LeYOLO, nouvelle architecture embarquée pour la détection d'objets

Lilian Hollard[1], Lucas Mohimont[1], Nathalie Gaveau[2], Luiz Angelo Steffenel[1]

[1] Université de Reims Champagne-Ardenne,CEA, LRC DIGIT, LICIIS, Reims, France
[2] Université de Reims Champagne-Ardenne, INRAE, RIBP USC 1488, Reims, France

Résumé

La réduction du coût de calcul des réseaux neuronaux profonds est essentielle pour la détection d'objets en temps réel. Pourtant, les récents progrès reposent surtout sur le matériel plutôt que sur l'optimisation des modèles. Cela se remarque notamment dans les dernières architectures YOLO, où la vitesse prime sur la légèreté.

Pour répondre à ce défi, nous introduisons deux contributions majeures. D'abord, LeNeck, un cadre de détection rapide et précis, réduisant le nombre de paramètres. Ensuite, LeYOLO, un modèle optimisé pour YOLO, combinant compacité et haute précision. Ces solutions sont idéales pour les dispositifs à faible consommation, y compris les microcontrôleurs.

Mots-clés

Vision par ordinateur, Optimisation des réseaux de neurones, Architecture de réseaux de neurones, Microcontrôleur.

Abstract

Reducing the computational cost of deep neural networks is essential for real-time object detection. However, recent progress has been based mainly on hardware rather than model optimization. This is particularly noticeable in the latest YOLO architectures, where speed precedes lightness.

To address this challenge, we are introducing two significant contributions. Firstly, LeNeck is a fast and accurate detection framework that reduces the number of parameters. Secondly, LeYOLO is a model optimized for YOLO that combines compactness and high precision. These solutions are ideal for low-power devices, including microcontrollers.

Keywords

Computer vision, Neural Network Optimization, Neural Network Architecture, Microcontrollers.

1 Introduction

Un calcul efficace, un traitement en temps réel et une exécution à faible latence sont essentiels pour les dispositifs edge alimentés par l'IA, y compris les drones autonomes, les systèmes de surveillance, l'agriculture intelligente et les caméras intelligentes. Bien que le cloud computing offre une alternative pour exécuter des modèles puissants, il présente des inconvénients tels que la latence, les contraintes de bande passante et les risques de sécurité [1, 43, 39]. Dans les applications pratiques de détection d'objets, les avancées de l'apprentissage profond se sont principalement concentrées sur l'optimisation de la vitesse pour les GPU à haute performance, souvent au détriment de l'efficacité sur le matériel à faible puissance.

Initialement introduit par Joseph Redmon et al. [25], les modèles YOLO sont connus pour leur vitesse d'inférence en détection d'objets. Ces modèles ont connu des améliorations architecturales significatives au fil des ans, tirant parti des puissances de calcul modernes.

Malgré leur vitesse inhérente, il y a eu un mouvement notable dans le développement des modèles YOLO ces dernières années. Avec les rapides avancées des capacités des GPU et les nouvelles innovations matérielles, l'accent s'est déplacé des modèles légers à ceux privilégiant la vitesse d'inférence [15, 17, 37, 14, 38]. En conséquence, les modèles YOLO sont devenus nettement plus rapides malgré l'augmentation des paramètres et des FLOP [1].

Notre travail met en évidence le fait que, malgré leur vitesse impressionnante sur les GPU, les modèles YOLO ont du mal avec le matériel sans accélération pour l'IA tel que les microcontrôleurs et les micro-ordinateurs embarqués. Par exemple, sur les microcontrôleurs STMicroelectronics - largement utilisés dans la robotique et les applications IoT - les modèles YOLO modernes prennent plus d'une seconde par inférence sur les puces les plus puissantes (Section 4, Tableau 4), les rendant inadaptés aux applications en temps réel. Sur les microcontrôleurs moins puissants, des améliorations supplémentaires sont nécessaires pour réduire le temps d'inférence, un défi que nous abordons dans cette étude. Ces contraintes posent un défi critique pour les industries dépendant de l'IA à faible puissance, où l'efficacité énergétique, la petite taille du modèle et l'utilisation optimisée des ressources sont essentielles.

Dans les tâches de classification, les recherches sur l'optimisation des comptes de paramètres et des coûts computationnels ont produit des modèles notables comme Mobile-

1. Nous décrivons les opérations en virgule flottante comme FLOP, définissant toutes les opérations arithmétiques que le réseau de neurones nécessite pour effectuer une inférence. Dans notre article, 1 FLOP est environ 2 MADD ou 2 MACC. Ainsi, la variation des benchmarks tels que MobileNet diffère de leur article original.

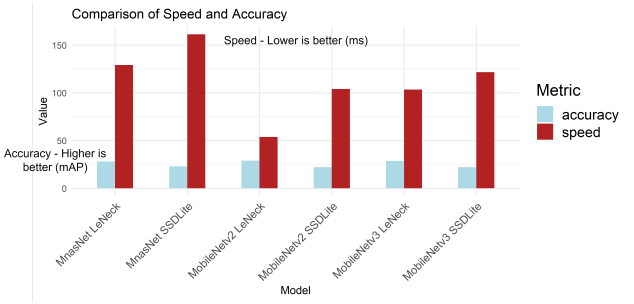

FIGURE 1 – Différence de vitesse (ms) et de précision (mAP) entre SSDLite et LeNeck sur STM32N6570-DK.

