

## Inteligență Artificială - Proiect 3

### Clasificarea imaginilor folosind modelul Bag-Of-Visual-Words

#### Obiectiv:

Tema acestui proiect este clasificarea imaginilor folosind modelul *Bag-Of-Visual-Words* (*sac de cuvinte vizuale*).

Funcțiile Matlab care vă vor ajuta la implementarea proiectului sunt în directorul 'cod'; datele de antrenare și testare sunt în directorul 'data' (Figura 1). Toate acestea le găsiți în materialele din arhivă.

Pentru acest proiect puteți forma grupuri de lucru de maxim 2 studenți. Toate detaliile legate de predarea proiectului le găsiți la sfârșitul acestui document.

**Introducere.** Recunoașterea obiectelor în imagini este una dintre probleme fundamentale în Vederea Artificială. Una din abordările cu cel mai mare succes în rezolvarea acestei probleme o constituie folosirea modelului *Bag-Of-Visual-Words* (prescurtat în continuare BOVW). Acest model reprezintă imaginile sub formă de histograme de cuvinte vizuale. În cadrul acestui proiect veți implementa un clasificator de imagini pe baza modelului BOVW



Figura 1: **Rândul de sus:** imagini de antrenare. Primele trei imagini sunt exemple de antrenare pozitive (imagini ce conțin mașini), următoarele trei imagini sunt exemple de antrenare negative (imagini ce nu conțin mașini). **Rândul de jos:** imagini de testare. Primele trei imagini ar trebui clasificate ca fiind pozitive, următoarele trei ca fiind negative.

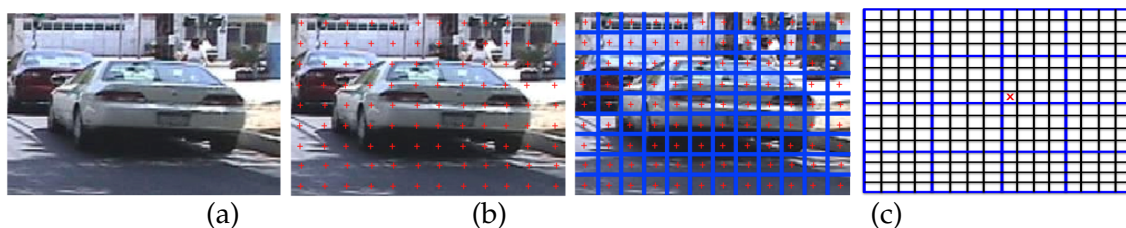


Figura 2: **Extragere de caracteristici:** a) imaginea input; b) puncte generate pe un caroiaj; c) pentru fiecare punct considerăm patch-ul (fereastră de culoare albastru) centrat în acest punct și de dimensiune  $16 \times 16$  pixeli. În funcție de dimensiunile imaginii, unele patch-uri se pot suprapune; d) vizualizare a unui patch împărțit în  $4 \times 4$  celule, fiecare conținând  $4 \times 4$  pixeli. Punctul roșu reprezintă centrul patch-ului.

care decide dacă o imagine test conține o mașină (vedere din spate) sau nu (Figura 1). Etapele intermediare în construirea unui astfel de clasificator sunt următoarele:

- extragerea de caracteristici (generarea unor puncte în imagine pentru detectarea caracteristicilor + descrierea caracteristicilor);
- construcția vocabularului de cuvinte vizuale pe baza caracteristicilor extrase;
- reprezentarea imaginilor cu modelul BOVW;
- clasificarea imaginilor.

Funcția Matlab *ruleazaBOVW.m* conține implementarea în Matlab a întregului proiect, apelând rând pe rând funcții care realizează pașii de mai sus. Această funcție este completată în întregime. Sarcina voastră este de a completa bucățile lipsă din cod astfel încât funcția *ruleazaBOVW.m* să ruleze întreg proiectul. Pe baza acestui lucru veți putea realiza experimentele care vă vor ajuta să răspundeți întrebărilor de la final.

### 1.1 Extragerea de caracteristici (0.5 puncte)

Caracteristicile pe care le veți folosi în acest proiect sunt histograme de gradienti orientați (prescurtat în continuare HOG). Pentru fiecare imagine (de antrenare sau de test) veți extrage descriptori HOG pentru anumite puncte din imagine.

