

Évolution des Méthodes R-CNN : De R-CNN à Faster R-CNN

Zaouche Djaouida

Cy-tech

November 26, 2024

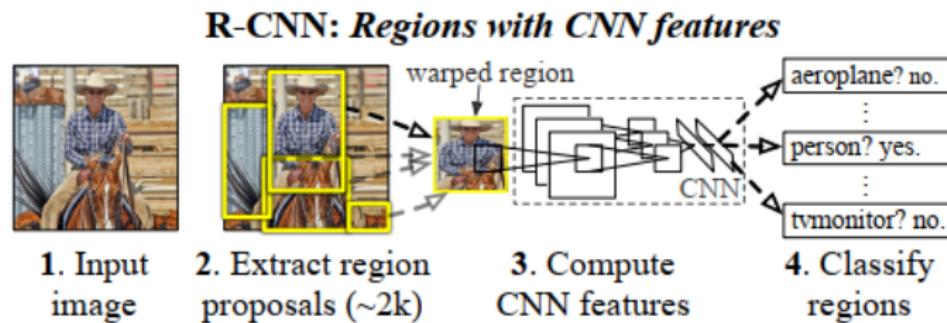
R-CNN

- R-CNN (Region-based Convolutional Neural Network) a été créé par Ross Girshick et ses collègues de l'Université de Californie à Berkeley en 2014. Leur article de référence, intitulé "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", a présenté le modèle R-CNN et a fait progresser de manière significative le domaine de la détection d'objets.

Détection d'objets avec R-CNN

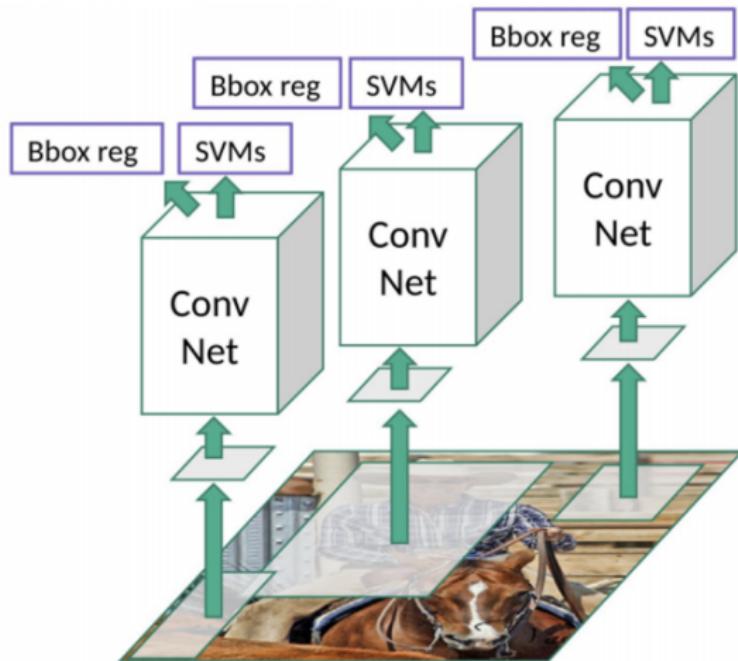
Le système de détection d'objets de R-CNN est composé de trois modules :

- Le premier module génère des propositions de régions indépendantes qui pourraient contenir des objets.
- Le deuxième module est un réseau neuronal convolutionnel qui extrait, pour chaque région, un vecteur de caractéristiques de taille fixe.
- Le troisième module est un ensemble de SVMs (Support Vector Machines) et de modèles de régression.



Les étapes de R-CNN

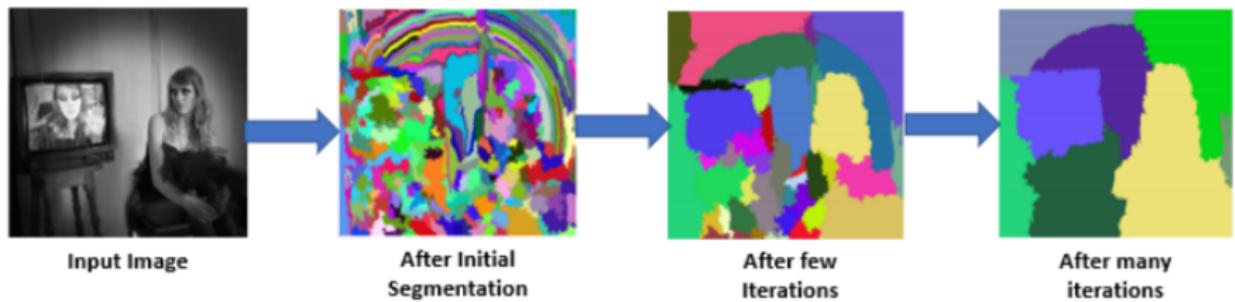
- La figure ci-dessous résume les principales étapes de R-CNN.



