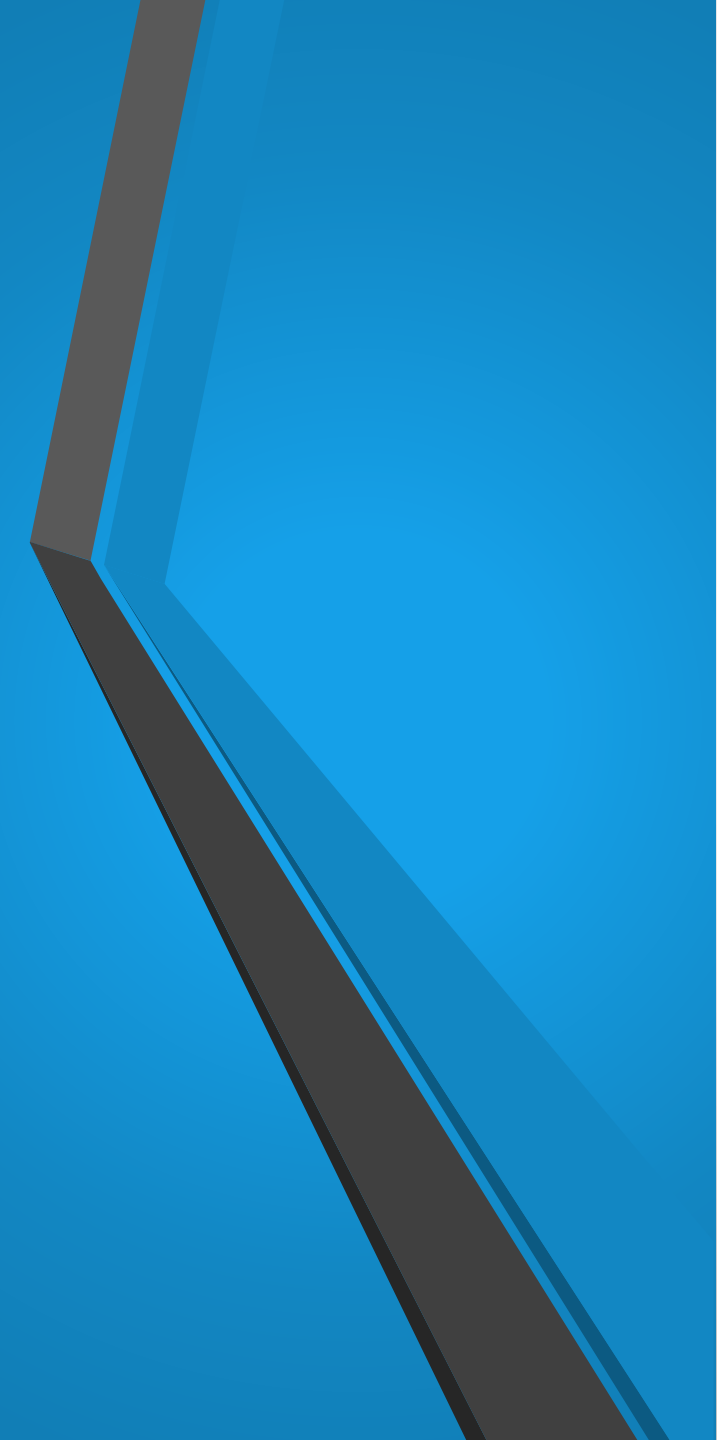


Mickaël REULIER

Détection d'objets - YOLO

SOMMAIRE

- Les bases de la détection d'objets
- Pourquoi la détection d'objets?
- Qu'est-ce que Yolo?
- Les différentes versions de Yolo
 - Yolo v1
 - Yolo v2
 - Yolo v3
 - Yolo v4



Les bases de la détection d'objets

La classification

- Qu'est-ce qu'il y a dans cette image?
- Déterminer la probabilité que l'objet dans l'image appartienne à différentes catégories (chien, chat, ...)
- Un seul objet par image



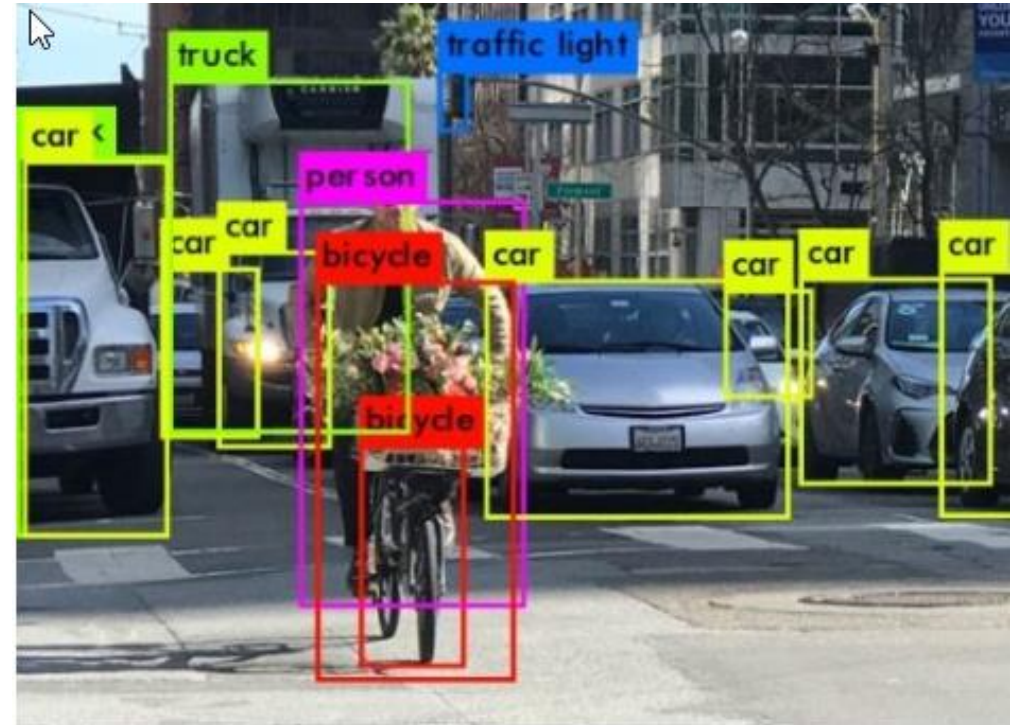
La localisation

- Qu'est-ce qu'il y a dans l'image et où?
- Déterminer la probabilité que l'objet dans l'image appartienne à différentes catégories (chien, chat, ...) et où se trouve l'objet dans l'image



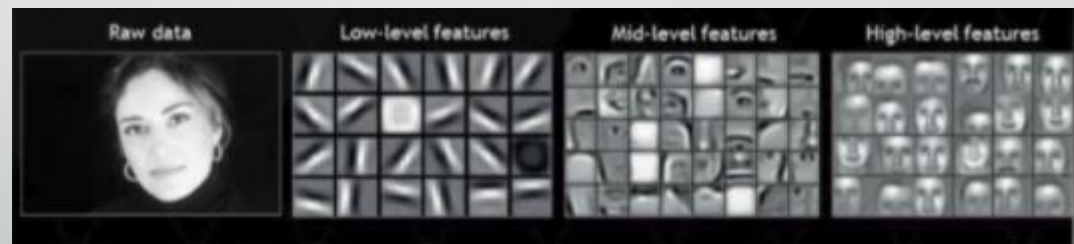
La détection d'objets

- Quels sont les objets dans l'image et où sont-ils?
- Classification d'objets multiples et leur position.
- On peut afficher un cadre autour de chaque objet que l'on appelle « bounding box »



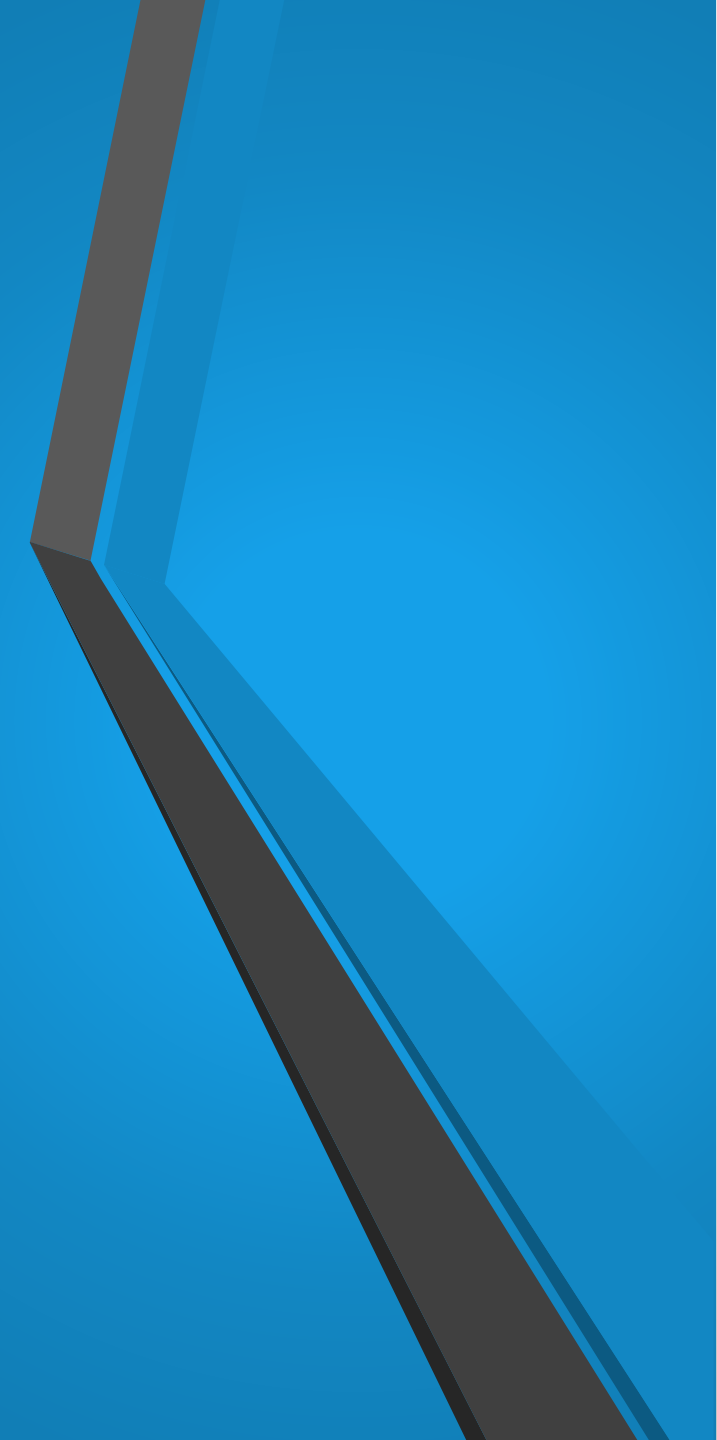
Caractéristiques ou « features »

- Chaque objet peut être classifié grâce à des caractéristiques ou « features »
- Ces caractéristiques sont extraites de l'image et données à des algorithmes pour la localisation et la classification



- **Détection d'objets à base de "machine learning"**
- [Viola-Jones object detection](#)
- [SIFT \(Scale-invariant feature transform\)](#)
- [HOG \(Histogram of oriented gradients\)](#)
- **Détection d'objets à base de "deep learning"**
- [R-CNN](#)
- [You Only Look Once\(YOLO\)](#)
- [Single Shot MultiBox Detector \(SSD\)](#)

Quelques
méthodes de
détection
d'objets



Pourquoi la détection
d'objets?

