Universitatea din Bucureşti
Facultatea de Matematică şi Informatică
Departamentul de Informatică

Radu Ionescu
Bogdan Alexe
Informatică Opţional, anul 3

Vedere Artificială - Tema 4

Clasificarea imaginilor folosind modelul Bag-Of-Visual-Words

Obiectiv:

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

Funcţiile Matlab care vă vor ajuta la implementarea temei sunt în directorul 'cod'; datele de antrenare şi testare sunt în directorul 'data' (Figura 1). Toate detaliile legate de predarea proiectului le găsiţi la sfârşitul acestui document.

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

Introducere. Recunoaşterea 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 veţi implementa un clasificator de imagini pe baza modelului BOVW care decide dacă o imagine test conţine o maşină (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);

- construcţia vocabularului de cuvinte vizuale pe baza caracteristicilor extrase;
- reprezentarea imaginilor cu modelul BOVW;
- clasificarea imaginilor.

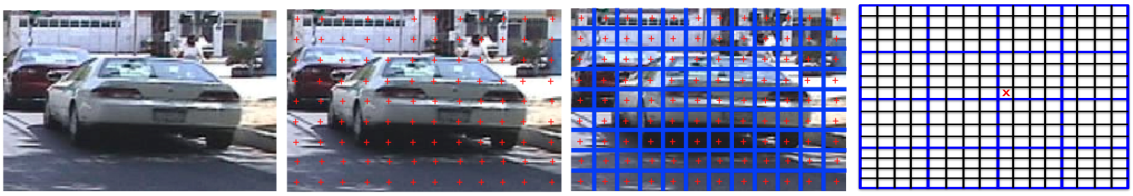
Funcţia Matlab *ruleazaBOVW.m* conţine implementarea în Matlab a întregii teme, apelând rând pe rând funcţii care realizează paşii de mai sus. Această funcţie este completată în întregime. Sarcina voastră este de a completa bucăţile lipsă din cod astfel încât funcţia *ruleazaBOVW.m* să ruleze întreaga tema. Pe baza acestui lucru veţi putea realiza experimentele care vă vor ajuta să răspundeţi întrebărilor de la final.

4.1 Extragerea de caracteristici (2 puncte)

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

Mai intâi, generaţi aceste puncte în imagine. Completaţi funcţia *genereazaPuncteCaroiaj.m*, care generează puncte într-o imagine pe baza unui caroiaj (o reţea de drepte orizontale şi verticale) ce se suprapune peste imaginea input (Figura 2b), lăsând o margine de 8 pixeli faţă de extremităţile imaginii (stânga, dreapta, sus, jos). Parametri $nrPuncteX = 10$ şi $nrPuncteY = 10$ definesc granularitatea caroiajului, cât de densă este reţeaua de drepte. Funcţia returnează o matrice $(nrPuncteX * nrPuncteY) \times 2$ de puncte care se regăsesc la intersecţia dreptelor. Funcţia care trebuie completată are forma următoare:

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



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

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 şi de dimensiune 16×16 pixeli. În funcţie de dimensiunile imaginii, unele patch-uri se pot suprapune; d) vizualizare a unui patch împărţit în 4×4 celule, fiecare conţinând 4×4 pixeli. Punctul roşu reprezintă centrul patch-ului.

Descrieţi fiecare punct de pe caroiaj cu un descriptor local, folosind HOG. Există mai multe variante de implementare pentru HOG, în această temă veţi implementa o variantă aleasă de noi. Completati funcţia *calculeazaHistogrameGradientiOrientati.m* care asociază fiecărui punct din imagine generat de funcţia *genereazaPuncteCaroiaj.m* un descriptor HOG de dimensiune 128. Descriptorul se obţine în felul următor:

- consideraţi pentru fiecare punct un patch (o fereastră din imagine) centrat în punctul respectiv şi împărţit în 4×4 celule (Figura 2c);
- fiecare celulă este pătrată, de latură dimensiuneCelula (= 4 pixeli, în implementare, Figura 2d);
- pentru fiecare celulă, calculaţi o histogramă de dimensiune 8 după orientările gradienţilor fiecărui pixel din celulă;
- concatenaţi cele 16 histograme ale celor 16 celule într-un singur vector de dimensiune 128.