Proposition de régions (1) (étape 1)

- Plusieurs méthodes ont été proposées pour générer des propositions de régions indépendantes.
- La méthode de recherche sélective, par exemple, consiste à fusionner ou à diviser des segments d'une image en utilisant différents indices visuels tels que la couleur, la texture et la forme, générant ainsi un ensemble varié de propositions de régions. Cette méthode a été utilisée dans R-CNN.
- Les auteurs de R-CNN utilisent l'algorithme de recherche sélective pour générer, pour chaque image, 2000 propositions de régions indépendantes. Ces régions sont généralement délimitées par des boîtes rectangulaires et représentent des objets.

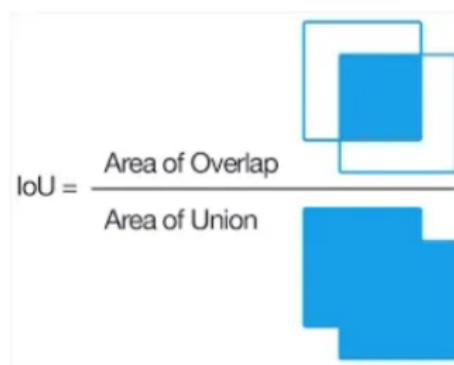
Proposition de régions (2)



Example of the Selective Search algorithm applied to an image.

Intersection over Union (IoU)

- L'intersection sur l'union (IoU) est une métrique utilisée pour mesurer le chevauchement entre deux zones, couramment utilisée dans les tâches de vision par ordinateur telles que la détection et la segmentation. La métrique IoU est comprise entre 0 et 1.



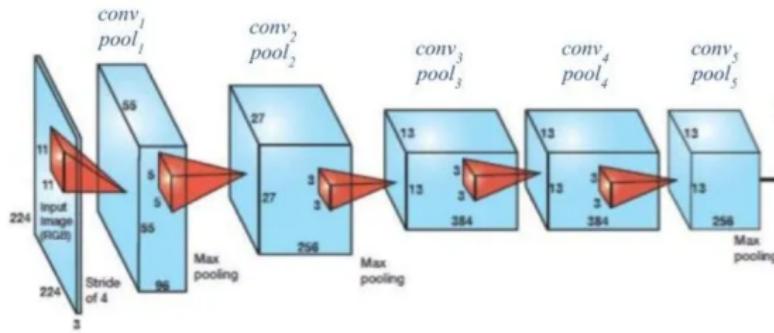
Étapes de l'extraction des caractéristiques (étape 2)

- Dans R-CNN, chaque région est transmise à un réseau neuronal convolutif.
- Les régions sont recadrées et redimensionnées à une taille fixe avant d'être traitées par la partie CNN du R-CNN.



Pré-entraînement supervisé (étape 2)

- Le CNN introduit par Krizhevsky et al., largement reconnu sous le nom d'AlexNet, se compose de 5 couches convolutionnelles et de 2 couches entièrement connectées. Initialement, il est entraîné sur le jeu de données de classification ILSVRC2012, une tâche de classification d'images en 1000 classes avec un jeu de données d'images substantiel, permettant aux couches convolutionnelles d'apprendre des caractéristiques d'image fondamentales.

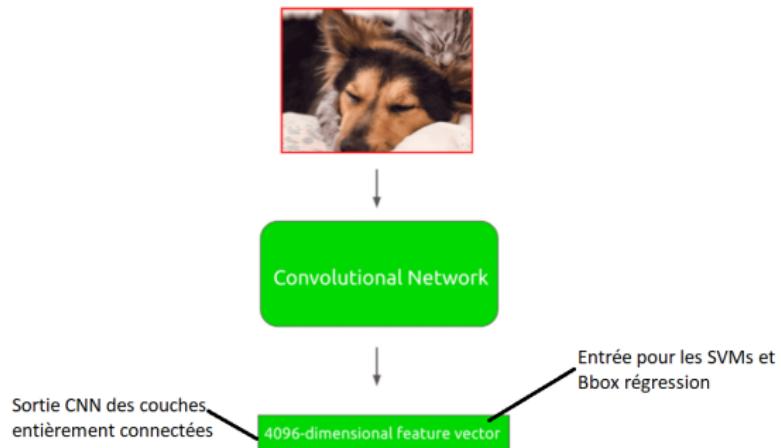


”Domain-Specific Fine-Tuning” (étape 2)

