AVANTIX

Avancement Simulateur IQ

BT1.0.0

Paramètres :

```
nfft = [128, 256, 512]
noverlap=int(nfft/2)
fc_rec = 5 GHz
fs_rec = [1, 5, 25, 100, 500, 1e3, 3e3] MS/s
dt_rec = 640*noverlap/fs_rec
sigma_noise = 1mV
sig_types = [[‘radar’], [‘radar’, ‘cellular’]]
snr_db = 3 à 40dB
Snr_db_eff > 8dB
```

A date : 16k enregistrements

Objectif : 50k enregistrements

FPS à obtenir

Img size = 512 x 512
Dt = 1s
Fs = 4GS/s
Overlap = 50%
Nffts = 128 | 256 | 512 | 1024 | 2048 | 8k | 32k | 131k

FPS = 366 094 → 2,7 µs / image !!!

```
fs = 4e9
n_samples = fs * 1 # 1 s
nb_imgs = 0

for nperseg in [128, 256, 512, 1024, 2048, 2**13, 2**15, 2**17]:
    noverlap = nperseg//2
    nfft = nperseg if nperseg >= 512 else 512

    width = int((n_samples - noverlap) / (nperseg - noverlap)) // 512
    height = (nfft // 512)
    nb_imgs_specific_fft = width  * height
    print(f"nfft = {nperseg} | nb_imgs per s = {nb_imgs_specific_fft}")
    nb_imgs += nb_imgs_specific_fft

nb_imgs
```

✓ 0.0s

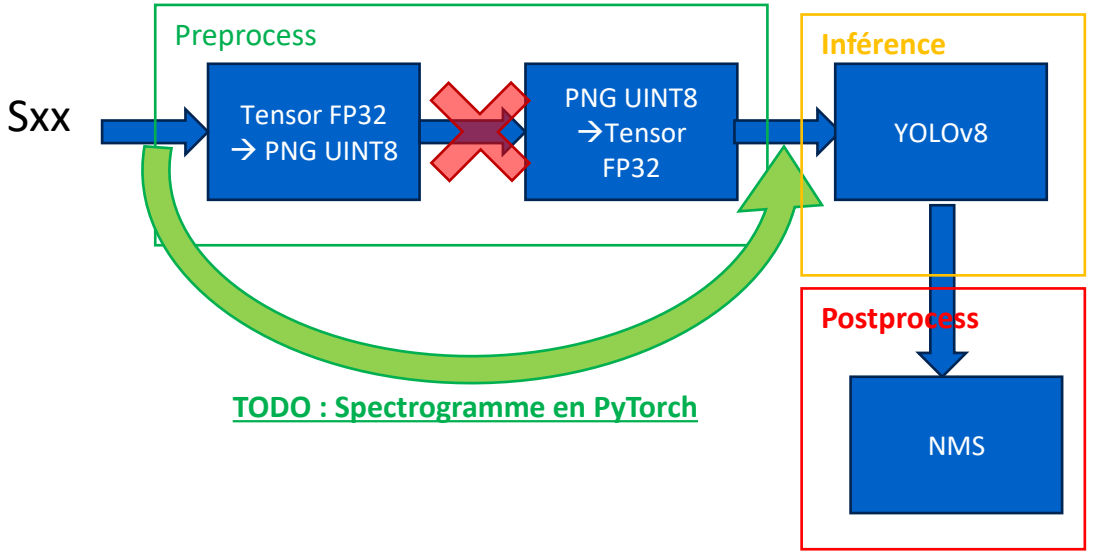
```
nfft = 128 | nb_imgs per s = 122070
nfft = 256 | nb_imgs per s = 61035
nfft = 512 | nb_imgs per s = 30517
nfft = 1024 | nb_imgs per s = 30516
nfft = 2048 | nb_imgs per s = 30516
nfft = 8192 | nb_imgs per s = 30512
nfft = 32768 | nb_imgs per s = 30464
nfft = 131072 | nb_imgs per s = 30464
```

FPS actuel

FP32
640x640
n_channels=3
YOLOV8 nano

Preprocess = 0ms
Inférence = 1,02 ms
Postprocess = 1,7 ms

→ FPS = 367
Facteur de gain à obtenir = 1000 !



