

Vedere Artificială (Computer Vision)

Bogdan Alexe

bogdan.alexe@fmi.unibuc.ro

anul 2, master Informatică, semestrul I, 2019-2020

Vedere Artificială - Tema 2

Regăsirea imaginilor folosind modelul Bag of Visual Words

Obiectiv

Scopul acestei teme este implementarea unui sistem de regăsire de imagini dintr-o bază de date. Sistemul primește ca date de intrare o imagine cu o anumită scenă specifică (clădire, monument, obiectiv turistic, în limba engleză se folosește denumirea de "landmark") și are ca date de ieșire toate imaginile din baza de date ce conțin acea scenă, fotografiată de obicei din unghiuri diferite. Sistemul ce urmează să îl implementați va folosi modelul Bag of Visual Words, prezentat la curs.

Descrierea datelor

Directorul *database* conține 50 de clase, fiecare având 10 imagini ale unei scene specifice. Figura 1 conține cele 10 imagini din clasa 45. În total sunt 500 de imagini care vor alcătui baza de date în care veți căuta imaginile de tip query.

Directorul *queries* conține 50 de imagini de tip query, fiecare astfel de imagine conține o scenă specifică din cele 50 și nu se regăsește în baza de date. Sistemul vostru va fi evaluat pe baza unor astfel de imagini. Ideal, pentru fiecare imagine de tip query sistemul vostru va întoarce în primele 10 imagini similară acele imagini din baza de date care înfățișează scena respectivă.



Figura 1: Clasa 45 din baza de date conține 10 imagini cu aceeași scenă.

Tema 2 – cerințe

Cerințe

Implementați sistemul de regăsire a imaginilor din baza de date cu 500 de imagini.

Tema valorează 10 puncte. Punctajul este împărțit astfel:

- implementarea corectă cu rezultate moderate bune ale sistemului (toate componentele prezentate fără aspectul computațional eficient și verificarea geometrică) - 6 puncte;
- performanță de timp bună pentru o imagine query (sub 5 secunde) - 1 punct;
- implementarea verificării geometrice cu rezultate mai bune ale sistemului - 2 puncte;
- din oficiu - 1 punct;

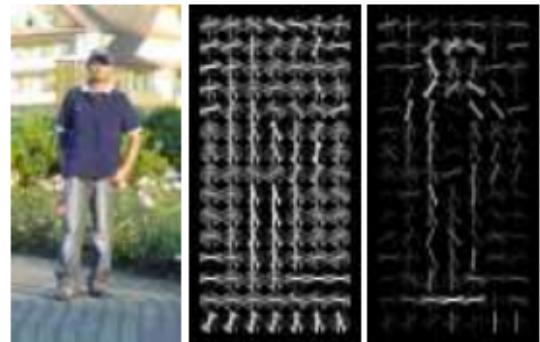
Termenul limită de prezentare al proiectului este joi, 6 februarie 2020.

Recapitulare – cursul trecut

- Glisarea unei ferestre

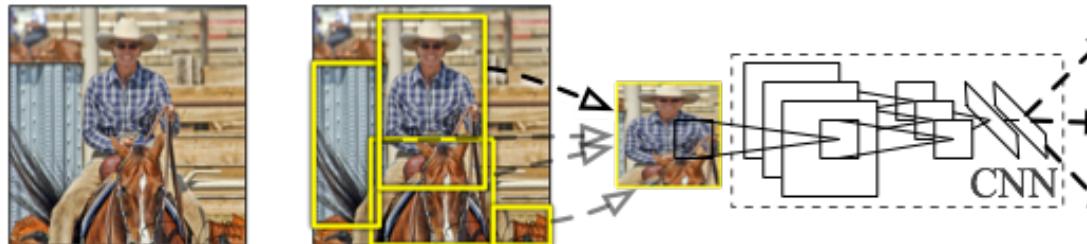


- HOG pentru detectarea oamenilor, detectare facială

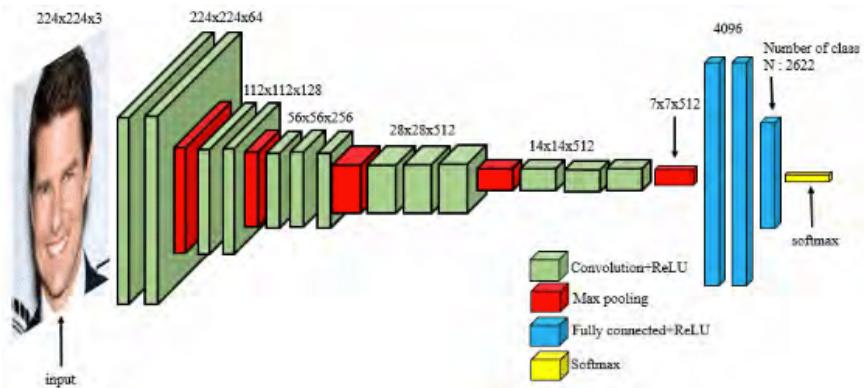
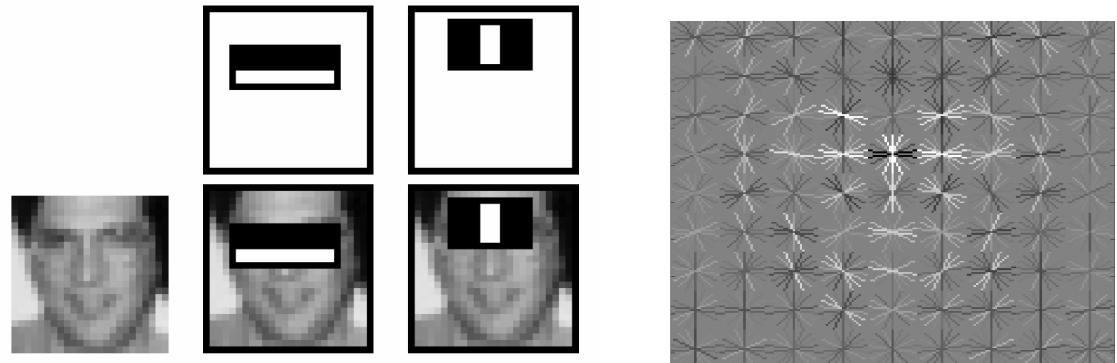


Cursul de azi

- Detectarea de obiecte în imagini cu R-CNN



- Detectare facială
 - Viola-Jones
 - HOG
 - CNN-uri



Evaluare – recall și precizie pe o imagine

Cum evaluăm performanța unui detector?

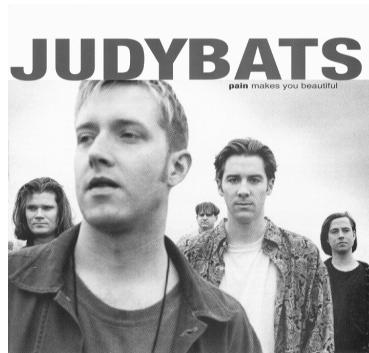
- comparăm ceea ce am obținut (ferestrele de maxim local se numesc detectii) cu ceea ce trebuia să obținem (ferestre adnotate conținând obiectele de interes)
- recall = câte fețe am găsit din cele adnotate =
$$\frac{\#\text{detectii adevarate}}{\#\text{ferestre adnotate}}$$
- precizie = câte detectii sunt adevărate =
$$\frac{\#\text{detectii adevarate}}{\#\text{detectii adevarate} + \#\text{detectii false}}$$



Ferestrele adnotate
Detectiile adevărate
Detectiile false
recall = $55/57 = 0.965$
precizie = $55/59 = 0.932$

Compararea a doi algoritmi

- se face pe numite baze de date pentru care se cunoaște ceea ce trebuia să întoarcă algoritmul de detectare facială (imaginile sunt adnotate)

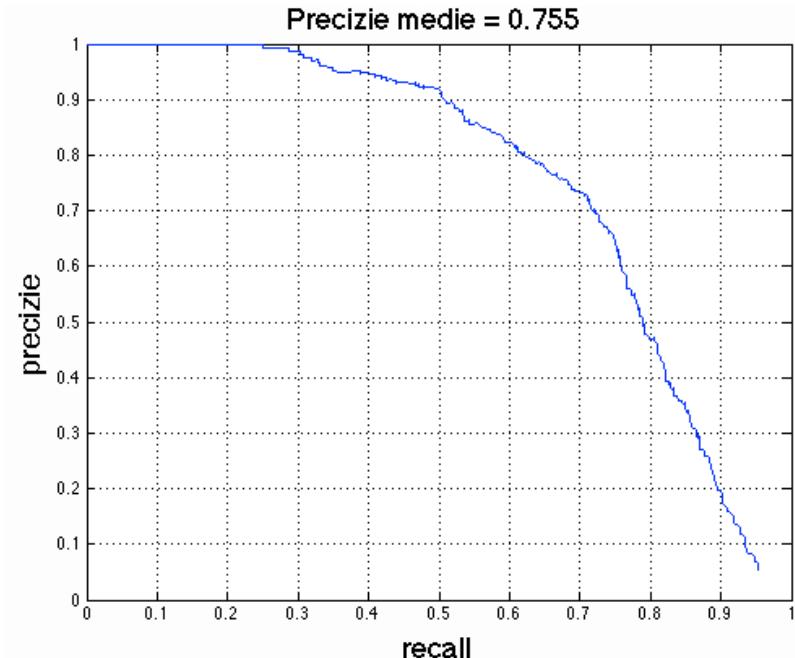


- se compară în funcție de precizie și recall sumarizate în graficul precizie-recall care oferă un număr: precizia medie a detectorului de fețe

Evaluare – recall și precizie pe toate imaginile

Se construiește graficul precizie-recall în felul următor:

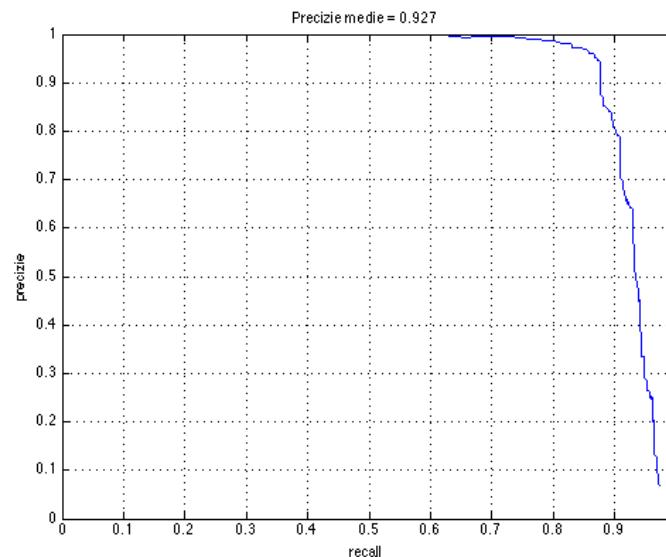
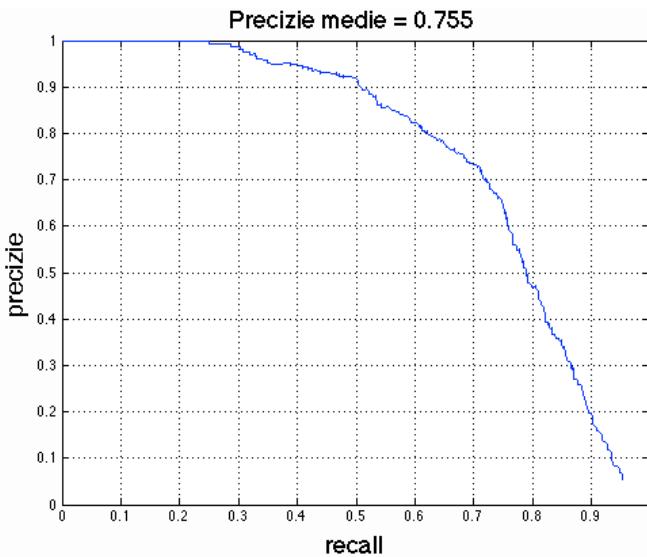
1. se rulează detectorul facial pe toate imaginile și se obține pentru fiecare imagine o mulțime de detectii;
2. se stabilește pentru fiecare imagine (pentru care avem adnotări) care sunt detectiile adevărate și cele false;
3. se ordonează descrescător toate detectiile în funcție de scorul lor;
4. pentru un threshold care descrește de la scorul cel mai mare la scorul cel mai mic se calculează precizia și recall-ul pentru toate detectiile cu scorul $>$ threshold



Valorea precizie medie sumarizează graficul precizie-recall prin aria de sub grafic
Pe baza acestei valori evaluăm performanța unui algoritm

Compararea a doi algoritmi

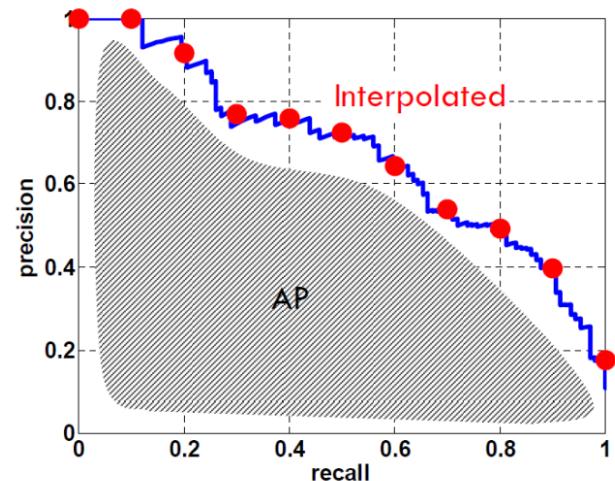
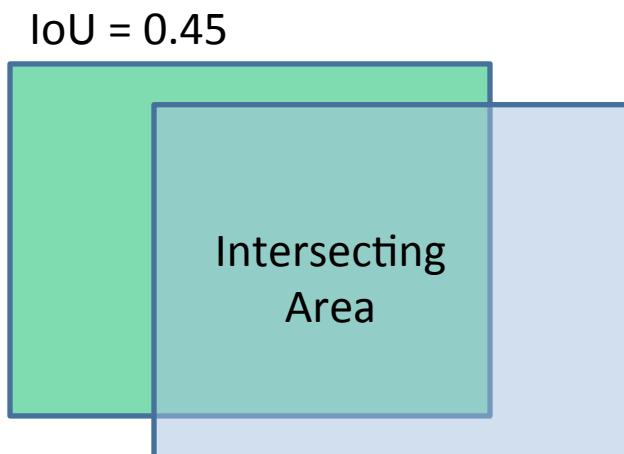
- se face pe numite baze de date pentru care se cunoaște ceea ce trebuia să întoarcă algoritmul de detectare facială (imaginile sunt adnotate)
- se compară în funcție de precizie, recall sumarizate în graficul precizie-recall care oferă un precizia medie (AP – Average Precision)



- algoritmul din dreapta este mai bun decât algoritmul din stânga

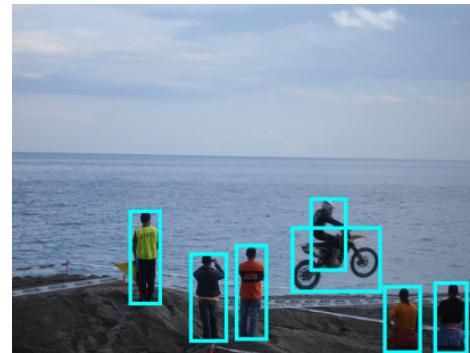
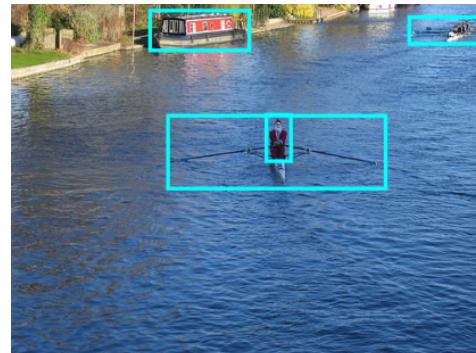
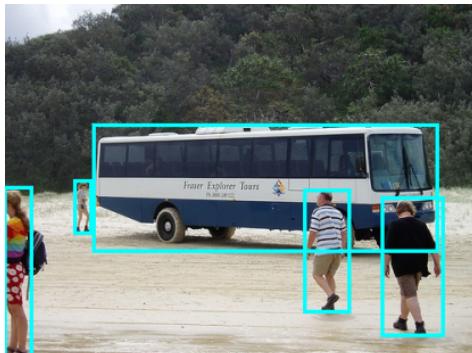
Evaluarea detectării de obiecte

- Seturi de date:
 - [PASCAL VOC](#) (2005-2012): 20 de clase, ~20,000 imagini
 - [MS COCO](#) (2014-prezent): 60 clase, ~300,000 imagini
 - [IMAGENET](#) (2012-prezent): 1000 clase, ~1,200,000 imagini
- Evaluare
 - output: pentru fiecare clasă, prezice ferestre (x_1, y_1, x_2, y_2) cu scoruri
 - metrică:
 - detecție adevărată: ≥ 0.5 (IoU) + nu e duplicat
 - graficul precizie-recall
 - Average Precision: aria de sub grafic (obținut prin interpolare)
 - medie după clase

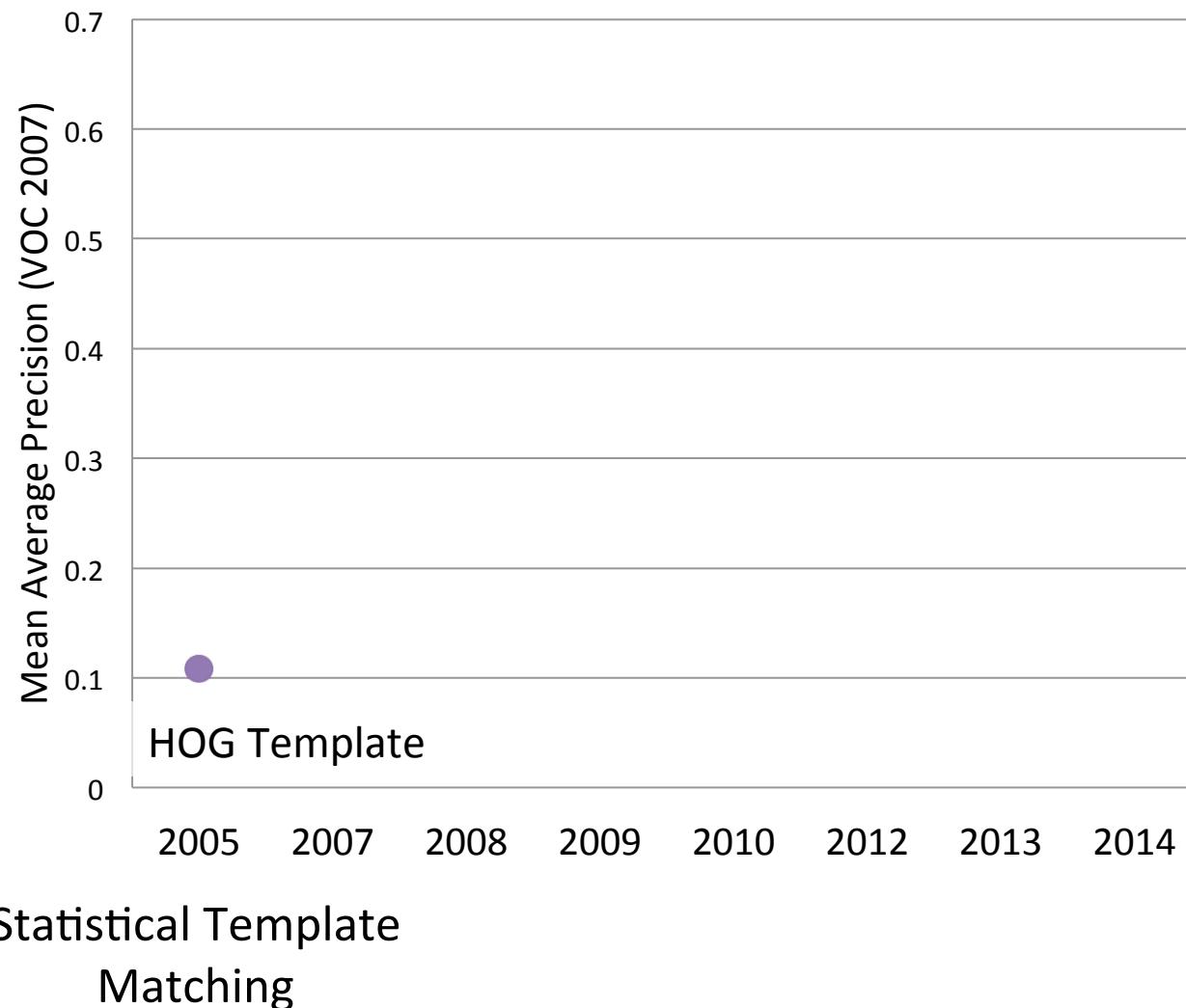


PASCAL VOC 2005-2012

- PASCAL Visual Object Classes Challenge
- 20 classes, ~10K images, ~25K annotated objects
- Training, validation, test data sets.
- Evaluation:
 - Average Precision (AP) per class
 - mean Average Precision

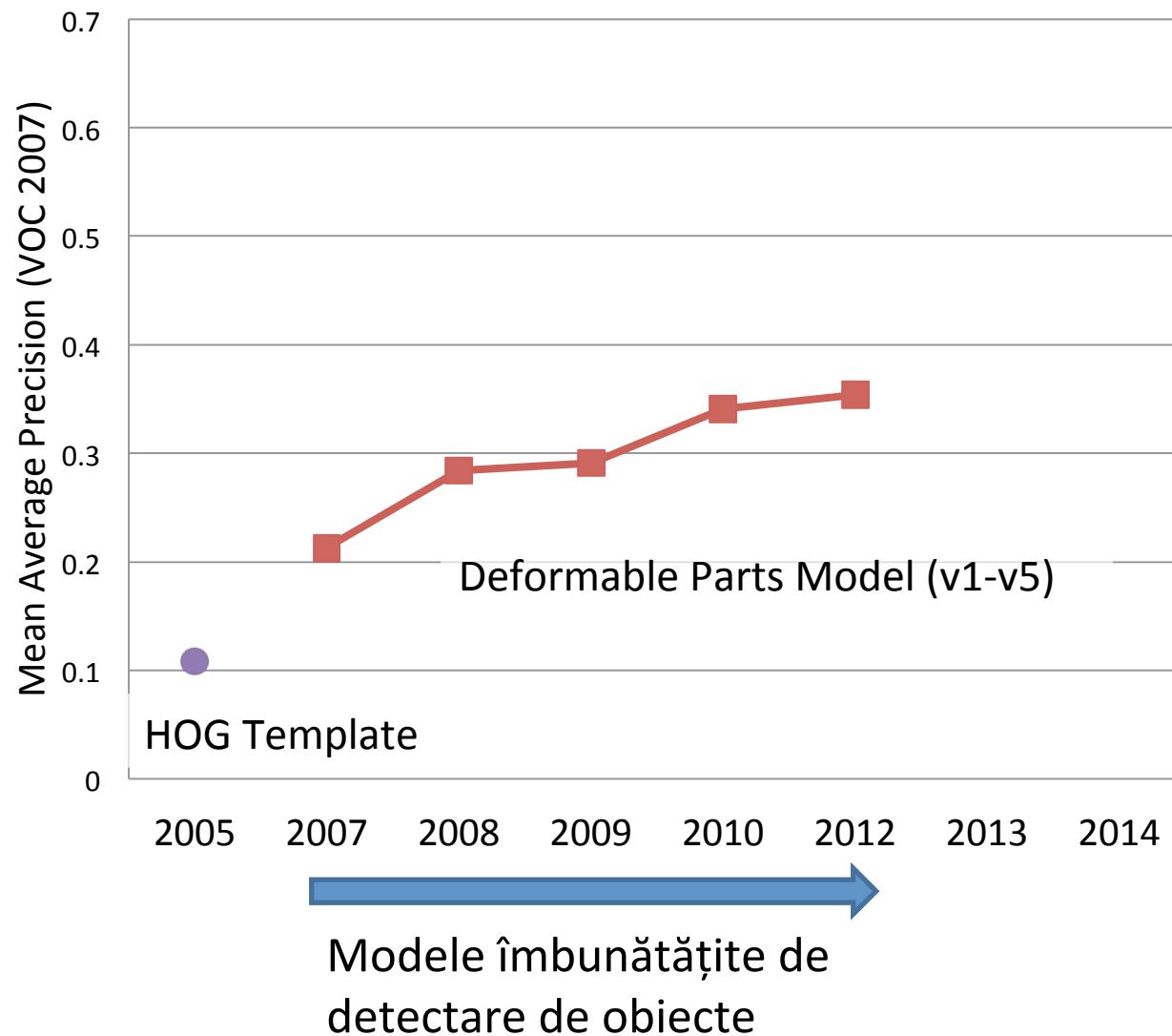


Evoluția performanței detectării de obiecte



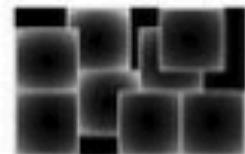
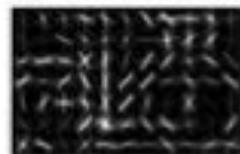
Statistical Template
Matching

