Détection de navires embarquable à bord de satellites

Thomas Goudemant, Benjamin Francesconi, Houssem Farhat, Lionel Daniel, Olivier Thiery, Erwann Kervennic, Adrien Girard, Seif Mzoughi

Détection de navires embarquable à bord de satellites

T. Goudemant[1], B. Francesconi[1], H. Farhat[1], L. Daniel[1], O. Thiery[2], E. Kervennic[1], S. Mzoughi[1], A. Girard[1]

[1]*Institut de Recherche Technologique Saint-Exupéry - France*
[2]*Geo4i - France*
Contact: thomas.goudemant@irt-saintexupery.com

***Abstract*—La surveillance maritime répond à d'importants enjeux à la fois écologiques, économiques et sécuritaires. Dans ces domaines, les systèmes d'alerte doivent être réactifs et autonomes. C'est dans ce cadre que cet article explore la faisabilité d'une solution logicielle de détection de navires embarquable sur une électronique de satellite d'observation haute-résolution en orbite-basse. L'article décrit plusieurs implémentations sur FPGA et GPU de cette solution basée sur le détecteur YOLOv3 entrainé sur une base de données inédite d'images annotées par un photo-interprète.**

***Index Terms*—Détection optique de navires, Imagerie satellite, IA embarquée, Surveillance autonome, Pêche illégale**

I. Introduction

A. Télédétection spatiale

L'observation de la Terre depuis l'espace permet de nombreuses applications aussi diverses que de la mesure du champ de gravité [6], la détection précoce de tempête [13], l'évaluation de la déforestation [16] ou de la pauvreté [8]. Notre article se focalise sur la surveillance maritime qui répond à des enjeux à la fois sécuritaires, écologiques, économiques :

- la stratégie militaire
- le contrôle du trafic maritime, dont la minimisation des risques, évitement de bateaux, détection de manoeuvres anormales [19]
- le contrôle des activités en mer, légales et illégales telles que le dégazage sauvage d'hydrocarbures [5], le transport de stupéfiants ou armes, les actes de pirateries, et la pêche illégale.

Concernant la pêche illégale, non déclarée et non réglementée, nommée IUU (*"Illegal, Unreported and Unregulated fishing"*), elle représenterait environ 19% des volumes de pêche et 10 milliards d'euros chaque année dans le monde [3]; cette pratique sauvage impacte la biodiversité et les 3 milliards de personnes qui dépendent de la pêche [4]. Normalement, les navires sont équipés du système radio d'identification AIS (*"Automatic Identification System"*), mais ce système peut être manuellement désactivé voire modifié afin de diffuser de fausses déclarations d'identité ou d'activité ; ces déclarations sont difficilement vérifiables car les moyens terrestres et aériens ne peuvent couvrir l'ensemble des surfaces maritimes. La surveillance par satellites permet ainsi de palier cette déficience et donne en plus de riches informations sur l'environnement. Télédétecter ces activités illégales de pêche ou transbordement n'est possible qu'avec des satellites de haute résolution spatiale, comme indiqué dans la table I. La résolution spatiale (GSD en anglais) désigne la distance (en mètres) au sol entre deux pixels d'une image satellite. Cette image peut avoir été acquise par un radar, SAR [21] en anglais, ou une caméra optique ; dans cet article, le mot image désigne les données acquises par une caméra optique. En plus de sa résolution spatiale, il est important d'en connaître sa résolution spectrale pour savoir combien et quelles couleurs seront captées par la caméra. Dans notre étude, les images proviennent de caméras percevant le Rouge, Vert et Bleu, RGB en anglais, et une quatrième bande spectrale proche-infrarouge que nous n'avons pas utilisée.

TABLE I
TÂCHES DE TYPE DRI (DÉTECTION / RECONNAISSANCE / IDENTIFICATION) RÉALISABLES PAR PHOTO-INTERPRÉTATION SELON LA GAMME DE RÉSOLUTION SPATIALE, POUR DES IMAGES OPTIQUES RGB

Résolution	Fonction	Remarques
2-15m	Détection (sauf petits bateaux)	Les navires de commerce et les grands navires de combat sont détectés sans pouvoir spécialement donner leur fonction
0.7-2 m	Reconnaissance	Les navires de commerce et les grands navires de combat sont détectés et il est possible de donner leur fonction. Les petits navires sont détectés sans pouvoir donner leur fonction.
0.7 m	Reconnaissance (Identification)	Les petits navires sont détectés et il est possible de donner leur fonction. L'identification (nom/immatriculation) n'est en général pas possible sauf cas particulier (navires remarquables connus)

B. Objectifs généraux des travaux et périmètre des résultats présentés

Les travaux présentés dans la suite de cet article sont menés dans le cadre du projet CIAR (Chaine Image Autonome et Réactive) de la branche Smart Technologies de l'IRT Saint-Exupery, dont l'objectif principal est le développement d'algorithmes de traitements d'images par IA embarquée sur cibles matérielles. Cet objectif est décliné en trois axes techniques de travail mis en oeuvre sur différents cas d'applications :

- Axe base de données d'images + annotations : s'approprier des bases de données existantes ou en

constituer de nouvelles répondant aux besoins des cas d'applications choisis (degré de confiance/qualité, représentativité, nombre d'images, etc) et aux besoins futurs des partenaires du projet.
- Axe algorithmique : concevoir et tester des algorithme d'IA pouvant être exécutés sur cibles matérielles à capacités limitée (calcul, puissance, mémoire...), adaptés aux images brutes en sortie capteur et robustes à la variabilité opérationnelle
- Axe portage sur cible : Maitriser les outils et le savoir-faire (ex : techniques de quantification des réseaux) pour le déploiement d'IA sur cible, optimiser les performances matérielles (débit, latence, consommation, empreinte sur cible...)

La suite de l'article présente les résultats préliminaires obtenus sur le cas d'application spécifique qu'est "la détection et la reconnaissance automatiques de navires dans des images satellites optiques RGB à l'aide de réseaux de neurones embarqués sur cibles matérielles".

C. Plan de l'article

L'article est structuré autour des trois grandes étapes du développement d'une Intelligence Artificielle embarquée :

- Section II : Description de la base de données d'images utilisée pour l'entraînement et l'évaluation des performances algorithmiques.
- Section III : Description de l'algorithme choisi, YOLOv3, de son entraînement et de ses performances obtenues avant déploiement.
- Section IV : Description des cibles électroniques utilisées, du déploiement sur cibles et performances embarquées.