Facteur de gain hypothétique sur l'inférence

SiLU → ReLU : 1
FP32→FP16 : 2
FP16 → Int8 : 2
3 channels → 1 channel : ε (voir effet nc)
V100 → H100 : 16 (H100 vs A100) x 2 (V100 vs A100)
Fusion de 3 FFT en 1 image : 1,2
Réduction de la taille du modèle : 2 ou 4 → si performances conservées
Définition 640 → 512 : 1,5
Suppression des FFT 256 et 1024 : 1,3
Post-process (filtrage des impulsions) = 0,25
Optimisation d'inférence (oneDNN) : 1,3

Validé
Certain
Probable
Inconnu / Estimé

Post-processing : Non-Max Suppression (NMS)

YOLO réalise des multi-détection et le NMS permet de nettoyer cela.

Pseudo-code

0. (Facultatif) Supprimer toutes les bbox ayant un score de proba inférieur à un seuil
1. Récupère la bbox ayant la plus haute proba
2. Supprime toutes les bbox qui overlap la bbox précédente
3. Repartir en 1 sauf si plus de bbox :
    - Itérer sur la classe suivante et repartir en 1
    - quitter le programme

NMS

Peu d'espoir sur une alternative plus rapide au NMS car c'est la version torchvision qui est utilisée dans YOLOv8

UPDATE: THESE ARE OLD RESULTS, SEE BOTTOM OF THREAD FOR IMPROVED RESULTS

	Speed mm:ss	COCO mAP @0.5...0.95	COCO mAP @0.5
ultralytics `'OR'`	8:20	39.7	60.3
ultralytics `'AND'`	7:38	39.6	60.1
ultralytics `'SOFT'`	12:00	39.1	58.7
ultralytics `'MERGE'`	11:25	**40.2**	**60.4**
torchvision.ops.boxes.nms()	**5:08**	39.7	60.3
torchvision.ops.boxes.batched_nms()	6:00	39.7	60.3

Par contre, possibilité de l'accélérer en choisissant des seuils plus sévères.

→ **Etudier l'impact sur les perfs de seuils plus élevés**

NMS : Seuil

TODO

NMS : Union des bbox

TODO

FP32 vs FP16

BT 0.1
FP32 : Train sur 50 epochs
FP16 : Quantification des inputs et du modèle

Precision	mAP@0,5 sur Test	Inférence ms/img
FP32	0,443	1,02
FP16	0,441	0,57

Maintien des performances pour une baisse de 44% du temps d'inférence

NANO vs PICO vs FEMTO

BT 0.1
50 epochs
FP32
640x640

YOLO	Loss : Train | Val	Inférence ms/img
Nano	1,03 | 1,28	1,02
Pico	2,59 | 4,57	0,54
Femto	3,66 | 4,61	0,36

Le temps d'inférence est fortement réduit mais les performances sont fortement impactées
→ Réaliser un HP tuning sur les modèles femto et pico