Evoluția performanței detectării de obiecte



Modele deformabile cu părți

- folosește drept caracteristici HOG
- învățare discriminativă cu variabile latente (poziția părților)
- mAP: 33.4% - 33.7%

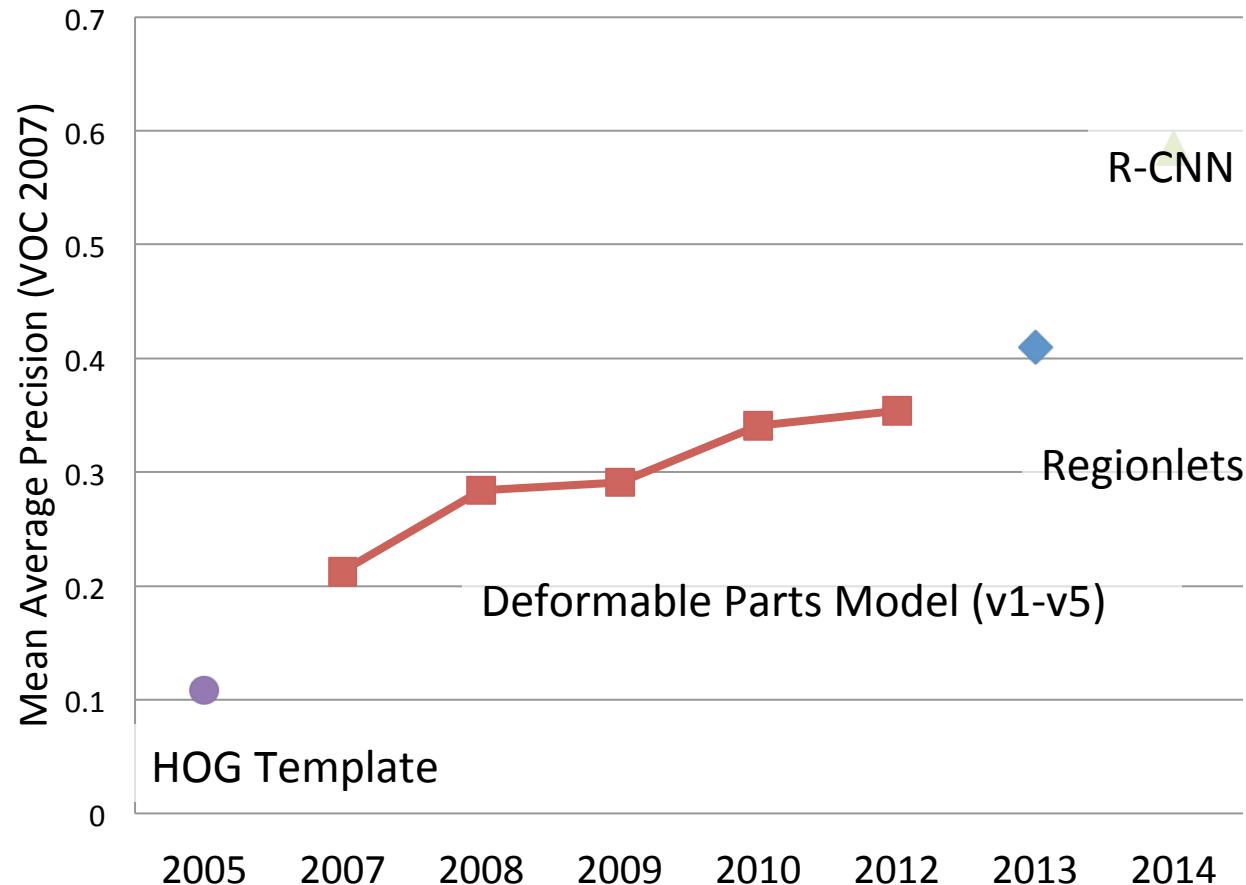


P.F. Felzenszwalb et al., "Object Detection with Discriminatively Trained Part-Based Models", PAMI 2010.

J.J. Lim et al., "Sketch Tokens: A Learned Mid-level Representation for Contour and Object Detection", CVPR 2013.

X. Ren et al., "Histograms of Sparse Codes for Object Detection", CVPR 2013.

Evoluția performanței detectării de obiecte



Punct cheie: învață caracteristici mai bune (discriminative) dintr-un volum de date mare (adnotate) și folosește la probleme cu puține date

Caracteristici mai bune

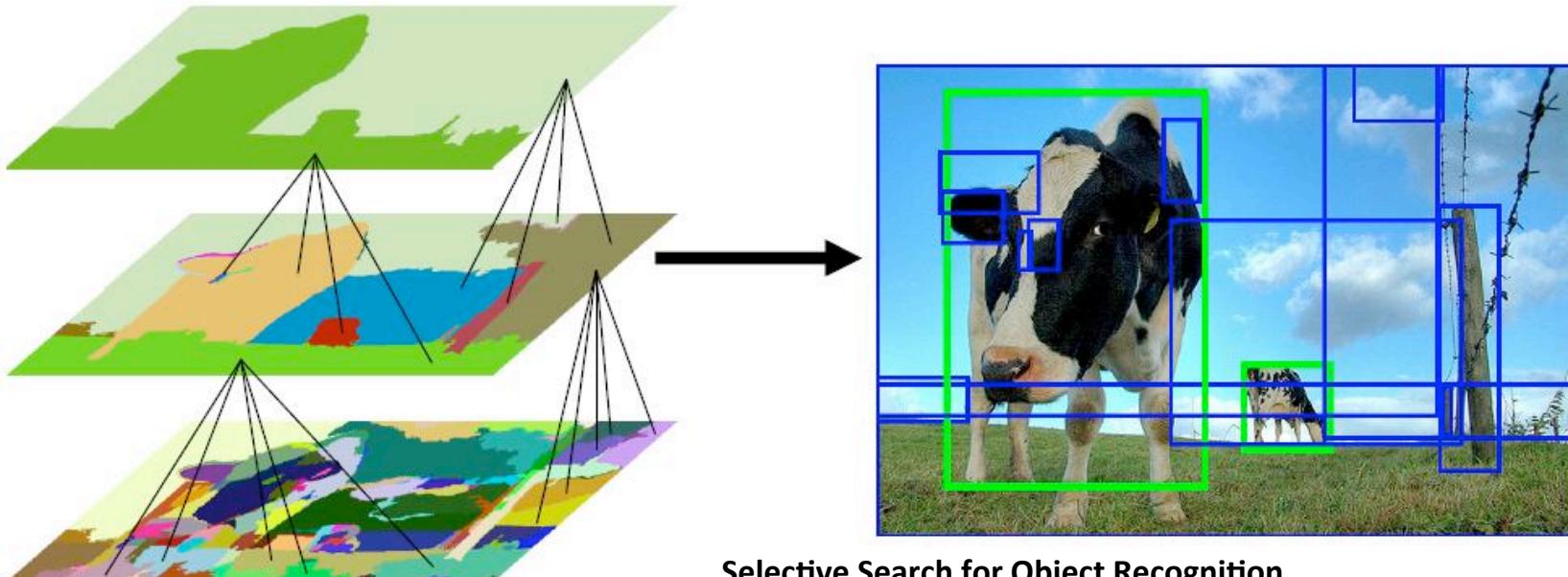
Evoluția performanței - glisare a ferestrei

	VOC 2007	VOC 2010
DPM v5 (Girshick et al. 2011)	33.7%	29.6%

Sistem de referință

Alternativă pentru glisarea ferestrei: propuneri candidat

- segmentează imaginea și obține ~5000 de regiuni candidat



Selective Search for Object Recognition

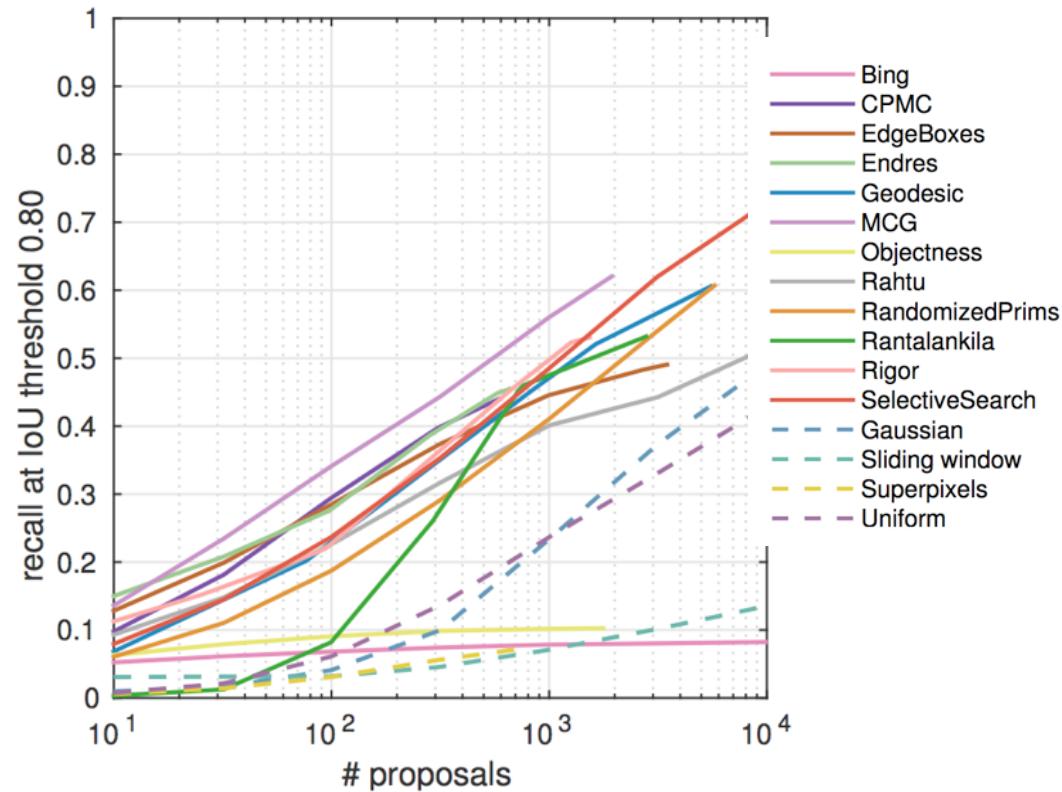
[J. R. R. Uijlings](#), [K. E. A. van de Sande](#), [T. Gevers](#), [A. W. M. Smeulders](#)
In International Journal of Computer Vision 2013.

Alternativă pentru glisarea ferestrei: propuneri candidat

- diversitate de algoritmi de segmentare (k-means aplicat pentru culoare, k-means aplicat pentru culoare + poziție, etc)
- multe alegeri ale hiperparametrilor (numărul de clusteri, etc.)
- abordare exhaustivă:
 - fiecare cluster este un posibil obiect candidat
 - mii de segmentări → mii de propuneri candidat

Alternativă pentru glisarea ferestrei: propuneri candidat

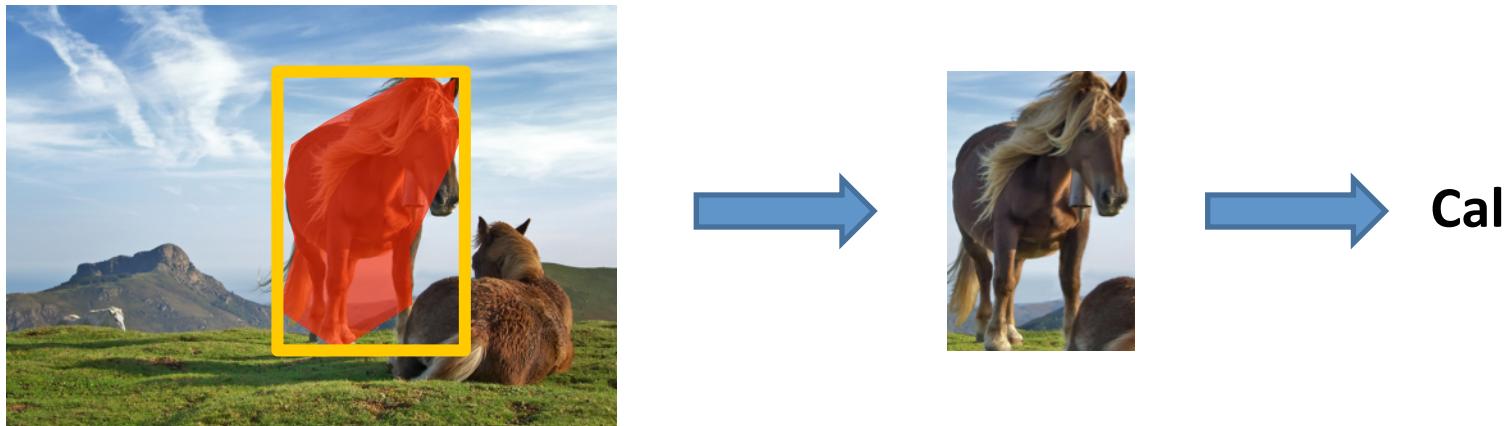
- zeci de metode de a genera propuneri de obiecte (“object proposals”)
- care este procentul din obiectele adnotate care sunt acoperite de ferestre?



What makes for effective detection proposals? J. Hosang, R. Benenson, P. Dollar, B. Schiele. In TPAMI 2015

Regiuni candidat

- fiecare regiune candidat este reprezentat de un grup de pixeli
- obține fereastra cea mai mică ce acoperă regiunea
- *se poate folosi apoi orice metodă de clasificare*



Evoluția performanței - propuneri candidat

	VOC 2007	VOC 2010
DPM v5 (Girshick et al. 2011)	33.7%	29.6%
UVA sel. search (Uijlings et al. 2013)		35.1%

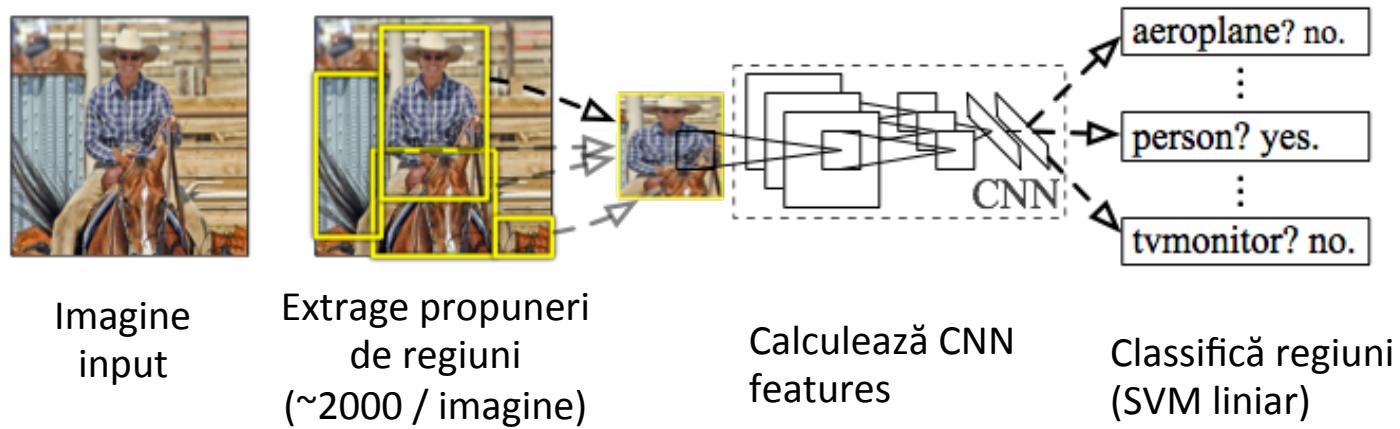
Sistem de referință

Evoluția performanței - propuneri candidat

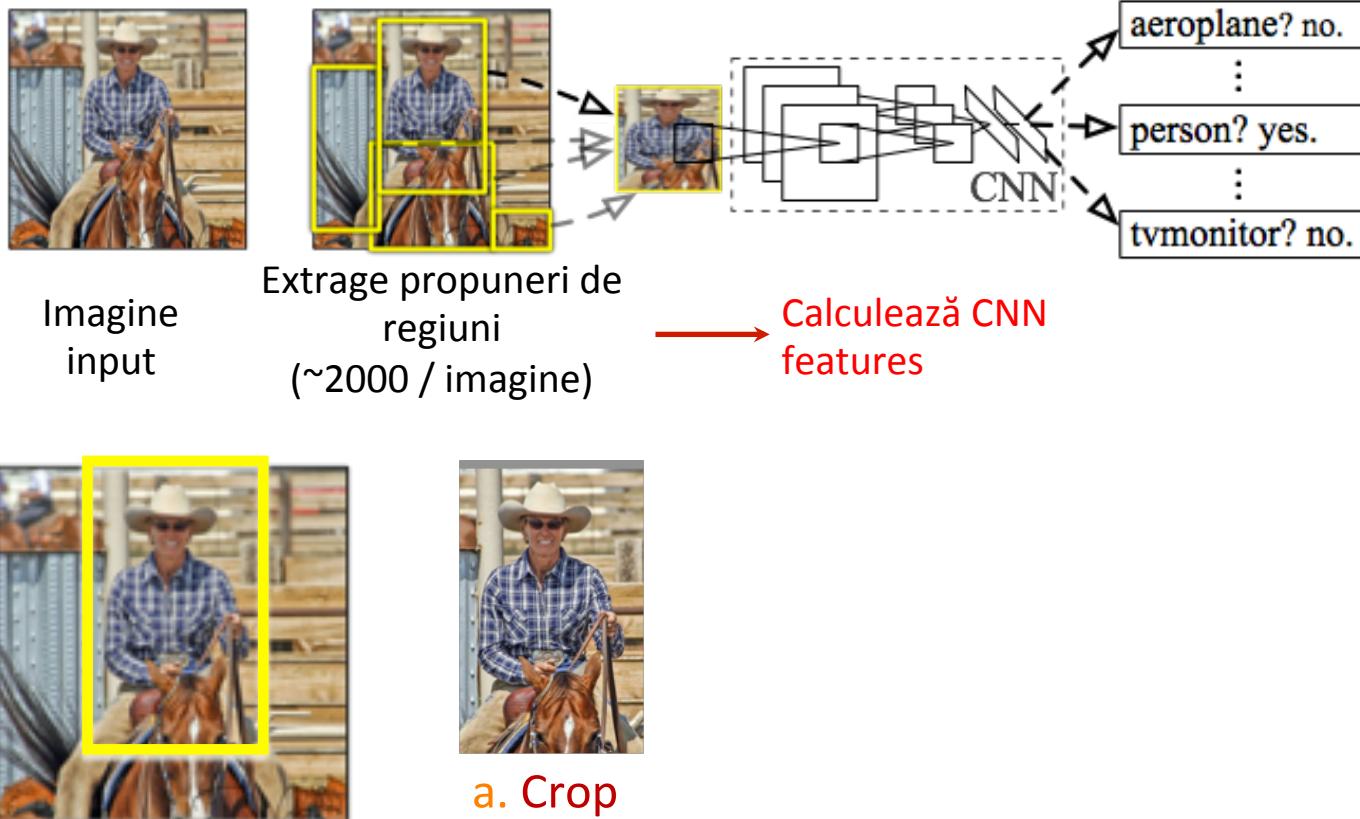
	VOC 2007	VOC 2010
DPM v5 (Girshick et al. 2011)	33.7%	29.6%
UVA sel. search (Uijlings et al. 2013)		35.1%
Regionlets (Wang et al. 2013)	41.7%	39.7%
SegDPM (Fidler et al. 2013)		40.4%