II. Base de données

Les algorithmes d'apprentissage, dont font partie les réseaux de neurones, nécessitent de grandes quantités d'exemples pour apprendre à réaliser une tâche donnée (ex : détection d'objet) dans un contexte donné (ex : télédétection de navires). Un exemple est généralement constitué d'une donnée à traiter (ex : image de mer contenant des navires) associée à une annotation contenant l'information que doit retourner l'algorithme (ex : positions des navires dans l'image et leurs types). La performance et la robustesse des algorithmes reposent donc en grande partie sur la qualité et la représentativité de la base de données d'images annotées utilisée pour l'apprentissage. La construction des bases de données est donc une étape critique dans le développement de solutions basées sur ce type d'algorithmes, nécessitant, selon l'application, une forte expertise dans le domaine visé. Dans le cas de la détection ou reconnaissance de navires par imagerie spatiale, l'annotation précise des images nécessite l'expertise d'un photo-interprète.

Quelques bases de données d'images haute-résolution utilisables pour de la détection ou reconnaissance de navires dans des images satellites optiques sont accessibles publiquement [14]. Malheureusement, aucune de ces bases de données ne répondait à l'ensemble des exigences suivantes :

- Localiser précisément les navires, à minima boite englobante orientées autour des navires
- Fournir les informations concernant la résolution des images, le type de capteur et les traitements appliquées
- Besoin de pouvoir contrôler la taille des images soumises aux algorithmes pour optimiser leurs performances sur cible embarquée.

Par conséquent, la construction d'une base de données de grande qualité a été confiée à la société GEO4I, partenaire du projet CIAR et spécialiste en fourniture et analyse d'images satellites. Ainsi, nous avons généré de manière incrémentale une base de données d'images sources (RGB, et proche infrarouge) de grandes dimensions (> 10 à 100s Mpixels) et à très haute résolution (30 à 50cm), annotées à un niveau de détails inédits par les analystes GEO4I grâce a des outils propriétaires adaptés à ce type d'images. Le tableau II synthétise les caractéristiques principales de cette base de données. Les images commandées sont issues du catalogue Maxar Imagery Products, corrigées au sol et donc non représentatives des images en sortie capteur à bord du satellite. Néanmoins, pour pouvoir simuler dans le futur les spécificités des images bord, les niveaux de corrections demandés ont été réduits au minimum.

Actuellement un tiers des navires est annoté avec le niveau de classification le plus détaillé (50 classes). Les autres navires sont annotés avec 10 classes. Une mise à jour de la base de données en 2022 mettra à niveau les annotations de ces navires. L'intérêt d'augmenter le niveau de spécificité des classes est double : d'une part cela permet de mieux ségréguer des navires aux caractéristiques communes et donc de faciliter la classification (par exemple il a priori est plus facile et naturel de reconnaître un cargo qu'un 'navire de commerce', classe abstraite regroupant des navires très différents) ; d'autre part l'intérêt opérationnel est accru car en identifiant la fonction ou le type précis du navire on est capable de dire si sa position ou son activité en cours semble légitime (surveillance) ou encore, dans le cas d'un navire militaire, cela permet d'avoir une idée de sa dangerosité. La figure 1 illustre quelques-une des classes de navires identifiées dans les images sources.

TABLE II
Caractéristiques la base de donnée source

Superficie totale	nb. de zones	Résolution des images	nb. total de navires annotés	nb. de classes
$\sim 2200 km^2$	47	0.3 - 0.5 m	$\sim$15900	$\sim$50

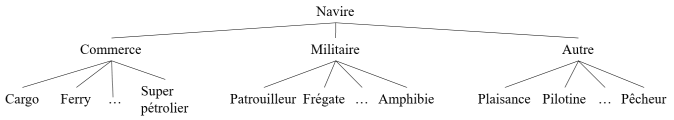

Fig. 1. Exemples de classes utilisées pour l'annotation des navires. La base de données contient environ 50 classes distinctes de navires

A partir de cette base de données source, GEO4I a généré des bases de données d'entraînement/validation/test nécessaires pour les algorithmes (voir Figure 2). L'outil utilisé par GEO4I est un SaaS (Software as a Service) collaboratif dédié à l'annotation d'images satellites ou d'images géoréférencées. Il a été développé initialement pour les besoins de la société afin de pouvoir créer et générer des bases de données d'apprentissage sur mesure. Il permet le détourage manuel d'objets, l'affectation de classes multiples (en effet un super pétrolier est également un navire et aussi un navire de commerce) et la génération automatique de bases de données d'apprentissage de façon paramétrable en termes de classes, de dimensions des images d'apprentissage ainsi que du type de détourage des navires (boites pour de la détection ou masques pour des besoins de segmentation).

Les travaux présentés dans cet article s'appuient sur une version préliminaire de la base de données d'images sources contenant environ 8000 navires distincts à des résolutions allant de 1.2 à 2m. Dans les images, les navires ont des tailles allant de quelques pixels à environ 350 pixels. Les scènes annotées ont été découpées en vignettes de taille 416x416 pixels, compatible des ressources mémoire des cibles hardware. Un recouvrement de 50% entre vignettes voisines et d'autres techniques de data-augmentation appliquées par le modèle lors de l'entraînement (redimensionnement, rognage et rotation aléatoire des images ainsi que distorsion des couleurs) sont appliqués pour augmenter artificiellement le nombre de navires et la variabilité des images. Les 2349 images résultantes ont été partitionnées aléatoirement en trois sous-ensembles :

- un jeu pour l'entraînement du modèle
- un jeu de validation: ces données ne sont pas utilisées pour entraîner le modèle mais permettent de valider que le réseau arrive à généraliser sur des données nouvelles. Ce jeu permet d'arrêter l'entraînement avant que le modèle "sur-apprenne" sur le jeu d'entraînement
- un jeu de test qui est utilisé pour évaluer les performances du modèle après l'entraînement.

Les tailles relatives des trois datasets ainsi que le nombre de bateaux annotés sont indiqués dans le tableau III.

TABLE III
RÉPARTITION DU DATASET EN TROIS SOUS-ENSEMBLES

	pourcentage du dataset	nombre d'imagettes	nombre d'annotations
train set	60%	1410	20825
val. set	20%	469	6847
test set	20%	470	6717

Afin de se concentrer sur le portage sur cible et simplifier le problème de reconnaissance, plus difficile à cette résolution, ces travaux préliminaires quantifient uniquement les performances de détection. Afin d'être représentatif de la majorité des capteurs haute résolution existants, seules les bandes RGB ont été utilisées pour ces travaux. Cela facilite également la réutilisation de la majorité des réseaux de neurones, conçus la plupart du temps pour des images RGB grand public. L'objectif de futurs travaux sera d'exploiter tout le potentiel offert par cette base de données en visant des tâches de reconnaissance détaillée des navires.