- Un ”fine-tune” du CNN est effectué en utilisant un jeu de données spécifique au domaine. Pour les besoins de la détection d’objets avec R-CNN, le jeu de données PASCAL VOC ou le jeu de données de détection ILSVRC 2013 est utilisé pour affiner davantage les paramètres du CNN.
- Concrètement, la couche de classification à 1000 classes d’un AlexNet typique (pour les 1000 classes du jeu de données ILSVRC 2012) est remplacée par une classification à $(N+1)$ classes, où N est le nombre de classes d’objets dans le nouveau jeu de données utilisé, et une classe supplémentaire est ajoutée pour l’étiquette de fond. N est égal à 20 dans le cas de VOC et à 200 dans le cas d’ILSVRC 2013.

Sorties du CNN (étape 2)

- Pour chaque région, le CNN génère un vecteur de caractéristiques à 4096 dimensions.



Classificateurs de catégories d'objets (1) (étape 3)

- Le vecteur de caractéristiques d'une région passe ensuite par une série de machines à vecteurs de support linéaires (SVMs), chaque SVM étant spécialisée dans la reconnaissance d'une classe d'objets spécifique. Par exemple, une SVM est entraînée à identifier les chiens, une autre à détecter les chats, et ainsi de suite.
- Etant donnée une SVM, les régions positives, pour cette SVM, contiennent la classe d'objet cible de la SVM.

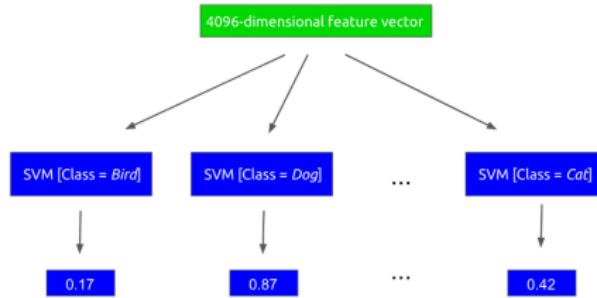
Classificateurs de catégories d'objets (2) (étape 3)

- Une proposition de région est considérée comme positive pour une SVM reconnaissant une classe d'objet C , si elle a une valeur IoU d'au moins 0,5 avec une "boîte de délimitation réelle" d'un objet associé à la classe C . Dans le cas contraire, elle est classée comme négative.



Les sorties des SVMs (étape 3)

- La carte des caractéristiques de chaque région proposée est introduite dans chaque SVM afin de calculer le score de vraisemblance pour cette région, indiquant la probabilité que la région appartienne à la classe respective associée au SVM.



- La classe d'une proposition de région est déterminée par le SVM qui attribue le score le plus élevé à cette région.

Modèle de régression (1) (étape 3)

- L'entrée de chaque modèle de régression est un ensemble de N paires d'entraînement $\{(P^{(i)}, G^{(i)})\}_{i=1,\dots,N}$, où :
 - ➊ $P = (P_x^{(i)}, P_y^{(i)}, P_w^{(i)}, P_h^{(i)})$ est une donnée prédite spécifiant d'une part les coordonnées en pixels du centre de la boîte de délimitation de la proposition $P^{(i)}$, et d'autre part, la largeur et la hauteur de $P^{(i)}$ en pixels.
 - ➋ $G^{(i)} = (G_x^{(i)}, G_y^{(i)}, G_w^{(i)}, G_h^{(i)})$ spécifie les coordonnées en pixels du centre de la boîte englobante réelle, ainsi que la largeur et la hauteur de $G^{(i)}$ en pixels.
- L'objectif d'un modèle de régression est d'apprendre une transformation qui corrige, pour chaque i , une boîte proposée $P^{(i)}$ en se basant sur une boîte réelle $G^{(i)}$. Pour la suite, nous ignorons l'exposant i à moins que cela ne soit nécessaire.