Sistem de referință

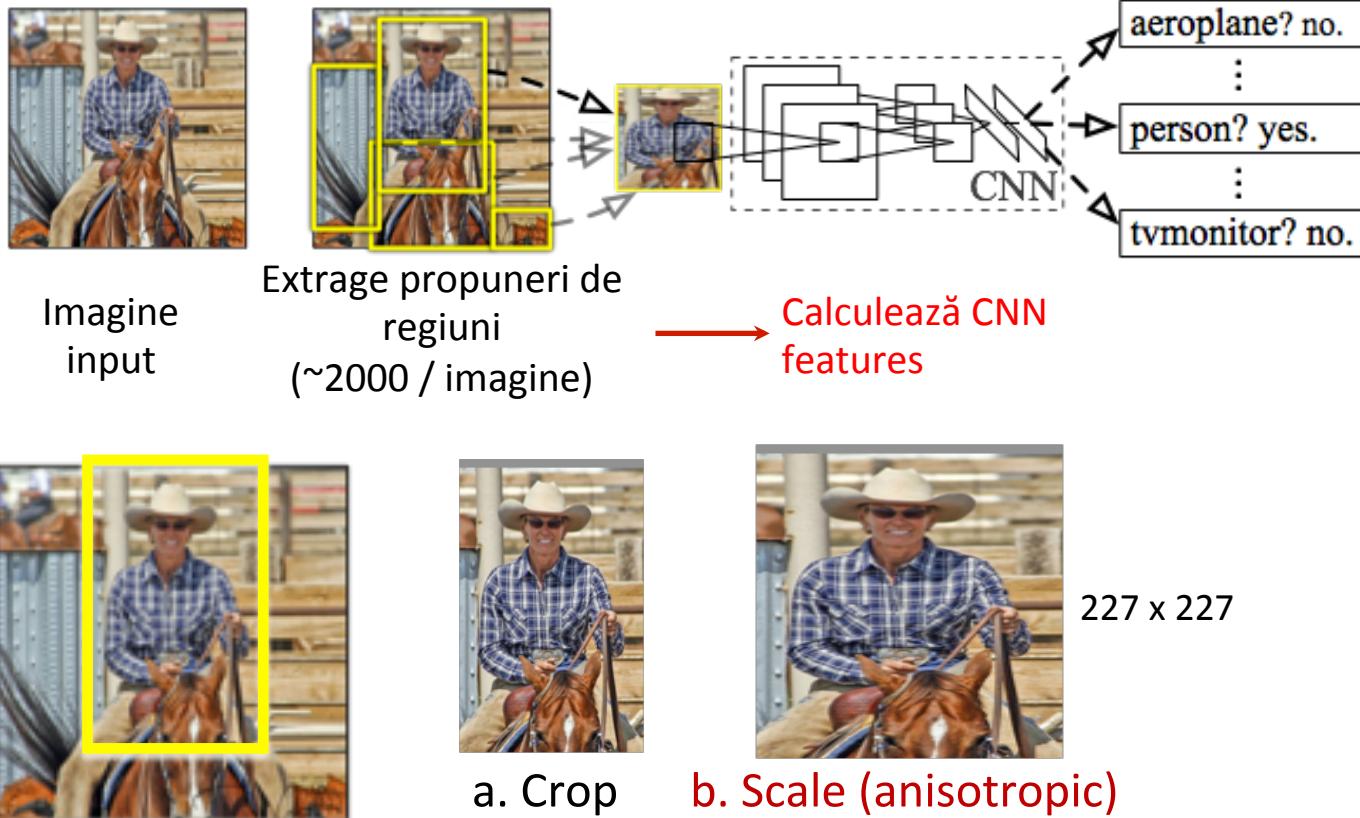
R-CNN: Regiuni + CNN features



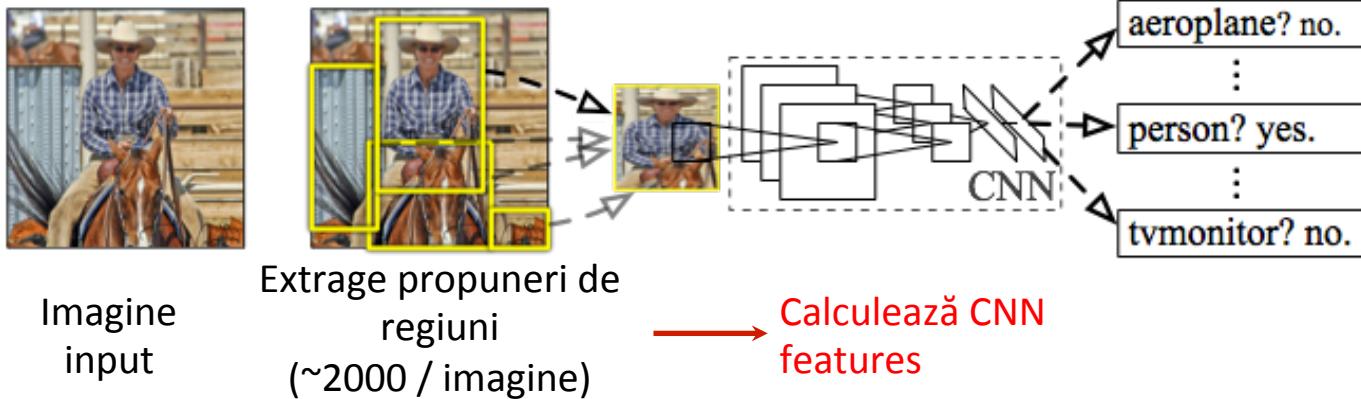
R-CNN: test time



R-CNN: test time



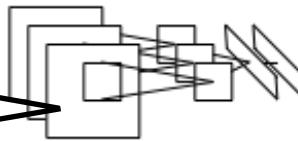
R-CNN: test time



a. Crop

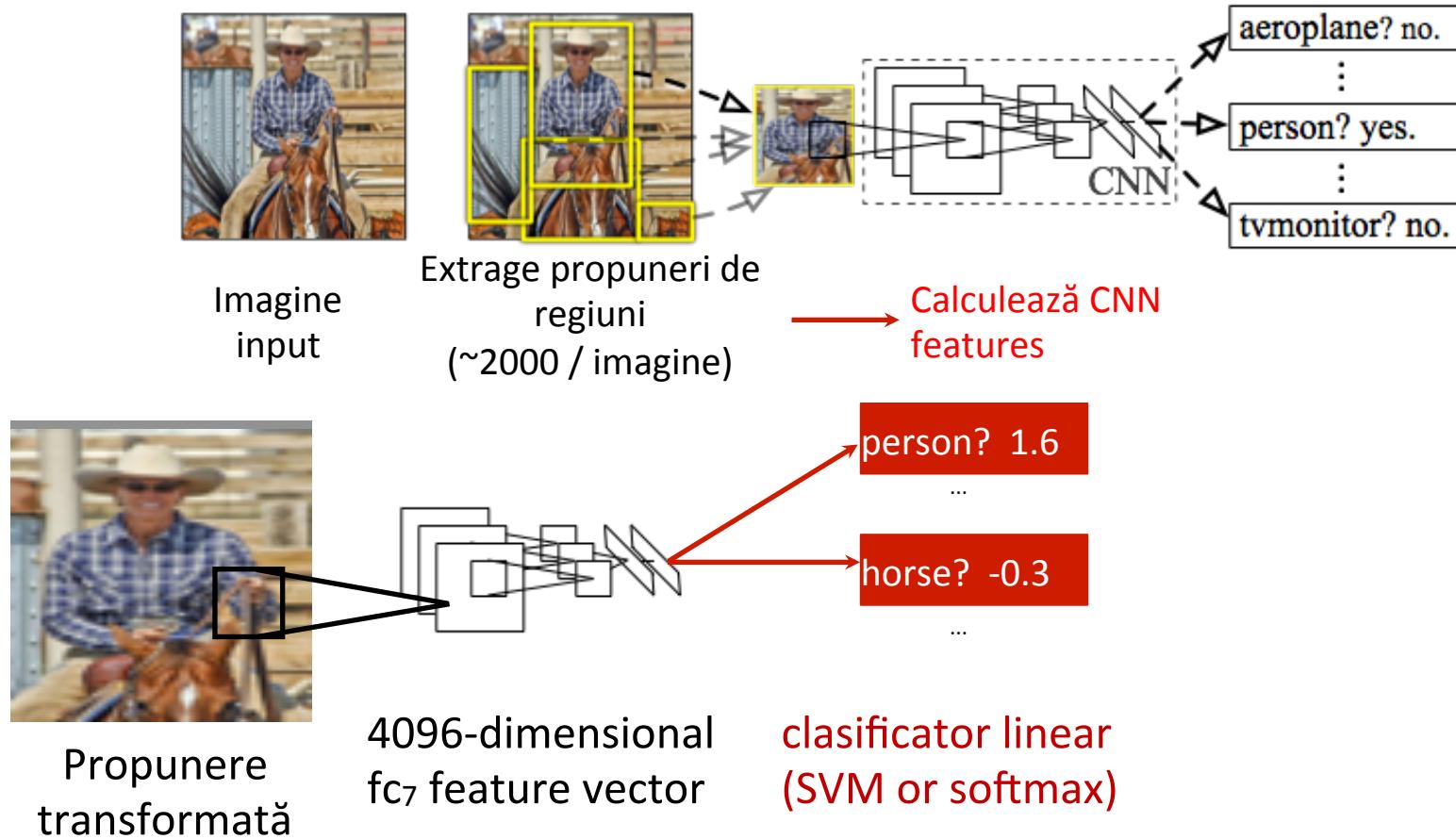


b. Scale (anisotropic)



c. Propagă înainte în rețea
Output: “fc₇” features

R-CNN: test time



R-CNN: test time



Propunere inițială

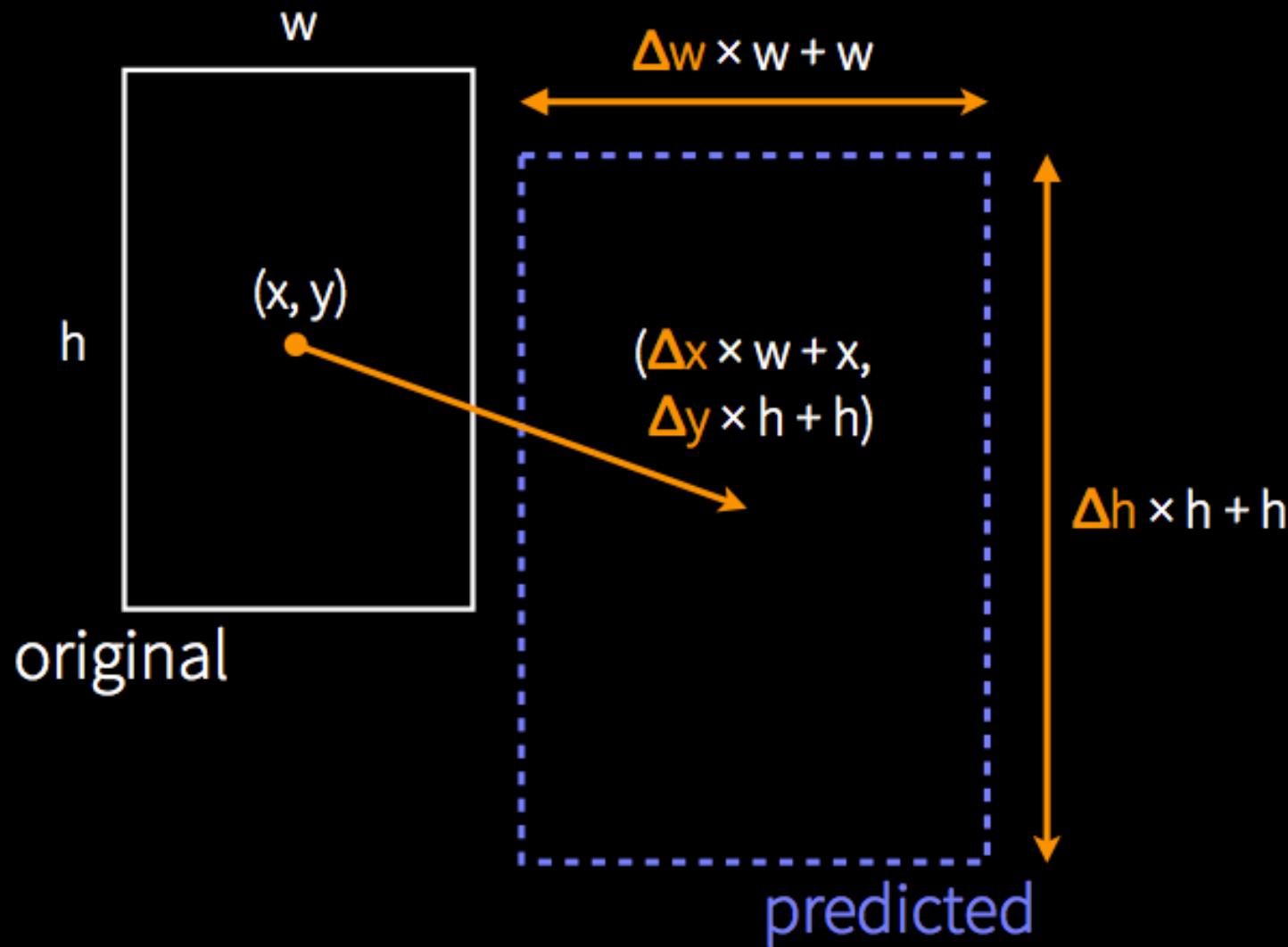
Linear regression
on CNN features



Predictie

Bounding-box regression

Bounding-box regression



Evoluția performanței - R-CNN

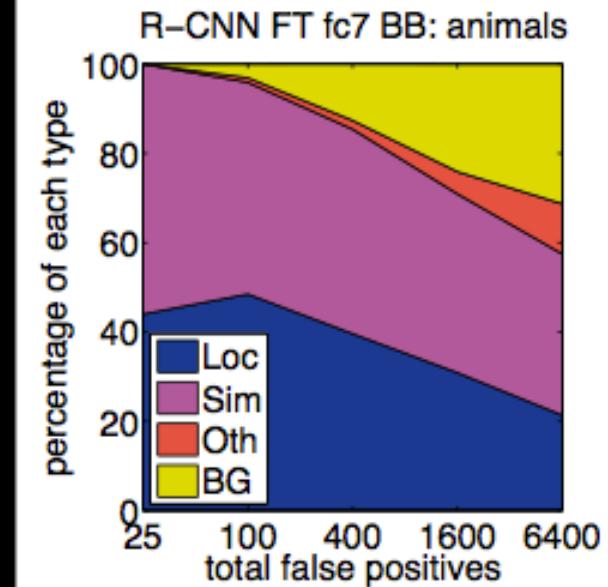
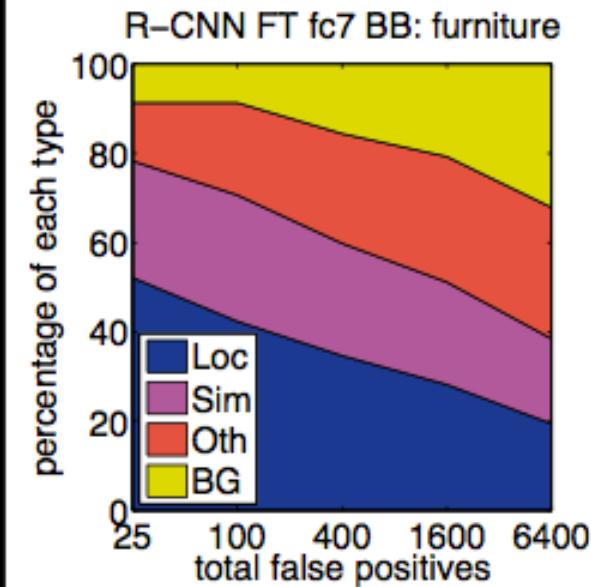
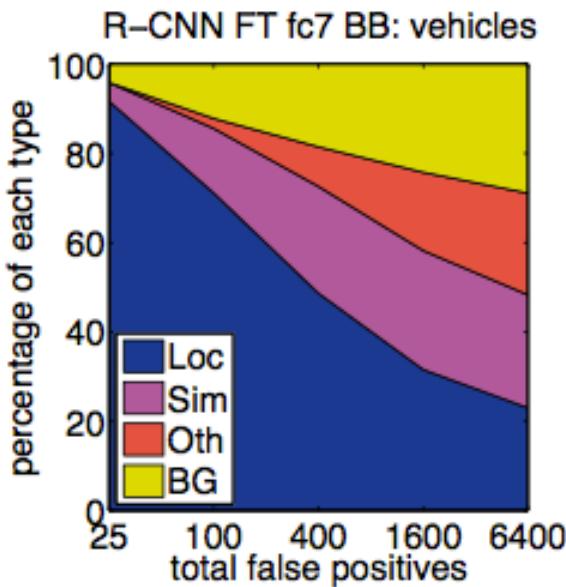
	VOC 2007	VOC 2010
DPM v5 (Girshick et al. 2011)	33.7%	29.6%
UVA sel. search (Uijlings et al. 2013)		35.1%
Regionlets (Wang et al. 2013)	41.7%	39.7%
SegDPM (Fidler et al. 2013)		40.4%

Sistem de referință

Evoluția performanței - R-CNN

	VOC 2007	VOC 2010
DPM v5 (Girshick et al. 2011)	33.7%	29.6%
UVA sel. search (Uijlings et al. 2013)		35.1%
Regionlets (Wang et al. 2013)	41.7%	39.7%
SegDPM (Fidler et al. 2013)		40.4%
R-CNN	54.2%	50.2%
R-CNN + bbox regression	58.5%	53.7%

False positive type distribution



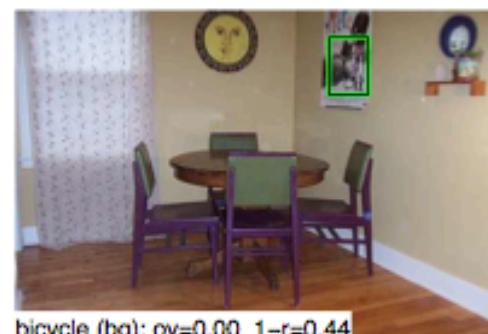
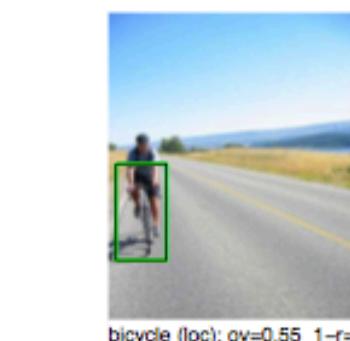
Loc = localization

Sim = similar classes

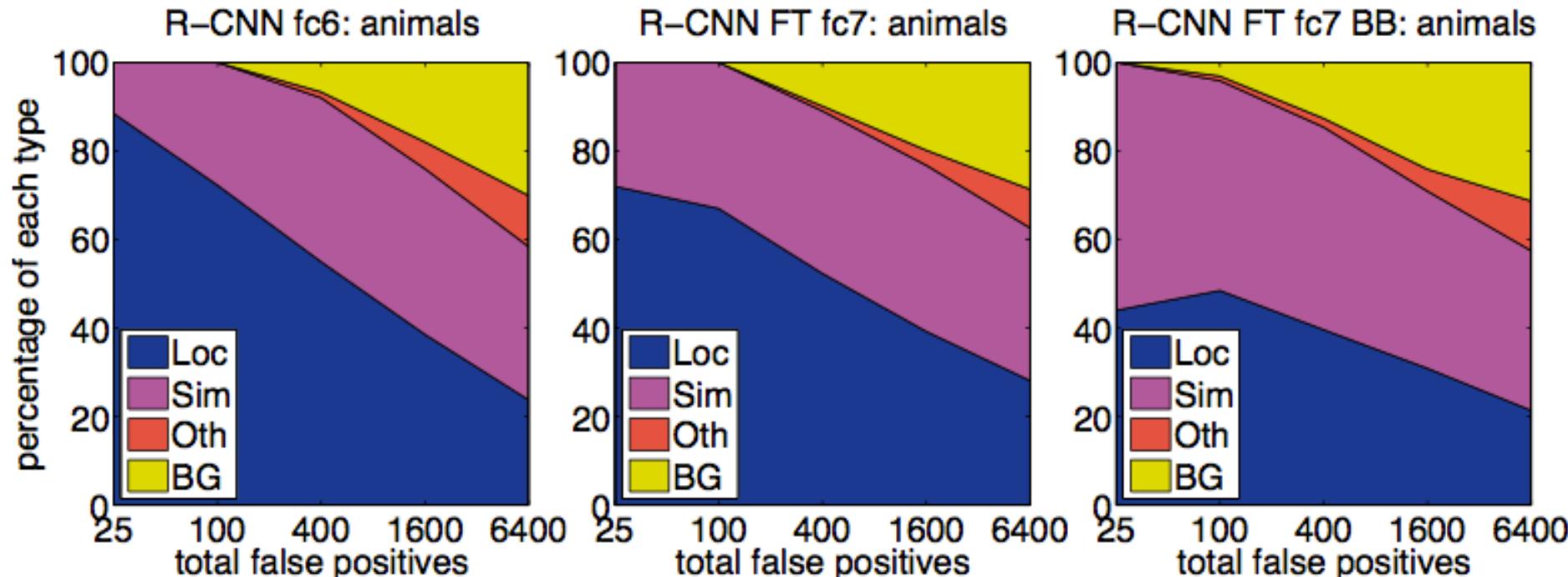
Oth = other / dissimilar classes

BG = background

Top bicycle FPs (AP = 72.8%)



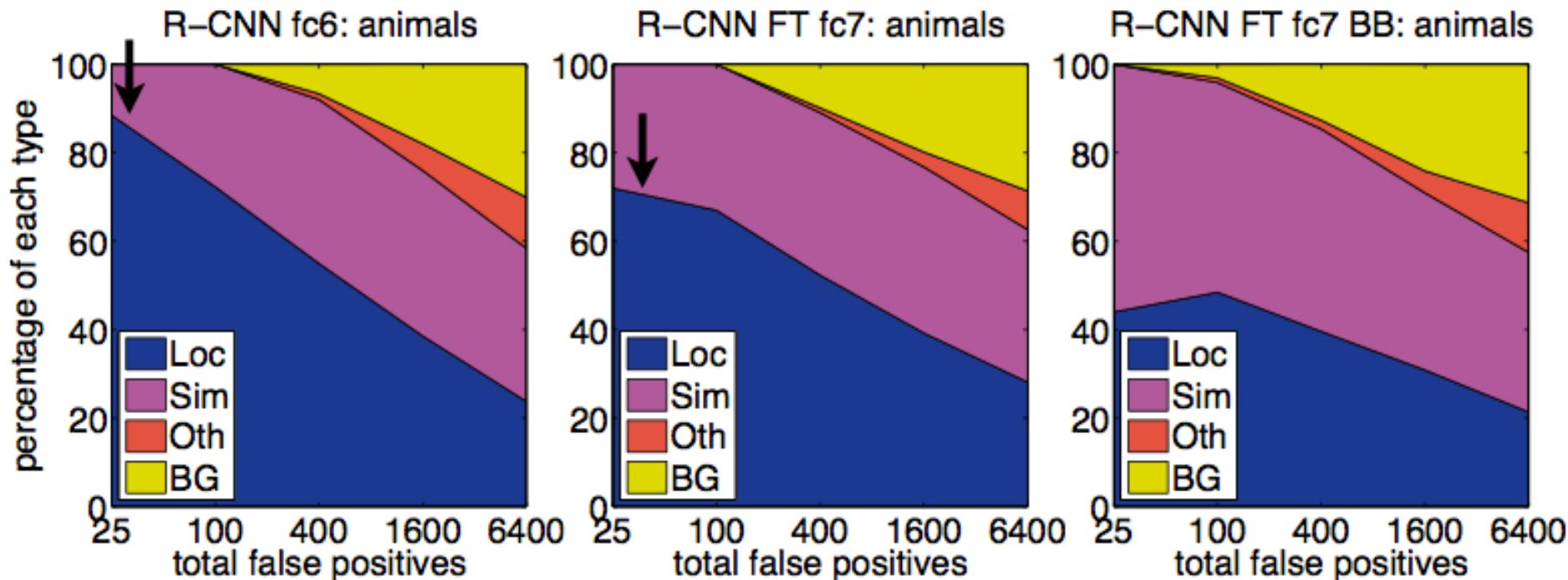
False positive analysis



No fine-tuning

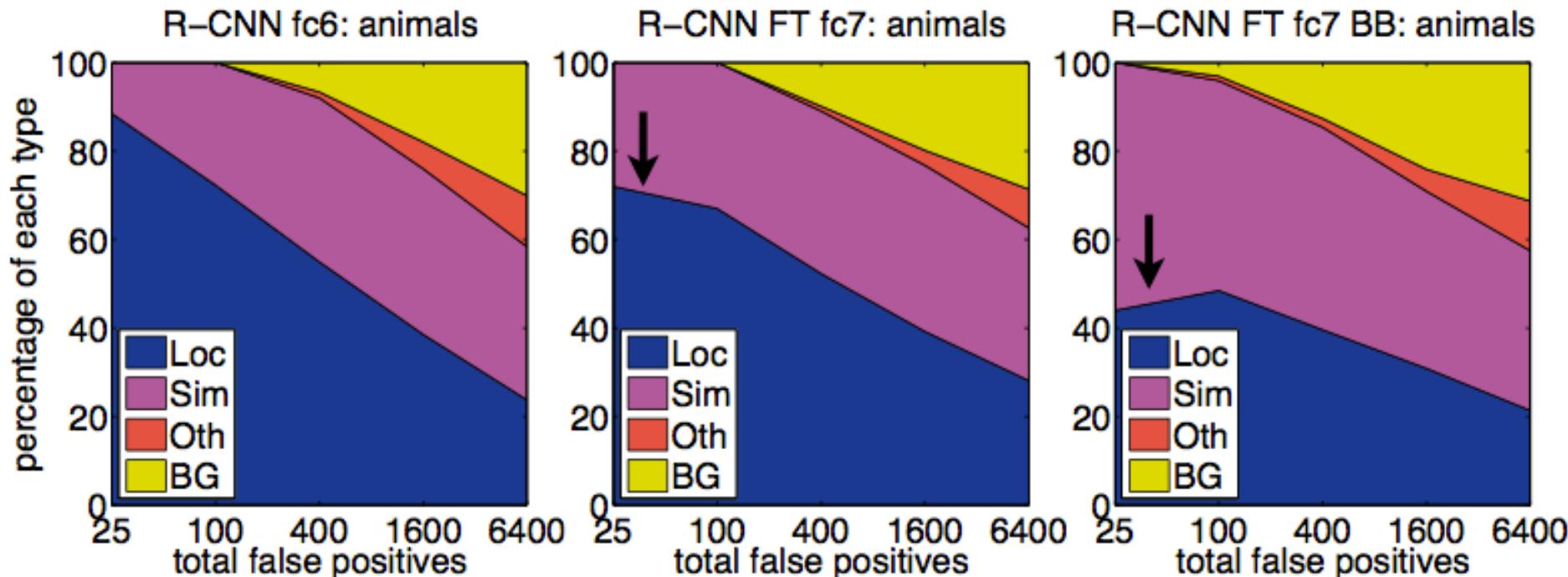
Analysis software: D. Hoiem, Y. Chodpathumwan, and
Q. Dai. "Diagnosing Error in Object Detectors." ECCV, 2012.

