

Concepțe și aplicații în Vederea Artificială

Bogdan Alexe

bogdan.alexe@fmi.unibuc.ro

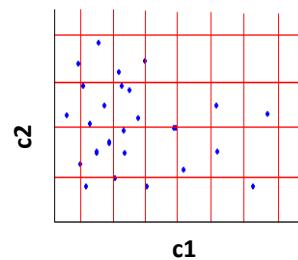
Radu Ionescu

radu.ionescu@fmi.unibuc.ro

Curs optional
semestrul I, 2023-2024

Cursul trecut

- Tema 1: Calculator automat de scor pentru jocul Double Double Dominoes
<https://tinyurl.com/CAVA-2023-TEMA1>
- Clasificarea imaginilor
 - reprezentarea caracteristicilor prin histograme
- Localizarea claselor de obiecte
 - metoda ferestrei glisante

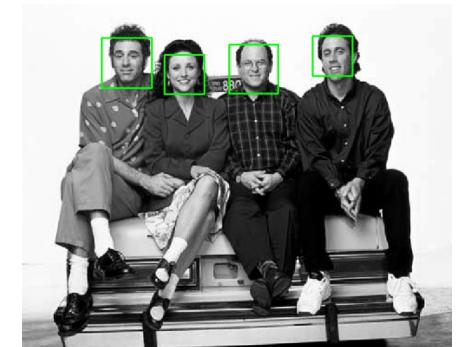


Cursul de azi

- Localizarea claselor de obiecte
 - metoda ferestrei glisante



- Detectare facială folosind metoda glisării ferestrei și histograme de gradienți orientați

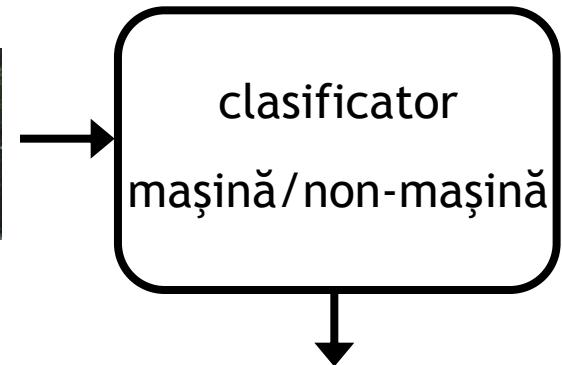




Localizarea obiectelor la nivel de fereastră:
metoda ferestrei glisante (sliding-window)

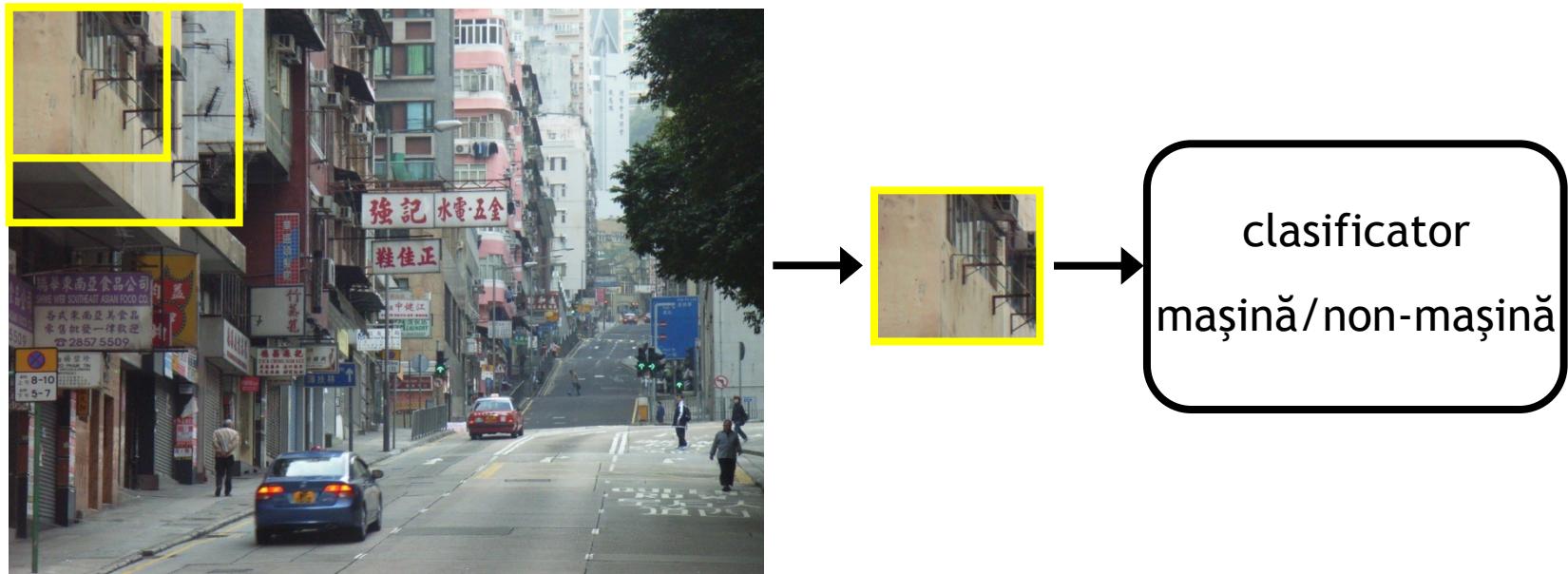
Ideea principală: detectare via clasificare

Componența de bază: un clasificator binar



Ideea principală: detectare via clasificare

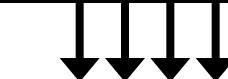
Dacă obiectul este într-o scenă aglomerată, glisează o fereastră căutând obiectul.



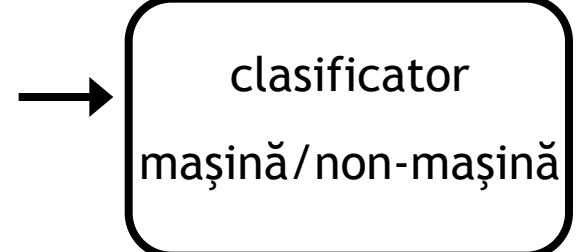
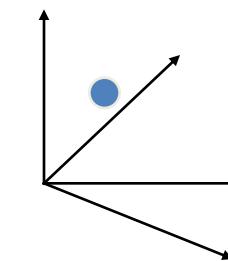
Ideea principală: detectare via clasificare

Pași:

1. Obținem exemple de învățare (pozitive + negative)
2. Definim caracteristici
3. Definim clasificator



Exemple de învățare

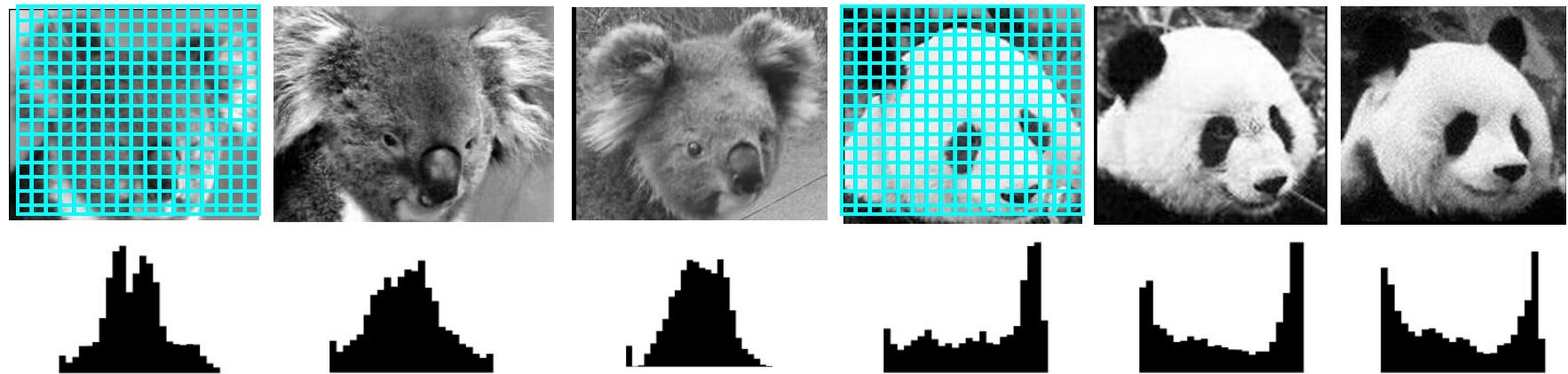
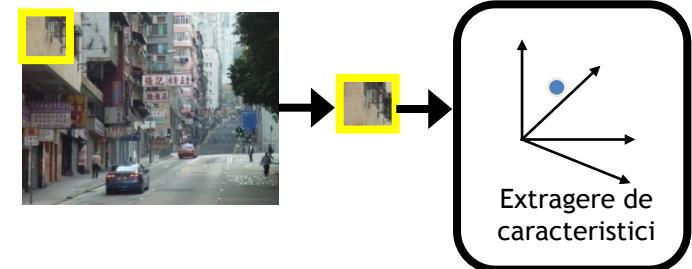


Extragere de
caracteristici

Ideea principală: detectare via clasificare

- Considerăm toate ferestrele dintr-o imagine
 - consideră ferestre poziționate în fiecare pixel, de mărimi diferite (+ orientări diferite)
- Pentru fiecare fereastră decide:
 - “Contine sau nu această fereastră o instanță a clasei de obiecte X?”

Extragere de caracteristici: înfățisarea globală a unei ferestre

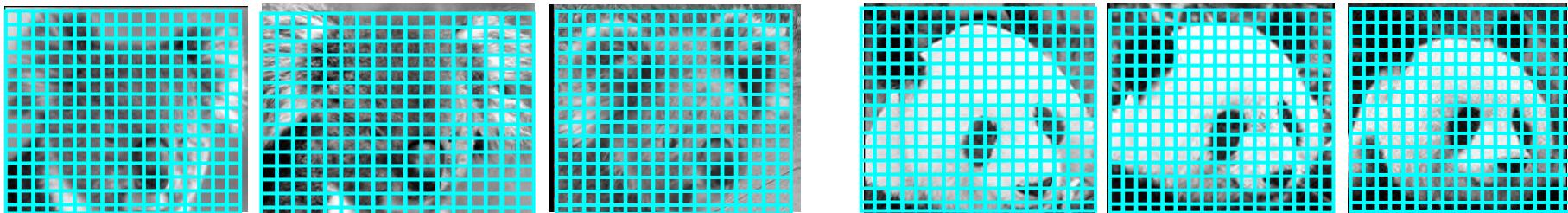


Descriptori ai conținutului vizual al imaginii/ferestrei

- histograme de intensități tonuri de gri/ culori
- vectori cu intensități ale pixelilor

Extragere de caracteristici: înfățisarea globală a unei ferestre

- reprezentări pe baza de pixeli – sensitive la mici translații

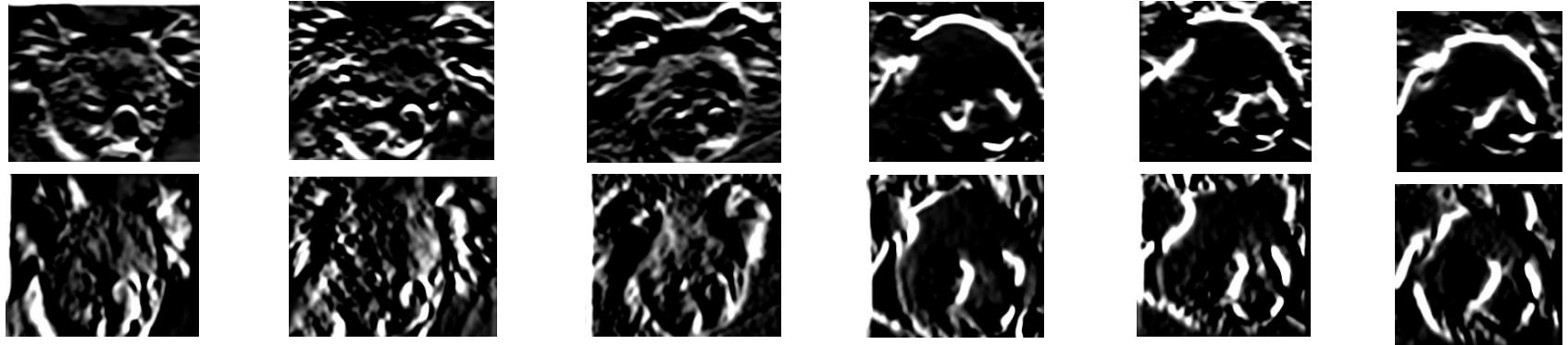


- descriptori pe bază de culoare/intensități de tonuri de gri sensitivi la iluminarea scenei și la variabilitatea înfățisării intra-clasă



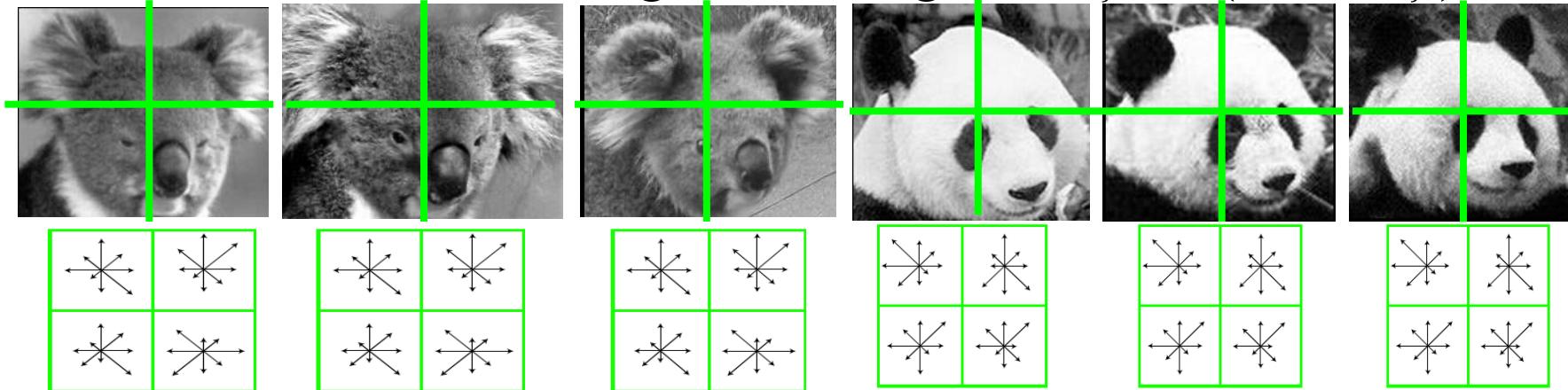
Reprezentări bazate pe gradienti

- muchiile, contururi, magnitudinea gradientilor (orientați)



