ÉTAT DE L'ART YOLO : DE YOLOV1 À YOLOV8

MARZOUG Ayoub
Data Engineering (DE)
INPT
Rabat
marzoug.ayoub@inemail.ine.inpt.ma

FEKRI Mohammed
Mathématiques, Informatique et Réseaux
INPT
Rabat
fekri@inpt.ac.ma

Mai 19, 2023

ABSTRACT

YOLO has become a central real-time object detection system for robotics, driverless cars, and video monitoring applications. We present a comprehensive analysis of YOLO's evolution, examining the innovations and contributions in each iteration from the original YOLO to YOLOv8. We start by describing the standard metrics and postprocessing ; then, we discuss the major changes in network architecture and training tricks for each model. Finally, we summarize the essential lessons from YOLO's development and provide a perspective on its future, highlighting potential research directions to enhance real-time object detection systems.

***K*eywords** YOLO · Object detection · Deep Learning · Computer Vision

1 Préambule

Ce document s'inscrit dans le sillage des projets exigibles au cours de la formation en ingénierie des données au sein de l'institut national des postes & télécommunications (INPT), et consiste en la réalisation d'un panorama synthétique et organisé des travaux et publications majeures existants en rapport avec la technologie de détection d'objets YOLO, ainsi qu'un aperçu sur son développement et ses cas d'usage. Il marque le dénouement du mini-projet bibliographie et initiation à la recherche encadré par Mr. FEKRI Mohammed à qui je témoigne une profonde déférence, et à l'honorable examen duquel je laisse cet ouvrage en souhaitant qu'il réponde à ses propres souhaits, ainsi qu'aux espérances et à l'attente de ses lecteurs. Il va de soi qu'il ne porte aucune responsabilité pour les quelques débordements auxquels j'ai pu me laisser aller ni pour les quelques fautes [1] que l'on ne manquera pas de trouver dans cette édition [2].

2 Introduction

La détection en temps réel d'objets s'est imposée comme un composant crucial dans de nombreuses applications, couvrant divers domaines tels que les véhicules autonomes, la robotique, la vidéo-surveillance et la réalité augmentée.

1. Dont le nombre suit une loi de Poisson.
2. Remerciements anticipés à tout lecteur qui m'aidera à réduire le paramètre de ladite loi.

Parmi les différents algorithmes de détection d'objets, le framework YOLO (You Only Look Once) s'est distingué par son remarquable équilibre entre vitesse et précision, permettant l'identification rapide et fiable des objets au sein des images. Depuis sa création, la famille YOLO a évolué au fil des itérations, chacune s'appuyant sur les versions précédentes pour pallier les limitations et améliorer les performances (voir Figure 1). Ce document vise à fournir une revue complète du développement du framework YOLO, depuis le YOLOv1 original jusqu'au dernier YOLOv8, en élucidant les innovations clés, les différences et les améliorations apportées à chaque version.

Le document commence par explorer les concepts fondamentaux et l'architecture du modèle YOLO original, qui a préparé le terrain pour les avancées ultérieures de la famille. Ensuite, on se penche sur les raffinements et les optimisations introduits dans chaque version, de YOLOv2 à YOLOv8. Ces améliorations comprennent divers aspects tels que la conception du réseau de neurones, les variations de la fonction de perte, les adaptations de la boîte d'ancrage et la mise à l'échelle de la résolution en entrée. En examinant ces développements, on cherche à offrir une compréhension systémique de l'évolution du framework YOLO et de ses implications sur la détection d'objets.

Outre l'examen des avancées spécifiques de chaque version de YOLO, le document met en évidence les compromis entre la vitesse et la précision qui ont émergé tout au long du développement du framework. D'où l'importance de tenir compte du contexte et des exigences d'applications spécifiques lors de la sélection du modèle YOLO le plus idoine. Enfin, on envisage les orientations futures de la technique YOLO, en évoquant les pistes potentielles de recherche et de développement qui façonneront les progrès en cours des systèmes de détection d'objets en temps réel.

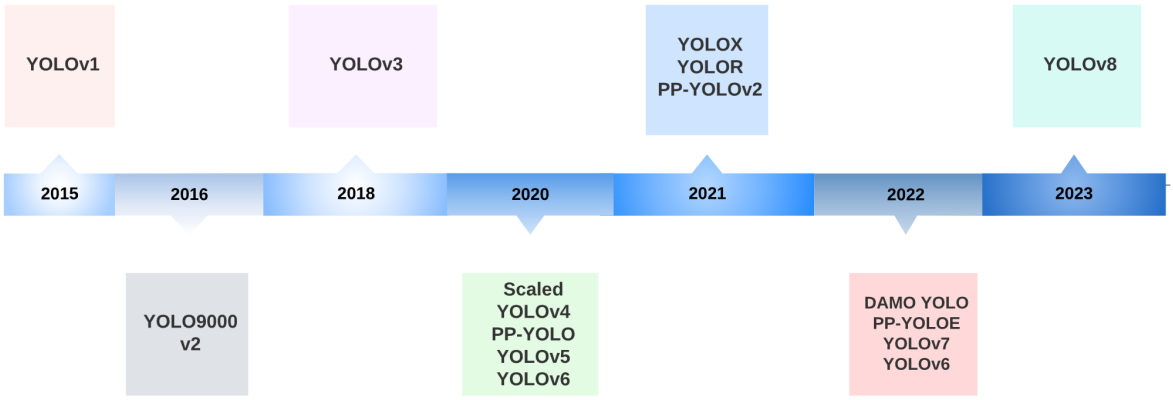

FIGURE 1 – Chronologie des versions YOLO

3 Applications de YOLO

Les capacités de détection d'objets en temps réel de YOLO se sont révélées inestimables dans les systèmes de véhicules autonomes, permettant l'identification rapide et le suivi de divers objets tels que les véhicules, les piétons [1, 2], les vélos et autres obstacles [3, 4, 5, 6]. Ces capacités ont été appliquées dans de nombreux domaines, notamment la reconnaissance d'actions [7] dans des séquences vidéo pour la surveillance [8], l'analyse sportive [9] et l'interaction homme-machine [10].

Les modèles YOLO ont été utilisés en agriculture pour détecter et classer les cultures [11, 12], les ravageurs et les maladies [13], contribuant aux techniques d'agriculture de précision et à l'automatisation des processus agricoles. Ils ont également été adaptés aux tâches de détection faciale dans les domaines de la biométrie, de la sécurité et des systèmes de reconnaissance faciale [14, 15].

Dans le domaine médical, YOLO a été utilisé pour la détection du cancer [16, 17], la segmentation de la peau [18] et l'identification de pilules [19], ce qui a permis d'améliorer la précision du diagnostic et l'efficacité des traitements. En télédétection, il a été utilisé pour la détection et la classification d'objets dans les images satellite et aériennes, contribuant à la cartographie de l'utilisation des terres, à la planification urbaine et à la surveillance de l'environnement [20, 21, 22, 23].

Les systèmes de sécurité ont intégré des modèles YOLO pour la surveillance en temps réel et l'analyse des flux vidéo, permettant la détection rapide d'activités suspectes [24], la distanciation sociale et la détection de masques faciaux [25]. Les modèles ont également été appliqués à l'inspection de surface pour détecter les défauts et les anomalies, améliorant le contrôle qualité dans les processus de fabrication et de production [26, 27, 28].

Dans les applications de trafic, les modèles YOLO ont été utilisés pour des tâches telles que la détection de plaques d'immatriculation [29] et la reconnaissance de panneaux de signalisation [30], contribuant au développement de systèmes de transport intelligents et de solutions de gestion du trafic. Ils ont été utilisés dans la détection et la surveillance de la faune pour identifier les espèces en voie de disparition pour la conservation de la biodiversité et la gestion des écosystèmes [31]. Enfin, YOLO a été largement utilisé dans les applications robotiques [32, 33] et la détection d'objets à partir de drones [34, 35].

4 Métriques de Détection d'Objets & Suppression non Maximale (NMS)

La Précision Moyenne (Average Precision, AP), traditionnellement appelée Précision Moyenne Globale (Mean Average Precision, mAP), est la métrique couramment utilisée pour évaluer les performances des modèles de détection d'objets. Elle mesure la précision moyenne pour l'ensemble des catégories, fournissant une valeur unique pour comparer différents modèles. Le jeu de données COCO ne fait pas de distinction entre AP et mAP. Dans le reste de ce document, on désignera cette métrique par AP.

Dans YOLOv1 et YOLOv2, le jeu de données utilisé pour l'entraînement et les tests était PASCAL VOC 2007 et VOC 2012 [36]. Néanmoins, à partir de YOLOv3, le jeu de données utilisé est Microsoft COCO (Common Objects in Context) [37]. La façon de calculer l'AP diffère pour ces jeux de données. Les sections suivantes expliqueront la logique derrière l'AP et comment elle est calculée.

4.1 Fonctionnement de l'AP

La métrique AP est basée sur les métriques de précision-rappel, gérant plusieurs catégories d'objets et définissant une prédiction positive à l'aide de l'Intersection sur Union (IoU).

Précision et Rappel : La précision mesure l'exactitude des prédictions positives du modèle, tandis que le rappel mesure la proportion de cas réellement positifs que le modèle identifie correctement. Il y a souvent un compromis entre la précision et le rappel ; par exemple, augmenter le nombre d'objets détectés (augmentation du rappel) peut entraîner plus de faux positifs (baisse de la précision). Pour prendre en compte ce compromis, la métrique AP intègre la courbe de précision-rappel qui trace la précision en fonction du rappel pour différents seuils de confiance. Cette métrique fournit une évaluation équilibrée de la précision et du rappel en considérant l'aire sous la courbe de précision-rappel.

Gestion de plusieurs catégories d'objets : Les modèles de détection d'objets doivent identifier et localiser plusieurs catégories d'objets dans une image. La métrique AP aborde cela en calculant la précision moyenne (AP) de chaque catégorie séparément, puis en prenant la moyenne de ces AP pour toutes les catégories (c'est pourquoi on l'appelle également précision moyenne globale). Cette approche garantit que les performances du modèle sont évaluées pour chaque catégorie individuellement, fournissant une évaluation plus complète des performances globales du modèle.

Intersection over Union : La détection d'objets vise à localiser avec précision les objets dans les images en prédisant des boîtes englobantes. La métrique AP intègre la mesure de l'Intersection over Union (IoU) pour évaluer la qualité des boîtes englobantes prédites. L'IoU est le rapport entre la superficie d'intersection et la superficie d'union de la boîte englobante prédite et de la boîte englobante réelle (voir Figure 2). Il mesure le chevauchement entre la boîte englobante réelle et la boîte englobante prédite. La référence COCO utilise plusieurs seuils d'IoU pour évaluer les performances du modèle à différents niveaux de précision de localisation.

4.2 Calcul de l'AP

L'AP est calculée différemment pour les jeux de données VOC et COCO. Dans cette section, on décrit comment elle est évaluée pour chaque dataset.

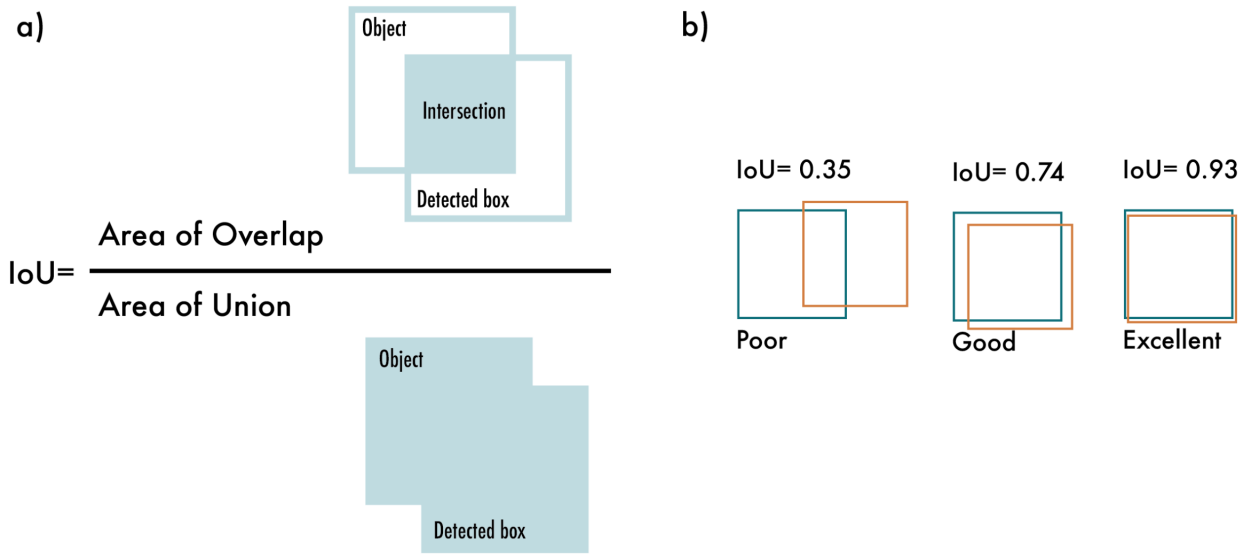

FIGURE 2 – Intersection over Union (IoU). a) L'IoU est calculé en divisant l'intersection des deux boîtes par l'union des boîtes ; b) exemples de trois valeurs différentes de l'IoU pour différents emplacements de boîtes.

Dataset VOC

Ce jeu de données comprend 20 catégories d'objets. Pour calculer l'AP pour VOC, on suit les étapes suivantes :

1. Pour chaque catégorie, calculer la courbe précision-rappel en faisant varier le seuil de confiance des prédictions du modèle.
2. Calculer la précision moyenne (AP) de chaque catégorie en utilisant un échantillonnage interpolé à 11 points de la courbe précision-rappel.
3. Calculer la précision moyenne globale (AP) finale en prenant la moyenne des APs pour les 20 catégories.

Dataset Microsoft COCO

Ce jeu de données comprend 80 catégories d'objets et utilise une méthode plus complexe pour calculer l'AP. Au lieu d'utiliser une interpolation à 11 points, il utilise une interpolation à 101 points, c'est-à-dire qu'il calcule la précision pour 101 seuils de rappel de 0 à 1 avec des incréments de 0,01. De plus, l'AP est obtenu en moyennant sur plusieurs valeurs d'IoU au lieu d'une seule, à l'exception d'une métrique d'AP courante appelée AP_{50}, qui est l'AP pour un seuil d'IoU unique de 0,5. Les étapes pour calculer l'AP dans COCO sont les suivantes :

1. Pour chaque catégorie, calculer la courbe précision-rappel en faisant varier le seuil de confiance des prédictions du modèle.
2. Calculer la précision moyenne (AP) de chaque catégorie en utilisant 101 seuils de rappel.
3. Calculer l'AP pour différents seuils d'Intersection sur Union (IoU), généralement de 0,5 à 0,95 avec un pas de 0,05. Un seuil d'IoU plus élevé nécessite une prédiction plus précise pour être considérée comme un vrai positif.
4. Pour chaque seuil d'IoU, prendre la moyenne des AP pour les 80 catégories.
5. Enfin, calculer l'AP global en faisant la moyenne des valeurs d'AP calculées à chaque seuil d'IoU.