False positive analysis



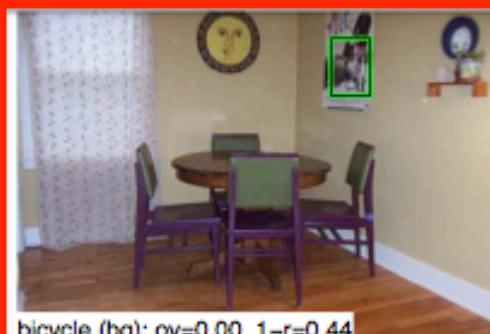
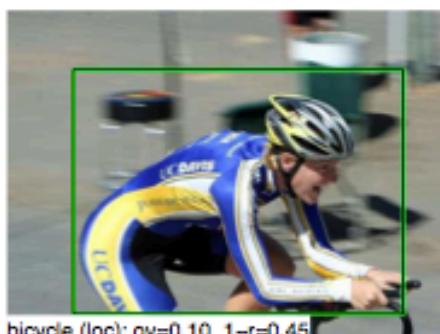
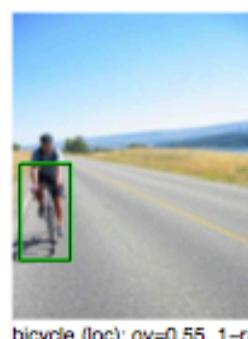
After fine-tuning

False positive analysis



After bounding-
box regression

Top bicycle FPs (AP = 72.8%)



False positive #15

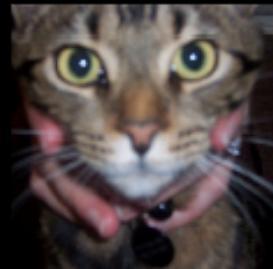


(zoom)

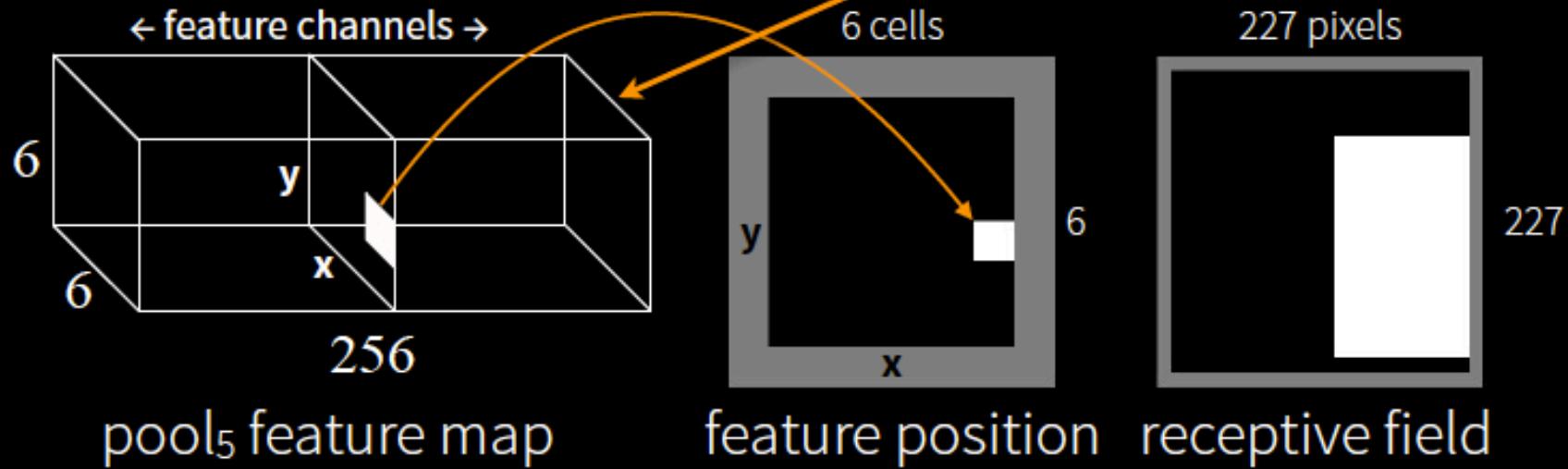
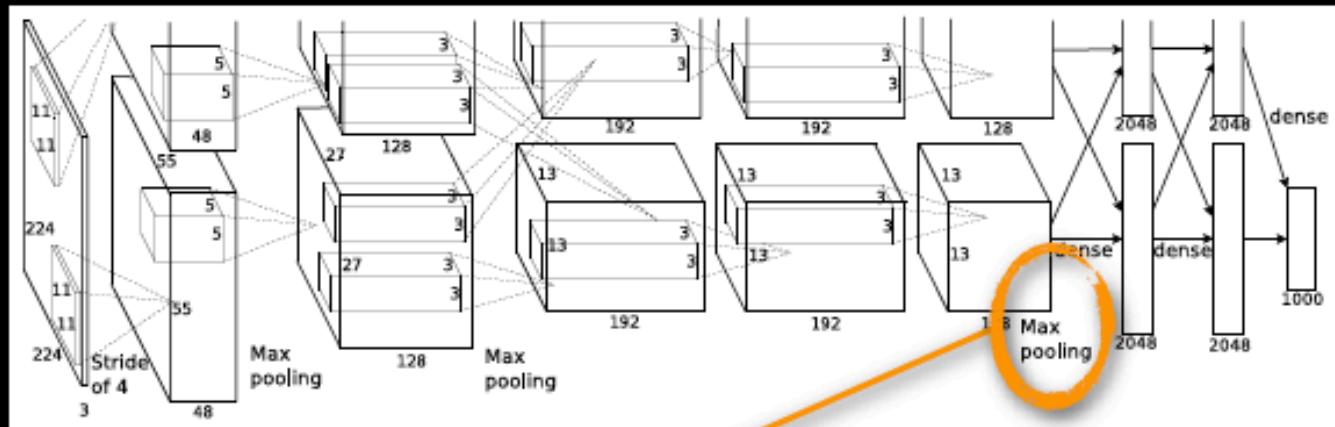


Unannotated bicycle

What did the network learn?



“stimulus”

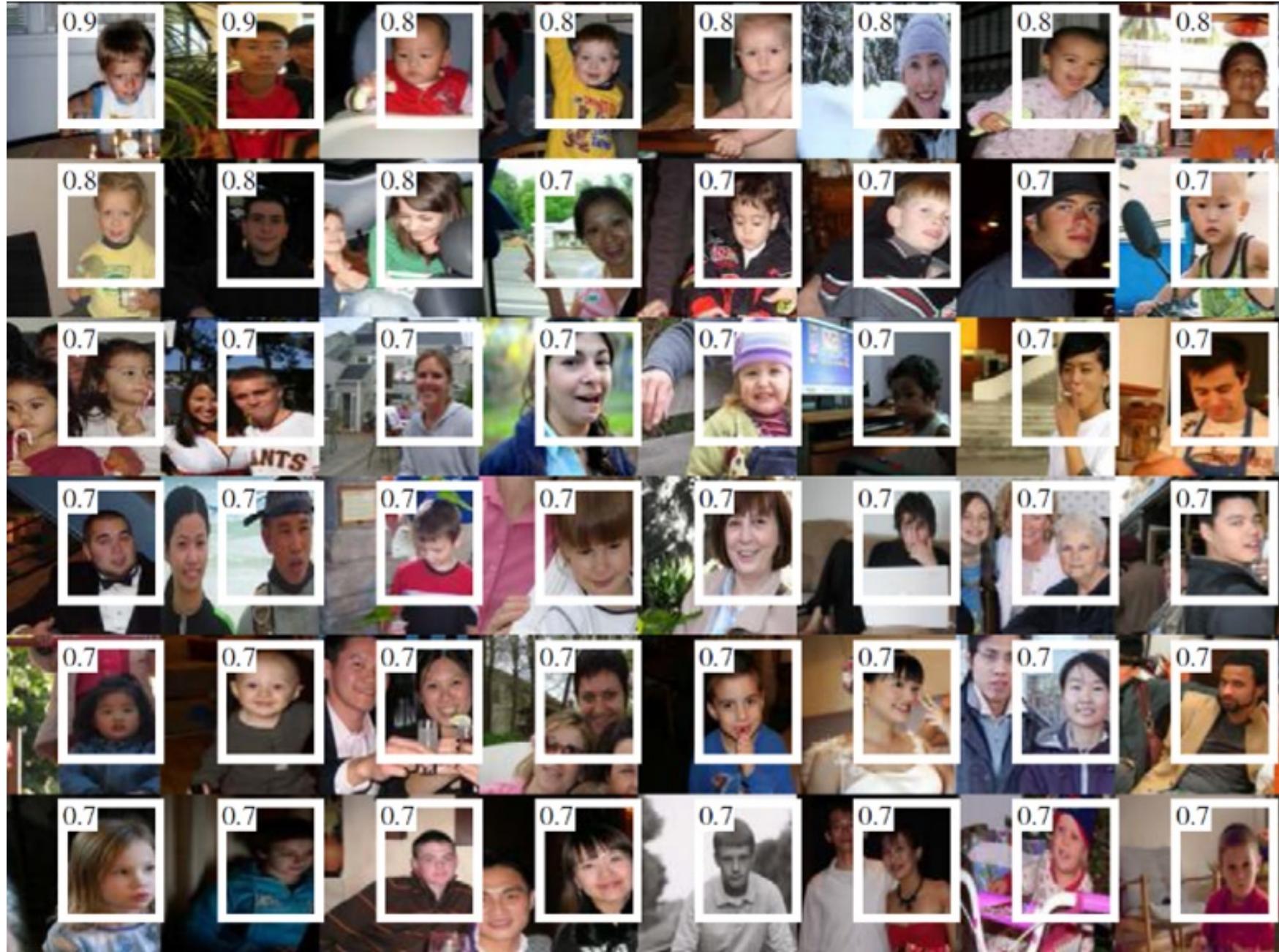


Visualize images that activate pool₅ a feature

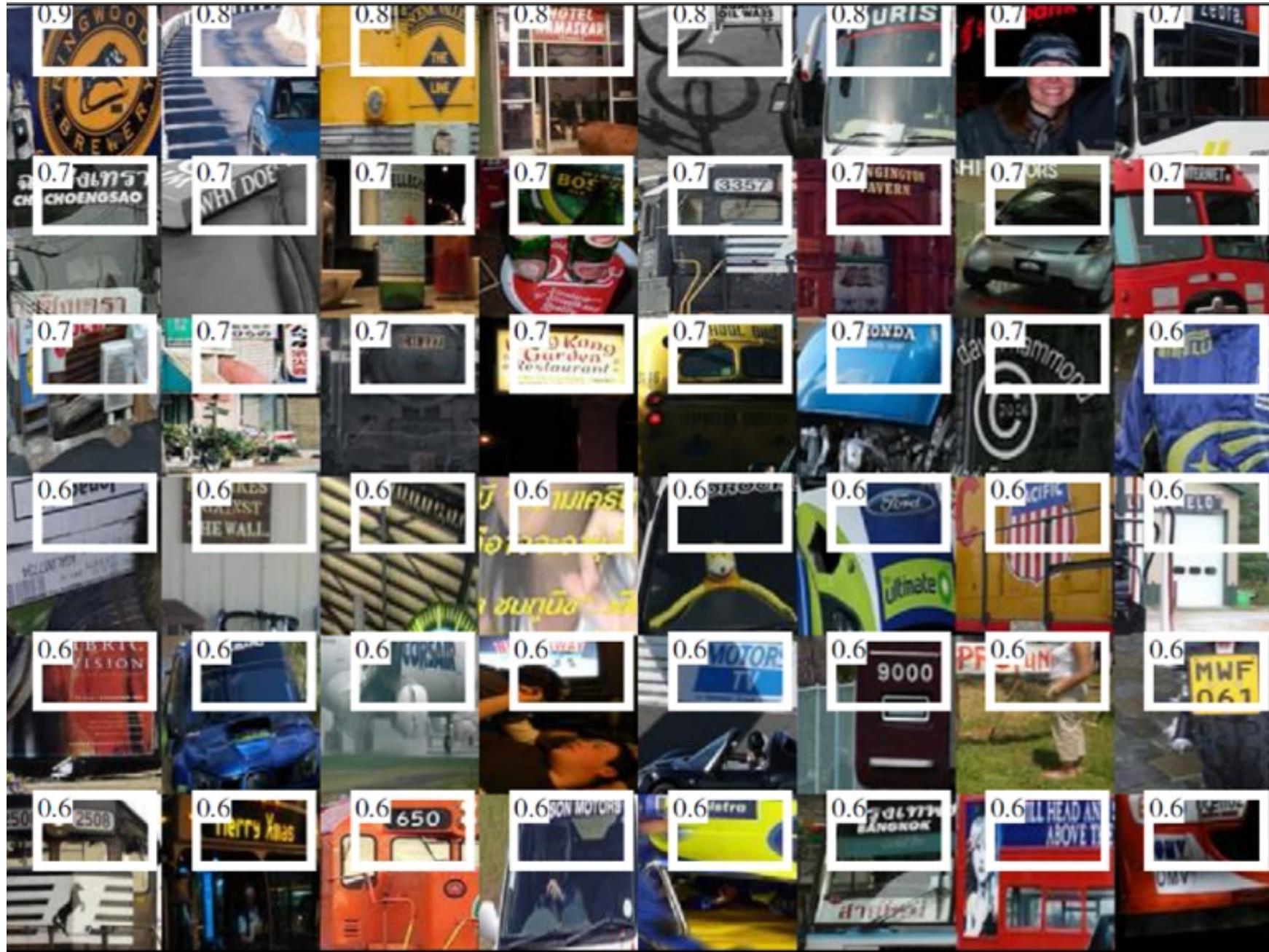
Imagini care activează un neuron din stratul pool₅



Imagini care activează un neuron din stratul pool₅



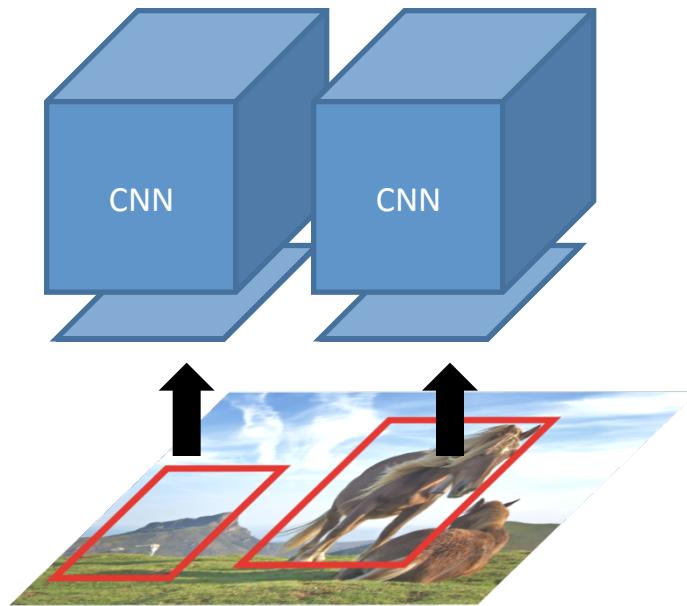
Imagini care activează un neuron din stratul pool₅



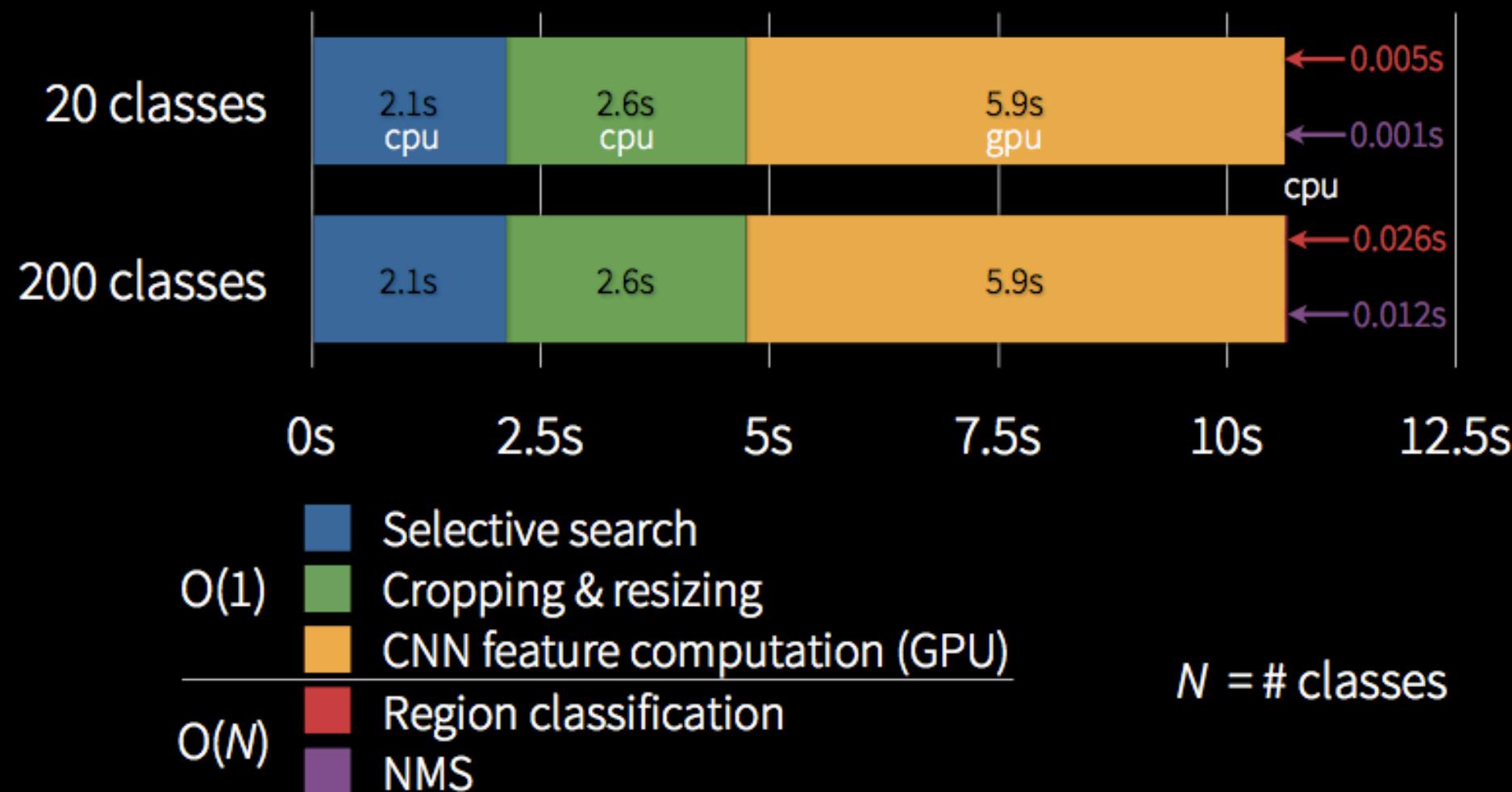
Antrenarea unui R-CNN

- antrenează o rețea neuronală conoluțională pe ImageNet pentru clasificare
- *fine-tuning* pentru detectare de obiecte:
 - problemă de clasificare!
 - propuneri candidat cu IoU > 50% sunt exemple pozitive

Accelerarea unui R-CNN



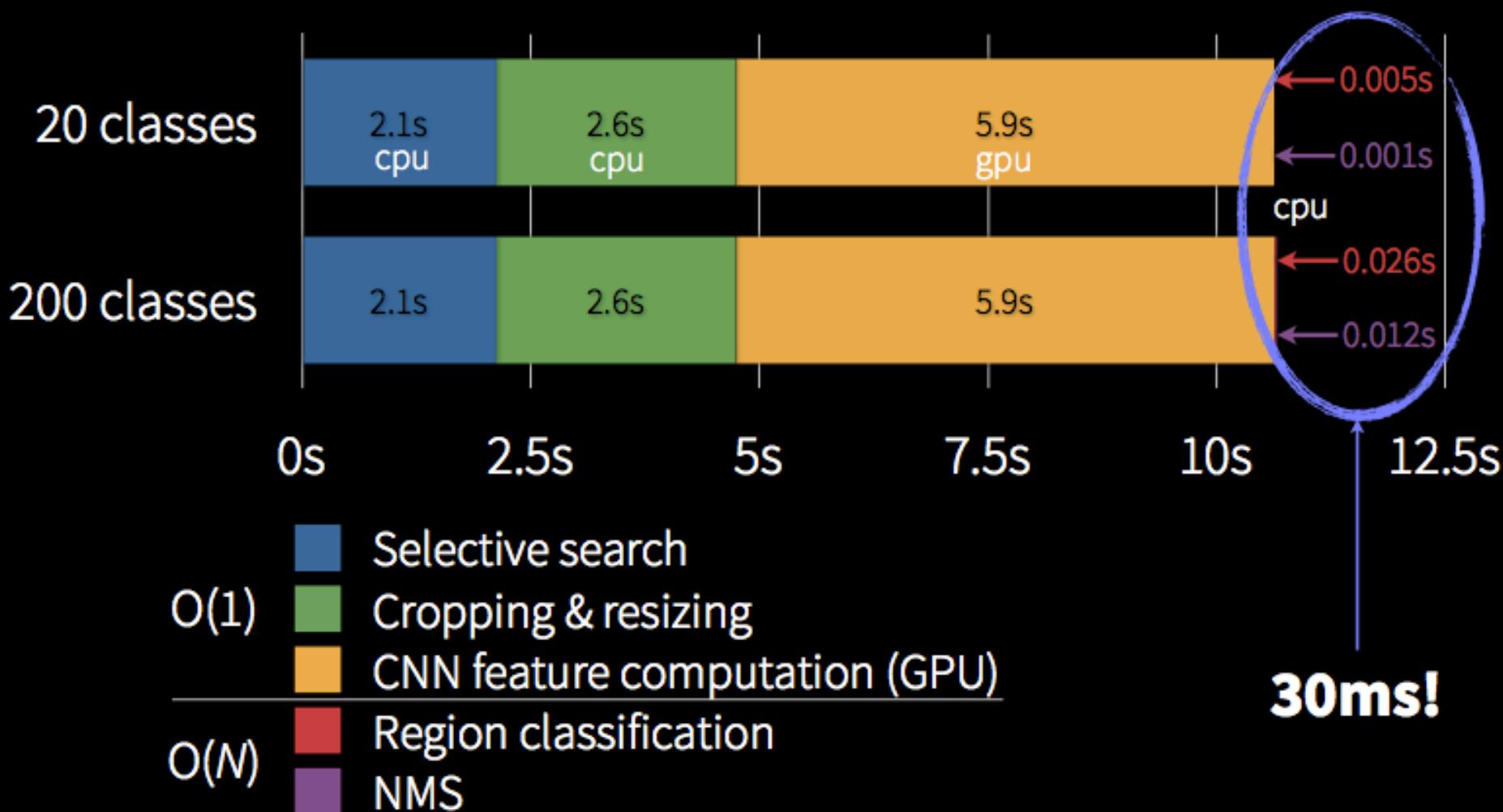
Detection speed & scalability



Hardware: Intel Core i7-3930K 3.2Ghz and NVIDIA Tesla K20c

We thank NVIDIA for generous hardware donations.

