

Vedere Artificială - Tema 4

Clasificarea imaginilor folosind modelul Bag-Of-Visual-Words

Obiectiv:

În această temă veți realiza clasificarea imaginilor folosind modelul *Bag-Of-Visual-Words*.

Funcțiile Matlab care vă vor ajuta la implementarea temei sunt în directorul 'cod'; datele de antrenare și testare sunt în directorul 'data' (Figura 1). 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 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);



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 caroi; c) pentru fiecare punct considerăm patch-ul (fereastră de culoare albastru) centrat în acest punct și de dimensiune 16×16 pixeli. În funcție de dimensiunile imaginii, unele patch-uri se pot suprapune; d) vizualizare a unui patch împărțit în 4×4 celule, fiecare conținând 4×4 pixeli. Punctul roșu reprezintă centrul patch-ului.

- 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 întregii teme, 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 întreaga tema. Pe baza acestui lucru veți putea realiza experimentele care vă vor ajuta să răspundeți întrebărilor de la final.

4.1 Extragerea de caracteristici (2 puncte)

Caracteristicile pe care le veți folosi în acest proiect sunt histogramme 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 *genereazaPuncteCaroi*.m, care generează puncte într-o imagine pe baza unui caroi (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 caroiului, 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:

```
puncteCaroi = genereazaPuncteCaroi(img, nrPuncteX, nrPuncteY, margine);
```

Descrieți fiecare punct de pe caroi cu un descriptor local, folosind HOG. Există mai multe variante de implementare pentru HOG, în această temă veți implementa o variantă aleasă de noi. Completați funcția *calculeazaHistogrameGradientiOrientati.m* care asociază fiecărui punct din imagine generat de funcția *genereazaPuncteCaroi*.m un descriptor HOG de dimensiune 128. Descriptorul se obține în felul următor:

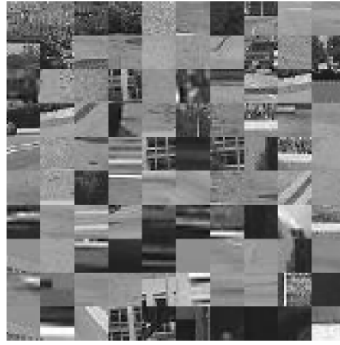


Figura 3: Vizualizare vocabular: în figură sunt afișate un număr de 100 de cuvinte vizuale.

- considerați pentru fiecare punct un patch (o fereastră din imagine) centrat în punctul respectiv și împărțit în 4×4 celule (Figura 2c);
- fiecare celulă este pătrată, de latură dimensiune Celula (= 4 pixeli, în implementare, Figura 2d);
- 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, dimCelula);

4.2 Construcția vocabularului (2 puncte)

Pentru aplicarea clasificării cu modelul BOVW veți construi un vocabular vizual. În acest sens veți clusteriza un număr mare de descriptorii 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 1×128 și îi vom numi *cuvinte vizuale*. În total sunt 10000 de descriptorii: din fiecare imagine de antrenare extragem $10 * 10$ 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.

4.3 Reprezentarea imaginilor cu modelul BOVW (2 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*.

4.4 Clasificarea imaginilor (2 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 această temă veți experimenta doi clasificatori: clasificatorul bazat pe cel mai apropiat vecin și clasificatorul liniar implementat cu un SVM (Support Vector Machine).

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_negativ);
```

b. Clasificatorul SVM liniar

Implementați în partea a doua 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). 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/'.

4.5 Predarea proiectului (2 puncte)

Puneți într-o arhivă cu numele **tema4_cod.zip** codul vostru Matlab (fără datele de antrenare și datele de testare).

Puneți într-un document cu numele **tema4_raspunsuri.pdf** răspunsurile voastre la următoarele cerințe:

- plotați performanța clasificării (procentul de imagini clasificate corect) pentru cei doi clasificatori (cel mai apropiat vecin, SVM liniar) în funcție de numărul de cuvinte vizuale $k = [5 \ 10 \ 25 \ 50 \ 100 \ 200 \ 500 \ 1000]$;
- 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 la punctul anterior 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?
- 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?

Trimiteți cele două fișiere (**tema4_cod.zip** și **tema4_raspunsuri.pdf**) la adresa de email **bo-gdan.alexe@fmi.unibuc.ro** precizând numele și grupa din care faceți parte.

Termenul limită de predare a proiectului este joi, 29 decembrie 2016, ora 23:59.