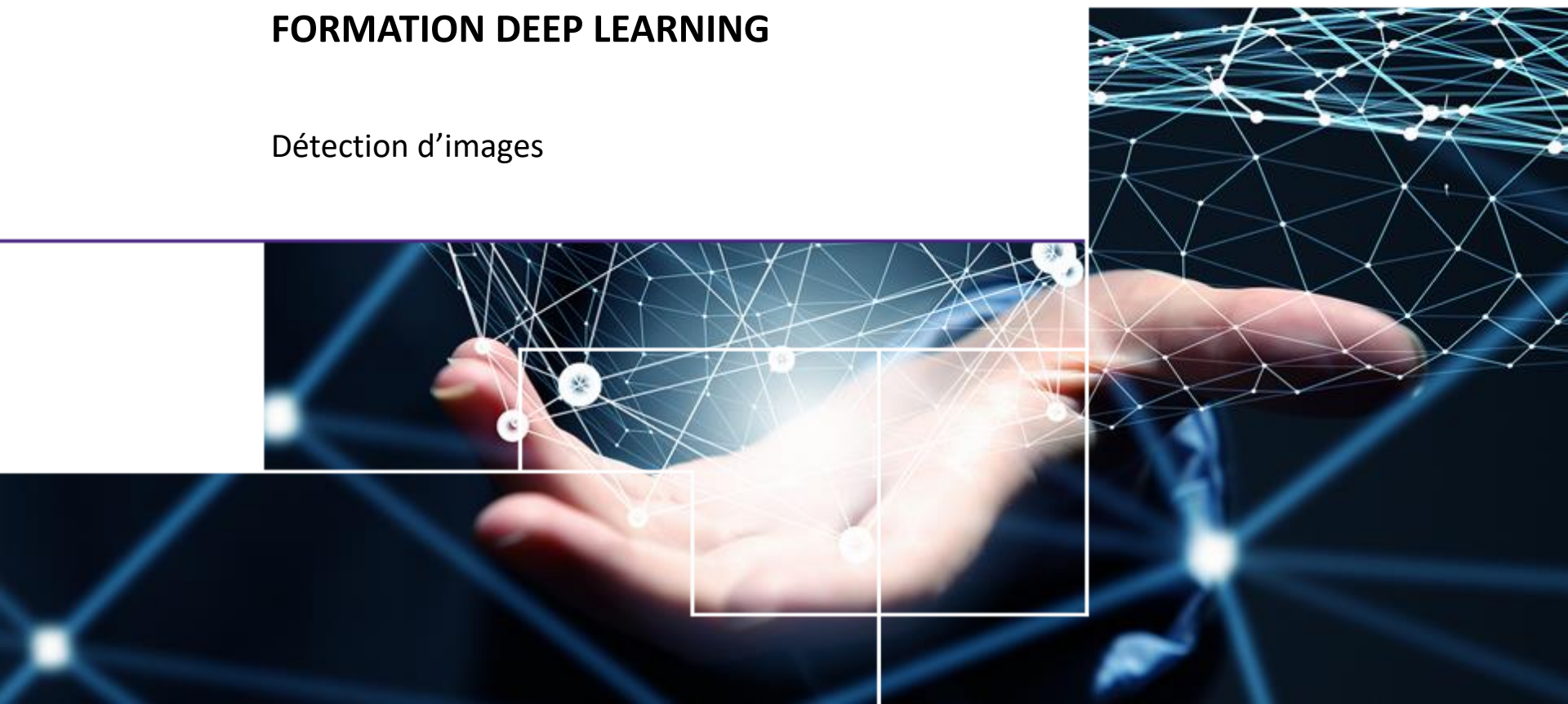


FORMATION DEEP LEARNING

Détection d'images





UNE BRÈVE HISTOIRE DE DÉTECTION

Classification



Output:

CAT

Détection



CAT
+ bounding box

**Avez-Vous une idée de comment
on pourrait faire de la détection ?**

POURQUOI DE NOUVEAUX MODÈLES ?

POURQUOI NE PAS DÉCOUPER NAÏVEMENT L'IMAGE EN RÉGIONS ALÉATOIRES ?

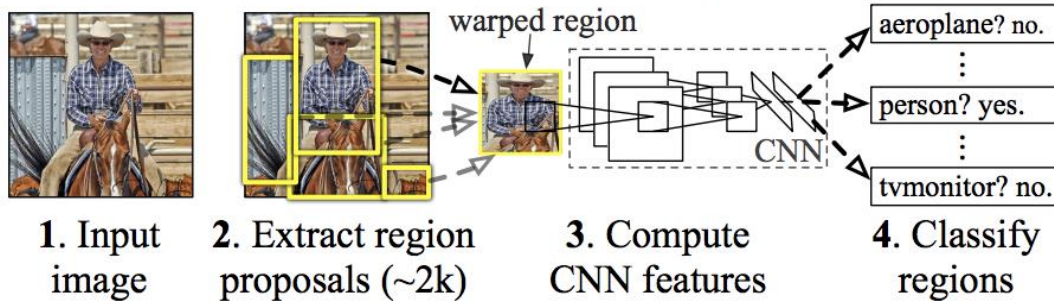


- Trop coûteux
- Procédé hasardeux
- une infinité de possibilité
- Impossible en temps réel