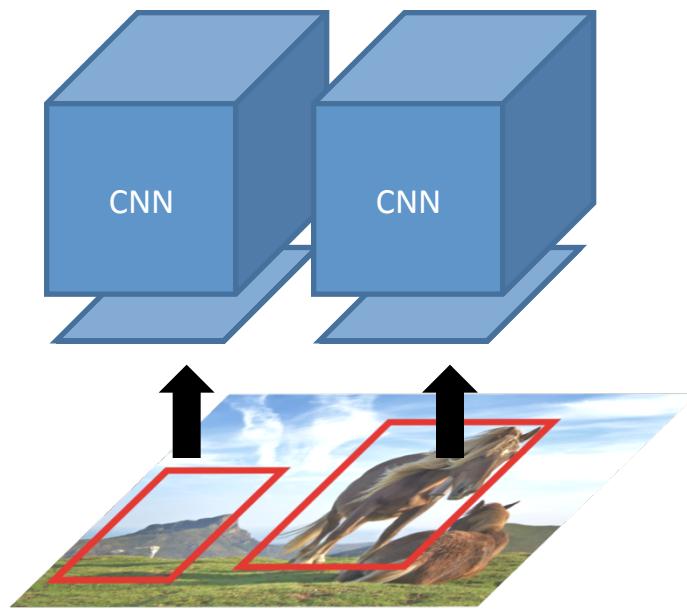
Detection speed & scalability



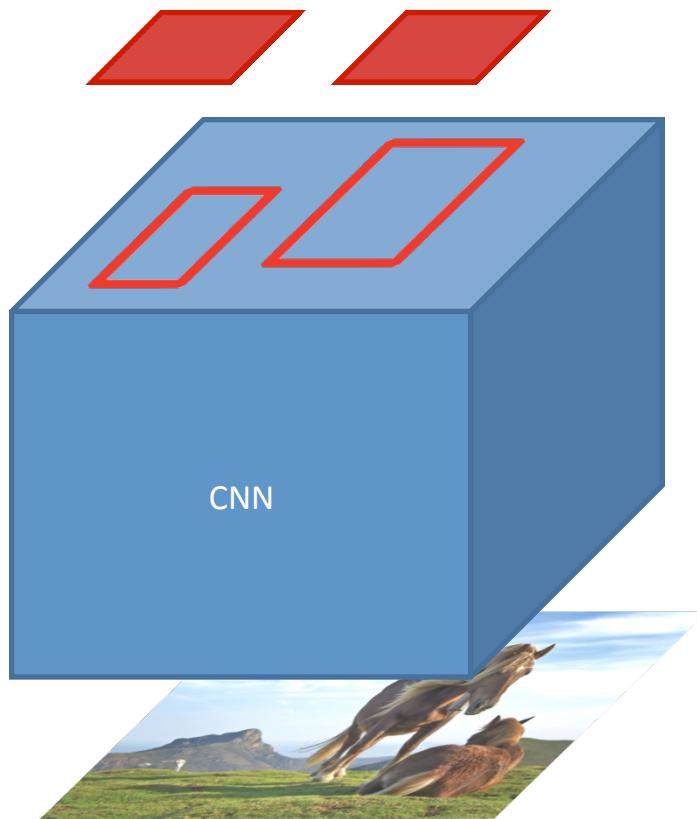
Hardware: Intel Core i7-3930K 3.2Ghz and NVIDIA Tesla K20c

We thank NVIDIA for generous hardware donations.

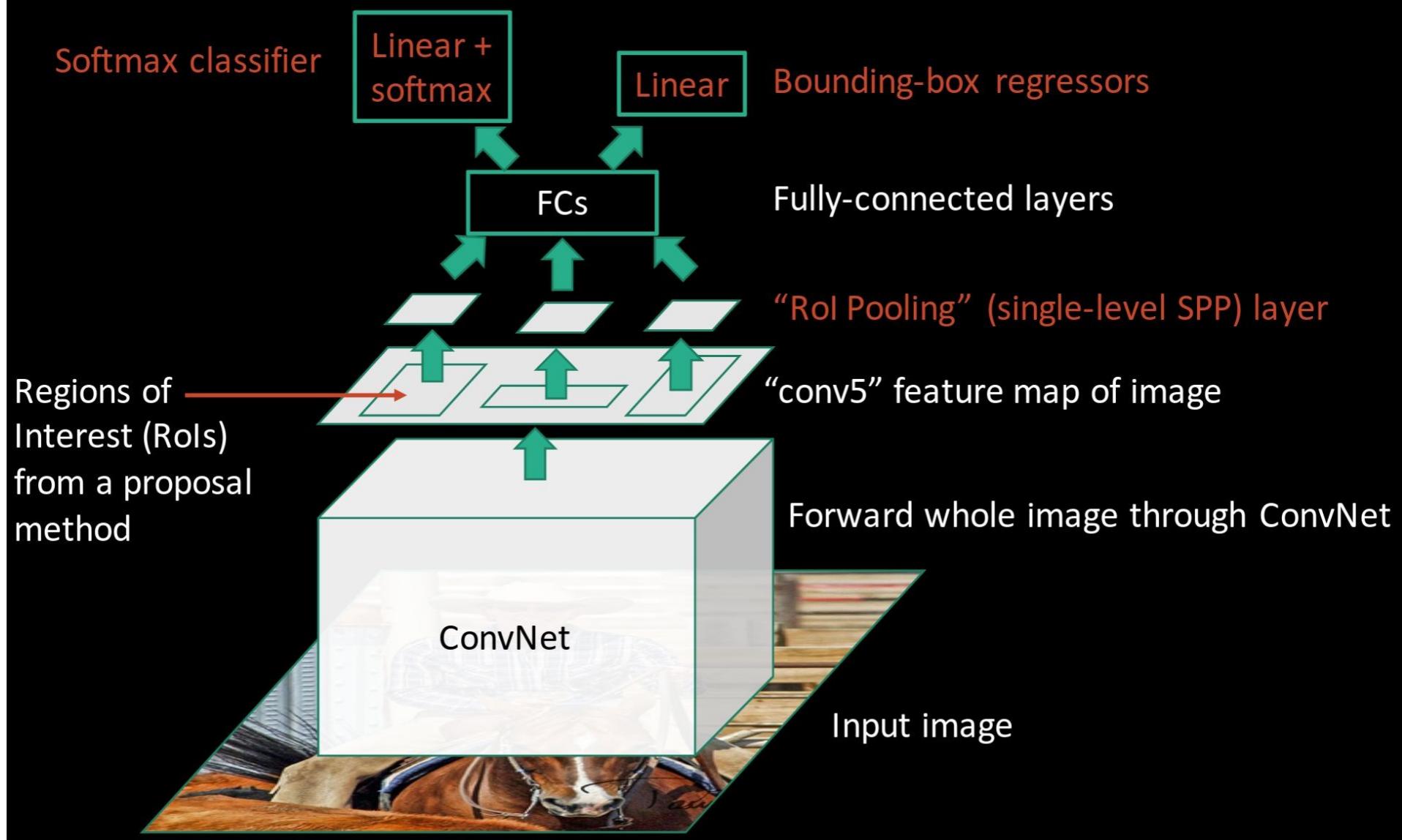
Accelerarea unui R-CNN



Accelerarea unui R-CNN

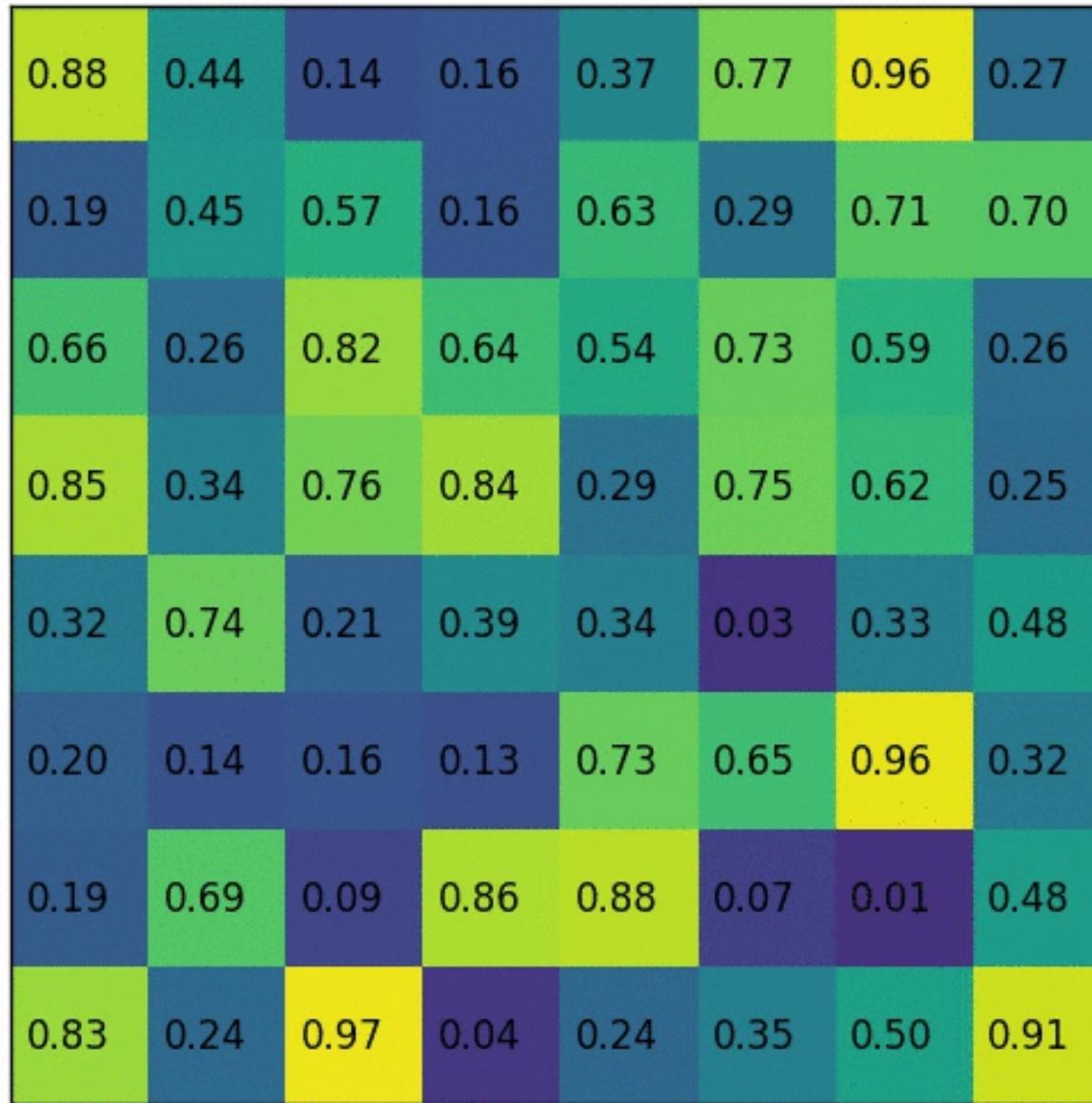


Fast R-CNN (test time)



ROI pooling layer

input



Fast R-CNN

	Fast R-CNN	R-CNN
Train time (h)	9.5	84
Speedup	8.8x	1x
Test time / image	0.32s	47.0s
Speedup	146x	1x
mean AP	66.9	66.0

Fast R-CNN

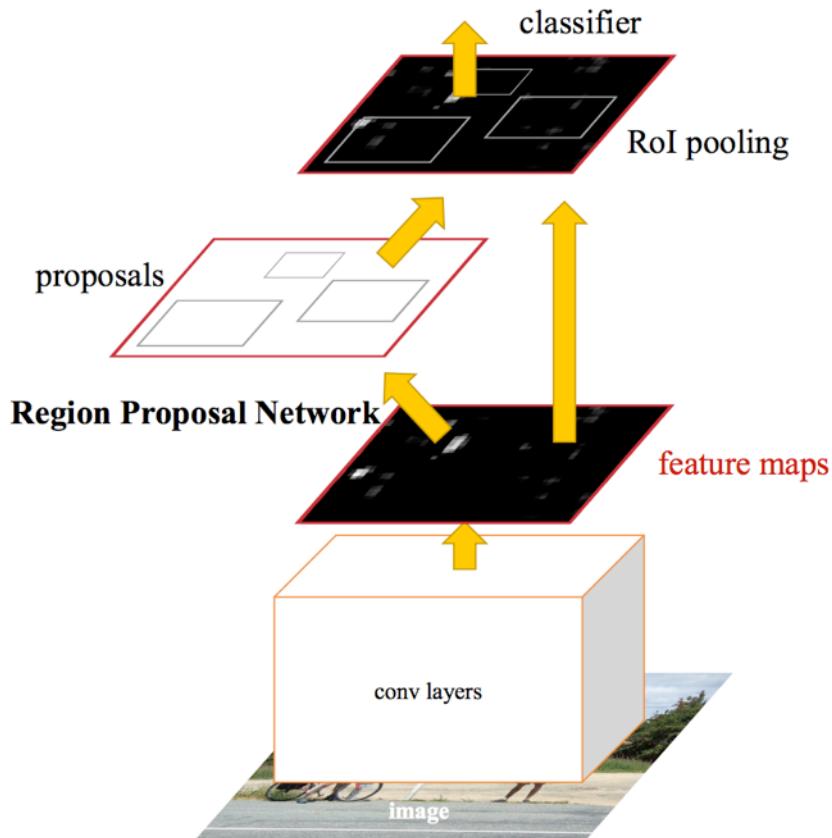
- partea cea mai consumatoare de timp:
 - generarea de propuneri de obiecte (în jur de câteva secunde)
- înceată
 - necesită segmentare
 - $O(1s)$ pe imagine

Faster R-CNN

- putem genera propuneri folosind rețeaua neuronală conlovuțională?
- pentru fiecare fereastră, asignează un objectness score

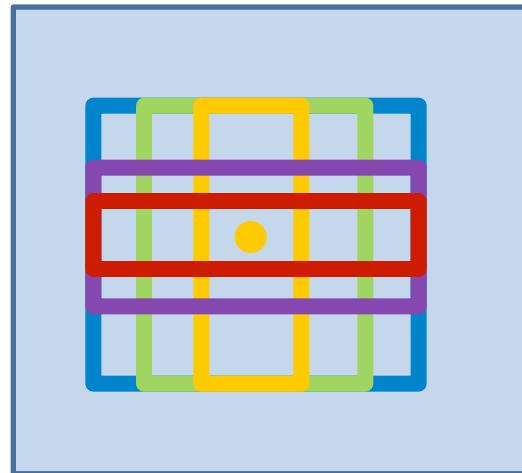
Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. S. Ren, K. He, R. Girshick, J. Sun. In *NIPS* 2015.

Faster R-CNN



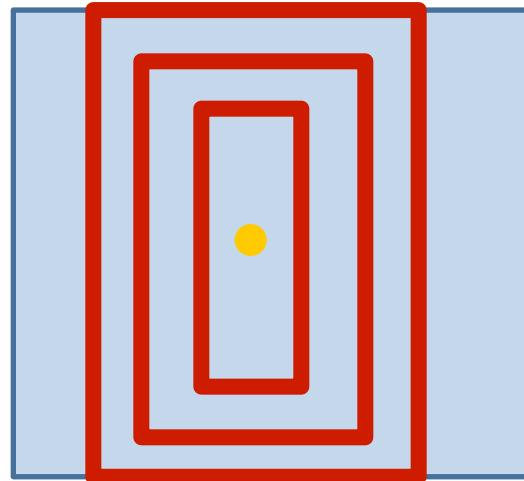
Faster R-CNN

- la fiecare poziție, consideră ferestre de mărimi și proporție (lățime/înălțime) diferite



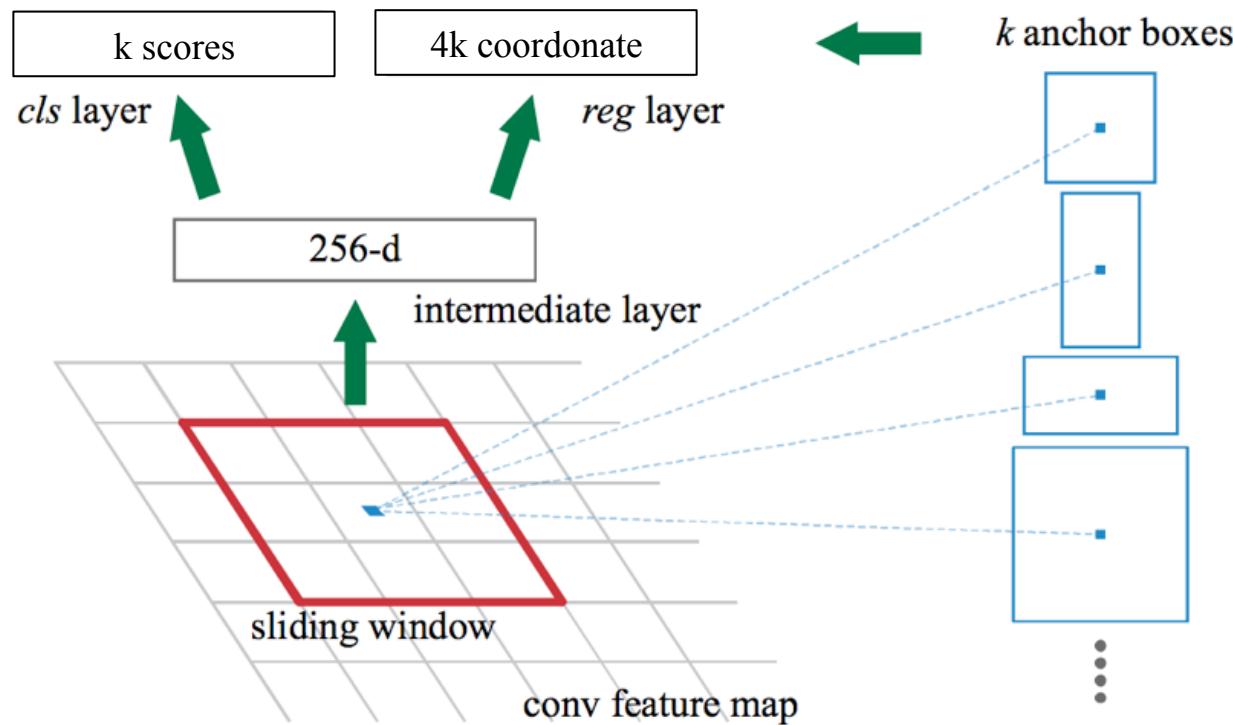
Faster R-CNN

- la fiecare poziție, consideră ferestre de mărimi și proporție (lățime/înălțime) diferite



Faster R-CNN

- la fiecare poziție, consideră ferestre de mărimi și proporție (lățime/înălțime) diferite



Faster R-CNN

- $s \text{ scale} * p \text{ proporții} = s * p \text{ ancore}$
- folosește un layer convolutional pentru a produce $s * p$ scoruri
- alege propunerile cu scorul cel mai mare

Faster R-CNN

Method	mean AP (PASCAL VOC)
Fast R-CNN	65.7
Faster R-CNN	67.0

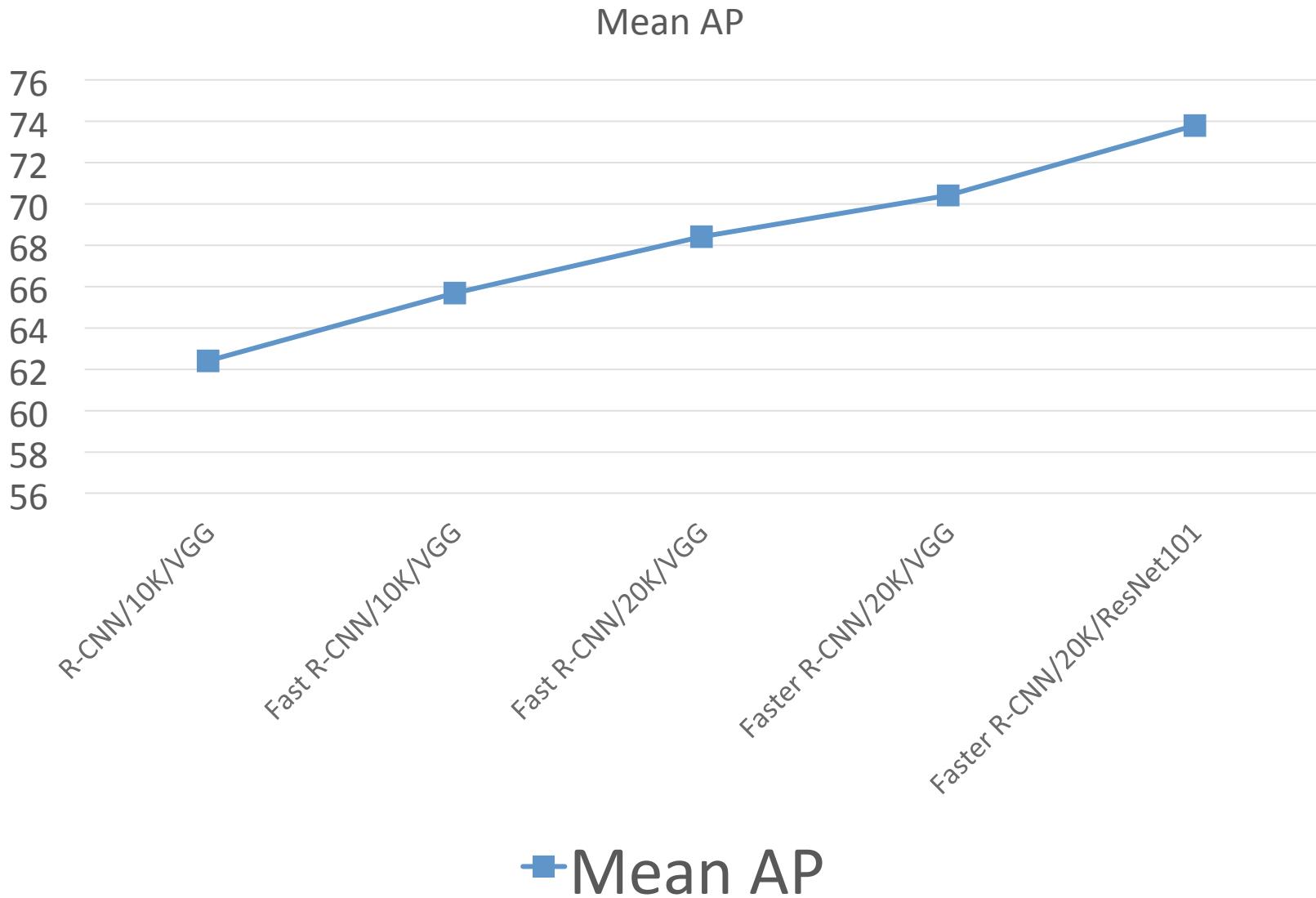
Impactul rețelei

ConvNet	mean AP (PASCAL VOC)
VGG	70.4
ResNet 101	73.8

Impactul de date adiționale

Method	Training data	mean AP (PASCAL VOC 2012 Test)
Fast R-CNN	VOC 12 Train (10K)	65.7
Fast R-CNN	VOC07 Trainval + VOC 12 Train	68.4
Faster R-CNN	VOC 12 Train (10K)	67.0
Faster R-CNN	VOC07 Trainval + VOC 12 Train	70.4

