MobileNet SSD : étude d'un détecteur d'objets embarquable entraîné sans images réelles

Julia Cohen, Carlos F Crispim-Junior, Jean-Marc Chiappa, Laure Tougne

MobileNet SSD : étude d'un détecteur d'objets embarquable entraîné sans images réelles

J. Cohen[1,2] C. Crispim-Junior[1] J.-M. Chiappa[2] L. Tougne[1]

[1] Univ Lyon, Lyon 2, LIRIS, F-69676 Lyon, France
[2] DEMS, Saint Bonnet de Mûre, France

{julia.cohen, carlos.crispim-junior, laure.tougne}@liris.cnrs.fr
jean-marc.chiappa@groupe-dems.fr

Résumé

La détection d'objets en environnement industriel est un défi, d'une part à cause de l'absence d'images réelles pour l'apprentissage, et d'autre part à cause de la contrainte de temps réel requise pour l'algorithme, en général embarqué dans un dispositif mobile. Grâce aux données de conception 3D, il est cependant possible de créer une base d'images synthétiques annotées automatiquement pour entraîner un réseau de neurones artificiel, moyennant une performance limitée par l'écart de domaine avec la réalité. Dans cet article, nous étudions la performance d'un réseau de neurones à convolutions (CNN) pour la détection d'objets en temps réel et avec une faible empreinte mémoire : Single-Shot Detector (SSD), avec un réseau MobileNet comme extracteur de caractéristiques. Nous montrons qu'une stratégie d'augmentation adéquate permet d'entraîner SSD avec uniquement des images synthétiques photoréalistes pour détecter des objets industriels dans des images réelles. En particulier, sur la base de données T-LESS, SSD obtient une meilleure performance qu'un réseau Mask R-CNN, avec MobileNet-V2 et MobileNet-V3 comme extracteur de caractéristiques. Nos résultats montrent une amélioration de la précision d'environ 10% obtenue grâce à l'augmentation des données. Le code et les modèles entraînés seront disponibles en ligne.

Mots Clef

Détection d'objets, base de données synthétique, applications mobiles.

Abstract

In an industrial environment, object detection is a challenging task due to the absence of real images during training and real-time requirements during the inference, usually embedded in a mobile device. Using 3D models, it is however possible to create a synthetic dataset of automatically annotated images to train a neural network, although the performance on real images is limited by the domain gap. In this paper, we study the performance of a Convolutional Neural Network (CNN) designed to detect objects in real-time and with low memory footprint: Single-Shot Detector (SSD) with a MobileNet as backbone. We show that extensive data augmentation enables SSD to learn on synthetic images only, and correctly detect industrial objects in real images. On the T-LESS dataset, SSD performs better than Mask R-CNN trained on the same synthetic images, with both MobileNet-V2 and MobileNet-V3 Large as its backbone. Our results also show a 10% improvement enabled by a well-chosen augmentation strategy. The code and trained models will be made available.

Keywords

Object detection, synthetic dataset, mobile applications

1 Introduction

Le secteur industriel est l'un des nombreux domaines dans lesquels l'automatisation de nombreuses tâches est possible grâce à la vision par ordinateur. De la maintenance à la manipulation par un bras robotisé, une détection précise d'objets variés est nécessaire. En parallèle, les réseaux de neurones artificiels ont vu leurs capacités décuplées, notamment grâce à l'amélioration constante des capacités de calcul des ordinateurs et la disponibilité de bases de données à grande échelle comme ImageNet [4] et MS COCO [21]. Ces bases de données, comprenant des images et les annotations qui leur sont associées, permettent d'ana-

FIGURE 1 : Images d'entraînement synthétiques (ligne supérieure) et images de test réelles (ligne inférieure) de la base de données T-LESS [18].

lyser le contenu d'images avec différents niveaux de précision. Entre la classification, à l'échelle d'une image entière, et la segmentation, à l'échelle d'un pixel, la détection d'objets consiste à localiser dans une image les différents objets (via une boîte englobante) et à associer une catégorie à chacun de ces objets. Les nombreux challenges proposés au fil des années, consistant à résoudre l'une ou l'autre de ces tâches sur une base de données spécifique, ont permis de grandes avancées dans les méthodes de traitement d'images. Parmi ces méthodes, les réseaux de neurones à convolution (CNN) ont montré une capacité supérieure à résoudre ces tâches à partir de très grandes quantités d'images.