III. MODÈLE DE DÉTECTION DE BATEAUX

A. Choix de l'algorithme

Comme dans la plupart des applications embarquées, les algorithmes et les cibles matérielles doivent respecter les contraintes liées à leur environnement, en particulier dans un contexte spatial.

Premièrement, ces applications sont caractérisées par le besoin d'une faible consommation énergétique ainsi que des capacités mémoires et de calculs réduites par rapport aux applications opérées au sol. De plus, dans le cadre de la détection de bateaux, des images hautes résolutions (≤ 50 cm) seront acquises à très haut débit (≥ 500 MPixels/s), imposant une contrainte embarquée forte sur le temps de traitement par pixel. Finalement, des contraintes liées au déploiement de l'algorithme sur cibles (GPU, FPGA et TPU) sont également à prendre en compte. L'architecture et les couches du réseau devront être compatibles avec les outils de déploiement, limitant le portage des réseaux les plus récents. De plus, la taille du réseau doit être facilement adaptable aux capacités de la cible.

L'entraînement du réseau s'effectuant sur un serveur, caractérisé par de grosses capacités de calculs, ce sont principalement les ressources à l'inférence qu'il faut optimiser. Afin de maximiser le ratio performance de détection/temps d'inférence, et en considérant les contraintes de déploiement lors du portage, le réseau YOLOv3 [18] a été sélectionné. Ce modèle étant constitué uniquement de convolutions, et non de couches avec une dimension d'entrée prédéfinie (pooling / couches denses / etc.), il accepte une taille d'image variable en entrée. Il présente ainsi l'avantage d'être adaptable aux capacités de la cible d'exécution. YOLOv3 a inspiré de récente architecture telles que EnhancedYOLOv3Tiny [15], YOLO-Ship [22], ShipYOLO [11] ou YOLOX [10], et cette dernière architecture semble maintenant supportée par l'outil de déploiement VITIS AI 2.0 [20].

B. Description du modèle YOLOv3

Cette architecture, représentée à la figure 3, est principalement composée:

- Du réseau de neurones à convolution (CNN) *Darknet-53* constitué de 53 couches de convolutions. Le composant principal de ces couches est le block résiduel (en mauve), introduit pour la première fois par [12]. Cette première partie du réseau agit comme un encodeur, extrayant les caractéristiques de l'image (*feature maps*) à différents niveaux de résolution.
- De couches de sur-échantillonage (en vert) qui agissent comme un décodeur en augmentant la résolution des *feature maps*. En sortie du sur-échantillonage, les informations générées sont mélangées (addition) avec certaines couches intermédiaires du CNN.

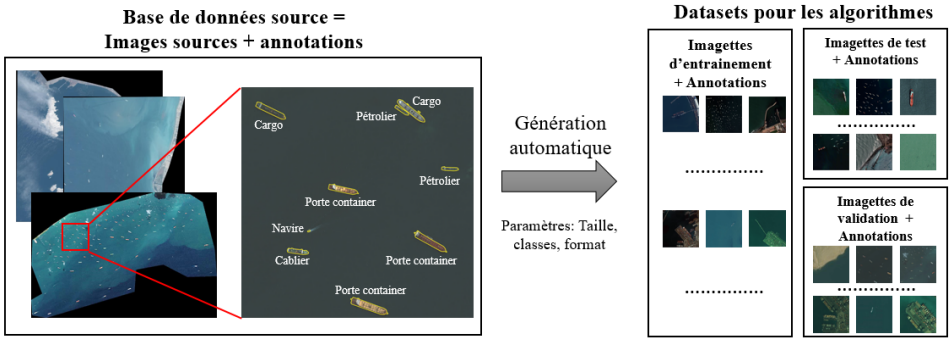

Fig. 2. Génération des datasets indépendants d'entraînement / validation / test pour l'apprentissage des réseaux de neurones et l'évaluation des performances. Imagery Products © 2021 Maxar Technologies.

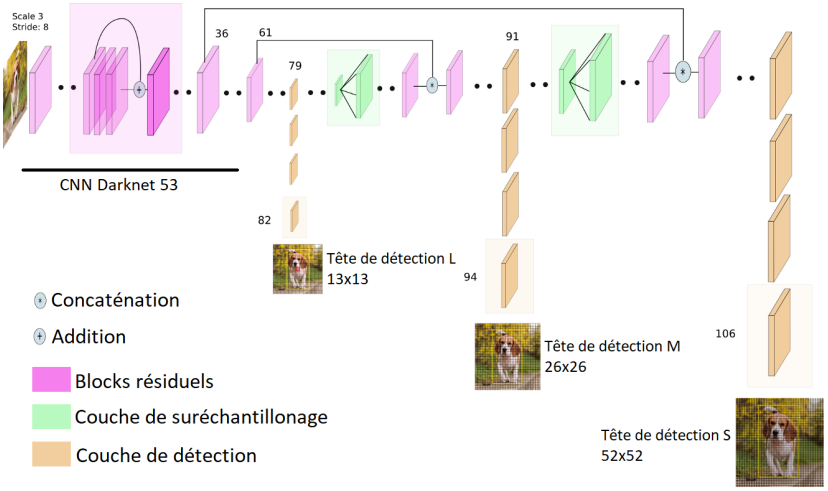

Fig. 3. Représentation de l'architecture de YOLOV3 comme présenté dans [7]. La dimension des images d'entrée est fixée à (416x416x3) ce qui influe sur le nombre de cellules des trois têtes de détection (13x13, 26x26 et 52x52 cellules)

- De trois têtes de détection (en jaune) se terminant par une grille de cellules. Chacune des ces cellules peut détecter et classifier jusqu'à trois objets. D'une grille à l'autre, le nombre de cellules et la dimension des objets détectés varient. Plus une grille de détection est située en fin du réseau, plus elle est générée à partir de couches haute-résolution, ce qui lui permet de détecter des plus petits objets.

Pour une image (3 canaux RGB) en entrée de dimension (W,H,3), la dimension des grilles en sortie de modèle est respectivement de, $(\frac{W}{32} \times \frac{H}{32})$, $(\frac{W}{16} \times \frac{H}{16})$ et $(\frac{W}{8} \times \frac{H}{8})$ cellules. Chacune de ces cellules produit 3 boites de prédictions (*bounding box*) représentées chacune par un vecteur BB de dimension $(5 + C)$ et défini par le vecteur (1) :

$$\text{BB} = (x, y, w, h, p_{obj}, p_0, .., p_{C-1}) \tag{1}$$

avec:

C = nombre de classes prédites par le réseau
xy = coordonnées du centre de la boite
wh = largeur et hauteur de la boite
p_{obj} = probabilité de présence d'un objet
p_i = probabilité d'appartenir à la classe i

Il est à noter que le réseau ne prédit pas directement les boites au bon format et qu'une mise en forme est nécessaire. Ce traitement, défini dans [18], consiste principalement en:

- la translation des coordonnées du centre de la boite en fonction de la position de la cellule sur la grille
- la modifications de l'*a priori* sur la dimension des boites (ancres, cf. section III-C2).