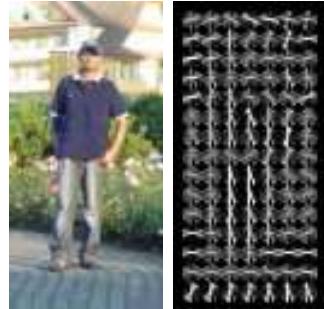
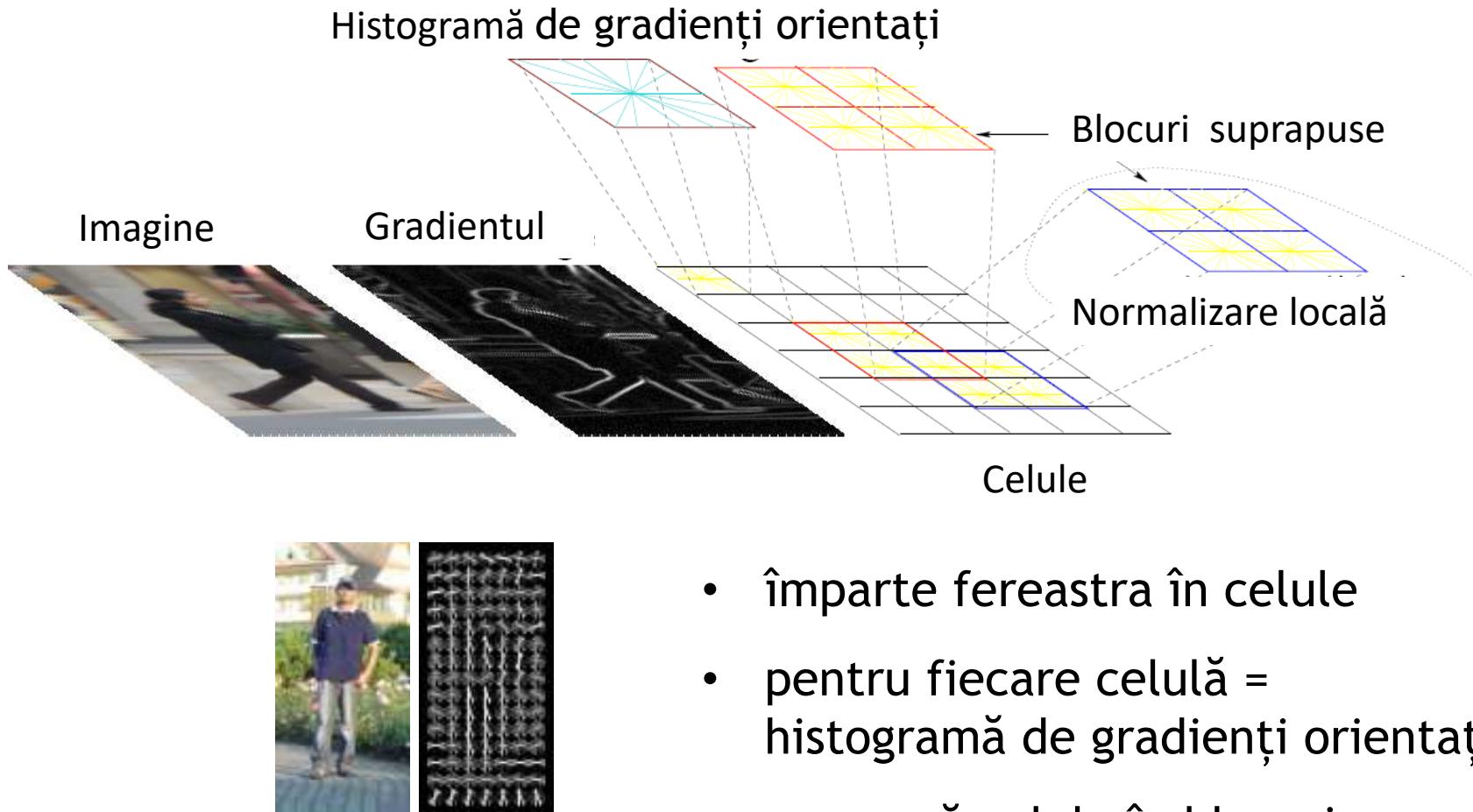
Reprezentări bazate pe gradienti

- muchiile, contururi, magnitudinea gradientilor (orientați)



- descrie distribuția locală a gradientilor cu o histogramă
 - local neordonată: oferă invarianță la mici translații și rotații
 - normalizare a contrastului: invarianță la iluminare

HOG pentru detectarea oamenilor



- împarte fereastra în celule
- pentru fiecare celulă = histogramă de gradienți orientați
- grupează celule în blocuri
- HOG = histograme de blocuri

Navneet Dalal and Bill Triggs,
Histograms of Oriented Gradients for Human Detection, CVPR05
<https://www.youtube.com/watch?v=7S5qXET179I>
<http://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf>

HOG pentru detectarea oamenilor



Slide adaptat după N.Dalal

HOG pentru detectarea oamenilor

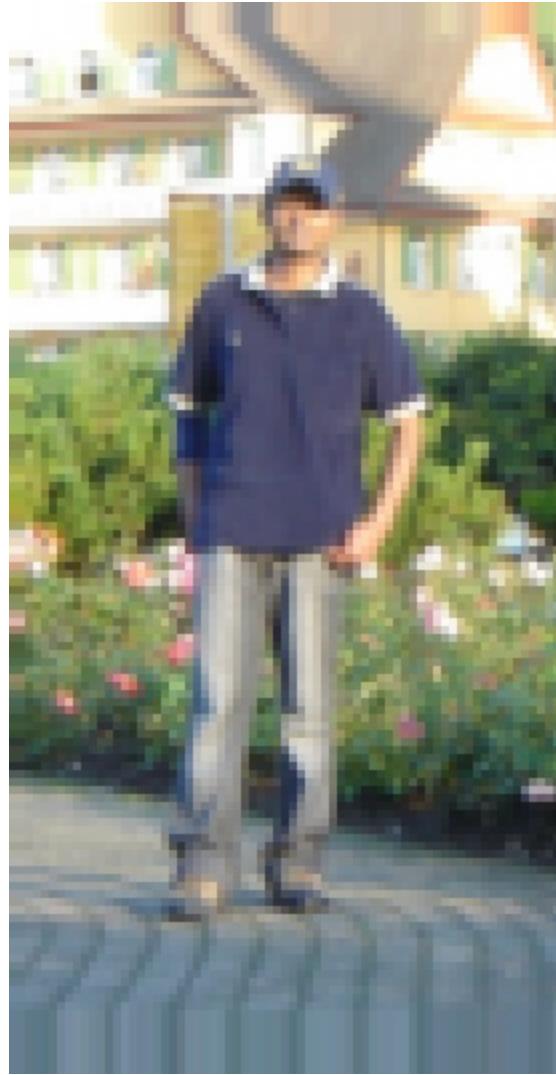
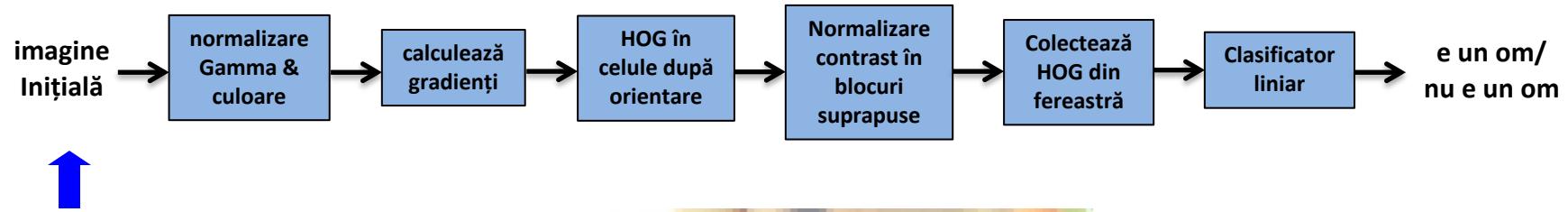


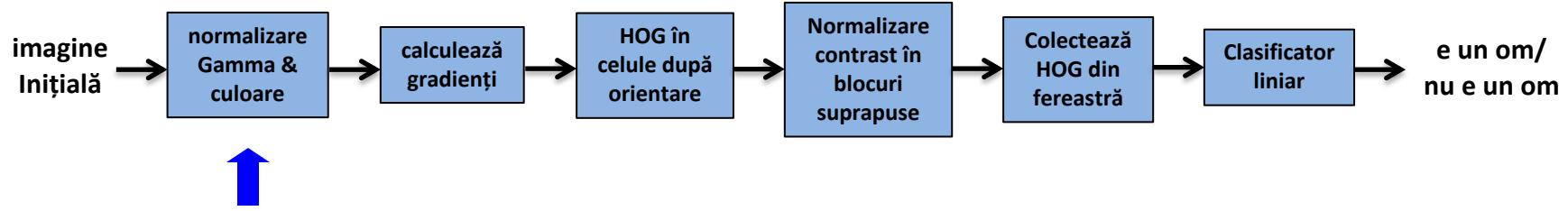
- redimensionează fereastra la 128 x 64 pixeli
- împarte imaginea în celule, fiecare celulă are 8 x 8 pixeli
- calculează gradientul imaginii (pentru fiecare pixel avem orientare și magnitudine)
- pentru fiecare celulă calculează o histogramă de gradienți orientați
- grupează celule în blocuri
- HOG = descriptor obținut prin concatenarea histogramelor a 105 blocuri

Navneet Dalal and Bill Triggs,
Histograms of Oriented Gradients for Human Detection, CVPR05

<https://www.youtube.com/watch?v=7S5qXET179I>
<http://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf>

Slide adaptat după K. Grauman





- Testat cu

- RGB
- tonuri de gri

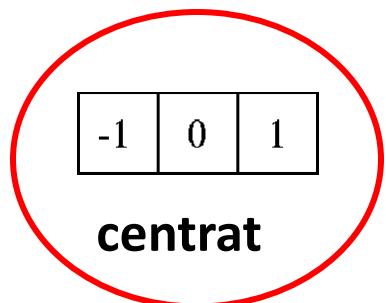
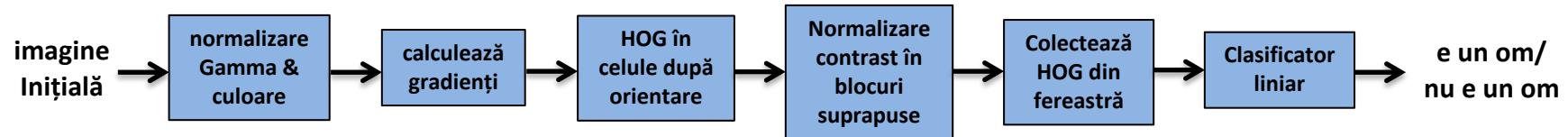
} performanță



- Normalizare Gamma (transformarea valorilor intensităților/colorilor)

- fără normalizare
- rădăcină pătrată
- logaritm

} performanță puțin mai bună

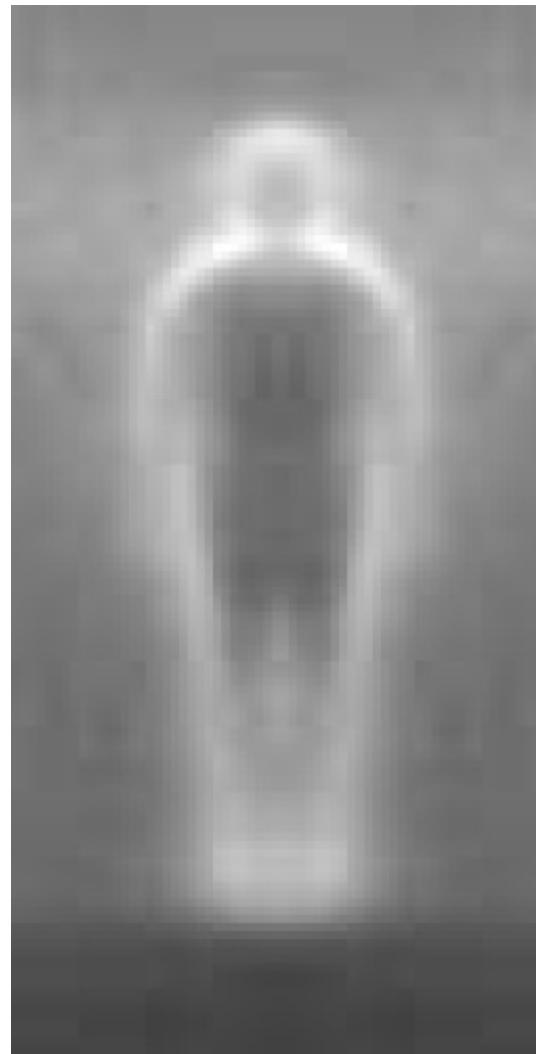


-1	1
----	---

necentrat

1	-8	0	8	-1
---	----	---	---	----

cubic corectat

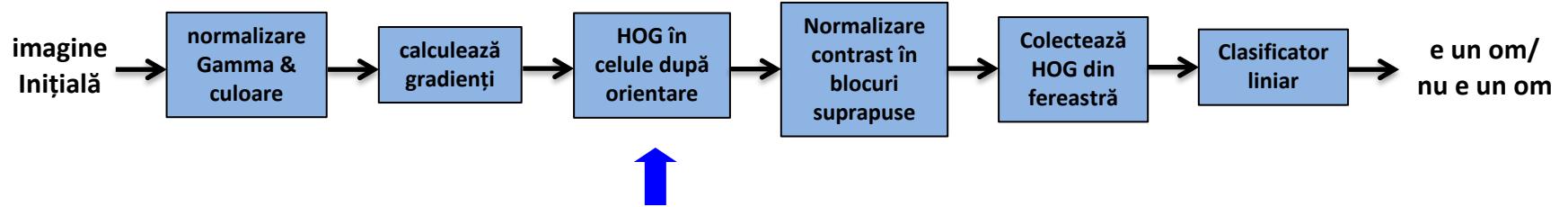


0	1
-1	0

diagonal

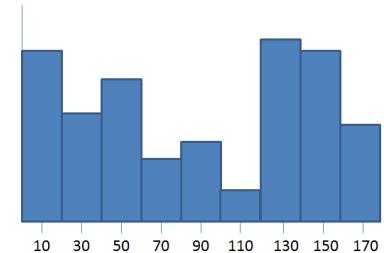
-1	0	1
-2	0	2
-1	0	1

Sobel

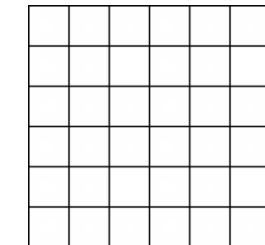


- histograme de gradienți orientați

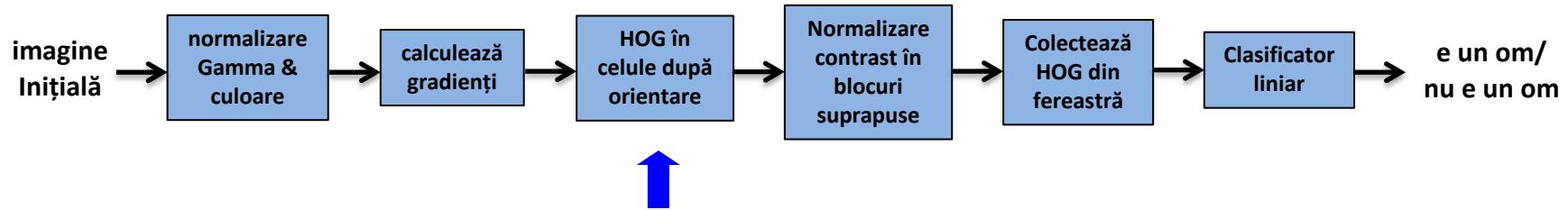
- după orientare: 9 intervale pentru orientări între $[0^\circ, 180^\circ]$



- după celule



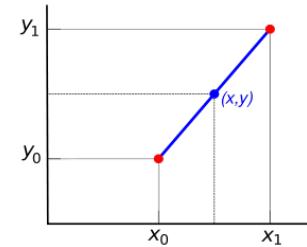
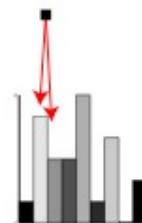
- contribuția fiecărui pixel direct proporțională cu magnitudinea gradientului



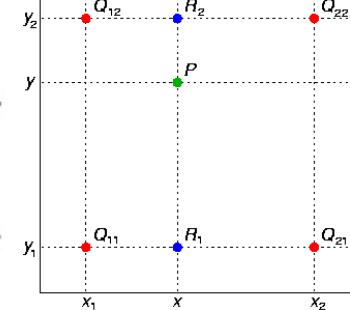
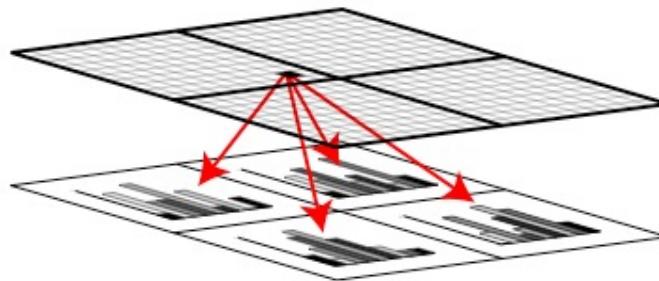
- histograme de gradienți orientați