Modèle de régression (2) (étape 3)

- Les mises à jour de P_x , P_y , P_w et P_h se font avec des translations en utilisant $d_x(P)$, $d_y(P)$, $d_w(P)$ et $d_h(P)$. Les deux fonctions $d_x(P)$ et $d_y(P)$ spécifient une translation invariante à l'échelle du centre de la boîte englobante de P , tandis que $d_w(P)$ et $d_h(P)$ spécifient des translations dans l'espace logarithmique de la largeur et de la hauteur de la boîte englobante de P . Les nouvelles valeurs de la boîte à prédire sont calculées comme suit :

$$\hat{G}_x = P_w \cdot d_x(P) + P_x$$

$$\hat{G}_y = P_h \cdot d_y(P) + P_y$$

$$\hat{G}_w = P_w \cdot \exp(d_w(P))$$

$$\hat{G}_h = P_h \cdot \exp(d_h(P))$$

Modèle de régression (3) (étape 3)

- Supposons que :
 - ➊ $d_x(P) = 0.2$ et $d_y(P) = -0.1$.
 - ➋ La largeur de la boîte P est $P_w = 100$ pixels et sa hauteur est $P_h = 200$ pixels.
- Alors, le déplacement réel appliqué au centre de la boîte P sera :
 - ➌ Nouveau $x = P_x + d_x(P) \times P_w = P_x + 0.2 \times 100 = P_x + 20$
 - ➍ Nouveau $y = P_y + d_y(P) \times P_h = P_y - 0.1 \times 200 = P_y - 20$

Modèle de régression (4) (étape 3)

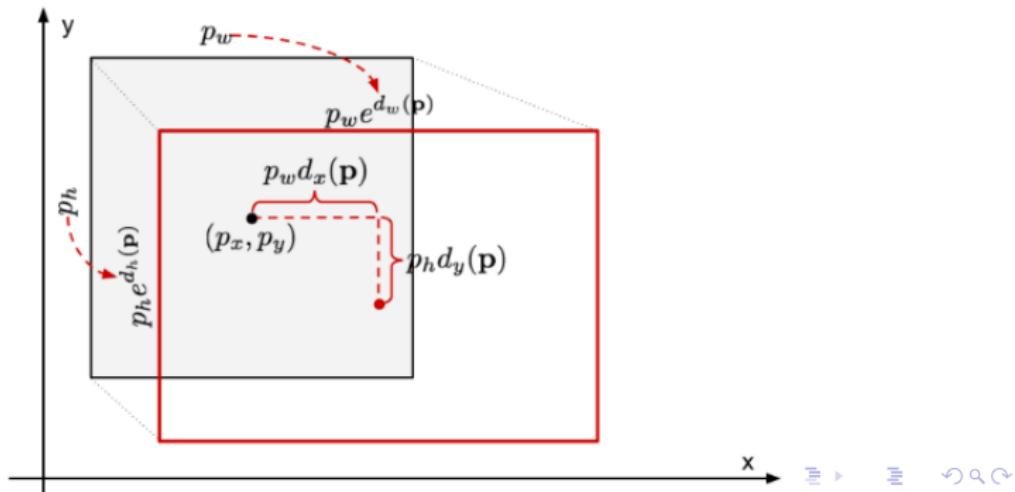
- La figure suivante shématise la translation du centre de la boîte de la proposition.

$$\hat{g}_x = p_w d_x(\mathbf{p}) + p_x$$

$$\hat{g}_y = p_h d_y(\mathbf{p}) + p_y$$

$$\hat{g}_w = p_w \exp(d_w(\mathbf{p}))$$

$$\hat{g}_h = p_h \exp(d_h(\mathbf{p}))$$



Modèle de régression (5) (étape 3)

- Le déplacement $d_w(P)$ (resp. $d_w(h)$) peut être interprété comme un décalage (ou un ajustement) qui est appliqué à la largeur de la boîte P dans l'espace logarithmique. Cela peut être formulé, par exemple pour P'_w , comme suit :

$$P'_w = \exp(\log(P_w) + d_w(P))$$

Ici, P'_w est la nouvelle largeur ajustée de la boîte P après avoir appliqué le décalage $d_w(P)$.

Modèle de régression (6) (étape 3)

- Supposons que la largeur initiale $P_w = 100$ pixels et imaginons que le modèle a appris un déplacement $d_w(P) = 0.1$.