Performanță R-CNN

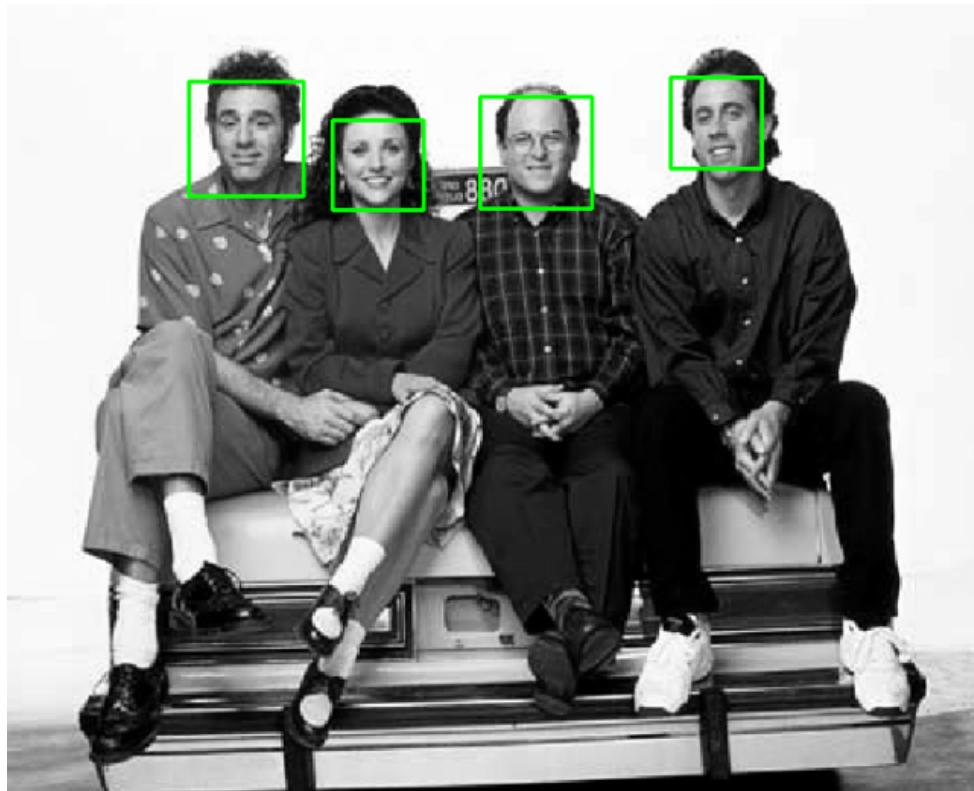


Performanță R-CNN

Table 6: Results on PASCAL VOC 2007 test set with Fast R-CNN detectors and VGG-16. For RPN, the train-time proposals for Fast R-CNN are 2000. RPN* denotes the unsharing feature version.

method	# box	data	mAP	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
SS	2000	07	66.9	74.5	78.3	69.2	53.2	36.6	77.3	78.2	82.0	40.7	72.7	67.9	79.6	79.2	73.0	69.0	30.1	65.4	70.2	75.8	65.8
SS	2000	07+12	70.0	77.0	78.1	69.3	59.4	38.3	81.6	78.6	86.7	42.8	78.8	68.9	84.7	82.0	76.6	69.9	31.8	70.1	74.8	80.4	70.4
RPN*	300	07	68.5	74.1	77.2	67.7	53.9	51.0	75.1	79.2	78.9	50.7	78.0	61.1	79.1	81.9	72.2	75.9	37.2	71.4	62.5	77.4	66.4
RPN	300	07	69.9	70.0	80.6	70.1	57.3	49.9	78.2	80.4	82.0	52.2	75.3	67.2	80.3	79.8	75.0	76.3	39.1	68.3	67.3	81.1	67.6
RPN	300	07+12	73.2	76.5	79.0	70.9	65.5	52.1	83.1	84.7	86.4	52.0	81.9	65.7	84.8	84.6	77.5	76.7	38.8	73.6	73.9	83.0	72.6
RPN	300	COCO+07+12	78.8	84.3	82.0	77.7	68.9	65.7	88.1	88.4	88.9	63.6	86.3	70.8	85.9	87.6	80.1	82.3	53.6	80.4	75.8	86.6	78.9

Detectare facială



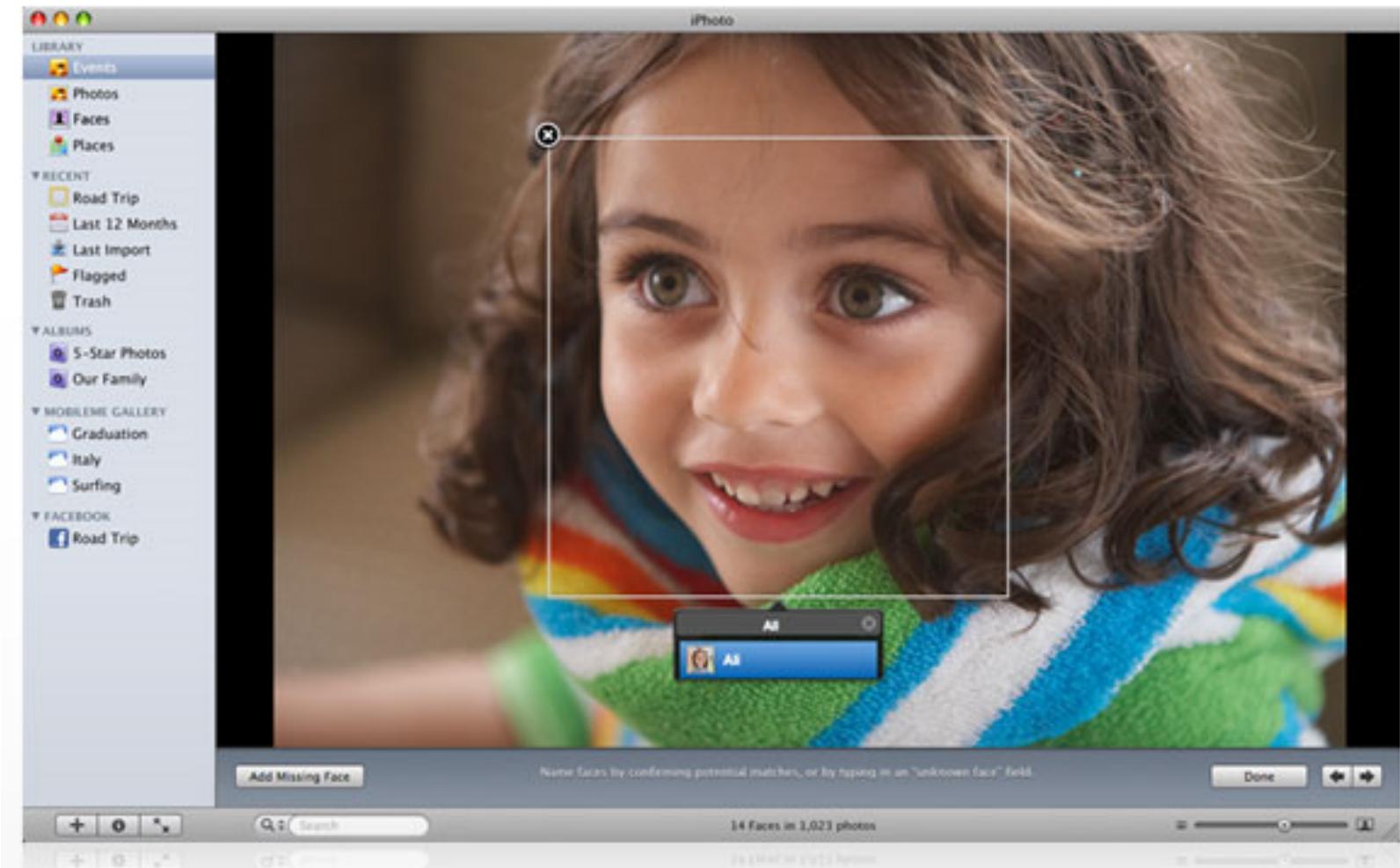
Demo – cel mai bun sistem de detectare facială



Detectare facială și identificare



Aplicație: Apple iPhoto

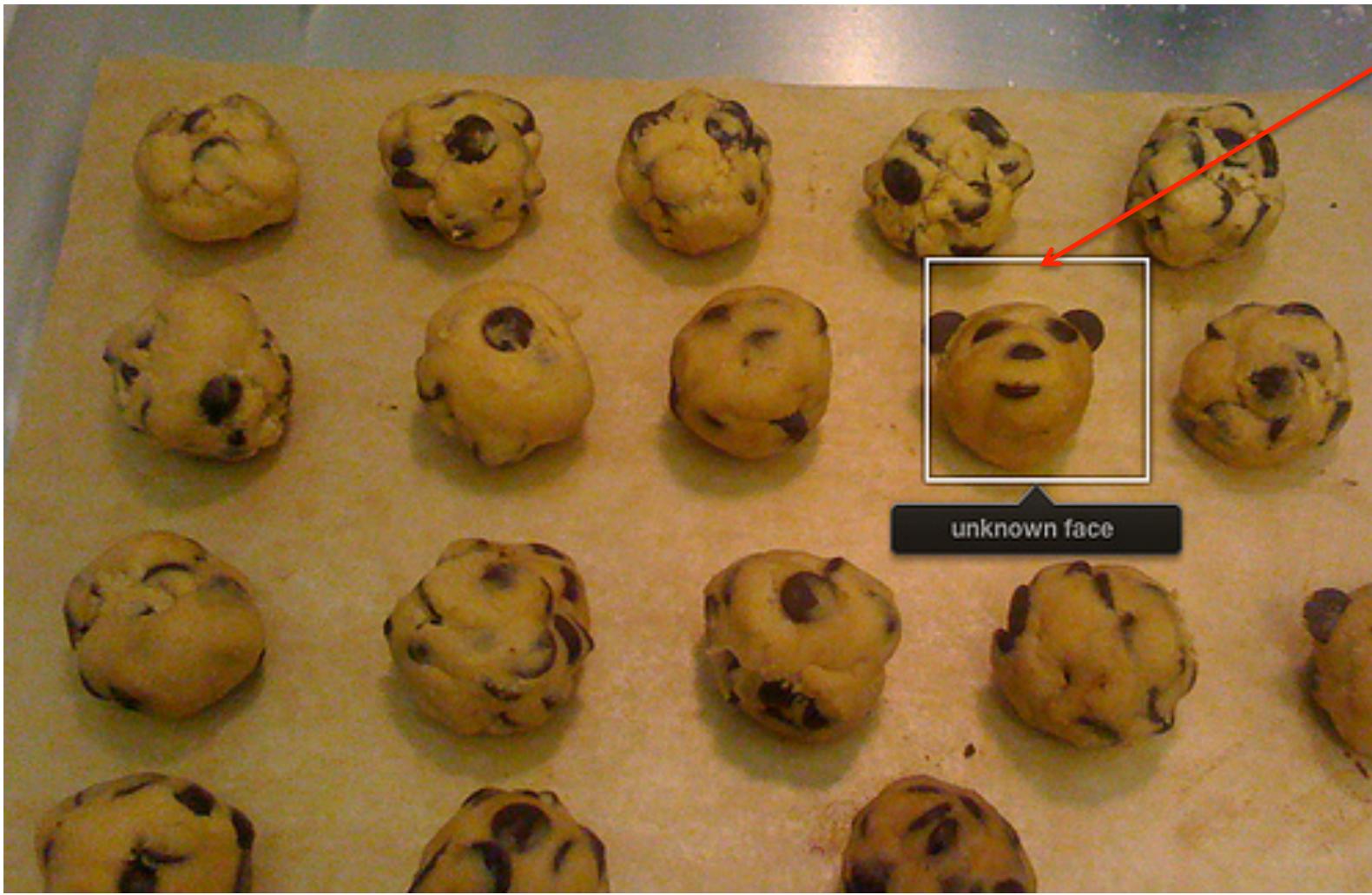


<http://www.apple.com/ilife/iphoto/>

Aplicație: Apple iPhoto

Exemplu
Fals Pozitiv

(detectorul
găsește o față
care nu există)



De ce o problemă grea detectarea facială?

- **postura:** frontal, profil
- **prezență sau absența unor componente structurale:** caracteristici ca barbă, mustață, ochelari pot fi prezente sau nu în imagine + variabilitatea lor în formă culoare, mărime
- **expresii faciale**
- **mascare:** fețele pot fi mascate parțial de alte obiecte. Într-o imagine cu un grup de oameni, unele fețe pot masca alte fețe
- **condițiile în care este făcuta fotografia:** lumina + caracteristicile camerei afectează înfățisarea unei fețe

Abordări pentru detectarea facială

Există multe abordări de succes. Cele mai cunoscute:

1. detector bazat pe metoda glisării ferestrei și HOG (cursul 12)
2. detectorul Viola-Jones (implementat in OpenCV/ Matlab) – azi
3. detector bazat pe metoda vectorilor proprii (eigen-faces)
4. detector bazat pe rețele neuronale de tip deep – multe implementări - azi

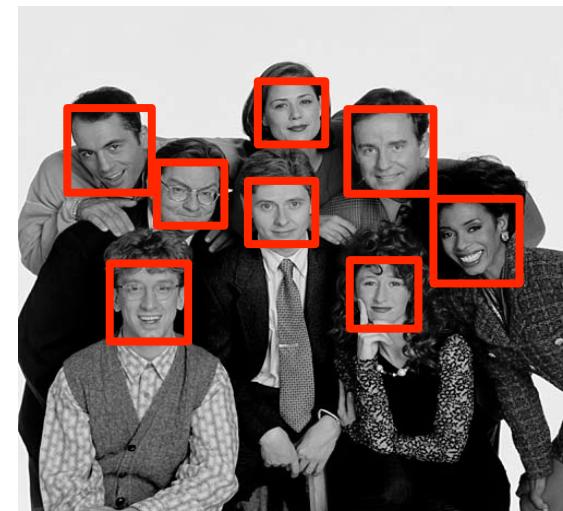
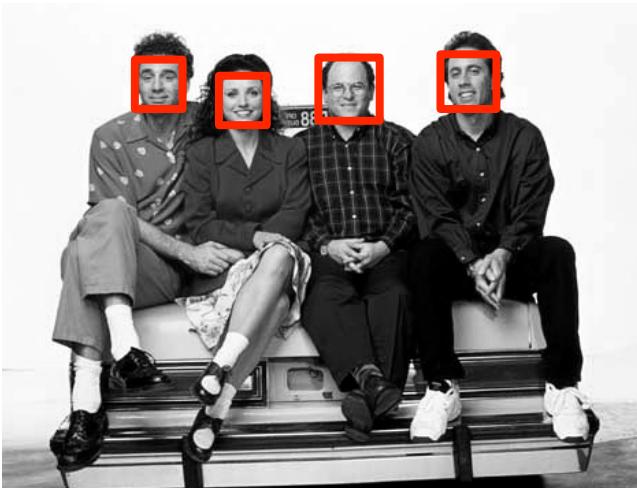
Detectare facială folosind metoda glisării ferestrei și histograme de gradienti orientați

Unde se află în imagine fețele umane?



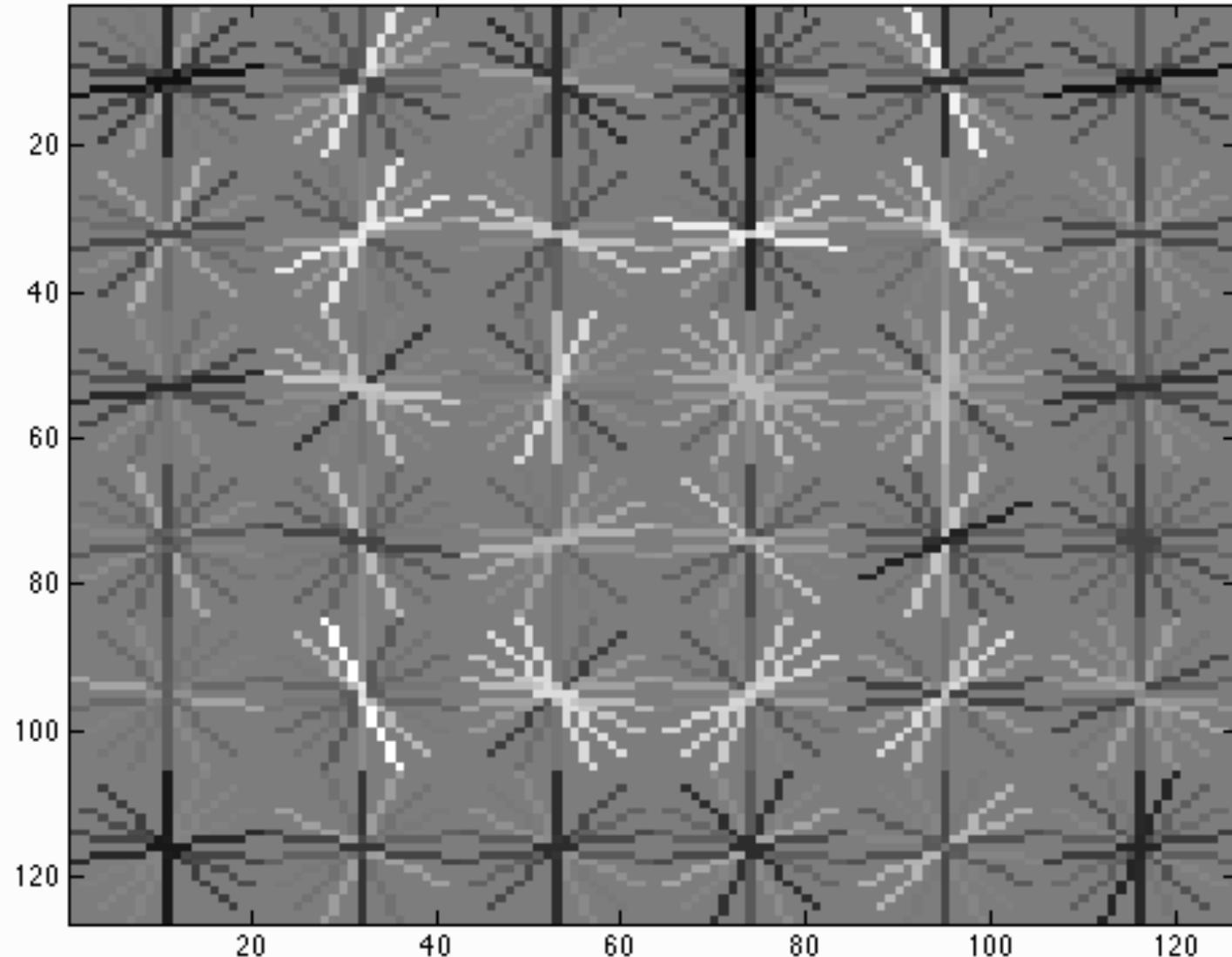
Detectare facială folosind metoda glisării ferestrei și histograme de gradienți orientați

Unde se află în imagine fețele umane?