Les différences dans le calcul de l'AP rendent difficile la comparaison directe des performances des modèles de détection d'objets entre les deux jeux de données. La norme actuelle utilise l'AP COCO en raison de son évaluation plus détaillée de la performance d'un modèle à différents seuils d'IoU.

4.3 Suppression non-maximale (NMS)

Non-Maximum Suppression (NMS) est une technique de post-traitement utilisée dans les algorithmes de détection d'objets pour réduire le nombre de boîtes englobantes (Bounding Boxes) superposées et améliorer la qualité globale de la détection. Les algorithmes de détection d'objets génèrent généralement plusieurs boîtes englobantes autour du même objet avec des scores de confiance différents. NMS élimine les boîtes englobantes redondantes et non pertinentes, ne conservant que les plus précises. L'algorithme 1 décrit la procédure en question. La figure 3 indique la sortie typique d'un modèle de détection d'objets contenant plusieurs boîtes englobantes superposées, ainsi que la sortie après l'application de NMS.

Algorithm 1 Algorithme de Suppression Non-Maximale

Entrées: Ensemble de boîtes englobantes prédites B, scores de confiance S, seuil de IoU τ, seuil de confiance T
Sortie: Ensemble de boîtes englobantes filtrées F

$F \leftarrow \emptyset$
$B \leftarrow \{b \in B \mid S(b) \geq T\}$
Trier les boîtes B par ordre décroissant de leurs scores de confiance
Tant que $B \neq \emptyset$ **Faire**
    Sélectionner la boîte b ayant le score de confiance le plus élevé
    Ajouter b à l'ensemble final des boîtes F : $F \leftarrow F \cup \{b\}$
    Retirer b de l'ensemble des boîtes B : $B \leftarrow B - \{b\}$
    Pour chaque boîtes restantes r dans B **Faire**
        Calculer l'indice de recouvrement IoU entre b et r : iou $\leftarrow \text{IoU}(b, r)$
        Si iou $\geq \tau$ **Alors**
            Retirer r de l'ensemble des boîtes B : $B \leftarrow B - \{r\}$

FIGURE 3 – Suppression des non-maximaux (NMS). a) Illustre la sortie typique d'un modèle de détection d'objets contenant plusieurs boîtes superposées. b) Illustre la sortie après l'application de la suppression des non-maximaux (NMS).

On va maintenant procéder à la description des différents modèles appartenant à la famille YOLO.

5 YOLO : You Only Look Once

L'article "YOLO" de Joseph Redmon et al., publié à CVPR 2016 [38], a présenté pour la première fois une approche en temps réel pour la détection d'objets de bout en bout. Le nom YOLO signifie "You Only Look Once", ce qui fait référence au fait qu'il était capable d'accomplir la tâche de détection en une seule passe du réseau, contrairement aux approches précédentes qui utilisaient soit des fenêtres glissantes suivies d'un classifieur nécessitant des centaines ou des milliers d'exécutions par image, soit des méthodes plus avancées qui divisaient la tâche en deux étapes, où la première étape détectait les régions possibles contenant des objets ou proposait des régions, et la deuxième étape exécutait un classifieur sur les propositions. De plus, YOLO utilisait une sortie plus simple basée uniquement sur la régression

pour prédire les sorties de détection, contrairement à Fast R-CNN [39] qui utilisait deux sorties distinctes, une pour la classification des probabilités et une pour la régression des coordonnées des boîtes.

5.1 Principe de YOLOv1

YOLOv1 fonctionne en unifiant les étapes de détection d'objets en détectant simultanément toutes les boîtes englobantes. Pour ce faire, YOLO divise l'image d'entrée en une grille $S \times S$ et prédit B boîtes englobantes de la même classe, avec leur confiance pour C classes différentes par élément de grille. Chaque prédiction de boîte englobante est composée de cinq valeurs : Pc, bx, by, bh, bw où Pc est le score de confiance de la boîte qui reflète à quel point le modèle est confiant que la boîte contient un objet et à quel point la boîte est précise. Les coordonnées bx et by sont les centres de la boîte par rapport à la cellule de grille, et bh et bw sont la hauteur et la largeur de la boîte par rapport à l'image complète. La sortie de YOLO est un tenseur de $S \times S \times (B \times 5 + C)$, éventuellement suivi de la suppression des non-maximaux (NMS) pour éliminer les détections en double.

Dans l'article original sur YOLO, les auteurs ont utilisé l'ensemble de données PASCAL VOC [36] qui contient 20 classes ($C = 20$) ; une grille de 7×7 ($S = 7$) et au plus 2 classes par élément de grille ($B = 2$), ce qui donne une prédiction de sortie de $7 \times 7 \times 30$.

La Figure 4 montre un vecteur de sortie simplifié en considérant une grille de trois par trois, trois classes et une classe unique par grille pour huit valeurs. Dans ce cas simplifié, la sortie de YOLO serait de $3 \times 3 \times 8$.

YOLOv1 a atteint une précision moyenne (AP) de 63,4 sur l'ensemble de données PASCAL VOC2007.

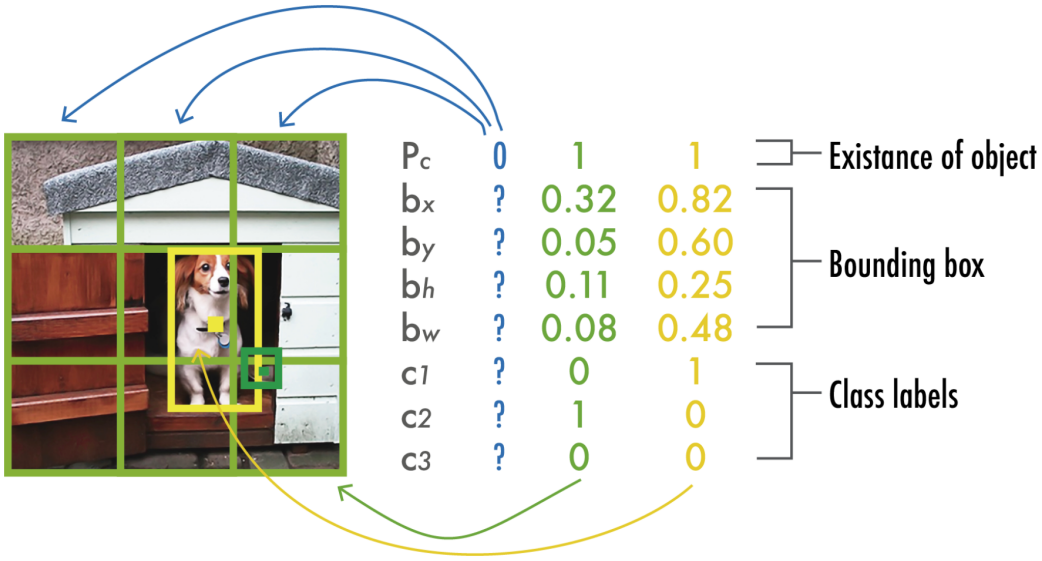

FIGURE 4 – Prédiction de sortie de YOLO. La figure représente de manière simplifiée un modèle YOLO avec une grille de trois par trois, trois classes et une prédiction de classe unique par élément de grille, ce qui produit un vecteur de huit valeurs.

5.2 Architecture de YOLOv1

L'architecture de YOLOv1 comprend 24 couches convolutionnelles suivies de deux couches entièrement connectées qui prédisent les coordonnées des boîtes englobantes et les probabilités. Toutes les couches utilisent des activations linéaires rectifiées avec fuite [40], à l'exception de la dernière qui utilise une fonction d'activation linéaire. Inspiré par GoogLeNet [41] et Network in Network [42], YOLO utilise des couches de convolution 1×1 pour réduire le nombre de cartes de caractéristiques et maintenir le nombre de paramètres relativement faible. Tableau 1 décrit l'architecture de YOLOv1 en tant que couches d'activation. Les auteurs ont également introduit un modèle plus léger appelé Fast YOLO, composé de neuf couches convolutionnelles.

TABLE 1 – Architecture de YOLO. Comprend 24 couches de convolution combinant des convolutions de 3×3 avec des convolutions de 1×1 pour la réduction des canaux. La sortie est une couche entièrement connectée qui génère une grille de 7×7 avec 30 valeurs pour chaque cellule de la grille afin de pouvoir accueillir dix coordonnées de boîtes englobantes (2 boîtes) avec 20 catégories.

	Type	Filtre	Taille/Stride	Sortie
	Conv	64	$7 \times 7/2$	224×224
	Max Pool		$2 \times 2/2$	112×112
	Conv	192	$3 \times 3/1$	112×112
	Max Pool		$2 \times 2/2$	56×56
$1 \times$	Conv	128	$1 \times 1/1$	56×56
	Conv	256	$3 \times 3/1$	56×56
	Conv	256	$1 \times 1/1$	56×56
	Conv	512	$3 \times 3/1$	56×56
	Max Pool		$2 \times 2/2$	28×28
$4 \times$	Conv	256	$1 \times 1/1$	28×28
	Conv	512	$3 \times 3/1$	28×28
	Conv	512	$1 \times 1/1$	28×28
	Conv	1024	$3 \times 3/1$	28×28
	Max Pool	$2 \times 2/2$	14×14	
$2 \times$	Conv	512	$1 \times 1/1$	14×14
	Conv	1024	$3 \times 3/1$	14×14
	Conv	1024	$3 \times 3/1$	14×14
	Conv	1024	$3 \times 3/2$	7×7
	Conv	1024	$3 \times 3/1$	7×7
	Conv	1024	$3 \times 3/1$	7×7
	FC		4096	4096
	Dropout 0.5			4096
	FC		$7 \times 7 \times 30$	$7 \times 7 \times 30$

5.3 Entraînement de YOLOv1

Les auteurs ont pré-entraîné les 20 premières couches de YOLO à une résolution de 224×224 en utilisant le jeu de données ImageNet [43]. Ensuite, ils ont ajouté les quatre dernières couches avec des poids initialisés aléatoirement et ont affiné le modèle avec les jeux de données PASCAL VOC 2007 et VOC 2012 [36] à une résolution de 448×448 pour augmenter les détails et améliorer la précision de la détection d'objets.

En ce qui concerne les augmentations, les auteurs ont utilisé des mises à l'échelle et des translations aléatoires d'au plus 20% de la taille de l'image d'entrée, ainsi que des variations aléatoires d'exposition et de saturation avec un facteur maximal de 1,5 dans l'espace colorimétrique HSV.

YOLOv1 utilisait une fonction de perte composée de plusieurs sommes des carrés des erreurs, comme illustré dans la Figure 5. Dans la fonction de perte, $\lambda_{coord} = 5$ est un facteur d'échelle qui donne plus d'importance aux prédictions des boîtes englobantes, et $\lambda_{noobj} = 0.5$ est un facteur d'échelle qui diminue l'importance des boîtes ne contenant pas d'objets.

Les deux premiers termes de la perte représentent la perte de localisation ; ils calculent l'erreur dans les prédictions des emplacements (x, y) et des tailles (w, h) des boîtes englobantes. Notez que ces erreurs ne sont calculées que dans les boîtes contenant des objets (représentées par le terme $\mathbb{1}_{ij}^{obj}$), pénalisant uniquement si un objet est présent dans cette cellule de la grille. Les troisième et quatrième termes de la perte représentent la perte de confiance ; le troisième terme mesure l'erreur de confiance lorsque l'objet est détecté dans la boîte ($\mathbb{1}_{ij}^{obj}$) et le quatrième terme mesure l'erreur de confiance lorsque l'objet n'est pas détecté dans la boîte ($\mathbb{1}_{ij}^{noobj}$). Étant donné que la plupart des boîtes sont vides, cette perte est pondérée par le terme λ_{noobj}. Le dernier composant de perte est la perte de classification qui mesure l'erreur quadratique des probabilités conditionnelles de classe pour chaque classe, uniquement si l'objet apparaît dans la cellule ($\mathbb{1}_{i}^{obj}$).

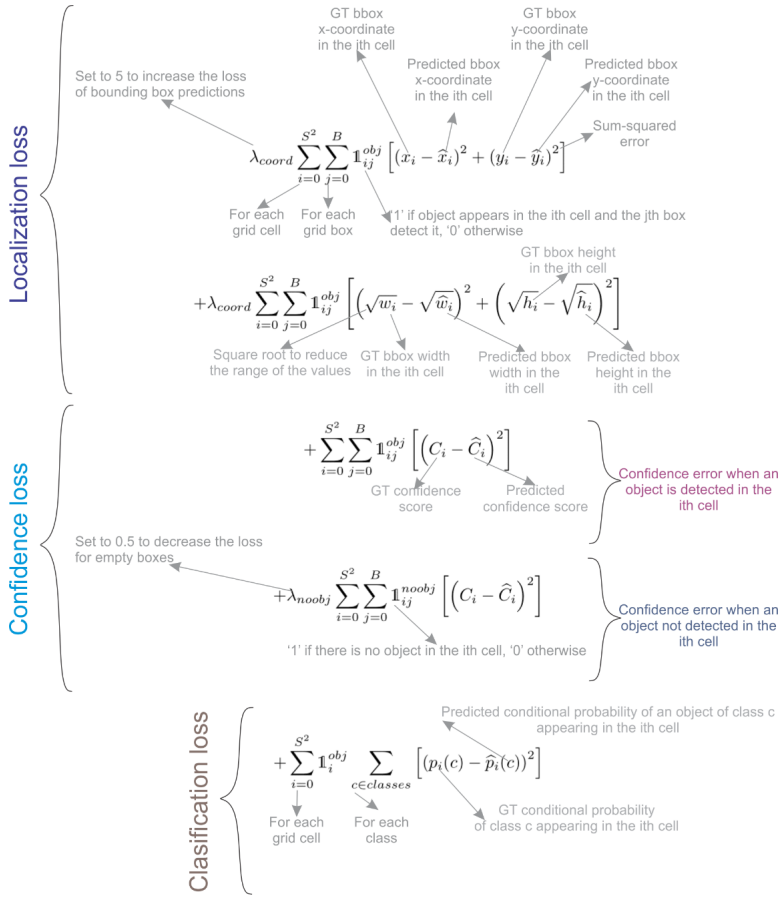

FIGURE 5 – Fonction de coût YOLO. Comprend une perte de localisation pour les coordonnées des boîtes englobantes, une perte de confiance pour la présence ou l'absence d'objet, et une perte de classification pour l'exactitude de la prédiction de la catégorie.

5.4 YOLOv1 : Force & Faiblesses

La structure simple de YOLO, ainsi que sa méthode novatrice de régression en une seule étape sur l'image complète, lui ont permis d'être beaucoup plus rapide que les détecteurs d'objets existants, offrant ainsi des performances en temps réel.

Cependant, bien que YOLO soit plus rapide que tout autre détecteur d'objets, l'erreur de localisation était plus importante par rapport aux méthodes de pointe telles que Fast R-CNN [39]. Trois principales causes expliquent cette limitation :

1. Il ne pouvait détecter au maximum que deux objets de la même classe dans une cellule de la grille, limitant ainsi sa capacité à prédire des objets proches.
2. Il avait du mal à prédire des objets avec des rapports d'aspect non observés dans les données d'entraînement.
3. Il apprenait à partir de caractéristiques d'objets grossières en raison des couches de sous-échantillonnage.