1

$$\log(P_w) = \log(100) \approx 4.605$$

2

$$\log(P'_w) = 4.605 + 0.1 = 4.705$$

3

$$P'_w = \exp(4.705) \approx 110.5 \text{ pixels}$$

Modèle de régression (7) (étape 3)

- Chaque fonction $d_*(P)$ (où $*$ est l'un des éléments suivants : x, y, h, w) est modélisée comme une fonction linéaire des caractéristiques de la couche pool5 de la proposition P , désignée par $\phi_5(P)$:

$$d_*(P) = w_*^T \phi_5(P),$$

où w_* est un vecteur de paramètres de modèle apprenables.

- L'apprentissage de w_* se fait en optimisant l'objectif des moindres carrés régularisés (régression ridge) :

$$\hat{w}_* = \arg \min_{\hat{w}_*} \sum_{i=1}^N (t_*^{(i)} - \hat{w}_*^T \phi_5(P^{(i)}))^2 + \lambda \|\hat{w}_*\|^2$$

Modèle de régression (8) (étape 3)

- Les cibles de régression t_* pour la paire d'apprentissage (P, G) sont définies comme suit :

$$t_x = \frac{G_x - P_x}{P_w},$$

$$t_y = \frac{G_y - P_y}{P_h},$$

$$t_w = \log\left(\frac{G_w}{P_w}\right),$$

$$t_h = \log\left(\frac{G_h}{P_h}\right).$$

Frame Title

An obvious benefit of applying such transformation is that all the bounding box correction functions, $d_i(\mathbf{p})$ where $i \in \{x, y, w, h\}$, can take any value between $[-\infty, +\infty]$. The targets for them to learn are:

$$\begin{aligned}t_x &= (g_x - p_x)/p_w \\t_y &= (g_y - p_y)/p_h \\t_w &= \log(g_w/p_w) \\t_h &= \log(g_h/p_h)\end{aligned}$$

A standard regression model can solve the problem by minimizing the SSE loss with regularization:

$$\mathcal{L}_{\text{reg}} = \sum_{i \in \{x, y, w, h\}} (t_i - d_i(\mathbf{p}))^2 + \lambda \|\mathbf{w}\|^2$$

The regularization term is critical here and RCNN paper picked the best λ by cross validation. It is also noteworthy that not all the predicted bounding boxes have corresponding ground truth boxes. For example, if there is no overlap, it does not make sense to run bbox regression. Here, only a predicted box with a nearby ground truth box with at least 0.6 IoU is kept for training the bbox regression model.

Modèle de régression (9) (étape 3)

- La figure suivante est un exemple de la correction d'une boîte englobante :



Bounding Box Regressor [Class = Dog]

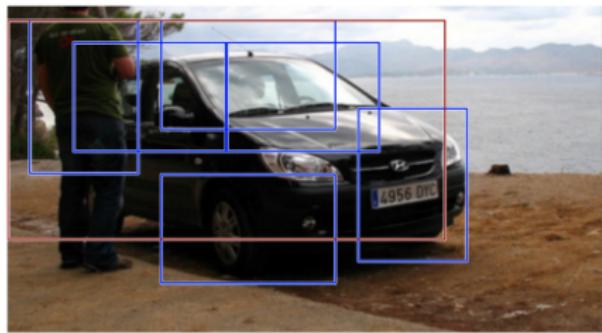


Non-Maximum Suppression

Le modèle peut identifier plusieurs boîtes de délimitation pour un même objet. La suppression non maximale (NMS) permet d'éviter les détections en double d'un même objet. Voici les étapes à suivre pour gérer les boîtes englobantes pour un objet :

- Triez les boîtes englobantes en fonction de leurs scores de confiance.
- Les boîtes dont le niveau de confiance est faible doivent être écartées.
- S'il reste des boîtes englobantes, sélectionnez celle qui a le score le plus élevé et éliminez toutes les autres boîtes qui ont une valeur d'intersection sur l'union (IoU) élevée (par exemple, supérieure à 0,5) par rapport à la boîte sélectionnée.
- Ce processus permet de ne conserver que la meilleure boîte englobante pour chaque objet détecté.