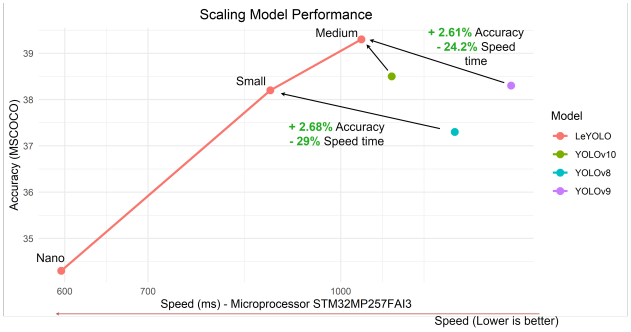

FIGURE 2 – Comparaison entre LeYOLO et les YOLO modernes, démontrant une meilleure précision pour moins de temps d'exécution sur STM32MP257FAI3.

Nets [12, 27, 11] et EfficientNets [30, 31]. Bien que ces modèles soient remarquables, ils sont principalement reconnus pour leurs capacités de classification exceptionnelles plutôt que pour la détection d'objets. Les recherches se sont principalement concentrées sur les classificateurs légers, souvent associés à un ajout de détection d'objets comme SSDLite [20, 27]. Bien que les classificateurs à faible nombre de paramètres combinés avec SSDLite offrent une meilleure vitesse sur les microcontrôleurs, leur précision est inférieure à celle de YOLO.

Nos recherches ont identifié un écart crucial : il y a peu d'effort sur l'optimisation des architectures de détection d'objets qui équilibrent l'efficacité des paramètres et le coût computationnel tout en maintenant une précision au niveau des YOLO modernes. Cet écart oblige les développeurs à choisir entre des modèles YOLO haute performance mais coûteux en calcul et des alternatives à faible puissance comme SSDLite, qui sacrifient la précision pour la vitesse. Notre travail vise à combler cette partie manquante de recherche en introduisant des modèles de détection d'objets plus efficaces adaptés aux applications d'IA edge.

Cet article introduit deux contributions principales.

1. La première est une alternative à SSDLite appelée LeNeck qui comble l'écart entre les classificateurs à faible nombre de paramètres et les petits modèles YOLO. En utilisant LeNeck au lieu de SSDLite, nous maintenons une vitesse d'inférence similaire tout en obtenant une bien meilleure précision (Figure 1)).

2. La deuxième contribution est LeYOLO - une nouvelle famille de modèles YOLO légers et efficaces. LeYOLO correspond à la précision des échelles YOLO plus petites tout en améliorant considérablement la vitesse d'inférence sur les microcontrôleurs (Figure 2).

Nos résultats montrent que cette approche rivalise avec les modèles YOLO à des échelles comparables. Nous démontrons qu'il est possible d'optimiser l'architecture des réseaux de neurones pour la détection d'objets grâce à une nouvelle méthode d'échelle entre les classificateurs légers et les modèles YOLO.

2 Etat de l'art

Notre travail se concentre sur le développement d'une architecture optimale pour la détection d'objets en combinant deux approches clés : les détecteurs d'objets optimisés pour la vitesse et les classificateurs à faible coût conçus pour minimiser le nombre de paramètres en utilisant des techniques bien établies. LeYOLO et LeNeck intègrent des éléments connus pour leur efficacité dans la réduction de paramètres. Plus précisément, nous utilisons des Inverted Bottleneck, initialement introduits dans MobileNetV2 [27] et ensuite affinés par EfficientNet [30, 31] et GhostNet [9, 33]. Les convolutions pointwise [18] et depthwise sont des composants cruciaux dans l'optimisation de l'architecture, contribuant de manière significative à des modèles comme MNASNet [29].

L'essor des classificateurs à faible coût a conduit à SSDLite, une variante optimisée de SSD intégrant des convolutions groupées basées sur MobileNets. Initialement conçu pour réduire les coûts de détection en utilisant VGG [28], SSDLite partage des similitudes avec les premiers modèles YOLO [26]. Depuis lors, aucune méthode n'a significativement surpassé SSDLite, bien que SSDLiteX [16] ait tenté d'améliorer ses performances.

Du côté de YOLO, les recherches ont exploré la réduction des paramètres dans les architectures principales. Les efforts de tinier-yolo, efficient yolo, mobile densenet et autres [4, 41, 7, 40] ont intégré des éléments de classificateurs légers comme les convolutions depthwise et des techniques plus anciennes telles que les modules fire [13] pour minimiser l'utilisation des paramètres.

EfficentDet [40] partage la philosophie centrale de notre modèle : utiliser des couches à faible coût computationnel (concaténation et additions, convolutions depthwise et pointwise). Cependant, EfficientDet nécessite trop d'informations sémantiques et trop d'états de blocage (attente des couches précédentes, graphes complexes), ce qui le rend difficile à optimiser pour une vitesse d'exécution rapide.

Les auteurs de YOLOF [3] ont opté pour un modèle avec une seule entrée et une seule sortie dans le Neck [2]. Bien

2. Partie du modèle qui partage plusieurs niveaux d'informations sémantiques

que cette conception soit théoriquement plus rapide et plus efficace en calcul, l'article YOLOF révèle une baisse significative de la précision lors de la comparaison d'un Neck à sortie unique (Single-in, Single-out - SiSO) avec un Neck à sorties multiples (Single-in, Multiple-out - SiMO).

Plus récemment, YOLOX [5] et YOLOv9 [38] ont introduit des alternatives légères avec des paramètres réduits. YOLOX remplace les convolutions standard par des convolutions depthwise de tailles de noyau plus grandes et traite des entrées d'image plus petites. YOLOv9 contribue de manière substantielle à l'optimisation des paramètres mais se concentre sur l'échelle YOLO standard plutôt que sur les architectures adaptées aux mobiles.

Enfin, Tinyssimo YOLO [24], basé sur les premiers modèles YOLO [25], se concentre sur la réduction des coûts computationnels pour permettre la détection d'objets sur les microcontrôleurs fonctionnant dans la gamme de puissance des milliwatts. Cependant, il ne parvient pas à atteindre la précision et l'efficacité même des plus petites variantes de YOLO ou des classificateurs basés sur SSDLite.