Dans le domaine industriel, il est souvent nécessaire de reconnaître dans des images des objets spécifiques à l'application, que l'on ne retrouve pas dans ces bases de données habituelles. Il devient alors nécessaire de créer des bases de données d'apprentissage spécifiques, une tâche qui demande beaucoup de ressources si elle est effectuée à la main. Par exemple pour détecter certains objets particuliers, il faut acquérir des images de ces objets dans des conditions variées (position, orientation, environnement, luminosité, etc.) et annoter chaque objet avec sa boîte englobante et sa catégorie. Deux approches peuvent être adoptées pour simplifier ce dur labeur : l'utilisation d'un outil d'annotation semi-automatique, ou la création et annotation automatique d'images synthétiques [3]. Cette seconde approche est préférée lorsque les modèles 3D des objets d'intérêt sont disponibles, ce qui est le cas pour la plupart des applications industrielles. De plus, ces applications nécessitent une méthode de traitement d'images sur un flux arrivant en temps réel, et sont déployées dans des appareils embarqués et mobiles qui ont des capacités de mémoire et de vitesse de calcul limitées. Ces conditions requises ne sont en général pas compatibles avec l'état-de-l'art, des réseaux de neurones de taille toujours plus importante et nécessitant toujours plus de capacité de calcul.

Nous étudions dans cet article la performance du réseau SSD avec différentes architectures de type MobileNet pour l'extraction de caractéristiques, les deux étant conçus pour des applications mobiles. Après avoir entraîné ces détecteurs d'objets avec des images synthétiques obtenues à partir des modèles 3D des objets d'intérêt, leur performance est évaluée sur des images réelles. Nous évaluons également différents niveaux de transformations de couleur et géométriques pour combler l'écart de domaine entre les images synthétiques et les images réelles. Ces approches sont évaluées sur la base de données publique T-LESS [18], une base de données dédiée à l'estimation de pose en 3D et pour laquelle les résultats de détection en 2D ne sont à notre connaissance pas fournis dans la littérature.

2 Travaux antérieurs

Dans cette section, nous décrivons les approches principales pour résoudre la tâche de détection d'objets, l'architecture des réseaux MobileNet, ainsi que différentes méthodes d'apprentissage sur images synthétiques.

2.1 Détection d'objets

Les réseaux à convolution permettent d'obtenir une très bonne performance sur la tâche de détection d'objets. Alors que les premiers modèles n'étaient composés que de quelques couches, des architectures plus larges et plus profondes les ont surpassés. Ces réseaux sont en général composés de deux sous-réseaux : une première étape qui propose des régions d'intérêt, et une seconde étape analysant chaque région proposée [14, 25]. Cette architecture en deux parties (*two-stage detectors*) augmente de façon conséquente le nombre de couches et de paramètres dans le réseau, ce qui le rend inadapté pour des applications embarquées ne disposant que de peu de mémoire et de capacités de calcul réduites. Ces inconvénients ont encouragé le développement de réseaux plus petits, dits *one-stage detectors*, qui localisent et identifient les objets en une seule étape pour un temps d'inférence largement réduit [11, 24].

2.2 Réseaux MobileNets

Comme la plupart des réseaux de neurones, les détecteurs d'objets sont composés d'un extracteur de caractéristiques suivi d'un ensemble de couches dédié à la résolution de la tâche. L'extraction de caractéristiques étant une étape cruciale, de nombreuses architectures ont été proposées, comme la série des réseaux MobileNet. MobileNet-V1 [5] a d'abord introduit la notion de convolution séparable et en profondeur (*depth-wise separable convolution*), ainsi que deux nouveaux hyper-paramètres pour ajuster la taille du réseau. Ces propositions permettent de diminuer le nombre de poids dans l'architectur sans perdre en performance par rapport à un réseau classique. Avec MobileNet-V2 [27], les couches de convolution simples ont été remplacées par des blocs résiduels inversés (*inverted residual blocks*). Ces blocs résiduels sont plus efficaces en mémoire et encouragent l'information à être compressée dans un espace de plus petite dimension. L'architecture MnasNet [9] s'appuie sur MobileNet-V2 et lui ajoute un module *squeeze-and-excite* pour des caractéristiques de meilleure qualité. Enfin, MobileNet-V3 [6] combine les différents éléments proposés précédemment en définissant comme bloc de base la séquence de modules *inverted residual+squeeze-and-excite*. L'architecture est encore optimisée en remplaçant les fonctions d'activation non-linéaires par leur équivalent linéaire par morceaux, plus efficaces à calculer. Un algorithme de recherche d'architecture est utilisé pour identifier la séquence de couches, ainsi que les paramètres de chaque couche (profondeur et résolution des couches, type de non-linéarité...), qui réduisent le nombre de poids tout en améliorant la performance. Alors que MobileNet-V3 a un temps d'inférence inférieur et obtient une meilleure performance de classification que son prédécesseur MobileNet-V2, son adoption par la communauté est encore très limitée.