Mai întâi, generați aceste puncte în imagine. Completați funcția *genereazaPuncteCaroiaj.m*, care generează puncte într-o imagine pe baza unui caroiaj (o rețea de drepte orizontale și verticale) ce se suprapune peste imaginea input (Figura 2b), lăsând o margine de 8 pixeli față de extremitățile imaginii (stânga, dreapta, sus, jos). Parametri  $nrPuncteX = 10$  și  $nrPuncteY = 10$  definesc granularitatea caroiajului, cât de densă este rețeaua de drepte. Funcția returnează o matrice  $(nrPuncteX * nrPuncteY) \times 2$  de puncte care se regăsesc la intersecția dreptelor. Funcția care trebuie completată are forma următoare:

*puncteCaroiaj = genereazaPuncteCaroiaj(img, nrPuncteX, nrPuncteY, margine);*

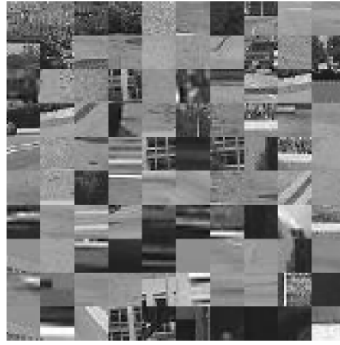


Figura 3: Vizualizare vocabular: în figură sunt afișate un număr de 100 de cuvinte vizuale.

Descrieți fiecare punct de pe caroiaj cu un descriptor local, folosind HOG. O variantă a acestui descriptor a fost predată la curs. Completați funcția *calculeazaHistogrameGradientiOrientati.m* care asociază fiecărui punct din imagine generat de funcția *genereazaPuncteCaroiaj.m* un descriptor HOG de dimensiune 128. Descriptorul se obține în felul următor:

- considerați pentru fiecare punct un patch (o fereastră din imagine) centrată în punctul respectiv și împărțit în  $4 \times 4$  celule (Figura 2c);
- fiecare celulă este pătrată, de latură dimensiune *dimensiuneCelula* (= 4 pixeli, în implementare, Figura 2c);
- pentru fiecare celulă, calculați o histogramă de dimensiune 8 după orientările gradientilor fiecărui pixel din celulă;
- concatenați cele 16 histograme ale celor 16 celule într-un singur vector de dimensiune 128.

Funcția *calculeazaHistogrameGradientiOrientati.m* returnează o matrice de dimensiuni  $N \times 128$  conținând descriptorii HOG pe linie pentru fiecare din cele  $N$  puncte dintr-o imagine. De asemenea, funcția returnează și fiecare patch asociat fiecărui punct într-o matrice  $N \times (16 * 16)$ , fiecare linie conținând un patch (luat coloană cu coloană). Această matrice este folosită la vizualizarea vocabularului (mulțimea de cuvinte vizuale, vedeți secțiunea următoare). Funcția pe care trebuie să o completați are forma următoare:

```
[descriptoriHOG, patchuri] = calculeazaHistogrameGradientiOrientati(img, puncte, dimensiuneCelula);
```

## 1.2 Construcția vocabularului (0.5 puncte)

Pentru aplicarea clasificării cu modelul BOVW veți construi un vocabular vizual. În acest sens veți clusteriza un număr mare de descriptori HOG folosind algoritmul *k-means*. Funcția *kmeans\_iter.m* este deja implementată.

Funcția *construiesteVocabular.m* implementează acest pas intermediar. Funcția primește ca input numele unui director, iar pentru fiecare imagine de antrenare din acest director calculează punctele din imagine pe baza unui caroiaj (apelează funcția *genereazaPuncteCaroiaj*),

extrage descriptorii pentru fiecare punct (apelează funcția *calculeazaHistogrameGradientiOrientati.m*). Apoi funcția clusterizează toți descriptorii extrași din imaginile de antrenare folosind algoritmul *k-means* (apelează funcția *kmeans\_iter.m*). Centri clusterilor sunt vectori de dimensiune  $1 \times 128$  și îi vom numi *cuvinte vizuale*. În total sunt 10000 de descriptori: din fiecare imagine de antrenare extragem  $10 \times 10$  puncte, caracterizăm fiecare punct cu un descriptor, există 50 de imagini de antrenare pozitive și 50 de imagini de antrenare negative.

Vizualizarea vocabularului obținut (mulțimea de cuvinte vizuale, Figura 3) se realizează apelând funcția *vizualizeazaDictionar.m*, funcție scrisă în întregime.

