Apprentissage hiérarchique simultané pour la détection efficace d'objets

Hamidreza Odabai Fard, Mohamed Chaouch, Quoc-Cuong Pham, Antoine Vacavant, Thierry Chateau

Apprentissage hiérarchique simultané pour la détection efficace d'objets

H. Odabai Fard[1,2] M. Chaouch[1] Q.-C. Pham[1] A. Vacavant[2] T. Chateau[3]

[1] CEA, LIST, Laboratoire Vision & Ingénierie des Contenus, France
[2] Institut Pascal, UMR6602, CNRS, Université de Blaise Pascal, Clermont-Ferrand, France
[3] Image Science for Interventional Techniques, UMR 6284, CNRS, Université d'Auvergne, France

hamidreza.odabaifard@cea.fr

Résumé

Dans cet article, nous présentons une nouvelle approche de détection multi-classes basée sur un parcours hiérarchique de classifieurs appris simultanément. Pour plus de robustesse et de rapidité, nous proposons d'utiliser un arbre de classes d'objets. Notre modèle de détection est appris en combinant les contraintes de tri et de classification dans un seul problème d'optimisation. Notre formulation convexe permet d'utiliser un algorithme de recherche pour accélérer le temps d'exécution. Nous avons mené des évaluations de notre algorithme sur les benchmarks PASCAL VOC (2007 et 2010). Comparé à l'approche un-contre-tous, notre méthode améliore les performances pour 20 classes et gagne 10x en vitesse.

Mots Clef

Apprentissage hiérarchique, détection rapide, SVM structuré.

Abstract

In this paper, we present a novel approach for multi-class object detection based on an efficient inference of a jointly learned hierarchy of classifiers. For better performance and speed, we suggest to use a hierarchy of classes. Our object detection model is learned jointly combining ranking and classification constraints into one optimization problem. Our convex forumulation allows to use a tree search algorithm to speed up run-time. The evaluation is carried out on the PASCAL VOC (2007 and 2010) dataset. Compared to a One-Versus-All, we can improve detection performance for 20 classes and gain a speed-up of 10x.

Keywords

Hierarchical learning, fast detection, structured SVM.

1 Introduction

La détection d'objet de différentes classes dans les images présente plusieurs difficultés. L'algorithme d'apprentissage doit pouvoir traiter de données avec des variations inter et intra-classe. De plus, le temps d'exécution ne doit pas croître de manière exponentielle avec le nombre de classes.

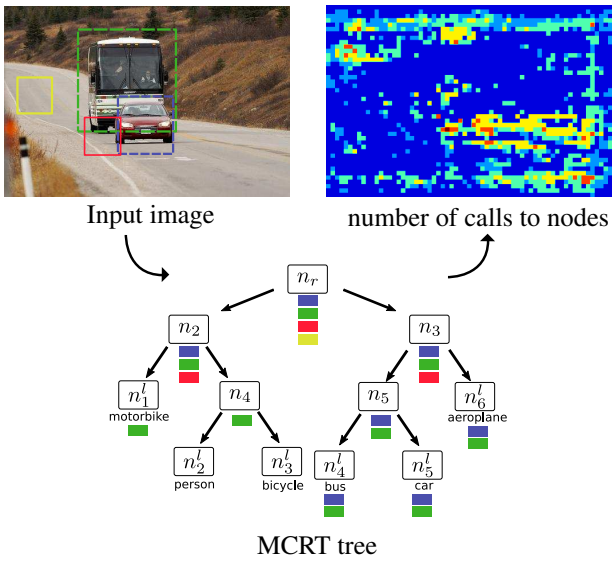


FIGURE 1 – Cette figure illustre la traversée d'un arbre pour les régions d'une image indiquées en haut-gauche . Le fond en jaune est rejeté après le passage par n_r. Celui en rouge est évaluée par plus de filtres. La carte de chaleur illustre le nombre de filtres évalués pour chaque pixel de l'image en entrée où le bleu signifie qu'un seul filtre est appliqué.

Dans ce travail, nous proposons une approche nommée *multi-class classification and ranking tree* MCRT améliorant la performance comparée à un-contre-tous SVM (OvA). Le choix du descripteur est transparent pour notre algorithme.