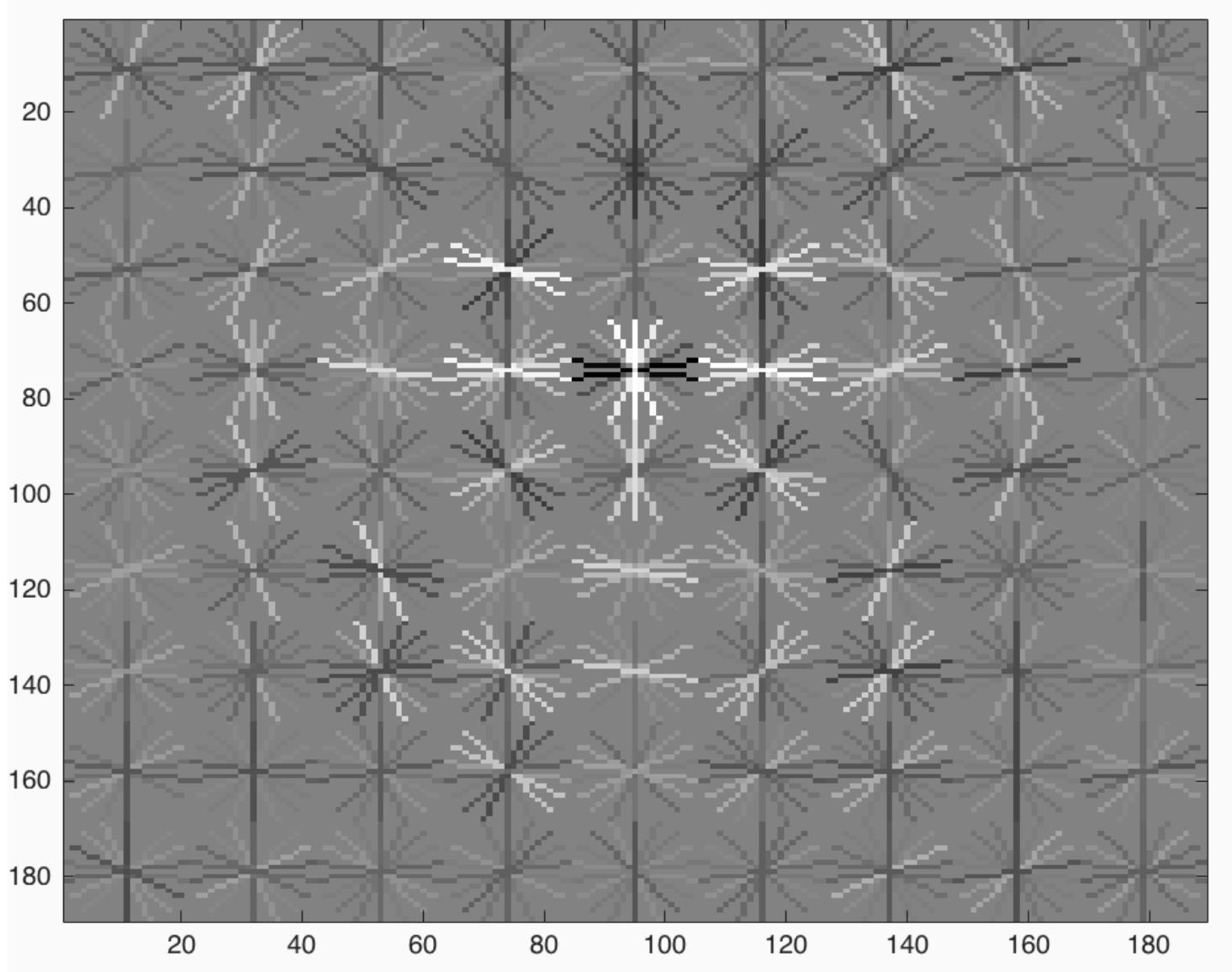


Localizare la nivel de fereastră dreptunghiulară

HOG pentru detectare facială



HOG pentru detectare facială



Detectorul facial Viola-Jones bazat pe glisarea ferestrei

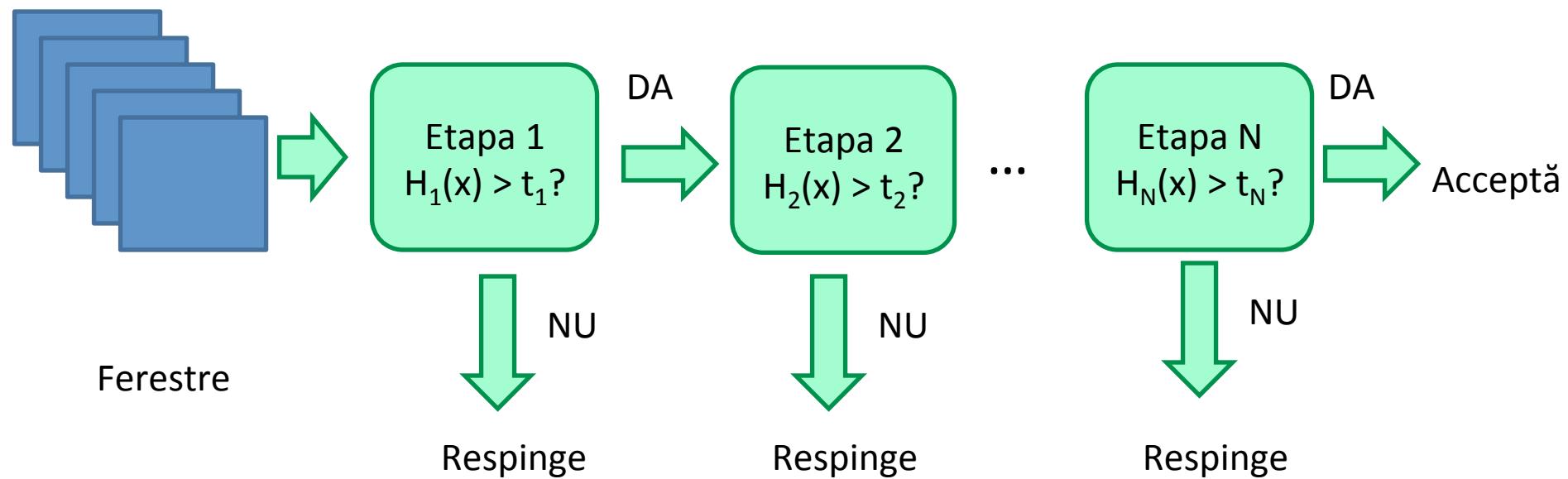
Detector rapid (15 frame-uri/secundă) bazat pe:

- eliminarea rapidă de ferestre care nu conține fețe (cu probabilitate mare) – pe bază de cascade de clasificatori
- folosirea de caracteristici discriminative, rapid de calculat

P. Viola and M. Jones. [*Rapid object detection using a boosted cascade of simple features.*](#) CVPR 2001.

P. Viola and M. Jones. [*Robust real-time face detection.*](#) IJCV 57(2), 2004.

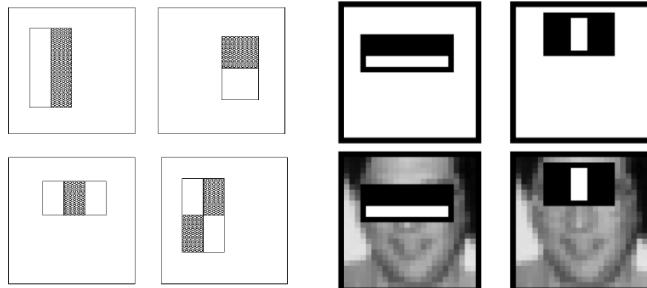
Cascade de clasificatori



- alege un threshold pentru rata a obține o rată mică de fals negative (pentru recall mare)
- clasificatori rapizi de evaluat la început, în cascadă
- clasificatori înceți de evaluat mai târziu, dar multe ferestre nu ajung aici, sunt deja respinse înainte

Extragere de caracteristici Haar

Filtre “dreptunghiulare” - caracteristici Haar

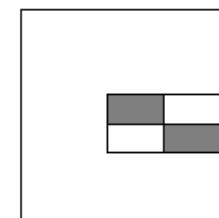
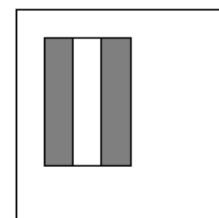
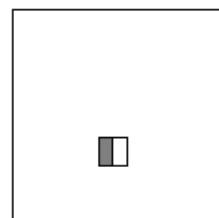
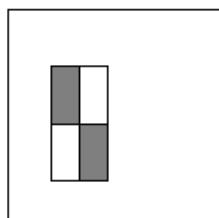
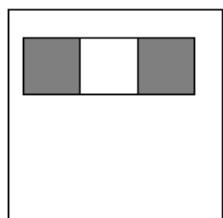
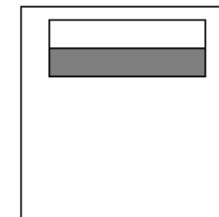
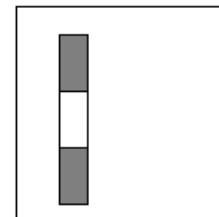
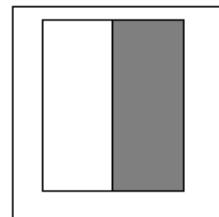
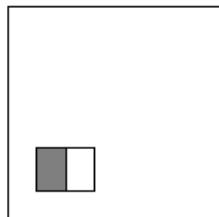
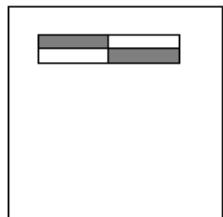
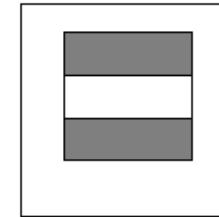
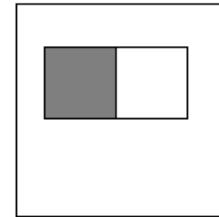
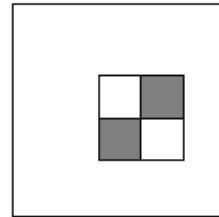
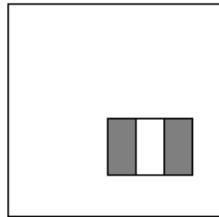
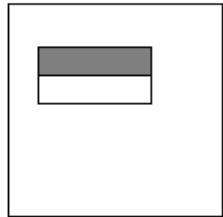


Caracteristici măsurate = diferență în intensitate dintre regiuni adiacente
(alb = +1, negru = -1)

$$\text{valoare} = \sum (\text{pixeli în regiunea albă}) - \sum (\text{pixeli în regiunea neagră})$$



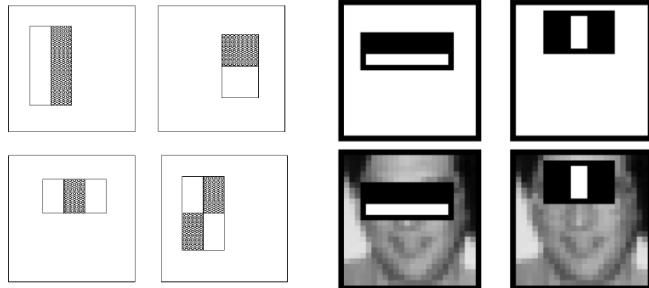
Colecție mare de filtre



Se consideră toate
filtrele posibile în
funcție de :
poziție, scală/
mărime și tip:
160,000+
caracteristici
măsurate pentru o
fereastră de
dimensiuni 24 x 24
pixeli

Extragere de caracteristici Haar

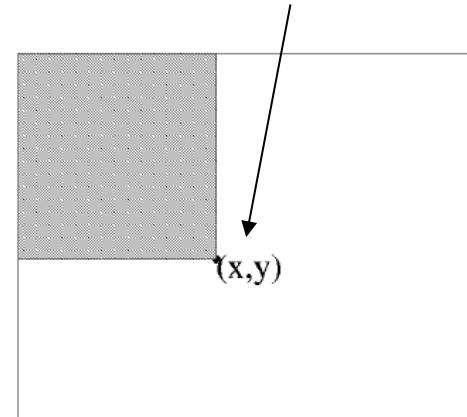
Filtre “dreptunghiulare” - caracteristici Haar



Caracteristici măsurate = diferență în intensitate dintre regiuni adiacente
(alb = +1, negru = -1)

Calcul eficient cu **imagini integrale**: suma intensităților din orice dreptunghi poate fi calculată în timp constant

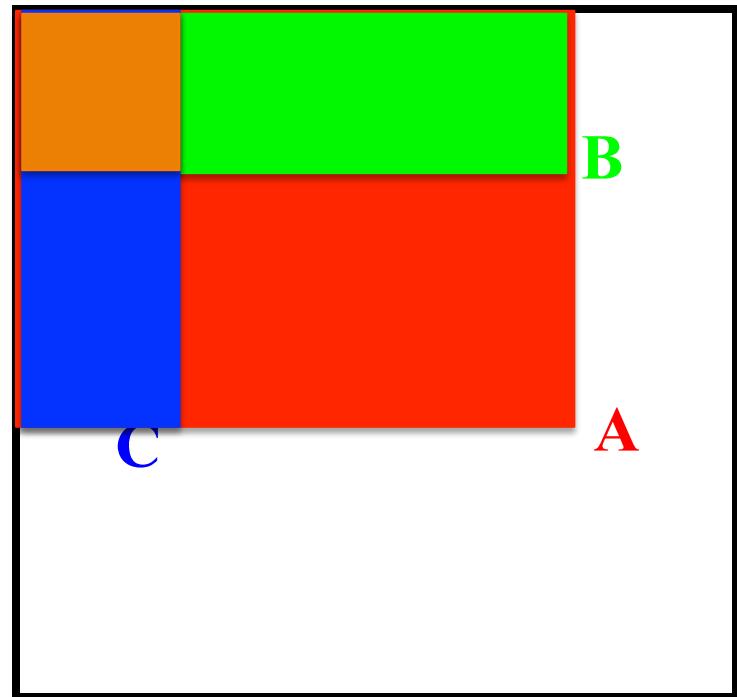
Valoarea la (x,y) este suma tuturor intensităților pixelilor deasupra și la stânga lui (x,y)



Imagine integrală

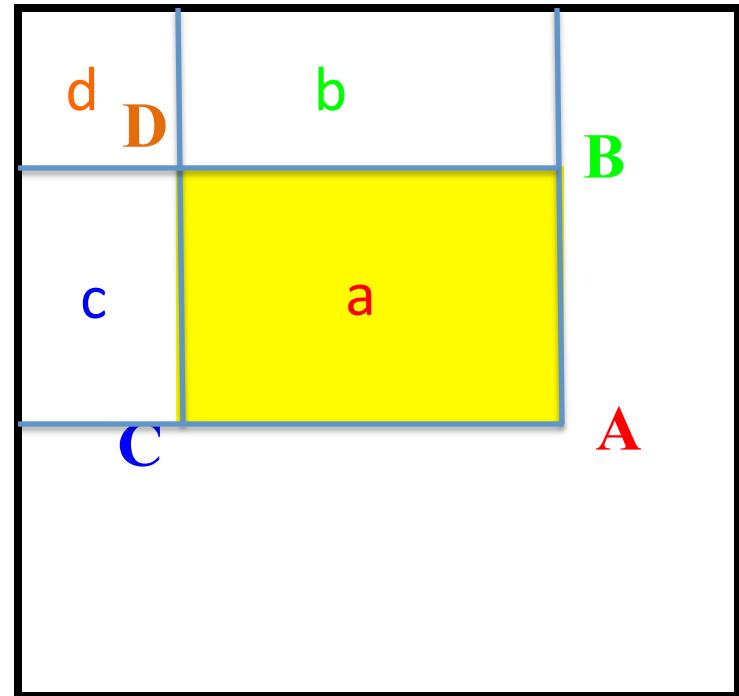
Extragere de caracteristici Haar

- Fie A,B,C si D valorile imaginii integrale pentru punctele din colțul dreptunghiului
- Suma intensităților din dreptunghiul galben poate fi calculată foarte simplu astfel:
$$\text{suma} = \textcolor{red}{A} - \textcolor{green}{B} - \textcolor{blue}{C} + \textcolor{orange}{D}$$



Extragere de caracteristici Haar

- Fie A,B,C si D valorile imaginii integrale pentru punctele din colțul dreptunghiului
- Suma intensităților din dreptunghiul galben poate fi calculată foarte simplu astfel:
$$\begin{aligned} \text{suma} &= A - B - C + D \\ &= (a + b + c + d) - (b + d) - (c + d) + d = a \end{aligned}$$
- Numai 3 operații (o adunare + 2 scăderi) necesare pentru a calcula suma intensităților din fiecare dreptunghi!



Imagine integrală

Exemplu de învățare/antrenare

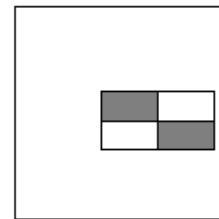
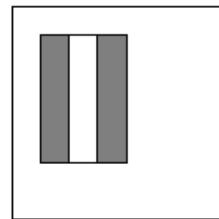
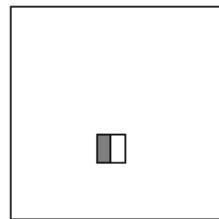
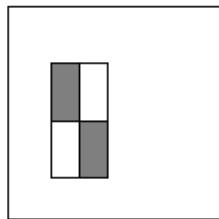
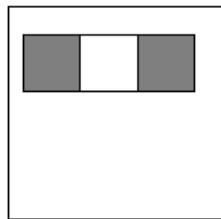
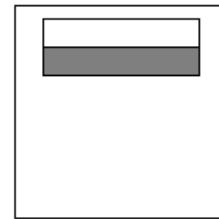
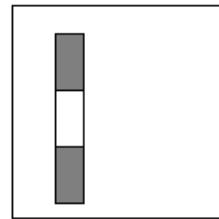
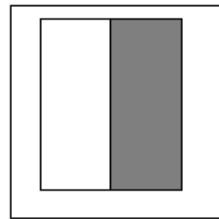
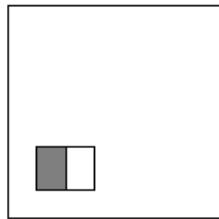
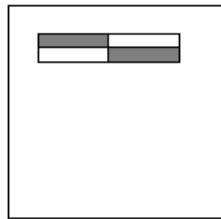
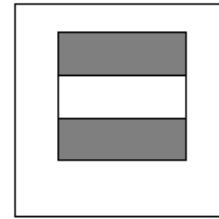
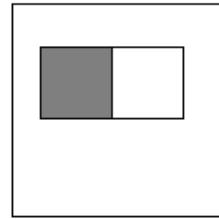
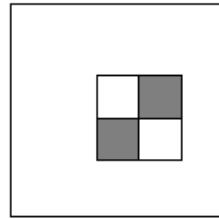
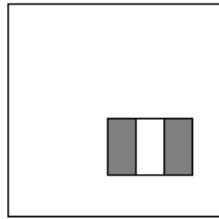
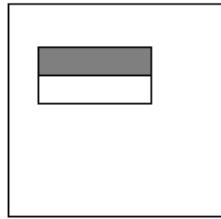
Fețe



non-Fețe



Colectie mare de filtre



Se consideră toate
filtrele posibile în
funcție de :
poziție, scală/
mărime și tip:
160,000+
caracteristici
măsurate pentru o
fereastră de
dimensiuni 24 x 24
pixeli

Selectarea caracteristicilor

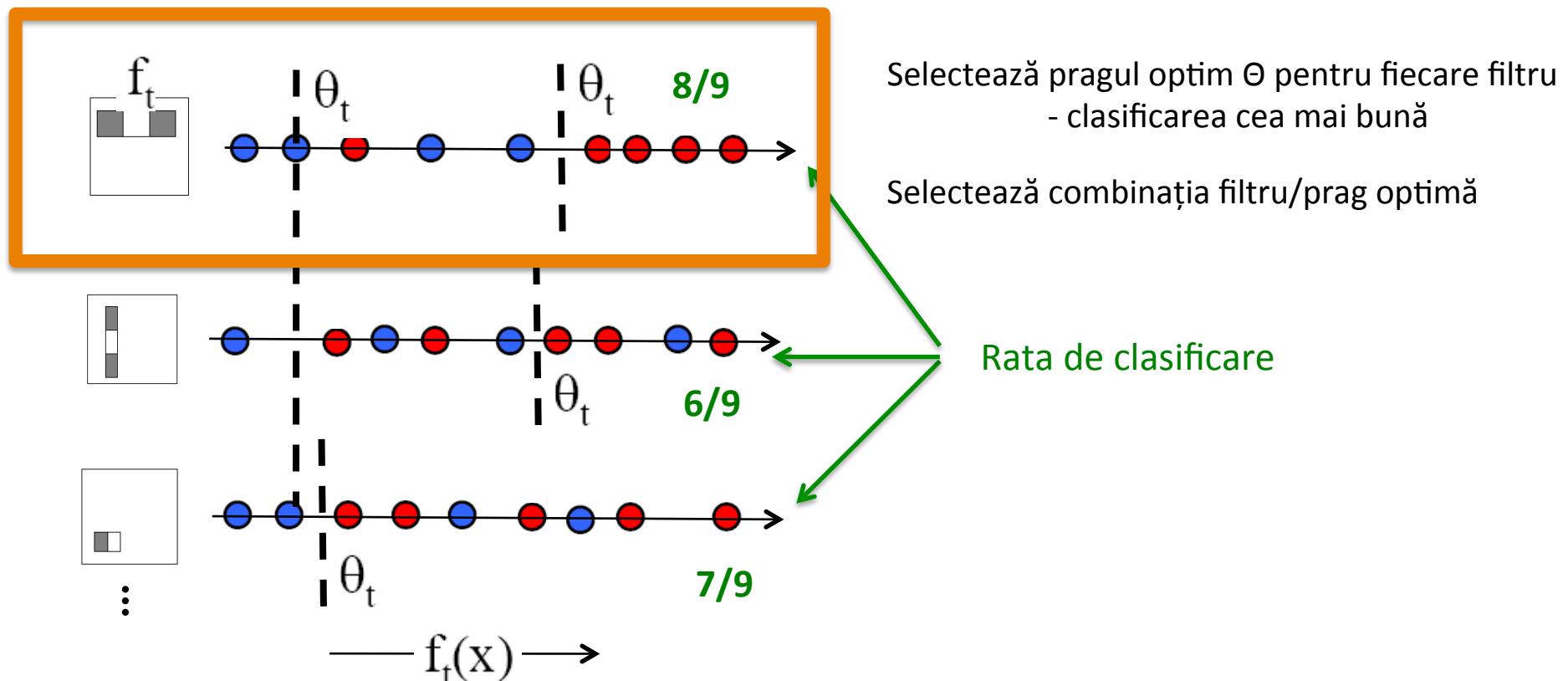
- pentru o fereastră de dimensiuni 24x24 pixeli, numărul de caracteristici Haar posibile (filtre dreptunghiulare) este $\sim 160,000$!
- pentru o imagine test, este foarte lent să evaluăm întregul set de 160,000 de caracteristici
- putem crea un clasificator bun folosind numai o submulțime mică de caracteristici (din cele $\sim 160,000$ posibile) ?
- cum să selectăm o asemenea submulțime?

Boosting

- Boosting – obține un clasificator binar ‘puternic’ (cu acuratețe mare) prin combinarea mai multor clasificatori binari ‘slabi’ (cu acuratețe mică)
 - un clasificator slab trebuie doar să performeze mai bine decât 50% în clasificarea binară (mai bine decât o simplă ‘ghicire’)
- Învățarea/antrenarea constă din mai mulți pasi:
 - la fiecare pas, selectăm clasificatorul ‘slab’ care clasifică cel mai bine exemplele care erau ‘greu’ de clasificat pentru clasificatorii anteriori
 - exemplele “greu” de clasificat – capătă ponderi (Adaptive Boosting – AdaBoost)

AdaBoost pentru selectarea caracteristicii + clasificatorului

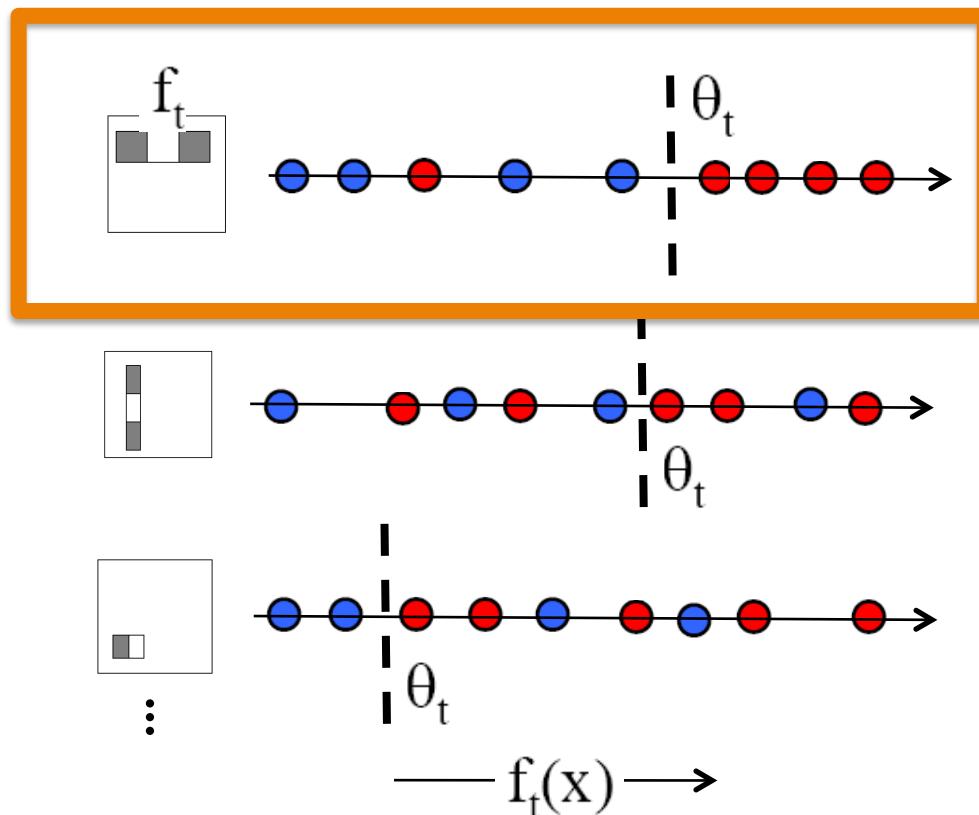
- Vrem să selectăm filtrul dreptunghiular și pragul care separă cel mai bine exemplele de învățare **pozitive (fete)** de cele **negative (non-fete)**, în termeni de #exemple incorect clasificate.