Quelques exemples d'application

- La reconnaissance d'écriture: OCR [Optical Character recognition](#)
- La détection de visage: [face detection](#)
- La robotique: prendre/transporter, trier des objets: [bin picking](#)
- Le suivi et le comptage d'objet/personne: [video tracking](#)
- La conduite autonome: [self driving](#)



Qu'est-ce que Yolo?

Yolo: c'est aussi cela mais ce n'est pas notre
sujet 😊

- YOLO est un acronyme, cela signifie You Only Live Once « on ne vit qu'une fois»
- Une chanson de Maître Gims: [Yolo Clip Officiel](#)

Yolo: You Look Only Once

- Parmi les algorithmes de détection d'objet avec des réseaux neuronaux, il y a 2 groupes:
 - Les algos de **classification**: il y a 2 étapes principales.
 - La sélection de zones ([Region of Interest \(ROI\)](#)) où la probabilité d'avoir un objet est grande
 - On applique chaque zone en entrée du réseau de convolution ([Convolution Neural Network](#)) pour détecter/classifier l'objet en question.
 - Un exemple est le [R-CNN](#)

Yolo: You Look Only Once

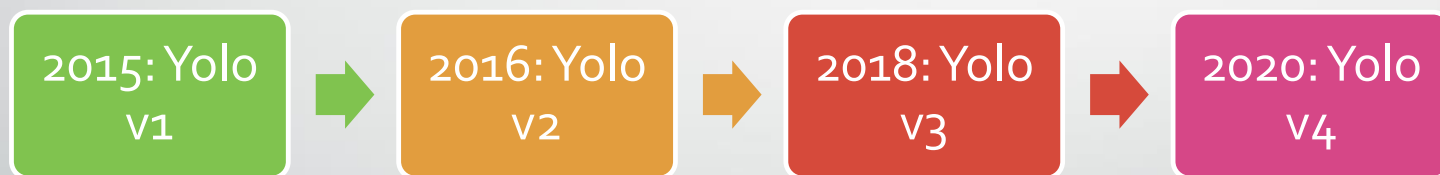
- Les algos de **régression**:
 - Au lieu de sélectionner des régions, on va fournir l'image entière au réseau. Le réseau se chargera de déterminer les « bounding box » et la probabilité de la catégorie des objets.
 - Ceci permet une détection plus rapide des objets.
 - Yolo est un de ces algos de régression.



Les différentes versions de Yolo

Historique

- Créateur: [Joseph Chet Redmon](#)



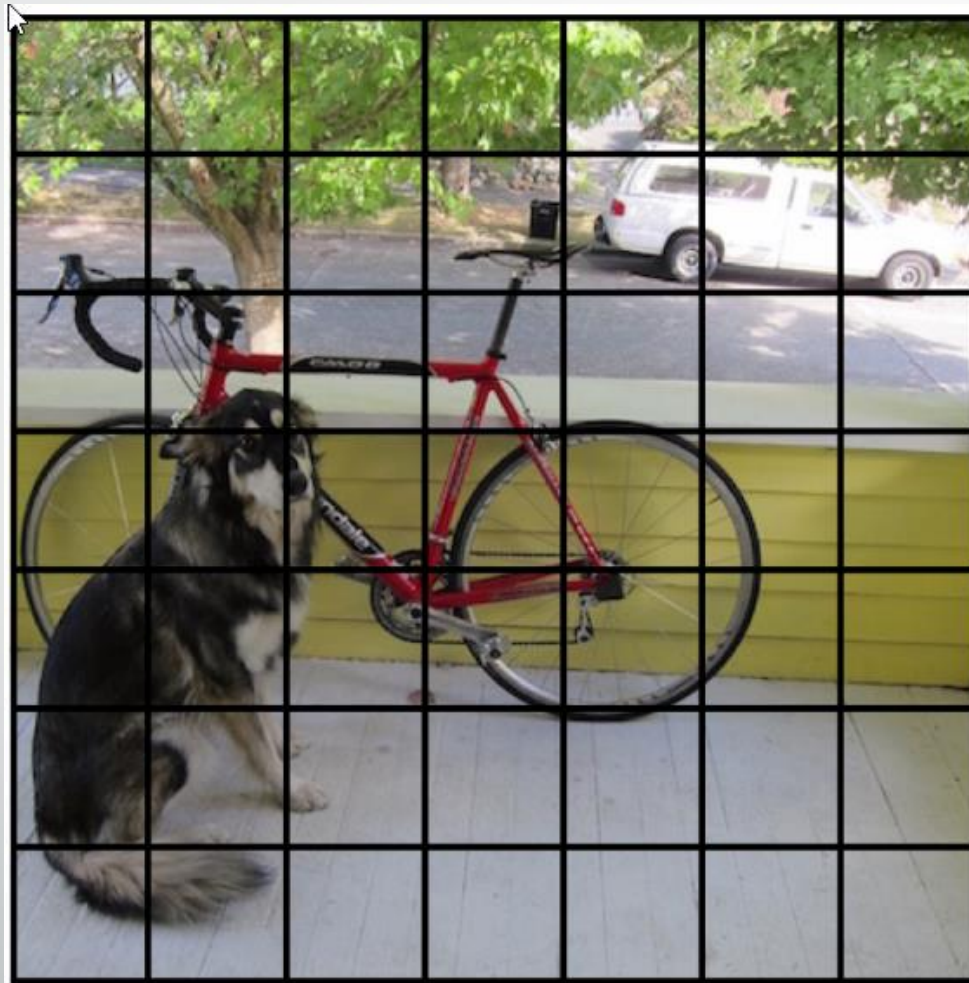


Yolo v1

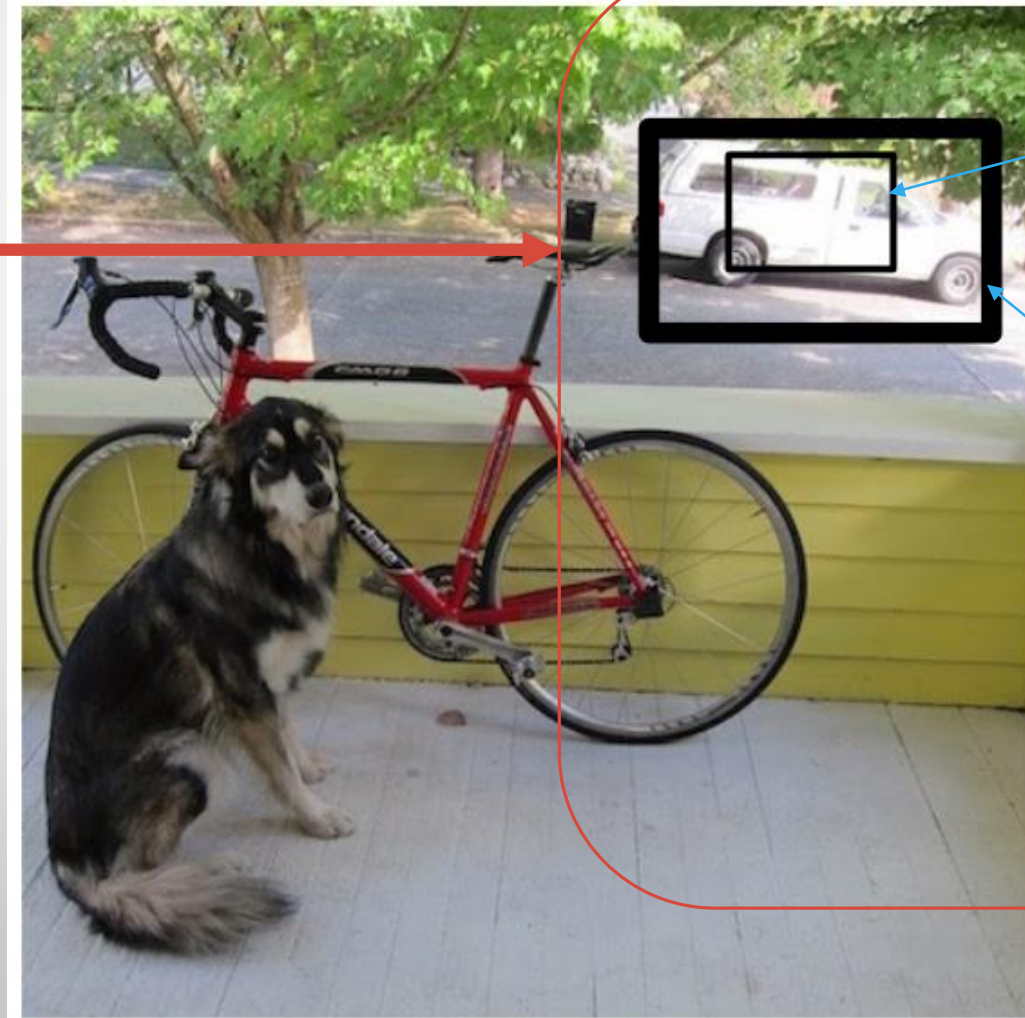
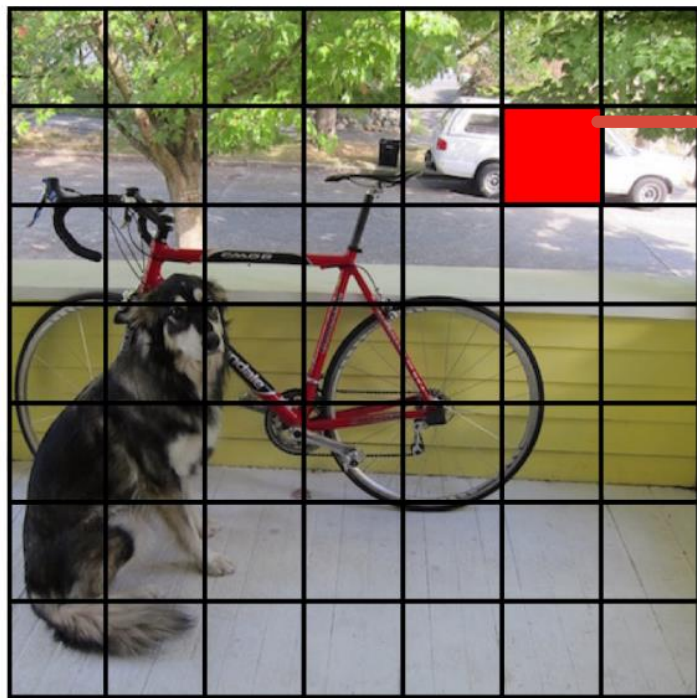
Les principes de la détection

- Source: présentation [YOLO CVPR 2016](#)

Découpage de l'image pour créer une grille



Pour chaque cellule de la grille, l'algo fait une prédiction des cadres « bounding boxes » et la confiance dans cette prédiction (1)