Pour accroître la performance de détection nous regroupons les classes de manière hiérarchique, ce qui permet de partager des exemples entre des classes. La ligne de décision finale est non-linéaire à travers les nombreux classifieurs. Pour éviter de créer un nœud supplémentaire pour la classe *fond*, nous combinons des contraintes de classification et de tri. Pendant la détection, MCRT trie

les différentes classes et si le meilleur score est négatif la région est classifiée comme étant du fond. Une autre contribution est l'utilisation d'un algorithme de recherche rapide inspirée de A*, pour trouver la bonne classe dans l'arbre. A chaque niveau de l'arbre, le chemin produisant le plus grand score est calculée grâce à une heuristique dédiée. Cette approche est différente du modèle *coarse-to-fine* car les étages interagissent ensemble.

Cet article est organisé comme suit : la section 2 présente les travaux relatifs à notre problématique. La section 3 introduit la notion et le système de reconnaissance puis la section 4 décrit l'algorithme se basant sur ces éléments. La section 5 nous évaluons notre technique avant de conclure.

2 État de l'art

Ces dernières années, un grand intérêt est prêté à la classification hiérarchique multi-classe. Les algorithmes nécessitent de traiter plus de données et de gérer la complexité d'apprentissage et de détection. Par la suite, nous passons en revue quelques travaux pertinents de l'état de l'art, très vaste sur ce sujet.

Détection d'objet L'idée de partage des exemples, en combinaison avec une approche de type "boosting", a été proposé par Torralba *et al.*[14]. Des classifieurs communs entre classes permettent de contribuer aux scores d'un sous-ensemble de classes. Salakhutdinov *et al.*[12] a proposé de regrouper les classes en fonction de leurs similarités et des distributions de leurs exemples. Les filtres de l'arbre sont entraînes avec L-SVM [7]. Contrairement à notre approche, le temps de détection est proportionnel aux nombre de nœuds dans l'arbre.

Au lieu de travailler au niveau des *caractéristiques* partagées par les exemples, d'autres approches combinent directement les parties communes entre les catégories. Razavi *et al.*[11] créent un dictionnaire commun où les parties votent pour les différentes classes. Les auteurs de [4] utilisent le modèle par parties déformables en appliquant une fonction de hachage pour accélérer le calcul du produit scalaire entre les modèles de parties et l'image. La complexité de leur approche est indépendante du nombre de classes mais ne permet pas de choisir entre la performance de détection et rapidité d'éxécution.

Classification hiérarchique avec SVM La classification hiérarchique a été notamment appliquée pour la classification des images. Par exemple les travaux [2, 8] ont explorés les arbres de décisions pour faire une rapide catégorisation des images. L'apprentissage de filtres binaires s'effectue selon une stratégie *top-down*. Dans [5], l'arbre et ces filtres sont appris ensemble. Structured SVM [15] est une approche pour apprendre tous ces filtres simultanément. Ces approchent ne peuvent pas être appliquées dans notre cas car il n'existe pas de solution pour modéliser explicitement la classe fond de la scène (c'est-à-dire les parties de l'image ne contenant aucun objet d'intérêt). Ainsi, nous proposons dans cet article un mode de classification avec une classe négative dominante (le fond) comme c'est le cas dans la détection d'objet.

3 Système

Nous voulons classifier chaque position dans une image I appartenant à une des k classes positives $\mathcal{Y}^+ \equiv \{y_1, \cdots, y_k\}$ ou attribuer l'étiquette du fond $y_{bg} = -1$. Nous proposons un modèle combinant les techniques de *tri* et de *classification*. Pour chaque région la bonne classe doit avoir le meilleur rang entre les k catégories. Si le score est négatif, l'hypothèse est classifiée comme *background*. Notre algorithme MCRT attribue à chaque position dans une pyramide de caractéristiques un score et une étiquette. Dans notre systéme, nous considérons que la pyramide de caractéristiques est l'ensemble de caractéristiques pour différentes échelles de l'image. Le score $score(x)$ pour une position x est calculé par le meilleur score de chaque classe

$$\text{score}(x) = \max_{y \in \{1,\ldots,k\}} \text{score}_y(x). \tag{1}$$

La classe prédite $\hat{y}$ est donnée par la fonction de décision finale $h : x \to y$ calculant une étiquette $\hat{y}$ pour chaque x :

$$\hat{y} = h(x) = \begin{cases} -1 & \text{, si score}(x) \leq 0 \\ \arg \text{score}(x) & \text{, sinon.} \end{cases} \tag{2}$$

Notre modèle de détection multi-classe est défini par un arbre *coarse-to-fine* où les classes sont les feuilles et les nœuds intermédiaires sont déterminés en regroupant les classes similaires de l'étage au-dessous. Chaque nœud intermédiaire est associé aux classes correspondant aux feuilles de l'arbre. Ceci permet le partage de caractéristiques entre les catégories. De plus, la combinaison des classifieurs de chaque noeud donne une ligne de séparation non-linéaire dans l'espace de caractéristiques facilitant la distinction des classes. Nous proposons d'accélérer l'inférence de l'arbre avec un algorithme de recherche pour trouver le chemin avec le plus grand score (*cf.* paragraphe 3.2).

