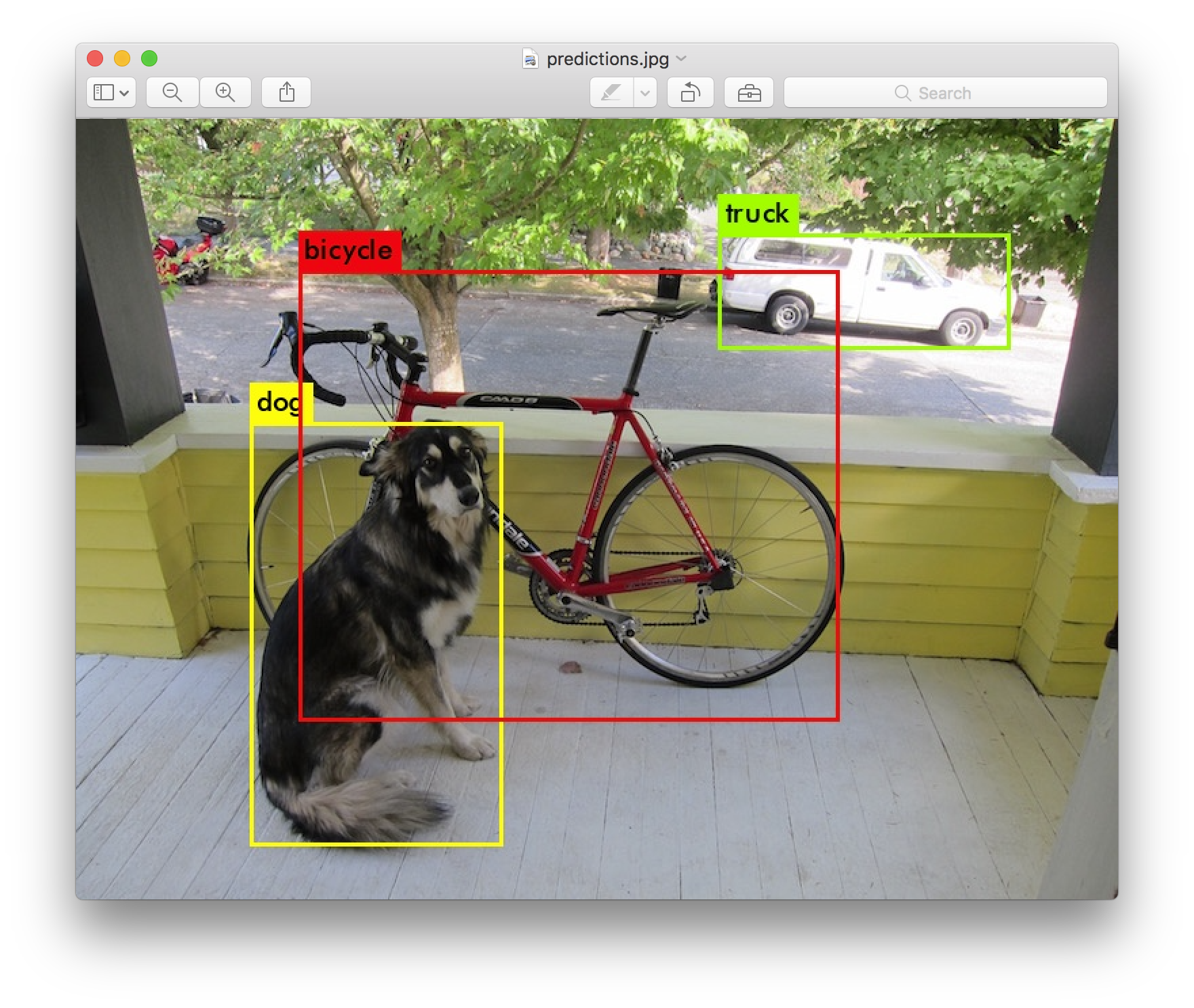


**Recherches bibliographique sur Yolov3, Yolov4 et Yolov5**

# Qu'est-ce que YOLO?

YOLO (You only look once) est un système de détection d’objets en temps réel à la pointe de la technologie. Il s’agit d’un algorithme de reconnaissance et de localisation d'objets basé sur un réseau neuronal profond. Sa plus grande caractéristique est qu'il fonctionne très vite. Par exemple, si vous entrez une image, le système affichera les objets qu'il contient et la position de chaque objet (le cadre rectangulaire contenant l'objet).



# Les informations sur YOLOv3

L'adresse du site YOLOv3：<https://pjreddie.com/darknet/yolo/>

YOLOv3: une amélioration incrémentale

À 320 × 320, YOLOv3 fonctionne en 22 ms à 28,2 mAP, aussi précis qu'un SSD mais trois fois plus rapide. Quand nous regardons l'ancienne métrique de détection .5 IOU mAP YOLOv3 est assez bonne. Il atteint 57,9 AP50 en 51 ms sur un Titan X, contre 57,5 AP50 en 198 ms par RetinaNet, des performances similaires mais 3,8 fois plus rapides.