6 YOLOv2 : Meilleur, plus Rapide & plus Performant

YOLOv2 a été publié à CVPR 2017 [44] par Joseph Redmon et Ali Farhadi. Il présentait plusieurs améliorations par rapport au YOLO original, le rendant meilleur tout en conservant sa rapidité, et également plus performant - capable de détecter 9000 catégories ! Les améliorations étaient les suivantes :

1. **Normalisation par lots** (batch normalization) sur toutes les couches de convolution, améliorant la convergence et agissant comme un régulariseur pour réduire le surajustement (overfitting).
2. **Classifieur à haute résolution**. À l'instar de YOLOv1, le modèle a été pré-entraîné avec ImageNet à une résolution de 224×224. Cependant, cette fois-ci, il a été affiné pendant dix époques sur ImageNet avec une résolution de 448×448, améliorant ainsi les performances du réseau sur des entrées de résolution plus élevée.
3. **Entièrement convolutif**. Les couches denses ont été supprimées et une architecture entièrement convolutive a été utilisée.
4. **Utilisation de boîtes d'ancrage pour prédire les boîtes englobantes**. Ils ont utilisé un ensemble de boîtes prédéfinies ou boîtes d'ancrage, qui sont des boîtes avec des formes prédéfinies utilisées pour correspondre aux formes prototypiques des objets, comme illustré dans la Figure 6. Plusieurs boîtes d'ancrage sont définies pour chaque cellule de grille, et le système prédit les coordonnées et la classe pour chaque boîte d'ancrage. La taille de la sortie du réseau est proportionnelle au nombre de boîtes d'ancrage par cellule de grille.
5. **Clusters de dimensions**. Le choix de bonnes boîtes d'ancrage aide le réseau à apprendre à prédire des boîtes englobantes plus précises. Les auteurs ont appliqué une classification k-moyennes (k-means) sur les boîtes englobantes d'entraînement pour trouver de bonnes boîtes d'ancrage. Ils ont sélectionné cinq boîtes d'ancrage offrant un bon compromis entre le rappel et la complexité du modèle.
6. **Prédiction directe de la position**. Contrairement à d'autres méthodes qui prédisaient des décalages [45], YOLOv2 a suivi la même philosophie et a prédit les coordonnées de l'emplacement par rapport à la cellule de grille. Le réseau prédit cinq boîtes englobantes pour chaque cellule, chacune avec cinq valeurs $t_x, t_y, t_w, t_h et t_o$, où t_o est équivalent à Pc de YOLOv1, et les coordonnées finales des boîtes englobantes sont obtenues comme illustré dans la Figure 7.
7. **Caractéristiques plus fines**. YOLOv2, par rapport à YOLOv1, a supprimé une couche de pooling pour obtenir une carte de caractéristiques ou une grille de 13×13 pour des images d'entrée de 416×416. YOLOv2 utilise également une couche de passage qui prend la carte de caractéristiques de taille $26 \times 26 \times 512$ et la réorganise en empilant les caractéristiques adjacentes dans des canaux différents au lieu de les perdre via un sous-échantillonnage spatial. Cela génère des cartes de caractéristiques de taille $13 \times 13 \times 2048$ concaténées dans la dimension des canaux avec les cartes de taille inférieure $13 \times 13 \times 1024$ pour obtenir des cartes de caractéristiques de taille $13 \times 13 \times 3072$. Veuillez consulter le Tableau 2 pour les détails architecturaux.
8. **Entraînement multi-échelle**. Étant donné que YOLOv2 n'utilise pas de couches entièrement connectées, les entrées peuvent avoir différentes tailles. Pour rendre YOLOv2 robuste à différentes tailles d'entrée, les auteurs ont entraîné le modèle de manière aléatoire en modifiant la taille de l'entrée - de 320×320 jusqu'à 608×608 - tous les dix lots.

Avec toutes ces améliorations, YOLOv2 a obtenu une précision moyenne (AP) de 78,6% sur le jeu de données PASCAL VOC2007, comparé à 63,4% pour YOLOv1.

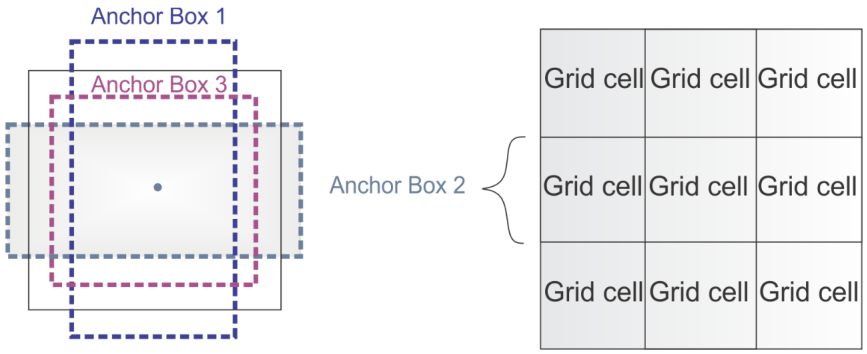

FIGURE 6 – Ancres. YOLOv2 définit plusieurs boîtes d'ancrage pour chaque cellule de la grille.

6.1 Architecture de YOLOv2

L'architecture de base utilisée par YOLOv2 est appelée Darknet-19, composée de 19 couches de convolution et de cinq couches de max-pooling. Tout comme l'architecture de YOLOv1, elle s'inspire du réseau Network in Network [42],

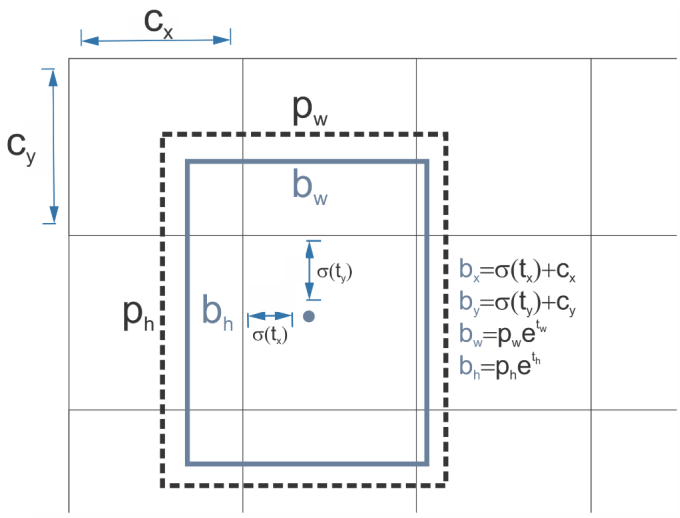

FIGURE 7 – Prédiction des boîtes englobantes. Les coordonnées du centre de la boîte sont obtenues à partir des valeurs prédites t_x, t_y qui passent par une fonction sigmoïde et sont décalées par la position de la cellule de la grille c_x, c_y. La largeur et la hauteur de la boîte finale utilisent la largeur préalable p_w et la hauteur préalable p_h, mises à l'échelle par e^{t_w} et e^{t_h} respectivement, où t_w et t_h sont prédits par YOLOv2.

utilisant des convolutions 1×1 entre les convolutions 3×3 pour réduire le nombre de paramètres. De plus, comme mentionné précédemment, ils utilisent la normalisation par lots (batch normalization) pour régulariser et faciliter la convergence. Le Tableau 2 présente l'ensemble de l'architecture Darknet-19 avec la tête de détection d'objets. YOLOv2 prédit cinq boîtes englobantes, chacune avec cinq valeurs et 20 classes lorsqu'il est utilisé avec l'ensemble de données PASCAL VOC.

La tête de classification d'objets remplace les quatre dernières couches de convolution par une seule couche de convolution avec 1000 filtres, suivie d'une couche de pooling global moyen et d'une couche Softmax.

6.2 YOLO9000 : Version Accrue de YOLOv2

Les auteurs ont introduit une méthode pour l'entraînement de la classification et de la détection conjointes dans le même article. Ils ont utilisé les données d'étiquetage de détection provenant de COCO [37] pour apprendre les coordonnées des boîtes englobantes et les données de classification provenant d'ImageNet afin d'augmenter le nombre de catégories qu'il peut détecter. Pendant l'entraînement, ils ont combiné les deux ensembles de données de telle sorte que lorsqu'une image d'entraînement de détection est utilisée, elle rétropropage le réseau de détection, et lorsqu'une image d'entraînement de classification est utilisée, elle rétropropage la partie de l'architecture liée à la classification. Le résultat est un modèle YOLO capable de détecter plus de 9000 catégories, d'où le nom YOLO9000.

7 YOLOv3

YOLOv3 [46] a été publié dans ArXiv en 2018 par Joseph Redmon et Ali Farhadi. Il a introduit des modifications significatives ainsi qu'une architecture plus vaste afin de rivaliser avec les dernières avancées de l'état de l'art tout en maintenant des performances en temps réel. Dans la suite, nous décrivons les changements par rapport à YOLOv2.

1. **Prédiction des boîtes englobantes** : Tout comme dans YOLOv2, le réseau prédit les quatre coordonnées (t_x, t_y, t_w et t_h) pour chaque boîte englobante. Cependant, dans YOLOv3, une amélioration majeure est apportée en introduisant un score d'objectivité pour chaque boîte englobante, obtenu à l'aide d'une régression logistique. Ce score est établi à 1 pour l'anchor box présentant le chevauchement le plus élevé avec la boîte réelle, tandis que les autres boîtes d'ancrage se voient attribuer la valeur 0. Contrairement à Faster R-CNN [45], YOLOv3 associe uniquement une seule anchor box à chaque objet de la vérité terrain. De plus, en cas

TABLE 2 – Architecture de YOLOv2. La structure principale Darknet-19 (couches 1 à 23) est associée à la tête de détection composée des quatre dernières couches de convolution et de la couche de passage qui réorganise les caractéristiques de la 17^e sortie de $26 \times 26 \times 512$ en une sortie de $13 \times 13 \times 2048$, suivie d'une concaténation avec la 25^e couche. La dernière convolution génère une grille de 13×13 avec 125 canaux pour accueillir 25 prédictions (5 coordonnées + 20 classes) pour cinq boîtes englobantes.

Nom	Type	Filtre	Taille/Stride	Sortie
1	Conv/BN	32	$3 \times 3/1$	$416 \times 416 \times 32$
2	Max Pool		$2 \times 2/2$	$208 \times 208 \times 32$
3	Conv/BN	64	$3 \times 3/1$	$208 \times 208 \times 64$
4	Max Pool		$2 \times 2/2$	$104 \times 104 \times 64$
5	Conv/BN	128	$3 \times 3/1$	$104 \times 104 \times 128$
6	Conv/BN	64	$1 \times 1/1$	$104 \times 104 \times 64$
7	Conv/BN	128	$3 \times 3/1$	$104 \times 104 \times 128$
8	Max Pool		$2 \times 2/2$	$52 \times 52 \times 128$
9	Conv/BN	256	$3 \times 3/1$	$52 \times 52 \times 256$
10	Conv/BN	128	$1 \times 1/1$	$52 \times 52 \times 128$
11	Conv/BN	256	$3 \times 3/1$	$52 \times 52 \times 256$
12	Max Pool		$2 \times 2/2$	$26 \times 26 \times 256$
13	Conv/BN	512	$3 \times 3/1$	$26 \times 26 \times 512$
14	Conv/BN	256	$1 \times 1/1$	$26 \times 26 \times 256$
15	Conv/BN	512	$3 \times 3/1$	$26 \times 26 \times 512$
16	Conv/BN	256	$1 \times 1/1$	$26 \times 26 \times 256$
17	Conv/BN	512	$3 \times 3/1$	$26 \times 26 \times 512$
18	Max Pool		$2 \times 2/2$	$13 \times 13 \times 512$
19	Conv/BN	1024	$3 \times 3/1$	$13 \times 13 \times 1024$
20	Conv/BN	512	$1 \times 1/1$	$13 \times 13 \times 512$
21	Conv/BN	1024	$3 \times 3/1$	$13 \times 13 \times 1024$
22	Conv/BN	512	$1 \times 1/1$	$13 \times 13 \times 512$
23	Conv/BN	1024	$3 \times 3/1$	$13 \times 13 \times 1024$
24	Conv/BN	1024	$3 \times 3/1$	$13 \times 13 \times 1024$
25	Conv/BN	1024	$3 \times 3/1$	$13 \times 13 \times 1024$
26	Reorg layer 17			$13 \times 13 \times 2048$
27	Concat 25 and 26			$13 \times 13 \times 3072$
28	Conv/BN	1024	$3 \times 3/1$	$13 \times 13 \times 1024$
29	Conv	125	$1 \times 1/1$	$13 \times 13 \times 125$

d'absence d'assignation d'une anchor box à un objet, cela n'entraîne qu'une perte de classification, sans perte de localisation ou de confiance.

2. **Prédiction des classes** : Au lieu d'utiliser un softmax pour la classification, YOLOv3 utilise une perte d'entropie croisée (cross-entropy) binaire pour entraîner des classificateurs logistiques indépendants. Ainsi, le problème est formulé comme une classification multi-étiquettes. Cette modification permet d'attribuer plusieurs étiquettes à la même boîte, ce qui peut être nécessaire dans certains jeux de données complexes [47] où des étiquettes se chevauchent. En guise d'exemple, un même objet peut être à la fois une personne et un homme.
3. **Nouvelle architecture de base** : YOLOv3 dispose d'un extracteur de caractéristiques plus grand composé de 53 couches de convolution avec des connexions résiduelles. La section 7.1 du papier décrit l'architecture plus en détail.
4. **Spatial Pyramid Pooling** (**SPP**) : Bien que cela ne soit pas mentionné dans le papier, les auteurs ont également ajouté à l'extracteur de caractéristiques un bloc SPP modifié [48] . Ce bloc concatène plusieurs sorties de max pooling sans sous-échantillonnage (pas $= 1$), chacune avec différentes tailles de noyaux $k \times k$, où $k = 1, 5, 9, 13$, ce qui permet d'obtenir un champ récepteur plus large. Cette version est appelée YOLOv3-spp et a été la version la plus performante, améliorant l'AP_{50} de 2,7%.
5. **Prédictions multi-échelles** : Similairement aux Feature Pyramid Networks [49], YOLOv3 prédit trois boîtes à trois échelles différentes. La section 7.2 du papier décrit en détail le mécanisme de prédiction multi-échelles.
6. **Antécédents des boîtes englobantes** : Comme dans YOLOv2, les auteurs utilisent également la méthode des k-moyennes pour déterminer les antécédents des boîtes d'ancrage. La différence est que dans YOLOv2, ils

utilisaient un total de cinq boîtes par cellule, tandis que dans YOLOv3, ils utilisent trois boxes pour trois échelles différentes.