3.1 Notre modèle de détection

Dans ce paragraphe, nous donnons des détails sur notre système de détection et introduisons les notations. Un arbre T représentant k classes est formellement défini par |T| nœuds où n_r est la racine, n_y^l est la feuille associée à la classe y. De plus, n_i $i \in \{1, \cdots, |\text{T}|\}$ désigne n'importe quel nœud dans l'arbre. La Fig. 1 illustre cette notation. De plus, nous notons anc(n_i) l'ensemble des ancêtres du nœud n_i incluant lui-même et desc(n_i) est l'ensemble des descendants du noeud n_i (n_i est exclu).

Un modèle de détection pour un arbre T est défini par |T| filtres $\{w_r, w_2, \cdots, w_{|\text{T}|}\}$, w_i étant un filtre pour le nœud i. Le vecteur de décision global w est défini par l'ensemble des w_i. Soit maintenant $\phi_i(x)$ le vecteur de caractéristiques du noeud n_i dans l'arbre et $\Phi_j(x)$ la concaténation de tous les vecteurs $\phi_i(x)$ des noeuds $n_i \in \text{anc}(n_j)$. Le score pour la classe y à la position x est la somme des scores de tous les filtres individuels sur son chemin :

$$\text{score}_y(x) = w^T \cdot \Phi_y^l(x). \tag{3}$$

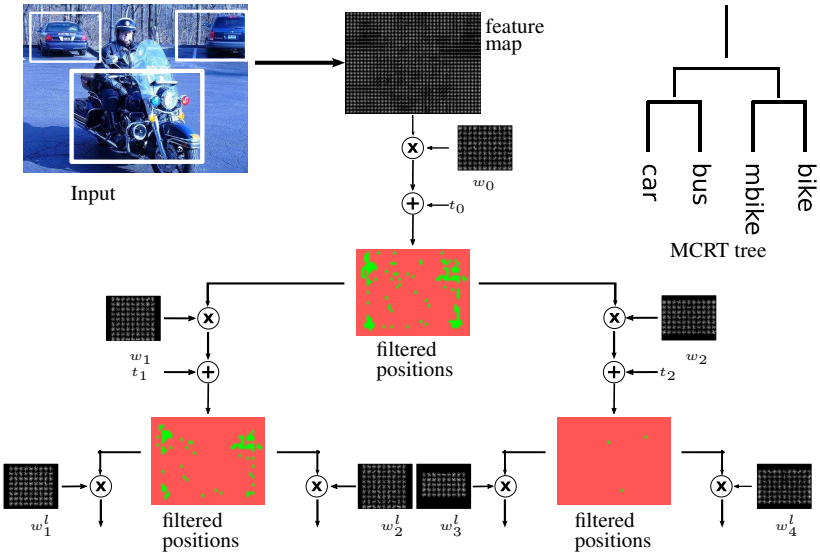


FIGURE 2 – Illustration des positions où les filtres de T sont appliqués (le rouge indique les positions sans évaluations).

La réponse finale définie par l'Eq. 2 est calculée par les scores individuels des classes données par l'Eq. 3. La combinaison de plusieurs filtres linéaires permet d'avoir une ligne de décision finale non-linéaire. On peut noter que la complexité de ce modèle est $\mathcal{O}(|\mathrm{T}|)$ tandis que le nombre de filtres $|\mathrm{T}|$ augmente de manière significative avec k.

3.2 Inférence rapide

Pendant l'évaluation de l'arbre, notre objectif est d'évaluer uniquement les nœuds se retrouvant sur le chemin vers la feuille de la bonne classe. Par exemple, seuls les filtres $\{w_r, w_2, w_3, w_5, w_6^l, w_4^l, w_5^l\}$ sont appliqués lorsqu'on évalue la région de la voiture dans la Fig. 1. Aussi, notre algorithme de recherche rejette le *fond* dans les premiers étages. La figure 2 visualise sur un exemple, les régions (en verts) où les filtres sont utilisés dans T.