### 1.3 Reprezentarea imaginilor cu modelul BOVW (0.5 puncte)

Folosind vocabularul de cuvinte vizuale obținut, veți reprezenta fiecare imagine ca o histogramă de cuvinte vizuale. Această reprezentare, cunoscută și sub numele de reprezentare BOVW, înregistrează ce cuvinte vizuale apar în imagine și de asemenea frecvența lor de apariție.

Completați codul funcției *calculeazaHistogramaBOVW.m* care calculează histograma BOVW asociată fiecărei imagini. Funcția primește ca input mulțimea de descriptori HOG extrași din imagine și vocabularul de cuvinte vizuale. Pentru obținerea histogramei imaginii, calculați pentru fiecare descriptor HOG în parte cuvântul vizual cel mai apropiat (în sensul distanței Euclidiene) și numărați câți descriptori sunt asignați fiecărui cuvânt vizual. Funcția pe care trebuie să o completați are forma următoare:

*histogramaBOVW = calculeazaHistogramaBOVW(descriptoriHOG, cuvinteVizuale)*

Completați codul funcției *calculeazaHistogramaBOVW\_director.m* care citește toate imaginile dintr-un director și calculează histograma BOVW asociată fiecărei imagini. Output-ul acestei funcții este o matrice cu numărul de linii egal cu numărul de imagini și numărul de coloane egal cu numărul de cuvinte vizuale. Fiecare linie reprezintă histograma BOVW a unei imagini. Funcția pe care trebuie să o completați are forma următoare:

*histogrameBOVW = calculeazaHistogrameBOVW\_director(numDirector, cuvinteVizuale)*

Apelând această funcție procesăm exemplele pozitive (imagini ce conțin mașini) și exemplele negative (imagini ce nu conțin mașini) din directoarele *masini-exempleAntrenare-pozitive* și *masini-exempleAntrenare-negative*. Colectăm histogramele rezultante în matricele *histogrameBOVW\_exemplePozitive* și *histogrameBOVW\_exempleNegative*.

### 1.4 Clasificarea imaginilor (1.5 puncte)

Funcția *clasificaBOVW.m* (funcția este scrisă în întregime) implementează etapa de clasificare pe baza unui clasificator. Sarcina voastră este de a scrie funcțiile care implementează clasificatorii. În acest proiect veți experimenta trei clasificatori: clasificatorul bazat pe cel

mai apropiat vecin, clasificatorul Bayes și clasificatorul liniar antrenat pe baza unui SVM (detalii la curs).

#### a. Clasificatorul cel mai apropiat vecin

Realizați mai întâi clasificarea folosind principiul celui mai apropiat vecin. Pentru clasificarea unei imagini test, calculați histograma de cuvinte vizuale și asigurați eticheta (pozitivă sau negativă) corespunzătoare celui mai apropiat vecin.

Completați funcția *clasificaBOVWCelMaiApropiatVecin.m* care compară histograma BOVW a unei imagini test cu histogramele exemplarelor pozitive și exemplarelor negative calculate anterior și returnează eticheta corespunzătoare celui mai apropiat vecin (eticheta 1 dacă cel mai apropiat vecin este un exemplu pozitiv, 0 altfel). Funcția pe care trebuie să o completați are forma următoare:

*eticheta* = *clasificaBOVWCelMaiApropiatVecin(histogramaBOVW\_test, histogrameBOVW\_pozitive, histogrameBOVW\_negative);*

#### b. Clasificatorul Bayes

Implementați în partea a doua clasificatorul Bayes. Fie  $P(Masina|hist)$  probabilitatea ca o imagine test să conțină o mașină pe baza observației că histograma sa BOVW este *hist*. A-priori, probabilitatea ca o imagine să conțină o mașină este  $P(Masina)$ , iar probabilitatea ca histograma BOVW a unei imagini să fie *hist* este  $P(hist)$ . Conform teoremei lui Bayes avem:

$$P(Masina|hist) = \frac{P(hist|Masina) \cdot P(Masina)}{P(hist)}.$$

Numitorul poate fi rescris astfel:

$$P(hist) = P(hist|Masina) \cdot P(Masina) + P(hist|NonMasina) \cdot P(NonMasina),$$

unde  $P(NonMasina)$  este probabilitatea ca o imagine test să nu conțină o mașină. Evident,  $P(Masina) + P(NonMasina) = 1$ .