7.1 Architecture de YOLOv3

L'architecture de base présentée dans YOLOv3 est appelée Darknet-53. Elle a remplacé toutes les couches de max-pooling par des convolutions avec pas et a ajouté des connexions résiduelles. Au total, elle comprend 53 couches de convolution. La Figure 8 illustre les détails de l'architecture.

Le noyau Darknet-53 parvient à des précisions Top-1 et Top-5 comparables à celles du modèle ResNet-152, mais avec une vitesse presque deux fois supérieure.

Layer	Filters	size	Repeat		Output size
Image					416 × 416
Conv	32	3 × 3/1	1		416 × 416
Conv	64	3 × 3/2	1		208 × 208
Conv	32	1 × 1/1	Conv		208 × 208
Conv	64	3 × 3/1	Conv	×1	208 × 208
Residual			Residual		208 × 208
Conv	128	3 × 3/2	1		104 × 104
Conv	64	1 × 1/1	Conv		104 × 104
Conv	128	3 × 3/1	Conv	×2	104 × 104
Residual			Residual		104 × 104
Conv	256	3 × 3/2	1		52 × 52
Conv	128	1 × 1/1	Conv		52 × 52
Conv	256	3 × 3/1	Conv	×8	52 × 52
Residual			Residual		52 × 52
Conv	512	3 × 3/2	1		26 × 26
Conv	256	1 × 1/1	Conv		26 × 26
Conv	512	3 × 3/1	Conv	×8	26 × 26
Residual			Residual		26 × 26
Conv	1024	3 × 3/2	1		13 × 13
Conv	512	1 × 1/1	Conv		13 × 13
Conv	1024	3 × 3/1	Conv	×4	13 × 13
Residual			Residual		13 × 13

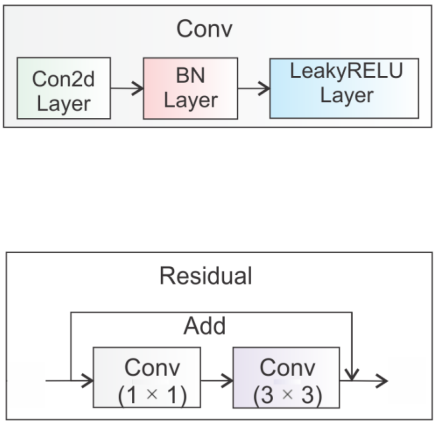

FIGURE 8 – Noyau central de YOLOv3. Constitué de 53 couches de convolution, chacune étant dotée de la normalisation par lots (batch normalization) et de l'activation Leaky ReLU. De plus, des connexions résiduelles relient l'entrée des convolutions 1×1 dans l'ensemble du réseau à la sortie des convolutions 3×3. L'architecture présentée ici ne comprend que le noyau central, elle n'inclut pas la tête de détection composée de prédictions multi-échelles.

7.2 YOLOv3 : Prédictions Multi-Échelles

Outre une architecture plus grande, une caractéristique essentielle de YOLOv3 est la prédiction multi-échelle, c'est-à-dire des prédictions à plusieurs tailles de grille. Cela a permis d'obtenir des boîtes avec plus de détails et a considérablement amélioré la prédiction des petits objets, qui était l'une des principales faiblesses des versions précédentes de YOLO.

L'architecture de détection multi-échelle illustrée dans la figure 9 fonctionne comme suit : la première sortie marquée comme **y1** est équivalente à la sortie de YOLOv2, où une grille de 13×13 définit la sortie. La deuxième sortie **y2** est composée en concaténant la sortie après $(Res \times 4)$ de Darknet-53 avec la sortie après $(Res \times 8)$. Les cartes des caractéristiques ont des tailles différentes, c'est-à-dire 13×13 et 26×26, il y a donc une opération de sur-échantillonnage

avant la concaténation. Enfin, en utilisant une opération de sur-échantillonnage, la troisième sortie **y3** concatène les cartes des caractéristiques de taille 26×26 avec les cartes des caractéristiques de taille 52×52.

Pour le jeu de données COCO avec 80 catégories, chaque échelle fournit un tenseur de sortie avec une forme de $N \times N \times [3 \times (4 + 1 + 80)]$, où $N \times N$ est la taille de la carte des caractéristiques (ou de la cellule de grille), le 3 indique les boîtes par cellule et le $4 + 1$ inclut les quatre coordonnées et le score de présence de l'objet.

Layer	Filters	size	Repeat		Output size
Image					416×416
Conv	32	$3 \times 3/1$	1		416×416
Conv	64	$3 \times 3/2$	1		208×208
Conv	32	$1 \times 1/1$	Conv		208×208
Conv	64	$3 \times 3/1$	Conv	$\times 1$	208×208
Residual			Residual		208×208
Conv	128	$3 \times 3/2$	1		104×104
Conv	64	$1 \times 1/1$	Conv		104×104
Conv	128	$3 \times 3/1$	Conv	$\times 2$	104×104
Residual			Residual		104×104
Conv	256	$3 \times 3/2$	1		52×52
Conv	128	$1 \times 1/1$	Conv		52×52
Conv	256	$3 \times 3/1$	Conv	$\times 8$	52×52
Residual			Residual		52×52
Conv	512	$3 \times 3/2$	1		26×26
Conv	256	$1 \times 1/1$	Conv		26×26
Conv	512	$3 \times 3/1$	Conv	$\times 8$	26×26
Residual			Residual		26×26
Conv	1024	$3 \times 3/2$	1		13×13
Conv	512	$1 \times 1/1$	Conv		13×13
Conv	1024	$3 \times 3/1$	Conv	$\times 4$	13×13
Residual			Residual		13×13

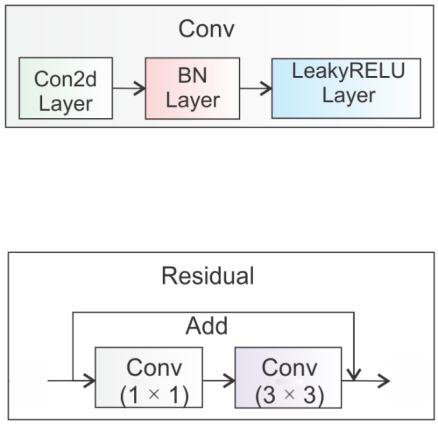

FIGURE 9 – Architecture de détection multi-échelle de YOLOv3. La sortie de l'architecture Darknet-53 en est divisée en trois différentes marquées comme **y1**, **y2** et **y3**, chacune ayant une résolution augmentée. Les boîtes prédites finales sont filtrées à l'aide de la suppression non maximale. Les blocs CBL (Convolution-BatchNorm-Leaky ReLU) comprennent une couche de convolution avec normalisation par lots et activation Leaky ReLU. Les blocs Res comprennent un bloc CBL suivi de deux structures CBL avec une connexion résiduelle, comme illustré dans la figure 8.

7.3 Performance de YOLOv3

Lorsque YOLOv3 a été publié, le référentiel pour la détection d'objets est passé de PASCAL VOC à Microsoft COCO [37]. Par conséquent, à partir de ce moment-là, tous les modèles YOLO sont évalués sur l'ensemble de données MS COCO. YOLOv3-spp a obtenu une précision moyenne (AP) de 36,2% et une précision moyenne à 50% d'intersection sur union (AP_{50}) de 60,6% à une vitesse de 20 images par seconde (FPS), ce qui le positionne à la pointe de la technologie de l'époque, tout en étant deux fois plus rapide.

8 Architecture : Épine dorsale, Cou & Tête

À cette époque, l'architecture des détecteurs d'objets a commencé à être décrite en trois parties : l'épine dorsale, le cou et la tête. La figure 10 présente un diagramme général de l'épine dorsale, du cou et de la tête.

L'épine dorsale est responsable de l'extraction de caractéristiques utiles à partir de l'image d'entrée. Il s'agit géné-

ralement d'un réseau de neurones convolutionnel (CNN) entraîné sur une tâche de classification d'images à grande échelle, telle qu'ImageNet. L'épine dorsale capture des caractéristiques hiérarchiques à différentes échelles, avec des caractéristiques de bas niveau (par exemple, les contours et les textures) extraites dans les couches plus basses, et des caractéristiques de haut niveau (*e.g.* les parties des objets et les informations sémantiques) extraites dans les couches plus profondes.

Le cou est un composant intermédiaire qui relie l'épine dorsale à la tête. Il agrège et affine les caractéristiques extraites par l'épine dorsale, en se concentrant souvent sur l'amélioration de l'information spatiale et sémantique à différentes échelles. Le cou peut comprendre des couches de convolution supplémentaires, des réseaux de pyramides de caractéristiques (FPN) [49] ou d'autres mécanismes visant à améliorer la représentation des caractéristiques.

La tête est le composant final d'un détecteur d'objets ; elle est responsable de la génération de prédictions à partir des caractéristiques fournies par l'épine dorsale et le cou. Elle se compose généralement d'un ou plusieurs sous-réseaux spécifiques à la tâche qui effectuent la classification, la localisation et, plus récemment, la segmentation d'instance et l'estimation de pose. La tête traite les caractéristiques fournies par le cou, générant des prédictions pour chaque candidat d'objet. En fin de compte, une étape de post-traitement, telle que la suppression des non-maximaux (NMS), filtre les prédictions qui se chevauchent et ne conserve que les détections les plus confiantes.

Dans le reste des modèles YOLO, on analyse les architectures en évoquant l'épine dorsale, le cou et la tête.

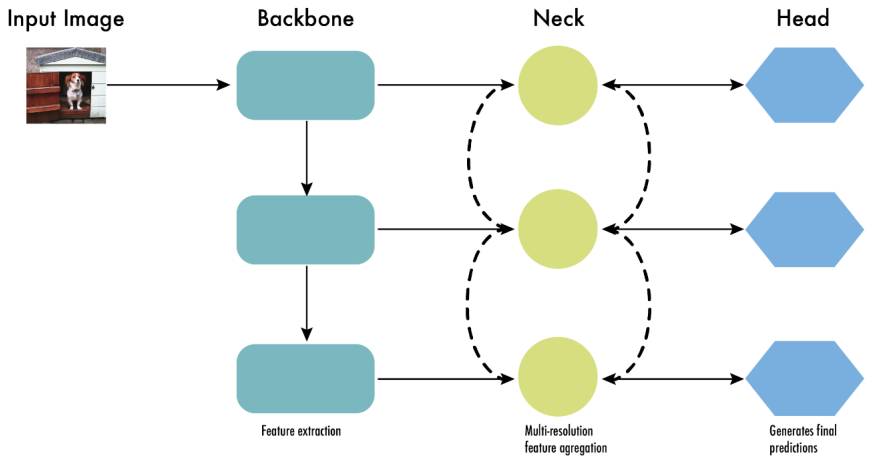

FIGURE 10 – L'architecture des détecteurs d'objets modernes peut être décrite en termes d'épine dorsale, de cou et de tête.

9 YOLOv4

Deux années se sont écoulées sans qu'une nouvelle version de YOLO ne soit publiée. Ce n'est qu'en avril 2020 qu'Alexey Bochkovskiy, Chien-Yao Wang et Hong-Yuan Mark Liao ont publié un article sur YOLOv4 dans ArXiv [50]. Au début, il semblait étrange que de nouveaux auteurs présentent une nouvelle version "officielle" de YOLO ; cependant, YOLOv4 a conservé la même philosophie que YOLO, c'est-à-dire en temps réel, open source, à prise unique et utilisant le framework Darknet. Les améliorations apportées étaient si satisfaisantes que la communauté a rapidement adopté cette version en tant que YOLOv4 officiel.

YOLOv4 a tenté de trouver un équilibre optimal en expérimentant de nombreux changements regroupés en deux catégories : les "bag-of-freebies" et les "bag-of-specials". Les "bag-of-freebies" sont des méthodes qui modifient uniquement la stratégie d'entraînement et augmentent le coût de l'entraînement, mais n'augmentent pas le temps d'inférence, le plus courant étant l'augmentation des données. En revanche, les "bag-of-specials" sont des méthodes qui augmentent légèrement le coût de l'inférence mais améliorent considérablement la précision. Parmi ces méthodes, on peut citer l'élargissement du champ récepteur [48, 51, 52], la combinaison de caractéristiques [53, 49, 54, 55] et le post-traitement [56, 40, 57, 58], entre autres.

Nous résumons les principaux changements apportés à YOLOv4 dans les points suivants :

1. **Architecture améliorée avec l'intégration de "bag-of-specials" (BoS)** : Les auteurs ont expérimenté plusieurs architectures pour le backbone, telles que ResNeXt50 [59], EfficientNet-B3 [60] et Darknet-53. L'architecture qui a donné les meilleurs résultats était une modification de Darknet-53 avec des connexions partielles entre les différentes étapes (CSPNet) [61] et une fonction d'activation Mish [57] en tant que cou (Voir figure 11). Pour le cou, ils ont utilisé une version modifiée du spatial pyramid pooling (SPP) [48] de YOLOv3-spp et des prédictions multi-échelles comme dans YOLOv3, mais avec une version modifiée du path aggregation network (PANet) [62] et un module d'attention spatiale (SAM) [63]. Pour la détection, ils ont utilisé des ancres comme dans YOLOv3. Par conséquent, le modèle a été appelé *CSPDarknet53-PANet-SPP*. Les connexions partielles entre les différentes étapes (CSP) ajoutées à Darknet-53 permettent de réduire le calcul du modèle tout en maintenant la même précision. Le bloc SPP, comme dans YOLOv3-spp, augmente le champ récepteur sans affecter la vitesse d'inférence. La version modifiée de PANet concatène les caractéristiques au lieu de les additionner, comme dans l'article original sur PANet.

2. **Intégration de "bag-of-freebies" (BoF) pour une approche d'entraînement avancée** : En plus des augmentations régulières telles que la luminosité aléatoire, le contraste, la mise à l'échelle, le recadrage, le retournement et la rotation, les auteurs ont mis en œuvre une augmentation par mosaïque qui combine quatre images en une seule, ce qui permet de détecter des objets en dehors de leur contexte habituel et réduit également la nécessité d'une grande taille de mini-lot pour la normalisation par lots. Pour la régularisation, ils ont utilisé DropBlock [64] comme alternative à Dropout [65] pour les réseaux de neurones convolutifs, ainsi que le lissage des étiquettes de classe [66, 67]. Pour le détecteur, ils ont ajouté la perte CIoU [68] et la normalisation en mini-lot croisée (CmBN) pour collecter des statistiques sur l'ensemble du lot plutôt que sur des mini-lots individuels, comme dans la normalisation par lots classique [69].

3. **Entraînement auto-contradictoire (SAT)** : Pour rendre le modèle plus robuste aux perturbations, une attaque adversaire est effectuée sur l'image d'entrée afin de créer une tromperie selon laquelle l'objet réel n'est pas présent dans l'image, tout en conservant l'étiquette d'origine pour détecter l'objet correct.

4. **Optimisation des hyperparamètres avec des algorithmes génétiques** : Pour trouver les hyperparamètres optimaux utilisés pour l'entraînement, des algorithmes génétiques ont été utilisés sur les 10% premières périodes, ainsi qu'un planificateur d'apprentissage à anneaux cosinus [70] pour ajuster le taux d'apprentissage pendant l'entraînement. Celui-ci commence par une réduction lente du taux d'apprentissage, suivie d'une réduction plus rapide à mi-parcours, et se termine par une légère réduction.