Nous procédons de la manière suivante : Chaque nœud évalue ses enfants et additionne le score de leur chemin au score actuel. De plus, notre approche ajoute une estimation de l'importance de tous les poids des chemins restants. Ceci permet d'estimer le meilleur chemin de manière itérative. Nous gardons en mémoire ces estimations et choisissons progressivement le chemin ayant le plus grand score. Même si un chemin n'était pas emprunté dés le début, il se peut qu'il soit traversé plus tard.

Nous introduisons une fonction de classification $g(x, n_i)$:

$$g(x, n_i) = w^T \cdot \Phi_i(x) \tag{4}$$

et une heuristique t_i qui a une valeur constante définie pendant l'apprentissage (cf. paragraphe 4.4). L'estimation finale à chaque nœud est donnée par la fonction de gain

$$f(x, n_i) = g(x, n_i) + t_i. \tag{5}$$

La formulation (2) permet implicitement l'utilisation de cet algorithme de recherche pour accélérer l'évaluation car la classe détectée est donnée par le chemin le plus probable. Cette approche est inspirée par la méthode de recherche de plus court chemin A*. L'estimation est choisie pour être toujours plus grande ou égale au vrai score final. Il s'agit d'une estimation optimiste pour atteindre la bonne classe. Ainsi, t_i est admissible et notre algorithme garantit de trouver le chemin optimal.

4 Apprentissage hiérarchique

Dans ce paragraphe, nous présentons notre algorithme d'apprentissage hybride. Il apprend de manière automatique la structure de l'arbre, les dimensions des nœuds, leurs vecteurs de poids et les heuristiques.

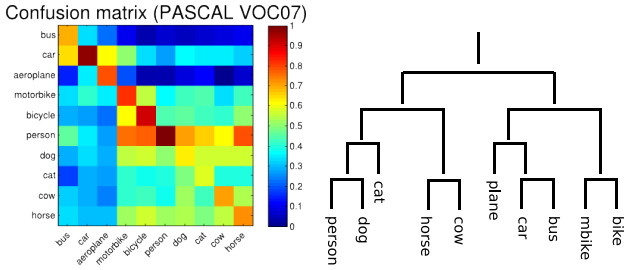
4.1 Construction de la taxonomie

L'erreur de la classification dépend du pouvoir discriminant des filtres individuels. Les classes qui peuvent être confondues sont regroupées ensemble pour améliorer l'apprentissage de ces filtres. Nous construisons d'abord une matrice de similartié $\mathcal{S} : k \times k$ qui mesure l'affinité s_{ij} entre les paires de classe $(y_i, y_j) \in \mathcal{Y}^+ \times \mathcal{Y}^+$ sur la base de validation. L'affinité s_{ij} est la médiane des valeurs obtenues en classifiant les exemples de la classe y_i avec un détecteur de la classe y_j. Ce détecteur est un simple détecteur HOG, mais nous pourrions employer n'importe quel autre type de filtre dans notre algorithme. De manière plus générale pour un nœud n_i, ses deux enfants (n_{c_1}, n_{c_2}) sont choisis pour minimiser la similarité inter-classe :

$$(n_{c_1}, n_{c_2}) = \arg\min_{\tilde{n}_{c_1}, \tilde{n}_{c_2}} \{\mathtt{sim}(\tilde{n}_{c_1}, \tilde{n}_{c_2}) \mid \text{toutes comb. } (\tilde{n}_{c_1}, \tilde{n}_{c_2})\}. \tag{6}$$

$\mathtt{sim}(n_{c_1}, n_{c_2}) = \sum_{y_i \in n_{c_1}} \sum_{y_j \in n_{c_2}} s_{ij}$ est la similarité entre des super-classes. Ce problème est résolu de manière hiérarchique avec une classification spectrale (spectral clustering) [13] sur la matrice de similarité. Cette approche

a l'avantage de comparer les caractéristiques sur plusieurs échelles et d'être indépendante du domaine d'application. La Fig. 3 montre un exemple de matrice de similarité et de sa hiérarchie.



(a) (b)

FIGURE 3 – (a) La matrice de similarité $\mathcal{S}$ pour $k = 10$. (b) L'arbre résultant de la matrice de similarité $\mathcal{S}$.