2.3 Images d'entraînement synthétiques

L'entraînement de réseaux à convolution est possible sans aucune image réelle en créant des images synthétiques à partir de modèles 3D. L'approche la plus naïve consiste à coller des projections 2D des objets 3D sur des images de fond réelles [17, 19, 3]. Si cette approche fonctionne pour reconnaître ces mêmes objets dans certaines images réelles, la performance est limitée. Ceci est dû à l'absence de réalisme, en particulier au niveau des contours des objets. Une approche plus performante consiste à générer la totalité de l'image de façon synthétique en utilisant un moteur de rendu dédié aux jeux vidéos, tels que Unity[1] ou Unreal Engine[2], capable de créer une scène ou un environnement entièrement virtuels, avec les modèles à l'intérieur [13, 26]. Cependant, les limitations de ces moteurs de rendu, comme la difficulté à faire varier automatiquement des paramètres et à générer des annotations, ont poussé la communauté à développer des moteurs de rendu photoréalistes spécialement pour l'apprentissage automatique, comme BlenderProc [8] par exemple. Une autre approche consiste à utiliser des techniques spécifiques à la Réalité Augmentée pour fondre les éléments synthétiques dans le monde réel [1]. Enfin, la randomisation de domaine est une approche plébiscitée lorsqu'il est facile de modifier l'apparence des images synthétiques, puisqu'elle permet de générer suffisamment de variations dans les images d'entraînement pour que les images réelles apparaissent comme une variation parmi d'autres pour le réseau de neurones [28, 7].

Dans la suite de cet article, nous évaluons différents détecteurs d'objets basés sur les extracteurs de caractéristiques MobileNet-V2 et V3, dans le cadre de l'apprentissage sur images synthétiques photoréalistes uniquement.

3 Méthode proposée

3.1 Architectures étudiées

Parmi les architectures *single-stage* pour la détection d'objets, nous avons choisi le modèle SSD [11], dont la taille dépend surtout de son extracteur de caractéristiques. Alors que l'extracteur original est communément remplacé par ResNet-50 [15] pour de meilleures performances ou par MobileNet-V2 [27] pour une inférence en temps réel, nous avons choisi de comparer les performances obtenues avec MobileNet-V2 et MobileNet-V3. MobileNet-V3 existe en deux tailles, notées *Large* et *Small*, qui comportent 15 et 11 blocs de base, et des profondeurs maximum de 960 et 576 couches de caractéristiques, respectivement. Nous évaluons ici les trois architectures possibles, identifiées comme V2-SSD, V3-SSD et V3small-SSD.
Le réseau SSD est composé d'un ensemble de couches "*extra*" et de couches de prédiction appelées "en-tête" (*header layers*), qui produisent la position des objets dans l'image ainsi que leur catégorie (Figure 2). Les couches *extra* sont ajoutées à la suite de l'extracteur de caractéristiques, alors que les couches d'en-tête prennent comme entrée les cartes de caractéristiques à plusieurs niveaux de résolution : deux en-têtes sont connectés directement à l'extracteur de caractéristiques, puis un en-tête par couche *extra*. Une dernière étape de "suppression de non-maxima" (*NMS*) est ajoutée comme post-traitement pour éliminer les prédictions multiples. Le modèle complet est construit comme indiqué dans [27] et [6] : le premier en-tête est connecté à la couche C4 de MobileNet, le second en-tête ainsi que les couches *extra* sont connectés à la couche C5. L'architecture V3-SSD est représentée dans la Figure 2, avec la profondeur des cartes de caractéristiques indiquées en-dessous.