3 Optimisation des détecteurs d'objets en temps réel pour les microcontrôleurs

Les détecteurs d'objets modernes reposent sur des blocs d'architecture qui exploitent pleinement le matériel moderne. Les convolutions standard et les structures parallèles ou multi-branches sont couramment utilisées. Cependant, ces conceptions sont trop gourmandes en ressources pour les microcontrôleurs. Conçu pour être hautement efficace, le bloc de construction principal de LeYOLO optimise à la fois les paramètres et le mAP (mesure de précision pour la détection d'objet). Il s'appuie sur une structure bien connue appelée Inverted Bottleneck, couramment utilisée dans les réseaux de neurones efficaces comme MobileNets [12, 27, 11] et EfficientNets [40, 31].

Au lieu d'utiliser de grands filtres coûteux pour traiter les images, LeYOLO décompose le processus en étapes plus petites et plus efficaces en utilisant trois couches de convolution principales. Notre bloc applique une convolution 1×1 qui projette les cartes de caractéristiques des canaux C de $x \in R^{B,C,H,W}$ en un tenseur de dimension d (où $d \geq C$). Ensuite, une convolution depthwise $k \times k$ traite efficacement les caractéristiques spatiales. Enfin, une autre convolution pointwise 1×1 ramène les canaux à leur taille d'origine. Bien que de nombreux articles utilisant des Inverted Bottleneck modifient la convolution pointwise finale pour produire un nombre de canaux différent de l'entrée, LeNeck et LeYOLO ne suivent pas cette approche. Au lieu de cela, nous nous appuyons uniquement sur des convolutions standard séparées lors de la transition entre les tailles de cartes de caractéristiques après le sous-échantillonnage. Ces convolutions ajustent intrinsèquement à la fois le nombre de canaux et la taille de la carte de caractéristiques, éliminant ainsi le besoin de transformations supplémentaires au sein du Inverted Bottleneck.

Astuce d'optimisation. Normalement, la première convolution 1×1 étend les canaux avant le traitement. Cependant, si le nombre de canaux n'a pas besoin de changer (si $C == d$), nous supprimons la première convolution pointwise. Ce petit changement réduit considérablement le nombre de calculs, en particulier dans les premières couches où les images sont grandes.

Impact sur la vitesse et la précision. L'élimination des calculs inutiles rend le réseau plus rapide et plus efficace tout en maintenant une haute précision (Section 3.3). Cette optimisation est particulièrement bénéfique pour l'exécution de modèles de détection d'objets sur des dispositifs à faible puissance et à ressources limitées. Pour comparaison, SSDLite ne commence à partager des informations sémantiques qu'au niveau P4 [3] tandis que les détecteurs d'objets classiques et modernes commencent au niveau P3, qui fournit des détails spatiaux plus riches mais à un coût computationnel plus élevé. En réduisant stratégiquement les calculs redondants dans les premières couches, LeNeck atteint la même vitesse que SSDLite tout en exploitant le niveau P3 plus informatif, résultant en une meilleure performance de détection sans surcoût computationnel. Le modèle utilise la fonction d'activation SiLU σ, comme dans les versions modernes de YOLO (YOLOv7, YOLOv9) pour une meilleure performance.

Nous définissons les dimensions d'entrée et de sortie comme C et la dimension étendue comme d. Pour les filtres $W_1 \in R^{1,1,C,d}$, $W_2 \in R^{k,k,1,d}$, et $W_3 \in R^{1,1,d,C}$, notre approche peut être représentée comme suit :

$$y = \begin{cases} W_3 \otimes \sigma[W_2 \otimes \sigma(W_1 \otimes x)] & \text{si } d \neq C \\ W_3 \otimes \sigma[W_2 \otimes \sigma(W_1 \otimes x)] & \text{si } d = C \text{ et } W_1 = \text{Vrai} \\ W_3 \otimes \sigma[W_2 \otimes (x)] & \text{si } d = C \text{ et } W_1 = \text{Faux} \end{cases} \quad (1)$$

3.1 LeNeck - Détecteur d'objets polyvalent

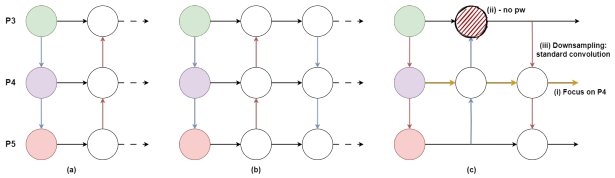

FIGURE 3 – Différence entre le Neck LeYOLO proposé et un agrégateur de caractéristiques sémantiques efficace. (a) Correspond à FPN [6]. (b) Représente PANnet [42]. Enfin, (c) est notre solution proposée.

Dans la détection d'objets, nous appelons le Neck la partie du modèle qui agrège plusieurs niveaux d'informations sémantiques, partageant les informations de couches distantes aux premières couches. Historiquement, les chercheurs ont utilisé un PANet [42] ou FPN [19] pour partager efficacement les cartes de caractéristiques, permettant plusieurs

3. P4 est le niveau sémantique de l'information correspondant à la taille de l'entrée divisée par 2^4.

niveaux de détection en reliant plusieurs informations sémantiques P_i au PANet et leurs sorties respectives comme illustré dans la Figure 3(a). Pour créer LeNeck, nous avons identifié un aspect très important dans la composition des réseaux de neurones profonds. Nous avons remarqué qu'il y a constamment une répétition significative des couches au niveau sémantique équivalent à P4. Nous avons trouvé cela dans tous les MobileNets [12, 27, 11], dans l'optimisation des Inverted Bottleneck dans EfficientNets [30, 31] et EfficientDet [40], ainsi que dans les architectures plus récentes avec des mécanismes de self-attention comme MobileViTs [22, 23, 35], EdgeNext [21], et FastViT [34], qui sont conçus pour la vitesse. Plus intéressant encore, les modèles conçus par Neural Architecture Search (NAS) [29, 11, 30] utilisent également ce schéma. Par conséquent, nous introduisons LeNeck, un agrégateur de caractéristiques sémantiques efficace qui utilise le niveau sémantique P4 comme principal conducteur pour fusionner les informations de P3 et P5 (Figure 3.(i)). Le calcul à P3 et P5 n'est effectué qu'une seule fois, garantissant l'efficacité (P3 utilise trop de taille spatiale, et P5 utilise un nombre très étendu de canaux).