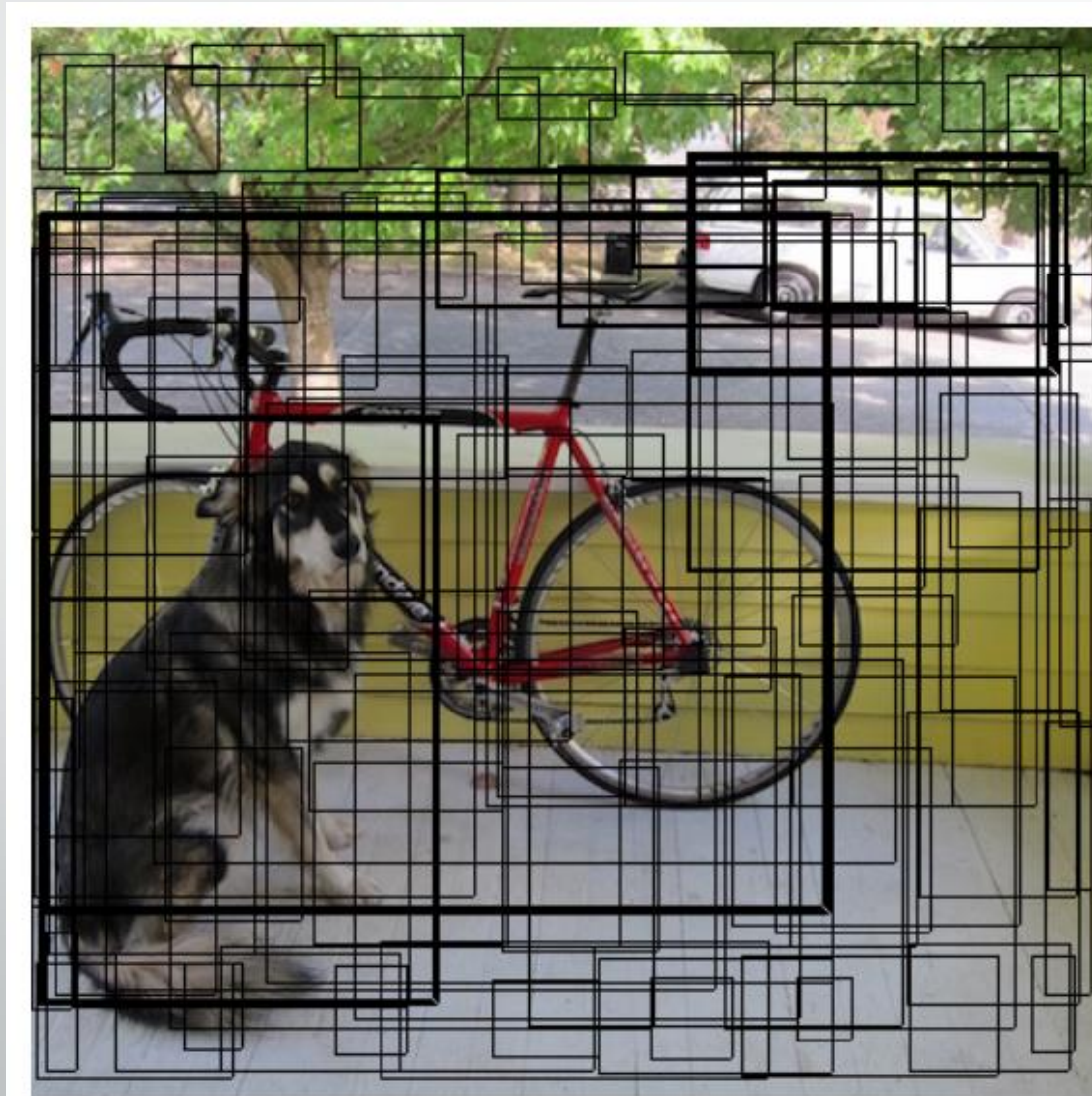


Box 1
Coordonnées1:
 x_1, y_1, h_1, w_1
Confiance-1

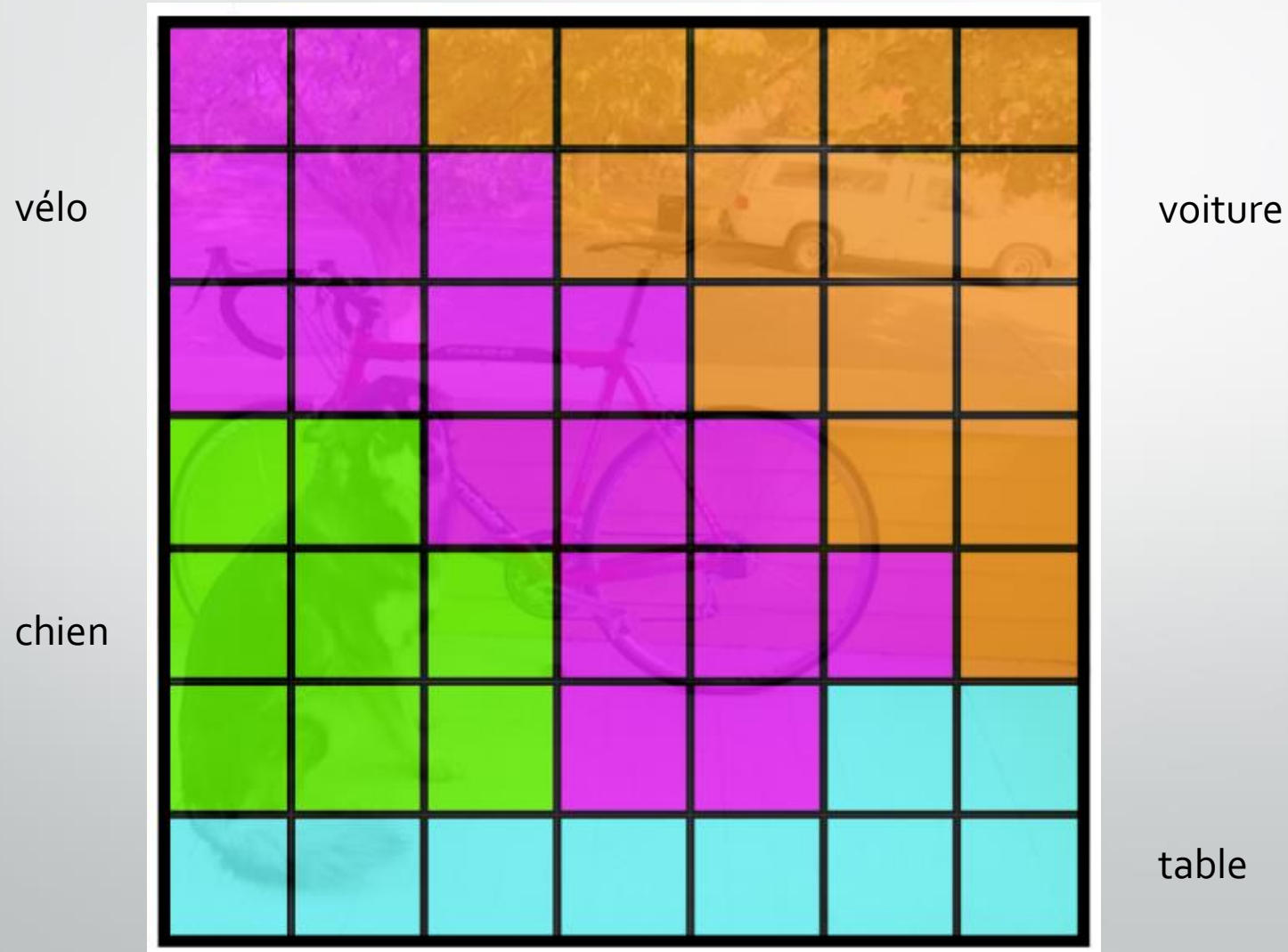
Box 2
Coordonnées2:
 x_2, y_2, h_2, w_2
Confiance-2

On considère ici 2 « boxes » par cellule $S = 2$

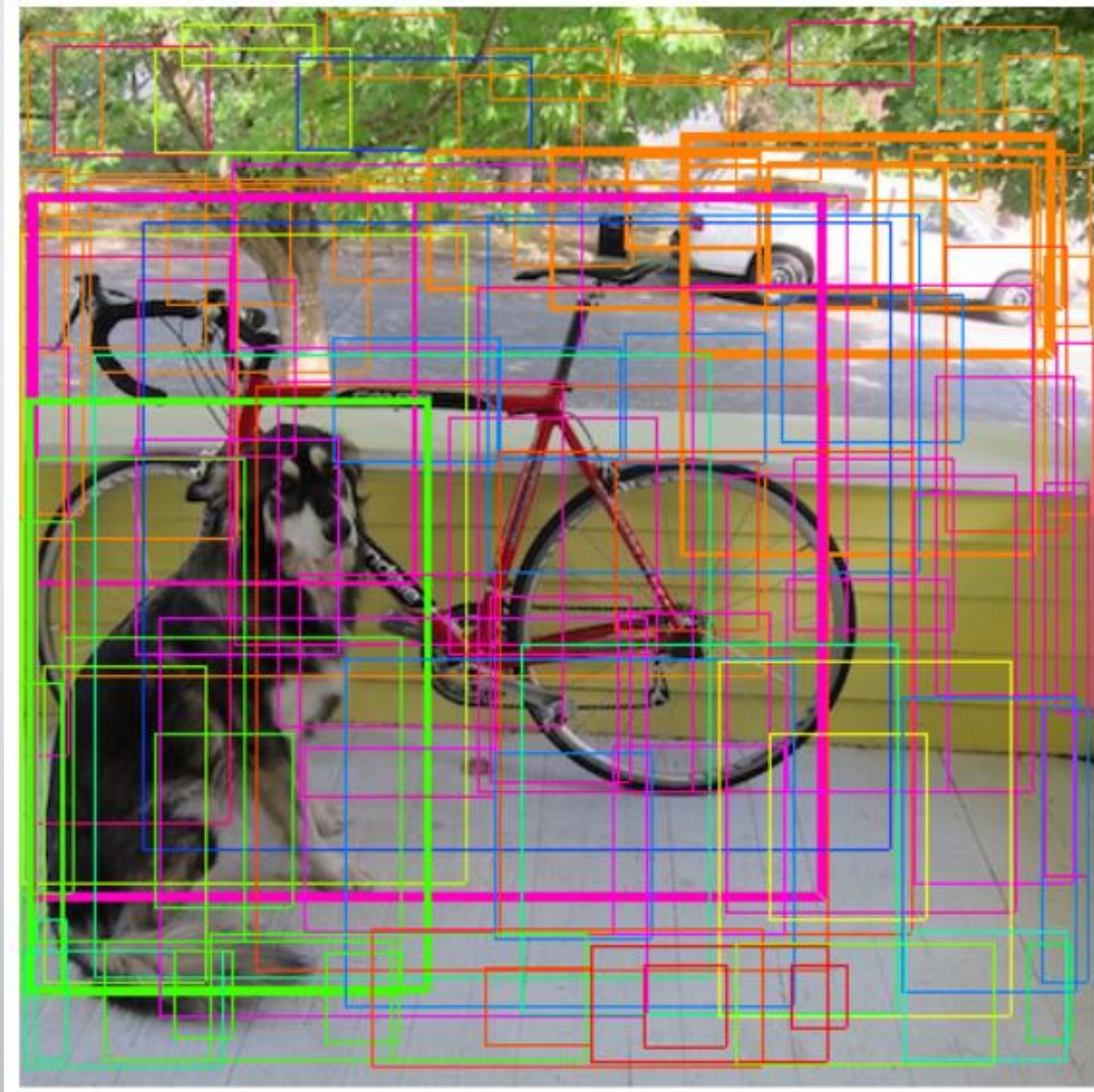
Si on illustre toutes les « bounding boxes » trouvées sur toutes les cellules