Înlocuind în prima ecuație, obținem probabilitatea a-posteriori ca o imagine test să conțină o mașină pe baza observației histogramei sale BOVW *hist*:

$$P(Masina|hist) = \frac{P(hist|Masina) \cdot P(Masina)}{P(hist|Masina) \cdot P(Masina) + P(hist|NonMasina) \cdot P(NonMasina)}$$

Clasificarea unei imagini test implică calculul probabilităților  $P(Masina|hist)$  și  $P(NonMasina|hist)$ . Cum  $P(NonMasina|hist) = 1 - P(Masina|hist)$ , este suficient pentru clasificare să evaluăm numai  $P(Masina|hist)$ . Pentru a calcula această probabilitate a-posteriori trebuie să calculăm probabilitățile condiționate  $P(hist|Masina)$  și  $P(hist|NonMasina)$  care apar la numărător și numitor. Probabilitatea a-priori  $P(Masina)$  o considerăm = 0.5. Calculăm cele două probabilități condiționate astfel:

- matricele *histogrameBOVW\_exemplePozitive* și *histogrameBOVW\_exempleNegative* conțin pe fiecare linie o histogramă a unei imagini de antrenare (pozitivă sau negativă). Fiecare coloană  $i$  înregistrează de câte ori cuvântul vizual  $i$  apare în imaginea respectivă.
- fie  $X(i)$  variabila aleatoare care numără de câte ori apare cuvântul vizual  $i$  în imaginile pozitive (matricea *histogrameBOVW\_exemplePozitive*). Presupunem că  $X(i)$  are o distribuție normală  $\mathcal{N}(\mu_{poz}(i), \sigma_{poz}(i))$ , unde  $\mu_{poz}(i)$  și  $\sigma_{poz}(i)$  reprezintă media și deviația standard a variabilei aleatoare  $X(i)$ . Acești 2 parametri pot fi estimați din matricea *histogrameBOVW\_exemplePozitive* considerând coloana  $i$ . Completați funcția *calculeazaMedieDeviatieStandardCuvinteVizuale.m* care calculează media și deviația standard pentru fiecare cuvânt vizual  $i$ . Funcția pe care trebuie să o completați are forma următoare:

`[mu, sigma] = calculeazaMedieDeviatieStandardCuvinteVizuale(histogrameBOVW);`

- în mod similar, considerăm  $Y(i)$  variabila aleatoare care numără de câte ori apare cuvântul vizual  $i$  în imaginile negative (matricea *histogrameBOVW\_exempleNegative*). Presupunem că  $Y(i)$  are o distribuție normală  $\mathcal{N}(\mu_{neg}(i), \sigma_{neg}(i))$ , unde  $\mu_{neg}(i)$  și  $\sigma_{neg}(i)$  reprezintă media și deviația standard a variabilei aleatoare  $Y(i)$ . Folosiți funcția scrisă *calculeazaMedieDeviatieStandardCuvinteVizuale.m* pentru a calcula  $\mu_{neg}(i)$  și  $\sigma_{neg}(i)$  din matricea *histogrameBOVW\_exempleNegative*.

Pentru a clasifica o imagine test cu histograma sa BOVW *hist*, trebuie să calculați probabilitatea a-posteriori  $P(Masina|hist)$ . Realizați acest lucru după cum urmează:

- fie  $Z(i)$  valoarea coloanei  $i$  din histograma test *hist*.  $Z(i)$  numără de câte ori apare cuvântul vizual  $i$  în imaginea test. Probabilitatea de a observa  $Z(i)$  într-o imagine pozitivă, conform imaginilor de antrenare pozitive, este  $P(Z(i)|\mathcal{N}(\mu_{poz}(i), \sigma_{poz}(i)))$ .
- similar,  $P(Z(i)|\mathcal{N}(\mu_{neg}(i), \sigma_{neg}(i)))$  reprezintă probabilitatea de a observa  $Z(i)$  într-o imagine negativă.
- calculați probabilitățile condiționate  $P(hist|Masina)$  și  $P(hist|NonMasina)$  folosind formulele de mai sus și utilizând funcția Matlab *normpdf* pentru calculul probabilităților de tipul  $P(Z(i)|\mathcal{N}(\mu(i), \sigma(i)))$ . Presupunând că aparițiile cuvintelor vizuale sunt independente, cele două probabilități condiționate se pot calcula astfel:

$$P(hist|Masina) = \prod_{i=1}^K P(Z(i)|\mathcal{N}(\mu_{poz}(i), \sigma_{poz}(i))),$$

$$P(hist|NonMasina) = \prod_{i=1}^K P(Z(i)|\mathcal{N}(\mu_{neg}(i), \sigma_{neg}(i))).$$

Întrucât produsul de probabilități foarte mici poate crea probleme numerice, logaritmați totul astfel încât produsul se transformă în sumă.