4.2 Formulation du problème

Nous proposons une approche combinant le tri et la classification regroupés dans un seul problème d'optimisation. Etant données n^+ exemples positifs et n^- exemples négatifs, notre but est de minimiser le risque empirique sur la base d'apprentissage. Ceci conduit au programme quadratique avec les contraintes linéaires suivantes :

$$\min_{w,\xi_i(j)\geq 0} \frac{1}{2}\|w\|^2 + C\left(\sum_{i=1}^{n^+}(\xi_i + \sum_{j=1}^{n^+}\xi_{ij}) + \sum_{i,j}^{n^-\times n^+}\xi_{ij}\right) \quad (7a)$$

$$avec \ \forall y_i \in \mathcal{Y}^+, \forall y_j \in \mathcal{Y}^+ : w^T \cdot \delta\Phi_i(y_j) \geq 1 - \xi_{ij} \quad (7b)$$

$$: w^T \cdot \Phi^l_{y_i}(x_i) \geq 1 - \xi_i \quad (7c)$$

$$\forall y_i \in \{y_{bg}\}, \forall y_j \in \mathcal{Y}^+ : -w^T \cdot \Phi^l_{y_j}(x_i) \geq 1 - \xi_{ij}, \quad (7d)$$

où $\delta\Phi_i(y) = \Phi^l_{y_i}(x_i) - \Phi^l_y(x_i)$. $w = (w_r, \cdots, w_{|\mathrm{T}|})$ est la concaténation des vecteurs de poids de chaque nœud dans l'arbre T. La fonction objectif (7a) souvent appliquée avec des SVMs est soumise à deux types de contraintes : (i) contraintes de classification (7c,7d) forçant les poids du filtre global à rejeter les négatifs ; (ii) contraintes de tri (7b) entre les exemples de classes positives assurant que le plus grand score soit calculé pour le bon chemin dans l'arbre T.

4.3 Optimisation avec des plans sécants

Nous utilisons la méthode des plans sécants [9] pour réduire le temps d'apprentissage. On reformule (7) en utilisant une seule variable ξ pour toutes les contraintes donnant le problème d'optimisation suivant :

$$\min_{w,\xi\geq 0} \frac{1}{2}\|w\|^2 + C\xi \quad (8a)$$

$$avec \ \forall(\bar{y}_1, \cdots, \bar{y}_n) \in \mathcal{Y}^{+^n} :$$
$$\frac{1}{n}\{\sum_{i=1}^{n^+} w^T \cdot (\delta\Phi_i(\bar{y}) + \Phi^l_{y_i}(x_i)) \quad (8b)$$
$$-\sum_{i=1}^{n^-} w^T \cdot \Phi^l_{\bar{y}}(x_i)\} \geq 1 - \xi.$$

Le nombre de contraintes augmente de manière exponentielle avec n. Mais l'Eq. 8 peut être résolue efficacement en construisant itérativement un sous-ensemble de contraintes. Contrairement à la formulation (7), seulement une contrainte est ajoutée de manière incrémentale au sous-ensemble. L'algorithme procède de la manière suivante :

1. initialisation du sous-ensemble de contraintes $\mathcal{W} \leftarrow \emptyset$: Au lieu d'ajouter toutes les contraintes, on ajoute une contrainte après l'autre.
2. trouver la contrainte la plus forte : Pour chaque exemple x_i de la classe y_i on cherche une autre classe $\bar{y}_i \in \mathcal{Y}^+ \setminus y_i$ qui lui est le plus proche : $\bar{y}_i = \arg\max_{y\in\mathcal{Y}^+} 1 + w^T \cdot \Phi^l_y(x_i)$.
3. Ajout de la nouvelle contrainte au sous-ensemble : $\frac{1}{n}\{\sum_{i=1}^{n^+}(w^T \cdot \Phi_{y_i}(x_i) - \max(w^T \cdot \Phi_{\bar{y}_i}(x_i), 0) - \sum_{i=1}^{n^-} w^T \cdot \Phi^l_{\bar{y}_i}(x_i)\} \geq 1 - \xi \rightarrow \mathcal{W}$.
4. Optimisation du problème (8) en utilisant un *solver* quadratique sur $\mathcal{W}$.
5. Répétition des étapes (1)-(4) jusqu'à ce que la solution $(w, \xi + \epsilon)$ satisfasse la contrainte 8b.

L'étape 2 cherche la classe $\bar{y}_i$ qui est la plus proche de l'exemple (x_i, y_i). L'étape 3 ajoute la somme de la contrainte la plus forte au sous-ensemble $\mathcal{W}$ considérant les contraintes de classification et de tri.