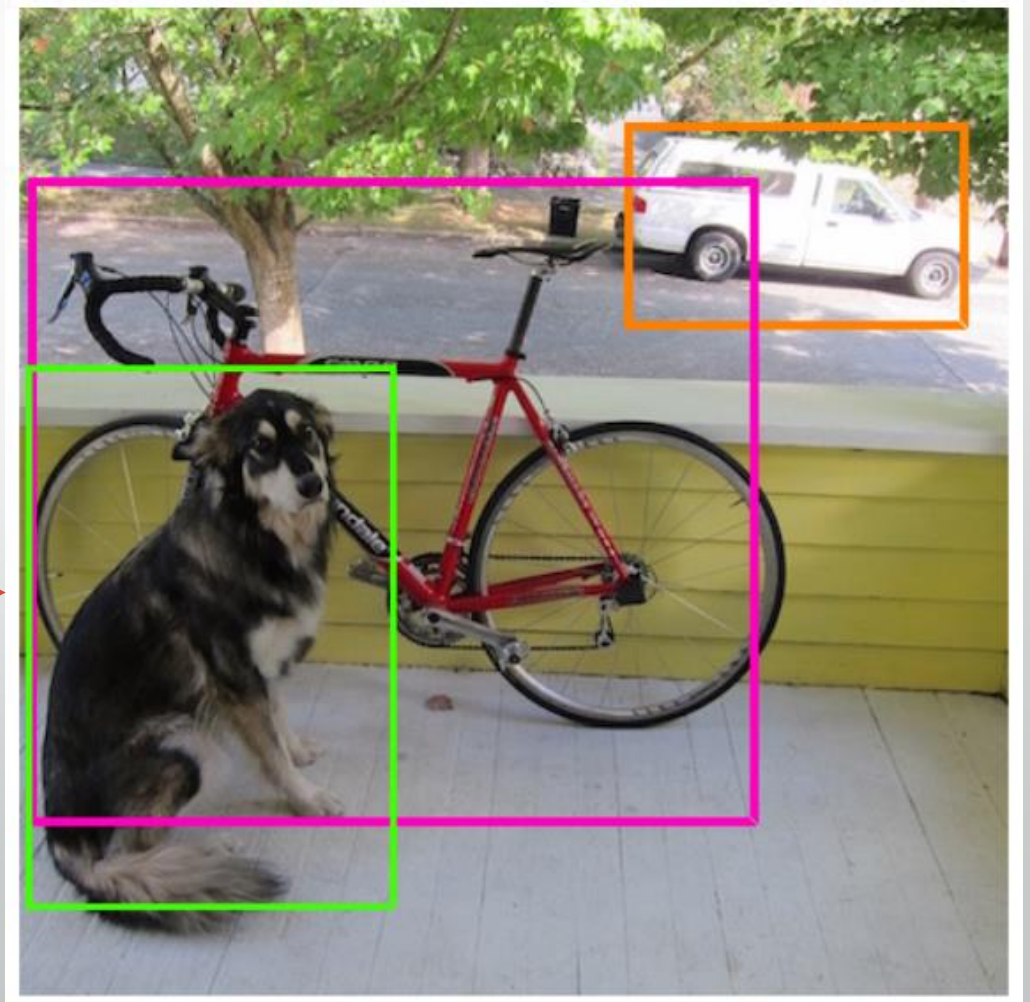
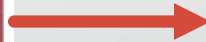
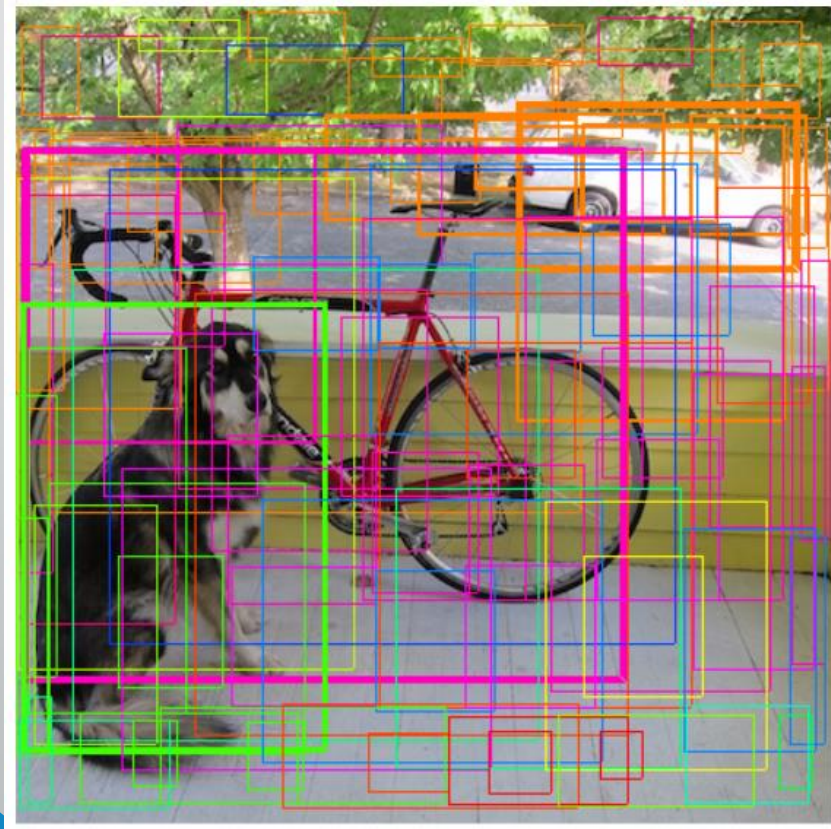
Pour chaque cellule une probabilité pour chaque classe (chien, chat) est prédite. On représente ici seulement la classe dont la proba est la plus grande



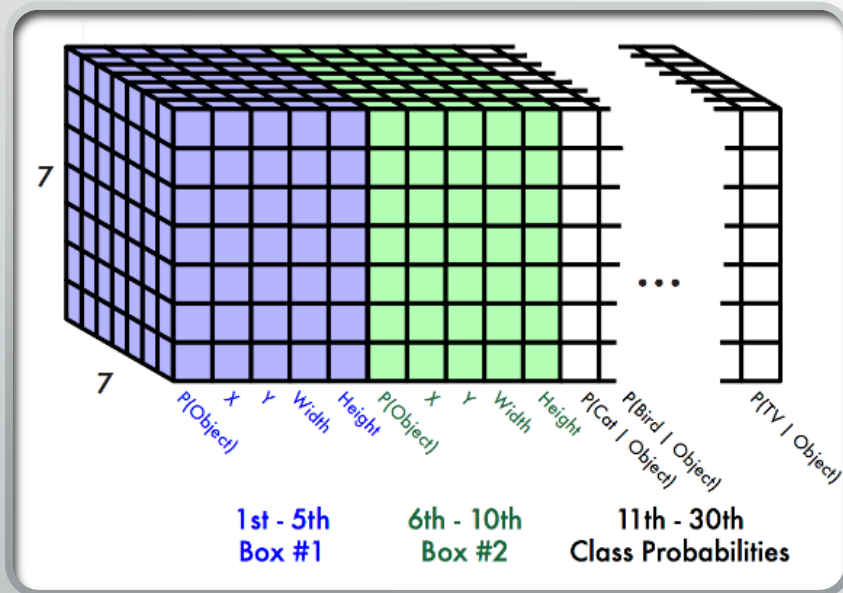
En associant les « boxes » et les classes



On élimine les « boxes » dont la confiance est inférieure à un seuil (threshold detection) et on applique la suppression des « boxes » qui se chevauchent (Non Max suppression, NMS)



Nombre de sorties du réseau



- Pour une grille de 7x7 cellules, 2 « bounding boxes » par cellule et 20 classes
- Pour chaque cellule on a
 - Pour chaque box: 4 valeurs de coordonnées (x, y, h, w) et 1 valeur de confiance pour chaque « box »
=> 5 valeurs
 - Pour chaque classe: 1 valeur de probabilité.
- On obtient $7 \times 7 \times ((2 \times 5) + 20) = 1470$ sorties

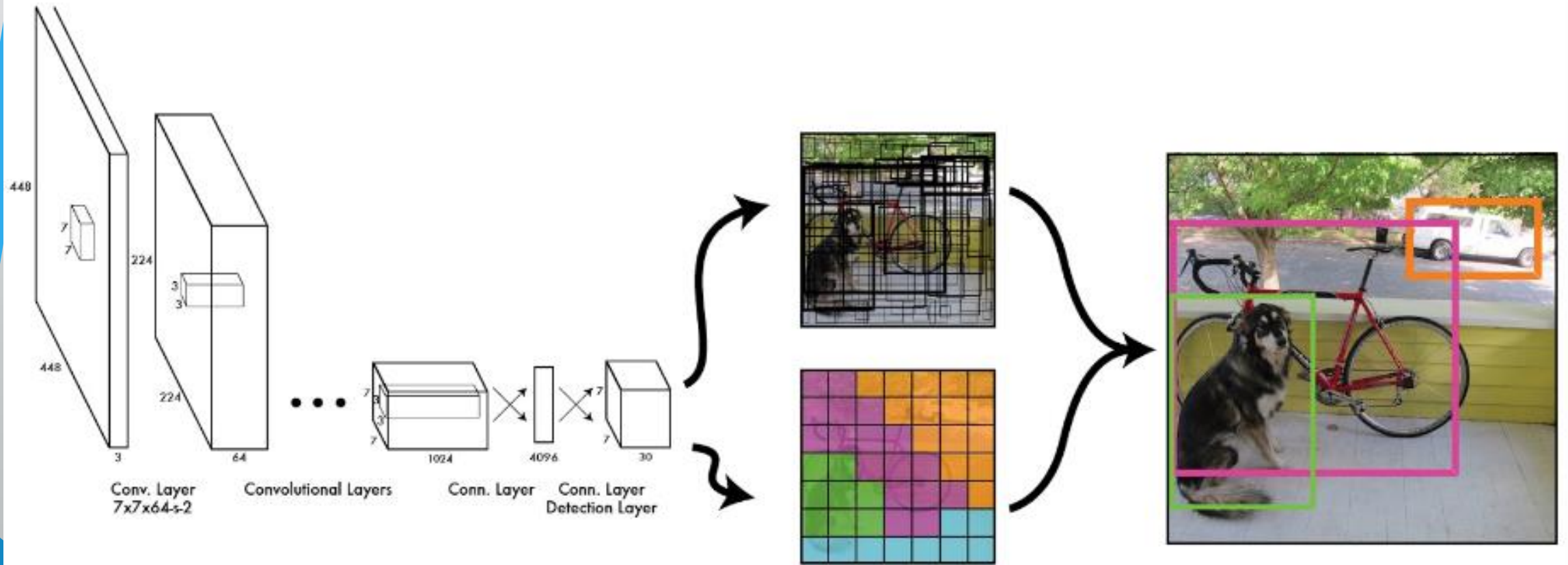
Quelques détails : coordonnées et confiance, quelle cellule doit détecter l'objet?