PROPOSER 2000 RÉGIONS À CLASSER

R-CNN: *Regions with CNN features*



selective search

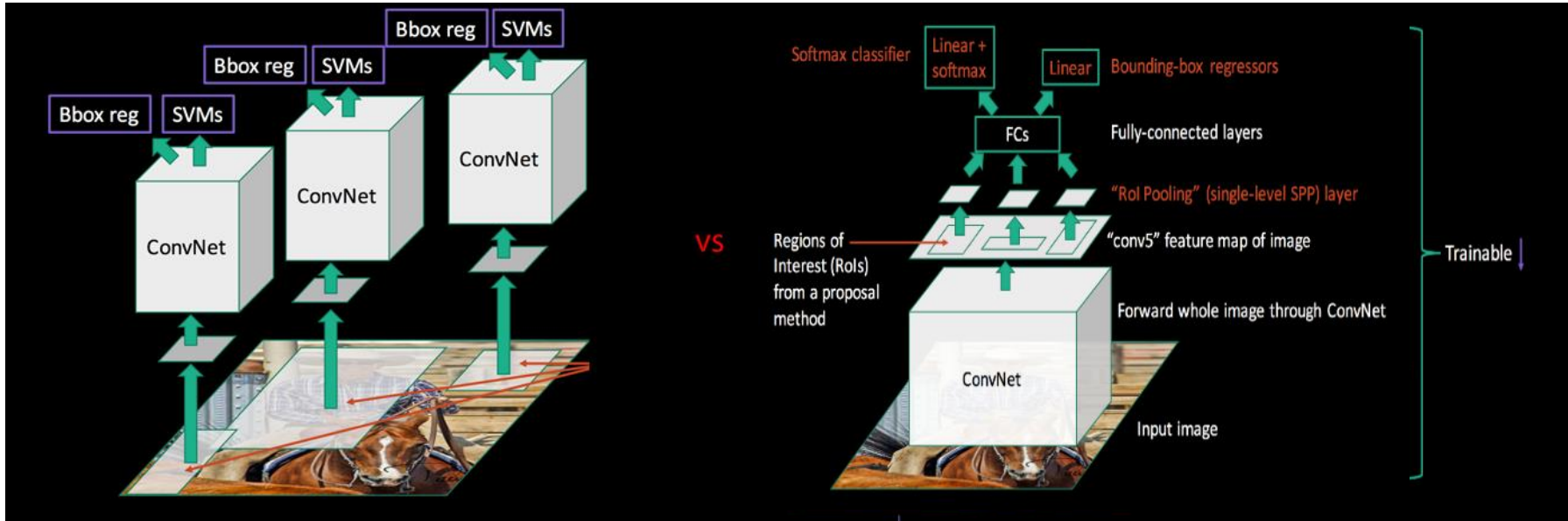


Pas besoin d'entraînement, extrait, se base sur des régions similaires (couleur, texture, taille, etc.)

- Se base sur un algorithme pour extraire 2000 régions
- AlexNet en feature extractor (transfert learning)
- SVM classificateur <https://towardsdatascience.com/r-cnn-3a9beddfd55a> +

- Très long (~47 secondes par image)
Algorithme de sélection non apprenant

COMMENT FAIRE PLUS VITE ?

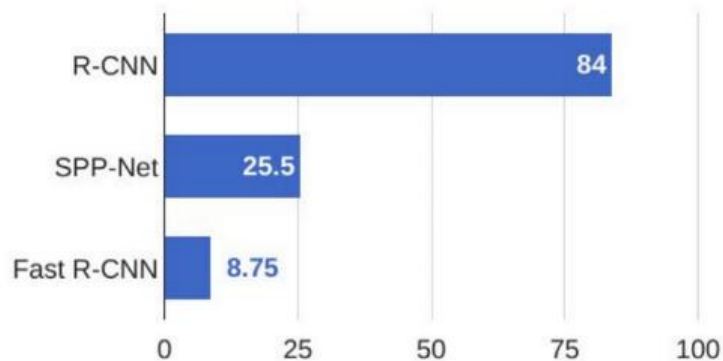


- On passe toute l'image au CNN pour obtenir une feature map de toute l'image (on utilise le convnet une étape avant)
- On utilise la feature map pour extraire des « region proposals » à l'aide de Selective Search
- On utilise un Region Of Interest layer pour uniformiser la taille des « region proposals ».
- Feed forward en classifier + Bbox

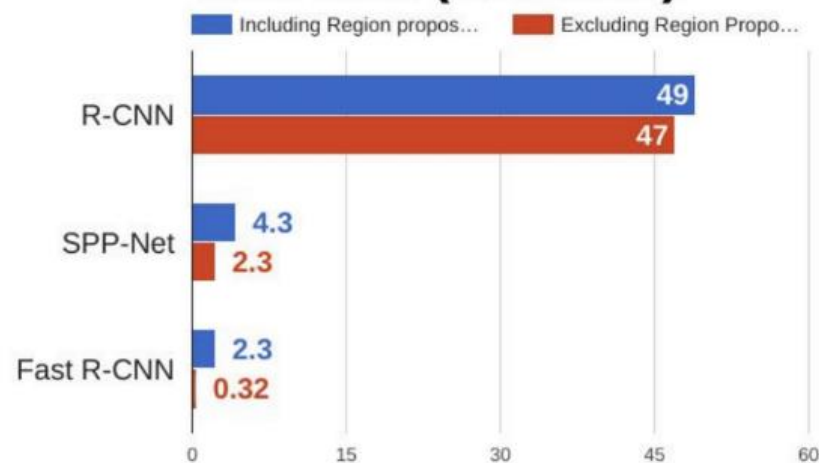
FAST R-CNN

BENCHMARK

Training time (Hours)