Une fois cette mise en forme appliquée, les objets de la classe i présents dans l'image sont détectés en filtrant les BB dont le score est supérieur à un seuil, l'*Object threshold*:

$$score = p_{obj} \times p_i \tag{2}$$

Cependant, un même objet est régulièrement détecté par plusieurs cellules voisines (cf. figure 4). Afin de ne conserver qu'une seule boite, un algorithme appliqué en post-traitement, le NMS (*Non Maxima Suppression*), va détecter les boites correspondant au même objet et les supprimer. Un seuil, le *NMS threshold*, permet de définir à partir de quel recouvrement l'algorithme considère que deux boîtes correspondent au même navire. Ce seuil est fixé à 50% pour nos simulations.

Fig. 4. Illustration de la détection d'un même objet par plusieurs cellules voisines. Après application de l'algorithme NMS, seule la boite en blanc est conservée. Imagery Products © 2018 Maxar Technologies.

C. Entraînement du réseau de détection

1) Fonction de coût: Comme défini par [17], afin de détecter les objets annotés $(\hat{x}, \hat{y}, \hat{w}, \hat{h}, \hat{p_{obj}}, \hat{p_0}, .., \hat{p_{C-1}})$, le réseau est entraîné à minimiser une fonction de coût (*loss function*).

Cette coût se divise en plusieurs composantes pondérées par différents coefficients λ:

- coût de localisation (λ_{coord}): elle est liée à l'erreur sur les coordonnées (x_i, y_i, w_i, h_i) des boites prédites
- coût de non-détection (λ_{obj}): elle est liée à l'erreur de non détection d'un objet
- coût de fausse détection (λ_{noobj}): elle est liée à l'erreur de prédire un objet inexistant
- coût de classification (λ_{class}): elle est liée à l'erreur sur la classification d'un objet

2) Choix des ancres: Les ancres représentent un a priori sur la dimension des bateaux à détecter. Au nombre de 9, elles sont associées chacune à une des 3 grilles et à une des 3 prédictions des cellules. Afin de correspondre au mieux à la distribution des bateaux du jeu de données, les dimensions (w,h) des navires sont divisées en 9 classes à l'aide de l'algorithme de clustering K-means. Les ancres sont ensuite déterminées en sélectionnant les dimensions moyennes de ces 9 classes.

Durant l'inférence, chacune des prédictions des cellules génèrent un couple (δ_w, δ_h) qui correspond à la modification à apporter à l'ancre (p_w, p_h) pour obtenir la largeur et hauteur de la prédiction selon l'équation (3):

$$w = p_w \times e^{\delta_w} \quad \text{et} \quad h = p_h \times e^{\delta_h} \tag{3}$$

3) Procédure d'entraînement: Pour entraîner ce modèle à notre cas d'utilisation, nous avons choisi une implémentation de YOLOv3 en Keras disponible sur [9]. Nous avons modifié son architecture afin qu'il ne détecte que la classe bateau, le nombre de classes *C* est donc égal à un. Le réseau a été entraîné avec 3 GPU Nvidia GTX 1080 Ti. Les hyperparamètres utilisés pour l'entraînement sont listés ci-dessous. Mis à part les ancres et le redimensionnement des images qui ont été optimisés, le choix des ces valeurs correspond à l'état de l'art.

- Max *epochs*: 300
- Taille du *batch*: 24 soit 8 par GPU
- Redimensionnement aléatoire des images entre 384x384 et 448x448
- Taux d'apprentissage initial: 1e-4
- Initialisation des poids: réseau Darknet-53 pré-entrainé sur ImageNet
- Paramètres de la fonction d'erreur:

$$\lambda_{coord} = \lambda_{noobj} = \lambda_{class} = 1, \ \lambda_{obj} = 5 \tag{4}$$

- Ancres: [5,7,7,13,12,17,14,9,16,29,28,16,41,38,87,57,111,136]

D. Performances

Avant de présenter les performances du modèle, nous définissons différents concepts utiles pour la suite :

- IoU: de l'anglais *Intersection over Union*, ratio entre l'intersection et l'union des surfaces des boites prédite et attendue. L'IoU vaut 1 lorsque la prédiction est parfaite et 0 lorsque les boites n'intersectent pas.
- VP, vrai positif: prédiction correcte. Comme présenté à la section III-B, une prédiction est correcte si son score (2) est supérieur à l'*Object threshold* (fixé à 30%) et que l'IoU avec la vérité terrain est supérieure à un seuil (fixé à 50%).
- FP, faux positif: détection ne correspondant pas à un bateau, ou relative à un bateau déjà détecté.
- FN, faux négatif: un navire non détecté (score ou IoU avec la vérité terrain insuffisant).

À la deuxième colonne de la figure 6, plusieurs résultats d'inférence sur le jeu de test sont représentés. Le modèle est capable de détecter la majorité des vérités terrain, et ce, pour différents états de mer et morphologies de navires. Les performances du réseau pour les 3 jeux de données sont reprises dans le tableau IV et sont définies ainsi :

- Précision: proportion de prédictions correctes par rapport à l'ensemble des prédictions.
- Rappel: proportion de prédictions correctes par rapport à l'ensemble des navires présents dans les images.
- F1-Score: la moyenne harmonique de la précision et du rappel.
- AP50: une métrique classique qui correspond à l'aire sous la courbe précision-rappel pour un IoU$\geq 50\%$.

TABLE IV
PERFORMANCES DU MODÈLE EN FONCTION DU JEU DE DONNÉES

dataset	Précision	Rappel	F1-Score	AP50
Train	0,405	0,779	0,533	0,654
Valid	0,395	0,756	0,519	0,618
Test	0,416	0,784	0,543	0,676

Nous constatons que le réseau se comporte légèrement mieux sur le jeu de test que le jeu d'entraînement. Bien que le modèle soit censé se comporter mieux sur le jeu d'entraînement, l'évolution des performances lors de l'entraînement est un processus bruité et il est possible que

les performances sur le jeu de test dépassent celles du jeu d'entraînement.