- la cellule qui est chargée de détecter un objet est la cellule qui est au centre de l'objet.
- Pour chaque "box":
 - x et y sont les coordonnées du centre de la "box" (valeurs entre 0 et 1)
 - Hauteur et largeur (h, w) entre 0 et 1 par rapport à la taille de l'image
 - La confiance est la probabilité que l'on ai un objet dans la « box » * « l'IOU » (intersection entre la région prédite et la région réelle de l'objet)



Exemple « d'IoU »

Le chemin complet de détection



Quelques résultats

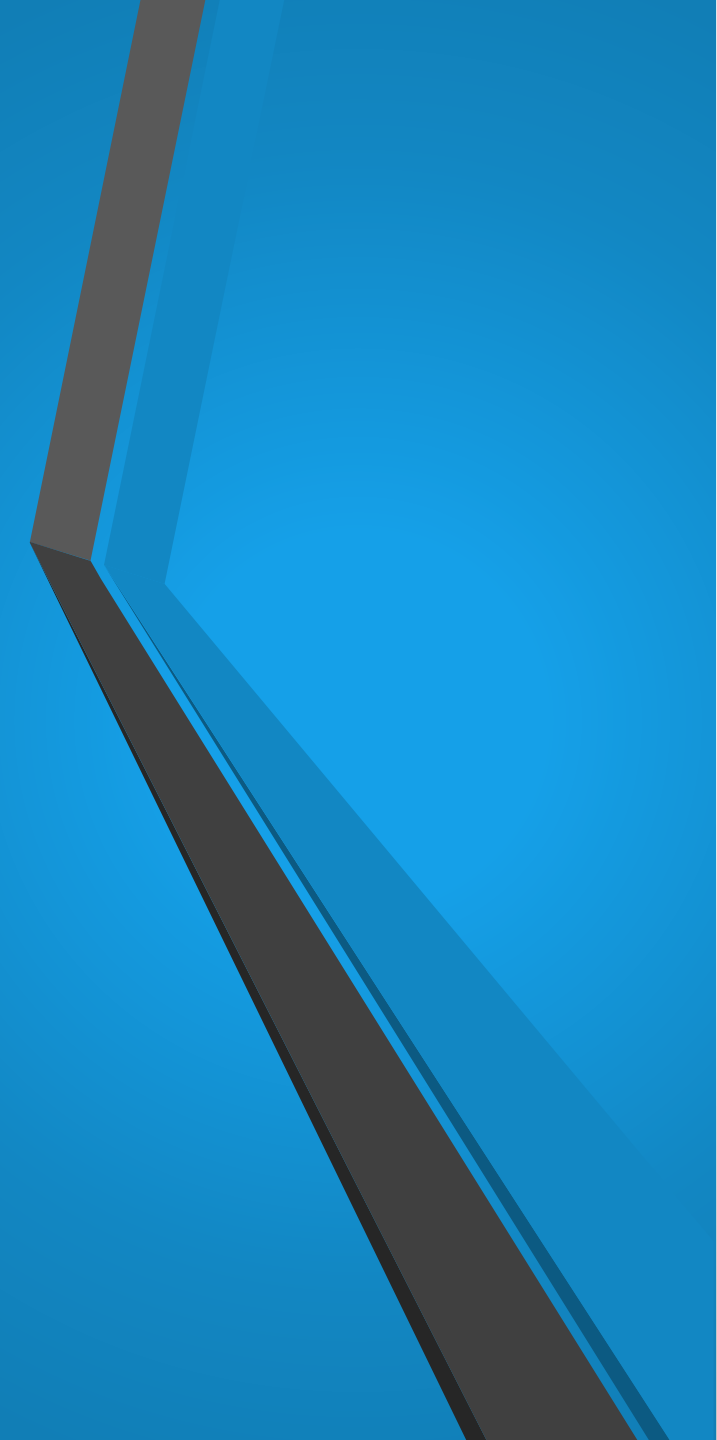
Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO γ	2007+2012	52.7	155
YOLO	2007+2012	63.4	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

FPS: frame per second/image par seconde
mAP: (mean Average) Precision

Yolo v1 se concentre sur la rapidité plutôt que la précision, tout en gardant une précision correcte


Les principaux inconvénients

- 1 seul objet détecté par cellule => difficile de détecter les petits objets
- La taille et le ratio hauteur/largeur des « bounding box » étant appris à partir des données, le modèle n'arrive pas à généraliser sur des tailles d'objets de ratios différents de ceux appris.
- Le modèle génère des erreurs de localisation supérieures aux autres modèles



Yolo v2: « Better, Faster, Stronger »

Source: présentation [YOLO CVPR 2017](#)



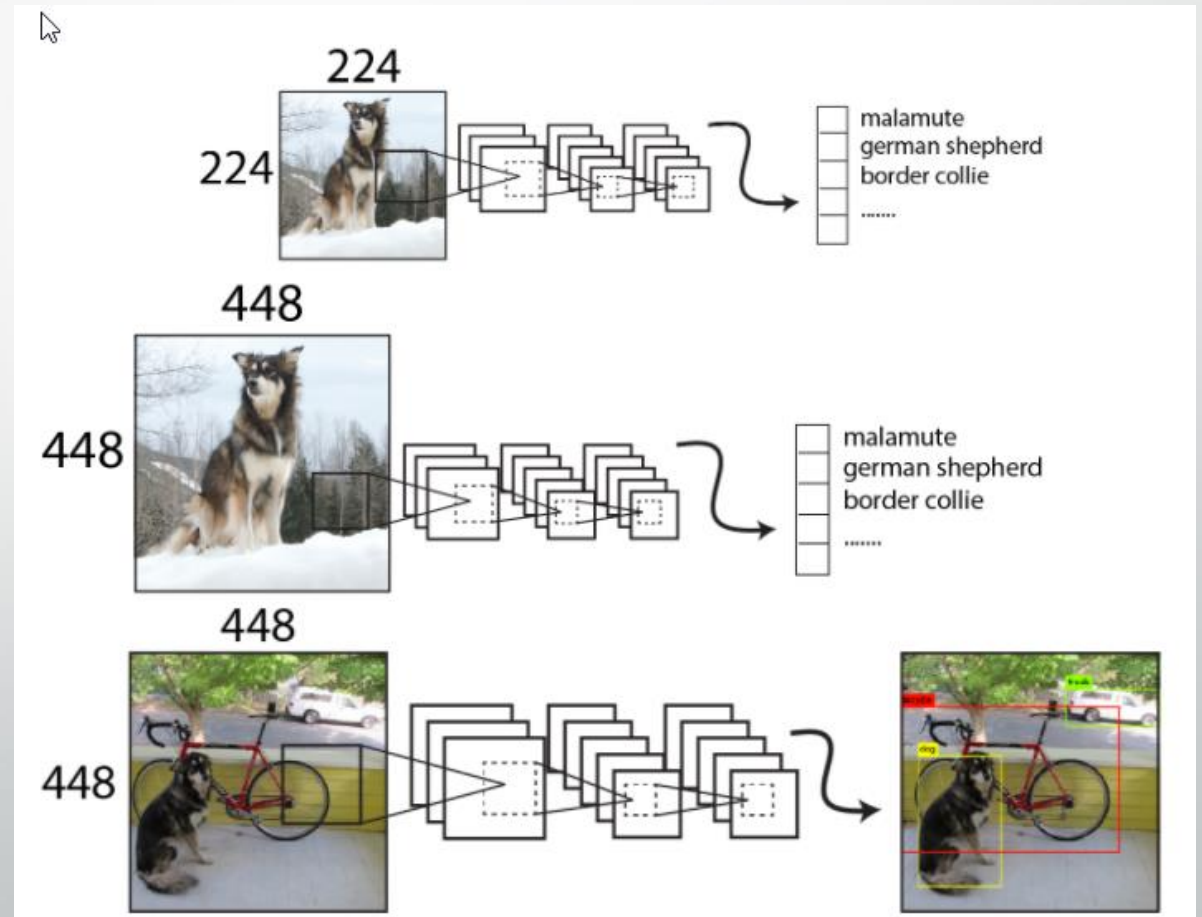
« Better »: Meilleure
précision

Affiner les poids des filtres de convolution via la classification (+3,5% mAP)

Entrainement avec ImageNet image 224×224

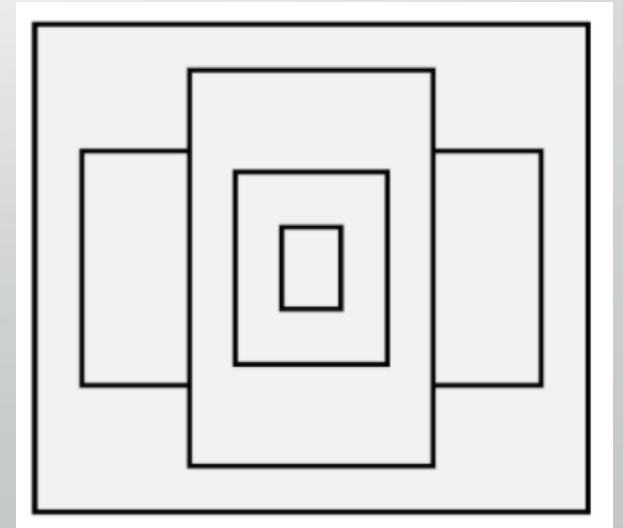
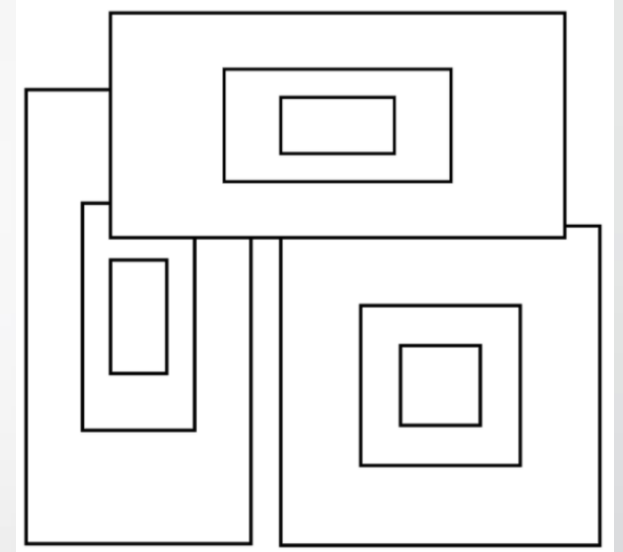
Entrainement supplémentaire des images agrandie en 448×448

Utilisation de l'entraînement 448×448
En entrée du réseau Yolo v1



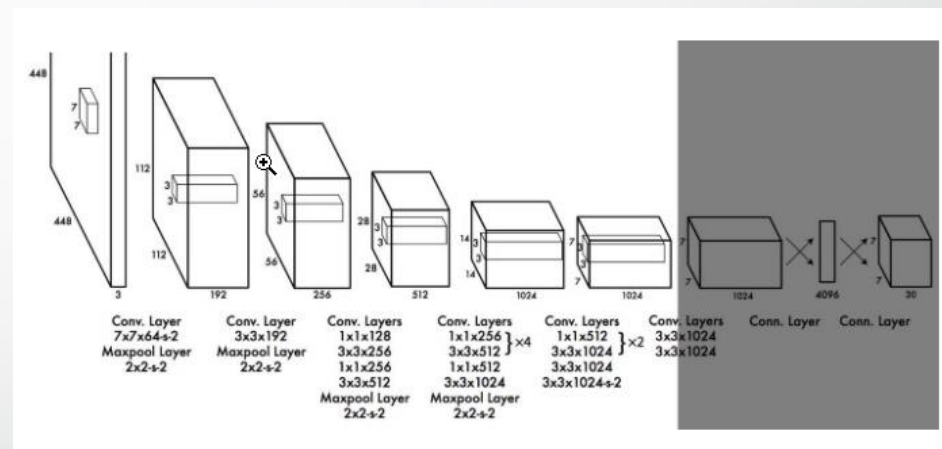
Utilisation « d'anchor box » avec des tailles prédéfinies au lieu de « box » avec des tailles calculées par la prédiction (+5% mAP)

- Pour la détection d'objet, d'autres algos de détection utilisent des « box » avec des tailles prédéfinies (x9) nommées « anchor box » qui représentent les ratios « classiques » des objets.
- Pour Yolo v2, après avoir utilisé l'algorithme k-mean cluster sur les « box » des images d'entraînement, il a été démontré que **5 « Boxes »** suffisent pour obtenir des résultats aussi bons que les 9 habituelles => « 5 Boxes » par cellule au lieu de 2 précédemment.



