UNIVERSITATEA ALEXANDRU IOAN CUZA DIN IAȘI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

Analiza Algoritmului Bag of Visual Words

propusă de

Guler Afanasie

Sesiunea: Iulie, 2018

Coordonator științific

Lect. dr. Ignat Anca

Universitatea “Alexandru Ioan Cuza” din Iași

Facultatea de Informatică

Analiza Algoritmului Bag of Visual Words

Propusă de:

Guler Afanasie

Sesiunea: iulie, 2018

Coordonator Ştiinţific:

Lect. Dr. Ignat Anca

Universitatea “Alexandru Ioan Cuza” din Iași

Facultatea de Informatică

Cuprins

Listă figuri

Listă tabele

Abbreviations

BOVW Bag of Visual Words

SIFT Scale-Invariant Feature Transform

SURF Speeded-Up Robust Features

DoG Diferenta Gaussiana

SVM Support Vector Machine

BRIEF Binary Robust Independent Elementary Features

STAR Center Surrounded Extrema

CenSurE Center Surrounded Extrema

SVC Support Vector Classification

ORB Oriented FAST and Rotated BRIEF

FAST Features from Accelerated Segment Test

CNN Retele Neuronale Convultionale

KNN K-Nearest Neighbors

RBF Radial Basis Function

MLP Multilayer perceptron

NLP Neuro-linguistic programming

Avizat,

Îndrumător lucrare de licență

Lect.dr. Ignat Anca

Data 27 Iunie, 2018

DECLARAŢIE PRIVIND ORIGINALITATE ŞI RESPECTAREA DREPTURILOR DE AUTOR

Subsemntatul Guler Afanasie cu domiciliul în Vaslui, Județul Vaslui, născut la data de 22 martie 1995, identificat prin CNP 1950322803941, absolvent al Universităţii "Alexandru Ioan Cuza" din Iaşi, Facultatea de Informatică, specializarea Informatică, promoţia 2018, declar pe propria răspundere, cunoscând consecinţele falsului în declaraţii în sensul art. 326 din Noul Cod Penal şi dispoziţiile Legii Educaţiei Naţionale nr. 1/2011 art.143 al. 4 si 5 referitoare la plagiat, că lucrarea de licenţă cu titlul: "Analiza algoritmului Bag of Visual Words" elaborată sub îndrumarea d-na Ignat Anca, pe care urmează să o susţină în faţa comisiei este originală, îmi aparţine şi îmi asum conţinutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licenţă să fie verificată prin orice modalitate legală pentru confirmarea originalităţii, consimţind inclusiv la introducerea conţinutului său într-o bază de date în acest scop.

Am luat la cunoştinţă despre faptul că este interzisă comercializarea de lucrări ştiinţifice în vederea facilitării falsificării de către cumpărător a calităţii de autor al unei lucrări de licenţă, de diploma sau de disertaţie şi în acest sens, declar pe proprie răspundere că lucrarea de faţă nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Iaşi, 27 Iunie, 2018

Guler Afanasie

DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licenţă cu titlul "Analiza algoritmului Bag of Visual Words", codul sursă al programelor şi celelalte conţinuturi (grafice, multimedia, date de test etc.) care însoţesc această lucrare să fie utilizate în cadrul Facultăţii de Informatică. De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea Alexandru Ioan Cuza Iaşi să utilizeze, modifice, reproducă şi să distribuie în scopuri necomerciale programele-calculator, format executabil şi sursă, realizate de mine în cadrul prezentei lucrări de licenţă.

Iaşi, 27 Iunie, 2018

Guler Afanasie

1 Abstract

Creierul uman poate face foarte ușor diferența între un leu și un jaguar, această diferențiere pare ușoară pentru că creierul nostru este foarte bun la clasificarea imaginilor, cu toate acestea pentru un calculator această diferențiere poate fi foarte dificilă.[Abadi et al., 2015]

Crearea unui model capabil să diferențieze imaginile în funcție de conținutul acestora este obiectivul propus pentru această lucrare licență, pentru aceasta voi folosi conceptul de "visual words" [Berthier Ribeiro-Neto, 1999] pentru un model împrumutat din Neuro-linguistic programming (NLP) "Bag of Words" care își are rădăcinile din articolul "Distributional Structure" [Harris, 1954], și are la bază analiza imaginii/textului în funcție de cuvintele vizuale/cuvintele care îl alcătuiesc.

2 Introducere

Clasificarea imagininilor este o problemă fundamentală în domeniul procesării de imagine, de a lungul timpului au fost propuse o varietate de abordari diferite.

Aplicabilitiatea unui asemenea algoritm poate fi multiplă așa cum identificarea obiectului din imagine fară prezenţa unui factor uman poate fi de un ajutor in multe domenii.

Un exemplu de utilizare ar putea fi detectarea animalelor prezente în un parc național, presupunând că există amplasate niște camere în acest parc, prezenţa animalelor va fi rar surprinsă însa un clasificator ar putea determina daca în imagine se află sau nu un animal şi chiar să determine specia animalului.