Le Tableau 3 répertorie la sélection finale des BoFs et BoS pour le backbone et le détecteur.

9.1 Performance de YOLOv4

Évalué sur le jeu de données MS COCO test-dev 2017, YOLOv4 a atteint un AP de 43,5% et un AP_{50} de 65,7% à plus de 50 images par seconde sur une carte graphique NVIDIA V100.

10 YOLOv5

YOLOv5 [72] a été publié quelques mois après YOLOv4 en 2020 par Glenn Jocher. Au moment de la rédaction de cet article, il n'y avait pas de publication scientifique sur YOLOv5, mais d'après le code, on sait qu'il utilise de nombreuses améliorations décrites dans la section YOLOv4, avec la principale différence étant qu'il a été développé en utilisant PyTorch plutôt que Darknet. YOLOv5 est un logiciel open source maintenu activement par Ultralytics, avec plus de 250 contributeurs et de nouvelles améliorations régulières. YOLOv5 est facile à utiliser, à entraîner et à déployer. Ultralytics propose une version mobile pour iOS et Android ainsi que de nombreuses intégrations pour l'étiquetage, l'entraînement et le déploiement.

YOLOv5 offre cinq versions d'échelle : YOLOv5n (nano), YOLOv5s (small), YOLOv5m (medium), YOLOv5l (large) et YOLOv5x (extra large).

La version de YOLOv5 publiée au moment de la rédaction de cet article est la version 7.0, comprenant des versions de YOLOv5 capables de classification et de segmentation d'instance.

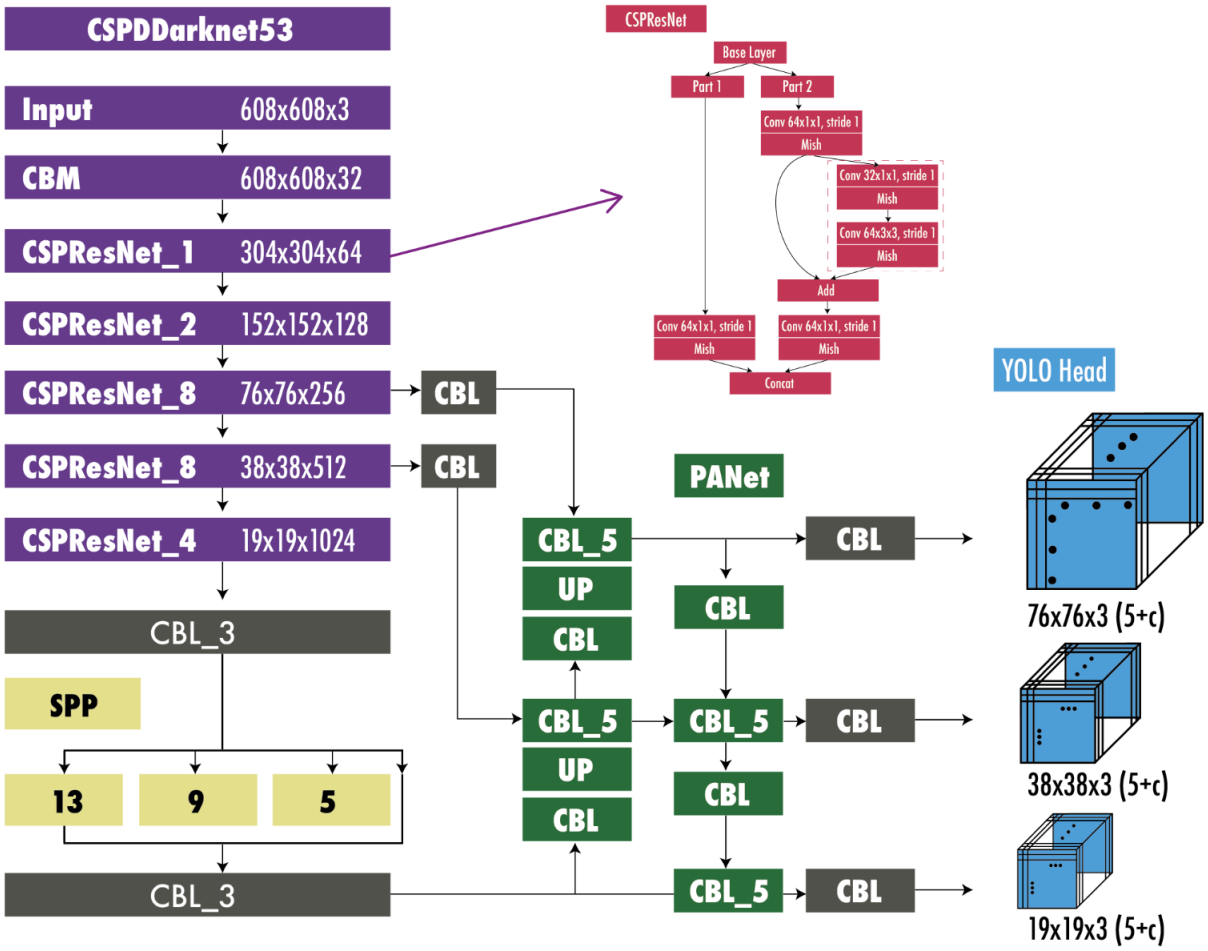

FIGURE 11 – Architecture de YOLOv4 pour la détection d'objets. Les modules présents dans le diagramme sont les suivants : **CMB** (Convolution + Normalisation par lots + Activation Mish), **CBL** (Convolution + Normalisation par lots + Leaky ReLU), **UP** (upsampling), **SPP** (Spatial Pyramid Pooling) et **PANet** (Path Aggregation Network). Le diagramme est inspiré de [71].

10.1 Performance de YOLOv5

Évalué sur le jeu de données MS COCO test-dev 2017, YOLOv5x a obtenu un AP de 50,7% avec une taille d'image de 640 pixels. En utilisant une taille de lot (batch size) de 32, il peut atteindre une vitesse de 200 FPS sur une carte graphique NVIDIA V100. En utilisant une taille d'entrée plus grande de 1536 pixels, YOLOv5 atteint un AP de 55,8%.

11 Scaled-YOLOv4

Un an après YOLOv4, les mêmes auteurs ont présenté Scaled-YOLOv4 [73] lors de la conférence CVPR 2021. Contrairement à YOLOv4, Scaled YOLOv4 a été développé avec Pytorch au lieu de Darknet. La principale nouveauté était l'introduction de techniques de mise à l'échelle vers le haut et vers le bas. La mise à l'échelle vers le haut signifie produire un modèle qui augmente la précision au détriment de la vitesse ; en revanche, la mise à l'échelle vers le bas consiste à produire un modèle qui augmente la vitesse en sacrifiant la précision. De plus, les modèles réduits nécessitent moins de puissance de calcul et peuvent fonctionner sur des systèmes embarqués.

L'architecture réduite a été appelée YOLOv4-tiny ; elle a été conçue pour les GPU d'entrée de gamme et peut fonctionner à 46 images par seconde sur un Jetson TX2 ou à 440 images par seconde sur un RTX2080Ti, atteignant un AP de 22%

TABLE 3 – Sélection finale de bag-of-freebies (BoF) et de bag-of-specials (BoS) dans YOLOv4. Les BoFs sont des méthodes qui améliorent les performances sans coût d'inférence supplémentaire, mais qui nécessitent des temps d'entraînement plus longs. En revanche, les BoS sont des méthodes qui augmentent légèrement le coût d'inférence, mais qui améliorent considérablement la précision.

Backbone	**Détecteur**
Bag-of-Freebies	**Bag-of-Freebies**
Augmentation des données	Augmentation des données
- Mosaic	- Mosaic
- CutMix	- Entraînement auto-adversaire
Regularisation	Perte CIoU
- DropBlock	Normalisation en mini-lots croisée (CmBN)
Lissage des étiquettes de classe	Élimination de la sensibilité à la grille
	Multiples ancres pour une vérité terrain unique
	Scheduler de réduction de cosinus
	Optimisation des hyperparamètres
	Formes d'entraînement aléatoires
Bag-of-Specials	**Bag-of-Specials**
Activation Mish	Activation Mish
Connexions partielles entre les étapes (CSP)	Bloc de pyramide spatiale (SPP)
Connexions résiduelles pondérées multi-entrées	Module d'attention spatiale (SAM)
	Réseau d'agrégation de chemin (PAN)
	Distance-IoU & Suppression non maximale

sur MS COCO.

L'architecture agrandie du modèle a été appelée YOLOv4-large, qui incluait trois tailles différentes : P5, P6 et P7. Cette architecture a été conçue pour les GPU en cloud et a atteint des performances de pointe, surpassant tous les modèles précédents [74, 75, 76] avec un AP de 56% sur MS COCO.

12 YOLOR

YOLOR [77] a été publié sur ArXiv en mai 2021 par la même équipe de recherche que YOLOv4. YOLOR signifie *You Only Learn One Representation*. Dans cet article, les auteurs ont adopté une approche différente ; ils ont développé une approche d'apprentissage multitâche qui vise à créer un modèle unique pour différentes tâches (*e.g.* classification, détection, estimation de pose) en apprenant une représentation générale et en utilisant des sous-réseaux pour créer des représentations spécifiques à chaque tâche. Avec l'idée que la méthode d'apprentissage conjoint traditionnelle conduit souvent à une génération de caractéristiques sous-optimales, YOLOR vise à surmonter cela en encodant les connaissances implicites des réseaux neuronaux pour les appliquer à plusieurs tâches, de manière similaire à la façon dont les humains utilisent leurs expériences passées pour aborder de nouveaux problèmes. Les résultats ont montré que l'introduction de connaissances implicites dans le réseau neuronal bénéficie à toutes les tâches.

Évalué sur le jeu de données MS COCO test-dev 2017, YOLOR a obtenu un AP de 55,4% et un AP_{50} de 73,3% à 30 images par seconde sur une NVIDIA V100.

13 YOLOX

YOLOX [78] a été publié sur ArXiv en juillet 2021 par une équipe de recherche de Megvii Technology. Développé en utilisant Pytorch et se basant sur YOLOV3 d'Ultralytics comme point de départ, il présente cinq changements principaux : une architecture sans ancrage, plusieurs positifs, une tête découplée, une attribution d'étiquettes avancée et des augmentations fortes. Il a atteint des résultats de pointe en 2021 avec un équilibre optimal entre vitesse et précision,

avec un AP de 50,1% à 68,9 images par seconde sur Tesla V100. Dans ce qui suit, on décrit les cinq principaux changements de YOLOX par rapport à YOLOv3 :

1. **Sans ancrage**. Depuis YOLOv2, toutes les versions ultérieures de YOLO étaient des détecteurs basés sur des ancres. YOLOX, s'inspirant des détecteurs d'objets de pointe sans ancrage tels que CornerNet [79], CenterNet [80] et FCOS [81], est revenu à une architecture sans ancrage, simplifiant ainsi le processus d'entraînement et de décodage. Le sans ancrage a augmenté l'AP de 0,9 points par rapport à la ligne de base YOLOv3.
2. **Multiples positifs**. Pour compenser les grands déséquilibres produits par l'absence d'ancres, les auteurs utilisent un échantillonnage central [81] où ils attribuent la zone centrale de 3×3 comme positifs. Cette approche a augmenté l'AP de 2,1 points.
3. **Tête découplée**. Dans [82, 83], il a été démontré qu'il pouvait y avoir un désalignement entre la confiance de classification et la précision de localisation. En raison de cela, YOLOX sépare ces deux éléments en deux têtes (comme illustré dans la figure 12), l'une pour les tâches de classification et l'autre pour les tâches de régression, améliorant l'AP de 1,1 point et accélérant la convergence du modèle.
4. **Attribution d'étiquettes avancée**. Dans [84], il a été démontré que l'attribution des étiquettes de vérité terrain pouvait être ambiguë lorsque les boîtes de plusieurs objets se chevauchent, et la procédure d'attribution a été formulée comme un problème de transport optimal (OT). YOLOX, s'inspirant de ce travail, a proposé une version simplifiée appelée simOTA. Ce changement a augmenté l'AP de 2,3 points.
5. **Augmentations fortes**. YOLOX utilise les augmentations MixUP [85] et Mosaic. Les auteurs ont constaté que la préparation préalable sur ImageNet n'était plus bénéfique après l'utilisation de ces augmentations. Les augmentations fortes ont augmenté l'AP de 2,4 points.

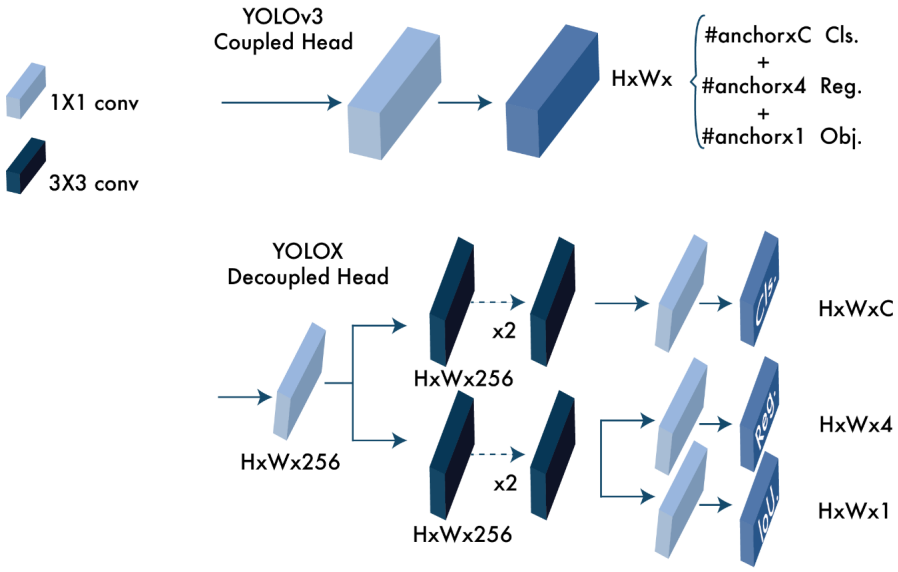

FIGURE 12 – Différence entre la tête de YOLOv3 et la tête découplée de YOLOX. Pour chaque niveau du FPN, ils ont utilisé une couche de convolution 1×1 pour réduire le nombre de canaux de caractéristiques à 256, puis ils ont ajouté deux branches parallèles avec deux couches de convolution 3×3 chacune pour les tâches de confiance de classement (classification) et de localisation (régression). La branche IoU est ajoutée à la tête de régression.

14 YOLOv6

YOLOv6 [86] a été publié en septembre 2022 dans ArXiv par le département de l'IA de Meituan Vision. Tout comme YOLOv4 et YOLOv5, il propose différents modèles de différentes tailles pour des applications industrielles. Suivant la tendance des méthodes basées sur les points d'ancrage [78, 81], YOLOv6 adopte un détecteur sans points d'ancrage. Les principales nouveautés de ce modèle sont résumées ci-dessous :

1. **Nouvelle architecture de base basée sur RepVGG** [87] appelée EfficientRep qui utilise une parallélisme plus élevé que les architectures précédentes de YOLO. Pour le neck, ils utilisent PAN [62] amélioré avec des blocs RepBlocks [87] ou CSPStackRep [61] pour les modèles plus grands. Et inspirés par YOLOX, ils ont développé une tête découplée efficace.