Nous observons également que la précision du modèle est largement inférieure au rappel. Cela signifie que l'algorithme a tendance à générer plus de faux positifs que de faux négatifs. En analysant les images de la figure 6, la majorité des faux positifs proviennent de cellules voisines détectant plusieurs fois le même navire et non de zones de l'image sans navires. Il est cependant possible, en modifiant le *NMS threshold*, d'améliorer les précision du modèle au détriment du rappel et de l'AP50 (cf. tableau V). Ce compromis entre précision et rappel étant simplement un paramètre du post-traitement, il peut être modifié au cours de la mission et ne nécessite pas de ré-entraînement du modèle. Dans notre cas, nous avons choisi arbitrairement d'optimiser la métrique AP50, et avons donc conservé une valeur de 50% pour le *NMS threshold*.

TABLE V
MÉTRIQUES SUR LE JEU DE TEST EN FONCTION DU *NMS threshold*

NMS Thr.	Précision	Rappel	F1-Score	AP50
10%	0,600	0,674	0,635	0,603
30%	0,520	0,724	0,605	0,640
50%	0,416	0,784	0,543	0,676
70%	0,275	0,824	0,412	0,661

Finalement, la distribution, pour le jeu de test, des vrais positifs, des faux négatifs, des faux positifs ainsi que du nombre total de navires annotés en fonction de la dimension (en pixel) du navire est résumée par la figure 5. Un bon algorithme minimise, pour toutes les dimensions de navires, les FP (courbe rouge) et les FN (courbe jaune) et le nombre de VP (courbe verte) tend vers le nombre total de bateaux (courbe bleue). Dans cette figure, nous constatons que le modèle se comporte mieux pour les grands navires que les petits. En effet, la proportion de faux positifs et de faux négatifs est plus importante pour les petits navires ($\leq$ 15 pixels). Au-delà d'une dimension de 20 pixels, le nombre de bateaux non-identifiés chute en dessous de 10% et la proportion de faux positifs est inférieure aux vrais positifs. En effet, les performances du modèle s'améliorent avec la résolution et donc avec la taille des navires.

IV. DÉPLOIEMENT SUR ÉLECTRONIQUE EMBARQUÉE

A. Présentation des différentes cibles

Pour le déploiement de l'algorithme, nous avons sélectionné trois cibles différentes dans le but de se rapprocher des différentes contraintes opérationnelles (consommation et capacités de mémoire et de calcul réduites) et de couvrir différentes solutions hardware (FPGA et GPU). Les caractéristiques de ces cibles sont reprises dans le tableau VI.

1) NVIDIA Jetson Nano: destinée au portage d'applications d'IA embarquées à faible coûts et à consommation réduite, elle permet de valider un portage sur GPU avec peu de ressources.

2) NVIDIA Jetson NX: de la même famille que la Jetson Nano, elle offre des performances supérieures à faible consommation. Un critère dans le choix de cette cible est qu'il existe

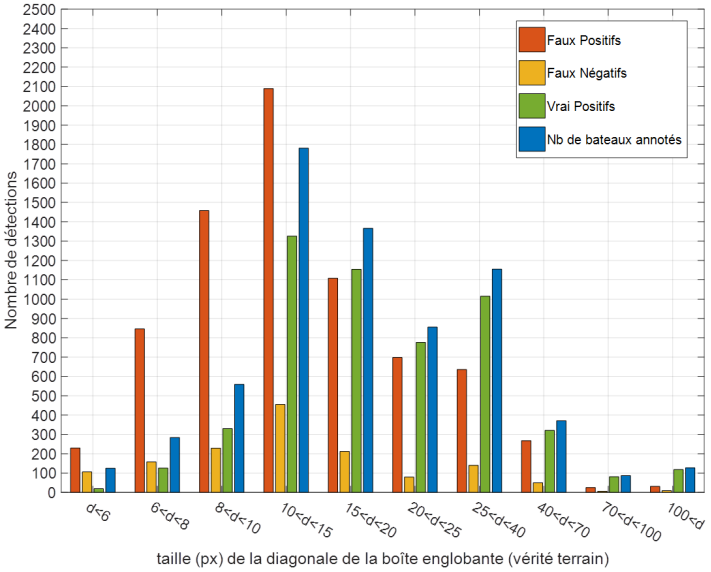

Fig. 5. Répartition du nombre de FP, FN, VP et de bateaux sur le jeu de test en fonction de la diagonale de la boite englobante

une version [1] compatible avec des applications embarquées (drone, avion, etc.) mais non spatiales. Une autre cible de la même famille, la Jetson TM TX2i, a également été certifiée pour l'environnement spatial [2].

3) Zynq UltraScale+ MPSoC ZCU104: cette carte d'évaluation est composée d'un MPSOC (*Multi-processor System on Chip*) ZU7EV. La partie logique programmable est pré-configurée avec une architecture matérielle optimisée pour l'exécution de réseaux de neurones, fournie par le constructeur Xilinx. Elle inclut des accélérateurs matériels (DPU : *Deep Learning Processor Unit*) permettant d'accélérer l'inférence des principales couches de réseaux de neurones à convolutions.

TABLE VI
CARACTÉRISTIQUES DES CIBLES

	NVIDIA Jetson Nano	NVIDIA Jetson NX	Zynq UltraScale+ MPSoc ZCU104
Perfo.	472 GFLOPs	21 TOPs	n.a.
Alim.	5V	entre [10V,20V]	12V
Conso.	$\leq 10W$	$\leq 20W$	$\leq 20W$
Microproc.	ARM Cortex-A57 4 coeurs à 1,43 GHZ	ARM NVIDIA Carmel 64bit 6 cœurs	Quadcore Arm Cortex-A53, Dualcore Arm Cortex-R5
GPU	NVIDIA Maxwell 128 coeurs CUDA	NVIDIA Volta 384 cœurs	Mali-400 MP2 GPU
Accél.	n.a.	NVIDIA Deep Learning Accelerator	Logique Prog. 504K cellules logiques, 1728 DSP
Mémoire RAM	4 GB LPDDR4	8GB LPDDR4	2GB DDR4
Quantif.	FP16	FP16/INT8	INT8 (VITIS AI Quantizer)
Autre	Codec vidéo H.264/H.265	Codec vidéo H.264/H.265	Codec vidéo H.264/H.265

B. Procédure de portage

Le déploiement d'un algorithme nécessite tout d'abord de convertir notre modèle tf.keras en un format ne contenant plus de poids entraînables. Cette opération, dénommée *"graph freezing"*, est disponible dans la librairie Tensorflow et nous permet d'obtenir un modèle au format *pb*. Une fois cette opération effectuée, le portage sur une cible est dépendant des outils proposés par le constructeur.