Odată ce imaginile sunt reprezentate ca matrici cea mai simpla abordare în identificarea similaritaţilor ar putea fi calcularea unei distanțe Euclidiene intre pixelii imagini, aceasta abordare este posibila însa apar unele probleme precum:

- imagnile pot fi de diferite dimensiuni
- obiectul din imagine poate fi rotit
- luminozitatea diferă de la o poză la alta
- obiectul poate fi parţial acoperit
- distanţa de la camera pană la obiect diferă

Aceste probleme pot fi atribuite prin folosirea unor descriptori de mărime fixă independenți de poziționare în imagine, aceasta este abordarea modelul BOVW.

Modelul BOVW poate fi împartit în urmatorii 4 paşi:

- Extragerea trăsaturilor caracteristice imagini(cuvintele)
- Construcția vocabularului clusterizând aceste trăsaturi, astfel dimensiunea vocabularului este dată de numarul de clustere.
- Clasificarea imaginilor după aceste trasaturi, stabilirea cuvintelor care intră în componenţa unei anumite categorii.

- Obţinerea categorii care este cea mai apropiată de imaginea de test.

În BOVW un cuvânt vizual este considerată o trasatură, o parte a imaginii care reprezintă o caracteristică a obiectului prezent în imagine, o mini imagine, în cadrul acestei implementari este considerat punct de interes zona care conține un unghi.

Cu ajutorul algoritmilor de detectare a unghiurilor vom gasi aceste puncte de interes și le vom folosi pentru clasificarea imaginilor.

3 Contribuții

Prin analizarea algoritmului de clasificare BOVW am adus următoarele contribuții:

- Oferirea de informații din aceiași perspectivă pentru algoritmii de extragere a descriptorilor din imagini, a clasificatorilor în contextul clasificarii imaginilor la fel și parametrilor pentru aceștia.
- Testarea în aceleași condiții a diverselor implementări posibile ale algoritmului, oferirea unor statistici complete cu referire la timpul de execuție, timp de antrenament, timp de testare, toleranța la diverși factori precum dimensiunea imaginilor de antrenament, numarul imaginilor de antrenament, modul în care obiectele sunt surprinse în imagini, etc.
- Compararea modelului cu alte implementări posibile, testarea acestora pentru același seturi de date, același hardware cu pointare spre caracteristicile superioare dar și cele inferioare față de modelul BOVW.
- Crearea unui model BOVW modular, ușor configurabil, pentru utilizarea mai multor algoritmi de extragere a descriptorilor, mai multor implementări de clasificatori, diverși parametri la algoritmul de clusterizare sau la modul de utilizare a imaginilor din setul de imagini.

4 Descrierea Algoritmului

Cum am specificat anterior BOVW poate fi împarţit in 4 etape: Extragerea descriptorilor, construcţia vocabularului, clasificarea imaginilor și testarea modelului rezultat. Pentru primele 3 etape avem posibilitatea de alege algoritmi diferiţi, fiecare având posibile puncte forte dar si puncte slabe, astfel pentru fiecare etapa o să descriu potenţialii algoritmi pentru implementarea algoritmului BOVW pentru ca apoi să-i testez și să-i selectez pe cei care dau cele mai bune rezultate.

4.1 Extragerea descriptorilor

În "computer vision"[1] descriptorii de imagini reprezintă informații cu privire la conţinutului imaginii, ei descriu elemente caracteristice precum formă, culoare, textură.

Modelul BOVW propune folosirea descriptorilor de formă pentru secvenţe mici ale imaginii de marime prestabilită, această abordare rezolvă 2 din problemele expuse anterior, și anume faptul ca distanţa faţa de camera poate varia, astfel prin selectarea descriptorilor în un mod piramidal putem obtine o invarianţă pentru distanţă, cealaltă problemă rezolvată de această abordare este faptul că un obiect acoperit de un alt obiect strain poate parea diferit cu toate ca sunt prezente destul caracteristici care ne pot da informatii destule cu legatura la clasa obiectului din imagine.

Pentru implementarea primului pas din algoritmul BOVW pot fi considerate “cuvinte” toate secventele marime N x M, unde N, M sunt predefinite. Unele din aceste mini imagini nu sunt relevante pentru clasificare împactand atât performanța algoritmului cât și acuratețea, astfel pentru a depista zonele care oferă cat mai multa informaţie vom folosi dectectori a unghiurilor care ne oferaă mai multă informaţie cu legatură la imagine. Mini imaginile cu castig maxim de informaţie sunt acele care conţin unghuiri, pentru a depista aceste zone avem la dispozitie mai multe metode.

Algoritmul lui Moravec [Moravec, 1980] este unul din primii algoritmi pentru detectarea unghiurilor, acesta iterează prin toti pixeli comparându-i cu vecinii săi, similari-

[1]Subdomeniu din știința calculatoarelor care se ocupă cu obținerea informațiilor de nivel înalt din conținut video digital

taţile sunt date de suma diferenţelor patratelor.