Nous réduisons le calcul - en particulier au niveau P3 en raison de la grande taille spatiale - en supprimant la première convolution pointwise (Figure 3.(ii)). Après une étude comparative (Section 3.3) réalisée sur le backbone de LeYOLO à l'échelle nano, nous avons saisi l'opportunité de supprimer les convolutions pointwise coûteuses en temps puisque les canaux d'entrée de P3 concaténés avec les caractéristiques suréchantillonnées de P4 résultent en la dimension d requise par la convolution depthwise intermédiaire de notre Inverted Bottleneck optimisé présenté dans la section 3. Chaque nombre de canaux d'entrée, ainsi que le nombre de canaux étendus du Inverted Bottleneck, ne dépasse jamais 6. L'entrée de P3 est $32C$ tandis que la dernière couche cachée du Neck de LeYOLO étend les canaux d égale 192.

Comme les convolutions standard ne sont pas très efficace en nombre de paramètres et de calcul, nous nous limitons à l'utiliser deux fois. De P3 à P4, et de P4 à P5 pour effectuer un sous-echantillonage (Figure 3.iii)).

3.2 Backbone de LeYOLO

Notre mise en œuvre implique la minimisation de l'échange d'informations inter-couches sous la forme de $I(X;h_1) \geq I(X;h_2) \geq ... \geq I(X;h_n)$, avec n égal à la dernière couche cachée du backbone du réseau de neurones, en garantissant que le nombre de canaux d'entrée/sortie ne dépasse jamais une différence de ratio de 6 de la première couche cachée à la dernière. De plus, plutôt que d'augmenter la complexité computationnelle de notre modèle comme [37, 38, 10, 2], nous avons opté pour une mise à l'échelle plus efficace, intégrant la théorie du goulot d'étranglement inversé de Dangyoon Han et al. [8] qui stipulait que les convolutions pointwise ne devraient pas dépasser un ratio de 6 dans le Inverted Bottleneck.

TABLE 1 – Architecture du backbone de LeYOLO

Input	Operator	exp size	out size	NL	s
P0	conv2d, 3x3	-	16	SI	2
P1	conv2d, 1x1	16	16	SI	1
P1	bneck, 3x3, **pw=False**	16	16	SI	2
P2	bneck, 3x3	**96**	32	SI	2
P3	bneck, 3x3	96	32	SI	1
P3	bneck, 5x5	96	64	SI	2
P4	bneck, 5x5	192	64	SI	1
P4	bneck, 5x5	192	64	SI	1
P4	bneck, 5x5	192	64	SI	1
P4	bneck, 5x5	192	64	SI	1
P4	bneck, 5x5	**576**	96	SI	2
P5	bneck, 5x5	576	96	SI	1
P5	bneck, 5x5	576	96	SI	1
P5	bneck, 5x5	576	96	SI	1
P5	bneck, 5x5	576	96	SI	1

TABLE 2 – Amélioration de LeNeck et LeYOLO (best of).

Améliorations	mAP	GFLOP
base (LeYOLO nano)	34.3	2.64
+3x3	32.9	2.877
+5x5	34.9	3.946
+5x5 après P4	34.2	3.19
+Sous-echantillonage 3x3	34.6	3.011
+aucun pw backbone et neck	34.1	2.823
+Ratio d'expansion de 2 au lieu de 3 dans LeNeck	34.3	2.64

3.3 Etude comparative

Une étude comparative en apprentissage automatique est une méthode de recherche qui teste l'impact de couches, de caractéristiques ou de techniques spécifiques en les désactivant ou en les remplaçant. L'objectif est d'identifier quels paramètres sont cruciaux pour la performance du modèle, guidant le développement d'un détecteur d'objets entièrement optimisé. Nous utilisons LeYOLO dans son intégralité (backbone + LeNeck) pour l'étude comparative afin d'affiner les deux contributions (Tableau 2).

Nous avons d'abord exploré diverses configurations de taille de noyau. Bien que des noyaux plus grands améliorent généralement les performances, ils nécessitent également plus de ressources de calcul. Le choix optimal était une convolution 5×5 après le sous-échantillonnage P4.

En suivant les idées de ConvNeXt, nous avons utilisé des convolutions séparées pour le sous-échantillonnage. Cependant, l'utilisation d'un noyau 3×3 au lieu de 5×5 dans cette configuration a conduit à de meilleurs résultats.

Enfin, nous avons fait deux optimisations critiques : la réduction du ratio d'expansion dans le Inverted Bottleneck de 3 à 2 et l'élimination de la première convolution pointwise coûteuse dans les premières couches du backbone et au niveau P3 dans le Neck. Ces modifications ont considérablement réduit le coût computationnel du modèle tout en entraînant une perte de précision de -0,3 mAP, que nous avons jugée négligeable compte tenu des gains d'efficacité.