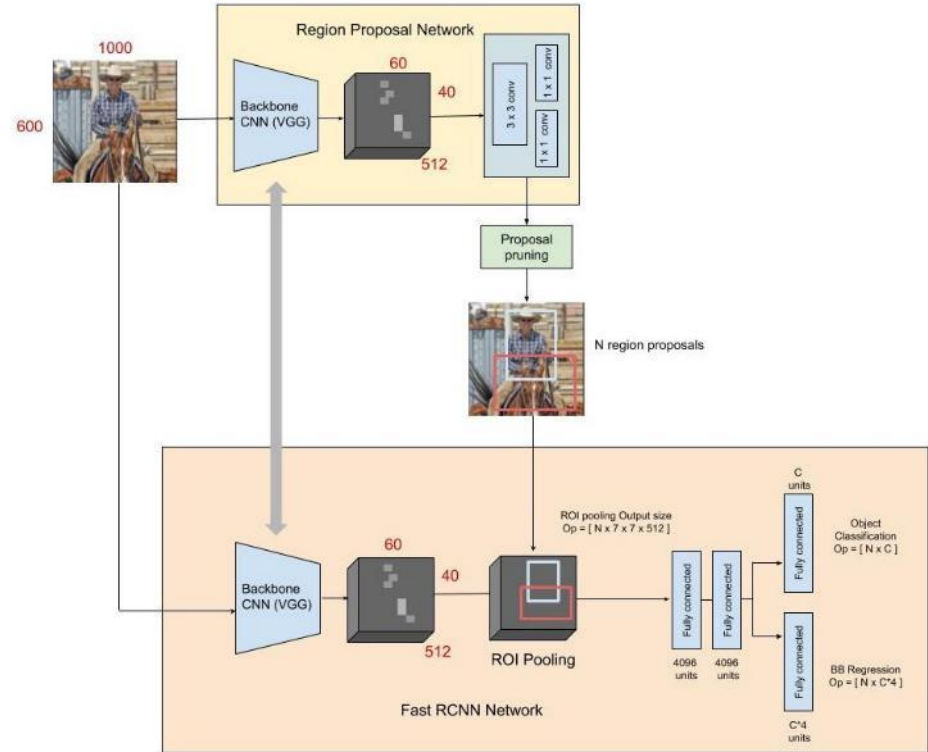
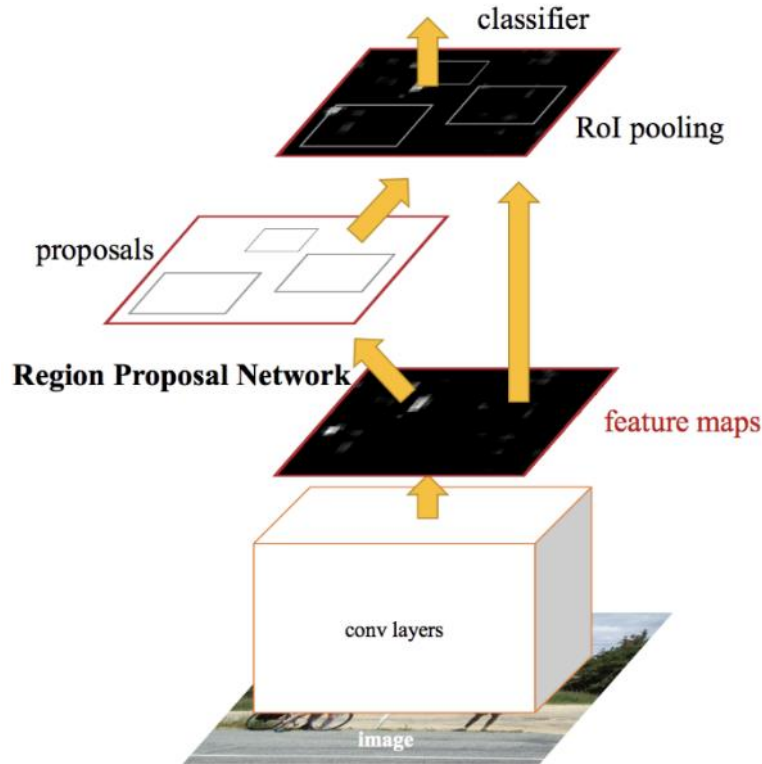


Test time (seconds)



FASTER R-CNN

COMMENT FAIRE PLUS VITE ?

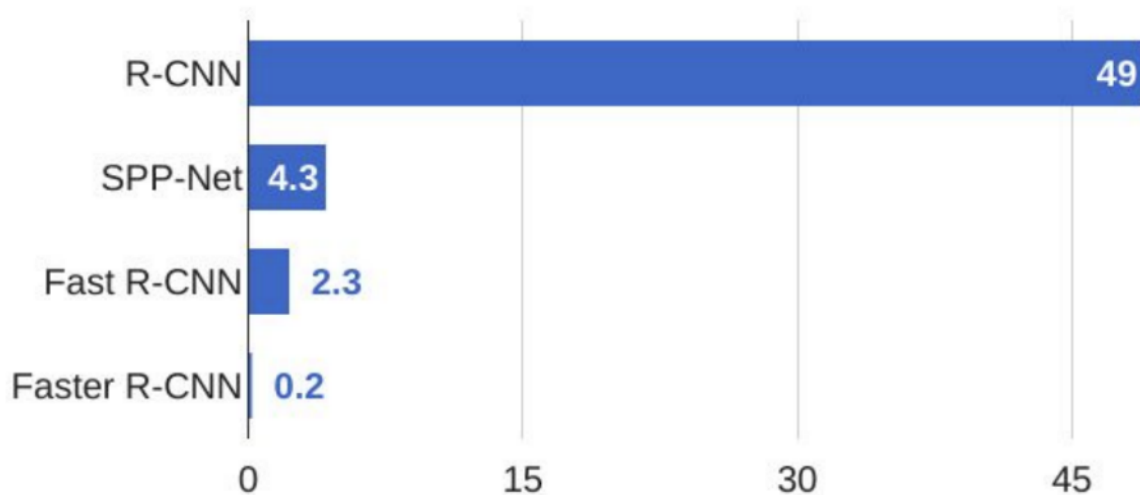


- On utilise un Neural Network pour extraire les « region proposals »
- L'algorithme de « selective search » basé sur les textures est supprimé

FASTER R-CNN

BENCHMARK

R-CNN Test-Time Speed



You Only Look Once

Détéction = Localisation + Classification

Historiquement la localisation et la classification étaient deux tâches successives :

- Engendrent des erreurs
- Plus long

A l'état de l'art, les réseaux de détection fusionnent les deux tâches et donnent en même temps localisations et classifications.