3.2 Augmentation des données

Pour combler l'écart de domaine entre les images synthétiques et la réalité, nous modifions les images synthétiques photoréalistes à l'aide de différentes méthodes d'augmentation. Nous utilisons pour cela la bibliothèque logicielle *Albumentations* [2] pour appliquer des distorsions colorimétriques et géométriques, chacune avec une probabilité aléatoire d'être appliquée, et certaines d'entre-elles avec des paramètres pris aussi de façon aléatoire dans un intervalle défini expérimentalement. Les différents types de modifications sont : une altération de la luminosité, du contraste, de la saturation et de la teinte des images ; un changement de couleur (*color shift*) ; du flou moyen, Gaussien, médian ou de mouvement ; du bruit Gaussien, multiplicatif ou de Poisson ; une symétrie verticale. Après ces augmentations, un morceau de 400x400 pixels est extrait de façon aléatoire dans l'image, et redimensionné à la taille d'entrée du réseau, soit 224x224 pixels.

4 Expériences et Résultats

4.1 Base de données

Nous évaluons notre méthode sur la base de données T-LESS [18], dédiée à la détection et à l'estimation de pose 6D des objets dans des images, et contenant les modèles 3D et des images RGB-D correspondant à 30 objets de type industriel, sans texture ni couleur. 50000 images synthétiques photoréalistes ont également été publiées dans le

TABLEAU 1 : Performance de la détection d'objets sur la base de données T-LESS. La précision moyenne (mAP) est en %.

Architecture	mAP
Mask R-CNN	32,8
V3small-SSD (*aug. faible*)	18,6
V3-SSD (*aug. faible*)	36,3
V2-SSD (*aug. faible*)	38,3
V3small-SSD (*aug. forte*)	23,5
V3-SSD (*aug. forte*)	46,1
V2-SSD (*aug. forte*)	**47,7**

[1] unity.com
[2] unrealengine.com

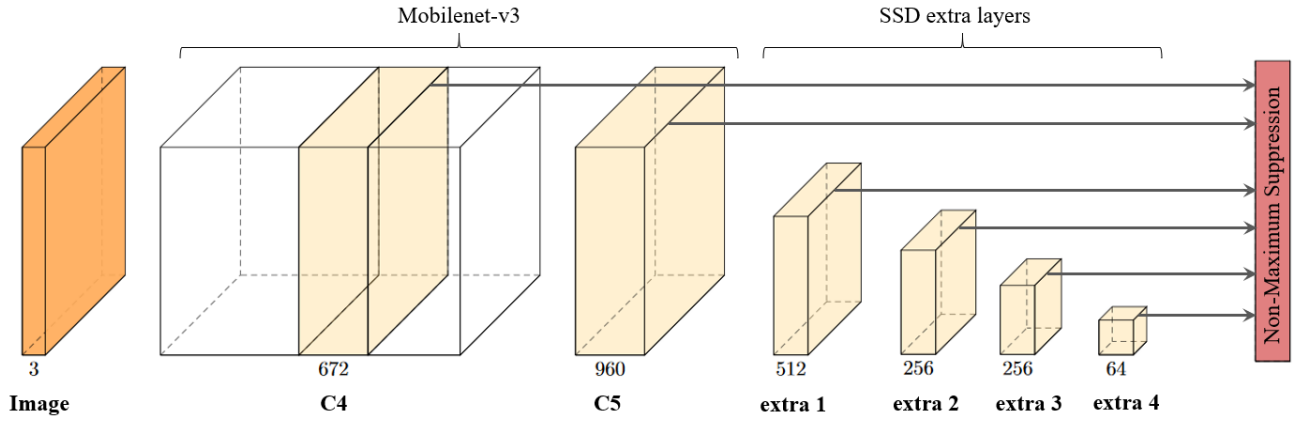

FIGURE 2 : Architecture du détecteur d'objets : SSD avec extracteur de caractéristiques MobileNet-V3. Les flèches correspondent aux couches de prédiction du modèle SSD (régression des coordonnées de boîtes englobantes et classification de chaque prédiction).

TABLEAU 2 : Temps d'inférence pour une image (224x224 pixels), mémoire allouée sur le GPU et nombre de paramètres.