Algoritmul creează zone de comparare dacă aceste zone sunt similare atunci nu exista nici un unghi, daca pe orizontală zonele sunt diferite insa pe vertical similare atunci ne aflam pe o margine, daca zonele sunt diferite între ele atunci am detectat unghiul. Algoritmul insa nu este perfect asa cum acesta nu este isotropic, astfel daca există o margine care nu e in zona vecinilor(orizontal, vertical, diagonal) atunci suma diferenţelor patratelor va fi mare si va indica un unghi incorect.

Detectorul lui Harris [Harris and Stephens, 1988] a apărut ca o îmbunătaţire la algoritmul lui Moravec, îmbunatatirea adusă a fost modul în care este deplasată zona de calcul, acesta foloseste unghiul care este dat de o margine gasită. Acest detector s-a dovedit a fi mai eficient, insă acesta nu este invariant la marimea imaginii.

Diferenţă Gaussiană [Davidson and Abramowitz, 2002] prin blurarea imaginii de 2 ori cu intensitaţi diferite și calcularea diferenţei dintre acestea putem obţine valori care ne pot indica unde sunt marginile şi unghiurile în imagine. Aceasta abordare este numită diferenţa Gaussiană, aceasta abordare este mai eficientă decât Laplacian of Gaussian (LoG) care are la baza detecotrul lui Harris si foloseşte la fel blurarea Gaussiana pentru a rezolva problema invariaţei pe marime.

$$DoG = G_{\sigma_1} - G_{\sigma_2} = \frac{1}{\sqrt{2\pi}}\left(\frac{1}{\sigma_1}e^{-(x^2+y^2)/2\sigma_1^2} - \frac{1}{\sigma_2}e^{-(x^2+y^2)/2\sigma_2^2}\right)$$

Pentru alegerea punctelor de interes din o imagine cea mai suitabilă soluţie este Diferenţa Gaussiană, aceasta este mai eficientă decât restul şi ne rezolvă problema mărimii obiectului.

Pentru rezolvarea problemei rotației obiectului in imagine, fiecare punct considerat de interes va fi pozitionat după varful unghiului, de fiecare data când un punct de interes va fi detectat acesta va fi rotit dupa acelaşi principiu, astfel va fi în aceiaşi pozitie cu un potenţial descriptor similar al aceluiaşi obiect din altă imagine. Având punctele de interes detectate le putem folosi ca centru pentru descriptor, astfel selectăm pixelii din jurul acestuia pentru a crea mini imaginea.

Pentru a rezolva problema invarianței pe marime ar trebui sa blurăm imaginea în funcţie de marimea unghiului detectat, aşa cum un obiect poate fi localizat mai aproape sau departe de obiectiv.

4.1.1 SIFT

Algoritmul SIFT (Scale-Invariant Feature Transform) a fost introdus in 2004 de catre [G.Lowe, 2004]. SIFT foloseşte diferenţa gaussiană şi detectorul Harris pentru identificarea punctelor de interes. Passi algoritmului SIFT vor fi descrişi în paragrafele urmatoare.

Localizarea punctelor de interes in SIFT se realizeaza prin folosirea Diferenta Gaussiana (DoG), pentru o acurateţă mai buna sunt interpolate punctele apropiate. Pentru invarianta pe diferite dimensiuni sunt folosite mai multe octave [2].

Deoarece DoG returnează multe muchii prin unghiurile detectate, SIFT mai introduce o etapa prin care elemină valorile care se adeveresc a fi muchii.

Atribuirea unei orientări este necesară pentru a putea identifica descriptorii în un mod eficient, acestia trebuie să fie orientaţi la fel, dacă aceialaşi descriptor are rotaţii diferite algoritmul trebuie să-l salveze la fel.

Pe imaginea blurată la pasul anterior se calculează o histogramă cu 36 de orientari, fiecare acoperind 10 grade.

Aceasta se calculează folosind diferenţele între pixeli:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y)^2) + (L(x,y+1) - L(x,y-1)^2)}$$
$$\theta(x,y) = atan2(L(x,y+1) - L(x,y-1), L(x+1,y) - L(x-1,y))$$

Cel mai lung vector va fi orientarea imaginii, dacă există mai multe cu aceleaşi valori sunt creaţi mai multi descriptori identici cu orientari diferite.

[2] Imagini obținute prin blurarea cu mai multe intensitați imaginea de bază

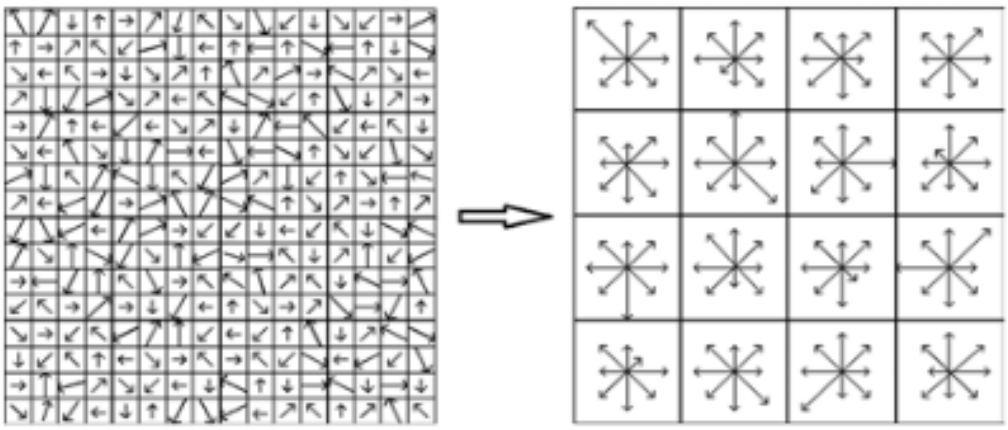


Figure 1: Forma descriptorului SIFT [its, 2014]

