qui diverge → A débuguer

Underfitting présent → Plus gros modèle après résolution du pb précédent

YOLOv9

https://arxiv.org/pdf/2402.13616.pdf

Résout le problème de la perte d'information dans les réseaux profonds en proposant le programmable gradient information (PGI) et le generalized ELAN (GELAN). Cela a permis d'obtenir un nouvel état de l'art des perfs sur MSCOCO.
Des branches sont ajoutées durant l'apprentissage uniquement ne pénalisant pas le temps d'inférence.

Table 1. Comparison of State-of-the-Art Real-Time Object Detectors

Model	size (pixels)	AP^{val} 50-95	AP^{val} 50	AP^{val} 75	params (M)	FLOPs (B)
YOLOv9-S	640	46.8	63.4	50.7	7.2	26.7
YOLOv9-M	640	51.4	68.1	56.1	20.1	76.8
YOLOv9-C	640	53.0	70.2	57.8	25.5	102.8
YOLOv9-E	640	55.6	72.8	60.6	58.1	192.5

Detection (COCO) | Detection (Open Images V7) | Segmentation (COCO) | Classification (ImageNet) | Pose (CO >

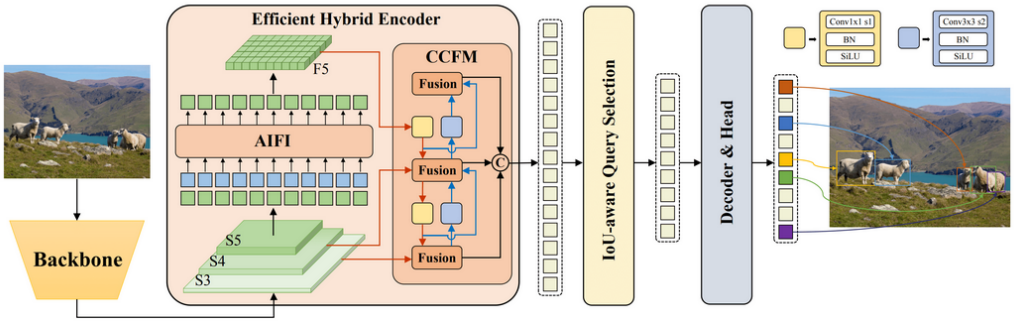
See Detection Docs for usage examples with these models trained on COCO, which include 80 pre-trained classes.

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

RT DETR

Overview

Real-Time Detection Transformer (RT-DETR), developed by Baidu, is a cutting-edge end-to-end object detector that provides real-time performance while maintaining high accuracy. It leverages the power of Vision Transformers (ViT) to efficiently process multiscale features by decoupling intra-scale interaction and cross-scale fusion. RT-DETR is highly adaptable, supporting flexible adjustment of inference speed using different decoder layers without retraining. The model excels on accelerated backends like CUDA with TensorRT, outperforming many other real-time object detectors.



Modèle intéressant pour faire du temps-réel **mais n'a pas (pour le moment) d'implémentation OBB**

Est beaucoup moins considéré car il nécessite un gros préapprentissage or ce n'est pas un frein pour notre cas d'usage.

AVANTIX

MERCI