Le premier à le faire était YOLO : You Look Only Once

**Il existe de multiples versions qui s'améliorent au fur à mesure yolov4 > yolov3
(attention Yolo v5)**

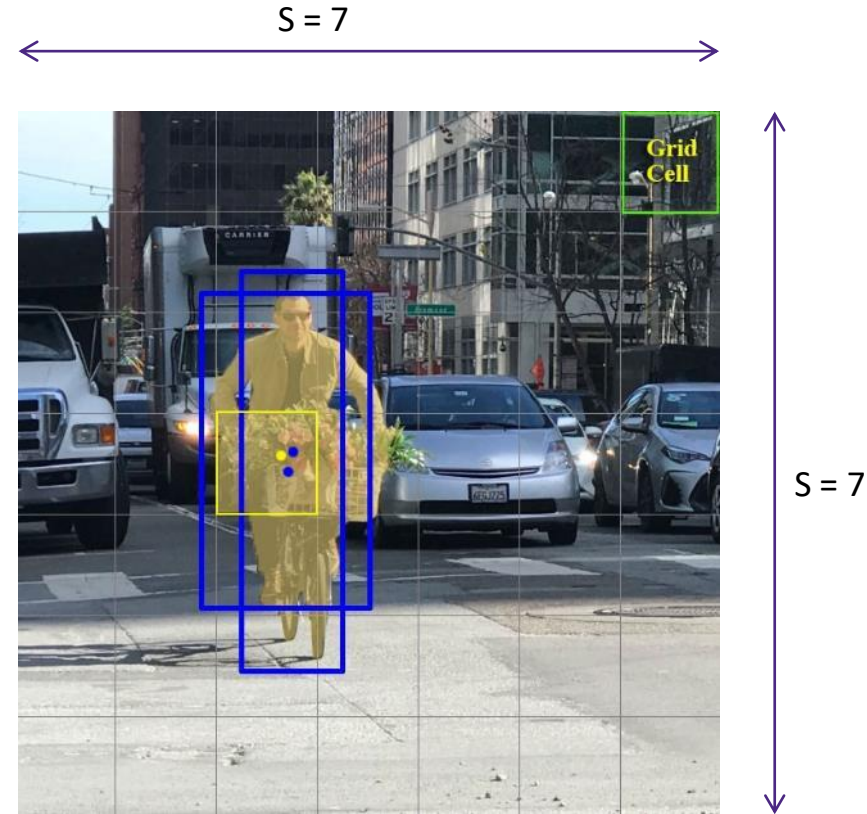
COMMENT ÇA FONCTIONNE?

- Divise l'image en région et prédit des boîtes et des probabilités pour chacune de ces régions avec un indice de confiance



COMMENT ÇA FONCTIONNE?

- On découpe l'image en plusieurs cellules
- Chaque cellule sera responsable de la prédiction d'un nombre limité de boox (anchors bbox).
- Une Bbox peut être plus grande qu'une cellule, il faut juste que le centre de la bbox soit attribuée à une cellule

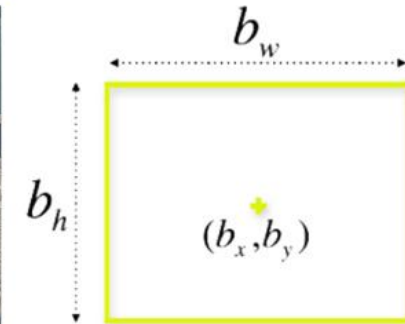


QU'EST CE QUE PRÉDIRE UNE BBOX ?

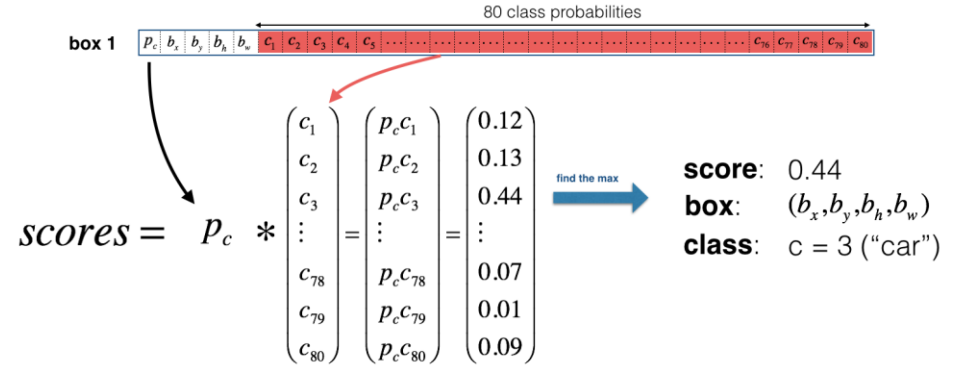
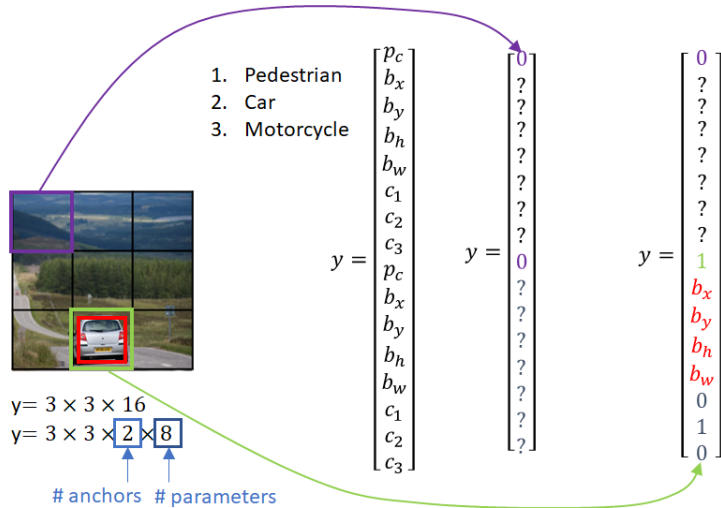
- Une Bbox est constituée des :
- Dimensions (bx, by, bh, bw)
- Score de confiance
- Classe



