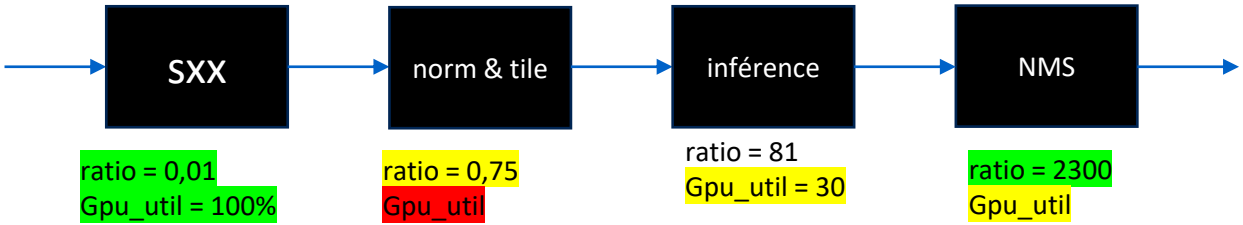
AVANTIX

Avancement Simulateur IQ

Etapes du pipeline

- SXX = abs(fft(iq.reshape()*window)
- norm_and_tile = (sxx/d1).reshape()
- Inference = model(sxx_tiled)
- NMS : nms(preds)
- Merging npersegs :
  - Intra
  - Inter
- Feature extraction:
  - DDC …

Ratio en cours



Légende

Confirmé

A confirmé

Non évalué

AVANTIX

Temps contraint : inférence

10/01/2024

Divers

Warmup :
Pas de gain à avoir exactement même taille de tensor d'une inférence à une autre contrairement au warmup des Sxx

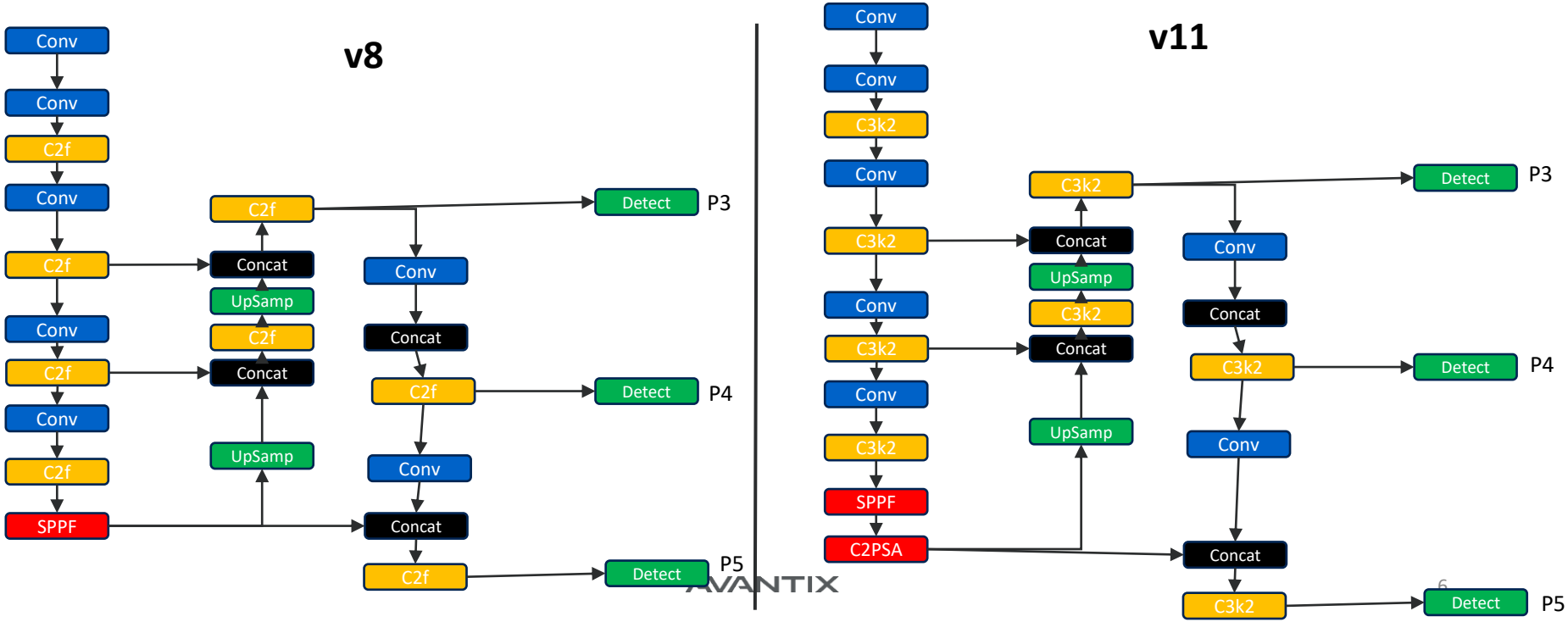
NMS :
Post-process : NMS (512µs/img) + ~~formatage des résultats (204 µs)~~
Suppression du formatage des résultats. Seuls les paramètres des xywhr nous intéressent