Architecture	Inférence (ms)	Mémoire (M)	Paramètres (M)
Mask R-CNN	463	181,4	44,0
V3-SSD	35	20,1	4,9
V2-SSD	28	14,5	3,5
V3small-SSD	33	10,6	2,6

cadre de l'édition 2020 du BOP challenge [10], permettant à tous les participants d'utiliser les mêmes images d'entraînement. Ces images ont été générées à l'aide de l'outil de rendu physique réaliste BlendeProc [8] (Figure 1, ligne supérieure). BlenderProc permet de créer une scène virtuelle avec de nombreux paramètres pris de façon aléatoire : choix des objets ; choix des objets distracteurs (objets que l'on ne souhaite pas détecter) ; position des objets, caméras et lumières. Les annotations sont automatiquement générées afin de créer sans effort de très nombreuses images d'entraînement photoréalistes. Nous avons entraîné nos différentes architectures avec ces images synthétiques uniquement, dont 1000 ont servi à la validation des hyperparamètres. Les images de test sont les 1000 images réelles acquises par le capteur Primesense CARMINE 1.09 (Figure 1, ligne inférieure). Nous nous sommes concentrés sur la détection d'objets dans les images en couleur, laissant les modalités et annotations supplémentaires pour une étude ultérieure.

4.2 Métriques d'évaluation et comparaisons

Pour chaque architecture, nous avons calculé la précision moyenne mAP (*mean Average Precision*) telle qu'elle est définie dans le challenge Pascal VOC [12], en utilisant l'implémentation proposée par Padilla *et al.* [23] et la méthode "*Every-Point Interpolation*" qui utilise tous les points de la courbe Précision-Rappel (par opposition à la méthode "*Eleven-Point Interpolation*" qui n'utilise que 11 points). Nous avons également relevé les temps d'inférence et la taille des différentes architectures étudiées.

Nous avons comparé nos résultats à ceux d'un réseau Mask R-CNN [16] avec ResNet-50-FPN comme extracteur de caractéristiques, entraîné sur les mêmes images synthétiques. Mask R-CNN étant un réseau en deux étapes (*two-stage detector*) de taille bien plus importante que SSD, il obtient en général une meilleure précision. Cependant, l'inférence avec une telle architecture est beaucoup plus lente, ce qui rend difficile son utilisation pour des applications mobiles, embarquées et en temps réel. Nous utilisons le modèle entraîné par les gagnants du BOP challenge 2020 [20], qui sert de première étape dans leur méthode d'estimation de pose. Nous appliquons le même calcul de performance aux prédictions données par ce réseau.
Pour identifier l'influence de l'augmentation des images que nous avons appliquée, nous avons également entraîné nos différentes architectures avec une augmentation moins forte et similaire à celle appliquée à Mask R-CNN. Celle-ci comprend l'application de flou, la modification de la netteté des images, ainsi que des altérations de contraste, luminosité et couleur. Dans la suite de cet article, *aug. faible* fait référence à l'augmentation faible des données, et *aug. forte* à notre augmentation plus importante décrite Section 3.2.
Enfin, il est à noter que les images d'entraînement ont une taille de 540x720 pixels pour Mask R-CNN et 224x224 pour SSD. Pour une comparaison plus juste du temps d'inférence, les images sont redimensionnées à 224x224 pour Mask R-CNN, bien que cela ne corresponde pas à la meilleure performance.