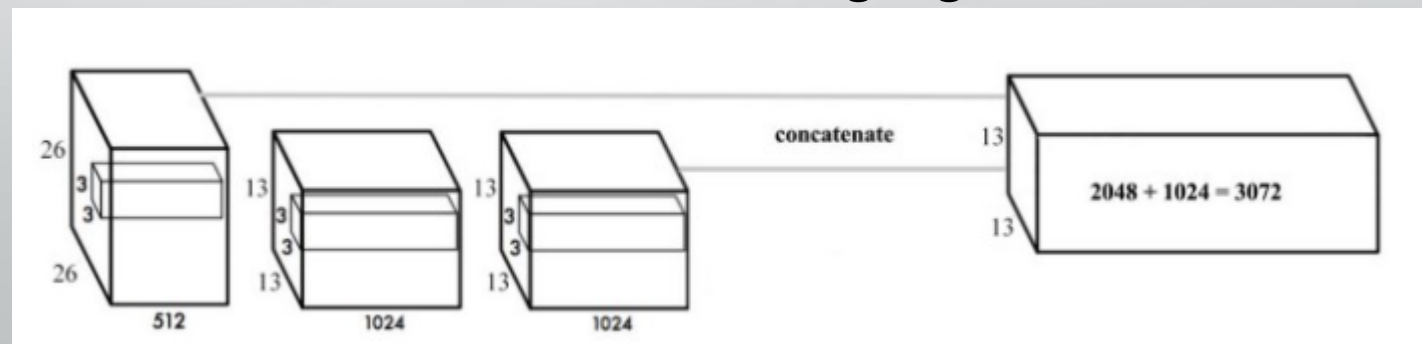
Evolution du réseau suite au passage aux « anchors box »

- Les dernières couches du réseau « **connected layers** » qui servaient à la prédiction des « bounding boxes » sont **supprimées**. Il ne reste que des couches de « convolution/pooling »
- La **prédiction des classes** est faite pour **chaque** « anchor box » au lieu de chaque cellule



Evolution du réseau (suite)

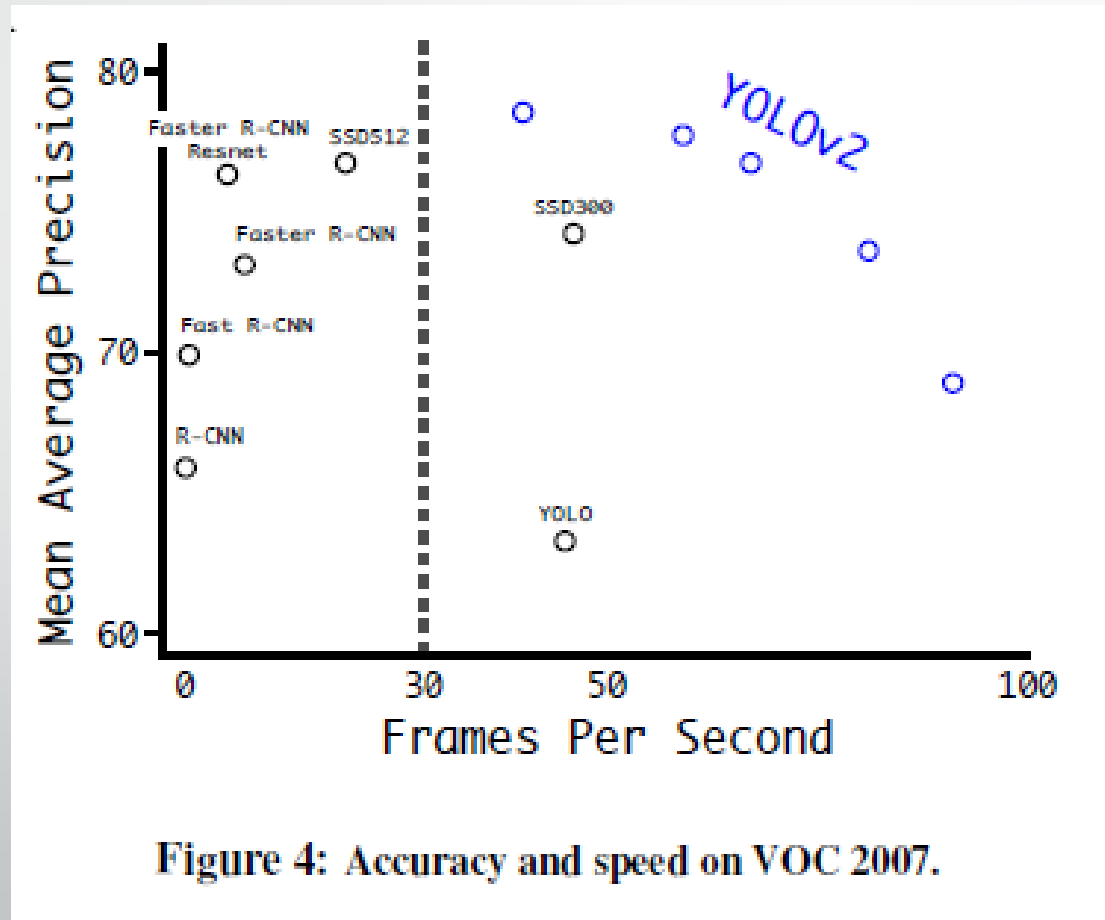
- La **dernière couche de « pooling » est enlevée** pour avoir une plus grande finesse (« resolution »), ce qui nous donne une couche pour les caractéristiques (features map) de dimension $14*14*1024$ (au lieu de $7*7*1024$).
- Le réseau divise par 32 les dimensions de l'image d'entrée. Pour avoir un nombre impair de localisation (13) dans la « feature map », **l'image d'entrée est redimensionnée à $(13*32) \times (13*32) = 416 \times 416$ (au lieu de 448×448)**
- Afin de mieux détecter les petits objets, une concaténation est faite entre les « features » de la couche $26*26$ et la couche $13*13$



Entrainement avec plusieurs taille d'image

- Maintenant que le réseau ne contient que des couches de « convolution/pooling », il est possible **d'entraîner le réseau avec différentes tailles** d'image tant qu'elles sont multiples de 32.
 - exemples: 320x320 ce qui donnera en sortie 10x10, 352x352 pour une sortie 11x11, ...
- Le **réseau** ainsi entraîné avec des tailles d'images qui changent tous les n « batches », pourra être **utilisé pour ces différentes tailles d'image**. On peut ainsi jouer sur la taille des images pour soit avoir plus de précision, soit avoir une meilleure performance (image par seconde).

Quelques résultats





« Faster »: plus rapide

De googlenet à darknet-19

- YOLO-v1 utilise un réseau de convolution basé sur googlenet.
- Darknet-19 est un nouveau réseau qui demande « seulement » 5.58 milliards d'opérations au lieu de 8,52 milliards avec Yolo-v1 pour une image de 224x224.

La vitesse n'est pas le nombre de FLOPs*


	Top 1	Top 5	FLOPs	GPU Speed
VGG-16	70.5	90.0	30.95 Bn	100 FPS
Extraction (YOLOv1)	72.5	90.8	8.52 Bn	200 FPS
Resnet50	75.3	92.2	7.66 Bn	90 FPS
Darknet19	74.0	91.8	5.58 Bn	200 FPS

Le FLOPs est important pour la vitesse mais le nombre de données transmises entre les couches demandant beaucoup d'accès à la mémoire joue aussi un grand rôle.

Ainsi, on obtient une performance moins bonne avec Resnet50 que YOLOv1

* FLOPs = Floating operation (nombre d'opération flottante)

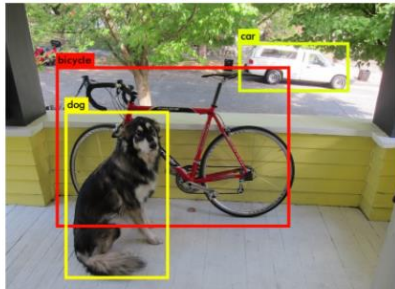
« Stronger »: plus d'objets
différents détectés (plus
de classes)