4.4 Détermination des heuristiques

Nous cherchons à déterminer les heuristiques les plus strictes tout en évitant les erreurs sur la base d'apprentissage. Chaque heuristique t_i du nœud n_i est déterminée de haut vers le bas et de gauche à droite. Ces valeurs sont des estimations optimistes du vrai gain du score de la classe correcte.

Chaque annotation i des k classes est associée à une image I_i et une position. Soit D_i un ensemble de détections dans I_i où chaque élément x correspond à une instance valide. Une instance valide est une région correctement détectée $y_i = \hat{y}_i = arg\max g(x, n^l_y)$ et ayant un recouvrement δ_o suffisant avec l'annotation de I_i.

$$D_i = \{\forall x \in I_i \mid \hat{y} = y_i \wedge \ \texttt{intersection}(x, I_i) \geq \delta_o\}. \quad (9)$$

De plus, soit D l'ensemble de toutes les détections valides $D = \{D_o, \cdots, D_{n^+}\}$. Alors, t_i est donné par :

$$\forall(x, y) \in D : t_j \geq \min \sum_{n_p \in \text{chemin}(n_j, n_y)} \underbrace{w(n_j)^T \cdot \phi_p(x)}_{\text{score de filtres individuels}}, \quad (10)$$

où chemin(n_j, n_y) est l'ensemble de nœuds sur le chemin de n_i à n_j. A chaque nœud, nous supprimons les instances

multiples ayant un score strictement plus grand que t_i. Toutefois, il reste toujours au moins une détection x pour chaque annotation I_i :

$$\forall i, \exists (x,y) \in D_i \text{ et } n_c \in \text{chemin}(n_r, n_y^l), n_{\bar{c}} \in \text{T} : f(x, n_c) \geq f(x, n_{\bar{c}}),$$

avec $f(x, n_i)$ la fonction de gain (*cf.* paragraphe 3.2).
Dans notre étude, nous avons considéré des heuristiques admissibles par rapport aux données d'apprentissage. Nous pouvons également choisir des heuristiques non admissibles en retirant un certain pourcentage des meilleures détections. Cela produit des heuristiques encore plus strictes accélérant le processus de détection au coût de la performance de la détection.

5 Résultats

Dans cette section, nous évaluons notre approche sur les *benchmarks* PASCAL VOC'07 et VOC'10 [6] en suivant leurs protocoles. Chaque base contient 20 classes avec plus de 12000 objets annotés. Elle est partagée de manière égale en base d'apprentissage et de test. Nous avons choisi 3 configurations pour différentes valeurs de $k = \{8, 10, 20\}$ pour nos expériences. Nous avons sélectionné les classes suivantes : {'bus', 'bicycle', 'motorbike', 'car', 'aeroplane', 'person', 'cow', 'horse', 'dog', 'cat'} où pour $k = 8$ nous prenons les 8 premières entrées. Parmi ces classes se trouvent des classes avec beaucoup de caractéristiques communes (*e.g.*'car' et 'bus') et d'autres moins (*e.g.*'person' et 'aeroplane'). Nous étudions l'influence (1) de mélanger les contraintes de classification et de tri, (2) de faire la détection avec une hiérarchie et (3) d'accélérer la détection avec l'algorithme de recherche dans l'arbre.

Modèles de détection Pour valider les différents aspects de notre méthode, nous comparons la performance de 5 algorithmes de détection (cf. Tab. 1) : (1) Un-contre-tous (OvA) optimise chaque détecteur indépendemment. Les fonctions de décisions finales sont calibrées [10]. Cette stratégie de détection applique k *classifieurs* binaires et conserve le score le plus élevé. (2) Le modèle MCR (multiclass ranking) apprend simultanément tous les classifieurs sans hiérarchie en mixant les contraintes de classification et de tri. (3) Le modèle MCRT apprend une hiérarchie et utilise des heuristiques admissibles pour traverser l'arbre. (4) *f*MCRT est la version rapide. Contrairement à MCRT, cette approche utilise des heuristiques "moins admissibles" obtenant la même performance de détection que OvA tout en étant considérablement plus rapide. (5) Le modèle *e*MCRT applique de manière exhaustive tous les filtres de l'arbre. Le choix des caractéristiques ne dépend pas de notre formulation. Nous avons adapté l'outil SVMStruct [9] pour inclure la définition des contraintes et pour apprendre tous les modèles. Nous avons choisi l'histogramme des gradients orientés (HOG [3]) fournis par [7] comme *caractéristiques* très reconnus pour leur efficacité. Ils sont extraits autour du centre d'une fenêtre glissante dont les dimensions sont définies par chaque nœud.