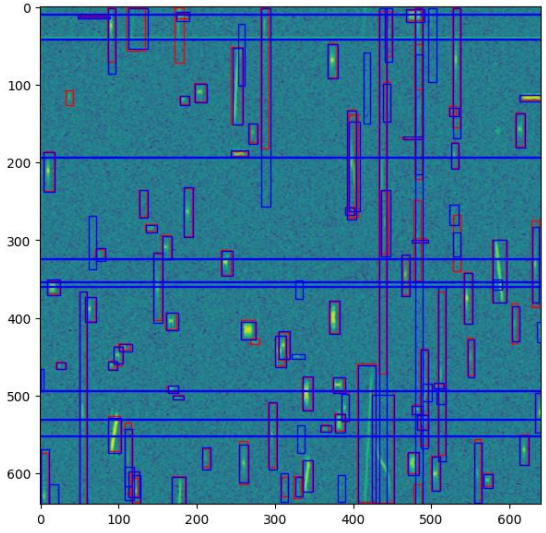
SiLU vs ReLU : Performances

BT 0.1
50 epochs

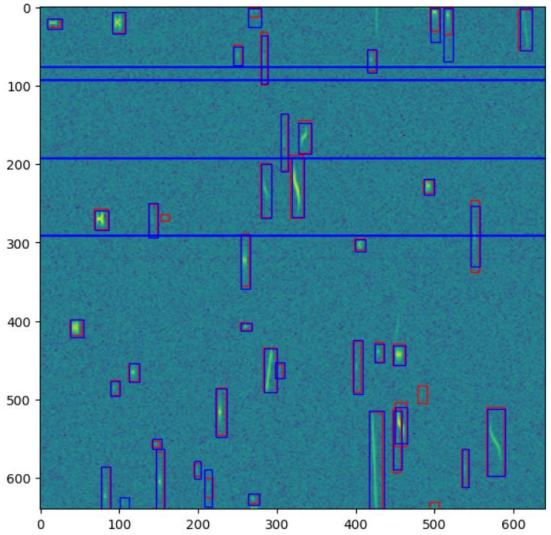
Activation	Loss : Train | Val
SiLU	1,0344 | 1,276
ReLU	1.1125 | 1.279