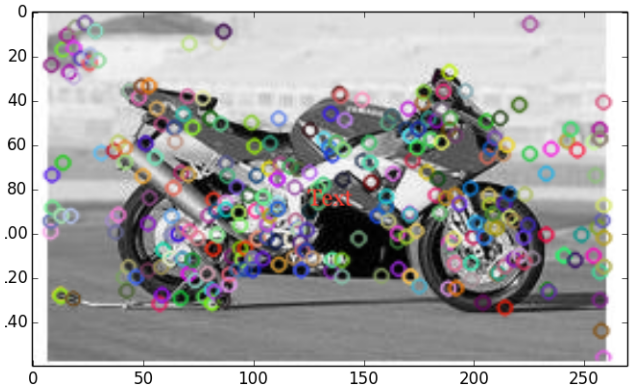


Figure 2: Detectarea descriptorilor SIFT

Crearea descriptorilor în SIFT se realizeaza prin folosirea unei matrici de 16 x 16 pixeli din jurul punctului de interes. Acești 16 pixeli sunt luaţi din imaginea care aparţine octavei sale. Din aceştia este creata o histograma cu 8 vectori fiecare, aceasta reprezentare este salvată în un vector unidimensional de marimea de 128 elemente (figura 1). Acest vector este apoi normalizat pentru a crea invariaţa faţă de lumină sau contrast a acestora.

Am aplicat algorimtul pe o imagine(figura 2) unde putem vedea descriptorii detectaţi de acesta.

Utilizând algoritmul SIFT de pe imagini de dimensiuni de aproximativ 250X150 se pot extrage în jur de 300 de descriptori, aceasta valoarea insa poate varia în funcţie de densitatea unghiurilor din imagine, acestea pot fi limitate la un numar dorit de utilizator prin reglarea sensivitaţii DoG Distanţa euclideană între acest tip de descriptori ne pot

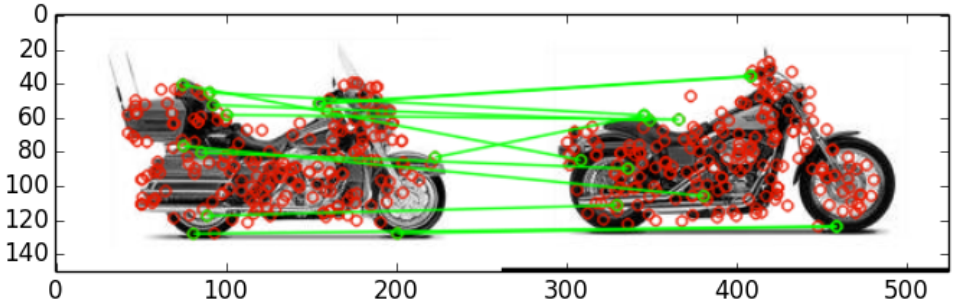


Figure 3: Potrivirea descriptorilor SIFT

confirma similaritaţile între caracteristicile extrase. Am extras descriptorii pentru doua imagini din aceiaşi categorie şi putem observa potrivirile descriptorilor acestora in figura 3.

Pentru potrivirea descriptorilor am folosit Flann matcher, care are la bază un clasificator de tip K-Nearest Neighbors (KNN). Se poate aplica şi un algoritm de tip brute-force însa acesta pentru un numar mare de descriptori va fi mult mai lent.

4.1.2 SURF

(Speeded-Up Robust Features) SURF [Herbert Bay and Gool, 2006] a fost prezentat ca o alternativă mai rapidă a algoritmului SIFT.

Detectarea punctelor de interes la algoritmul SURF se face prin o abordare care aduce un bonus de eficienţă faţă de algoritmul SIFT, acesta folosinduse de matricea Hessiană.

Pentru scalarea pe dimensiuni diferite spre deosebire de abordarea folosită de SIFT, SURF foloseşte maşti care sunt marite odata cu avansarea algoritmului, astfel for fi folosite masti încrementând dimensiunile acestora 9×9, 15×15, 21×21, 27×27,...

Pe matricile obţinute folosind maştile se consideră puncte de înteres acele zone al căror determinant al matricii Hessiene este maxim.

Interpolarea punctelor de interes este şi la SURF o etapă importantă, a şa cum diferenţele între octave pot fi destul de mari.