2. **Attribution des étiquettes** en utilisant l'approche d'apprentissage d'alignement de tâche introduite dans TOOD [88].

3. **Nouvelles fonctions de perte pour la classification et la régression**. Ils utilisent une perte de classification VariFocal [89] et une perte de régression SIoU [90]/GIoU [91].

4. **Stratégie d'auto-distillation** pour les tâches de régression et de classification.

5. **Schéma de quantification** pour la détection utilisant RepOptimizer [92] et la distillation par canal [93] qui ont contribué à obtenir un détecteur plus rapide.

14.1 Résultats de YOLOv6

Évalué sur le jeu de données MS COCO test-dev 2017, YOLOv6-L a obtenu un AP de 52,5% et un AP50 de 70% à environ 50 images par seconde sur une NVIDIA Tesla T4.

15 YOLOv7

YOLOv7 [94] a été publié dans ArXiv en juillet 2022 par les mêmes auteurs que YOLOv4 et YOLOR. À l'époque, il surpassait tous les détecteurs d'objets connus en termes de vitesse et de précision dans une plage de 5 FPS à 160 FPS. Comme YOLOv4, il a été entraîné uniquement avec l'ensemble de données MS COCO, sans pré-entraînement des modèles de base. YOLOv7 a proposé plusieurs modifications d'architecture et une série de "bag-of-freebies", qui ont augmenté la précision sans affecter la vitesse d'inférence, seulement le temps d'entraînement.

Les modifications d'architecture de YOLOv7 sont les suivantes :

- **Extended efficient layer aggregation network (E-ELAN)**. ELAN [95] est une stratégie qui permet à un modèle profond d'apprendre et de converger de manière plus efficace en contrôlant le chemin du gradient le plus court le plus long. YOLOv7 a proposé E-ELAN qui fonctionne pour les modèles avec des blocs de calcul empilés de manière illimitée. E-ELAN combine les caractéristiques de différents groupes en les mélangeant et en fusionnant leur cardinalité pour renforcer l'apprentissage du réseau sans détruire le chemin de gradient original.
- **Mise à l'échelle du modèle pour les modèles basés sur la concaténation**. La mise à l'échelle génère des modèles de différentes tailles en ajustant certaines caractéristiques du modèle. L'architecture de YOLOv7 est une architecture basée sur la concaténation dans laquelle les techniques de mise à l'échelle standard, telles que la mise à l'échelle de la profondeur, provoquent un changement de ratio entre le canal d'entrée et le canal de sortie d'une couche de transition, ce qui entraîne une diminution de l'utilisation matérielle du modèle. YOLOv7 a proposé une nouvelle stratégie de mise à l'échelle des modèles basés sur la concaténation, dans laquelle la profondeur et la largeur du bloc sont mises à l'échelle avec le même facteur pour maintenir la structure optimale du modèle.

Les "bag-of-freebies" utilisés dans YOLOv7 comprennent :

- **Convolution re-paramétrisée planifiée**. Comme dans YOLOv6, l'architecture de YOLOv7 s'inspire également des convolutions re-paramétrisées (RepConv) [87]. Cependant, ils ont constaté que la connexion identité dans RepConv détruit la résiduelle dans ResNet [53] et la concaténation dans DenseNet [96]. Pour cette raison, ils ont supprimé la connexion identité et l'ont appelée RepConvN.
- **Attribution d'étiquettes grossières pour la tête auxiliaire et attribution d'étiquettes fines pour la tête principale**. La tête principale est responsable de la sortie finale, tandis que la tête auxiliaire aide à l'entraînement.
- **Normalisation par lots (batch normalization) dans la couche conv-bn-activation**. Cela intègre la moyenne et la variance de la normalisation par lots dans le biais et le poids de la couche de convolution lors de l'inférence.
- **Connaissance implicite** inspirée de YOLOR [77].
- **Moyenne mobile exponentielle** comme modèle d'inférence final.

15.1 Comparison avec YOLOv4 et YOLOR

Dans cette section, on met en évidence les améliorations apportées par YOLOv7 par rapport aux modèles YOLO précédents développés par les mêmes auteurs.

Par rapport à YOLOv4, YOLOv7 a réussi à réduire de 75% le nombre de paramètres et de 36% le calcul, tout en améliorant simultanément la précision moyenne (AP) de 1,5%.

En revanche, par rapport à YOLOv4-tiny, YOLOv7-tiny a réussi à réduire les paramètres et le calcul de 39% et 49% respectivement, tout en maintenant la même AP.

Enfin, par rapport à YOLOR, YOLOv7 a réduit le nombre de paramètres et le calcul de 43% et 15% respectivement, tout en enregistrant une légère augmentation de 0,4% de l'AP.

15.2 Résultats de YOLOv7

Évalué sur l'ensemble de données MS COCO test-dev 2017, YOLOv7-E6 a obtenu une précision moyenne (AP) de 55,9% et un AP_{50} de 73,5% avec une taille d'entrée de 1280 pixels, à une vitesse de 50 images par seconde (FPS) sur un NVIDIA V100.

16 DAMO-YOLO

DAMO-YOLO [97] a été publié en novembre 2022 par le groupe Alibaba. Inspiré par les technologies actuelles, DAMO-YOLO comprenait les éléments suivants :

- **Recherche d'architecture neuronale** (**NAS**). Ils ont utilisé une méthode appelée MAE-NAS [98] développée par Alibaba pour trouver automatiquement une architecture efficace.
- "**Neck**" (**cou**) **large**. Inspiré par GiraffeDet [99], CSPNet [61] et ELAN [95], les auteurs ont conçu un neck qui peut fonctionner en temps réel appelé Efficient-RepGFPN.
- "**Head**" (**tête**) **petite**. Les auteurs ont constaté qu'un neck large et une tête petite donnent de meilleures performances, et ils n'ont laissé qu'une couche linéaire pour la classification et une pour la régression. Ils ont appelé cette approche ZeroHead.
- **Assignation des étiquettes AlignedOTA**. Les méthodes d'assignation dynamique des étiquettes, telles que OTA[84] et TOOD[88], sont devenues populaires en raison de leurs améliorations significatives par rapport aux méthodes statiques. Cependant, le désalignement entre la classification et la régression reste un problème, en partie en raison du déséquilibre entre les pertes de classification et de régression. Pour résoudre ce problème, leur méthode AlignOTA introduit la focal loss [75] dans le coût de classification et utilise l'IoU de la prédiction et de la boîte ground truth comme étiquette souple, permettant la sélection d'échantillons alignés pour chaque cible et résolvant le problème d'une perspective globale.
- **Distillation des connaissances**. Leur stratégie proposée se compose de deux étapes : le professeur guide l'élève dans la première étape et l'élève se perfectionne de manière indépendante dans la deuxième étape. De plus, ils intègrent deux améliorations dans l'approche de distillation : le module Align, qui adapte les caractéristiques de l'élève à la même résolution que celles du professeur, et la température dynamique par canal, qui normalise les caractéristiques du professeur et de l'élève pour réduire l'impact des différences de valeur réelle.

Les auteurs ont généré des modèles adaptés nommés DAMO-YOLO-Tiny/Small/Medium, le meilleur modèle atteignant une précision moyenne (AP) de 50,0% à une vitesse de 233 images par seconde (FPS) sur un NVIDIA V100.

17 YOLOv8

YOLOv8 [100] a été publié en janvier 2023 par Ultralytics, la société qui a développé YOLOv5. Étant donné qu'à l'heure de la rédaction de cet article, il n'y avait pas encore de document sur YOLOv8, nous avons besoin de perspectives sur les décisions architecturales par rapport aux autres versions de YOLO. Suivant la tendance actuelle, YOLOv8 est sans ancre, réduisant le nombre de prédictions de boîtes et accélérant le processus de Non-Maximum Suppression (NMS). De plus, YOLOv8 utilise l'augmentation par mosaïque lors de l'entraînement ; cependant, étant donné qu'il a été constaté que cette augmentation peut être néfaste si elle est utilisée pendant toute la durée de l'entraînement, elle est

désactivée pour les dix dernières époques.

YOLOv8 peut être exécuté à partir de l'interface de ligne de commande (CLI) ou peut également être installé en tant que package PIP. De plus, il est fourni avec plusieurs intégrations pour l'étiquetage, l'entraînement et le déploiement.

YOLOv8 propose cinq versions échelonnées : YOLOv8n (nano), YOLOv8s (small), YOLOv8m (medium), YOLOv8l (large) et YOLOv8x (extra large).

17.1 Performance de YOLOv8

Évalué sur le jeu de données MS COCO test-dev 2017, YOLOv8x a obtenu un AP de 53,9% avec une taille d'image de 640 pixels (comparé à 50,7% pour YOLOv5 avec la même taille d'entrée), avec une vitesse de 280 FPS sur un NVIDIA A100 et TensorRT.

18 PP-YOLO, PP-YOLOv2, et PP-YOLOE

Les modèles PP-YOLO ont évolué parallèlement aux modèles YOLO que nous avons décrits. Cependant, nous avons décidé de les regrouper dans une seule section car ils ont commencé avec YOLOv3 et se sont progressivement améliorés par rapport à la version PP-YOLO précédente. Néanmoins, ces modèles ont eu une influence sur l'évolution de YOLO. PP-YOLO [76], similaire à YOLOv4 et YOLOv5, était basé sur YOLOv3. Il a été publié sur ArXiv en juillet 2020 par des chercheurs de Baidu Inc. Les auteurs ont utilisé la plateforme d'apprentissage en profondeur PaddlePaddle [101], d'où le nom *PP*. Suivant la tendance que nous avons observée à partir de YOLOv4, PP-YOLO a ajouté dix astuces existantes pour améliorer la précision du détecteur, tout en maintenant la vitesse inchangée. Selon les auteurs, cet article n'avait pas pour but d'introduire un nouveau détecteur d'objets, mais de montrer comment construire un meilleur détecteur étape par étape. La plupart des astuces utilisées par PP-YOLO sont différentes de celles utilisées dans YOLOv4, et celles qui se chevauchent utilisent une implémentation différente. Les changements apportés par PP-YOLO par rapport à YOLOv3 sont les suivants :

1. **Un réseau de neurones ResNet50-vd** remplace le réseau DarkNet-53 en utilisant une architecture augmentée avec des convolutions déformables [102] dans la dernière étape, ainsi qu'un modèle pré-entraîné distillé, qui présente une meilleure précision de classification sur ImageNet. Cette architecture a été appelée ResNet5-vd-dcn.
2. **Une taille de batch plus grande** pour améliorer la stabilité de l'apprentissage, passant de 64 à 192, ainsi qu'une planification d'apprentissage et un taux d'apprentissage mis à jour.
3. **Maintien des moyennes mobiles** pour les paramètres entraînés et utilisation de ces moyennes au lieu des valeurs finales entraînées.
4. **DropBlock** est appliqué uniquement à la FPN.
5. **Une perte IoU** est ajoutée dans une autre branche en plus de la perte L1 pour la régression des boîtes englobantes.
6. **Une branche de prédiction IoU** est ajoutée pour mesurer la précision de localisation, ainsi qu'une perte sensible à IoU. Lors de l'inférence, YOLOv3 multiplie la probabilité de classification et le score d'objectivité pour calculer la détection finale, PP-YOLO multiplie également l'IoU prédit pour tenir compte de la précision de localisation.
7. **Une approche Grid Sensitive** similaire à YOLOv4 est utilisée pour améliorer la prédiction du centre des boîtes englobantes à la frontière de la grille.
8. **Matrix NMS** [103] est utilisé, ce qui permet une exécution parallèle, le rendant plus rapide que le NMS traditionnel.
9. **CoordConv** [104] est utilisé dans pour la convolution 1×1 du Feature Pyramid Network (FPN) ainsi que sur la première couche de convolution de la tête de détection. CoordConv permet au réseau d'apprendre l'invariance translationnelle, améliorant ainsi la localisation des détections.
10. **Spatial Pyramid Pooling** est utilisé uniquement sur la carte des caractéristiques supérieure pour augmenter le champ réceptif du réseau de base.

18.1 Améliorations et prétraitement de PP-YOLO

PP-YOLO utilisait les améliorations et le prétraitement suivants :

1. Mixup Training [85] avec un poids échantillonné à partir d'une distribution $Beta(\alpha, \beta)$ où $\alpha = 1,5$ et $\beta = 1,5$.
2. Distorsion aléatoire des couleurs.
3. Expansion aléatoire.
4. Recadrage aléatoire et retournement aléatoire avec une probabilité de 0,5.
5. Normalisation z-score des canaux RGB avec une moyenne de [0,485, 0,456, 0,406] et un écart-type de [0,229, 0,224, 0,225].
6. Multiples tailles d'image sélectionnées de manière égale parmi [320, 352, 384, 416, 448, 480, 512, 544, 576, 608].

18.2 Résultats de PP-YOLO

Évalué sur le jeu de données MS COCO test-dev 2017, PP-YOLO a obtenu une précision moyenne (AP) de 45,9% et une précision moyenne à IoU 0,5 (AP_{50}) de 65,2% à une vitesse de 73 images par seconde (FPS) sur une carte graphique NVIDIA V100.

18.3 PP-YOLOv2

PP-YOLOv2 [105] a été publié sur ArXiv en avril 2021 et a apporté quatre améliorations à PP-YOLO qui ont permis d'augmenter les performances de 45,9% d'AP à 49,5% d'AP à une vitesse de 69 images par seconde (FPS) sur une carte graphique NVIDIA V100. Les modifications de PP-YOLOv2 par rapport à PP-YOLO sont les suivantes :

1. **Remplacement du backbone ResNet50 par ResNet101**.
2. **Utilisation du réseau d'agrégation de chemins** (**PAN**) à la place de FPN, similaire à YOLOv4.
3. **Utilisation de la fonction d'activation Mish**. Contrairement à YOLOv4 et YOLOv5, la fonction d'activation Mish a été appliquée uniquement dans la partie de détection pour garder le backbone inchangé avec ReLU.
4. **Des tailles d'entrée plus grandes** aident à améliorer les performances sur les petits objets. Ils ont augmenté la plus grande taille d'entrée de 608 à 768 et réduit la taille du lot de 24 à 12 images par GPU. Les tailles d'entrée sont choisies de manière uniforme parmi [320, 352, 384, 416, 448, 480, 512, 544, 576, 608, 640, 672, 704, 736, 768].
5. **Une branche modifiée pour la prise en compte de l'indice de localisation** (**IoU**). Ils ont modifié le calcul de la perte IoU pour utiliser un format d'étiquette souple plutôt qu'un format de poids souple.