Non-Maximum Suppression



Before non-max suppression



After non-max suppression

Goulot d'étranglement

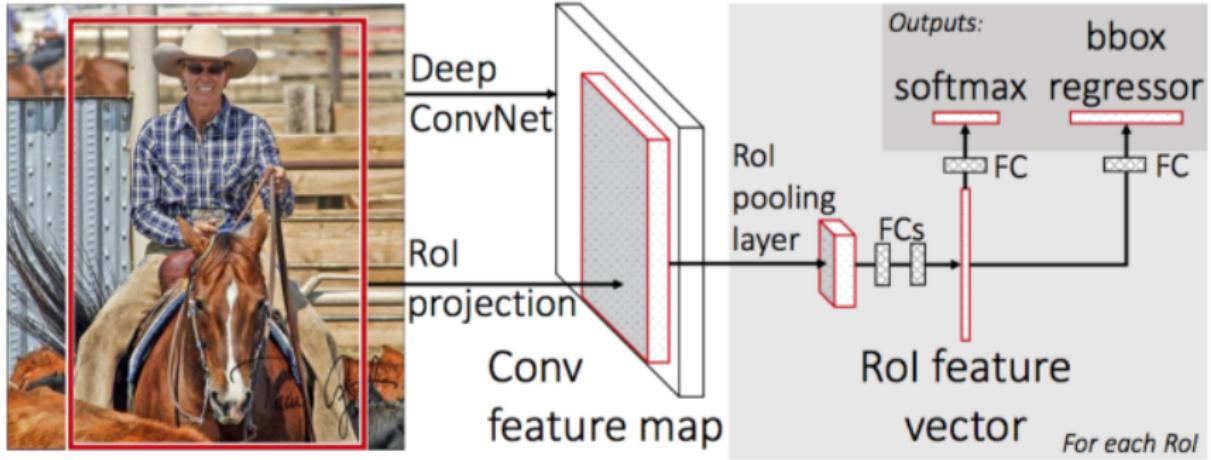
L'examen des étapes d'apprentissage du modèle R-CNN révèle que l'apprentissage d'un R-CNN est coûteux en temps en raison des étapes intensives suivantes :

- Exécution d'un algorithme de recherche sélective pour générer 2000 propositions de régions pour chaque image.
- Extraction d'un vecteur de caractéristiques CNN pour chaque région de chaque image (N images \times 2000 régions).
- Le processus global repose sur trois modèles distincts avec un minimum de calculs partagés : le réseau neuronal convolutif (CNN) pour l'extraction des caractéristiques, le classificateur SVM pour l'identification des objets cibles, et le modèle de régression pour affiner les boîtes de délimitation des régions détectées.

Fast R-CNN

- Pour accélérer le R-CNN, Girshick (2015) a optimisé le processus de formation en combinant les trois modèles dans un cadre unique, appelé Fast R-CNN.
- Au lieu d'extraire indépendamment des vecteurs de caractéristiques CNN pour chaque proposition de région, ce modèle effectue une seule passe sur l'ensemble de l'image. Ceci permet, à toutes les propositions de régions, de partager le même vecteur de caractéristiques. Ce vecteur de caractéristiques est ensuite utilisé pour le classificateur d'objets et le régresseur de boîtes de délimitation.
- En résumé, le calcul partagé accélère considérablement le R-CNN.

Fast R-CNN



Region of interest pooling

- Dans Fast R-CNN, une Region of Interest (RoI) est une sous-région d'une image où un objet potentiel est détecté.
- Chaque RoI est extraite à partir des caractéristiques de l'image entière (calculées par un CNN partagé) et redimensionnée pour une taille fixe à l'aide de l'algorithme RoI pooling. Cela permet d'extraire des caractéristiques spécifiques de chaque région, afin de classifier et affiner les limites des objets dans ces zones sans recalculer les caractéristiques pour chaque proposition.

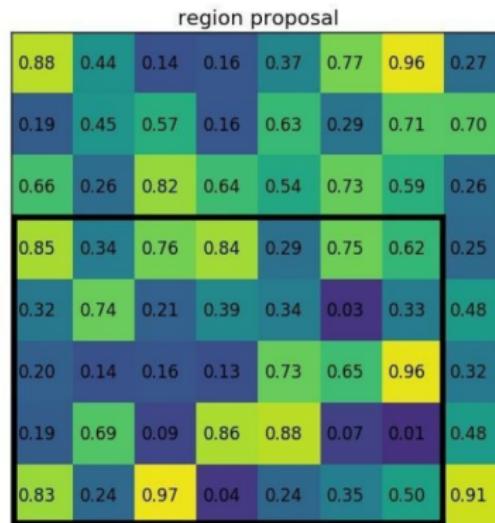
Region of interest pooling-Exemple

- Considérons un petit exemple pour comprendre comment cela fonctionne. Nous allons effectuer un Region of Interest (RoI) pooling sur une seule carte de caractéristiques de 8×8 , une région d'intérêt, et une taille de sortie de 2×2 . Notre carte de caractéristiques d'entrée ressemble à ceci :