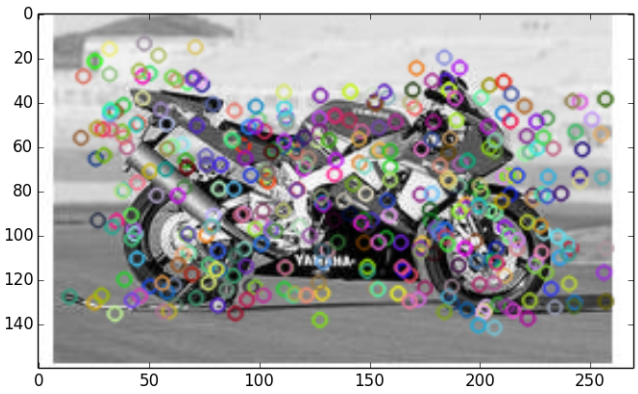


Figure 4: Detectarea descriptorilor SURF

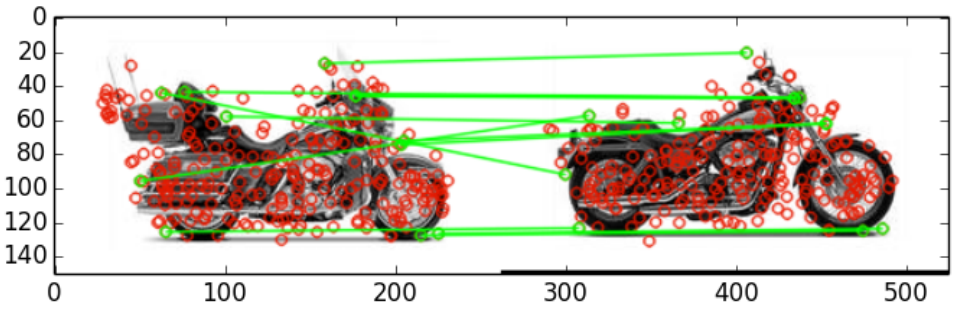


Figure 5: Potrivirea descriptorilor SURF

Crearea descriptorilor în SURF se foloseste de matricea Wavelet pentru a crea un descriptor cu 64 sau 128 de dimensiuni folosind o zonă de 20 x 20 pixeli din octava în care a fost detectat punctul de interes. Putem vedea descriptorii detectaţi pentru aceiași imagine de mai sus în figura 4.

Descriptorii SURF pot fi comparaţi în acelaşi mod ca şi descriptorii SIFT având în vedere faptul ca au aceiaşi dimensiune, pot fi folosite aceiaşi structură de date pentru memorarea acestora şi acelaşi algoritm de potrivire (Flann matcher) putem obţine rezultate similare(figura 5).

4.1.3 FAST

FAST (Features from Accelerated Segment Test) [Rosten and Drummond, 2006] [Damian Eads and Rosten, 2013] este un algoritm de detectare a unghiurilor iar pentru BOVW îl poate fi folosit pentru detecţia punctelor de interes. FAST îşi are ca scop să atingă o performanţă

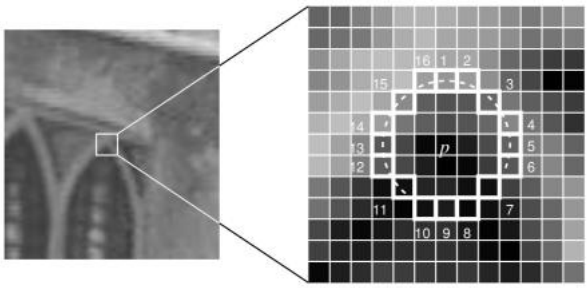


Figure 6: Detectorul FAST [its, 2014]

comparabilă cu alţi detectori de unghiuri însa cu un consum de resurse redus.

- Alege un pixel P canditat pentru punct de interes
- Alege un prag t
- Luăm un cerc de 16 pixeli în jurul pixelului ales (Figura 6)
- Dacă există un unghi atunci există n puncte în cercul de 16 pixel care sunt mai luminoşi decât $I_p + t$ sau mai intunecate decât $I_p - t$, valoarea implicită a lui n este 12.
- Un test rapid a fost propus pentru a depista dacă un punct de interes nu este unghi. Sunt examinati doar 4 pixeli petntru început 1 şi 9, dacă ambii sunt mai luminoşi sau inchişi se mai examinează şi punctele 5 şi 13. Cel putin 3 din 4 trebuie sa fie mai luminoşi sau inchişi pentru ca P să fie unghi.

4.1.4 BRIEF

BRIEF [Michael Calonder and Fua, 2010] după cum ştim SIFT foloseşte vectori de 128 de elemente care ocupă 512 octeţi, doar o mica parte din aceasta informaţie este utilă, astfel se poate aplica o metoda de hashing pentru a crea un nou descriptor mai eficient, de 128 de biti pentru a reprezenta imaginea, pentru compararea acestor descriptori se poate folosi distanţa hamming care este mai putin costisitoare faţă de distanţa euclidiană.

Descriptorii BRIEF sunt de acest tip aceştia însa folosesc o metodă unică descrisă în lucrarea [Michael Calonder and Fua, 2010], această metodă îmbunataţeste viteza de creare a descriptorului.