18.4 PP-YOLOE

PP-YOLOE [106] a été publié sur ArXiv en mars 2022. Il apporte des améliorations par rapport à PP-YOLOv2 en atteignant une performance de 51,4% AP à 78,1 FPS sur NVIDIA V100. Les principaux changements de PP-YOLOE par rapport à PP-YOLOv2 sont les suivants :

1. **Architecture sans ancrage**. Suivant les tendances de l'époque influencées par les travaux de [81, 80, 79, 78], PP-YOLOE utilise une architecture sans ancrage.
2. **Nouvelle architecture de base et couche intermédiaire**. Inspirés par TreeNet [107], les auteurs ont modifié l'architecture de la base et de la couche intermédiaire en utilisant des blocs RepResBlocks qui combinent des connexions résiduelles et denses.
3. **Apprentissage de l'alignement des tâches** (**TAL**). YOLOX a été le premier à soulever le problème de l'alignement des tâches, où la confiance en la classification et la précision de localisation ne sont pas toujours en accord. Pour réduire ce problème, PP-YOLOE a mis en œuvre le TAL tel que proposé dans TOOD [88], qui comprend une attribution dynamique des étiquettes combinée à une perte d'alignement des tâches.
4. **Tête efficace alignée sur les tâches** (**ET-head**). Contrairement à YOLOX où les têtes de classification et de localisation sont découplées, PP-YOLOE utilise une seule tête basée sur TOOD pour améliorer la vitesse et la précision.
5. **Pertes Varifocal** (**VFL**) et **Distribution focal** (**DFL**). VFL [89] pondère la perte des échantillons positifs en utilisant le score cible, en donnant plus de poids à ceux ayant une IoU élevée. Cela donne la priorité aux échantillons de haute qualité lors de l'entraînement. De même, les deux utilisent le score de classification

conscient de l'IoU (IACS) comme cible, permettant l'apprentissage conjoint de la qualité de classification et de localisation, assurant ainsi la cohérence entre l'entraînement et l'inférence. DFL [108] étend la perte focale des étiquettes discrètes aux étiquettes continues, ce qui permet une optimisation réussie des représentations améliorées combinant l'estimation de qualité et la prédiction de classe. Cela permet une représentation précise de la distribution flexible dans les données réelles, éliminant ainsi le risque d'incohérence.

Comme pour les versions précédentes de YOLO, les auteurs ont généré plusieurs modèles à différentes échelles en faisant varier la largeur et la profondeur de la base et de la couche intermédiaire. Les modèles sont appelés PP-YOLOE-s (petit), PP-YOLOE-m (moyen), PP-YOLOE-l (grand) et PP-YOLOE-x (extra large).

19 Discussion

Ce document examine 15 versions de YOLO, allant du modèle original YOLO au plus récent YOLOv8. Le tableau 4 fournit un aperçu des versions de YOLO examinées. À partir de ce tableau, nous pouvons identifier plusieurs motifs clés :

- **Ancrages** : Le modèle original YOLO était relativement simple et n'utilisait pas d'ancrages, tandis que les modèles les plus avancés utilisaient des détecteurs à deux étapes avec des ancrages. YOLOv2 a incorporé des ancrages, ce qui a permis d'améliorer la précision de prédiction des boîtes englobantes. Cette tendance s'est maintenue pendant cinq ans jusqu'à ce que YOLOX introduise une approche sans ancrage qui a atteint des résultats de pointe. Depuis lors, les versions ultérieures de YOLO ont abandonné l'utilisation des ancrages.
- **Framework** : Initialement, YOLO a été développé en utilisant le cadre de travail Darknet, et les versions ultérieures ont suivi cette voie. Cependant, lorsque Ultralytics a porté YOLOv3 vers PyTorch, les versions restantes de YOLO ont été développées en utilisant PyTorch, ce qui a entraîné une augmentation des améliorations. Un autre langage d'apprentissage profond utilisé est PaddlePaddle, un cadre de travail open-source initialement développé par Baidu.
- **Architecture de base** : Les architectures de base des modèles YOLO ont subi d'importants changements au fil du temps. À partir de l'architecture Darknet, qui comprenait des couches de convolution simples et des couches de max pooling, les modèles ultérieurs ont incorporé des connexions partielles entre les étapes (CSP) dans YOLOv4, une reparamétrisation dans YOLOv6 et YOLOv7, et une recherche d'architecture neuronale dans DAMO-YOLO.
- **Performance** : Bien que les performances des modèles YOLO se soient améliorées au fil du temps, il convient de noter qu'ils privilégient souvent un équilibre entre la vitesse et la précision plutôt que de se concentrer uniquement sur la précision. Ce compromis est un aspect essentiel du cadre de travail YOLO, permettant la détection en temps réel d'objets dans diverses applications.

TABLE 4 – Résumé des architectures YOLO. La métrique rapportée pour YOLO et YOLOv2 était basée sur VOC2007, tandis que les autres sont rapportées sur COCO2017.

Version	Date	Ancre	Framework	Backbone	AP (%)
YOLO	2015	Non	Darknet	Darknet24	63.4
YOLOv2	2016	Oui	Darknet	Darknet24	63.4
YOLOv3	2018	Oui	Darknet	Darknet53	36.2
YOLOv4	2020	Oui	Darknet	CSPDarknet53	43.5
YOLOv5	2020	Oui	Pytorch	Modified CSP v7	55.8
PP-YOLO	2020	Oui	PaddlePaddle	ResNet50-vd	45.9
Scaled-YOLOv4	2021	Oui	Pytorch	CSPDarknet	56.0
PP-YOLOv2	2021	Oui	PaddlePaddle	ResNet101-vd	50.3
YOLOR	2021	Oui	Pytorch	CSPDarknet	55.4
YOLOX	2021	Non	Pytorch	Modified CSP v5	51.2
PP-YOLOE	2022	Non	PaddlePaddle	CSPRepResNet	54.7
YOLOv6	2022	Non	Pytorch	EfficientRep	52.5
YOLOv7	2022	Non	Pytorch	RepConvN	56.8
DAMO-YOLO	2022	Non	Pytorch	MAE-NAS	50.0
YOLOv8	2023	Non	Pytorch	YOLO v8	53.9

19.1 Compromis entre vitesse et précision

La famille de modèles de détection d'objets YOLO a constamment cherché à équilibrer vitesse et précision, visant à fournir des performances en temps réel sans compromettre la qualité des résultats de détection. Au fur et à mesure de l'évolution du framework YOLO à travers ses différentes itérations, ce compromis a été un thème récurrent, chaque version cherchant à optimiser ces objectifs concurrents de manière différente. Dans le modèle original YOLO, l'accent principal était mis sur l'obtention d'une détection d'objets à grande vitesse. Le modèle utilisait un seul réseau neuronal convolutif (CNN) pour prédire directement les emplacements et les classes des objets à partir de l'image d'entrée, permettant un traitement en temps réel. Cependant, cette priorité donnée à la vitesse entraînait un compromis en termes de précision, notamment lorsqu'il s'agissait de petits objets ou d'objets avec des boîtes englobantes qui se chevauchent.

Les versions ultérieures de YOLO ont introduit des améliorations pour remédier à ces limitations tout en conservant les capacités en temps réel du framework. Par exemple, YOLOv2 (YOLO9000) a introduit des ancres et des couches de transfert pour améliorer la localisation des objets, ce qui a permis d'obtenir une précision accrue. De plus, YOLOv3 a amélioré les performances du modèle en utilisant une architecture d'extraction de caractéristiques multi-échelles, ce qui permet une meilleure détection d'objets à différentes échelles.

Le compromis entre vitesse et précision est devenu plus nuancé à mesure que le framework YOLO évoluait. Des modèles tels que YOLOv4 et YOLOv5 ont introduit des innovations, comme de nouvelles architectures de réseau, des techniques d'augmentation de données améliorées et des stratégies d'entraînement optimisées. Ces développements ont permis d'obtenir des gains significatifs en termes de précision sans affecter de manière drastique les performances en temps réel des modèles.

À partir de Scaled YOLOv4, tous les modèles YOLO officiels ont affiné le compromis entre vitesse et précision, offrant différentes échelles de modèles adaptées à des applications spécifiques et à des exigences matérielles. Par exemple, ces versions proposent souvent des modèles légers optimisés pour les dispositifs embarqués, sacrifiant la précision en échange d'une complexité computationnelle réduite et de temps de traitement plus rapides.

20 YOLO : Quel Avenir ?

À mesure que le framework YOLO continue d'évoluer, nous anticipons que les tendances et les possibilités suivantes façonneront les développements futurs :

Intégration des dernières techniques : Les chercheurs et les développeurs continueront d'affiner l'architecture YOLO en utilisant les méthodes les plus récentes en matière d'apprentissage profond, d'augmentation des données et de techniques d'entraînement. Ce processus d'innovation continu améliorera probablement les performances, la robustesse et l'efficacité du modèle.

Évolution des références de comparaison : La référence actuelle pour l'évaluation des modèles de détection d'objets, COCO 2017, pourrait éventuellement être remplacée par une référence plus avancée et plus exigeante. Cela reflète la transition entre la référence VOC 2007 utilisée dans les deux premières versions de YOLO, soulignant le besoin de références plus exigeantes à mesure que les modèles deviennent plus sophistiqués et précis.

Prolifération des modèles YOLO et leurs applications : À mesure que le framework YOLO progresse, nous nous attendons à une augmentation du nombre de modèles YOLO publiés chaque année, ainsi qu'à une expansion correspondante des applications. À mesure que le framework devient plus polyvalent et puissant, il sera probablement utilisé dans des domaines encore plus variés, allant des appareils électroménagers aux voitures autonomes.

Expansion vers de nouveaux domaines : Les modèles YOLO ont le potentiel d'étendre leurs capacités au-delà de la détection et de la segmentation d'objets, en se diversifiant dans des domaines tels que le suivi d'objets dans les vidéos et l'estimation des points clés en 3D. À mesure que ces modèles évoluent, ils peuvent devenir la base de nouvelles solutions qui abordent un éventail plus large de tâches de vision par ordinateur.

Adaptabilité à divers matériels : Les modèles YOLO s'étendront également aux plateformes matérielles, des appareils IoT aux clusters informatiques haute performance. Cette adaptabilité permettra le déploiement de modèles YOLO dans différents contextes, en fonction des exigences et des contraintes de l'application. De plus, en adaptant les modèles aux spécifications matérielles spécifiques, YOLO peut être rendu accessible et efficace pour un plus grand nombre d'utilisateurs et d'industries.

Références

[1] W. Lan, J. Dang, Y. Wang, and S. Wang. "Pedestrian detection based on YOLO network model". pages 1547–1551. IEEE, 2018.

[2] W.-Y. Hsu and W.-Y. Lin. "Adaptive fusion of multi-scale YOLO for pedestrian detection". *IEEE Access*, vol. 9, pages 110063–110073, 2021.

[3] A. Benjumea, I. Teeti, F. Cuzzolin, and A. Bradley. "YOLO-z : Improving small object detection in YOLOv5 for autonomous vehicles". *arXiv preprint arXiv*, pages 2112–11798, 2021.

[4] N. M. A. A. Dazlee, S. A. Khalil, S. Abdul-Rahman, and S. Mutalib. "Object detection for autonomous vehicles with sensor-based technology using YOLO". *International Journal of Intelligent Systems and Applications in Engineering*, vol. 10, no. 1, pages 129–134, 2022.

[5] S. Liang, H. Wu, L. Zhen, Q. Hua, S. Garg, G. Kaddoum, M. M. Hassan, and K. Yu. "Edge YOLO : Real-time intelligent object detection system based on edge-cloud cooperation in autonomous vehicles". *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pages 25345–25360, 2022.

[6] Q. Li, X. Ding, X. Wang, L. Chen, J. Son, and J.-Y. Song. "Detection and identification of moving objects at busy traffic road based on YOLO v4". vol. 21, no. 1, pages 141–148, 2021.

[7] S. Shinde, A. Kothari, and V. Gupta. "Yolo based human action recognition and localization". *Procedia computer science*, vol. 133, pages 831–838, 2018.

[8] A. H. Ashraf, M. Imran, A. M. Qahtani, A. Alsufyani, O. Almutiry, A. Mahmood, and M. Attique. "Weapons detection for security and video surveillance using cnn and yolo-v5s". *CMC-Comput. Mater. Contin*, vol. 70, pages 2761–2775, 2022.

[9] Y. Zheng and H. Zhang. "Video analysis in sports by lightweight object detection network under the background of sports industry development". *Computational Intelligence and Neuroscience*, vol. 2022, 2022.

[10] H. Ma, T. Celik, and H. Li. "Fer-yolo : Detection and classification based on facial expressions". In *Image and Graphics : 11th International Conference, ICIG 2021, Haikou, China, August 6–8, 2021, Proceedings, Part I 11*, pages 28–39. Springer, 2021.

[11] Y. Tian, G. Yang, Z. Wang, H. Wang, E. Li, and Z. Liang. "Apple detection during different growth stages in orchards using the improved yolo-v3 model". *Computers and electronics in agriculture*, vol. 157, pages 417–426, 2019.

[12] D. Wu, S. Lv, M. Jiang, and H. Song. "Using channel pruning-based yolo v4 deep learning algorithm for the real-time and accurate detection of apple flowers in natural environments". *Computers and Electronics in Agriculture*, vol. 178, pages 105742, 2020.

[13] M. Lippi, N. Bonucci, R. F. Carpio, M. Contarini, S. Speranza, and A. Gasparri. "A yolo-based pest detection system for precision agriculture". In *2021 29th Mediterranean Conference on Control and Automation (MED)*, pages 342–347. IEEE, 2021.

[14] W. Yang and Z. Jiachun. "Real-time face detection based on yolo". In *2018 1st IEEE international conference on knowledge innovation and invention (ICKII)*, pages 221–224. IEEE, 2018.

[15] W. Chen, H. Huang, S. Peng, and C. Zhou, C.and Zhang. "Yolo-face : a real-time face detector". *The Visual Computer*, vol. 37, pages 805–813, 2021.

[16] M. A. Al-Masni, M. A. Al-Antari, J.-M. Park, G. Gi, T.-Y. Kim, P. Rivera, E. Valarezo, M.-T. Choi, S.-M. Han, and T.-S. Kim. "Simultaneous detection and classification of breast masses in digital mammograms via a deep learning yolo-based cad system". *Computer methods and programs in biomedicine*, vol. 157, pages 85–94, 2018.

[17] Y. Nie, P. Sommella, M. O'Nils, C. Liguori, and J. Lundgren. "Automatic detection of melanoma with yolo deep convolutional neural networks". In *2019 E-Health and Bioengineering Conference (EHB)*, pages 1–4. IEEE, 2019.

[18] H. M. Ünver and E. Ayan. "Skin lesion segmentation in dermoscopic images with combination of yolo and grabcut algorithm". *Diagnostics*, vol. 9, no. 3, p. 72, 2019.

[19] L. Tan, T. Huangfu, L. Wu, and W. Chen. "Comparison of retinanet, ssd, and yolo v3 for real-time pill identification". *BMC medical informatics and decision making*, vol. 21, pp/ 1–11, 2021.

[20] L. Cheng, J. Li, P. Duan, and M. Wang. "A small attentional yolo model for landslide detection from satellite remote sensing images". *Landslides*, vol. 18, no. 8, pp. 2751–2765, 2021.