TABLE 1 – Propriétés des modèles de détection.

Modèle	Classif.	Tri	Hierarchie	Inférence
OvA	✓			$\|\text{T}\|$
MCR	✓	✓		$\|\text{T}\|$
MCRT	✓	✓	✓	$< \|\text{T}\|$
*f*MCRT	✓	✓	✓	$\ll \|\text{T}\|$
*e*MCRT	✓	✓	✓	$\|\text{T}\|$

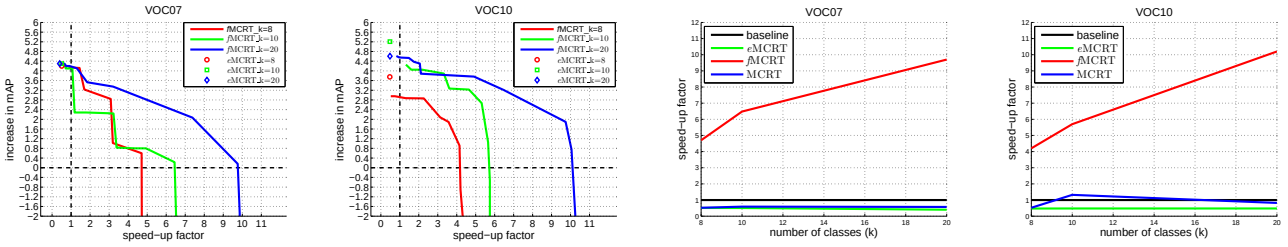
Performance de détection Le Tab. 2 résume les performances de détection en mAP (*mean average precision*) des méthodes pour différents nombres de classe k. MCR atteint des performances similaires à OvA. L'apprentissage simultané des contraintes de *tri* et de classification entre les classes d'objet et le *fond* permet de trouver une ligne de décision stable. Les meilleurs résultats sont obtenus avec *e*MCRT, une extension de MCR incluant la hiérarchie. L'augmentation du nombre de filtres linéaires et le partage des caractéristiques améliore de manière significative les performances globales de détection.

Complexité de détection Pour VOC'07 les bases d'apprentissage et de validation sont utilisées pour l'apprentissage et la base test pour l'évaluation. Pour VOC'10 la partie apprentissage est utilisée pour apprendre et la partie validation pour l'évaluation [1]. Les Fig. 4a et 4b montrent le compromis entre mAP et le gain en vitesse par rapport au OvA (méthode de référence). On note tout d'abord que l'utilisation de l'algorithme de recherche permet d'ajuster le compromis entre ces deux critères. Dans presque tous les cas, utiliser l'algorithme de recherche dans l'arbre produit des résultats similaires à l'évaluation de tous les nœuds avec des temps d'execution plus rapides sauf pour $k = \{8, 10\}$ dans VOC'10 (cela est probablement dû à un manque d'exemples d'apprentissage pour généraliser les heuristiques). Comparé à OvA, nous constatons une amélioration du temps de détection pour le même mAP. Pour $k = 20$ nous avons un facteur de gain 10 pour les 2 jeux de données. La technique de recherche dans l'arbre permet à la fois de rejeter les négatifs tôt dans la hiérarchie et de trouver le chemin vers la bonne classe pour les positifs. Les Fig. 4c et 4d présentent le gain relatif en vitesse pour les 4 méthodes comparées à OvA en fonction du nombre de classes k. Le coût d'exécution de *e*MCRT augmente en fonction de k. Ceci n'est pas surprenant car le nombre de nœuds dans l'arbre augmente rapidement avec k. MCRT utilise des heuristiques admissibles et est ainsi légèrement plus rapide. Le meilleur temps de détection est atteint par *f*MCRT qui, contrairement au *e*MCRT, réduit la charge de calcul pour de grandes valeurs de k. La Fig. 5 montre le nombre de nœuds appliqués pendant l'exécution par MCRT et *f*MCRT.

1. La politique de VOC'10 (transférer ses résultats par le site Web du *benchmark*) conduit à des temps d'évaluation trop longs pour cette partie.