4.2 Clasificarea imaginilor

O parte importantă a algoritmului BOVW este modul in care descriptorii sunt folosiţi pentru a clasifica cât mai bine imaginea, Așa cum aplicarea algoritmului SIFT poate extrage un număr mare de descriptori iar pentru o acurateţe mai bună e nevoie de un numar mare de date de antrenament, astfel eficienţa este de o importanţă sporită. Problema poate poate fi imparţită in 2 parţi:

- Gruparea descriptorilor similari care reprezintă aceiaşi mini imagine
- Maparea grupurilor clasei careia fac parte

4.2.1 Brute Force

Cea mai simplă abordare este un model brute-force, astfel compăram imaginea de test cu fiecare imagine de antrenament, o asemenea abordare are o acuratete bună însa eficienţa este compromisă, Astfel pe un set de date cu 2 categorii, 28 de imagini de antrenament si 4 imagini de test toate cele 4 imagini au fost clasificate corect însa timpul de clasificare al unei imagine este aprixiimativ 2 secunde. Urmatoarea secvenţă de cod returnează indexul celei mai apropiate imagini de imaginea de test, astfel categoria cu cele mai multe potriviri cu imaginea de test va fi clasificarea imaginii.

```
import cv2
def classify_image(img, imlist):
  for i in range(len(imlist)):
    kp1, des1 = sift.detectAndCompute(img, None)
    kp2, des2 = sift.detectAndCompute(train_img, None)
    matches = flann.knnMatch(des1, des2, k=2)
    for i, (m, n) in enumerate(matches):
    if m.distance < 0.75 * n.distance:
                category[i]+=1
    return max(category)
```

4.2.2 Clusterizare si Clasificare

O abordare mai eleganta si eficientă este utilizarea unei clusterizări, astfel obţinem un set de k clustere unde un cluster reprezintă o caracteristică, un cuvânt vizual. În acest mod putem eficient detecta caracteristica care este reprezentată de un descriptor.

K-Means este un algoritm de clusterizare des utilizat, aşa cum noi nu cunoastem tipul cuvantului vizual avem nevoie de gruparea mini-imaginilor similare intr-un mod eficient, K-Means este ceea de ce avem nevoie.

O caracteristică poate să fie prezentă în mai multe categorii de imagini, astfel pentru o categorizare corecta a imaginii trebuie să cunoaştem ce caracteristici sunt specifice pentru fiecare categorie, această problema poate fi rezolvată de un clasificator.

Alegerea unui clasificator poate fi dificilă pentru acest tip de problemă, astfel vom încerca o serie de clasificatori şi la final cel care va scoate cele mai bune rezultate va fi selectat.

SVC este un clasificator bazat pe Support Vector Machine (SVM) care folosește punctele de frontieră pentru a delimita clasele, astfel cele mai apropiate puncte de cealaltă clasa sunt folosite pentru determinarea frontierei, aceasta este aleasă astfel încat laţimea acesteia să fie maximă.

KNN este un algoritm de clasificare care atribuie clasa în funcţie de majoritatea a K vecini ai punctului de test.

Bayes Naiv considera fiecare parametru independent la clasificare, excluzând faptul că poate există o corelaţie între aceștia.

Multilayer perceptron (MLP) este un o reţea neuronala cu minim 3 straturi, acest timp de reţea poate fi folosită pentru clasificarea unor puncte care nu pot fi separate liniar. Reţeaua este formată din mai mulţi perceptroni organizaţi pe mai multe nivele unde un perceptron este un algoritm de determinare daca elementul apartine sau nu unei clase.

Testare Pentru o comparaţie cu modul brute-force am testat pe un set de 30 de date de antrenament şi 30 date de test din 3 categorii diferite extrase din acelaşi dataset aceasta abordare a atins 93% acurateţe iar antrenamentul dureaza mai puţin de 5 secunde. Desigur pe un set mai mare de date şi mai multe categorii aceasta acurateţe se schimba însa comparatativ cu modelul brute force acesta este mai eficient, înregistrând si o acurateţe foarte buna. Pentru acest test am folosit descriptori u-SURF, clusterizare K-Means cu 30 de clustere şi clasificatorul SVC.

5 Implementare Algoritm

Pentru implementarea algoritmului BOVW am apelat la diverşi algoritmi de extragere a descriptorilor precum SIFT , SURF, BRIEF, Oriented FAST and Rotated BRIEF (ORB), aceştia se aseamănă prin forma descriptorilori toţi fiind de lungime fixă de 128 elemente reale sau boolene. Aceşti algoritmi sunt oferiţi de libraria OpenCV [Bradski, 2000] pe care o să o folosesc.

Pentru clasificarea descriptorilor şi maparea acestora imaginilor este evidentă superioritatea unui algoritm care se bazează pe clusterizare şi clasificare, pentru clusterizare vom folosi doar K-Means dar pentru clasificare o să testăm o suită mai mare de clasificatori precum SVC, KNN, MLP, etc.