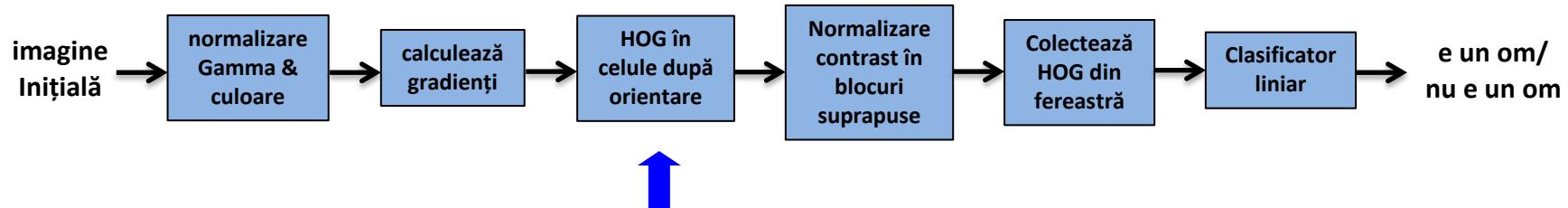
- interpolare triliniară:

- liniară după orientare



- biliniară după celule



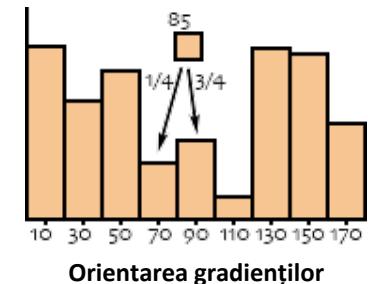


- exemplu de interpolare

- interpolare triliniară:

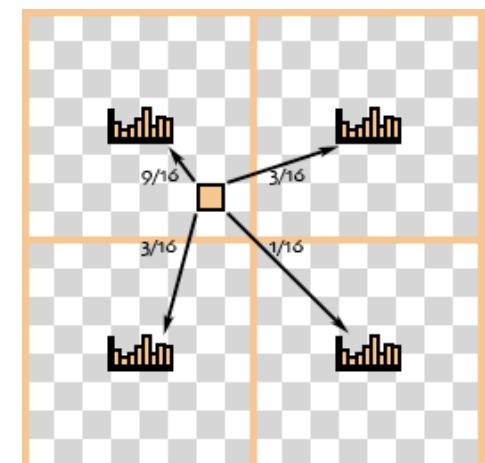
- liniară după orientare

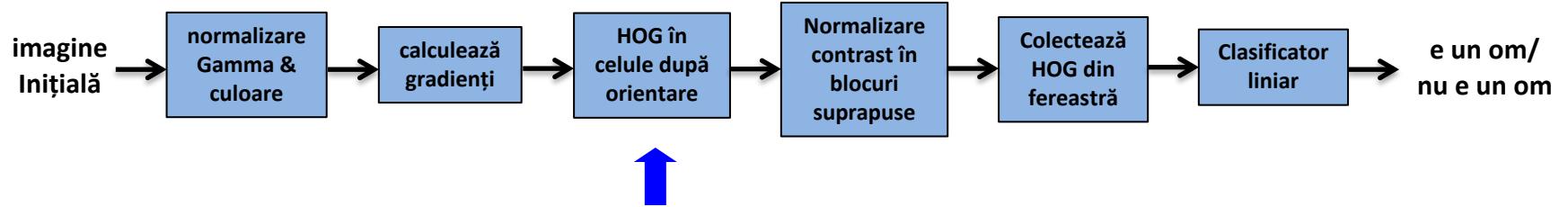
- $\Theta = 85^\circ$
 - distanțe față de centri $70^\circ, 90^\circ$ sunt de $15^\circ, 5^\circ$
 - $5/20 = \frac{1}{4}$ contribuție în intervalul cu centru 70°
 - $15/20 = \frac{3}{4}$ contribuție în intervalul cu centru 90°
 - 'soft-assignment'



- biliniară după celule

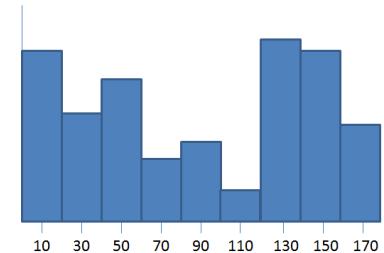
- distanțe față de centri celulelor: stânga – 2, dreapta – 6, sus – 2, jos – 6
 - contribuții parțiale: $6/8 = \frac{3}{4}$ - stânga, $2/8 = \frac{1}{4}$ dreapta
 - contribuții parțiale: $6/8 = \frac{3}{4}$ - sus, $2/8 = \frac{1}{4}$ jos
 - contribuții totale:
 - $6/8 * 6/8 = 9/16$ stânga – sus
 - $6/8 * 2/8 = 3/16$ stânga – jos
 - $2/8 * 6/8 = 3/16$ dreapta – sus
 - $2/8 * 2/8 = 1/16$ dreapta – jos



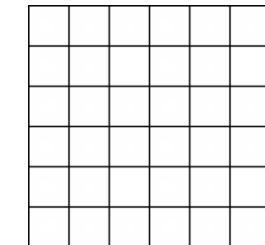


- histograme de gradienți orientați

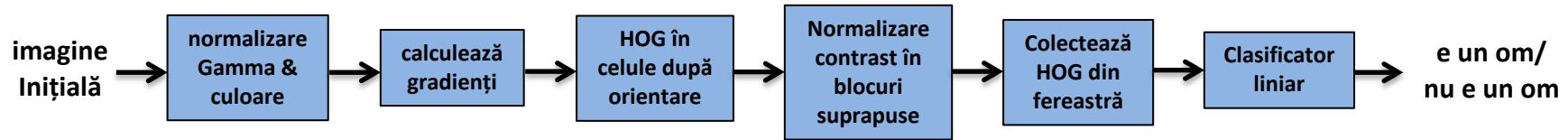
- după orientare: 9 intervale pentru orientări între $[0^\circ, 180^\circ]$



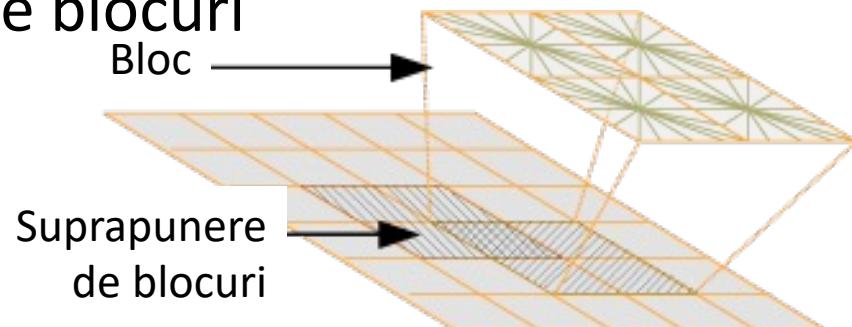
- după celule



- contribuția fiecărui pixel direct proporțională cu magnitudinea gradientului



- bloc = grup de 2x2 celule
- histograma blocului = concatenarea histogramelor celor 4 celule componente (histogramele sunt normalizate în funcție de bloc)
- o celulă face parte din mai multe blocuri
- normalizare
 - diferite normalizări posibile

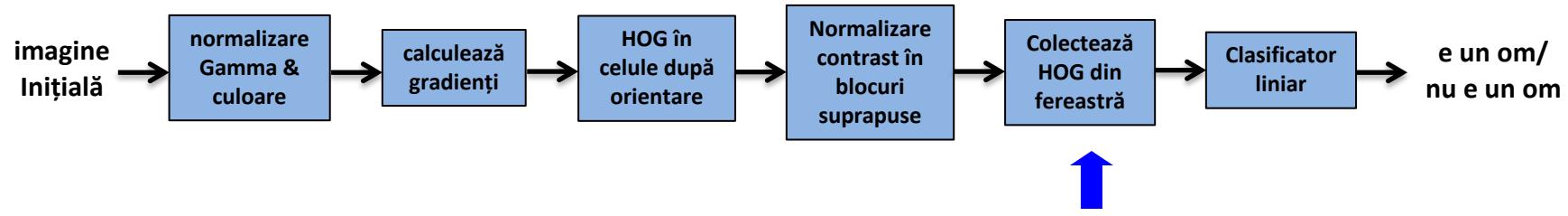


$$\mathbf{v} \rightarrow \frac{\mathbf{v}}{\|\mathbf{v}\|_1 + \epsilon}$$

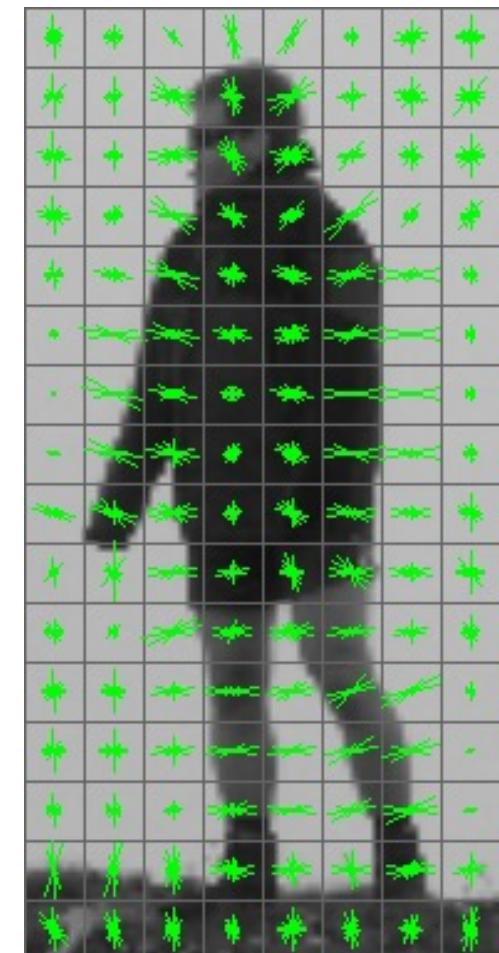
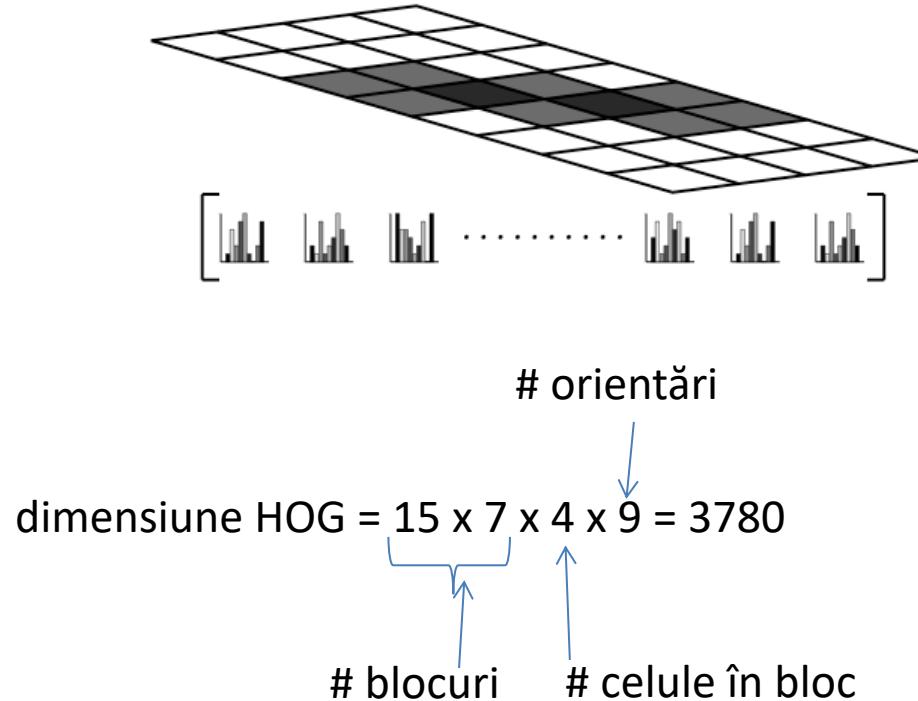
$$\mathbf{v} \rightarrow \frac{\mathbf{v}}{\sqrt{\|\mathbf{v}\|_2^2 + \epsilon}}$$

$$\mathbf{v} \rightarrow \sqrt{\frac{\mathbf{v}}{\|\mathbf{v}\|_1 + \epsilon}}$$

Slide adaptat după N. Dalal



- concatenare de histograme de blocuri

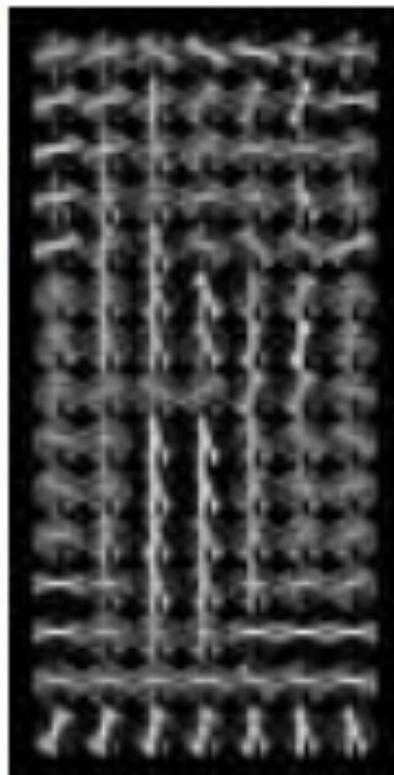


Mașini cu vectori suport + HOG

- Vizualizare



Imagine test



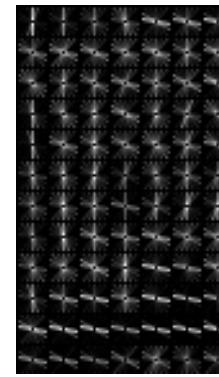
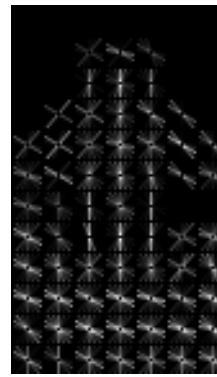
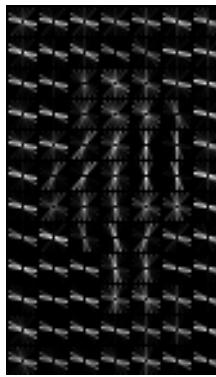
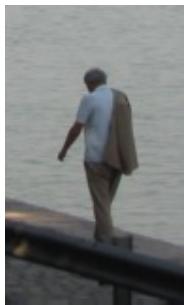
descriptorul HOG



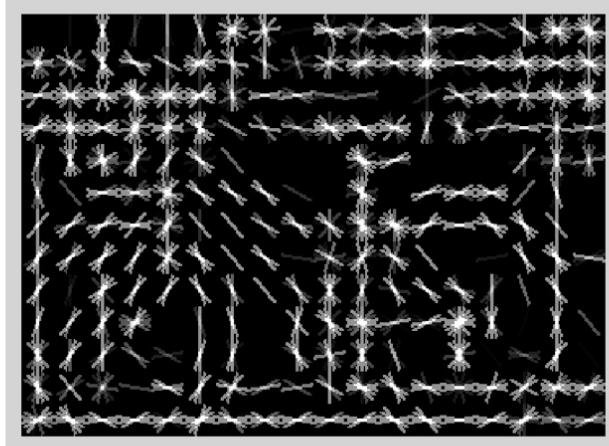
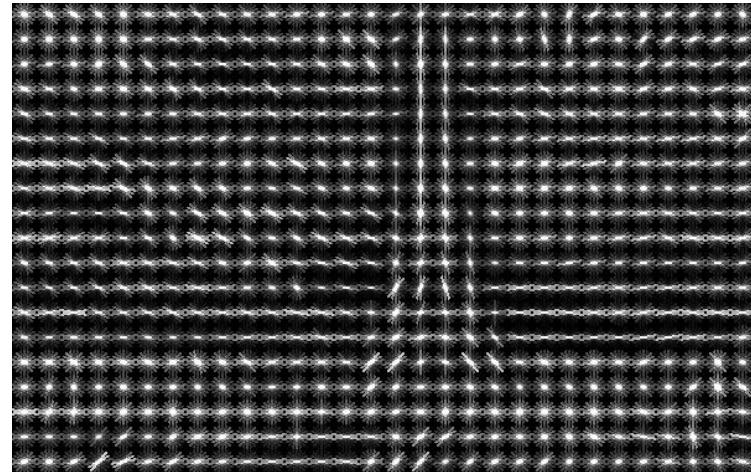
descriptorul HOG
(doar orientările pentru
ponderi pozitive din w)