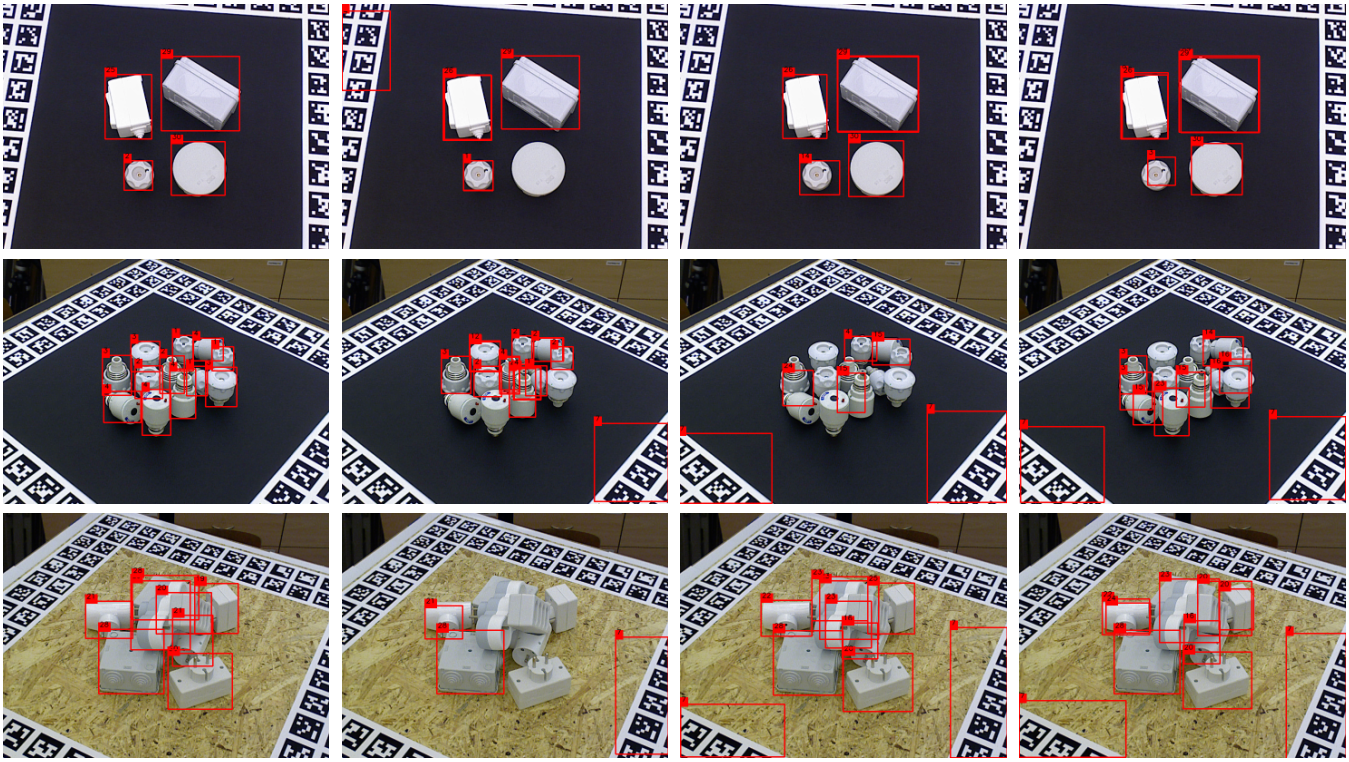

FIGURE 3 : Résultats de la détection d'objets. De gauche à droite : vérité-terrain, Mask R-CNN [20], V3-SSD (notre méthode), V2-SSD (notre méthode).

4.3 Détails d'implémentation

Le réseau MobileNet-V2 utilisé est le réseau pré-entraîné sur la base de données ImageNet [4] fourni par la bibliothèque *torchvision* [22]. Le réseau MobileNet-V3 n'étant pas encore disponible dans *torchvision* au moment de cette étude, nous avons utilisé une implémentation PyTorch disponible sur GitHub [3] qui fournit également les poids pré-entraînés sur ImageNet, pour les réseaux *Large* et *Small*. Cette architecture étant basée sur une première version de l'article avant sa publication lors de la conférence ICCV, l'architecture diffère dans la taille de la 14e couche du modèle *Large* (la profondeur est de 672 au lieu de 960).
Toutes les architectures ont été optimisées jusqu'à convergence par descente de gradient stochastique (*SGD*) et les paramètres suivants ont été utilisés : *learning rate* de $5e{-}2$, *momentum* de $9e{-}1$, *weight decay* de $12e{-}6$, et une taille de *batch* de 32 images. Ces hyper-paramètres ont été identifiés expérimentalement par recherche aléatoire avec V3-SSD et appliqués aux autres architectures. Le code correspondant est disponible sur GitHub [4].