1) Déploiement sur les Jetsons: Le portage d'un réseau chez le constructeur NVIDIA consiste à convertir le modèle *pb* vers le format ONNX. Le passage vers le format ONNX fige les dimensions de l'image d'entrée et le nombre de cellules dans les grilles en sortie. Ce modèle est ensuite optimisé et compilé en flottant 16bits à l'aide de la librairie TensorRT 8.0.1, appelée à travers son API en Python. Pour la Jetson NX, il est également possible de compiler le modèle en entier 8bits (INT8). La conversion des valeurs flottantes en entiers est réalisée par TensorRT et nécessite une calibration à l'aide des images d'entraînement. Une fois le portage effectué, le modèle s'exécute sur GPU à travers un code Python. Il est à noter que les fonctions de post-traitement et le NMS présentés à la section III-B sont exécutés par ce script sur le CPU.

Un modèle plus complet, intégrant le post-traitement ainsi que le NMS dans le GPU, a été développé. Une fois déployé, ce dernier présente de meilleures performances en temps d'inférence, étant donné la parallélisation du post-traitement de chaque cellule. Ces résultats sont présentés dans la section IV-C.

2) Déploiement sur la ZCU104: Le portage sur la ZCU104 nécessite la librairie VITIS-AI 1.3 de chez Xilinx qui compile le modèle en un jeu d'instructions dédié permettant d'exécuter efficacement de nombreuses architectures de réseaux de neurones à partir de modèles entraînés avec les librairies TensorFlow, PyTorch et Caffe. Dans un premier temps, le déploiement consiste à quantifier en entier 8bits le modèle *pb* à l'aide d'une calibration sur les images d'entraînements. Une fois quantifié, le modèle est compilé pour la cible. La compilation du modèle fige les dimensions de l'image d'entrée et le nombre de cellules dans les grilles en sortie. Une fois le portage effectué, le modèle s'exécute à l'aide d'un code en C++. Les fonctions de post-traitement et le NMS présentés à la section III-B sont exécutés par ce code sur le CPU.

C. Performances

Différentes prédictions, en fonction du portage, sont reprises dans la figure 6 et les performances dans le tableau VII. Du côté des Jetsons, nous constatons que le portage en flottant ne dégrade pas les performances. La calibration en INT8 est optimisée par NVIDIA et ne dégrade l'AP50 que de 2%. Côté ZCU104, la calibration dégrade également de 2% l'AP50.

Nous avons également estimé le temps d'inférence moyen par imagette en fonction de la cible, de la taille du batch (nombre d'images traitées en parallèle) et en faisant également varier la taille des images en entrée (416x416, 640x640 et 1024x1024 pixels). Ces résultats, repris dans le tableau VIII, nous permettent de tirer plusieurs conclusions.

TABLE VII
COMPARATIF DES PERFORMANCES DU MODÈLE SUR LE DATASET DE TEST AVANT ET APRÈS PORTAGE

Cible	Quant.	Précision	Rappel	F1-Score	AP50
modèle tf.keras	FP64	0,416	0,784	0,543	0,676
Jetson NX	FP16	0,415	0,784	0,543	0,675
Jetson NX	INT8	0,386	0,779	0,516	0,654
Jetson Nano	FP16	0,416	0,785	0,544	0,676
ZCU104	INT8	0,388	0,781	0,518	0,659

Pour les Jetsons:

- Le débit n'évolue pas significativement avec la taille du batch et des images. Une justification probable serait que les traitements utilisent la totalité des ressources du GPU et qu'il n'est pas possible de paralléliser d'avantage.
- La majorité du temps de calcul provient du post-traitement (mise en forme des sorties + NMS, cf. section III-B) qui est effectué de manière séquentielle par le CPU. La version du code (*"NX+NMS"*) incluant ce post-traitement dans le GPU permet d'obtenir des performances en temps similaire à l'inférence seule.
- La quantification en entier 8bits ne réduit pas significativement le temps d'inférence
- Les performances en temps de calcul de la Jetson NX sont environ 8 fois supérieures à celle de la Jetson Nano.

Pour la ZCU104:

- Le temps d'inférence est constant quelles que soient les dimensions de l'image: il semblerait que la ZCU n'est pas utilisée à sa capacité maximale et que le modèle peut se redimensionner et être parallélisé en fonction des dimensions de l'image en entrée. Nous constatons de très bon débit pour de grandes taille d'images.
- Similairement aux autres cibles, le post-traitement nécessite un temps de calcul important réduisant le débit total.

Finalement, dans le tableau IX, nous avons également repris l'occupation mémoire ainsi que la consommation du modèle après portage sur différentes cibles. Il est à noter que sur les Jetsons, la mémoire RAM est partagée entre le CPU et le GPU. Son occupation est dès lors directement proportionnelle à la taille du modèle. Pour la ZCU, le modèle est déployé dans le FPGA et la RAM allouée n'est pas un bon indicateur de l'occupation du modèle. La puissance consommée par la carte n'a pas pu être obtenue pour la ZCU104.

Nous constatons que:

- Sur les différentes cibles, la mémoire allouée évolue avec la taille des images et du batch. Cependant, pour les Jetsons, il n'est pas possible de connaître la répartition entre inférence/post-traitement de cette consommation mémoire.
- Pour la Jetson NX, la quantification en entiers permet de réduire drastiquement la consommation de puissance

TABLE VIII
COMPARATIF DU TEMPS D'EXÉCUTION DU MODÈLE EN FONCTION DES PARAMÈTRES DE PORTAGE

			Inférence		Infér. + Post-Trait.	
Cible	taille imag.	Batch Size	T par image	Débit Mpx/s	T par image	Débit Mpx/s
Jetson NX FP16	416	BS1	24ms	7,13	**178ms**	0,97
		BS4	20ms	8,48	172ms	1,01
		BS8	20ms	8,74	168ms	1,03
	640	BS4	46ms	8,88	314ms	1,31
	1024		118ms	8,86	1073ms	0,98
+INT8	416	BS1	22ms	7,87	176ms	0,98
+NMS			n.a.	n.a.	**27ms**	**6,34**
Jetson Nano	416	BS1	188ms	0,92	400ms	0,43
		BS4	195ms	0,89	396ms	0,44
		BS8	190ms	0,91	392ms	0,44
	640	BS4	414ms	0,99	806ms	0,51
	1024		929ms	1,13	2302ms	0,46
ZCU104	416	BS1	83ms	2,08	96ms	1,81
	640		83ms	4,92	206ms	1,99
	1024		83ms	12,62	537ms	1,95

TABLE IX
COMPARATIF DE LA CONSOMMATION DES CIBLES EN FONCTION DES PARAMÈTRES DE PORTAGE

Cible	taille imag.	Batch Size	Conso. RAM Repos (GB)	Δ Conso. RAM (GB)	Conso. Puiss. Repos (W)	Δ Conso. Puiss. (W)
Jetson NX FP16	416	BS1	∼3.1	0,91	∼4.50	1,9
		BS4		0,90		2,4
		BS8		0,99		3,4
	640	BS4		1,39		4,3
	1024			1,56		5,0
+INT8	416	BS1		0,90		0,4
+NMS				0,90		4,2
Jetson Nano	416	BS1	∼1.48	0,90	∼2.7	1,2
		BS4		0,99		1,1
		BS8		1,12		1,5
	640	BS4		0,95		1,2
	1024			1,64		1,4
ZCU 104	416	BS1	∼0,37	0,41	n.a.	n.a.
	640			0,53		n.a.
	1024			0,71		n.a.