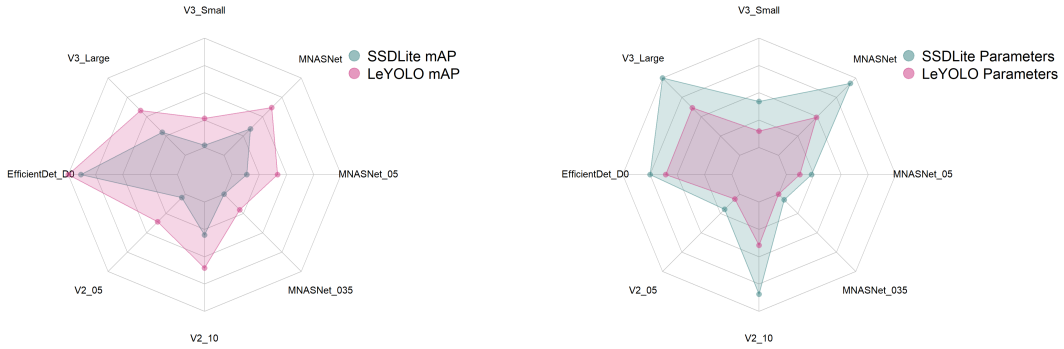

FIGURE 4 – LeYOLO comparé à SSDLite, avec un meilleur ratio paramètre-précision.

4 Résultats experimentaux

Nous entraînons chaque réseau de neurones avec les mêmes hyperparamètres et techniques d'augmentation de données, tels que SGD, avec un taux d'apprentissage de 0,01 et un momentum de 0,9. Nous nous appuyons principalement sur l'augmentation de données mosaïque ainsi que sur hsv de $\{0,015, 0,7, 0,4\}$ et une translation d'image de 0,1. En ce qui concerne les spécificités de l'entraînement, nous avons utilisé une taille de batch de 96 sur 4 GPU P100. La performance est évaluée sur l'ensemble de validation de MSCOCO en utilisant la précision moyenne (mAP). Pour LeYOLO, nous offrons une variété de modèles inspirés de la base architecturale présentée ci-dessus. Une approche classique implique la mise à l'échelle du nombre de canaux, de couches et de la taille d'entrée de l'image. Traditionnellement, la mise à l'échelle met l'accent sur les configurations de canaux et de couches, intégrant parfois divers schémas de mise à l'échelle (Tableau 3).

TABLE 3 – Différentes échelles pour l'entrainement de LeYOLO ainsi que leurs résultats.

Models	Nano	Small	Medium	Large
Input spatial size	640	640	640	768
Channels ratio	x1	x1.33	x1.33	x1.33
Layer ratio	x1	x1	x1.33	x1.33
mAP	34.3	38.2	39.3	41

LeYOLO évolue de la version Nano à la version Large avec une mise à l'échelle liée à ce qu'EfficientDet a apporté : répétition des canaux de 1.0 à 1.33, couches de 1.0 à 1.33, et taille spatiale pour les besoins d'entraînement de 640×640 à 768×768. Plusieurs tailles spatiales sont utilisées à des fins d'évaluation, allant de 320×320 à 768×768 Nous évaluons LeYOLO à des tailles spatiales réduites, tous les résultats étant présentés dans le Tableau 8.
Nous évaluons la vitesse du modèle sur deux microprocesseurs : STM32MP257FAI3 et STM32N6570-DK. Les deux utilisent des cœurs Arm Cortex, équilibrant faible consommation d'énergie et capacité de calcul efficace. Ces microcontrôleurs peuvent réaliser une inférence en temps réel à une résolution de 320×320 à 640×640 le calcul devient plus exigeant, mais LeYOLO traite toujours chaque inférence en moins d'une seconde, surpassant les modèles YOLO modernes (Tableau 4).

TABLE 4 – Vitesse d'inférence de LeYOLO (640×640) et sa précision sur appareil embarqué (Onnx - STM32MP257FAI3).

Models	mAP	Speed(ms)
LeYOLO Nano	**34.3**	**596**
LeYOLO Small	**38.2**	**877.9**
LeYOLO Medium	**39.3**	**1039**
YOLOv10 Nano [36]	38.5	1099
YOLOv8 Nano [14]	37.3	1235
YOLOv9 Tiny [38]	38.3	1371

4.0.1 Détection d'objets mobile

LeYOLO surpasse les détecteurs d'objets de type YOLO sur les dispositifs embarqués ou ceux avec une puissance de calcul limitée. Nous fournissons un tableau détaillé (Tableau 8) montrant le nombre de FLOPs, et nous observons une corrélation entre cette métrique et la vitesse d'exécution sur les dispositifs à faible ressource de calcul (Tableau 4).

TABLE 5 – Vitesse d'inférence de LeNeck (320x320) et sa précision sur appareil embarqué (Onnx - STM32MP257FAI3).

Models	SSDLite	**LeNeck**	SSDLite	**LeNeck**
	Speed(ms)		mAP.95	
V3-Small	**146.2**	165.9	16.0	**21.3**
V3-Large	**286.5**	292.3	22	**28.1**
V2-1.0	259.2	**256.3**	22.1	**28.6**
MNASNet 0.5	167.7	**155.3**	18.5	**24.6**
MNASNet	306.2	**262.3**	23	**28.9**
LeYOLO Nano		165.4		25.2

Nous intégrons l'état de l'art des backbones à faible nombre de paramètres avec LeNeck. Quel que soit le backbone utilisé, tous les nombres de canaux, P3,P4 et P5 spécificités de répétition restent les mêmes. À P3, la première convolution pointwise n'est jamais utilisée, comme dans le LeYOLO de base, ce qui entraîne le premier filtre étant la convolution

TABLE 6 – Vitesse d'inférence de LeNeck (320x320) et sa précision sur appareil embarqué (Onnx - STM32N6570-DK).