**2.1 Comparaison de vitesse**

**图表, 折线图

描述已自动生成**

YOLOv3 fonctionne beaucoup plus rapidement que les autres méthodes de détection avec des performances comparables. Les temps d'un M40 ou d'un Titan X, ce sont essentiellement le même GPU.

* 1. **Prédiction de la boîte englobante (Bounding Box Prediction)**

YOLOv3 prédit un score d'objectivité pour chaque boîte englobante à l'aide de la régression logistique. Cela devrait être 1 si la boîte englobante antérieure chevauche un objet de vérité terrain de plus que toute autre boîte englobante antérieure. Si la boîte englobante antérieure n'est pas la meilleure mais chevauche un objet de vérité terrain de plus d'un certain seuil, nous ignorons la prédiction.

图示, 示意图

描述已自动生成

Boîtes englobantes avec priors de dimension et prédiction d'emplacement. Nous prédisons la largeur et la hauteur de la boîte sous forme de décalages par rapport aux centres de gravité du cluster. Nous prédisons les coordonnées du centre de la boîte par rapport à l'emplacement de l'application du filtre à l'aide d'une fonction sigmoïde.

* 1. **Prédiction de classe (Class Prediction)**

Chaque boîte prédit les classes que la boîte englobante peut contenir en utilisant une classification à étiquettes multiples. Nous n'utilisons pas de softmax, nous utilisons simplement des classificateurs logistiques indépendants. nous utilisons la perte d'entropie croisée binaire pour les prédictions de classe.

* 1. **Prédictions à toutes les échelles (Predictions Across Scales)**

YOLOv3 prédit des boîtes à 3 échelles différentes. Nous utilisons toujours le clustering k-means pour déterminer nos a priori de boîte englobante. Nous avons simplement choisi en quelque sorte 9 grappes et 3 échelles de manière arbitraire, puis divisons les grappes uniformément sur les échelles.

* 1. **Extracteur de fonctionnalités (Feature Extractor)**

Darknet-53

Le réseau utilise des couches convolutives successives 3 × 3 et 1 × 1, mais dispose désormais également de connexions de raccourci et est nettement plus grand. Il a 53 couches convolutives.

**手机屏幕截图

描述已自动生成**

Darknet-53 est beaucoup plus puissant que Darknet-19, est meilleur que ResNet-101 et 1,5 fois plus rapide et a des performances similaires à ResNet-152 et est 2 fois plus rapide.表格

描述已自动生成

Quand on regarde la «vieille» métrique de détection de mAP à IOU = .5 (ou AP50 dans le graphique) YOLOv3 est très fort, il a des performances APS relativement élevées. Cependant, ses performances sont comparativement moins bonnes sur les objets de taille moyenne et grande.

表格

描述已自动生成

# Les informations sur YOLOv4

L’adresse du site YOLOv4 : <https://arxiv.org/abs/2004.10934>

YOLOv4 utilise de nouvelles fonctionnalités: WRC, CSP, CmBN, SAT, activation Mish, augmentation des données Mosaic, CmBN, régularisation DropBlock et perte CIoU, et combine certaines d'entre elles pour obtenir des résultats de pointe: 43,5% AP (65,7% AP50) pour l'ensemble de données MS COCO à une vitesse en temps réel de ∼65 FPS sur Tesla V100.

**3.1 Comparaison de vitesse**

图表, 折线图

描述已自动生成

YOLOv4 fonctionne deux fois plus vite que EfficientDet avec des performances comparables. Améliore l'AP et le FPS de YOLOv3 de 10% et 12%, respectivement.

* 1. **Modèles de détection d'objets**

**图片包含 图表

描述已自动生成**

* 1. **Bag of freebies**

Augmentation des données : ajustements par pixel, simulation de problèmes d'occlusion d'objets, plusieurs images ensemble.

Résoudre le problème que la distribution sémantique dans l'ensemble de données peut avoir un biais.

Exprimer la relation du degré d'association entre différentes catégories avec la représentation dure à chaud : label smoothing.

**3.4 Bag of specials**

Modules de plugin :

élargissement du champ récepteur: SPP, ASPP, RFB

introduction d'un mécanisme d'attention: attention par canal, attention ponctuelle

intégration de fonctionnalités: SFAM, ASFF, BiFPN

bonne fonction d'activation

Méthodes de post-traitement: NMS, soft NMS, DIoU NMS

表格

描述已自动生成

* 1. **Méthodologie**

Deux options de réseaux de neurones en temps réel:

• Pour le GPU, nous utilisons un petit nombre de groupes (1 - 8) dans les couches convolutives: CSPResNeXt50 / CSPDarknet53

• Pour VPU - nous utilisons la convolution groupée, mais nous nous abstenons d'utiliser des blocs Squeeze-and-excitation (SE) - en particulier, cela inclut les modèles suivants: EfficientNet-lite / MixNet [76] / GhostNet [21] / Mo- bileNetV3

* 1. **Sélection d'architecture**

Nous choisissons le backbone CSPDarknet53, le module supplémentaire SPP, le col d'agrégation de chemin PANet et la tête YOLOv3 (à base d'ancre) comme architecture de YOLOv4.