Tensor-RT :
Problèmes à l'installation et à l'utilisation

Architecture YOLO

Objectif :

Profiler le modèle afin de proposer une version plus légère en essayant de moins dévier possible de l'architecture d'origine car pas le temps d'entrainer et vérifier chaque changement.

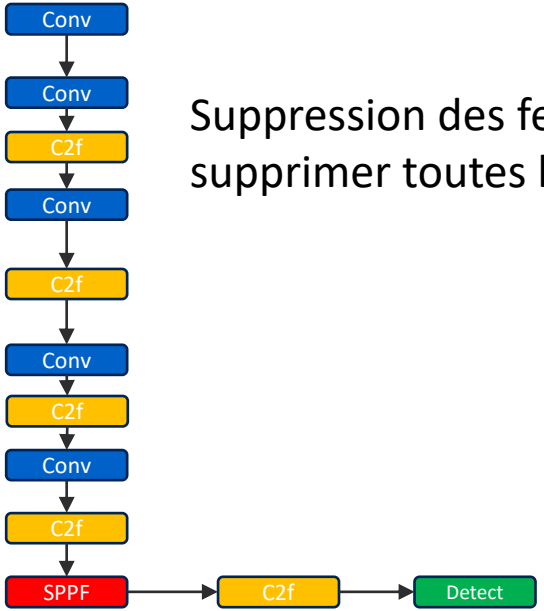


Profiling v8nano

Durée pour chaque étape:

out_backbone_p3: 381.75 ms (44.91% in total inference processing)
out_backbone_p4: 60.94 ms (7.17% in total inference processing)
out_backbone_p5: 51.29 ms (6.03% in total inference processing)
out_inter_neck: 45.04 ms (5.30% in total inference processing)
out_p3: 93.23 ms (10.97% in total inference processing)
out_p4: 43.95 ms (5.17% in total inference processing)
out_p5: 26.14 ms (3.07% in total inference processing)
out_angle_concat: 39.65 ms (4.66% in total inference processing)
out_angle_sigmoid: 0.24 ms (0.03% in total inference processing)
detect_bbox_cls0: 71.33 ms (8.39% in total inference processing)
detect_bbox_cls1: 23.38 ms (2.75% in total inference processing)
detect_bbox_cls2: 8.62 ms (1.01% in total inference processing)
out_cat: 0.59 ms (0.07% in total inference processing)
out_box_cls: 0.04 ms (0.00% in total inference processing)
dbox: 2.52 ms (0.30% in total inference processing)
dbox_cls_concat: 0.67 ms (0.08% in total inference processing)
final_output: 0.69 ms (0.08% in total inference processing)
Total = 850.07ms (Ratio RT=201)

Architecture P5 only



Suppression des features maps P3 et P4 → Cela permet de supprimer toutes les étapes d'upsampling et concaténation