Models	SSDLite	**LeNeck**	SSDLite	**LeNeck**
	Speed(ms)		mAP.95	
V3-Small	**46.91**	50.19	16.0	**21.3**
V3-Large	**121.6**	129.2	22	**28.1**
V2-1.0	104	**103.5**	22.1	**28.6**
MNASNet 0.5	86.31	**36.03**	18.5	**24.6**
MNASNet	161.3	**53.78**	23	**28.9**

TABLE 7 – Performance de LeNeck comparée à d'autres modèles de l'état de l'art combinés avec SSDLite sur MSCOCO.

Models	SSDLite	**LeNeck**	SSDLite	**LeNeck**
	Parameters(M)		mAP.95	
V3-Small	2.49	**1.34**	16.0	**21.3**
V3-Large	4.97	**3.33**	22	**28.1**
EfficientDetD0	3.9	**3.29**	34.6	**37.1**
V2-0.5	1.54	**0.98**	16.6	**23.3**
V2-1.0	4.3	**2.39**	22.1	**28.6**
MNASNet 0.35	1.02	**0.7**	15.6	**20.0**
MNASNet 0.5	1.68	**1.22**	18.5	**24.6**
MNASNet	4.68	**2.8**	23	**28.9**

depthwise de la taille exacte du nombre de canaux d'entrée équivalent du backbone.

LeNeck, en tant que détecteur d'objets général pour les classificateurs légers, conserve le même nombre de canaux [4] et répétition des couches [5] que la version Nano de LeYOLO. À partir d'une variété de classificateurs légers avec un faible nombre de paramètres et de FLOP, LeNeck a surpassé SSDLite dans tous les aspects de ce que nous attendons d'un modèle à faible coût - meilleure échelle de paramètres, meilleure précision, et enfin, bonne vitesse d'inférence - avec les résultats de vitesse d'inférence décrits dans les tableaux 5 - 6 et l'efficacité paramètres-précision décrite dans le Tableau 7 et la Figure 4.

4.0.2 Microcontrôleurs bas de gamme

Au-delà du traitement en temps réel, nous avons également benchmarké LeYOLO par rapport aux modèles YOLO modernes sur divers microcontrôleurs bas de gamme, comme le montre le Tableau 9, en utilisant la variante LeYOLO-Small (YOLOv10 n'est pas compatible avec ces types de microcontrôleurs). Selon le Tableau 8, LeYOLO-Small correspond à YOLOv8 à YOLOv10 en précision. De plus, il s'avère plus efficace en inférence sur ces microcontrôleurs, étendant la capacité de YOLO à fonctionner efficacement sur des dispositifs bas de gamme.

4.1 Analyse plus approfondie

Pour analyser plus en détail les résultats de vitesse, nous mettons en avant l'objectif de notre Inverted Bottleneck avec une couche pointwise optionnelle (voir la Section 3 pour plus d'informations). Dans LeYOLO Small, la convolution pointwise à haute résolution spatiale dans le backbone et au niveau P3 dans LeNeck entraîne une amélioration minimale de la précision, comme le montre la Section 3.3. Par rapport à l'Inverted Bottleneck classique, notre solution économise 8.5% de la vitesse d'inférence sur tous les STM32 benchmarkés dans l'article (Tableau 10-pw). Contrairement aux architectures YOLO standard, qui reposent sur une répétition de couches profondes avec moins de canaux, LeYOLO obtient une meilleure efficacité en utilisant un ratio d'extension de 2 au lieu de 3 tout en maintenant une profondeur de répétition minimale de 3. Ce choix de conception améliore la vitesse d'inférence de 17% tout en préservant la précision, comme le confirme notre étude expérimentale (Tableau 9-exp x3).

Notre modèle est presque aussi rapide que SSDLite tout en obtenant une précision significativement meilleure. La légère différence de vitesse provient de la taille de la carte de caractéristiques - SSDLite commence à P4, tandis que nous commençons à P3. Cependant, LeNeck reste suffisamment léger pour rivaliser avec SSDLite en vitesse. Étant donné qu'il conserve des informations spatiales plus riches, LeNeck fonctionne également mieux pour détecter diverses tailles d'objets (Tableau 8).

5 Discussions

LeNeck : Compte tenu de l'efficacité coût-efficacité de LeNeck, il existe une opportunité significative pour l'expérimentation sur différents backbones de modèles de classification de pointe. LeYOLO émerge comme une alternative prometteuse à SSD et SSDLite. Les résultats prometteurs obtenus sur MSCoco avec notre solution suggèrent une applicabilité potentielle à d'autres modèles de classification.

Efficacité computationnelle : Nous avons mis en œuvre une nouvelle mise à l'échelle pour les modèles YOLO, prouvant qu'il est possible d'atteindre des niveaux de précision très élevés tout en utilisant très peu de ressources computationnelles (FLOP). LeYOLO fournit des résultats très rapides sur les dispositifs embarqués.

6 Conclusion

Tout au long de cet article, nous avons introduit plusieurs optimisations clés :

1. Amélioration des performances en aval du classificateur : Pour un budget de paramètres donné, LeNeck surpasse SSDLite en réduisant le nombre de paramètres tout en améliorant la précision sur MSCOCO. L'intégration de LeNeck avec les backbones existants à faible nombre de paramètres améliore la précision et l'efficacité à plusieurs échelles.
2. Une alternative viable aux modèles YOLO de petite taille : Le backbone optimisé de LeYOLO et LeNeck surpassent les variantes équivalentes des YOLO en détection d'objets. Les choix architecturaux derrière le backbone de LeYOLO entraînent une meilleure mise à l'échelle et un meilleur rapport précision-paramètres et FLOP.