- presupunând că a-priori  $P(Masina) = P(NonMasina) = 0.5$ , evaluați  $P(Masina|hist)$ ;
- clasificați o imagine ca fiind pozitivă (conține o mașină) dacă  $P(Masina|hist) > 0.5$ ;
- funcția pe care trebuie să o completați are forma următoare:  

$$eticheta = \text{clasificaBOVWBayes}(\text{histogramaBOVW\_test}, \text{histogrameBOVW\_exemplePozitive}, \text{histogrameBOVW\_exempleNegative})$$

### c. Clasificatorul SVM liniar

Implementați în partea a treia clasificatorul bazat pe un SVM liniar folosindu-vă de implementarea oferită de biblioteca *vlfeat* disponibilă online la <http://www.vlfeat.org/>. În particular, veți folosi pentru antrenarea unui SVM liniar care să distingă exemplele pozitive de cele negative funcția *vl\_svmtrain* (aveți detalii despre ea aici [http://www.vlfeat.org/matlab/vl\\_svmtrain.html](http://www.vlfeat.org/matlab/vl_svmtrain.html)). Fișierul MEX *vl\_svmtrain* din directorul 'cod' merge pe calculatoare ce au ca sistem de operare Windows pe 64 de biți. Pentru alte sisteme de operare folosiți fișierele MEX corespunzătoare din directorul 'cod/mexfilesWindows+Mac+Linux/'.

- funcția pe care trebuie să o completați are forma următoare:  

$$eticheta = \text{clasificaBOVWLinierSVM}(\text{histogramaBOVW\_test}, \text{histogrameBOVW\_exemplePozitive}, \text{histogrameBOVW\_exempleNegative}).$$

## 1.5 Predarea proiectului

Puneți într-o arhivă cu numele **Nume\_PrenumeStudenti\_proiect3\_cod.zip** codul vostru Matlab (fără datele de antrenare și datele de testare).

Puneți într-un document cu numele **Nume\_PrenumeStudenti\_proiect3\_raport.pdf** răspunsurile voastre la următoarele cerințe:

- (3 puncte) plotați performanța clasificării (procentul de imagini clasificate corect) pentru cei trei clasificatori (cel mai apropiat vecin, clasificatorul Bayes, clasificatorul liniar) în funcție de numărul de cuvinte vizuale  $k = [5 \ 10 \ 25 \ 50 \ 100 \ 200 \ 500 \ 1000]$ . Care este cel mai bun clasificator din punct de vedere al performanței de clasificare? Care este cel mai rapid clasificator din punct de vedere al timpului de rulare? Justificați răspunsurile.
- (1 punct) pe baza graficului de la punctul anterior determinați valoarea  $k$  optimă pentru fiecare clasificator care conduce la o performanță optimă a clasificării. Determinarea valorii  $k$  optime se realizează pe exemplele de testare. Acest lucru ar trebui să se realizeze totuși pe mulțimea de antrenare. Care ar fi o soluție în acest sens?
- (1 punct) modificați implementarea pentru clasificatorul cel mai apropiat vecin astfel încât clasificarea să se realizeze în funcție de cei mai apropiați 5 vecini. Cum se modifică performanța clasificatorului în acest sens? Găsiți exemple de imagini test care sunt clasificate gresit. Cum va explicați clasificarea gresită? Justificați.

- (d) **(1 punct)** rulând funcția *ruleazaBOVW.m* de mai multe ori cu aceeași valoare  $k$ , veți observa că performanța clasificatorilor variază. Cum explicați acest lucru? Cum puteți caracteriza în acest caz performanța clasificatorilor?

Din oficiu: **1 punct.**

Trimiteți cele două fișiere (**Nume\_PrenumeStudenti\_proiect3\_cod.zip** și **Nume\_PrenumeStudenti\_proiect3\_raport.pdf**) la adresele de email ale supervizorilor vosri (**bogdan.alexe@fmi.unibuc.ro** sau **sergiu.nisioi@gmail.com** precizând numele studenților care au lucrat la proiect si grupa din care acestia fac parte.

Termenul limită de predare a proiectului este duminică, 21 ianuarie 2018, ora 23:59. Fiecare zi de întârziere în predarea proiectului se penalizează cu 10% din valoarea proiectului.