5.1 Seturi de imagini

La fel ca majoritatea problemelor de recunoaştere algoritmul se împarte in 2 categorii: antrenament si testare, pentru aceasta avem nevoie de cel puţin un set de date. Am folosit mai multe seturi de imagini cu caracteristici diferite pentru a acoperi o arie mai mare de moduri în care putem obține imaginile.

Aceste dataset-uri vor fi modificate corespunzator pentru scalare sau pentru testarea pe un numar mai mic de categorii sau imagini.

5.1.1 Corel 10k

[Guang-Hai Liu, 2015] Corel este un dataset de 10000 de imagini de mărime mici fixe (192x128 sau 128x192), fiecare categorie conținând 100 de imagini fiind 100 de categorii. Corel conține imagini cu animale, peisaje, oameni practicând diferite activități, etc.

Obiectele se pot afla la distanță mare de obiectiv sau poate fi surprinsă doar o parte a obiectului, deasemenea unghiul din care obiectul este surprins diferă de la o poză la alta. Fundalul poate fi de o altă categorie dacă există obiectul în ea. O parte din categorii pot fi ușor diferențiate prin culori diferite însă algoritmul propus nu folosește culorile pentru clasificare.

5.1.2 GHIM 10k

GHIM [Guang-Hai Liu, 2013] este un set de imagini de 10000 imagini din 20 categorii. Categoriile sunt diverse precum: apus de soare, barci, flori, cladiri, masini, munti, etc. Fiecare categorie contine 500 de imagini de marimi 400 x 300 sau 300 x 400.

În GHIM obiectele sunt amplasate la distanțe destul de asemanatoare fata de obiectiv, iar unghiul nu diferă la fel de mult, este încercată surprinderea întregului obiect pentru imagini fară margini prea mari, în unele imagini obiectul nu este complet, însa tendința predominantă este de a surprinde întregul obiect.

5.1.3 Caltech-101

Caltech 101 [Fei-Fei Li and Ranzato, 2003] este un set de imagini cu 101 categorii, sunt între 40 și 800 de imagini pe categorie majoritatea categoriilor având în jur de 50 de imagini. Rezoluția imaginilor este în jur de 300 x 200.

Caltech 101 poate fi caracterizat prin faptul că imaginile prezintă obiectul din același unghi, există categorii precum: "față de pumă" sau "mașină din plan", iar distanță de la obiectiv este predominant aceiași, dimensiunile imaginii variază semnificativ la fel și fundalul, este încercată surprinderea obiectului în mai multe contexte dar din aceiași poziție.

5.1.4 Caltech-256

Caltech 256 [Griffin, 2006] este o colecție de 30607 imagini aparținând 256 categorii cu rezoluții variind între 200-500 x 200-500. Variația între rezoluții este semnificativă, la fel este și numărul imaginilor pe categorie.

În acest dataset obiectele sunt expuse pe întreagă imagine cu foarte puțin spațiu pentru fundal, foarte des sunt întâlnite mai multe obiecte din aceiași categorie în imagine, la fel se întâlnesc și poze care nu cuprind întregul obiect.

5.2 Hardware

Pentru aceste rulari a fost folosit un calculator de tip Macbook Pro cu specificațiile: 8 Gb memorie și procesor dual core i5 de 2.7 GHz 4 thread-uri și o placa grafică integrată de 1536 Mb, acesta deține 128Gb memorie internă de tip SSD cu viteză de scriere și citire de aproximativ 500 Mb/s, sistemul Mac OS folosește această memorie drept "swap" astfel această viteză afectează la rândul ei viteza totală de antrenare/testare.

5.3 Descriptori

Extragerea descriptorilor este o parte caracteristică algoritmului astfel ea este de o importanţă sporită, vom folosi mai mulţi algoritmi care ne vor returna un set de descriptori de marime fixă: 128 de valori reale.

5.3.1 Extragerea descriptorilor

O astfel de clasă ca şi cea de mai jos va fi responsabilă de găsirea descriptorilor pentru o imagine. O variabila va desemna algoritmul folosit pentru a putea rapid schimba tipul detectorului dar si a descriptorilor, astfel putem crea un program care va folosi mai mulţi descriptori la aceiaşi rulare.

```
class Sift:
    def __init__(self):
        self.sift_object = cv2.xfeatures2d.SIFT_create()

    def get_features(self, image):
        kp, des = self.sift_object.detectAndCompute(image, None)
        return [kp, des]
```

5.3.2 Tipuri de descriptori

Deasemenea voi testa şi alţi algoritmi de detectare de puncte de interes precum: Center Surrounded Extrema (STAR) care este implementarea din OpenCV al algoritmului Center

Surrounded Extrema (CenSurE) [Agrawal M., 2008] şi BRIEF [Michael Calonder and Fua, 2010]. Sau ORB [Ethan Rublee and Bradski, 2011], care combină algoritmul FAST pentru detectarea punctelor şi BRIEF pentru crearea descriprilor, cei mai bun vor fi incluşi în statisticile pe care urmează să le prezint.