4. de 32 couches à 96 avec une ratio d'expansion de 2
5. répétition $l = 3$

TABLE 8 – Etat de l'art des détecteur d'objets compatible avec les microcontrôleurs STM32.

Models	Input Size	mAP	mAP50	mAP75	S	M	L	FLOP(G)	Parameters (M)
MobileNetv3-S[12]	320	16.1	-	-	-	-	-	**0.32**	1.77
MobileNetv2-x0.5[27]	320	16.6	-	-	-	-	-	0.54	1.54
MnasNet-x0.5[29]	320	18.5	-	-	-	-	-	0.58	1.68
LeYOLO-Nano	320	**25.2**	**37.7**	**26.4**	**5.5**	**23.7**	**48.0**	0.66	**1.1**
MobileNetv3[11]	320	22	-	-	-	-	-	1.02	3.22
LeYOLO-Small	320	**29**	42.9	30.6	6.5	29.1	53.4	1.126	1.9
LeYOLO-Nano	480	**31.3**	**46**	**33.2**	**10.5**	**33.1**	**52.7**	1.47	1.1
MobileNetv2[27]	320	22.1	-	-	-	-	-	1.6	4.3
MnasNet[29]	320	23	-	-	-	-	-	1.68	4.8
LeYOLO-Small	480	**35.2**	50.5	37.5	13.3	38.1	55.7	**2.53**	**1.9**
MobileNetv1[12]	320	22.2	-	-	-	-	-	2.6	5.1
LeYOLO-Medium	480	**36.4**	**52.0**	**38.9**	**14.3**	**40.1**	**58.1**	3.27	2.4
LeYOLO-Small	640	**38.2**	**54.1**	**41.3**	**17.6**	**42.2**	**55.1**	**4.5**	**1.9**
YOLOv5-n[15]	640	28	45.7	-	-	-	-	4.5	1.9
EfficientDet-D0[32]	512	33.80	52.2	35.8	12	38.3	51.2	5	3.9
LeYOLO-Medium	640	**39.3**	**55.7**	**42.5**	**18.8**	**44.1**	**56.1**	**5.8**	**2.4**
YOLOv9-Tiny[38]	640	38.3	53.1	41.3	-	-	-	7.7	2
LeYOLO-Large	768	**41**	**57.9**	**44.3**	**21.9**	**46.1**	**56.8**	**8.4**	**2.4**

TABLE 9 – Vitesse d'inférence et précision de LeYOLO (640x640) sur des dispositifs embarqués.

Device	LeYOLO Small	YOLOv8	YOLOv9
	Speed (s)		
STM32H74l-DISCO	**12.3**	13.7	13.6
STM32F769l-DISCO	**19**	21.5	22.1
STM32F746G-DISCO	**20**	25	24.5
STM32F469I-DISCO	**54.6**	73.6	72.5

TABLE 10 – Amélioration de la vitesse d'inférence de LeYOLO Small (640x640)

Device	LeYOLO Small	pw	exp x3
	Speed (s)		
STM32H74l-DISCO	**12.37**	13.52	14.87
STM32F769l-DISCO	**19.04**	20.64	22.68
STM32F746G-DISCO	**20**	22.36	24.73
STM32F469I-DISCO	**54.6**	59.23	65.28

3. Vitesse d'inférence améliorée : LeYOLO et LeNeck obtiennent une meilleure vitesse d'inférence que les détecteurs d'objets à faible nombre de paramètres de pointe, grâce à leur architecture optimisée.

Nos contributions sont particulièrement efficaces sur les dispositifs mobiles, embarqués et à faible puissance, se rapprochant d'un équilibre idéal entre l'efficacité des paramètres et la performance de détection. La réduction de la taille du modèle tout en maintenant la précision permet la détection d'objets directement sur de petits dispositifs avec une surcharge computationnelle minimale. Ce raffinement étape par étape rapproche les modèles YOLO des applications pratiques de l'IA pour le edge.

Nous encourageons une expérimentation plus approfondie avec notre proposition, en explorant diverses variantes de jeux de données adaptées aux besoins spécifiques de l'industrie. Nous visons à fournir une gamme plus large de comparaisons pour LeYOLO dans des scénarios impliquant des dispositifs mobiles avec des ressources computationnelles très limitées.

Remerciements

Ce travail a été soutenu par Chips Joint Undertaking (Chips JU) dans le projet EdgeAI "Technologies Edge AI pour une performance embarquée optimisée" du projet, accord de subvention n° 101097300.

Références

[1] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging IT platforms : Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25(6) :599–616, June 2009.

[2] Yuxuan Cai, Yizhuang Zhou, Qi Han, Jianjian Sun, Xiangwen Kong, Jun Li, and Xiangyu Zhang. Reversible column networks. *arXiv preprint arXiv :2212.11696*, 2023.

[3] Qiang Chen, Yingming Wang, Tong Yang, Xiangyu Zhang, Jian Cheng, and Jian Sun. You only look one-level feature. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13039–13048, 2021.

[4] Wei Fang, Lin Wang, and Peiming Ren. Tinier-YOLO : A Real-Time Object Detection Method for Constrained Environments. *IEEE Access*, 8 :1935–1944, 2020.

[5] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox : Exceeding yolo series in 2021. *arXiv preprint arXiv :2107.08430*, August 2021.

[6] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn : Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7036–7045, 2019.

[7] Mohammad Hajizadeh, Mohammad Sabokrou, and Adel Rahmani. MobileDenseNet : A new approach to object detection on mobile devices. *Expert Systems with Applications*, 215 :119348, April 2023.

[8] Dongyoon Han, Sangdoo Yun, Byeongho Heo, and YoungJoon Yoo. Rethinking channel dimensions for efficient model design. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 732–741, 2021.

[9] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. Ghostnet : More features from cheap operations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1580–1589, 2020.