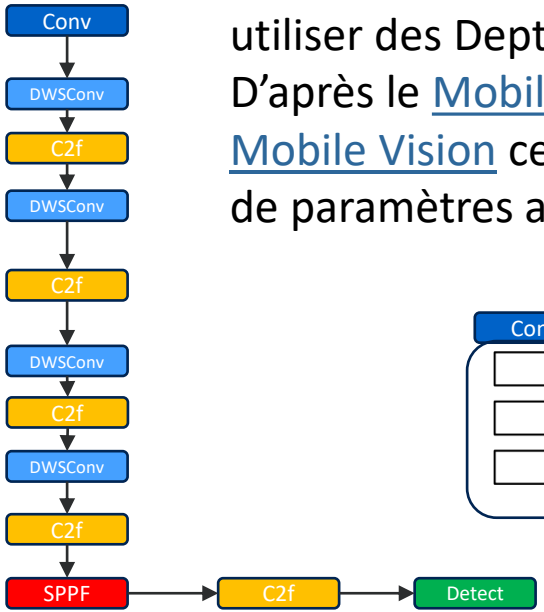
Nano v8 P5 only

Suppression des branches P3 & P4

Durée pour chaque étape:

```
out_backbone_p3: 381.47 ms (72.32% in total inference processing)
out_backbone_p4: 60.94 ms (11.55% in total inference processing)
out_backbone_p5: 51.39 ms (9.74% in total inference processing)
out_p5: 20.00 ms (3.79% in total inference processing)
out_angle_concat: 3.82 ms (0.73% in total inference processing)
out_angle_sigmoid: 0.13 ms (0.03% in total inference processing)
detect_bbox_cls2: 9.19 ms (1.74% in total inference processing)
out_cat: 0.02 ms (0.00% in total inference processing)
out_box_cls: 0.08 ms (0.01% in total inference processing)
dbox: 0.31 ms (0.06% in total inference processing)
dbox_cls_concat: 0.08 ms (0.02% in total inference processing)
final_output: 0.05 ms (0.01% in total inference processing)
Total = 527.50ms (Ratio RT=125)
```

Architecture P5 only + Depthwise Separable Conv



Afin de réduire la complexité des couches de convolutions, plutôt utiliser des Depthwise Separable Conv (DWSConv).
D'après le [MobileNets: Efficient Convolutional Neural Networks for Mobile Vision] cette optimisation permet de diviser par 7 le nombre de paramètres avec une perte faible en performance.

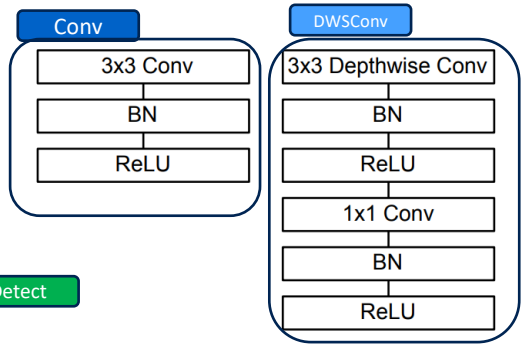


Table 4. Depthwise Separable vs Full Convolution MobileNet

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
Conv MobileNet	71.7%	4866	29.3
MobileNet	70.6%	569	4.2

https://arxiv.org/pdf/1704.04861

Conv vs DWSConv

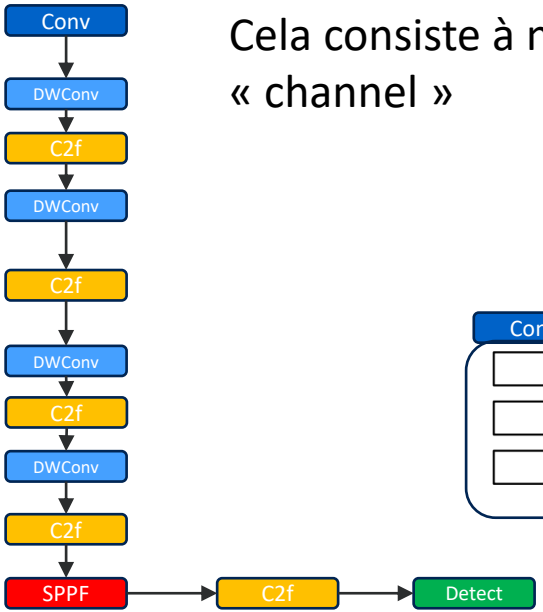
```
Duration for the classical conv layer: 2.80ms
Duration for the DWS conv layer: 1.90ms

Number of weights in classical conv: 4608
Number of weights in DWS conv: 656
```

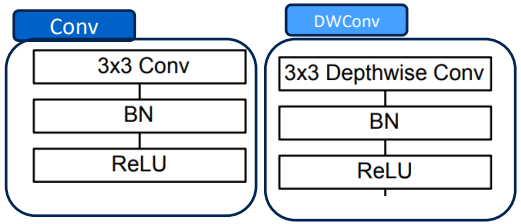
Avec 7x moins de paramètres, on obtient un gain de 1,5 seulement.

Après intégration dans le modèle YOLO, je n’ai quasi aucun gain voir une perte en ratio RT.

Architecture P5 only + Depthwise Conv

Version encore moins calculatoire : Depthwise Conv
Cela consiste à ne pas considérer les calculs selon la dimension « channel »





https://arxiv.org/pdf/1704.04861

Nano v8 P5 only + DWConv

```
Duration for the classical conv layer: 2.79ms
Duration for the DW conv layer: 1.11ms

Number of weights in classical conv: 4608
Number of weights in DW conv: 144
```

Avec 32x moins de paramètres, on obtient un gain de 2,5 seulement.

TODO

- Moins de channel par Conv
- SiLU vs ReLU
- Pruning : Mettre à zéro un certain pourcentage des poids du modèle
- EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks 8,6x plus petit et 6x plus rapide d'après les auteurs
- ParameterNet : https://arxiv.org/pdf/2306.14525

AVANTIX

MERCI