input								
0.88	0.44	0.14	0.16	0.37	0.77	0.96	0.27	
0.19	0.45	0.57	0.16	0.63	0.29	0.71	0.70	
0.66	0.26	0.82	0.64	0.54	0.73	0.59	0.26	
0.85	0.34	0.76	0.84	0.29	0.75	0.62	0.25	
0.32	0.74	0.21	0.39	0.34	0.03	0.33	0.48	
0.20	0.14	0.16	0.13	0.73	0.65	0.96	0.32	
0.19	0.69	0.09	0.86	0.88	0.07	0.01	0.48	
0.83	0.24	0.97	0.04	0.24	0.35	0.50	0.91	

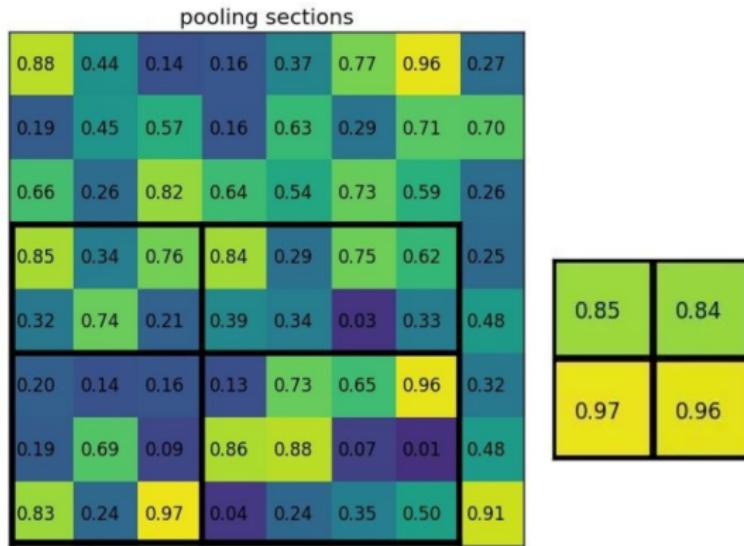
Region of interest pooling — exemple

- Supposons que nous ayons également une proposition de région (coordonnées en haut à gauche, en bas à droite) : (0, 3), (7, 8). Dans l'image, cela ressemblerait à ceci :



Region of interest pooling — exemple

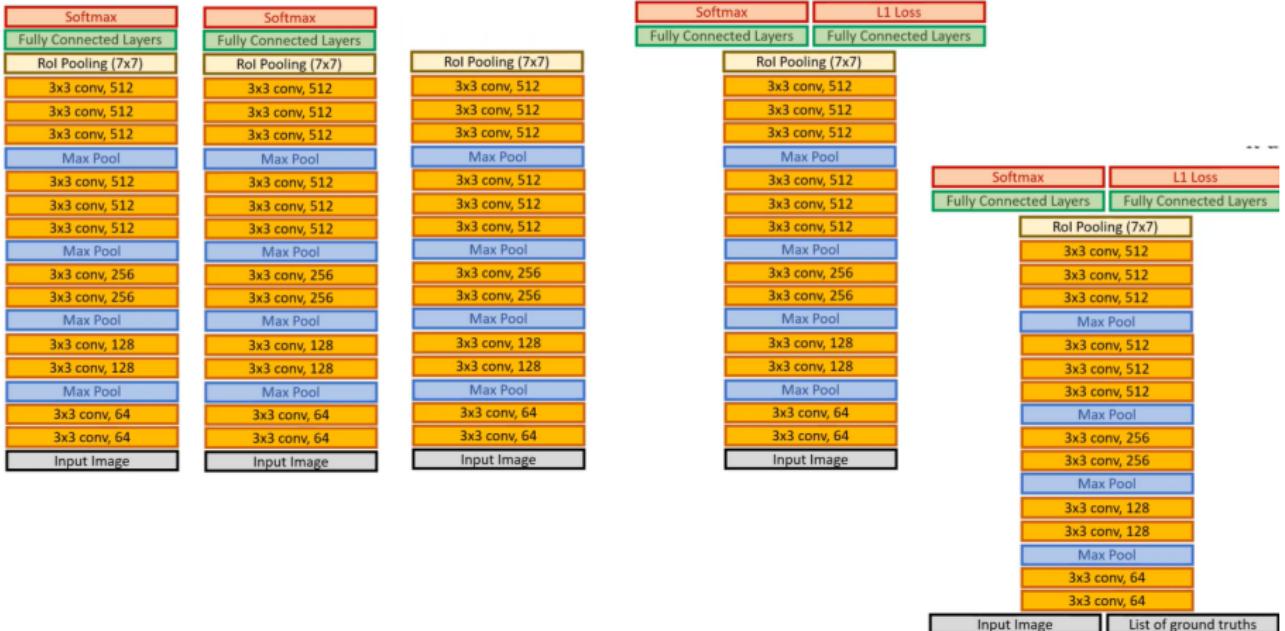
- Par souci de simplicité, nous ne conservons ici qu'une seule carte des caractéristiques de la proposition.
- En le divisant en (2×2) sections (car la taille de la sortie est de 2×2), nous obtenons :