V. CONCLUSION ET PERSPECTIVES

Cet article parcourt les différentes étapes du développement d'un détecteur optique de navires. À notre connaissance, ce détecteur à base de réseau neuronal semble être le premier embarquable dans un satellite, ce que nous avons démontré en le déployant sur différentes cibles électroniques (FPGA et GPU) faibles consommations et similaires à des cibles embarquables sur satellite. Dans un premier temps, comme pour tout algorithme d'apprentissage supervisé, la performance et la robustesse du modèle reposent en grande partie sur la qualité, la représentativité et la quantité d'échantillons de la base de données. Pour cette raison, nous avons confié à notre partenaire GEO4I la construction d'une base de données de grande qualité, multiclasses et de haute résolution. Ensuite, le modèle de détection a été sélectionné en tenant compte des contraintes liées au milieu spatial et à l'embarqué. C'est l'algorithme YOLOv3 qui a été retenu, offrant un bon ratio performances/consommation de ressources et étant compatible avec les différentes chaînes de déploiement. En fonction des besoins opérationnels, le compromis entre le taux de fausses alarmes et de non détections peut également être ajusté à en vol, et ce, sans ré-entraînement. Finalement, l'évolution des performances de l'algorithme ainsi que la consommation en ressources hardware ont été comparées après portage sur cibles. Nous avons pu constater que la quantification du modèle peut entraîner de légères pertes. Au niveau du débit de l'algorithme, le post-traitement non parallélisé, et plus particulièrement le NMS, contribue à la majeure partie du temps de traitement. Une implémentation parallélisée sur GPU est proposée, ce qui améliore très significativement l'inférence.

Ces travaux ont déjà permis de mettre en place une démonstration "sur table" : une webcam connectée à nos cartes électroniques embarquant l'algorithme décrit dans cet article permet la détection de navires dans une image satellite imprimée sur un large poster.

En guise de perspective, des études sont actuellement menées afin d'élargir la détection de navires à la tâche de reconnaissance et d'identification. À l'aide de notre base de données multiclasses, nous entraînons notre modèle à classifier ces navires par classe (civil, militaire, autres) et sous-classe (porte-avions, conteneur, pétrolier, etc.). Suite à la mise à jour des chaînes de déploiement, nous envisageons également de porter de nouvelles architectures plus performantes, telles que YOLOX, sur les cibles présentées et sur d'autres, les *TPU* (*Tensor Processing Unit*), spécifiquement développées pour accélérer les réseaux de neurones.

DÉCLARATION DE CONFLIT D'INTÉRÊT

Les auteurs déclarent n'avoir aucun conflit d'intérêt.

REMERCIEMENTS

Ces travaux ont été menés dans le cadre du projet CIAR (*Chaîne Image Autonome et Réactive*) de l'Institut de Recherche Technologique Saint-Exupéry. Les auteurs remercient les partenaires industriels et académiques du projet : Thales Alenia Space, Activeeon, Avisto, ELSYS Design, MyDataModels, Geo4i, Inria et le LEAT/CNRS.

REFERENCES

[1] Fiche technique de a179 - lightning. https://www.recabuk.com/datasheets/systems/aitech/A179.pdf. Accessed: 2022-05-05.

[2] Fiche technique de s-a1760 - venus. https://www.emcomo.de/fileadmin/user_upload/EMCOMO/PDF/S-A1760-Preliminary-Datasheet.pdf. Accessed: 2022-05-05.

[3] European Commission's factsheet about illegal fishing, jun 2021. [Online; accessed 13. May 2022].

[4] Illegal fishing and human rights abuse in China's distant water fleet, May 2022. https://ejfoundation.org/news-media/global-impact-of-illegal-fishing-and-human-rights-abuse-in-chinas-vast-distant-water-fleet-revealed-2[Online; accessed 13. May 2022].

[5] Filippo Maria Bianchi, Martine M. Espeseth, and Njål Borch. Large-Scale Detection and Categorization of Oil Spills from SAR Images with Deep Learning. Remote Sens., 12(14):2260, July 2020.

[6] Jan Martin Brockmann, Till Schubert, and Wolf-Dieter Schuh. An Improved Model of the Earth's Static Gravity Field Solely Derived from Reprocessed GOCE Data. Surv. Geophys., 42(2):277–316, March 2021.

[7] Yuan Dai, Weiming Liu, Haiyu Li, and Lan Liu. Efficient foreign object detection between psds and metro doors via deep neural networks. IEEE Access, 8:46723–46734, 2020.

[8] Ryan Engstrom, Jonathan Samuel Hersh, and David Locke Newhouse. Poverty from Space: Using High-Resolution Satellite Imagery for Estimating Economic Well-Being, December 2017.

[9] experiencor/keras yolo3. Dépot de l'implémentation de yolo-v3 sous keras. https://github.com/experiencor/keras-yolo3, 2020.

[10] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. YOLOX: exceeding YOLO series in 2021. CoRR, abs/2107.08430, 2021.

[11] Xu Han, Lining Zhao, Yue Ning, and Jingfeng Hu. ShipYOLO: An Enhanced Model for Ship Detection. J. Adv. Transp., 2021:1060182, June 2021.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. CoRR, abs/1512.03385, 2015.

[13] R. Hinz, J. I. Bravo, M. Kerr, C. Marcos, A. Latorre, and F. Membibre. EO ALERT: Machine Learning Based On Board Satellite Processing for Very Low Latency Convective Storm Nowcasting. Zenodo, October 2020.

[14] Bo Li, Xiaoyang Xie, Xingxing Wei, and Wenting Tang. Ship detection and classification from optical remote sensing images: A survey. Chin. J. Aeronaut., 34(3):145–163, Mar 2021.