Aucune influence de la fonction d'activation

SiLU sur Test



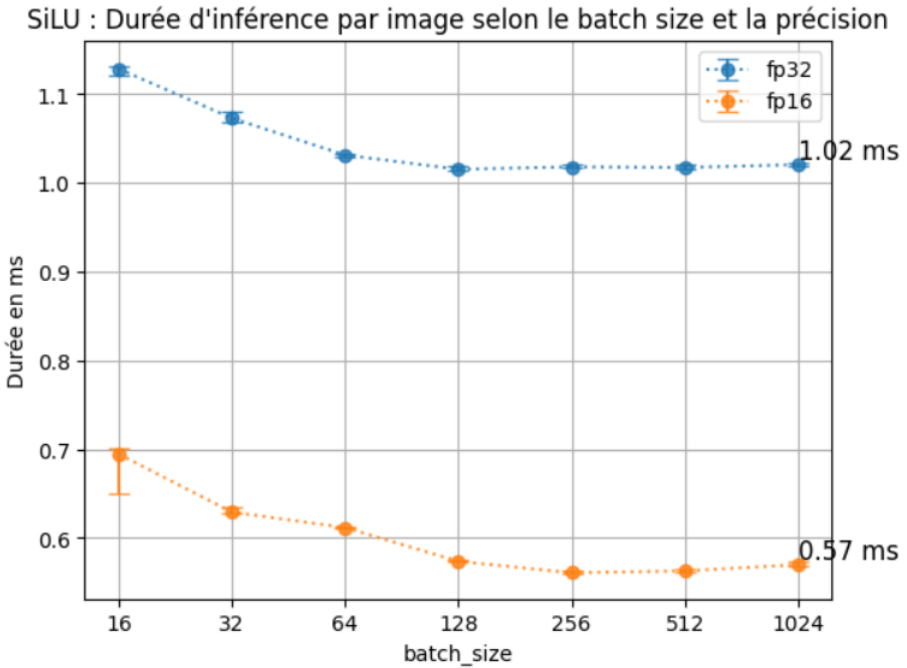
ReLU sur Test

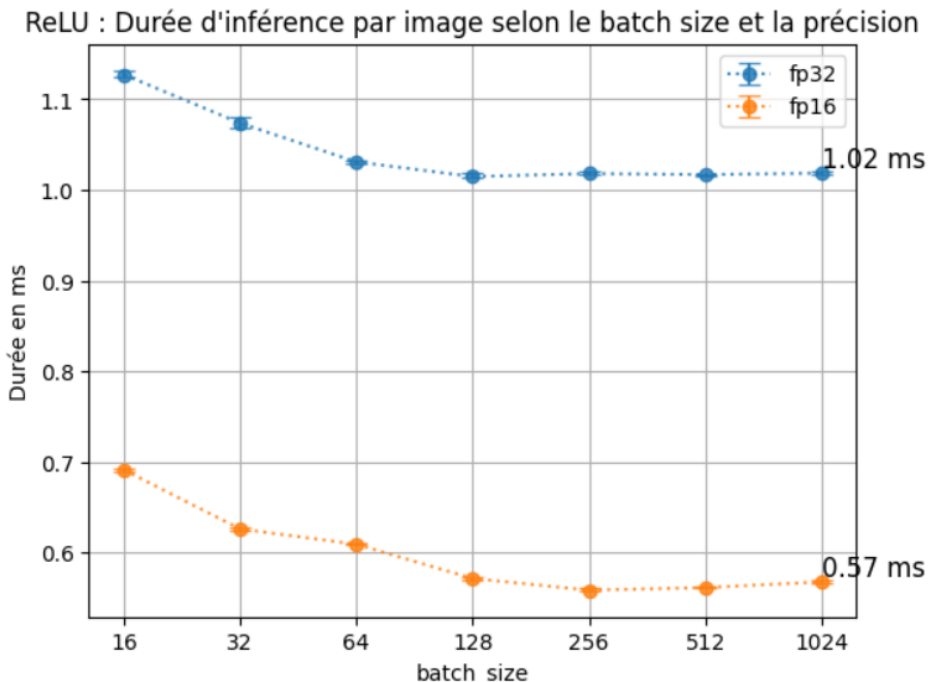


SiLU vs ReLU : Temps d'inférence

BT 0.1
50 epochs
640x640
FP32 & FP16

Aucune influence de la fonction d'activation





NC=3 vs NC=1

Le gain en calcul ne se fait que sur la première couche

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 6)         456

 max_pooling2d (MaxPooling2  (None, 14, 14, 6)         0
 D)

 conv2d_1 (Conv2D)           (None, 10, 10, 16)        2416

 max_pooling2d_1 (MaxPoolin  (None, 5, 5, 16)          0
 g2D)

 flatten (Flatten)           (None, 400)               0

 dense (Dense)               (None, 120)               48120

 dense_1 (Dense)             (None, 84)                10164

 dense_2 (Dense)             (None, 10)                850

=================================================================
Total params: 62006 (242.21 KB)
Trainable params: 62006 (242.21 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

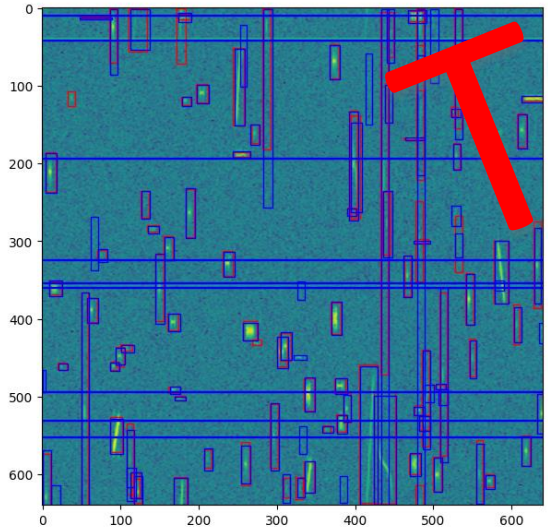
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 28, 28, 6)         156

 max_pooling2d_2 (MaxPoolin  (None, 14, 14, 6)         0
 g2D)

 conv2d_3 (Conv2D)           (None, 10, 10, 16)        2416

 max_pooling2d_3 (MaxPoolin  (None, 5, 5, 16)          0
 g2D)

 flatten_1 (Flatten)         (None, 400)               0

 dense_3 (Dense)             (None, 120)               48120

 dense_4 (Dense)             (None, 84)                10164

 dense_5 (Dense)             (None, 10)                850

=================================================================
Total params: 61706 (241.04 KB)
Trainable params: 61706 (241.04 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