$$y = (p_c, b_x, b_y, b_h, b_w, c)$$



MATRICE DE SORTIE



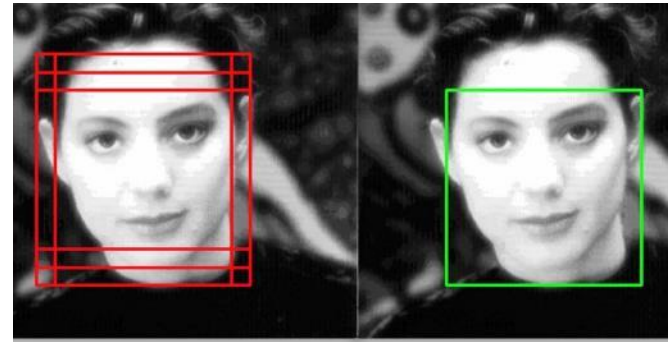
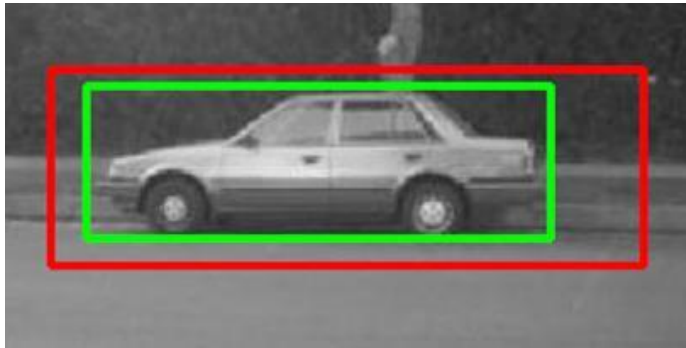
the box (b_x, b_y, b_h, b_w) has detected $c = 3$ ("car") with probability score: 0.44

- On prédit pour tous les cellules, plusieurs bboxes (=anchors bbox) pour toutes les classes un seuil de confiance
- La shape finale de sortie :
 $(S, S, B \times 5 + C)$

Où, S : Taille de Grille, B : nbr de bbox par cellule (anchors bbox)

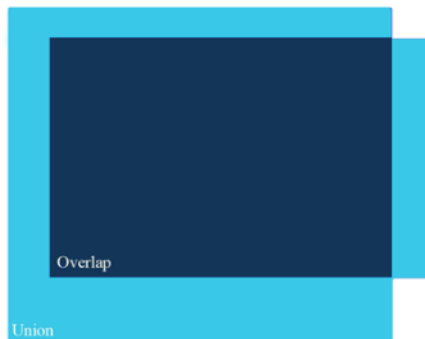
LES PROBLÈMES DE LA DÉTECTION

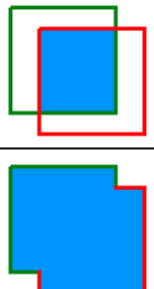
- **Comment mesurer la similarité entre 2 boîtes**
- **Comment faire pour éviter d'avoir plusieurs prédictions dans la même zone de l'image**



COMMENT ÉVALUER LES PERFORMANCES ?

L'IOU est un bon moyen de mesurer la performance entre les boîtes (bounding boxes) prédites et les boîtes attendues, théoriques



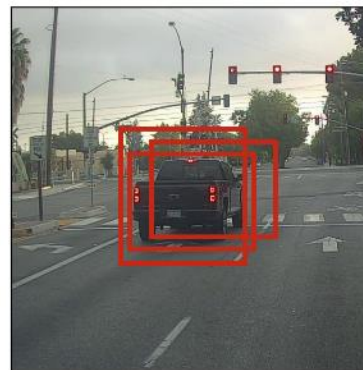
$$IOU = \frac{\text{area of overlap}}{\text{area of union}} = \frac{\text{area of overlap}}{\text{area of union}}$$


LA NON-MAX SUPPRESSION

COMMENT FAIRE POUR ÉVITER D'AVOIR PLUSIEURS PRÉDICTIONS
DANS LA MÊME ZONE DE L'IMAGE

1. Sélectionnez la case qui a le score le plus élevé.
2. Calculez son chevauchement avec toutes les autres cases et supprimez les cases qui le chevauchent plus qu'un seuil ($iou_threshold$).
3. Revenez à l'étape 1 et parcourez jusqu'à ce qu'il n'y ait plus de cases avec un score inférieur à la case actuellement sélectionnée.