Mașini cu vectori suport + HOG

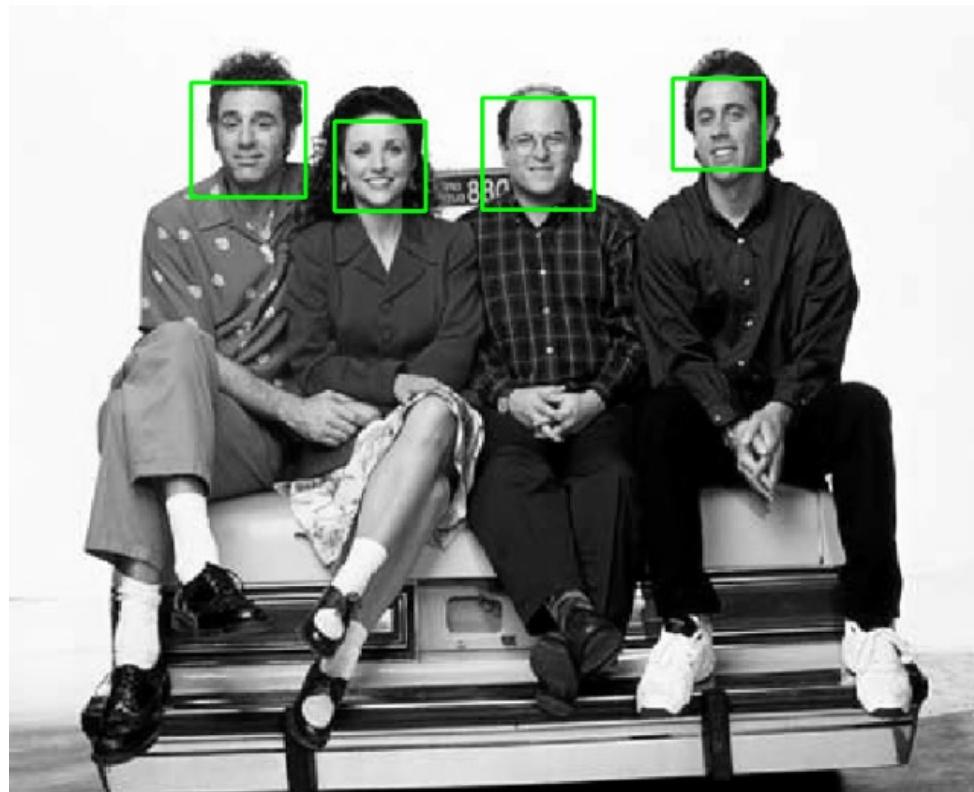
- Vizualizare



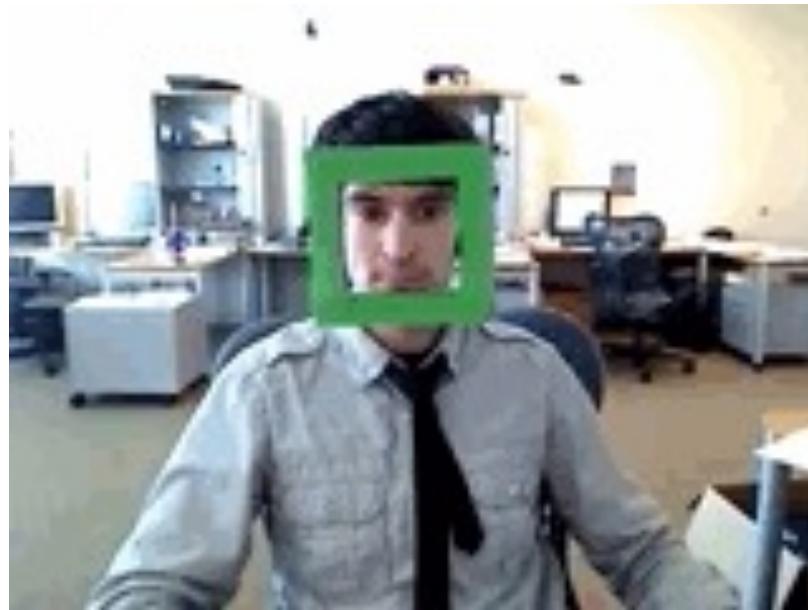
HOG și pentru alte clase



Detectare facială



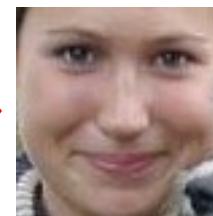
Demo – cel mai bun sistem de detectare facială



Detectare facială și identificare



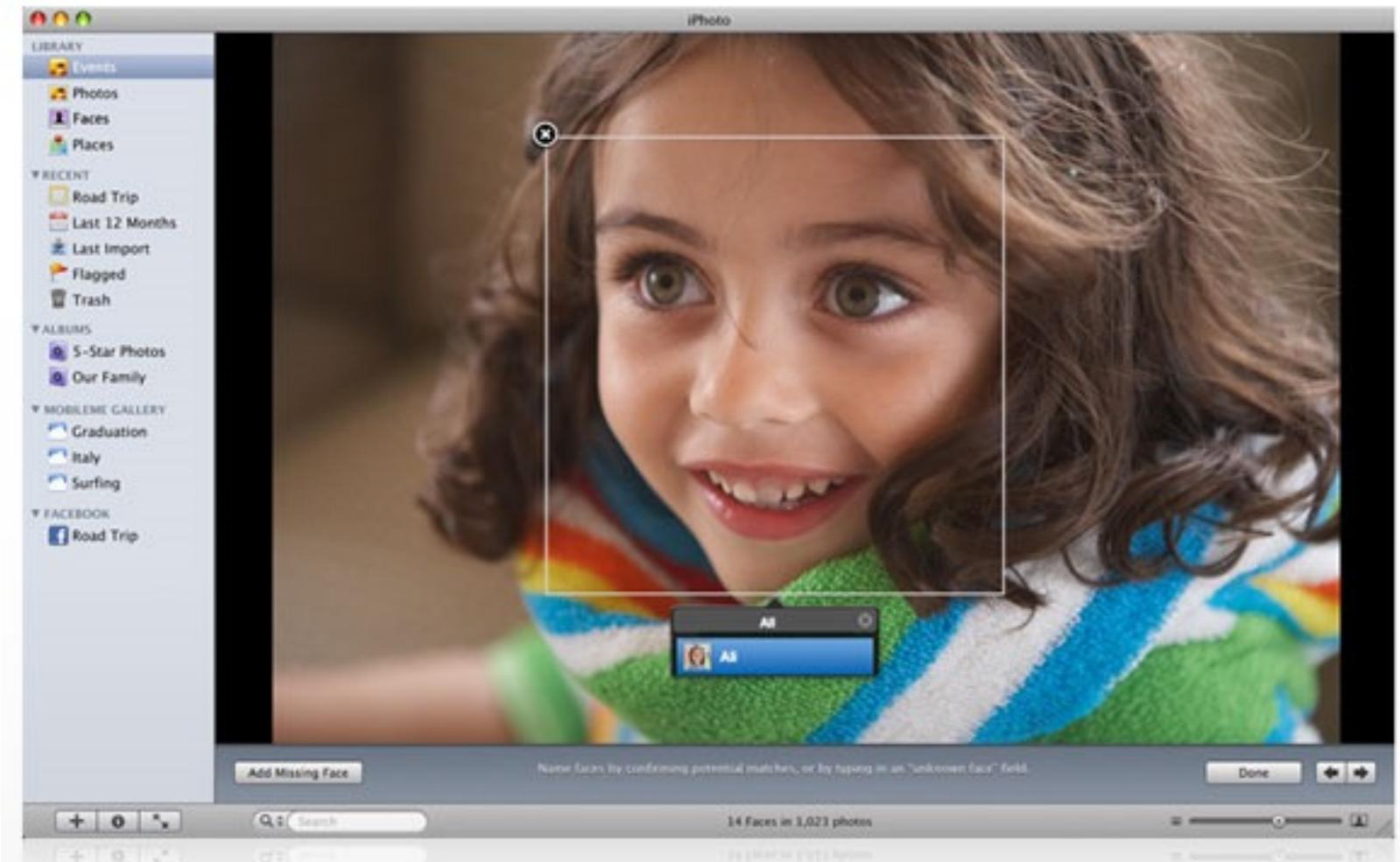
Detectare



Identificare

“Maria”

Aplicație: Apple iPhoto

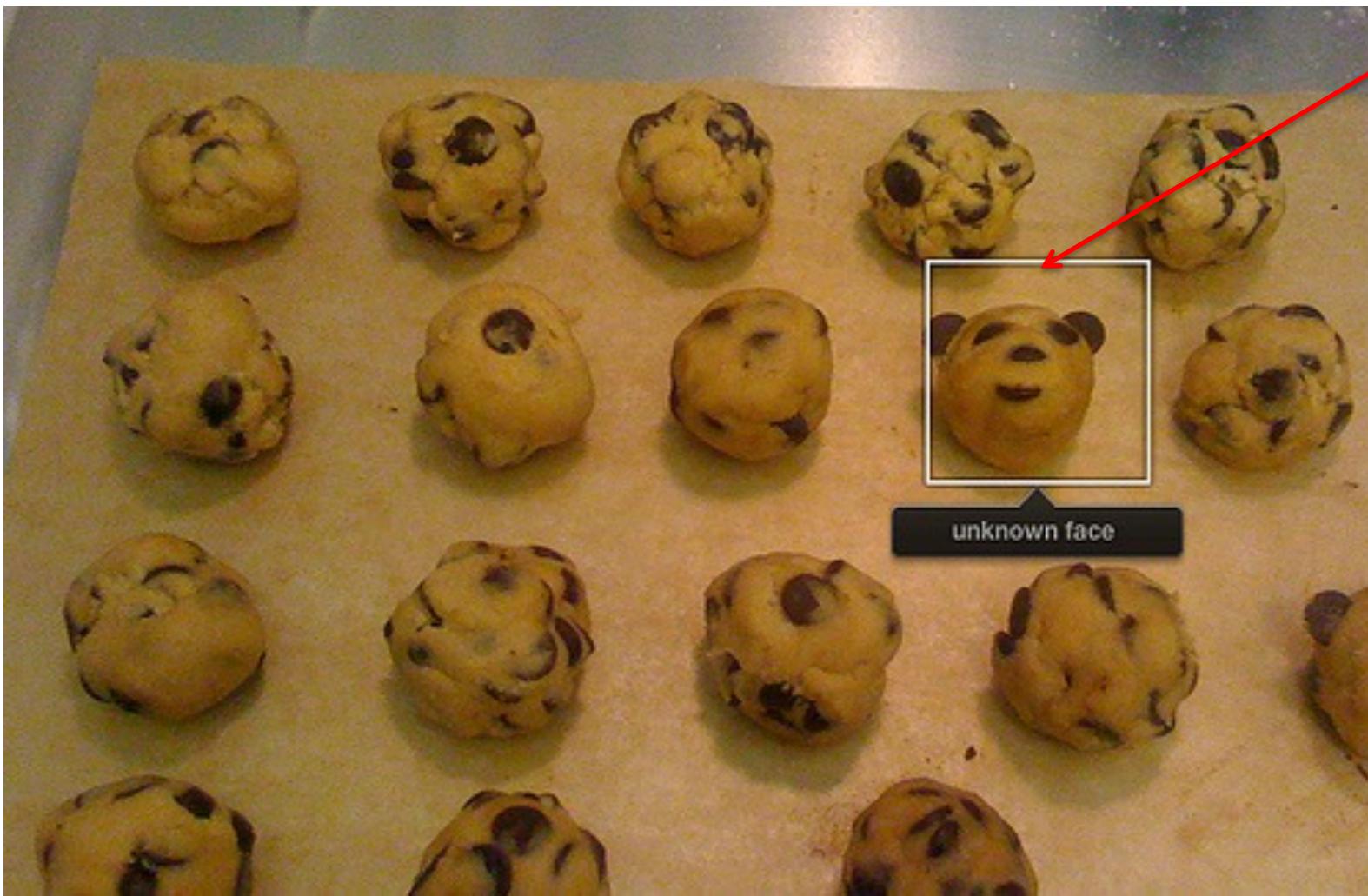


<http://www.apple.com/ilife/iphoto/>

Aplicație: Apple iPhoto

**Exemplu
Fals Pozitiv**

(detectorul
găsește o față
care nu există)



Reclamă Nikon

"Nikon S60 poate detecta până la 12 fețe."



De ce o problemă grea detectarea facială?

- **postura:** frontal, profil
- **prezența 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 fotografie:** lumina + caracteristicile camerei afectează înfățisarea unei fețe

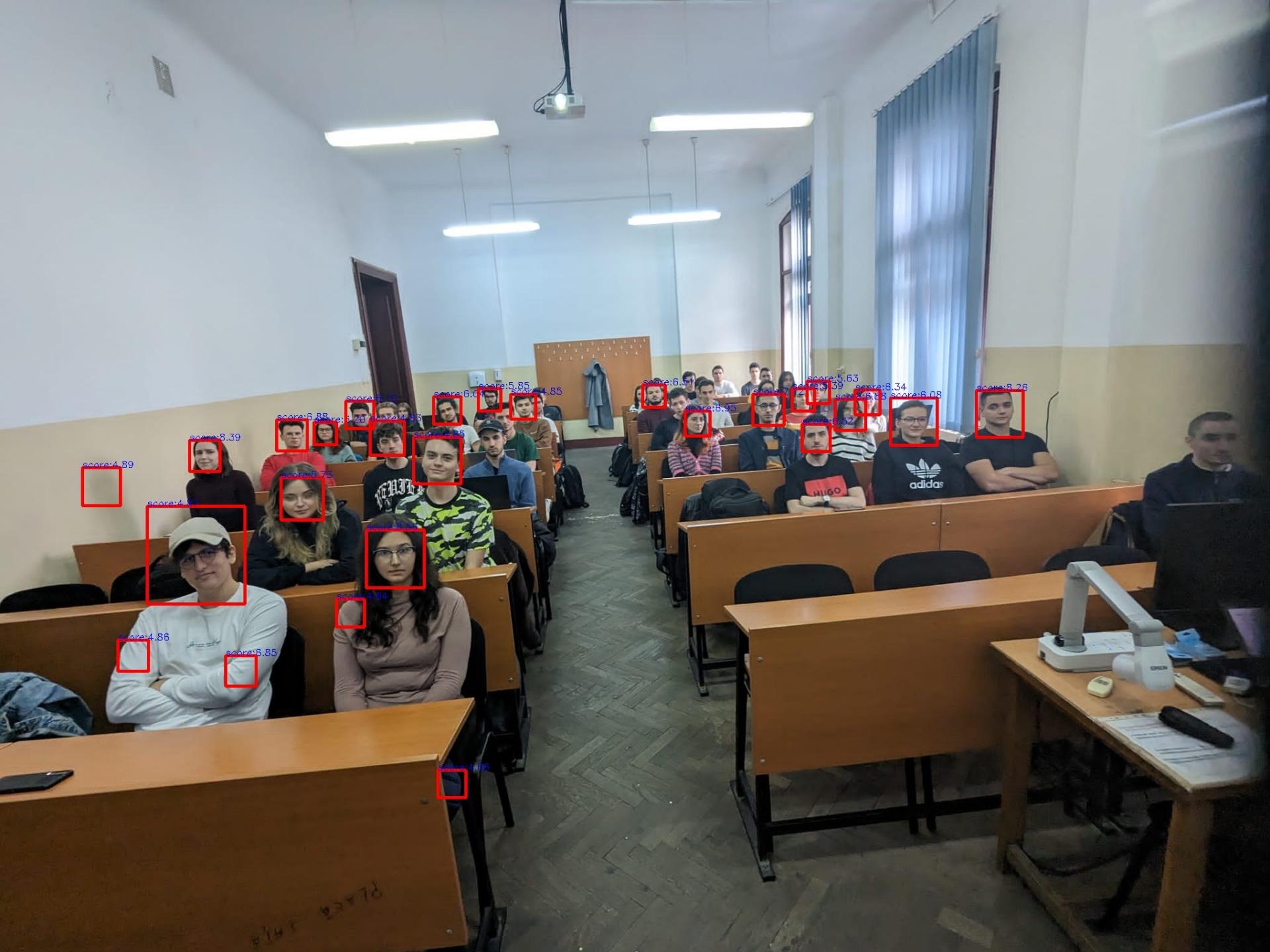
Abordări pentru detectarea facială

Există multe abordări de succes. Cele mai cunoscute:

1. detector bazat pe metoda glisării ferestrei și HOG (acest curs)
2. detectorul Viola-Jones (implementat în OpenCV/Matlab)
3. detector bazat pe metoda vectorilor proprii (eigen-faces)
4. detector bazat pe rețele neuronale de tip deep (master)









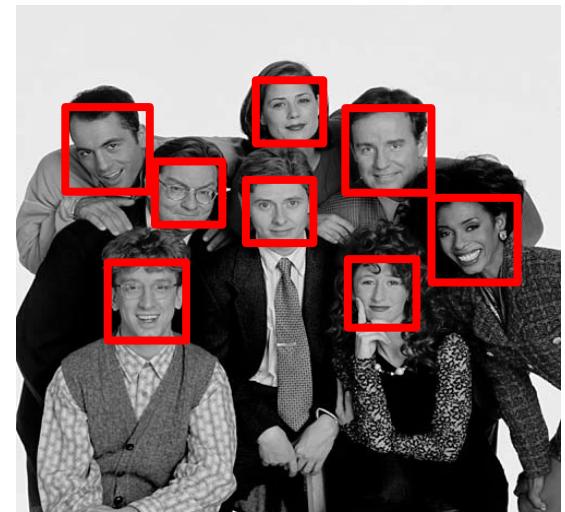
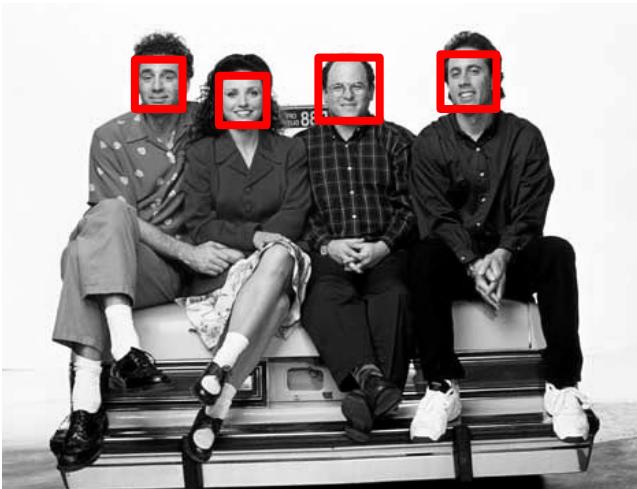
Detectare facială folosind metoda glisării ferestrei și histograme de gradienți orientați

Unde se află în imagine fețele umane?



Detectare facială folosind metoda glisării ferestrei și histograme de gradienți orientați

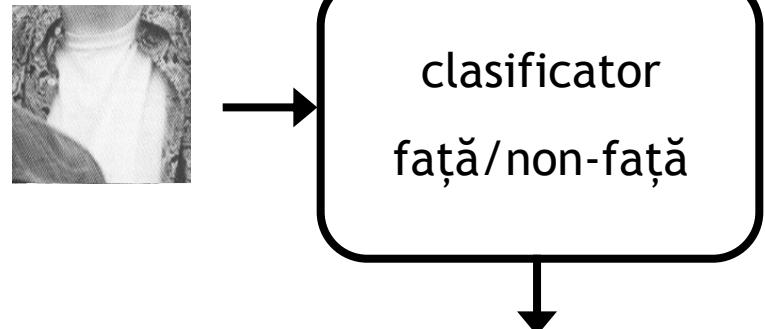
Unde se află în imagine fețele umane?



Localizare la nivel de fereastră dreptunghiulară

Metoda glisării ferestrei

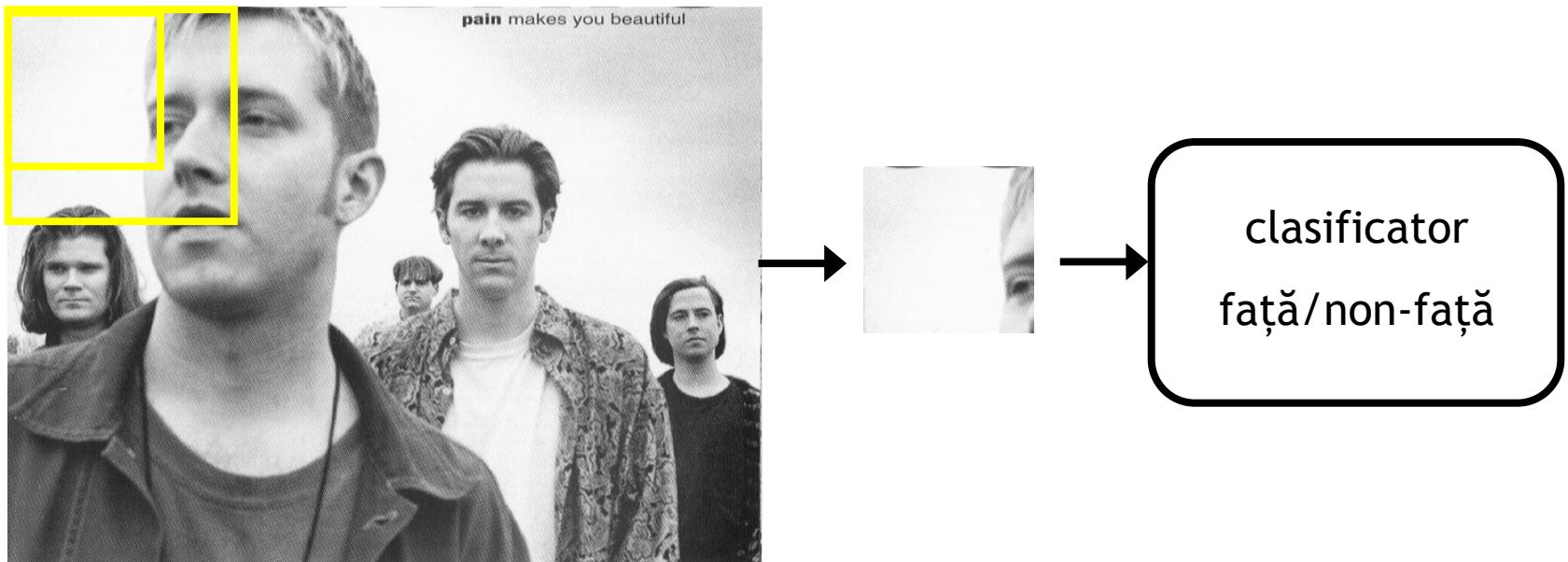
- Localizarea obiectelor la nivel de fereastră:
metoda ferestrei glisante (sliding-window)
 - detectare via clasificare: **clasificator binar** → conține sau nu fiecare fereastră din imagine instanță a clasei de obiecte X (față)?
 - consideră **ferestre poziționate în fiecare pixel**, de **mărimi diferite**



NDau e față.

Metoda glisării ferestrei

- Localizarea obiectelor la nivel de fereastră:
metoda ferestrei glisante (sliding-window)
 - detectare via clasificare: **clasificator binar** → conține sau nu fiecare fereastră din imagine instanță a clasei de obiecte X (față)?
 - consideră **ferestre poziționate în fiecare pixel**, de **mărimi diferite**



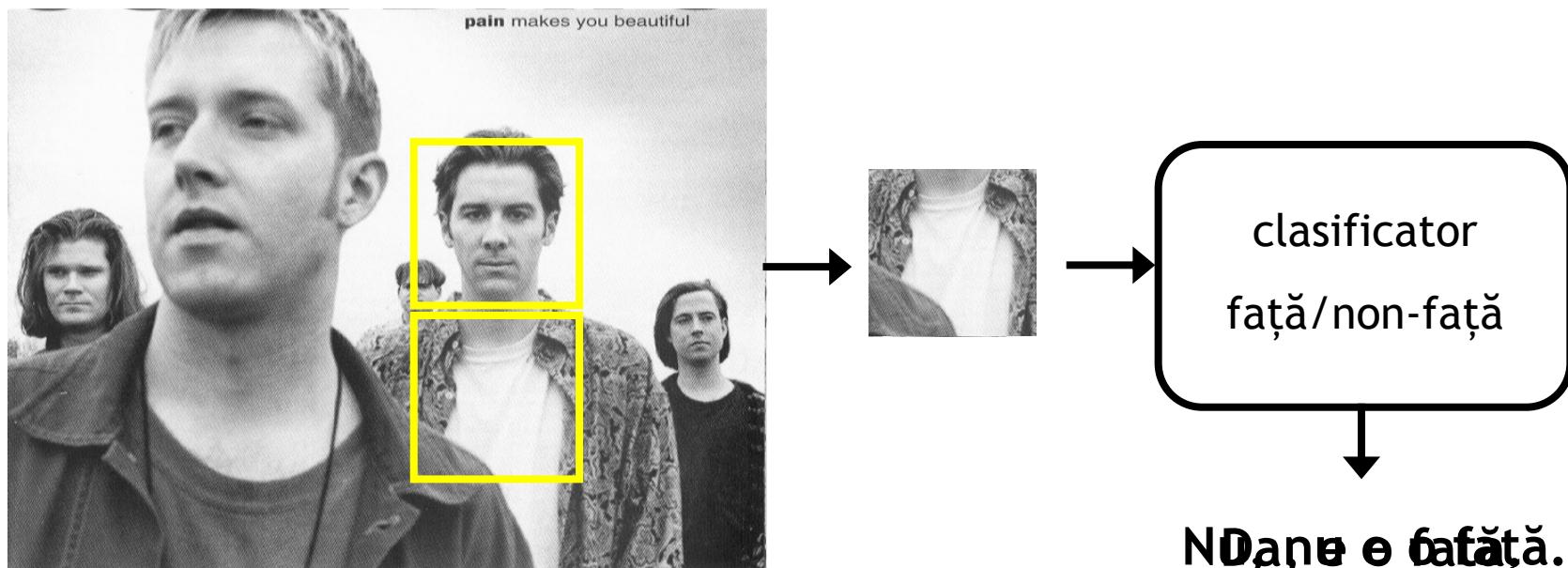
Implementarea în Python – laborator

- scriptul ‘RunProject.py’ conține întreaga implementare a proiectului
- multe funcții scrise de noi (Radu + Bogdan + Alexandra)
- funcții ce trebuie completate la laborator:
 - FacialDetector.get_positive_descriptors()
 - FacialDetector.get_negative_descriptors()
 - FacialDetector.run()
 - optional: adăugați antrenarea cu exemple puternic negative
- realizați experimente pe baza implementării

Etape în construcția detectorului facial

1. Învațarea unui clasificator

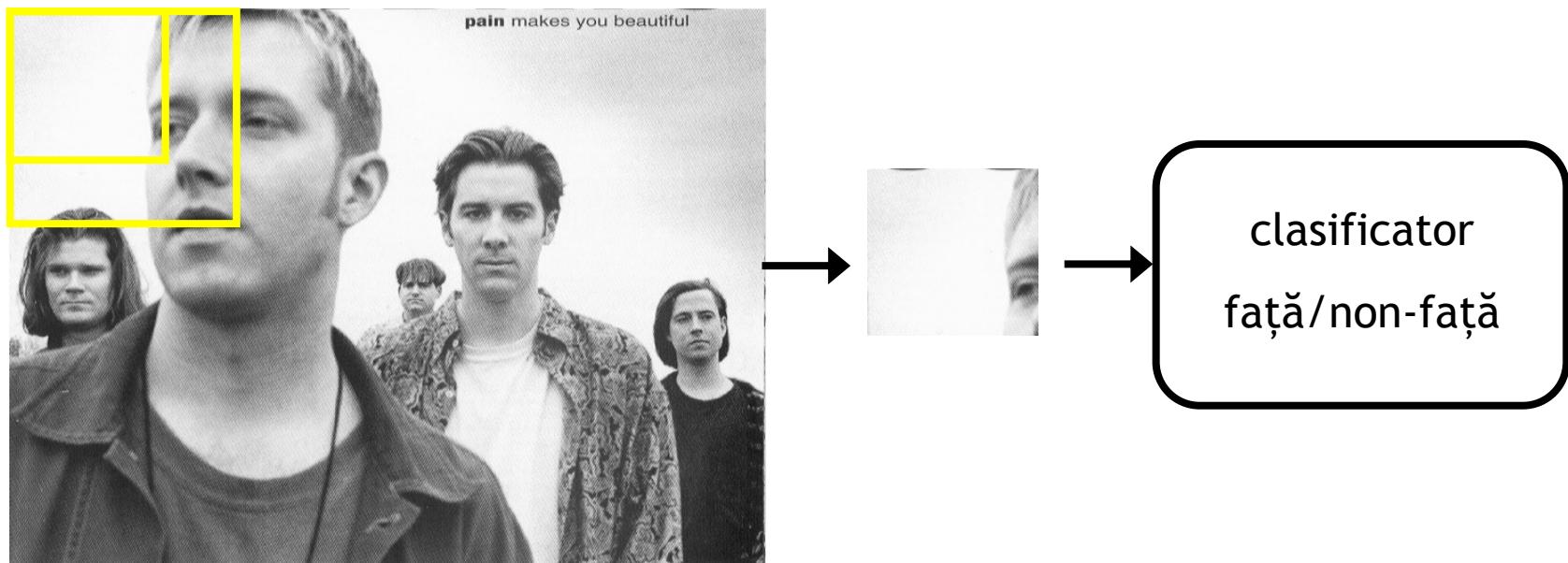
- **clasificator**: funcție care **asignează un scor** unei ferestre pe baza conținutului lor vizual (pixelii din interiorul ei)
- vrem să învățăm un clasificator care distinge ferestrele ce conțin fețe (**ideal, clasificatorul va asigna acestor ferestre un scor > 0**) de ferestrele ce nu conțin fețe (**ideal, clasificatorul va asigna acestor ferestre un scor < 0**)



Etape în construcția detectorului facial

2. Implementarea metodei glisării unei ferestre

- glisarea unei ferestre de la stânga la dreapta și de sus în jos
- clasificarea (asignarea unui scor) fiecărei ferestre pe baza clasificatorului învățat la etapa 1
- localizarea fețelor pe baza scorurilor ferestrelor



Etape în construcția detectorului facial

1. Învățarea unui clasificator

- clasificator: funcție care asignează un scor unei ferestre pe baza conținutului lor vizual (pixelii din interiorul ei)
- vrem să învățăm un clasificator care distinge ferestrele ce conțin fețe de ferestrele ce nu conțin fețe
- `FacialDetector.get_positive_descriptors()` + `FacialDetector.get_negative_descriptors()`
- optional: antrenare cu exemple puternic negative (pe baza etapei 2)

2. Implementarea metodei glisării unei ferestre

- glisarea unei ferestre de la stânga la dreapta și de sus în jos
- clasificarea (asignarea unui scor) fiecărei ferestre pe baza clasificatorului învățat la etapa 1
- localizarea fețelor pe baza scorurilor ferestrelor
- `FacialDetector.run()`

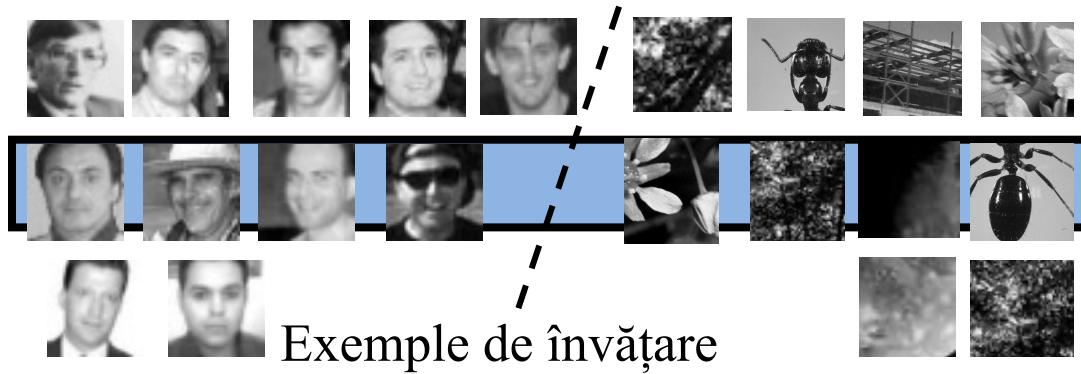
Învățarea unui clasificator

Ingrediente:

1. Multime de învățare/antrenare
 - exemple pozitive - ferestre care conțin fețe
 - exemple negative – ferestre care nu conțin fețe
2. Descrierea conținutului vizual al exemplelor de antrenare
 - descriptor pentru fiecare fereastră pe baza pixelilor din interior
 - descriptor = vector n-dimensional
3. Model de clasificator
 - cum arată funcția pe care o folosim la clasificare?
4. Învățarea parametrilor modelului
 - care sunt parametri optimi pentru care clasificatorul obține rezultate bune?