[21] M.-T. Pham, L. Courtrai, C. Friguet, S. Lefèvre, and A. Baussard. "Yolo-fine : One-stage detector of small objects under various backgrounds in remote sensing images". *Remote Sensing*, vol. 12, no. 15, p. 2501, 2020.

[22] W. Y. Qing, W. Liu, L. Feng, and W. Gao. "Improved yolo network for free-angle remote sensing target detection". *Remote Sensing*, vol. 13, no. 11, p. 2171, 2021.

[23] Z. Zakria, J. Deng, R. Kumar, M. S. Khokhar, J. Cai, and J. Kumar. "Multiscale and direction target detecting in remote sensing images via modified yolo-v4". *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 15, pp. 1039–1048, 2022.

[24] P. Kumar, S. Narasimha Swamy, P. Kumar, G. Purohit, and K. S. Raju. "Real-time, yolo-based intelligent surveillance and monitoring system using jetson tx2". In *Data Analytics and Management : Proceedings of ICDAM*, pages 461–471. Springer, 2021.

[25] K. Bhambani, T. Jain, and K. A. Sultanpure. "Real-time face mask and social distancing violation detection system using yolo". In *2020 IEEE Bangalore humanitarian technology conference (B-HTC)*, pages 1–6. IEEE, 2020.

[26] J. Li, J. Su, Z. Geng, and Y. Yin. "Real-time detection of steel strip surface defects based on improved yolo detection network". *IFAC-PapersOnLine*, vol. 51, no. 21, pp. 76–81, 2018.

[27] E. N. Ukhwah, E. M. Yuniarno, and Y. K. Suprapto. "sphalt pavement pothole detection using deep learning method based on yolo neural networ". In *2019 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, page 35–40. IEEE, 2019.

[28] Y. Du, N. Pan, Z. Xu, F. Deng, Y. Shen, and H. Kang. "Pavement distress detection and classification based on yolo network". *International Journal of Pavement Engineering*, vol. 22, no. 13, pp. 1659–1672, 2021.

[29] R.-C. Chen et al. "Automatic license plate recognition via sliding-window darknet-yolo deep learning". *Image and Vision Computing*, vol. 87, pp. 47–56, 2019.

[30] C. Dewi, R.-C. Chen, X. Jiang, and H. Yu. "Deep convolutional neural network for enhancing traffic sign recognition developed on yolo v4". *Multimedia Tools and Applications*, vol. 81, no. 26, pp. 37821–37845, 2022.

[31] A. M. Roy, J. Bhaduri, T. Kumar, and K. Raj. "Wildect-yolo : An efficient and robust computer vision-based accurate object localization model for automated endangered wildlife detection". *Ecological Informatics*, vol. 75, p. 101919, 2023.

[32] S. Kulik and A. Shtanko. "Experiments with neural net object detection system yolo on small training datasets for intelligent robotics". In *Advanced Technologies in Robotics and Intelligent Systems : Proceedings of ITR 2019*, pages 157–162. Springer, 2020.

[33] D. H. Dos Reis, D. Welfer, M. A. De Souza Leite Cuadros, and D. F. T. Gamarra. "Mobile robot navigation using an object recognition software with rgbd images and the yolo algorithm". *Applied Artificial Intelligence*, vol. 33, no. 14, pp. 1290–1305, 2019.

[34] O. Sahin and S. Oze. "Yolodrone : Improved yolo architecture for object detection in drone images". In *2021 44th International Conference on Telecommunications and Signal Processing (TSP)*, pages 361–365. IEEE, 2021.

[35] C. Chen, Z. Zheng, T. Xu, S. Guo, S. Feng, W. Yao, and Y. La. 'Yolo-based uav technology : A review of the research and its applications". *Drones*, vol. 7, no. 3, p. 190, 2023.

[36] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. 'The pascal visual object classes (voc) challenge". *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[37] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. "Microsoft coco : Common objects in context". In *European conference on computer vision*, pages 740–755. Springer, 2017.

[38] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. "You only look once : Unified, real-time object detection". In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[39] R. Girshick. "Fast r-cnn". In *Proceedings of the IEEE international conference on computer vision*, page 1440–1448, 2015.

[40] A. L. Maas, A. Y. Hannun, and A. Y. Ng, et al. "Rectifier nonlinearities improve neural network acoustic models". *Proc. icml*, vol. 30, p.3, Atlanta, Georgia, USA 2013.

[41] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. "Going deeper with convolutions". In *Proceedings of the IEEE conference on computer vision and pattern recognition*, page 1–9, 2015.

[42] M. Lin, Q. Chen, and S. Yan. "Network in network". *arXiv preprint arXiv*, pages 1312–4400, 2013.

[43] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, and M. Bernstein, et al. "Imagenet large scale visual recognition challenge". *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[44] J. Redmon and A. Farhad. "Yolo9000 : better, faster, stronger". In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[45] S. Ren, K. He, R. Girshick, and J. Sun. "Faster r-cnn : Towards real-time object detection with region proposal networks". *Advances in neural information processing systems*, vol. 28 2015.

[46] J. Redmon and A. Farhadi. "Yolov3 : An incremental improvement". *arXiv preprint arXiv :1804.02767*, 2018.

[47] I. Krasin, T. Duerig, N. Alldrin, V. Ferrari, S. Abu-El-Haija, A. Kuznetsova, H. Rom, J. Uijlings, S. Popov, and A. Veit et al. "Openimages : A public dataset for large-scale multi-label and multi-class image classification". *Dataset available from https ://github.com/openimages*, vol. 2, no. 3, p. 18, 2017.

[48] K. He, X. Zhang, S. Ren, and J. Sun. "Spatial pyramid pooling in deep convolutional networks for visual recognition". *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pages 1904–1916, 2015.

[49] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. "Feature pyramid networks for object detection". In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[50] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao. "Yolov4 : Optimal speed and accuracy of object detection". *arXiv preprint arXiv*, page 2004.10934, 2020.

[51] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. "Deeplab : Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs". *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[52] S. Liu and D. Huang, et al. "Receptive field block net for accurate and fast object detection". In *Proceedings of the European conference on computer vision (ECCV)*, pages 385–400, 2018.

[53] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition". In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[54] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. "Hypercolumns for object segmentation and fine-grained localization". In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 447–456, 2015.

[55] Q. Zhao, T. Sheng, Y. Wang, Z. Tang, Y. Chen, L. Cai, and H. Ling. "M2det : A single-shot object detector based on multi-level feature pyramid network". In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 9259–9266, 2019.

[56] K. He, X. Zhang, S. Ren, and J. Sun. "Delving deep into rectifiers : Surpassing human-level performance on imagenet classification". In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[57] D. Misra. "Mish : A self regularized non-monotonic neural activation function". *arXiv preprint arXiv :1908.08681*, vol. 4, no. 2, pp. 10–48550, 2019.

[58] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis. "Soft-nms–improving object detection with one line of code". In *Proceedings of the IEEE international conference on computer vision*, pages 5561–5569, 2017.

[59] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. "Aggregated residual transformations for deep neural networks". In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[60] M. Tan and Q. Le. "Efficientnet : Rethinking model scaling for convolutional neural networks". In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[61] C.-Y. Wang, H.-Y. M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh. "Cspnet : A new backbone that can enhance learning capability of cnn". In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 390–391, 2020.

[62] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia. "Path aggregation network for instance segmentation". In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8759–8768, 2018.

[63] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon. "Cbam : Convolutional block attention module". In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[64] G. Ghiasi, T.-Y. Lin, and Q. V. Le. "Dropblock : A regularization method for convolutional networks". *Advances in neural information processing systems*, vol. 31 2018.

[65] N. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. "Dropout : a simple way to prevent neural networks from overfitting". *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[66] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. "Rethinking the inception architecture for computer vision". In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[67] M. A. Islam, S. Naha, M. Rochan, N. Bruce, and Y. Wang. "Label refinement network for coarse-to-fine semantic segmentation". *arXiv preprint arXiv :1703.00551*, 2017.

[68] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren. "Distance-iou loss : Faster and better learning for bounding box regression". In *Proceedings of the AAAI conference on artificial intelligence*, volume vol. 34, pp. pages 12993–13000, 2020.

[69] S. Ioffe and C. Szegedy. "Batch normalization : Accelerating deep network training by reducing internal covariate shift". In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[70] I. Loshchilov and F. Hutter. "Sgdr : Stochastic gradient descent with warm restarts". *arXiv preprint arXiv :1608.03983*, 2016.

[71] S. Wang, J. Zhao, N. Ta, X. Zhao, M. Xiao, and H. Wei. "A real-time deep learning forest fire monitoring algorithm based on an improved pruned+ kd model". *Journal of Real-Time Image Processing*, vol. 18, no. 6, pp. 2319–2329, 2021.

[72] G. Jocher. "YOLOv5 by Ultralytics". `https://github.com/ultralytics/yolov5`, 2020. Accessed : Mai 14, 2023.

[73] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao. "Scaled-yolov4 : Scaling cross stage partial network". In *Proceedings of the IEEE/cvf conference on computer vision and pattern recognition*, pages 13029–13038, 2021.

[74] M. Tan, R. Pang, and Q. 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.

[75] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. "Focal loss for dense object detection". In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[76] X. Long, K. Deng, G. Wang, Y. Zhang, Q. Dang, Y. Gao, H. Shen, J. Ren, S. Han, and E. Ding, et al. "Pp-yolo : An effective and efficient implementation of object detector". *arXiv preprint arXiv :2007.12099*, 2020.

[77] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao. "You only learn one representation : Unified network for multiple tasks". *arXiv preprint arXiv :2105.04206*, 2021.

[78] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun. "Yolox : Exceeding yolo series in 2021". *arXiv preprint arXiv :2107.08430*, 2021.

[79] H. Law and J. Deng. "Cornernet : Detecting objects as paired keypoints". In *Proceedings of the European conference on computer vision (ECCV)*, pages 734–750, 2018.

[80] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian. "Centernet : Keypoint triplets for object detection". In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6569–6578, 2019.

[81] Z. Tian, C. Shen, H. Chen, and T. He. "Fcos : Fully convolutional one-stage object detection". In *Proceedings of the IEEE/CVF) international conference on computer vision, pages = 9627–9636, year = 2019*.

[82] G. Song, Y. Liu, and X. Wang. "Revisiting the sibling head in object detector". In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11563–11572, 2020.

[83] Y. Wu, Y. Chen, L. Yuan, Z. Liu, L. Wang, H. Li, and Y. Fu. "Rethinking classification and localization for object detection". In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10186–10195, 2020.

[84] Z. Ge, S. Liu, Z. Li, O. Yoshie, and J. Sun. "Ota : Optimal transport assignment for object detection". In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 303–312, 2021.

[85] H. Zhang, M. Cisse, Y.N. Dauphin, and D. Lopez-Paz. "mixup : Beyond empirical risk minimization". *arXiv preprint arXiv :1710.09412*, 2017.

[86] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, and W. Nie, et al. "Yolov6 : A single-stage object detection framework for industrial applications". *arXiv preprint arXiv :2209.02976*, 2022.

[87] X. Ding, X. Zhang, N. Ma, J. Han, G. Ding, and J. Sun. "epvgg : Making vgg-style convnets great again". In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13733–13742, 2021.

[88] C. Feng, Y. Zhong, Y. Gao, M.R. Scott, and W. Huang. "Tood : Task-aligned one-stage object detection". In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3490–3499. IEEE Computer Society, 2021.

[89] H. Zhang, Y. Wang, F. Dayoub, and N. Sunderhauf. "Varifocalnet : An iou-aware dense object detector". In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8514–8523, 2021.

[90] Z. Gevorgyan. "Siou loss : More powerful learning for bounding box regression". *arXiv preprint arXiv :2205.12740*, 2022.

[91] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese. "Generalized intersection over union : A metric and a loss for bounding box regression". In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 658–666, 2019.

[92] X. Ding, H. Chen, X. Zhang, K. Huang, J. Han, and G. Ding. "Re-parameterizing your optimizers rather than architectures". *arXiv preprint arXiv :2205.15242*, 2022.

[93] C. Shu, Y. Liu, J. Gao, Z. Yan, and C. Shen. "Channel-wise knowledge distillation for dense prediction". In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5311–5320, 2021.

[94] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao. "Yolov7 : Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors". *arXiv preprint arXiv :2207.02696*, 2022.

[95] C.-Y. Wang, H.-Y. M. Liao, and I.-H Yeh. "Designing network design strategies through gradient path analysis". *arXiv preprint arXiv :2211.04800*, 2022.

[96] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. "Densely connected convolutional networks". In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[97] X. Xu, Y. Jiang, W. Chen, Y. Huang, Y. Zhang, and X. Sun. "Damo-yolo : A report on real-time object detection design". *arXiv preprint arXiv :2211.15444*, 2022.

[98] Alibaba. "TinyNAS". `https://github.com/alibaba/lightweight-neural-architecture-search`, 2023. Accessed : Mai 22, 2023.

[99] Z. Tan, J. Wang, X. Sun, M. Lin, and H. Li, et al. "Giraffedet : A heavy-neck paradigm for object detection". In *International Conference on Learning Representations*, 2021.

[100] G. Jocher, A. Chaurasia, and J. Qiu. "YOLO by Ultralytics". `https://github.com/ultralytics/ultralytics,`, 2023. Accessed : Mai 22, 2023.

[101] Y. Ma, D. Yu, T. Wu, and H. Wang. "Paddlepaddle : An open-source deep learning platform from industrial practice". *Frontiers of Data and Computing*, vol. 1, no. 1, pp. 105–115, 2019.

[102] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. "Deformable convolutional networks". In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017.

[103] W. Xinlong, Z. Rufeng, K. Tao, L. Lei, and S. Chunhua. "Solov2 : Dynamic, faster and stronger". In *Proc. NIPS*, 2020.

[104] R. Liu, J. Lehman, P. Molino, F. Petroski Such, E. Frank, A. Sergeev, and J. Yosinski. "An intriguing failing of convolutional neural networks and the coordconv solution". *Advances in neural information processing systems*, vol. 31, 2018.

[105] X. Huang, X. Wang, W. Lv, X. Bai, X. Long, K. Deng, Q. Dang, S. Han, Q. Liu, and X. Hu, et al. "Pp-yolov2 : A practical object detector". *arXiv preprint arXiv :2104.10419*, 2021.

[106] S. Xu, X. Wang, W. Lv, Q. Chang, C. Cui, K. Deng, G. Wang, Q. Dang, S. Wei, and Y. Du, et al. "Pp-yoloe : An evolved version of yolo". *arXiv preprint arXiv :2203.16250*, 2022.

[107] L. Rao. "Treenet : A lightweight one-shot aggregation convolutional network". *arXiv preprint arXiv :2109.12342*, 2021.

[108] X. Li, W. Wang, L. Wu, S. Chen, X. Hu, J. Li, J. Tang, and J. Yang. "Generalized focal loss : Learning qualified and distributed bounding boxes for dense object detection". *Advances in Neural Information Processing Systems*, vol. 33, pp. 21002–21012, 2020.