[15] Hao Li, Lianbing Deng, Cheng Yang, Jianbo Liu, and Zhaoquan Gu. Enhanced YOLO v3 Tiny Network for Real-Time Ship Detection From Visual Image. IEEE Access, 9:16692–16706, January 2021.

[16] Ana María Pacheco-Pascagaza, Yaqing Gou, Valentin Louis, John F. Roberts, Pedro Rodríguez-Veiga, Polyanna da Conceição Bispo, Fernando D. B. Espírito-Santo, Ciaran Robb, Caroline Upton, Gustavo Galindo, Edersson Cabrera, Indira Paola Pachón Cendales, Miguel Angel Castillo Santiago, Oswaldo Carrillo Negrete, Carmen Meneses, Marco Iñiguez, and Heiko Balzter. Near Real-Time Change Detection System Using Sentinel-2 and Machine Learning: A Test for Mexican and Colombian Forests. Remote Sens., 14(3):707, February 2022.

[17] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. CoRR, abs/1506.02640, 2015.

[18] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. CoRR, abs/1804.02767, 2018.

[19] Sandeep Kumar Singh, Shradha Fowdur, Jakob Gawlikowski, and Daniel Medina. Leveraging Evidential Deep Learning Uncertainties with Graph-based Clustering to Detect Anomalies. ResearchGate, July 2021.

[20] Xilinx. Vitis-AI 2.0, May 2022. https://github.com/Xilinx/Vitis-AI/tree/master/models/AI-Model-Zoo/model-list/pt_yolox_TT100K_640_640_73G_2.0[Online; accessed 13. May 2022].

[21] Tianwen Zhang, Xiaoling Zhang, Jianwei Li, Xiaowo Xu, Baoyou Wang, Xu Zhan, Yanqin Xu, Xiao Ke, Tianjiao Zeng, Hao Su, Israr Ahmad, Dece Pan, Chang Liu, Yue Zhou, Jun Shi, and Shunjun Wei. SAR Ship Detection Dataset (SSDD): Official Release and Comprehensive Data Analysis. Remote Sens., 13(18):3690, September 2021.

[22] SuYu Zhou and Jun Yin. YOLO-Ship: A Visible Light Ship Detection Method. In 2022 2nd International Conference on Consumer Electronics and Computer Engineering (ICCECE), pages 113–118. IEEE, January 2022.

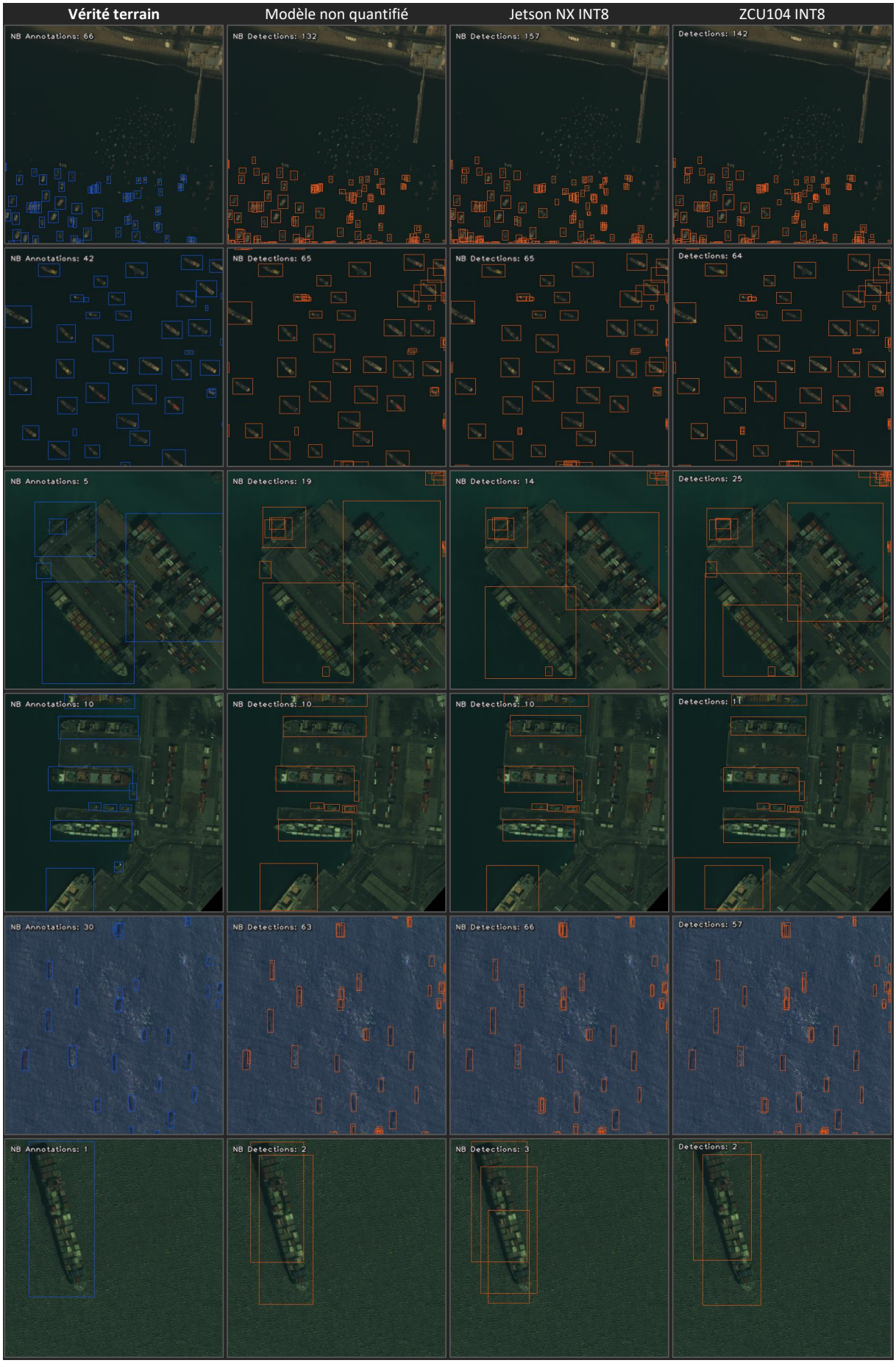

Fig. 6. Exemples de prédictions du modèle *YOLO-v3* sur six imagettes appartenant au jeu de test. Images de gauche à droite : vérité terrain, prédiction du modèle non quantifié (en flottant 16 ou 64bits), prédiction quantifiée sur la Jetson NX en entiers 8bits, prédiction sur la ZCU en entiers 8bits. La réalité terrain est représentée en bleu et les prédictions en rouge. Imagery Products © 2018 Maxar Technologies.