Before non-max suppression



Non-Max
Suppression



After non-max suppression



YOLO: Training, formally

Bounding box coordinate regression

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

= 1 if box j and cell i are matched together, 0 otherwise

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

Bounding box score prediction

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2$$

Class score prediction

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2$$

= 1 if box j and cell i are NOT matched together

$$+ \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

= 1 if cell i has an object present

Slide credit: [YOLO Presentation @ CVPR 2016](#)

47

$$\begin{aligned}
 & \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] && \text{1 when there is object, 0 when there is no object} \\
 & + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] && \text{Bounding Box Location (x, y) when there is object} \\
 & + \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2 && \text{Bounding Box size (w, h) when there is object} \\
 & + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2 && \text{Confidence when there is object} \\
 & + \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 && \text{1 when there is no object, 0 when there is object} \\
 & && \text{Confidence when there is no object} \\
 & && \text{Class probabilities when there is object}
 \end{aligned}$$

YOLO: LA LOSS

IMPLÉMENTATION PYTHON

```
ciou = tf.expand_dims(bbox_ciou(pred_xywh, label_xywh), axis=-1) # (8, 13, 13, 3, 1)
input_size = tf.cast(input_size, tf.float32)

# 每个预测框xxxiou_loss的权重 = 2 - (ground truth的面积/图片面积)
bbox_loss_scale = 2.0 - 1.0 * label_xywh[:, :, :, 2:3] * label_xywh[:, :, :, 3:4] / (input_size ** 2)
ciou_loss = respond_bbox * bbox_loss_scale * (1 - ciou) # 1. respond_bbox作为mask, 有物体才计算xxxiou_loss

# 2. respond_bbox作为mask, 有物体才计算类别loss
prob_loss = respond_bbox * tf.nn.sigmoid_cross_entropy_with_logits(labels=label_prob, logits=conv_raw_prob)

# 3. xxxiou_loss和类别loss比较简单。重要的是conf_loss, 是一个focal loss
# 分两步: 第一步是确定 grid_h * grid_w * 3 个预测框 哪些作为反例; 第二步是计算focal loss.
expand_pred_xywh = pred_xywh[:, :, :, :, np.newaxis, :] # 扩展为(?, grid_h, grid_w, 3, 1, 4)
expand_bboxes = bboxes[:, np.newaxis, np.newaxis, np.newaxis, :, :] # 扩展为(?, 1, 1, 1, 150, 4)
iou = bbox_iou(expand_pred_xywh, expand_bboxes) # 所有格子的3个预测框 分别 和 150个ground truth 计算iou. (?, grid_h, grid_w, 3, 150)
max_iou = tf.expand_dims(tf.reduce_max(iou, axis=-1), axis=-1) # 与150个ground truth的iou中, 保留最大那个iou. (?, grid_h, grid_w, 3, 1)

# respond_bgd代表 这个分支输出的 grid_h * grid_w * 3 个预测框是否是 反例 (背景)
# label有物体, respond_bgd是0. 没物体的话: 如果和某个gt(共150个)的iou超过iou_loss_thresh, respond_bgd是0; 如果和所有gt(最多150个)的iou都小于iou_loss_thresh, respond_bgd是1.
# respond_bgd是0代表有物体, 不是反例; 权重respond_bgd是1代表没有物体, 是反例.
# 有趣的是, 模型训练时由于不断更新, 对于同一张图片, 两次预测的 grid_h * grid_w * 3 个预测框 (对于这个分支输出) 是不同的。用的是这些预测框来与gt计算iou来确定哪些预测框是反例.
# 而不是用固定大小 (不固定位置) 的先验框.
respond_bgd = (1.0 - respond_bbox) * tf.cast(max_iou < iou_loss_thresh, tf.float32)

# 二值交叉熵损失
pos_loss = respond_bbox * (0 - K.log(pred_conf + K.epsilon()))
neg_loss = respond_bgd * (0 - K.log(1 - pred_conf + K.epsilon()))

conf_loss = pos_loss + neg_loss

# 回顾respond_bgd, 某个预测框和某个gt的iou超过iou_loss_thresh, 不被当作是反例。在参与“预测的置信位 和 真实置信位 的二值交叉熵”时, 这个框也可能不是正例(label里没标这个框是1的话)。这个
# 这种框一般是gt框附近的框, 或者是gt框所在格子的另外两个框。它既不是正例也不是反例不参与置信度loss的计算。(论文里称之为ignore)

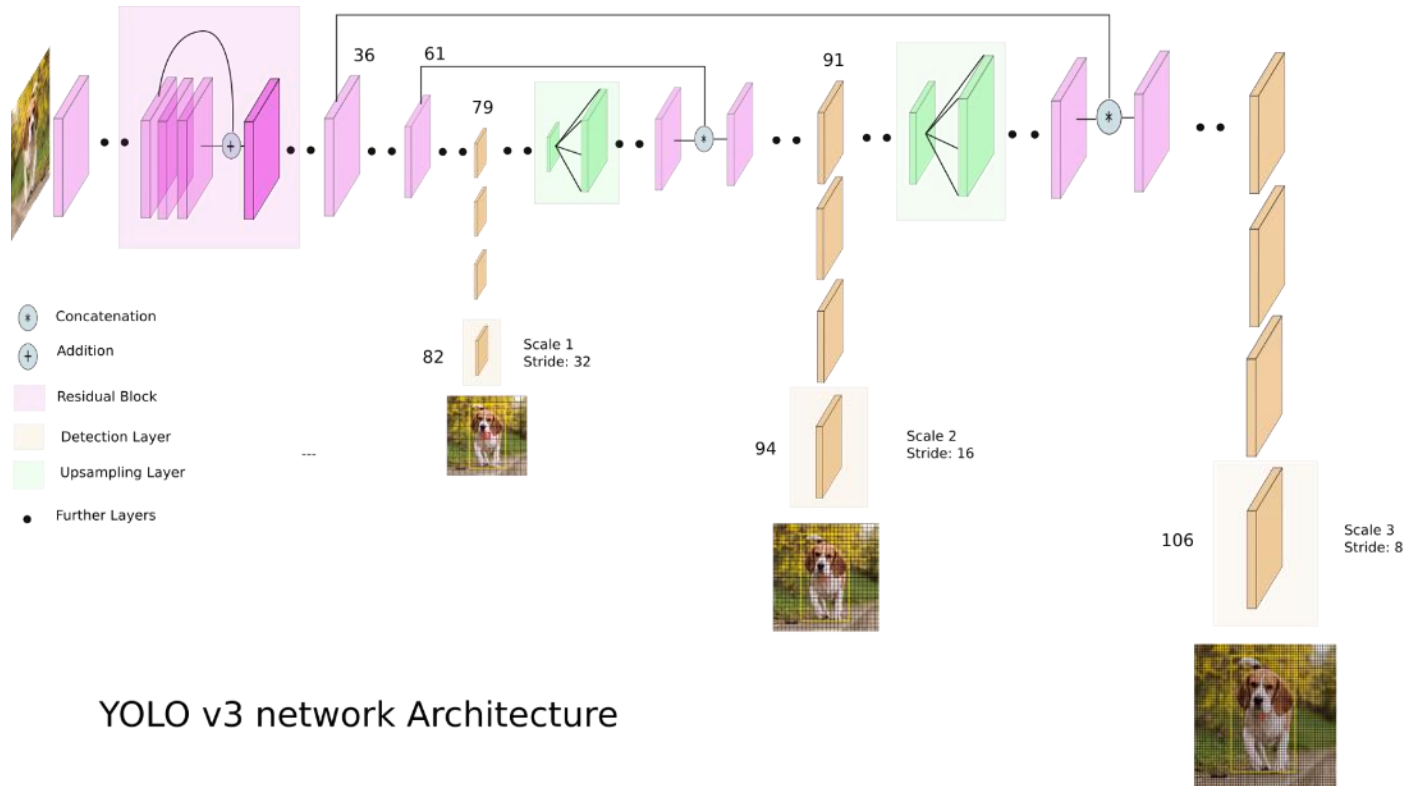
ciou_loss = tf.reduce_mean(tf.reduce_sum(ciou_loss, axis=[1, 2, 3, 4])) # 每个样本单独计算自己的ciou_loss, 再求平均值
conf_loss = tf.reduce_mean(tf.reduce_sum(conf_loss, axis=[1, 2, 3, 4])) # 每个样本单独计算自己的conf_loss, 再求平均值
prob_loss = tf.reduce_mean(tf.reduce_sum(prob_loss, axis=[1, 2, 3, 4])) # 每个样本单独计算自己的prob_loss, 再求平均值

return ciou_loss, conf_loss, prob_loss
```