* 1. **Sélection de BoF et BoS**

Nous n'avons pas hésité à choisir DropBlock comme méthode de régularisation. En ce qui concerne le choix de la méthode de normalisation, puisque nous nous concentrons sur une stratégie de formation qui n'utilise qu'un seul GPU, syncBN n'est pas pris en compte.

* 1. **Améliorations supplémentaires**

Afin de rendre le détecteur conçu plus adapté à la formation sur un seul GPU:

• Nous introduisons une nouvelle méthode d'augmentation des données de formation mosaïque et auto-adversaire (SAT)

• Nous sélectionnons des hyper-paramètres optimaux tout en appliquant des algorithmes génétiques

• Nous modifions certaines méthodes existantes pour rendre notre conception adaptée à un entraînement et une détection efficaces - SAM modifié, PAN modifié et normalisation en mini-lots croisés (CmBN)

图示

描述已自动生成

CmBN représente une version modifiée du CBN, comme le montre la figure 4, définie comme la normalisation croisée des mini-lots (CmBN). Cela collecte des statistiques uniquement entre les mini-lots d'un même lot.

Nous modifions SAM de l'attention spatiale à l'attention ponctuelle, et remplaçons la connexion de raccourci de PAN à la concaténation, comme le montrent respectivement les figures 5 et 6.

图示

描述已自动生成

# Les informations sur YOLOv5

L’adresse du site YOLOv5 : <https://github.com/ultralytics/yolov5>

Ce référentiel représente la recherche open-source d'Ultralytics sur les futures méthodes de détection d'objets, et intègre nos leçons apprises et les meilleures pratiques évoluées au fil de la formation de milliers de modèles sur des ensembles de données clients personnalisés avec notre précédent référentiel YOLO <https://github.com/ultralytics/yolov3>.

**4.1 Comparaison de vitesse**

La vitesse du GPU mesure le temps de bout en bout par image en moyenne sur 5000 images COCO val2017 à l'aide d'un GPU V100 avec une taille de lot de 32, et comprend le prétraitement d'image, l'inférence PyTorch FP16, le post-traitement et NMS. EfficientDet les données de google / automl à la taille du lot 8.

图表, 折线图, 散点图

描述已自动生成

* 1. **Points de contrôle pré-entraînés**

**表格

描述已自动生成**

* 1. **Exigences**

Python 3.8 ou version ultérieure avec toutes les dépendances requirements.txt installées, y compris torch> = 1.6. Pour installer, exécutez:

$ pip install -r requirements.txt

* 1. **Environnements**

YOLOv5 peut être exécuté dans l'un des environnements vérifiés à jour suivants (avec toutes les dépendances, y compris CUDA / CUDNN, Python et PyTorch préinstallées):

* **Google Colab Notebook** with free GPU:
* **Kaggle Notebook** with free GPU: <https://www.kaggle.com/ultralytics/yolov5>
* **Google Cloud** Deep Learning VM. See [GCP Quickstart Guide](https://github.com/ultralytics/yolov5/wiki/GCP-Quickstart)
* **Docker Image** <https://hub.docker.com/r/ultralytics/yolov5>. See [Docker Quickstart Guide](https://github.com/ultralytics/yolov5/wiki/Docker-Quickstart)
  1. **Entraînement**

Téléchargez COCO et exécutez la commande ci-dessous. Les temps de formation pour YOLOv5s / m / l / x sont de 2/4/6/8 jours sur un seul V100 (multi-GPU fois plus rapide). Utilisez la plus grande - taille de lot autorisée par votre GPU (tailles de lot indiquées pour les appareils de 16 Go).

文本, 信件

描述已自动生成

图形用户界面, 应用程序

描述已自动生成

# La différence entre YOLOv3, YOLOv4 et YOLOv5

YOLOv5 surpasse le cadre de détection de cible open source de Google EfficientDet. Bien que les développeurs de YOLOv5 ne l'aient pas explicitement comparé à YOLOv4, ils affirment que YOLOv5 peut atteindre une détection rapide de 140 FPS sur Tesla P100; en comparaison, , Le résultat de référence de YOLOv4 a été obtenu à 50 FPS.

图片包含 表格

描述已自动生成

La taille de YOLOv5 n'est que de 27 Mo, et YOLOv4 utilisant une architecture darknet a 244 Mo. Cela montre que YOLOv5 est vraiment petit, près de 90% plus petit que YOLOv4. En termes d'indicateurs de précision, YOLOv5 est équivalent à YOLOv4.

Résumé: YOLOv5 prétend être très rapide, a une taille de modèle très légère et équivaut à la référence YOLOv4 en termes de précision.