4.4 Résultats quantitatifs

Les résultats obtenus (Tableau 1) indiquent une meilleure précision avec les architectures V2-SSD et V3-SSD qu'avec Mask R-CNN pour le même niveau d'augmentation de données *aug. faible*. Avec des augmentations plus variées et complexes (*aug. forte*), la performance augmente encore, de $36,33\%$ à $46,1\%$ pour V3-SSD, et de $36,3\%$ à $47,7\%$ pour V2-SSD. Il est intéressant de noter que MobileNet-V2 obtient une meilleure précision que son successeur MobileNet-V3 sur les images de test alors que l'inverse est vrai sur l'ensemble de validation. Cela indiquerait que MobileNet-V2 a une meilleure capacité de généralisation aux images réelles. Une raison peut être son nombre de paramètres inférieur (Tableau 2), ce qui limite l'apprentissage sur les données d'entraînement de caractéristiques trop spécifiques aux image synthétiques. De même, Mask R-CNN ayant une plus grande capacité d'apprentissage que SSD, des caractéristiques représentatives des images synthétiques ont peut-être été apprises, ce qui empêche la généralisation aux images réelles. En revanche, V3small-SSD obtient les résultats les plus bas peu importe la stratégie d'augmentation de données appliquée.
Afin d'extraire des caractéristiques plus adaptées aux images réelles, certains travaux antérieurs ont évalué l'utilité de fixer les poids de l'extracteur de caractéristiques après son pré-entraînement sur ImageNet. Alors que cela a amélioré les résultats pour certains [17, 3], d'autres ont vu leur performance réduite [7] par rapport à un entraînement du réseau complet avec les images synthétiques. Dans notre cas, le réseau ne converge pas vers une solution satisfaisante lorsque seuls les poids du réseau SSD sont modifiés. Une hypothèse est que les caractéristiques apprises sur

[3] https ://github.com/d-li14/mobilenetv3.pytorch
[4] https ://github.com/cohenINSA/synthetic_ssd

les images de la base de données ImageNet s'appuient sur les couleurs et textures des objets, des informations non pertinentes pour faire une distinction entre les objets non texturés de la base de données T-LESS. Il semble donc nécessaire que les images synthétiques soient photoréalistes pour que l'extracteur de caractéristiques apprenne les caractéristiques importantes pour ces objets, qu'il va ensuite retrouver dans les images réelles.
En ce qui concerne le temps d'inférence, SSD est environ 10 fois plus rapide que Mask R-CNN (Tableau 2). Soulignons que la moitié du temps pris par SSD correspond à l'étape de suppression de non-maxima (*NMS*), qui n'est pas optimisée ici. La taille du modèle en mémoire est mesurée comme la mémoire allouée lorsque le modèle est chargé sur le GPU. V2-SSD est à la fois plus petit et plus rapide que V3-SSD, tout en généralisant mieux aux images réelles, ce qui justifie son utilisation pour des applications embarquées. Avec notre implémentation, V3-SSD et V3small-SSD ont une différence négligeable de temps d'inférence, bien que ce dernier contienne moitié moins de paramètres.

4.5 Résultats qualitatifs

Les détections obtenues avec Mask R-CNN, V3-SSD et V2-SSD sur les mêmes images de test, ainsi que la vérité terrain, sont représentés sur la Figure 3. Une erreur systématique est la détection des marqueurs autours de la scène comme l'un des objets d'intérêt (souvent l'objet de catégorie 7, un bloc rectangulaire avec 3 prises). Alors que Mask R-CNN manque de nombreux objets, les architectures SSD identifient au contraire plusieurs fois les mêmes objets, en leur attribuant des catégories différentes. Cette erreur peut provenir du calcul de NMS que nous avons choisi, qui autorise plusieurs catégories pour la même prédiction si leur indice de confiance est suffisament élevé. Si cette approche augmente le nombre de faux positifs, elle permet aussi une meilleure détection des objets superposés dans certaines images. Enlever ces détections multiples avec l'application du calcul standard de NMS permettrait de supprimer ces doublons, mais souvent en supprimant la bonne catégorie et en ne gardant que les faux positifs.

5 Conclusion

Nous avons évalué la pertinence d'un réseau de neurones profond à faible empreinte mémoire qui peut être embarqué pour la détection d'objets en milieu industriel, avec un entraînement entièrement sur images synthétiques. Nous avons observé qu'un réseau de taille plus importante tel que Mask R-CNN n'est pas ici la solution optimale, puisque nos architecture MobileNet-V2 SSD et MobileNet-V3 SSD obtiennent une meilleure précision dans la reconnaissance d'objets non texturés de la base de données T-LESS. Nos expériences soulignent l'importance de l'augmentation de données appliquée aux images d'entraînement. La meilleure performance est obtenue avec le réseau SSD et MobileNet-V2 comme extracteur de caractéristiques, pour une inférence en temps réel et une faible place en mémoire. Ces travaux peuvent être poursuivis par l'application des mêmes techniques d'augmentation de données au réseau Mask R-CNN, ou par l'utilisation d'une méthode de transfert de domaine pour rendre les images synthétiques plus réalistes avant l'entraînement d'un détecteur d'objets. L'utilisation conjointe d'images réelles et synthétiques peut également être une piste pour réduire l'écart entre les domaines.