Multime de învățare/antrenare

- avem nevoie de exemple pozitive și negative pentru învățare
- o posibilitate (pentru calculul HOG): imagini în tonuri de gri (pentru imagini RGB folosiți cv.cvtColor)



- este nevoie de mii de exemple pentru a putea învăța un clasificator care surprinde variabilitatea fețelor
- o posibilitate: exemple pozitive și negative de dimensiuni 36 x 36 pixeli

Multime de învățare/antrenare

- **exemple pozitive**

- ferestre (36x36 pixeli) ce conțin fețe
- se dau 6713 imagini (36x36) cropate cu fețe
- `Parameters.number_positive_examples = 6713`
- o imagine = o fereastră = 1 exemplu pozitiv

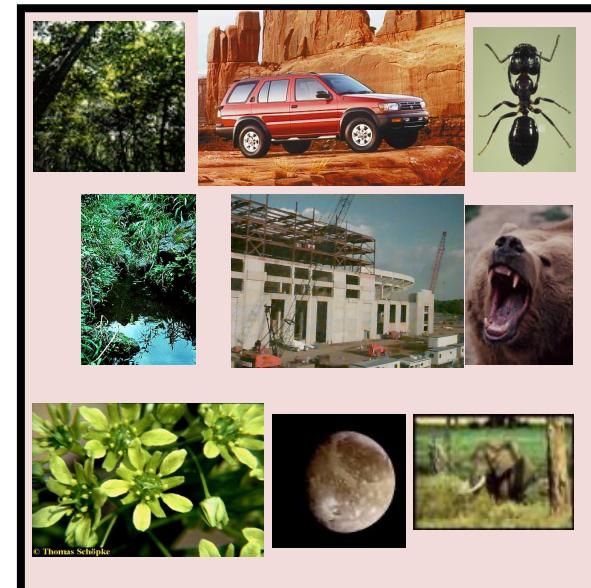


- **exemple negative**

- ferestre (36x36 pixeli) ce nu conțin fețe
- le veți obține din imagini care nu conțin fețe
- se dau 274 de imagini ce nu conțin fețe
- diverse dimensiuni ($> 36 \times 36$ pixeli)
- o imagine 100x100 pixeli = mii de ferestre

36×36 pixeli = mii de exemple negative

- `Parameters.number_negative_examples = 10000`
- $10000/274 \approx 37$ de ferestre din fiecare imagine

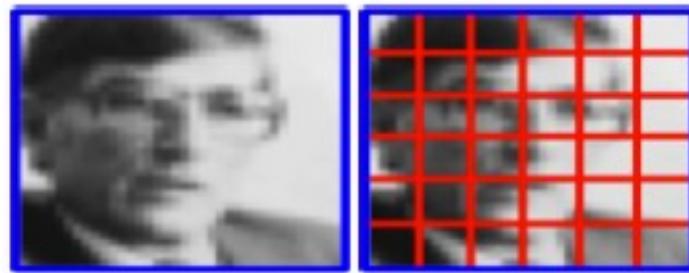


Descriptorul HOG al unei ferestre

- un exemplu de învățare = o fereastră de dimensiuni 36 x 36 pixeli
- `Parameters.dim_descriptor_cell = 36`
- descriem conținutul vizual al unei ferestre printr-un descriptor HOG
- se împarte o imagine/fereastră în celule HOG: implicit fiecare celulă are dimensiunea 6 x 6 pixeli
- `Parameters.dim_hog_cell = 6`



ZOOM

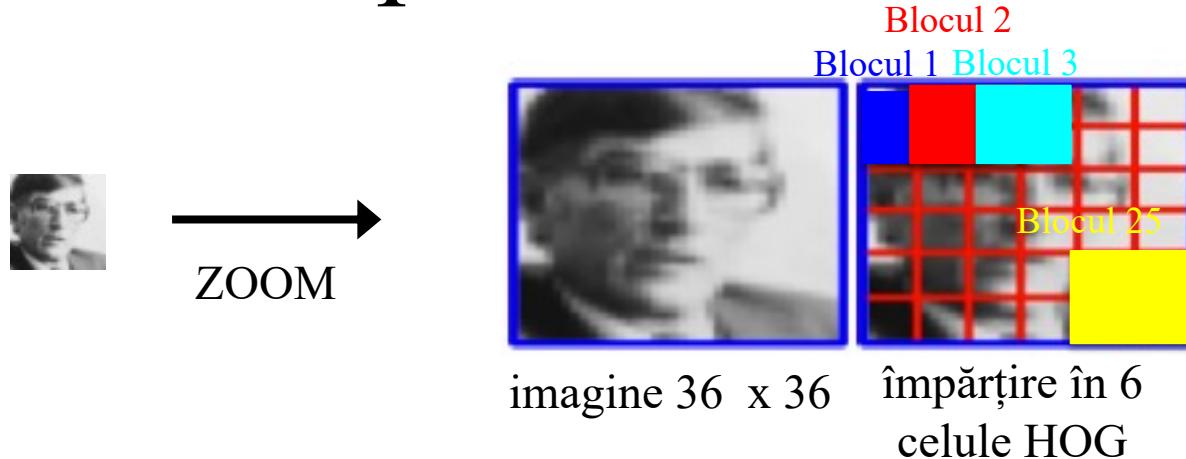


imagine 36 x 36

împărțire în 6
celule HOG

- se calculează descriptorul unei ferestre pe baza celulelor grupate în blocuri

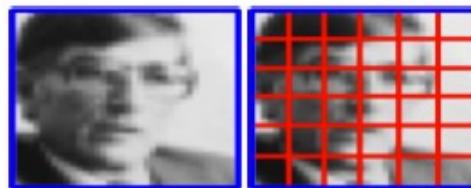
Descriptorul HOG al unei ferestre



- împarte imaginea în celule, fiecare celulă are 6×6 pixeli (`Parameters.dim_hog_cell = 6`)
- calculează gradientul imaginii (pentru fiecare pixel avem orientare și magnitudine)
- pentru fiecare celulă calculează o histogramă de gradienți orientați (9 bin-uri)
- grupează celule în blocuri, un bloc = 4 celule; fiecare celulă face parte din 4 blocuri = 4 normalizări L2;
- histograma unui bloc = 4 histograme ale celulelor = $4 \times 9 =$ vector de dimensiune 36
- HOG = descriptor obținut prin concatenarea histogramelor a 25 blocuri = $25 \times 36 = 900$

Descriptori pentru exemple de învățare

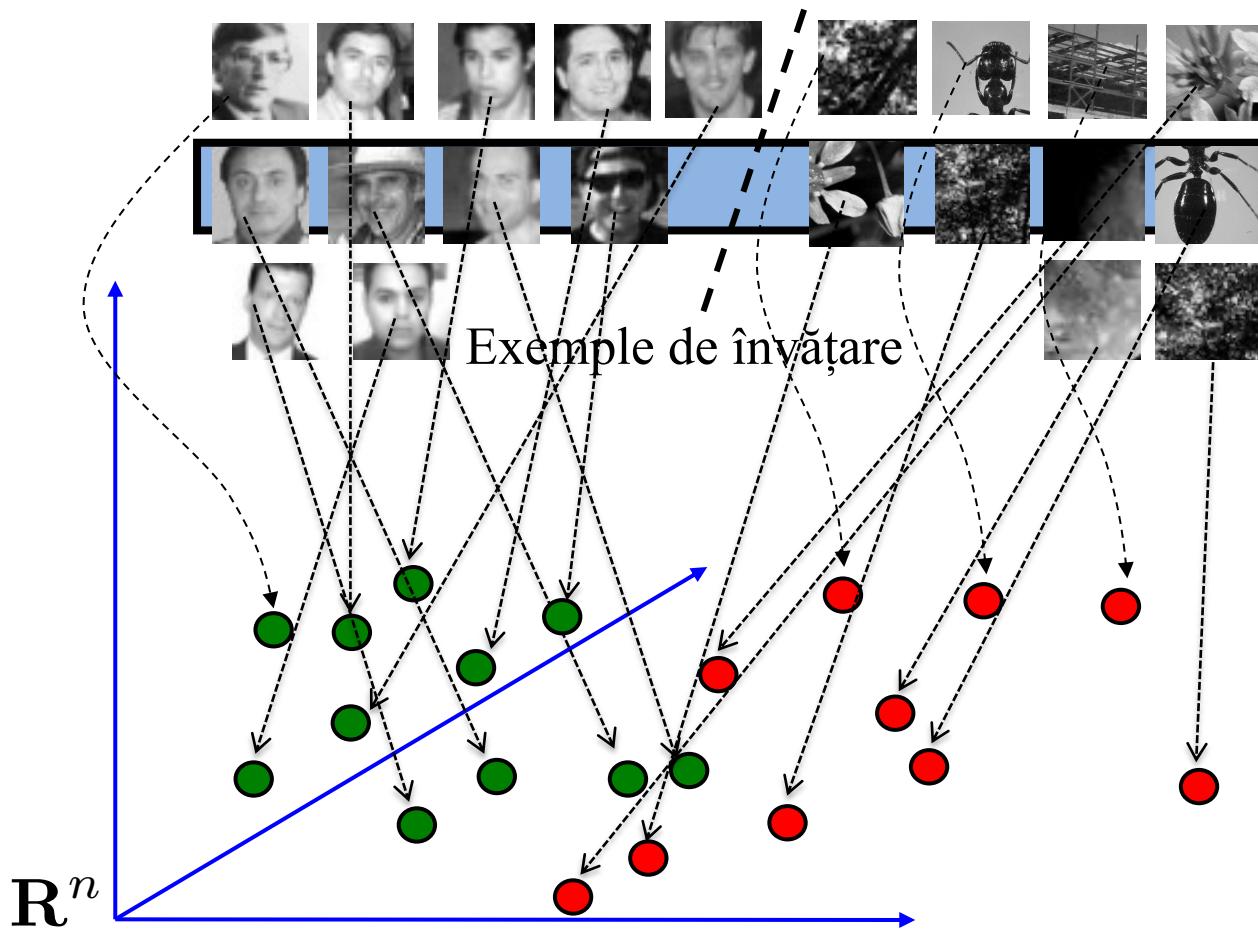
- învățarea unui clasificator se realizează pe baza descriptorilor pentru exemple pozitive și exemple negative.
- pentru obținere descriptori exemple pozitive trebuie să codați funcția `FacialDetector.get_positive_examples()` (6713 exemple pozitive x 900)
- pentru obținere descriptori exemple negative trebuie să codați funcția `FacialDetector.get_negative_examples()` (10000 exemple negative x 900)
- *împarte o imagine de orice dimensiuni în celule HOG și calculează descriptorii celulelor*



- pentru imaginile negative trebuie să construiți voi exemplele negative (extragăți ferestre 36x36 pixeli din imagini negative) pe baza celulelor HOG

Descriptori pentru exemple de învățare

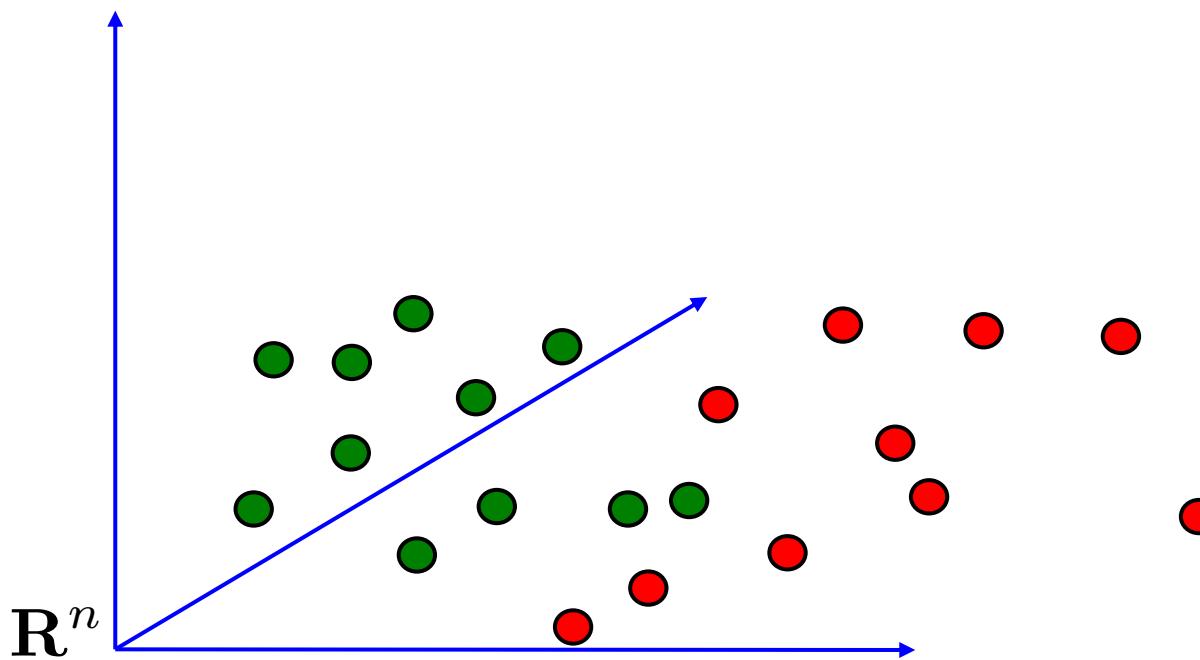
- Învățarea unui clasificator se realizează pe baza descriptorilor pentru **exemplu pozitive** și **exemplu negative**.