[10] Geoffrey Hinton. How to Represent Part-Whole Hierarchies in a Neural Network. *Neural Computation*, 35(3) :413–452, February 2023.

[11] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1314–1324, 2019.

[12] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets : Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv :1704.04861*, 2017.

[13] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet : Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv :1602.07360*, 2016.

[14] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics YOLO, January 2023.

[15] Glenn Jocher, Ayush Chaurasia, Alex Stoken, Jirka Borovec, NanoCode012, Yonghye Kwon, Kalen Michael, TaoXie, Jiacong Fang, imyhxy, Lorna, (Zeng Yifu), Colin Wong, Abhiram V, Diego Montes, Zhiqiang Wang, Cristi Fati, Jebastin Nadar, Laughing, UnglvKitDe, Victor Sonck, tkianai, yxNONG, Piotr Skalski, Adam Hogan, Dhruv Nair, Max Strobel, and Mrinal Jain. ultralytics/yolov5 : v7.0 - YOLOv5 SOTA Realtime Instance Segmentation, November 2022.

[16] Hyeong-Ju Kang. Ssdlitex : Enhancing ssdlite for small object detection. *Applied Sciences*, 13(21), 2023.

[17] Chuyi Li, Lulu Li, Hongliang Jiang, Kaiheng Weng, Yifei Geng, Liang Li, Zaidan Ke, Qingyuan Li, Meng Cheng, Weiqiang Nie, et al. Yolov6 : A single-stage object detection framework for industrial applications. *arXiv preprint arXiv :2209.02976*, September 2022.

[18] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv :1312.4400*, 2013.

[19] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[20] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd : Single shot multibox detector. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 21–37, Cham, 2016.

[21] Muhammad Maaz, Abdelrahman Shaker, Hisham Cholakkal, Salman Khan, Syed Waqas Zamir, Rao Muhammad Anwer, and Fahad Shahbaz Khan. EdgeNeXt : Efficiently Amalgamated CNN-Transformer Architecture for Mobile Vision Applications. In Leonid Karlinsky, Tomer Michaeli, and Ko Nishino, editors, *Computer Vision – ECCV 2022 Workshops*, Lecture Notes in Computer Science, pages 3–20, Cham, 2023.

[22] Sachin Mehta and Mohammad Rastegari. Mobilevit : Light-weight, general-purpose, and mobile-friendly vision transformer. In *International Conference on Learning Representations*, 2022.

[23] Sachin Mehta and Mohammad Rastegari. Separable Self-attention for Mobile Vision Transformers, June 2022.

[24] Julian Moosmann, Marco Giordano, Christian Vogt, and Michele Magno. Tinyissimoyolo : A quantized, low-memory footprint, tinyml object detection network for low power microcontrollers. In *2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pages 1–5, 2023.

[25] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once : Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, June 2016.

[26] Joseph Redmon and Ali Farhadi. Yolov3 : An incremental improvement. *arXiv preprint arXiv :1804.02767*, 2018.

[27] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2 : Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[28] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv :1409.1556*, 2014.

[29] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. MnasNet : Platform-Aware Neural Architecture Search for Mobile. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2815–2823, Long Beach, CA, USA, June 2019.

[30] Mingxing Tan and Quoc Le. EfficientNet : Rethinking Model Scaling for Convolutional Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 6105–6114. PMLR, May 2019.

[31] Mingxing Tan and Quoc Le. EfficientNetV2 : Smaller Models and Faster Training. In *Proceedings of the 38th International Conference on Machine Learning*, pages 10096–10106. PMLR, July 2021.

[32] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet : Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020.

[33] Yehui Tang, Kai Han, Jianyuan Guo, Chang Xu, Chao Xu, and Yunhe Wang. GhostNetV2 : Enhance Cheap Operation with Long-Range Attention. *Advances in Neural Information Processing Systems*, 35 :9969–9982, December 2022.

[34] Pavan Kumar Anasosalu Vasu, James Gabriel, Jeff Zhu, Oncel Tuzel, and Anurag Ranjan. Fastvit : A fast hybrid vision transformer using structural reparameterization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5785–5795, 2023.

[35] Shakti N Wadekar and Abhishek Chaurasia. Mobilevitv3 : Mobile-friendly vision transformer with simple and effective fusion of local, global and input features. *arXiv preprint arXiv :2209.15159*, October 2022.

[36] Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, and Guiguang Ding. Yolov10 : Real-time end-to-end object detection. *arXiv preprint arXiv :2405.14458*, 2024.

[37] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7 : Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7464–7475, June 2023.

[38] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. Yolov9 : Learning what you want to learn using programmable gradient information. *arXiv preprint arXiv :2402.13616*, 2024.

[39] Fangxin Wang, Miao Zhang, Xiangxiang Wang, Xiaoqiang Ma, and Jiangchuan Liu. Deep Learning for Edge Computing Applications : A State-of-the-Art Survey. *IEEE Access*, 8 :58322–58336, 2020.

[40] Zixuan Wang, Jiacheng Zhang, Zhicheng Zhao, and Fei Su. Efficient Yolo : A Lightweight Model For Embedded Deep Learning Object Detection. In *2020 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pages 1–6, July 2020.

[41] Wang Yang, Ding BO, and Li Su Tong. TS-YOLO :An efficient YOLO Network for Multi-scale Object Detection. In *2022 IEEE 6th Information Technology and Mechatronics Engineering Conference (ITOEC)*, volume 6, pages 656–660, March 2022.

[42] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.

[43] Zhi Zhou, Xu Chen, En Li, Liekang Zeng, Ke Luo, and Junshan Zhang. Edge Intelligence : Paving the Last Mile of Artificial Intelligence With Edge Computing. *Proceedings of the IEEE*, 107(8) :1738–1762, August 2019.