Output-ul unui filtru rectangular pe diverse imagini (**fete** sau **non-fete**)

AdaBoost pentru selectarea caracteristicii + clasificatorului

- Vrem să selectăm filtrul dreptunghiular și pragul care separă cel mai bine exemplele de învățare **pozitive (fete)** de cele **negative (non-fete)**, în termeni de #exemple incorect clasificate.



Output-ul unui filtru rectangular pe diverse imagini (**fete** sau **non-fete**)

Rezultă un clasificator slab:


$$h_t(x) = \begin{cases} +1 & \text{if } f_t(x) > \theta_t \\ -1 & \text{altfel} \end{cases}$$

Pentru pasul următor:

- ponderează exemplele în funcție de erori
- alege un alt filtru/prag

- Given example images $(x_1, y_1), \dots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \dots, T$:

- Normalize the weights,

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^n w_{t,j}}$$

so that w_t is a probability distribution.

- For each feature, j , train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t , $\epsilon_j = \sum_i w_i |h_j(x_i) - y_i|$.
- Choose the classifier, h_t , with the lowest error ϵ_t .
- Update the weights:

$$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$

where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.

- The final strong classifier is:

$$h(x) = \begin{cases} 1 & \sum_{t=1}^T \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^T \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

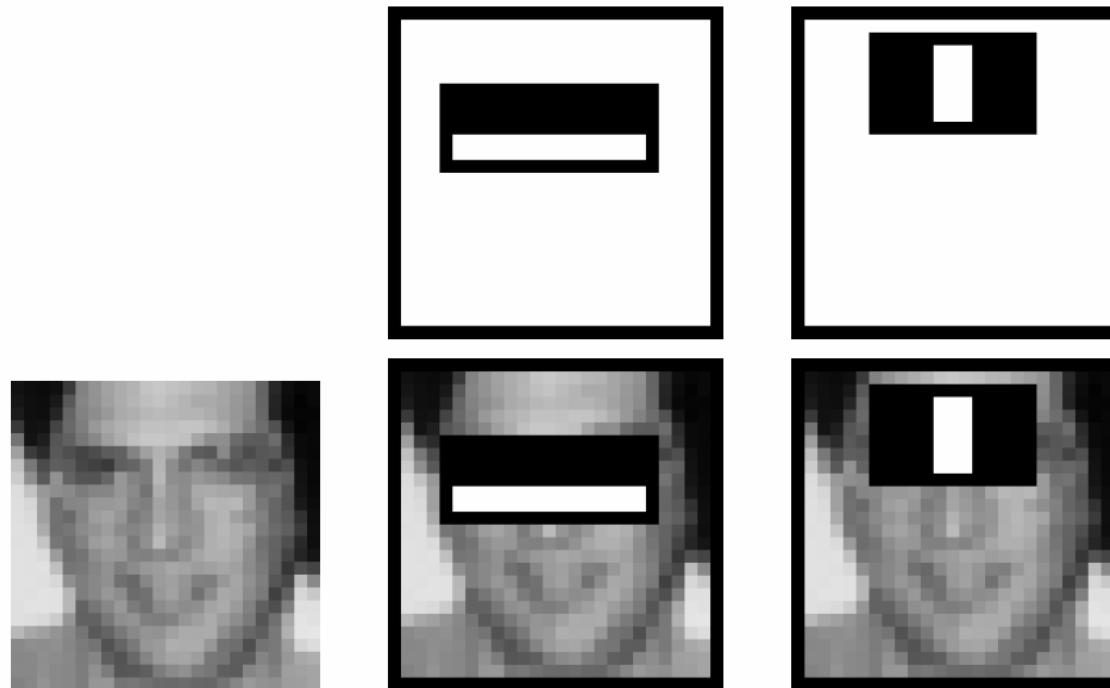
where $\alpha_t = \log \frac{1}{\beta_t}$

Etapa de învățare/antrenare

- inițial, toate exemplele de antrenare au aceeași pondere (la fel de ‘greu’ de clasificat)
- la fiecare pas:
 - găsește clasificatorul ‘slab’ care are cea mai mică rată de eroare (ponderată) pentru clasificarea exemplelor de învățare – alege combinație filtru/prag cu cea mai bună performanță de clasificare
 - mărește ponderea exemplelor de învățare clasificate greșit de actualul clasificator ‘slab’
- calculează clasificatorul final ca o combinație liniară de toți clasificatorii slabi (ponderea fiecărui clasificator ‘slab’ este direct proporțională cu performanța lui de clasificare)

Caracteristici selectate

- primele două caracteristici selectate:



această combinație de caracteristici conduce la

- rată de detectare de 100% (nu ratăm nicio față)
- 50% rată de detectii fals pozitive (jumătate din ferestrele returnate ca detectii nu sunt fețe)

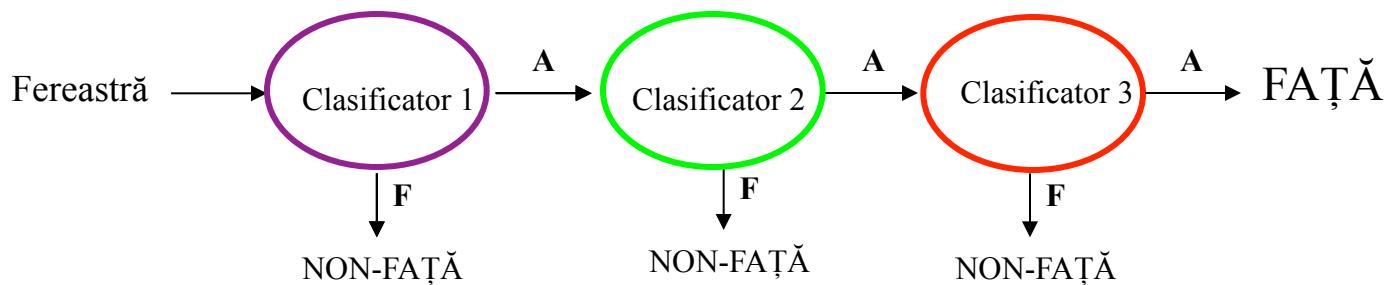
Performanță

- un clasificator cu 200 de caracteristici:
 - 95% rată de detectare (recall = câte fețe găsesc din cele existente în imagini)
 - rată de detecții fals pozitive 1 din 14084



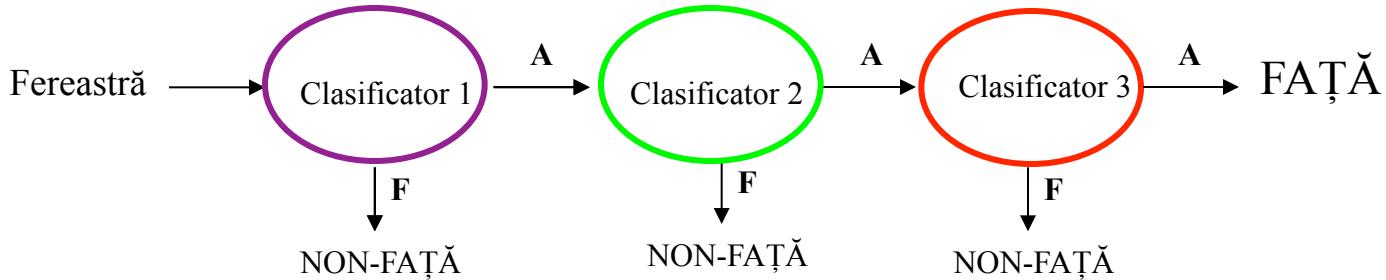
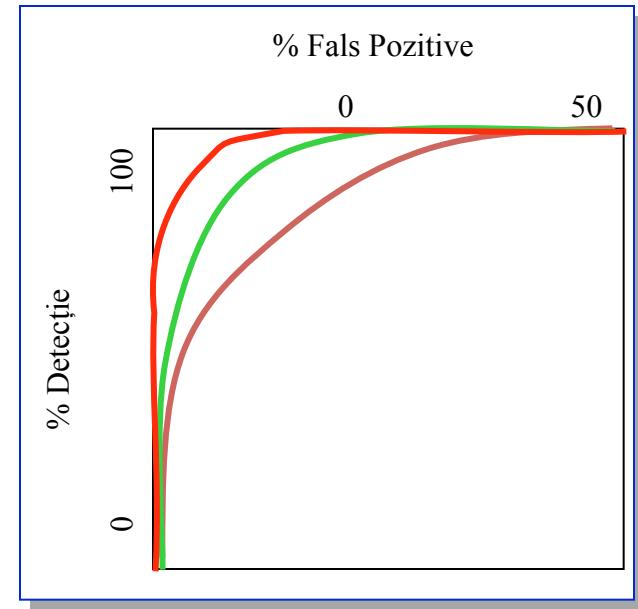
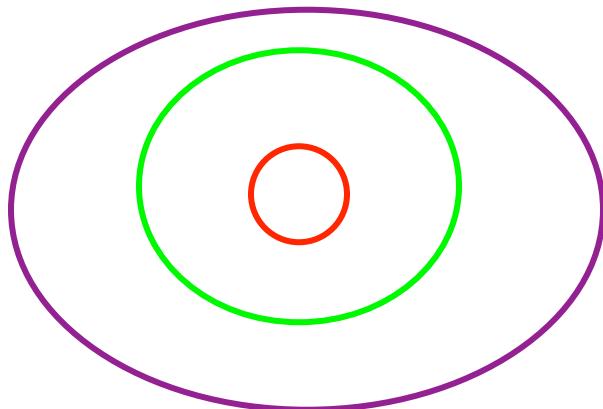
Cascadă de clasificatori

- începem cu clasificatori simpli care elimină multe dintre ferestrele non-fete, păstrând pe cele care conțin fețe
- un răspuns pozitiv pentru primul clasificator ($\text{output} > \Theta$) conduce la evaluarea unui clasificator (mai complex), și tot așa
- un răspuns negativ la orice pas conduce la eliminarea imediată a ferestrei (clasificarea ei ca un exemplu negativ – non-față)



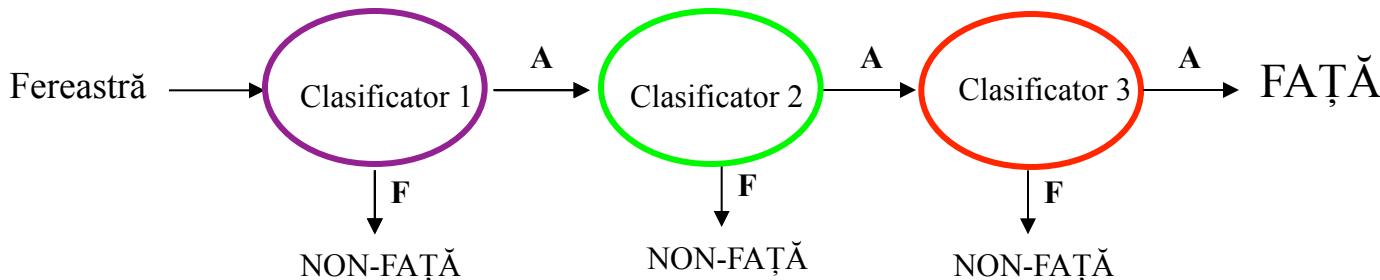
Cascadă de clasificatori

- evaluăm clasificatori în lanț care sunt din ce în mai complecși și au rată de detecție mare + rată de detectii fals pozitive mică



Cascadă de clasificatori

- rata de detectare și rata de detecții false pozitive le aflăm prin a multiplica valorile acestea de la fiecare clasificator încă din partea
- o rată de detectare de 0.9 și o rată de detecții false pozitive de ordin 10^{-6} pot fi obținute printr-o cascadă de 10 clasificatori astfel încât:
 - fiecare clasificator are o rată de detectare de 0.99 ($0.99^{10} \approx 0.9$)
 - fiecare clasificator are o rată de detecții false pozitive de 0.30 ($0.3^{10} \approx 6 \times 10^{-6}$)



Învățarea unei cascade de clasificatori

- fixăm rata de detectare și rata de detectii fals pozitive pentru fiecare etapă/clasificator
- adăugăm caracteristici (filtre/prag) la etapa curentă/ clasificatorul curent până când atingem performanța dorită (detectare + fals pozitive)
 - modificăm AdaBoost astfel încât să maximizăm rata de detectare (alegem alt prag Θ) (în loc să minimizăm eroare de clasificare)
 - testează pe un set de validare (în loc de set de învățare + test, avem set de învățare + set de validare + test)
- folosește detectiile fals pozitive de la etapa curentă ca exemple de învățare negative la etapa viitoare

Detalii de implementare

- timp de învățare: “săptămâni” pe o mașină Sun 466 MHz
- 38 de etape/clasificatori, în total 6061 caracteristici
- în medie 10 caracteristici evaluate /fereastră pentru o imagine test
- detectarea facială pentru o imagine 383 x 288 pixeli în 0.067 secunde

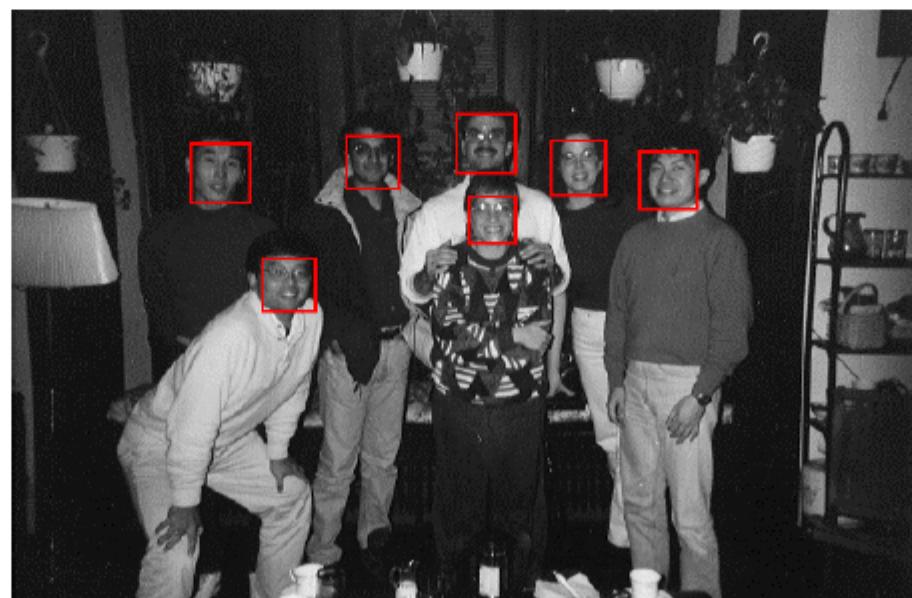
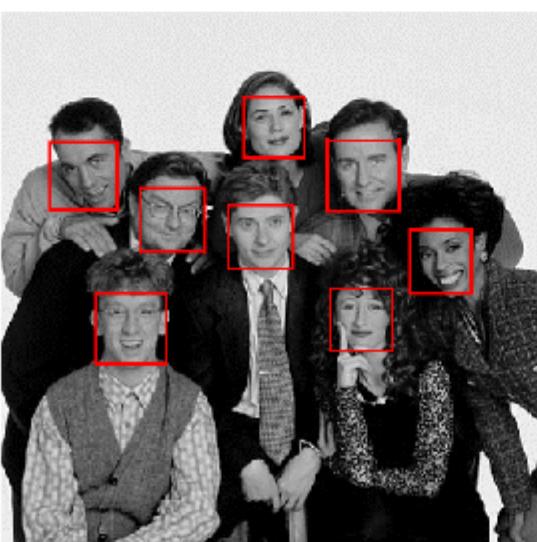
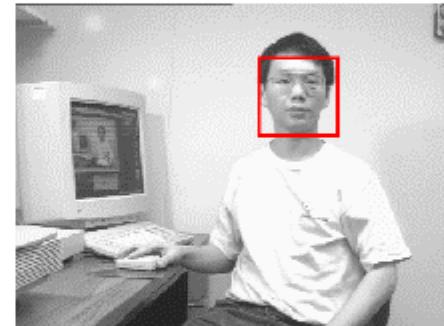
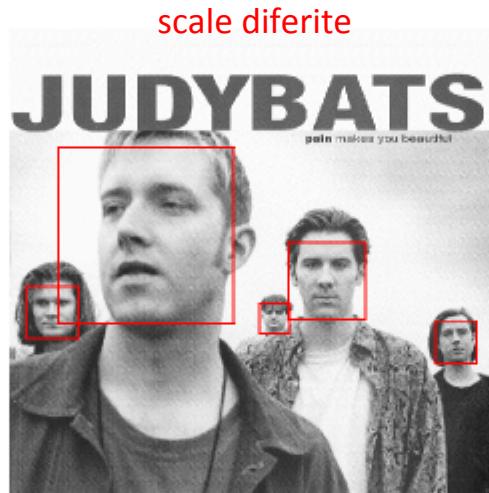
Detectorul facial Viola-Jones: sumar



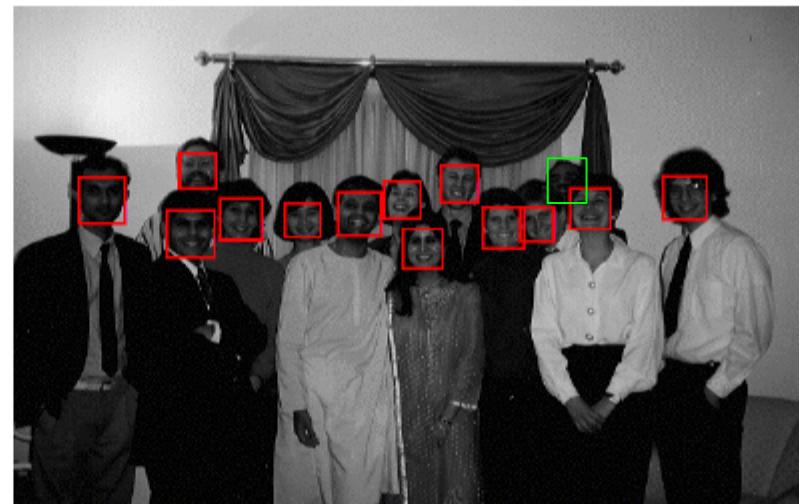
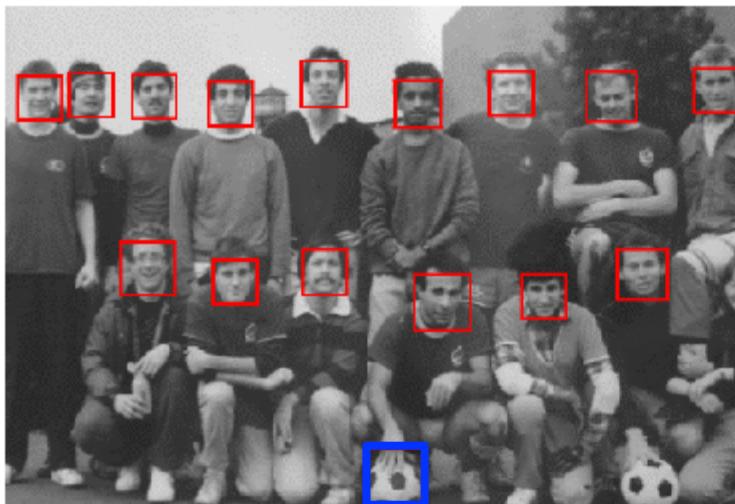
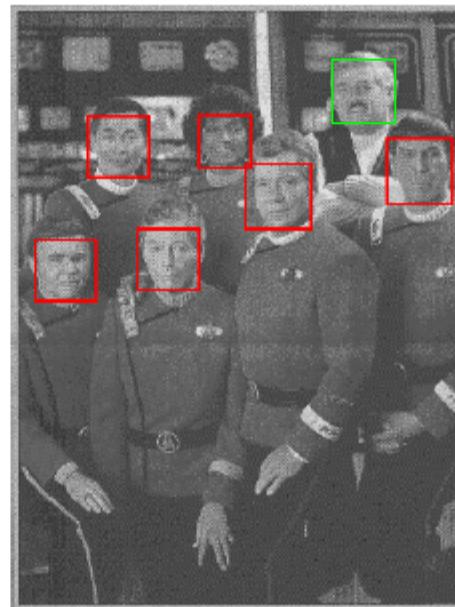
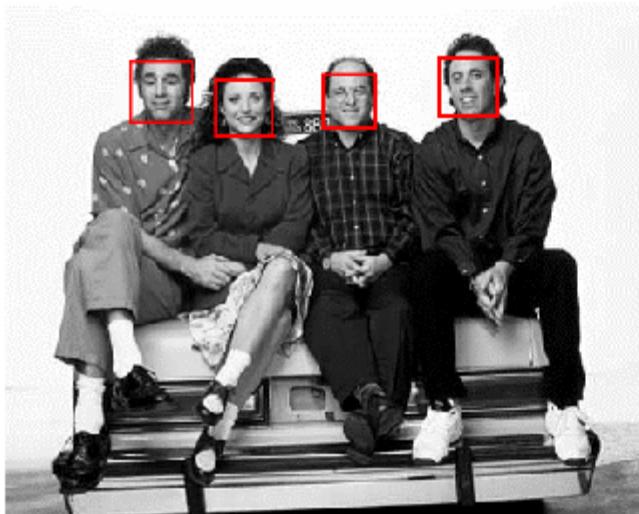
- î învață cu 5K exemple pozitive, 300M exemple negative
- detector facial în timp real - cascadă de 38 de clasificatori
- 6061 de caracteristici în total

Detectorul facial Viola-Jones: rezultate

Recall = $2/5 = 40\%$



Detectorul facial Viola-Jones: rezultate



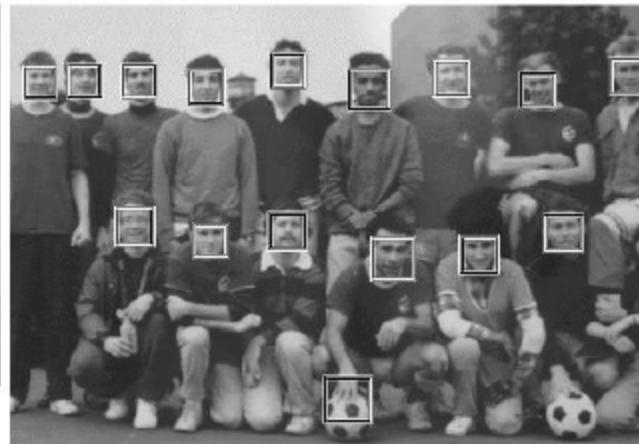
1 exemplu fals pozitiv pe imagine

Detectorul facial Viola-Jones: rezultate



Viola Jones Results

Speed = 15 FPS (in 2001)



Detector	10	31	50	65	78	95	167
Viola-Jones	76.1%	88.4%	91.4%	92.0%	92.1%	92.9%	93.9%
Viola-Jones (voting)	81.1%	89.7%	92.1%	93.1%	93.1%	93.2 %	93.7%
Rowley-Baluja-Kanade	83.2%	86.0%	-	-	-	89.2%	90.1%
Schneiderman-Kanade	-	-	-	94.4%	-	-	-
Roth-Yang-Ahuja	-	-	-	-	(94.8%)	-	-

MIT + CMU face dataset