Spațiul descriptorilor

- descriptori pentru **exemple pozitive** și **exemple negative**
- fiecare descriptor \mathbf{d}_f este de dimensiune n

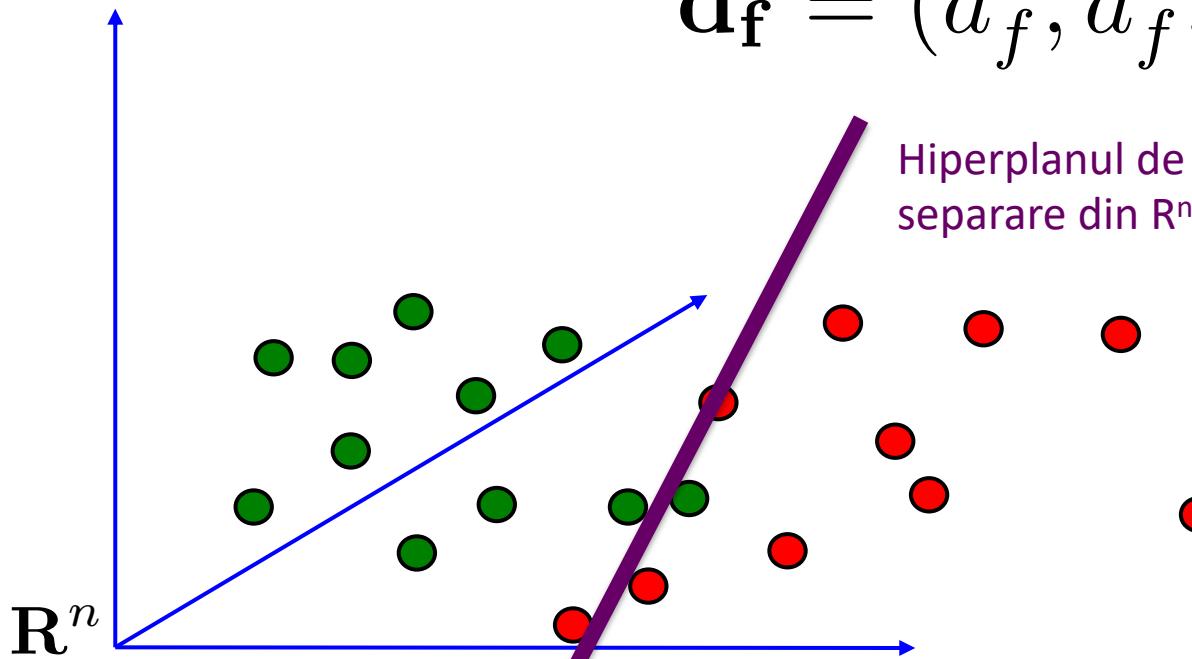
$$\mathbf{d}_f = (d_f^1, d_f^2, \dots, d_f^n)$$



Model de clasificator

- descriptori pentru **exemple pozitive** și **exemple negative**
- fiecare descriptor \mathbf{d}_f este de dimensiune n
- un posibil clasificator: clasificator liniar (hiperplan) care să separe cât mai bine **exemplile pozitive** de **exemplile negative**
- ecuația unui hiperplan în R^n : $\mathbf{w}^t \cdot \mathbf{d}_f + b = 0$

$$\mathbf{d}_f = (d_f^1, d_f^2, \dots, d_f^n)$$



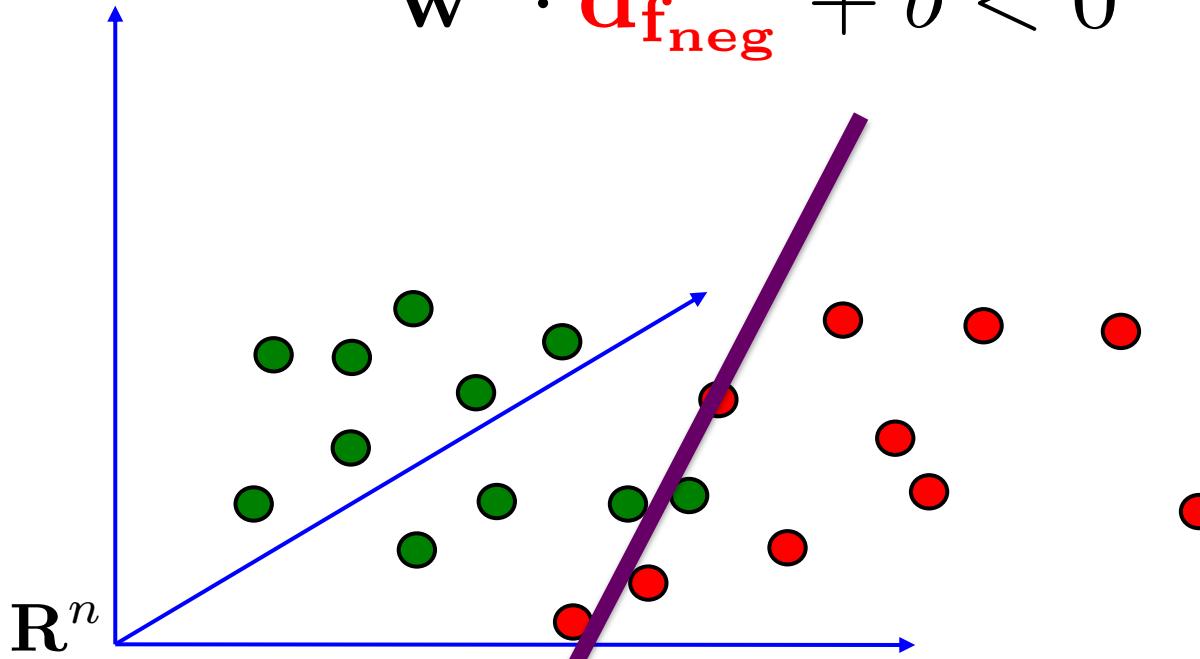
Model de clasificator

- regula de clasificare:
 - pentru **exemple pozitive** vrem să avem:

$$\mathbf{w}^t \cdot \mathbf{d}_{f_{\text{pos}}} + b > 0$$

- pentru exemple negative vrem să avem:

$$\mathbf{w}^t \cdot \mathbf{d}_{f_{\text{neg}}} + b < 0$$



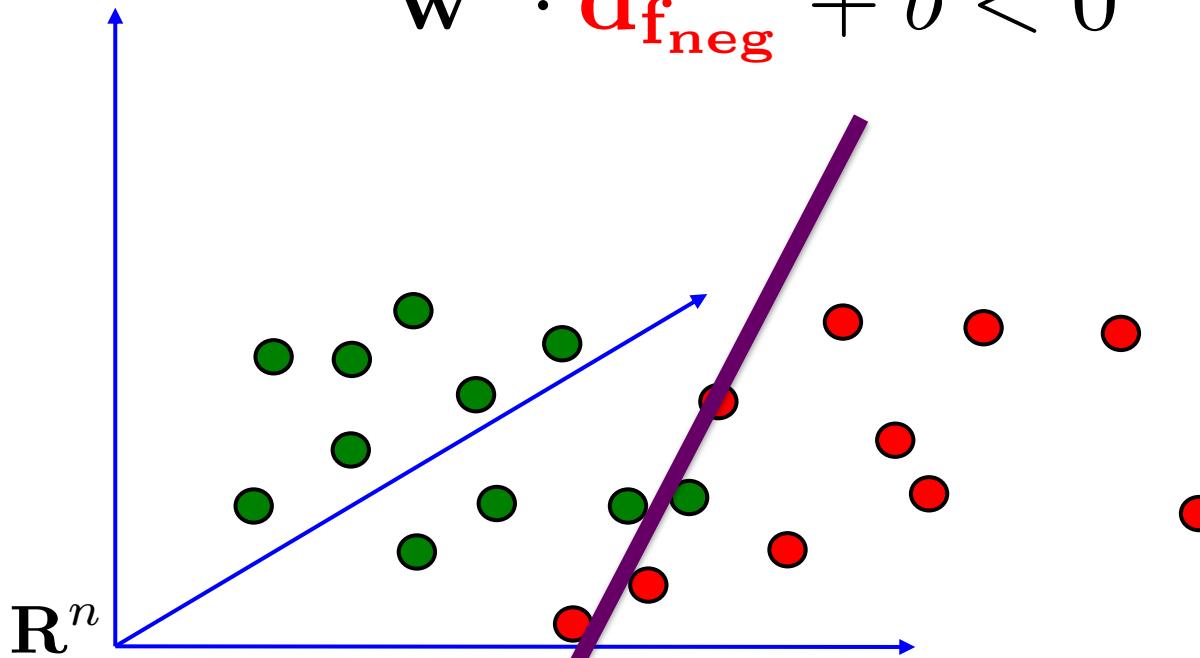
Învățarea parametrilor modelului

- învăță parametri (w, b) ai clasificatorului liniar (= hiperplan) astfel încât
 - pentru **exemple pozitive** vrem să avem:

$$\mathbf{w}^t \cdot \mathbf{d}_{\mathbf{f}_{\text{pos}}} + b > 0$$

- pentru exemple negative vrem să avem:

$$\mathbf{w}^t \cdot \mathbf{d}_{\mathbf{f}_{\text{neg}}} + b < 0$$



Vizualizare performanță clasificator

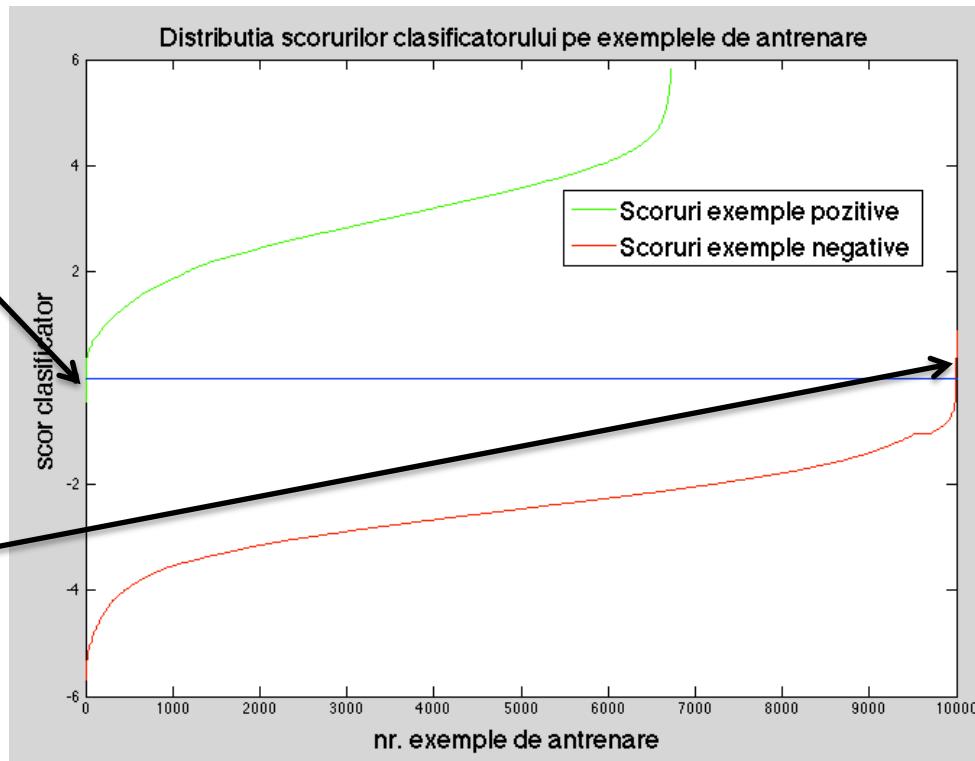
- învață parametrii (w, b) ai clasificatorului liniar (= hiperplan) astfel încât

$$\mathbf{w}^t \cdot \mathbf{d}_{f_{\text{pos}}} + b > 0$$

$$\mathbf{w}^t \cdot \mathbf{d}_{f_{\text{neg}}} + b < 0$$

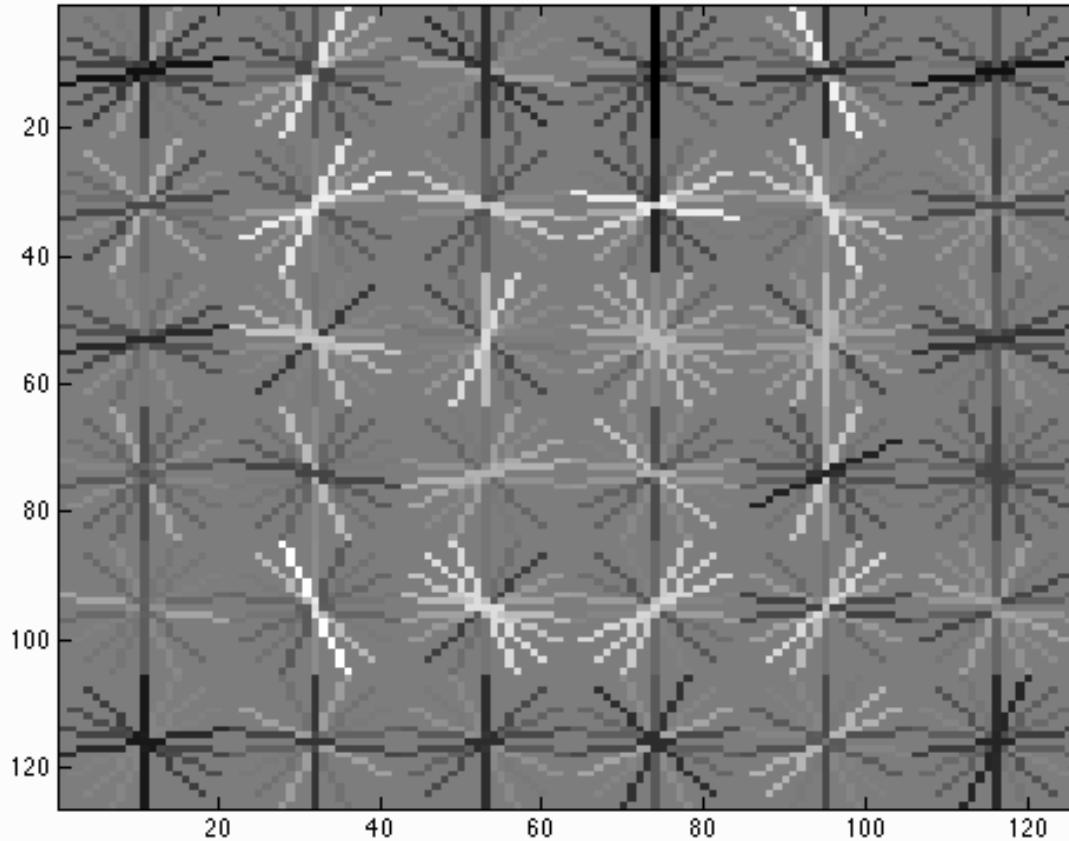
scoruri < 0
pentru exemple pozitive

scoruri > 0
pentru exemple negative



- aproape toate exemplele pozitive au un scor > 0
- aproape toate exemplele negative au un scor < 0

Vizualizare template HOG învățat



Implementarea metodei glisării unei ferestre

1. Pentru o imagine test folosiți o funcție pentru a crea celulele și a calcula descriptorii HOG asociați celulelor



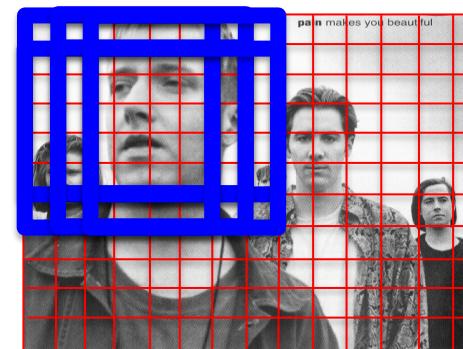
Implementarea metodei glisării unei ferestre

1. Pentru o imagine test calculați descriptorii HOG asociati celulelor
2. Evaluati toate ferestrele patratice f de dimensiune 36x36 pixeli
 - evaluarea clasificatorului: $w^t d_f + b$
3. Repetați 1+2 pentru mărimi diferite ale imaginii
 - localizarea fețelor din imagine de mărimi diferite



Implementarea metodei glisării unei ferestre

1. Pentru o imagine test calculați descriptorii HOG asociati celulelor
2. Evaluati toate ferestrele patratice f de dimensiune 36x36 pixeli
 - evaluarea clasificatorului: $w^t d_f + b$
3. Repetați 1+2 pentru mărimi diferite ale imaginii
 - localizarea fețelor din imagine de mărimi diferite



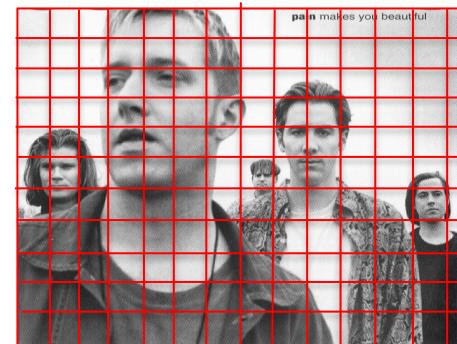
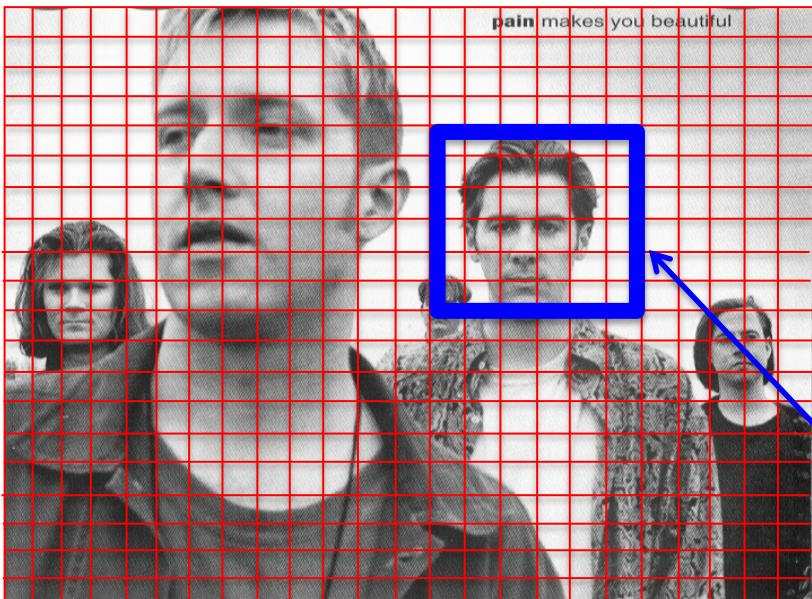
Implementarea metodei glisării unei ferestre