Données de détection et classification



- 100k images
- 80 classes
- Detection labels



- 14 million images
- 22k classes
- Classification labels



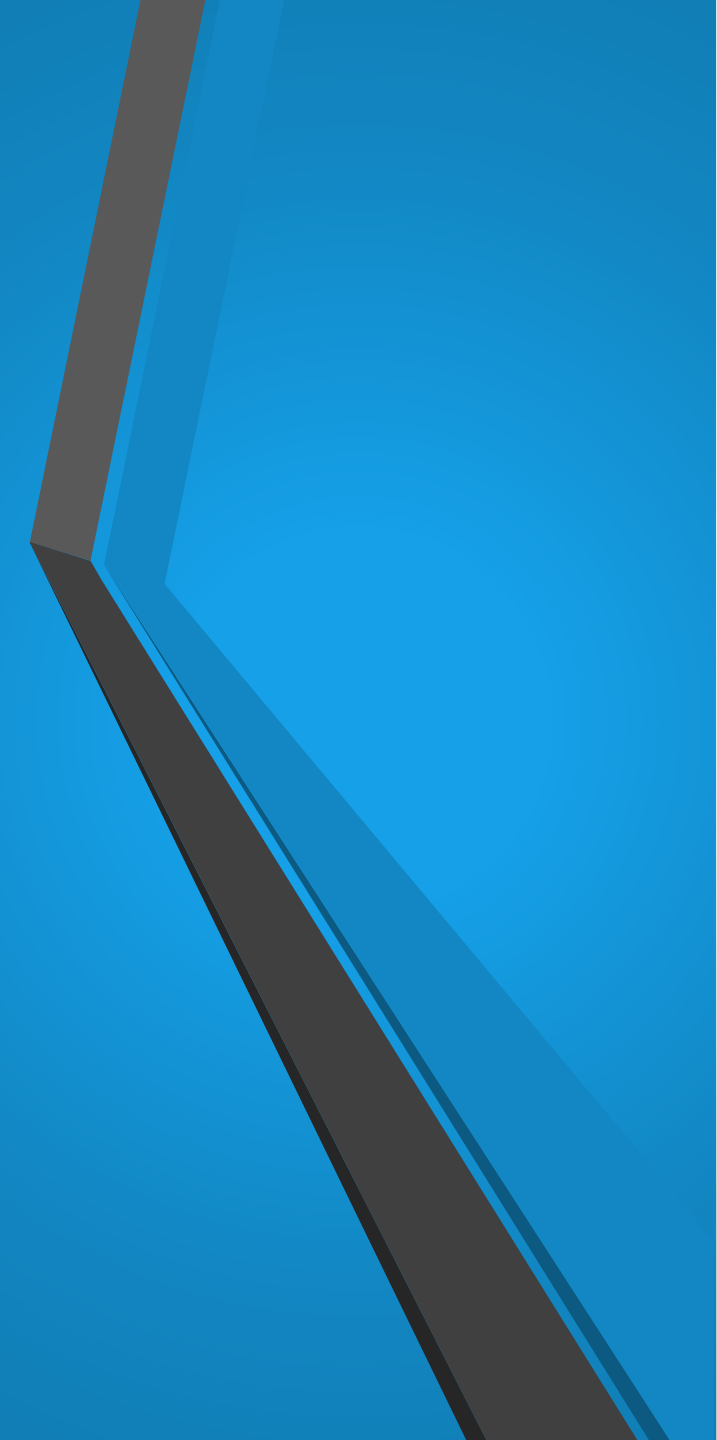
- Les bases de données pour la **classification** des objets sont très importantes. On peut différencier des **milliers de classes**.
- Les bases de données pour la **détection** sont bien moins fournies (**quelques dizaines de classe**) car elles nécessitent de rajouter la localisation de chaque objet dans l'image.
- C'est pourquoi les auteurs de Yolo propose une méthode pour entrainer le réseau en **combinant ces bases de données de classification et de détection**.

Comment utiliser Yolo-v2

- Suivre le lien suivant: [Yolo-v2](#)

Vidéo de démo

- <https://www.youtube.com/watch?v=VOC3huqHrss4>



Yolo v3:
« an incremental
improvement »

Objectif de Yolo v3

- L'objectif de Yolo-v3 est d'amener **quelques améliorations** pour améliorer la **précision** tout en gardant un réseau suffisamment performant pour du « real-time »

Evolution du réseau (1)

- Les **couches de « pooling »** sont **remplacées** par des couches de convolution => éviter de perdre les « low level feature »/caractéristiques obtenues par les premières couche de convolution et ainsi mieux détecter les petits objets.

Détection à 3 tailles différentes

- Le nouveau **réseau** darknet-53 est constitué de **53 couches**. On divise toujours au final taille /32 ($13*13$). Mais on fait la détection également sur des couches intermédiaires à /16 ($26*26$) et /8 ($52*52$).
 - /32 pour les gros objets
 - /16 pour les objets de moyenne taille
 - /8 pour les petits objets

Détection à 3 tailles différentes (2)

- **3 « Bounding box » par cellule** (au lieu de 5 précédemment)
 - $13*13*3 = 507$ "box" pour taille /32
 - $26*26*3 = 2028$ "box" pour taille /16
 - $52*52*3 = 8112$ "box" pour la taille /8

On a donc **10647 "box" prédites pour l'image**

Résultats

Backbone	Top-1	Top-5	Bn Ops	BFLOP/s	FPS
Darknet-19 [15]	74.1	91.8	7.29	1246	171
ResNet-101[5]	77.1	93.7	19.7	1039	53
ResNet-152 [5]	77.6	93.8	29.4	1090	37
Darknet-53	77.2	93.8	18.7	1457	78

	backbone	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L
<i>Two-stage methods</i>							
Faster R-CNN+++ [5]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [8]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [6]	Inception-ResNet-v2 [21]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [20]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	52.1
<i>One-stage methods</i>							
YOLOv2 [15]	DarkNet-19 [15]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [11, 3]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [3]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [9]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet [9]	ResNeXt-101-FPN	40.8	61.1	44.1	24.1	44.2	51.2
YOLOv3 608 × 608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9

Comment utiliser Yolo-v3

- Suivre le lien suivant: [Yolo-v3](#)



Yolo v4:

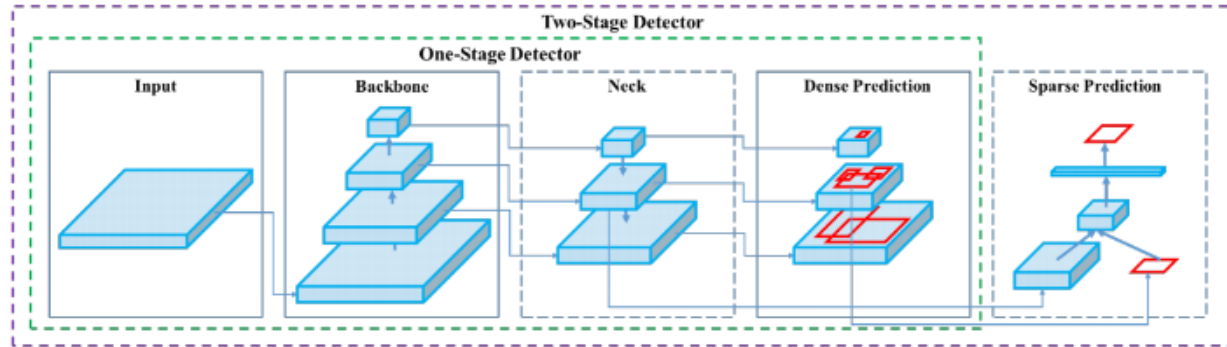
Source: [Yolo v4 presentation](#)

Des nouveaux auteurs?

- L'auteur de Yolo [Joseph Chet Redmon](#) arrête ses recherches sur le « computing vision »:

“I stopped doing CV research because I saw the impact my work was having. I loved the work but the military applications and privacy concerns eventually became impossible to ignore”

Architecture d'un réseau de détection



Backbone: couches permettant la création des caractéristiques/ « features »

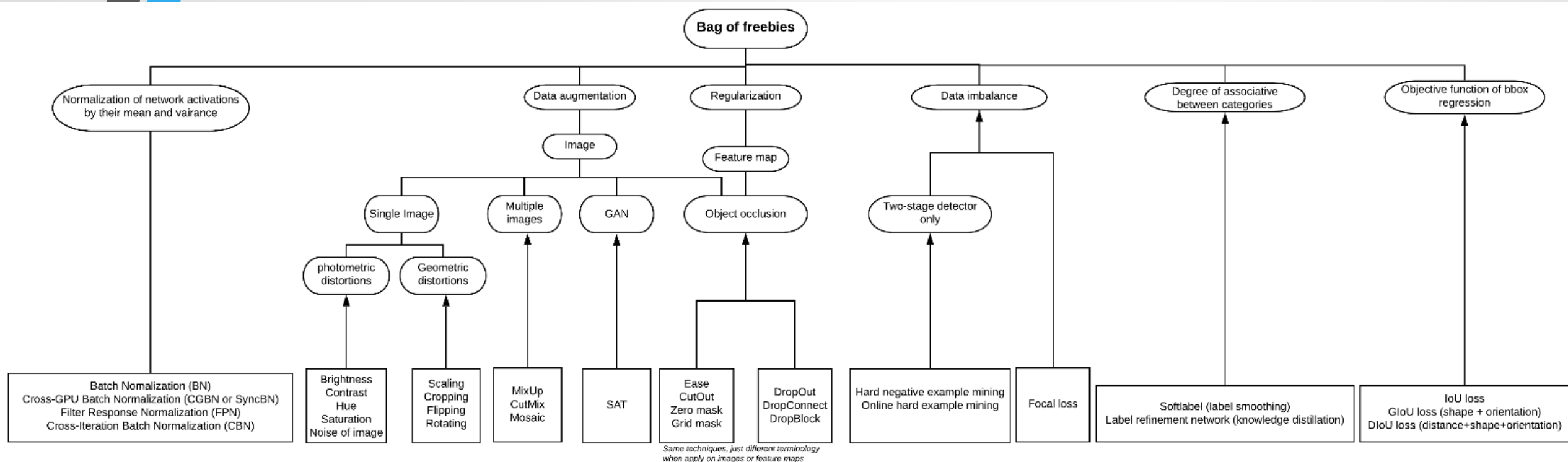
Neck: couches qui transmettent et combinent les « features » pour les donner à la/aux couches de prédiction

Head: couches qui prédisent les « box » et les classes

- **Input:** Image, Patches, Image Pyramid
- **Backbones:** VGG16 [68], ResNet-50 [26], SpineNet [12], EfficientNet-B0/B7 [75], CSPResNeXt50 [81], CSPDarknet53 [81]
- **Neck:**
 - **Additional blocks:** SPP [25], ASPP [5], RFB [47], SAM [85]
 - **Path-aggregation blocks:** FPN [44], PAN [49], NAS-FPN [17], Fully-connected FPN, BiFPN [77], ASFF [48], SFAM [98]
- **Heads:**
 - **Dense Prediction (one-stage):**
 - RPN [64], SSD [50], YOLO [61], RetinaNet [45] (anchor based)
 - CornerNet [37], CenterNet [13], MatrixNet [60], FCOS [78] (anchor free)
 - **Sparse Prediction (two-stage):**
 - Faster R-CNN [64], R-FCN [9], Mask R-CNN [23] (anchor based)
 - RepPoints [87] (anchor free)

« Bag of freebies »