Fast R-CNN 's stages

- ➊ Tout d'abord, il s'agit de pré-entraîner un réseau neuronal convolutionnel à des tâches de classification d'images.
- ➋ Proposer des régions en utilisant la recherche sélective (environ 2000 candidats par image).
- ➌ Modifier le CNN pré-entraîné :
 - Remplacer la dernière couche de mise en commun maximale du CNN pré-entraîné par une couche de mise en commun de la RoI. La couche de mise en commun des RoI produit des vecteurs de caractéristiques de longueur fixe pour chaque proposition de région. Le partage du calcul CNN est efficace..
 - Remplacer la dernière couche entièrement connectée et la dernière couche softmax par une couche entièrement connectée et une couche softmax sur $K + 1$ classes.
- ➍ Enfin, le modèle se divise en deux couches de sortie :
 - Un classificateur softmax sur $K + 1$ classes (similaire à R-CNN, où +1 représente la classe "d'arrière-plan"), fournissant une distribution de probabilité discrète par RoI.
 - Un modèle de régression de la boîte englobante qui prédit les décalages par rapport au RoI d'origine pour chacune des classes K .

Architecture de Fast R-CNN



Fast R-CNN, loss function (1)

La fonction loss totale pour une image peut être définie comme suit :

$$L(p, u, t, v) = L_{\text{cls}}(p, u) + \lambda \cdot \mathbf{1}_{[u \geq 1]} \cdot L_{\text{reg}}(t, v)$$

avec :

- L_{cls} : La classification loss, calculée en utilisant une softmax sur les probabilités de classe.
- L_{reg} : la perte de régression des boîtes englobantes (ou perte Smooth L1).
- p : Les probabilités de classification prédites pour chaque classe.
- u : La classe réelle de l'objet (label).
- t : Les coordonnées prédites de la boîte englobante.
- v : Les coordonnées réelles de la boîte englobante.
- λ : Un facteur d'équilibrage qui ajuste l'importance relative des deux pertes.
- $\mathbf{1}_{[u \geq 1]}$: une fonction indicatrice qui active la regression loss uniquement pour les boîtes contenant des objets (où $u \geq 1$).

Fast R-CNN, loss function (2)

- La classification loss, ou softmax loss, permet au modèle de déterminer la classe de l'objet détecté. Cette loss est calculée en utilisant l'entropie croisée (cross-entropy loss) :

$$L_{\text{cls}}(p, u) = -\log(p_u)$$

où p_u est la probabilité prédictive de la classe réelle u .

Fast R-CNN, loss function (3)

- La regression loss est basée sur la fonction Smooth L1 qui est moins sensible aux valeurs aberrantes que la fonction L2 classique (quadratique), ce qui rend l'apprentissage plus stable. La fonction Smooth L1 est définie comme :

$$L_{\text{reg}}(t, v) = \text{smooth}_{L_1}(t - v)$$

où :

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{si } |x| < 1 \\ |x| - 0.5 & \text{sinon} \end{cases}$$

- Ce terme permet au modèle de faire des ajustements précis pour la localisation des objets détectés.

Faster R-CNN

- Définir clairement le modèle Faster R-CNN.

Références bibliographique

- ① Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, <https://arxiv.org/abs/1311.2524>.
- ② Ross Girshick. "Fast R-CNN." Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, <https://arxiv.org/abs/1504.08083>.
- ③ Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." Advances in Neural Information Processing Systems (NeurIPS), 2015, <https://arxiv.org/abs/1311.2524>.