1. Pentru o imagine test calculați descriptorii HOG asociati celulelor
2. Evaluati toate ferestrele patratice f de dimensiune 36x36 pixeli
 - evaluarea clasificatorului: $w^t d_f + b$
3. Repetați 1+2 pentru mărimi diferite ale imaginii
 - localizarea fețelor din imagine de mărimi diferite
4. Maximele locale ale clasificatorului localizează fețele în imagine



Implementarea metodei glisării unei ferestre

1. Pentru o imagine test calculați descriptorii HOG asociati celulelor
2. Evaluati toate ferestrele patratice f de dimensiune 36x36 pixeli
 - evaluarea clasificatorului: $w^t d_f + b$
3. Repetați 1+2 pentru mărimi diferite ale imaginii
 - localizarea fețelor din imagine de mărimi diferite
4. Maximele locale ale clasificatorului localizează fețele în imagine



Maxim local
(fereastra cu scorul cel
mai mare)

Implementarea metodei glisării unei ferestre

1. Pentru o imagine test folosiți o funcția pentru a crea celulele și a calcula descriptorii HOG asociați celulelor
2. Evaluati toate ferestrele pătratice f de dimensiune 36x36 pixeli
 - evaluarea clasificatorului: $w^t d_f + b$
3. Repetați 1+2 pentru mărimi diferite ale imaginii
 - localizarea fețelor din imagine de mărimi diferite
4. Maximele locale ale clasificatorului localizează fețele în imagine

Codați toți acești pași în funcția `FacialDetector.run()`

La pasul 4 folosiți funcția `non_maximum_suppression` pentru a putea obține maximele locale

Puteți elimina foarte multe ferestre aplicând un threshold (implicit `Parameters.threshold = 0`): toate ferestrele cu scorul < threshold să fie eliminate (eliminați toate ferestrele cu scor mic).

Optional: antrenare cu exemple puternic negative

1. Rulați funcția `FacialDetector.run()` pe imaginile negative de antrenare (=274).
 - în aceste imagini toate ferestrele ar trebui să aibă un **scor < 0**
 - **nu există nici o față în aceste imagini**
2. Toate ferestrele din imagini negative pe care le obțineți cu **scor > 0** le adăugați exemplelor negative (implicit =10000).
 - trebuie să modificați funcția `FacialDetector.run()` pentru a returna și descriptorul (exemplul de antrenare)
3. Reantrenați clasificatorul cu noile exemple negative adăugate de la 2

Evaluare – detectii adevărate și false

Cum evaluăm performanța unui detector facial?

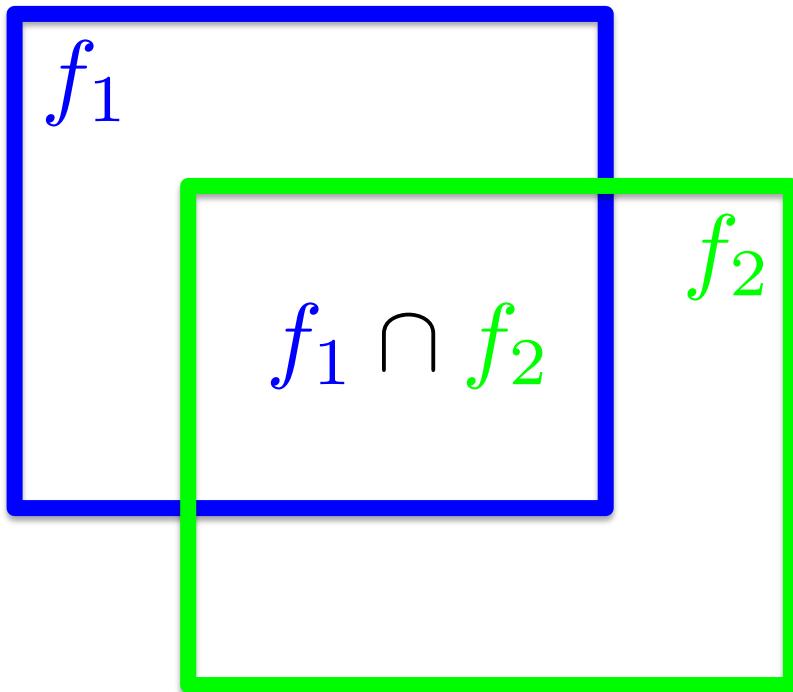
- comparăm ceea ce am obținut (ferestrele de maxim local se numesc detectii) cu ceea ce trebuie să obținem (ferestre adnotate conținând fețe)
- detectii adevărate** = detectii care se suprapun (intersecție/reuniune) > 0.3 cu o fereastră adnotată
- detectii false** = detectii care nu se suprapun > 0.3 cu o fereastră adnotată sau există o altă detectie de scor mai mare care se suprapune > 0.3 cu fereastra adnotată (se penalizează multiple detectii ale aceleiași fețe)

Imaginea: "class57.jpg" (verde=detectie adevarata, rosu=detectie falsa, galben=ground-truth adnotat), 55/57 gasite



Ferestrele adnotate
Detectiile adevărate
Detectiile false

Suprapunerea a două ferestre



$$suprapunere(f_1, f_2) = \frac{aria(f_1 \cap f_2)}{aria(f_1 \cup f_2)}$$

$$suprapunere(f_1, f_2) = \frac{aria(f_1 \cap f_2)}{aria(f_1) + aria(f_2) - aria(f_1 \cap f_2)}$$

Evaluare – recall și precizie pe o imagine

Cum evaluăm performanța unui detector facial?

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

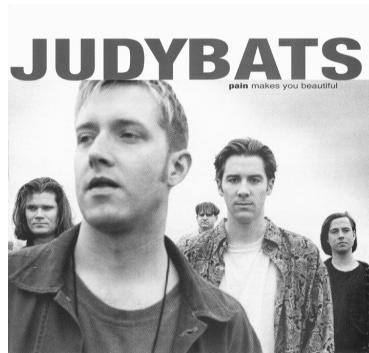
Imaginea: "class57.jpg" (verde=detectie adevarata, rosu=detectie falsa, galben=ground-truth adnotat), 55/57 gasite



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)



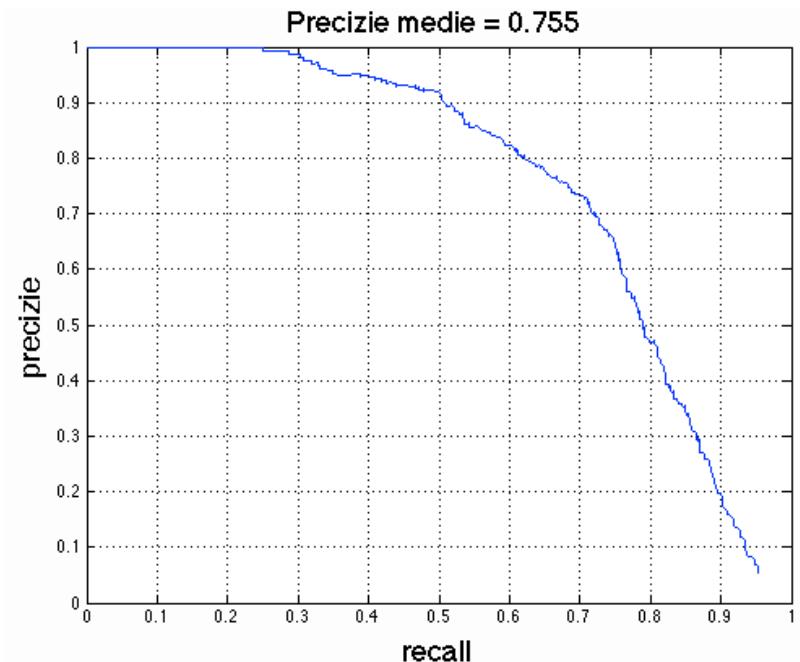
- se compară în funcție de precizie și recall sumarizate în graficul precizie-recall care oferă un număr: precizia medie a detectorului de fețe

Evaluare – recall și precizie pe toate imaginile

Se construiește graficul precizie-recall în felul următor:

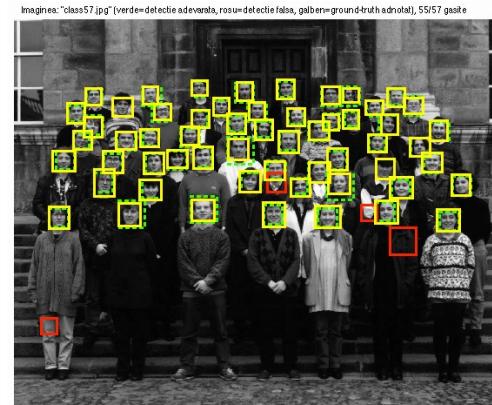
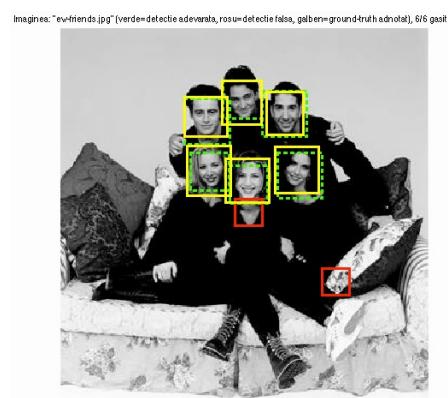
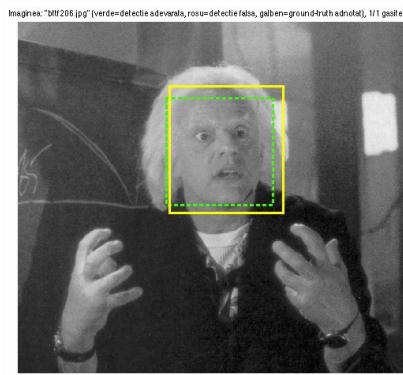
1. se rulează detectorul facial pe toate imaginile și se obține pentru fiecare imagine o mulțime de detectii;
2. se stabilește pentru fiecare imagine (pentru care avem adnotări) care sunt detectiile adevărate și cele false;
3. se ordonează descrescător toate detectiile în funcție de scorul lor;
4. pentru 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 $> \text{threshold}$

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

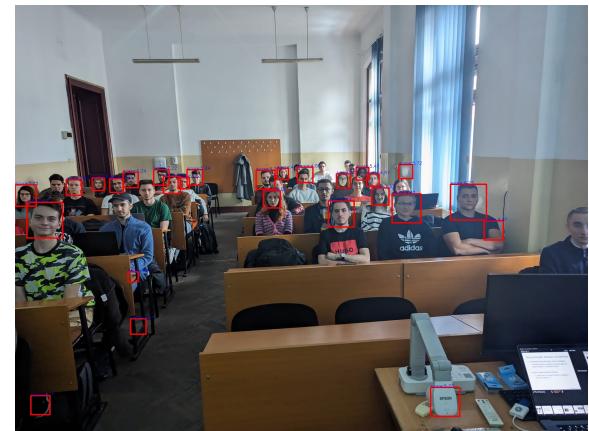
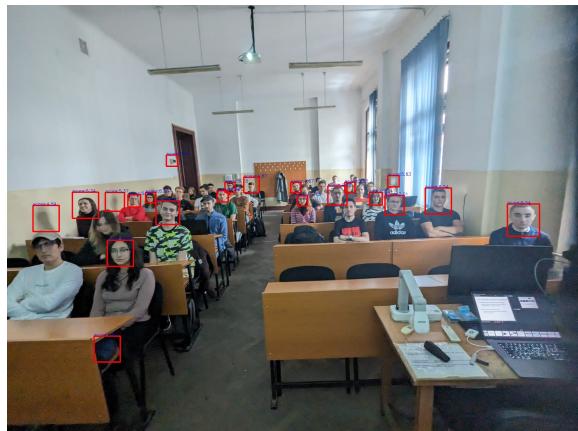


Vizualizare rezultate

1. Pentru seturi de date cu adnotări (CMU+MIT) folosiți funcția `show_detections_with_ground_truth` (Parameters.has_annotations = True)



2. Pentru seturi de date fără adnotări (imaginele de la curs) folosiți funcția `show_detections_without_ground_truth` (Parameters.has_annotations = False)



Experimente

Realizați experimente (vreți să obțineți o valoare precizie medie cât mai bună) după ce ați codat cele 3 funcții:

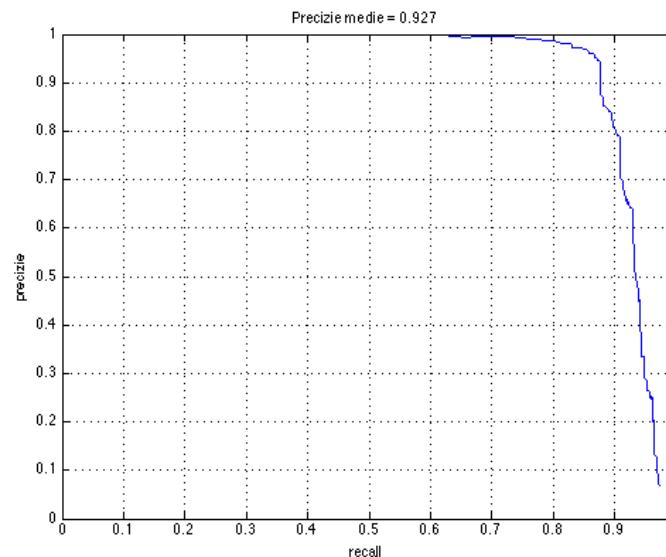
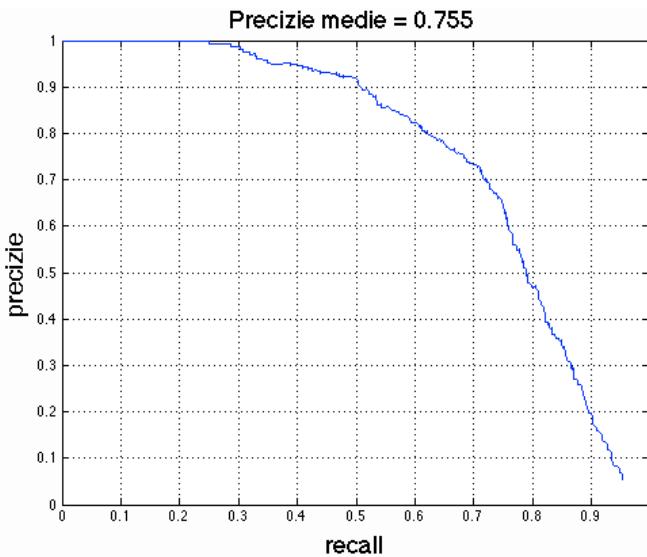
- FacialDetector.get_positive_examples()
- FacialDetector.get_negative_examples()
- FacialDetector.run()
- optional: adăugați antrenarea cu exemple puternic negative

Parametri:

- dim_window = dimensiuneFereastra = 36 (fix)
- dim_hog_cell = dimensiuneCelulaHOG = 6 (implicit) – încercați alte valori
- dim_descriptor_cell = dimensiuneDescriptorCelula = 36 (fix)
- number_positive_examples = numarExemplePozitive = 6713 (implicit) – creșteți numărul exemplelor
- number_positive_examples numarExempleNegative = 10000 (implicit) – încercați alte valori
- threshold = 0(implicit) – încercați alte valori (valori mari: recall mic, precizie mare; valori mici: recall mare, precizie mică) .threshold

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



- algoritmul din dreapta este mai bun decât algoritmul din stânga