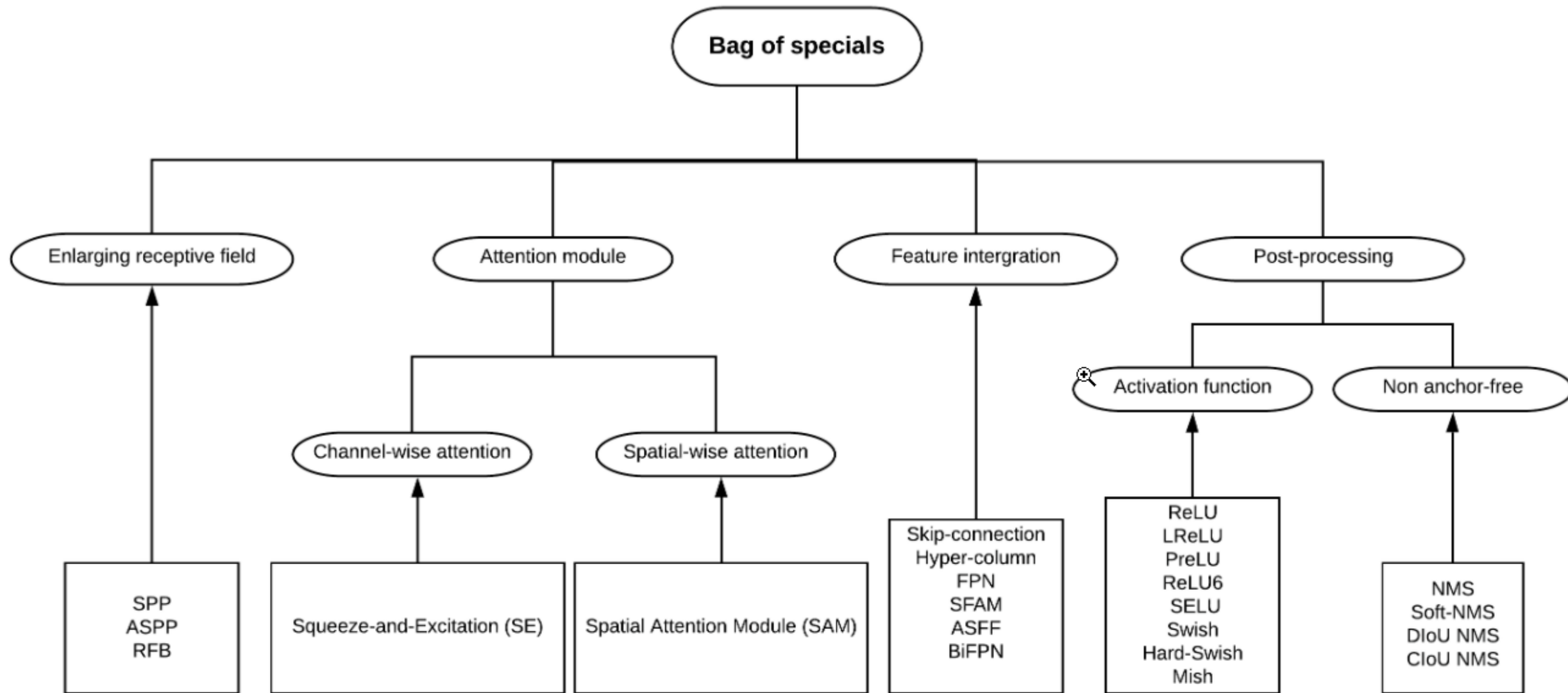
- « Bag of freebies »: ce sont les outils qui permettent de **changer l'entraînement sans impacter la durée de « l'inference »** (une prédiction hors entraînement) afin d'avoir de **meilleures précisions** dans les prédictions.



Source: [bag of freebies](#)

« Bag of specials »

- « Bag of specials »: ce sont les outils qui vont **changer « légèrement » la durée de « l'inference »** mais permettent d'avoir une **meilleure précision** dans les prédictions.



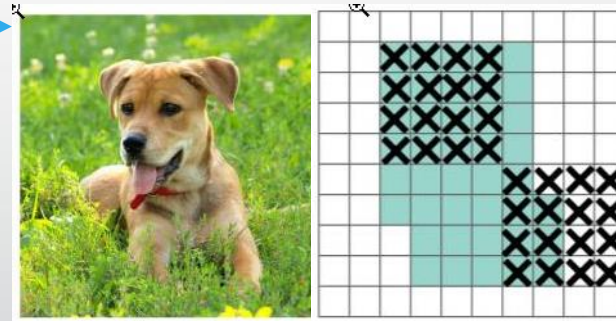
architecture de Yolo v4

- Backbone: CSPDarknet53
- Neck: SPP (Spatial Pyramid Pooling) + PANet (Path Aggregation Network)
- Head: YOLOv3

architecture de Yolo v4 (2)

- « Bag of freebies » pour le « Backbone »:

- CutMix data augmentation:
- Mosaic data augmentation:
- DropBlock regularization:



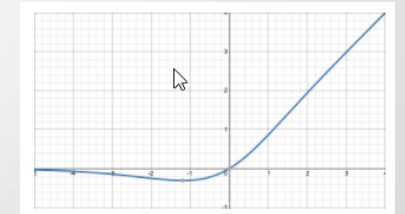
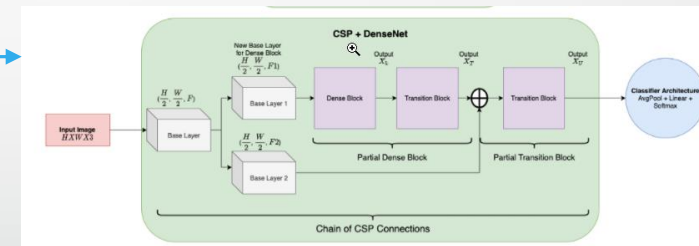
- Class label smoothing: ajouter du bruit sur le label / la classe $[0,1]$ devient $[0.5,0.95]$

[Bag of freebies examples](#)

architecture de Yolo v4 (2)

- « Bag of specials » pour le « Backbone »:

- Mish activation
- Cross-stage partial connections (CSP)



- Multi-input weighted residual connections (MiWRC): transmettre les poids d'une couche à une autre plus ou moins lointaine via des nouvelles connections entre les couches

[Bag of special](#)

architecture de Yolo v4 (2)

- « Bag of freebies » pour le « Head/detector »:
 - **CloU-loss (Complete-IoU)**: évolution de la technique de calcul de l'IoU
 - **CmBN (Cross mini-Batch Normalization)**: utiliser les variances et moyennes dans batch précédent pour mieux calculer estimer le batch actuel (être moins dépendant du biais introduit par la taille du batch actuel).
 - **DropBlock regularization**
 - **Mosaic data augmentation**
 - **Self-Adversarial Training**: 1/ ajouter du bruit dans l'image initiale pour que le réseau ne détecte plus l'objet. 2/ entraîner le réseau à détecter l'objet d'origine avec cette nouvelle image modifiée.

[Bag of freebies examples](#)

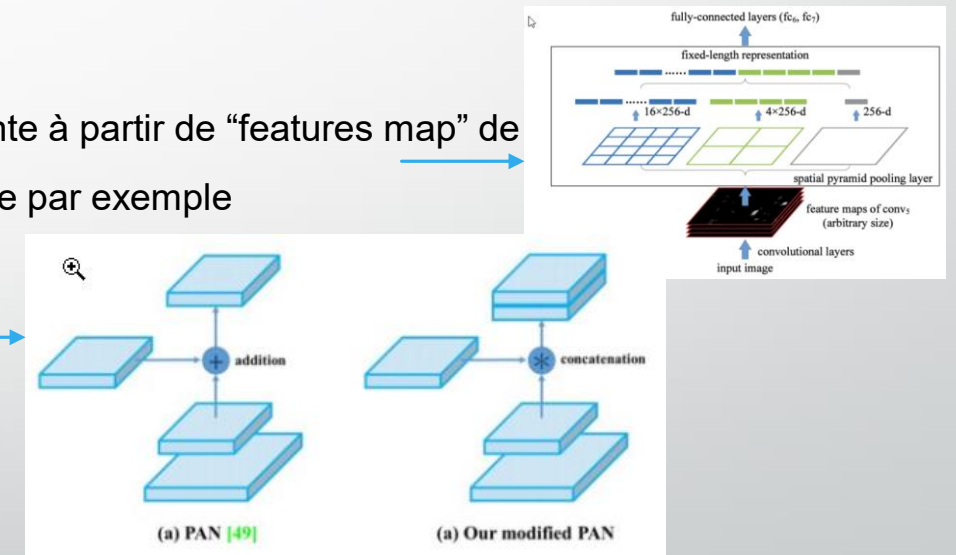
architecture de Yolo v4 (2)

- « Bag of freebies » pour le « Head/detector » (suite):
 - **Eliminate grid sensitivity**: amélioration de la formule de calcul des dimensions de la box.
 - **Using multiple anchors for a single ground truth**: c'est le fait d'avoir une multitude de box pour un objet donné
 - **Cosine annealing scheduler**: faire évoluer le learning rate de façon cyclique à l'aide de la fonction cosinus
 - **Optimal hyperparameters**: entrainer avec différents hyper paramètres et sélectionner les meilleurs
 - **Random training shapes**: le fait d'entrainer avec différentes taille d'images

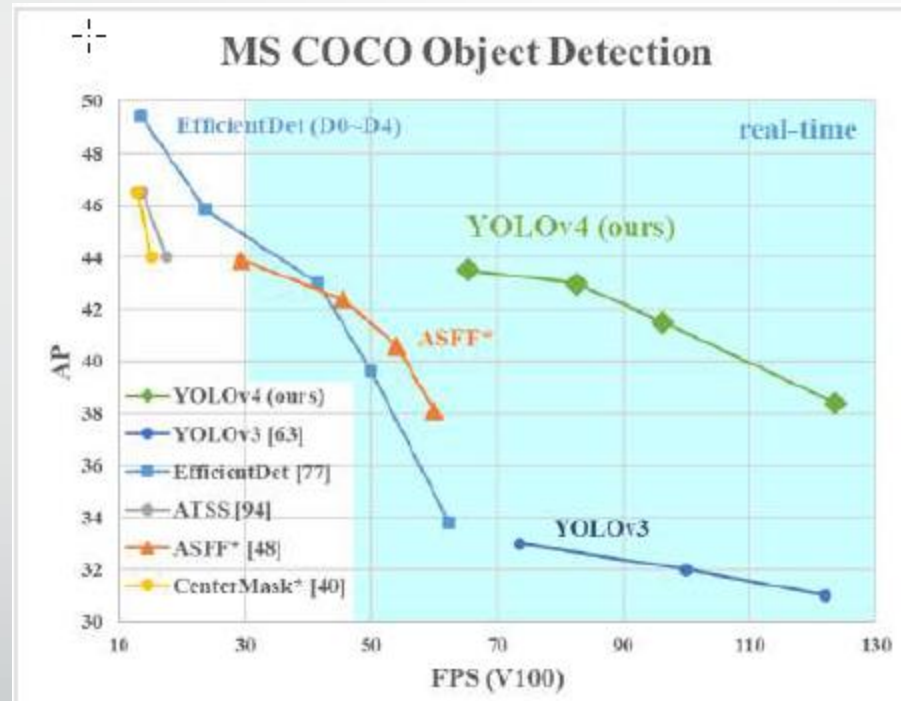
[Bag of freebies examples](#)

architecture de Yolo v4 (2)

- « Bag of specials » pour le « Head/detector »:
 - Mish activation,
 - **SPP-block**: générer des caractéristiques avec une taille constante à partir de “features map” de variable afin de l'utiliser en entrée d'une couche dense par exemple
 - **SAM-block**: augmenter les “features” les plus importantes
 - **PAN path-aggregation block**,
 - **DIoU-NMS**: évolution de la technique de suppression de box



Résultats



Démo Yolo v4 en entrée de Deepsort

- Voici un exemple de suivi d'objet « tracking » avec la sortie de Yolo v4 en entrée d'un algo de tracking DeepSort

<https://pythonawesome.com/object-tracking-implemented-with-yolov4-and-tensorflow/>

sources

- <https://robocademy.com/2020/05/01/a-gentle-introduction-to-yolo-v4-for-object-detection-in-ubuntu-20-04/>
- [Yolo papers v1->v4](#)
- [YOLO CVPR 2016](#)
- [YOLO CVPR 2017](#)
- <https://jonathan-hui.medium.com/real-time-object-detection-with-yolo-yolov2-28b1b93e2088>
- <https://www.youtube.com/watch?v=bDKgNRF2oTo>, [Yolo v4 presentation](#)
- [Bag of freebies examples](#)