Remerciements

Ce travail est permis par le financement CIFRE n.2018/0872 de l'ANRT.

Références

[1] H. A. Alhaija, S. K. Mustikovela, L. M. Mescheder, A. Geiger, and C. Rother. Augmented reality meets computer vision : Efficient data generation for urban driving scenes. *IJCV*, 126(9) :961–972, 2018.

[2] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin. Albumentations : fast and flexible image augmentations. *Information-an International Interdisciplinary Journal*, 11(2) :125, 2020.

[3] J. Cohen, C. Crispim-Junior, C. Grange-Faivre, and L. Tougne. CAD-based learning for egocentric object detection in industrial context. In *VISAPP*, volume 5, pages 644–651, 2020.

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet : A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009.

[5] A. Howard *et al.* MobileNets : Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv :1704.04861*, 2017.

[6] A. Howard *et al.* Searching for MobileNetV3. In *ICCV*, pages 1314–1324, 2019.

[7] J. Tremblay *et al.* Training deep networks with synthetic data : Bridging the reality gap by domain randomization. In *CVPR Workkshops*, pages 969–977, 2018.

[8] M. Denninger *et al.* BlenderProc. *arXiv preprint arXiv :1911.01911*, 2019.

[9] M. Tan *et al.* MnasNet : Platform-aware neural architecture search for mobile. In *CVPR*, pages 2820–2828, 2019.

[10] T. Hodan *et al.* BOP : Benchmark for 6d object pose estimation. In *ECCV*, pages 19–35, 2018.

[11] W. Liu *et al.* SSD : Single shot multibox detector. In *ECCV*, pages 21–37, 2016.

[12] M. Everingham, L. Gool, C. K. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) challenge. *IJCV*, 88(2) :303–338, 2010.

[13] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual worlds as proxy for multi-object tracking analysis. In *CVPR*, pages 4340–4349, 2016.

[14] R. Girshick. Fast R-CNN. In *ICCV*, pages 1440–1448, 2015.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[16] K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask R-CNN. *TPAMI*, 42(2) :386–397, 2020.

[17] S. Hinterstoisser, V. Lepetit, P. Wohlhart, and K. Konolige. On pre-trained image features and synthetic images for deep learning. In *ECCV Workshops*, pages 682–697, 2018.

[18] T. Hodan, P. Haluza, S. Obdrzalek, J. Matas, M. Lourakis, and X. Zabulis. T-LESS : An rgb-d dataset for 6d pose estimation of texture-less objects. In *WACV*, pages 880–888, 2017.

[19] J. Josifovski, M. Kerzel, C. Pregizer, L. Posniak, and S. Wermter. Object detection and pose estimation based on convolutional neural networks trained with synthetic data. In *IROS*, pages 6269–6276, 2018.

[20] Y. Labbé, J. Carpentier, M. Aubry, and J. Sivic. CosyPose : Consistent multi-view multi-object 6d pose estimation. In *ECCV*, pages 574–591, 2020.

[21] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco : Common objects in context. In *European Conference on Computer Vision*, pages 740–755, 2014.

[22] S. Marcel and Y. Rodriguez. Torchvision the machine-vision package of torch. In *ACM Multimedia*, pages 1485–1488, 2010.

[23] R. Padilla, S. L. Netto, and E. A. B. da Silva. A survey on performance metrics for object-detection algorithms. In *IWSSIP*, pages 237–242, 2020.

[24] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once : Unified, real-time object detection. In *CVPR*, pages 779–788, 2016.

[25] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN : Towards real-time object detection with region proposal networks. *TPAMI*, 39(6) :1137–1149, 2017.

[26] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez. The SYNTHIA Dataset : A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR*, pages 3234–3243, 2016.

[27] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. MobileNetV2 : Inverted residuals and linear bottlenecks. In *CVPR*, pages 4510–4520, 2018.

[28] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IROS*, pages 23–30, 2017.