NC=3 vs NC=1

BT 0.1
50 epochs

Number of channels	Loss (Train | Val) ↓	Inference duration (ms/img) ↓
3	1,0344|1,276	
1	|	

nc=3 sur Test



nc=1 sur Test

TODO

640x640 vs 512x512

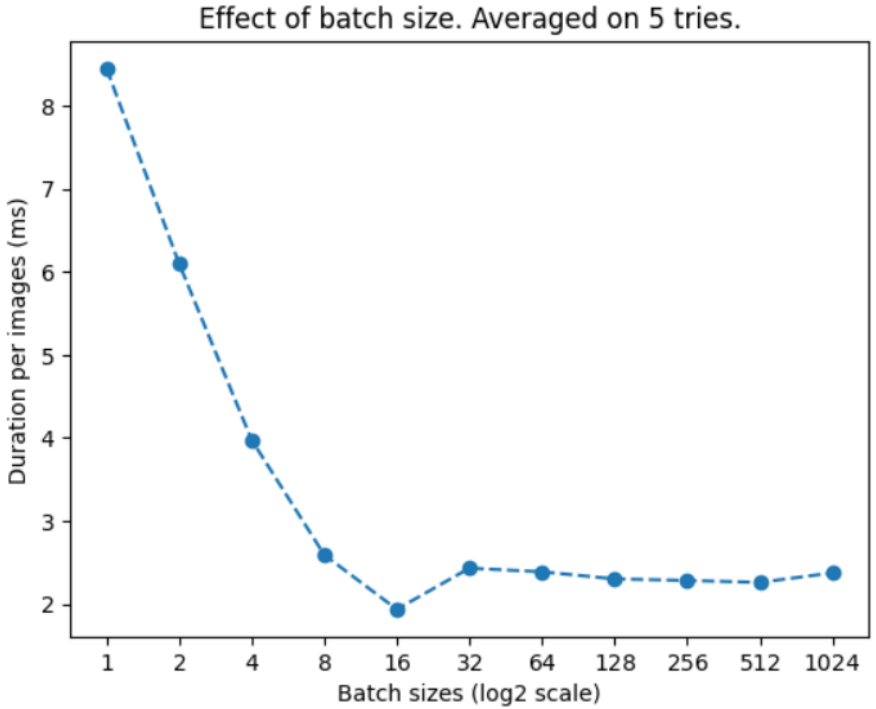
BT 0.1
50 epochs

Utilisation VRAM : PyTorch

Objet	Théorique (MiB)	Effectif (MiB)	DELTA (MiB)
V100	32 501	32 496	-5
Warm PyTorch	NA	311	NA
YOLOv8 nano	11,5	30	-18,6
100 tensors 640x640x3xFP32	470	470	0
Delete 100 tensors	0	470	-470
Création 100 tensors	470	0	+470
Inférence 100 tensors	CA DEPEND. Voir slides d'après		

La mémoire était effectivement libérée avant

Utilisation VRAM : Effet du batch size



Coût fixe existant

→ Profiter de la parallélisation

→ Essayer d'utiliser le plus de mémoire possible moins un delta de sécurité

Out of Video Memory

Utilisation VRAM : PyTorch en inférence

Images en entrée + BBOX

Nb BBOX fs=4GS/s, NPERSEG = 128, 256, …, NFFT=512, NOVERLAP = NPERSEG/2 et images 512x512x1

NPERSEG	Nombres de bboxes moyen	
128		
256		
512		
1024		

TODO Voir reuil2 excel

AVANTIX

MERCI