L'architecture

COMMENT ÇA FONCTIONNE?

- Améliore les performances en prenant des features adaptées aux tailles de boîtes



YOLO v3 network Architecture

Yolo 3 + bag of specials + bag of freebies = Yolo4

Modifications du réseau lui-même :

- *Fonction d'activation MISH*
- *SPP*
- *PAN*
- *Etc.*

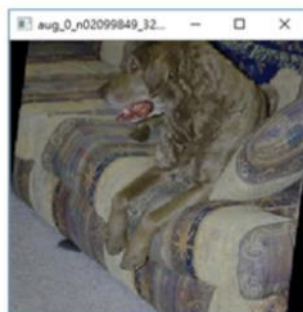
Ensemble d'améliorations
sans impact sur le réseau :

- la fonction de coût
- l'augmentation des données
- la cross mini-batch normalization
- Etc.

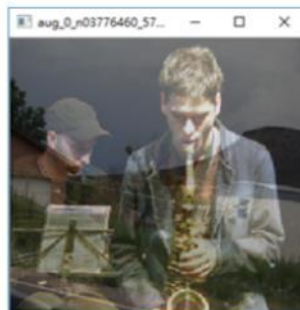
Pour une introduction vulgarisée : [source](#)

YOLO 4 : FREEBIES

DATA AUGMENTATION



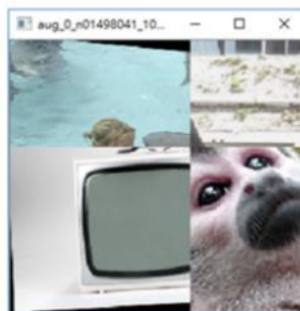
(a) Crop, Rotation, Flip, Hue, Saturation, Exposure, Aspect.