Funcţia *calculeazaHistogrameGradientiOrientati.m* returnează o matrice de dimensiuni $N \times 128$ conţinând descriptorii HOG pe linie pentru fiecare din cele N puncte dintr-o imagine. De asemenea, funcţia returnează şi fiecare patch asociat fiecărui punct într-o matrice $N \times (16 * 16)$, fiecare linie conţinând un patch (luat coloană cu coloană). Această matrice este folosită la vizualizarea vocabularului (mulţimea de cuvinte vizuale, vedeţi secţiunea următoare). Funcţia pe care trebuie să o completaţi are forma următoare:

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

4.2 Construcţia vocabularului (2 puncte)

Pentru aplicarea clasificării cu modelul BOVW veţi construi un vocabular vizual. În acest sens veţi clusteriza un număr mare de descriptori HOG folosind algoritmul *k-means*. Funcţia *kmeans_iter.m* este deja implementată.

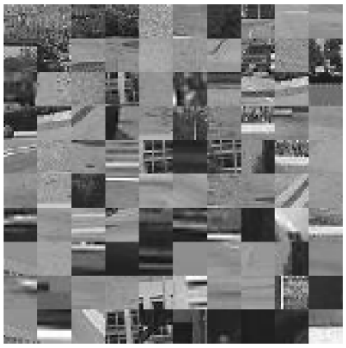


Figura 3: Vizualizare vocabular: în figură sunt afişate un număr de 100 de cuvinte vizuale.

Funcţia *construiesteVocabular.m* implementează acest pas intermediar. Funcţia primeşte ca input numele unui director, iar pentru fiecare imagine de antrenare din acest director calculează punctele din imagine pe baza unui caroiaj (apelează funcţia *genereazaPuncteCaroiaj*), extrage descriptorii pentru fiecare punct (apelează funcţia *calculeazaHistogrameGradientiOrientati.m*). Apoi funcţia clusterizează toţi descriptorii extraşi din imaginile de antrenare folosind algoritmul *k-means* (apelează funcţia *kmeans_iter.m*). Centri clusterilor sunt vectori de dimensiune 1×128 şi îi vom numi *cuvinte vizuale*. În total sunt 10000 de descriptori: din fiecare imagine de antrenare extragem $10 * 10$ puncte, caracterizăm fiecare punct cu

un descriptor, există 50 de imagini de antrenare pozitive şi 50 de imagini de antrenare negative.

Vizualizarea vocabularului obţinut (mulţimea de cuvinte vizuale, Figura 3) se realizează apelând funcţia *vizualizeazaDictionar.m*, funcţie scrisă în întregime.

4.3 Reprezentarea imaginilor cu modelul BOVW (2 puncte)

Folosind vocabularul de cuvinte vizuale obţinut, veţi reprezenta fiecare imagine ca o histogramă de cuvinte vizuale. Această reprezentare, cunoscută şi sub numele de reprezentare BOVW, înregistrează ce cuvinte vizuale apar în imagine şi de asemenea frecvenţa lor de apariţie.

Completaţi codul funcţiei *calculeazaHistogramaBOVW.m* care calculează histograma BOVW asociată fiecărei imagini. Funcţia primeşte ca input mulţimea de descriptori HOG extraşi din imagine şi vocabularul de cuvinte vizuale. Pentru obţinerea histogramei imaginii, calculaţi pentru fiecare descriptor HOG în parte cuvântul vizual cel mai apropiat (în sensul distanţei Euclidiene) şi număraţi câţi descriptori sunt asignaţi fiecărui cuvânt vizual. Funcţia pe care trebuie să o completaţi are forma următoare:

histogramaBOVW = calculeazaHistogramaBOVW(descriptoriHOG, cuvinteVizuale)

Completaţi codul funcţiei *calculeazaHistogramaBOVW_director.m* care citeşte toate imaginile dintr-un director şi calculează histograma BOVW asociată fiecărei imagini. Output-ul acestei funcţii este o matrice cu numărul de linii egal cu numărul de imagini şi numărul de coloane egal cu numărul de cuvinte vizuale. Fiecare linie reprezintă histograma BOVW a unei imagini. Funcţia pe care trebuie să o completaţi are forma următoare:

histogrameBOVW = calculeazaHistogrameBOVW_director(numeDirector, cuvinteVizuale)

Apelând această funcţie procesăm exemplele pozitive (imagini ce conţin maşini) şi exemplele negative (imagini ce nu conţin maşini) din directoarele *masini-exempleAntrenare-pozitive* şi *masini-exempleAntrenare-negative*. Colectăm histogramele rezultante în matricele *histogrameBOVW_exemplePozitive* şi *histogrameBOVW_exempleNegative*.