TABLE 2 – Résultats des 4 modèles. Par manque d'annotations de tests pour VOC'10, ces tests ne sont pas fait pour *f*MCRT.

dataset	VOC07						VOC10					
k	8		10		20		8		10		20	
Evaluation	mAP	Speed	mAP	Speed	mAP	Speed	mAP	Speed	mAP	Speed	mAP	Speed
OvA	16.1	1x	12.9	1x	8.8	1x	13	1x	11.9	1x	7.7	1x
MCR	17.4	1x	14.7	1x	12.2	1x	12	1x	9.4	1x	7.5	1x
*e*MCRT	**20.3**	0.51x	**17.2**	0.51x	**13.1**	0.4x	**14**	0.55x	**14**	0.56x	**9**	0.48x
*f*MCRT	16.1	**4.7**x	12.9	**6.5**x	8.8	**9.7**x	-	-	-	-	-	-



(a) (b) (c) (d)

FIGURE 4 – (a,b) Le compromis entre mAP et gain en vitesse pour les 3 modèles comparé à OvA. (c,d) Le gain en vitesse des 3 modéles en fonction du nombre de classes k.

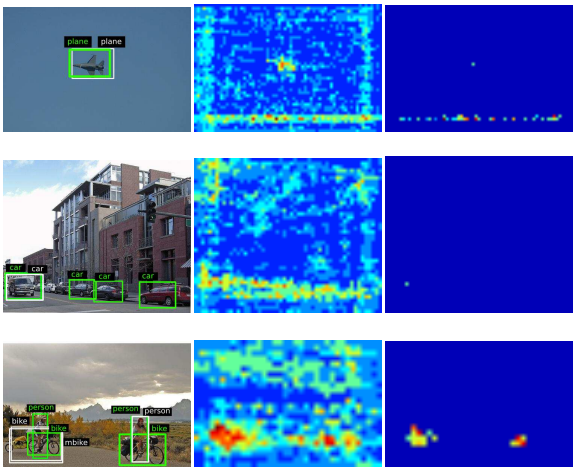


FIGURE 5 – Résultats obtenus avec MCRT et *f*MCRT. La colonne de gauche montre les résultats de détection. Les résultats de MCRT sont en vert et pour *f*MCRT en blanc. Les 2 autres colonnes montrent le nombre relatif de nœuds évalués par MCRT et *f*MCRT.

6 Conclusion

Nous avons présenté une méthode permettant d'accélérer la détection hiérarchique multi-classe. Cette tâche est formulée comme un problème de *tri* et de *classification*. Notre formulation de l'apprentissage permet naturellement d'utiliser notre algorithme de parcours d'arbre pour réduire le temps d'exécution. Nous pouvons choisir la performance de détection en fonction de contraintes de temps de calcul. D'autre part, les tests ont montrés que MCRT permet d'avoir des meilleurs résultats que OvA en sacrifiant la rapidité. Aussi, MCRT présente un *speed-up* significatif sans compromettre les performances de détection.

Le temps de calcul sur le CPU peut être encore réduit en combinant notre système avec un détecteur d'objet générique [1] pour filtrer des régions dans l'image. Nous sommes convaincus qu'en utilisant des descripteurs plus avancés comme les DPM [7], nous pouvons améliorer les résultats de l'état de l'art.

Références

[1] B. Alexe, T. Deselaers, and V. Ferrari. What is an object ? In *CVPR*, 2010.

[2] S. Bengio, J. Weston, and D. Grangier. Label embedding trees for large multi-class tasks. In *NIPS*, 2010.

[3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[4] T. Dean, M. A. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik. Fast accurate detection of 100000 object classes on a single machine. CVPR, 2013.

[5] J. Deng, S. Satheesh, A. Berg, and L. Fei-Fei. Fast and balanced : Efficient label tree learning for large scale object recognition. In *NIPS*, 2011.

[6] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2) :303–338, June 2010.

[7] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *PAMI*.

[8] G. Griffin and P. Perona. Learning and using taxonomies for fast visual categorization. 2008.

[9] T. Joachims, T. Finley, and C.-N. Yu. Cutting-plane training of structural svms. *Machine Learning*, 2009.

[10] J. C. Platt, N. Cristianini, and J. Shawe-taylor. Large margin dags for multiclass classification. 2000.

[11] N. Razavi, J. Gall, and L. J. V. Gool. Scalable multi-class object detection. In *CVPR*, 2011.

[12] R. Salakhutdinov, A. Torralba, and J. B. Tenenbaum. Learning to share visual appearance for multiclass object detection. In *CVPR*, 2011.

[13] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. PAMI*, 2000.

[14] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing visual features for multiclass and multiview object detection. *PAMI*, 2007.

[15] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *ICML*, 2004.