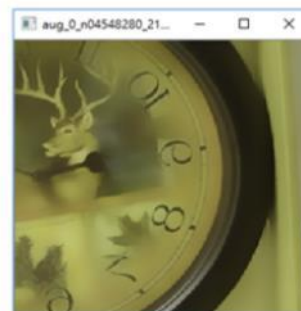
(b) MixUp



(c) CutMix



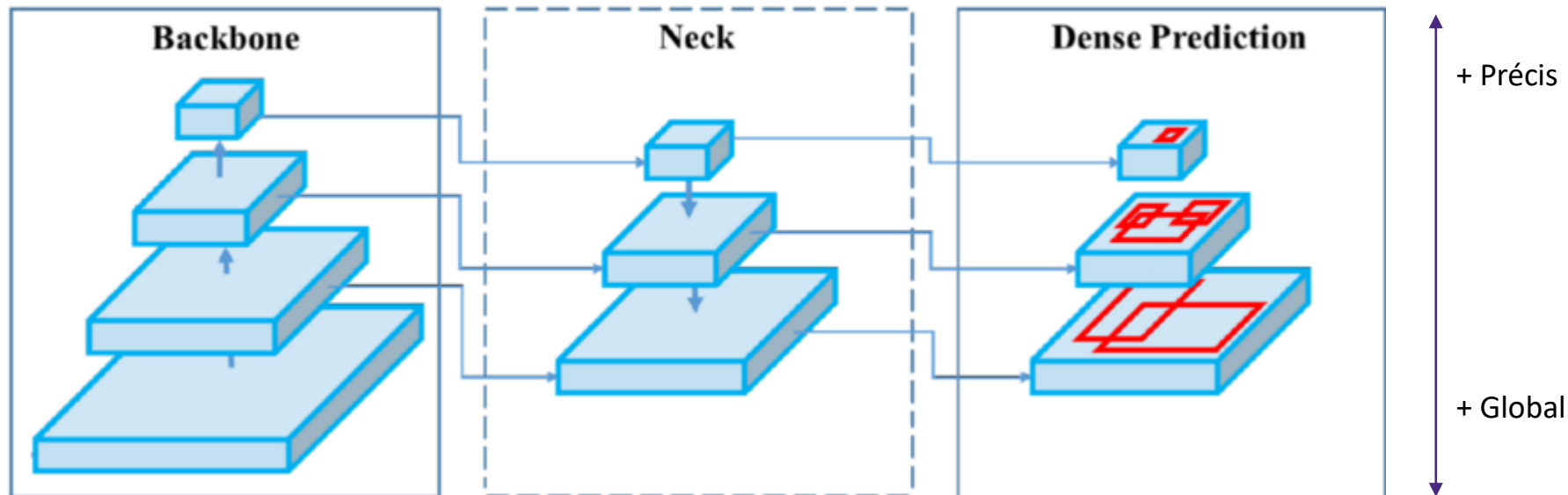
(d) Mosaic



(e) Blur

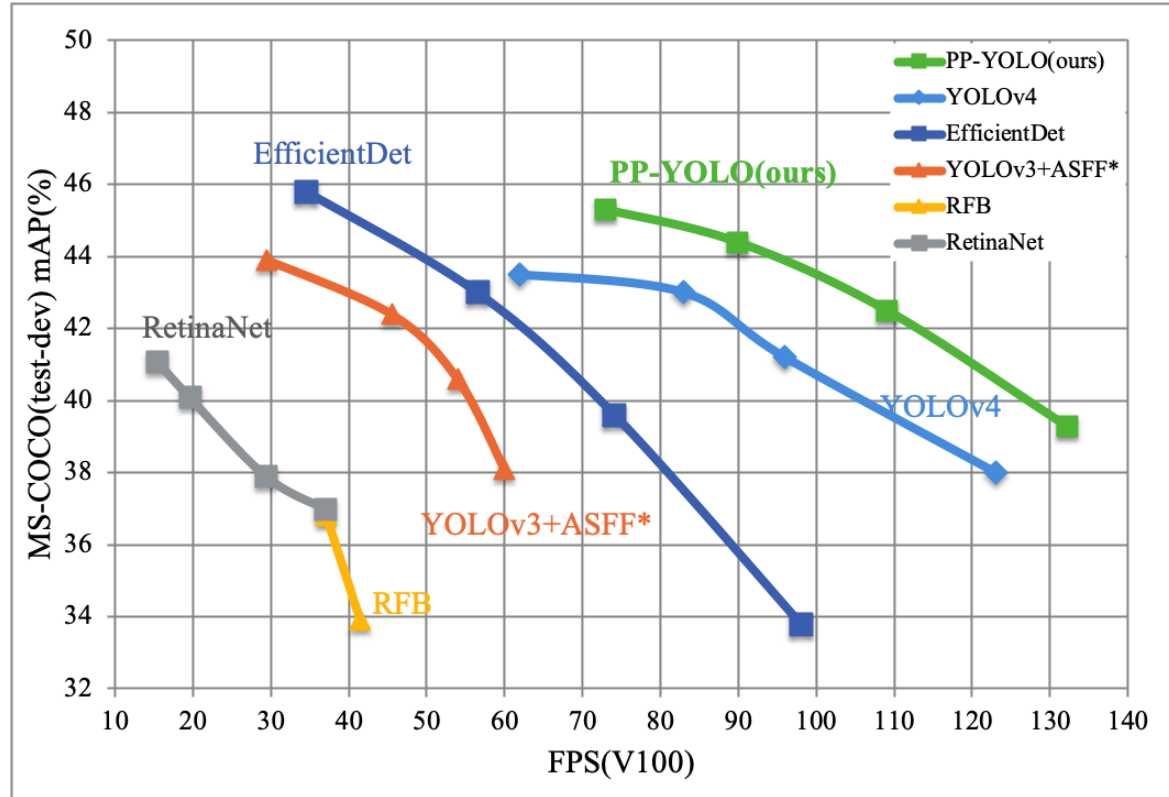
YOLO 4 : PRINCIPES D'ARCHITECTURES

3 PARTIES



- il peut être vu comme un “convertisseur” qui **converti l'image en entrée => features**
- Habituellement reprend des modèles entraînés sur ImageNet
- le “neck” a pour rôle d'extraire les features pertinentes de l'ensemble des couches de la backbone, et de les combiner en features utiles à notre tâche de détection
- la tête est responsable de la **décision finale** du réseau. (identique à yolo V3)

Pour approfondir



<https://towardsdatascience.com/yolo-v4-or-yolo-v5-or-pp-yolo-dad8e40f7109>

BENCHMARK PAR YOLO

Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	~ 1000 × 600
Fast YOLO	52.7	155	1	98	448 × 448
YOLO (VGG16)	66.4	21	1	98	448 × 448
SSD300	74.3	46	1	8732	300 × 300
SSD512	76.8	19	1	24564	512 × 512
SSD300	74.3	59	8	8732	300 × 300
SSD512	76.8	22	8	24564	512 × 512

[source](#)

- **Transfert Learning** issu d'un réseau entraîné sur ImageNet
- **Bon compromis** entre temps d'exécution et performance
- Existe en version « tiny », adaptée pour de l'embarqué
- **Utilisable en « temps réel »**
- **Peut détecter de multiples objets par image**