4.4 Clasificarea imaginilor (2 puncte)

Funcţia *clasificaBOVW.m* (funcţia este scrisă în întregime) implementează etapa de clasificare pe baza unui clasificator. Sarcina voastră este de a scrie funcţiile care implementează clasificatorii. În această temă veţi experimenta doi clasificatori: clasificatorul bazat pe cel mai apropiat vecin şi clasificatorul liniar implememtat cu un SVM (Support Vector Machine).

a. Clasificatorul cel mai apropiat vecin

Realizaţi mai intâi clasificarea folosind principiul celui mai apropiat vecin. Pentru clasificarea unei imagini test, calculaţi histograma de cuvinte vizuale şi asignaţi eticheta (pozitivă sau negativă) corespunzătoare celui mai apropiat vecin.

Completaţi funcţia *clasificaBOVWCelMaiApropiatVecin.m* care compară histograma BOVW a unei imagini test cu histogramele exemplelor pozitive şi exemplelor negative calculate anterior şi returnează eticheta corespunzătoare celui mai apropiat vecin (eticheta 1 dacă cel mai apropiat vecin este un exemplu pozitiv, 0 altfel). Funcţia pe care trebuie să o completaţi are forma următoare:

eticheta = clasificaBOVWCelMaiApropiatVecin(histogramaBOVW_test, histogrameBOVW_pozitive, histogrameBOVW_negative);

b. Clasificatorul SVM liniar

Implementaţi în partea a doua clasificatorul bazat pe un SVM liniar folosindu-vă de implementarea oferită de biblioteca *LibSVM* sau de biblioteca *vlfeat* disponibilă online la `http://www.vlfeat.org/`. În particular, veţi folosi pentru antrenarea unui SVM liniar care să distingă exemplele pozitive de cele negative funcţia *vl_svmtrain* (aveţi detalii despre ea aici `http://www.vlfeat.org/matlab/vl_svmtrain.html`). Fişierul MEX *vl_svmtrain* din directorul 'cod' merge pe calculatoare ce au ca sistem de operare Windows pe 64 de biţi. Pentru alte sisteme de operare folosiţi fişierele MEX corespunzătoare din directorul 'cod/mexfilesWindows+Mac+Linux/'.

4.5 Predarea proiectului (2 puncte)

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

Puneţi într-un document cu numele **tema4_raspunsuri.pdf** răspunsurile voastre la următoarele cerinţe:

(a) plotaţi performanţa clasificării (procentul de imagini clasificate corect) pentru cei doi clasificatori (cel mai apropiat vecin, SVM liniar) în funcţie de numărul de cuvinte vizuale k = [5 10 25 50 100 200 500 1000];

(b) pe baza graficului de la punctul anterior determinaţi valoarea k optimă pentru fiecare clasificator care conduce la o perfomanţă optimă a clasificării;

(c) determinarea valorii k optime la punctul anterior se realizează pe exemplele de testare. Acest lucru ar trebui să se realizeze totuşi pe mulţimea de antrenare. Care ar fi o soluţie în acest sens?

(d) rulând funcţia *ruleazaBOVW.m* de mai multe ori cu aceeaşi valoare k, veţi observa că perfomanţa clasificatorilor variază. Cum explicaţi acest lucru? Cum puteţi caracteriza în acest caz performanţa clasificatorilor?

Trimiteţi cele două fişiere (**tema4_cod.zip** şi **tema4_raspunsuri.pdf**) la adresa de email **raducu.ionescu@gmail.com** precizând numele şi grupa din care faceţi parte.
Termenul limită de predare a proiectului este joi, 28 decembrie 2017, ora 23:59.