Algoritmii pe care ii voi folosi pentru identificarea şi extragerea desriptorilor:

- SIFT
- SURF
- u-SURF foloseşte acelaşi algoritm SURF singura diferenţa fiind orientarea descriptorului aceasta va coincide mereu cu orientarea imaginii.
- puncte de interes SURF şi descriptori BRIEF
- ORB (BRIEF cu FAST)

5.3.3 Comparare descriptori

După cum am specificat anterior o să folosesc mai mulți descriptori.

dataset	SURF	u-SURF	SIFT	BRIEF cu SURF	ORB (BRIEF cu FAST)
Corel-10k 5 cat	52.67%	62.67%	57.33%	59.33%	45.33%
Corel-10k 10 cat	40.33%	51.67%	39.66%	41.0%	30.33%
Corel-10k 20 cat	33.33%	45.68%	30.19%	31.37%	24.7%
Caltech-101 5 cat	76.92%	73.62%	65.93%	64.83%	53.84%
Caltech-101 10 cat	47.80%	63.74%	46.70%	51.1%	41.75%
Caltech-256 5 cat	58.01%	65.19%	59.11%	55.8%	31.19%
Caltech-256 10 cat	48.62%	51.83%	33.94%	43.57%	n/a
GHIM-10k 3 cat	89.33%	94.66%	88.44%	92.0%	85.33%
GHIM-10k 5 cat	81.46%	85.46%	n/a	79.46%	65.73%

Table 1: Acuratețea implementarilor pe descriptori

dataset	SURF	u-SURF	SIFT	BRIEF cu SURF	ORB (BRIEF cu FAST)
Corel-10k 5 cat	58.96s	55.87s	127.3s	42.8s	177.9s
Corel-10k 10 cat	218.0s	188.66s	245.58s	117.98s	1233.54s
Corel-10k 20 cat	1616.64s	1944.6s	1218.08	1382.89s	4741.09s
Caltech-101 5 cat	286.6s	307.33s	294.52s	341.95s	255.43s
Caltech-101 10 cat	919.9s	785.7s	184.57%	1433.03s	2230.87s
Caltech-256 5 cat	2180.73s	2130.30s	5582.61s	3431.48s	n/a
Caltech-256 10 cat	7315.98s	7185.50s	324.45s	8525.15s	4344.28s
GHIM-10k 3 cat	2280.53s	2121.57s	2396.76s	3512.16s	2602.15s
GHIM-10k 5 cat	4172.87s	3886.55s	n/a	5742.86s	3800.0s

Table 2: Timpul de antrenare descriptori

dataset	SURF	u-SURF	SIFT	BRIEF cu SURF	ORB (BRIEF cu FAST)
Corel-10k 5 cat	1.18s	1.5s	2.12s	0.7s	0.6s
Corel-10k 10 cat	2.36s	1.39s	4.92s	1.54s	1.02s
Corel-10k 20 cat	4.37s	2.61s	10.48s	3.03s	2.18s
Caltech-101 5 cat	2.85s	1.68s	4.88s	1.68s	0.59s
Caltech-101 10 cat	5.01s	2.65s	5.6s	2.96s	1.13s
Caltech-256 5 cat	16.99s	8.44s	20.18s	8.8s	n/a
Caltech-256 10 cat	18.22s	11.21s	15.54s	13.4s	3.39s
GHIM-10k 3 cat	14.54s	8.76s	18.16s	9.71s	4.0s
GHIM-10k 5 cat	29.34s	16.53s	n/a	17.97s	6.46s

Table 3: Timpul de clasificare descriptori

Aceste rulari au fost efectuate pe un numar mic de categorii pentru a putea scala hardware-ul, pentru un numar mai mare de categorii este necesara mai multa putere de procesare, cea mai solicitata componenta de acest algoritm o reprezinta memoria.

Deoarece nu avem la dispozitie un claculator cu o putere mai mare de procesare putem reduce din numarul de descriptori selectati de algoritmi de detectie a unghiurilor, astfel pentru SIFT putem sa modificam numarul de descriptori obtinuti din o imagine, aceasta valoare nu este exact numarul pe care algoritmul il va returna acesta poate sa returneze

mai multi sau mai putini insa numarul lor se va afla in jurul acestei valori, valoare pe care o vom folosi va fi de 20 de descriptori pe imaginie.

Pentru SURF putem modifica valoare de limita pentru algoritmul de detectie a unghiurilor bazat pe matrica Hessiană, valoarea utilizată până acum și valoarea implicită este de 200, pentru urmatoarele testari vom folosi o valoare de 2000 pentru imaginile de antrenament însa 1000 pentru valorile de test, pentru a surprinde mai multi descriptori care in alte imagini de test pot fi mai pronuntați.

Din rulările pe un număr mic de categorii putem observa că cea mai bună acuratețe este oferită de algoritmul SURF cu utilizarea opțiunii de setare a orientării descriptorilor aceiași cu orientarea lui în poză.

5.4 Clusterizare

Pentru distanţa între descriptori vom folosi norma euclidiană, aceasta este singura opţiune care o avem la dispozitie oferită de implementarea algoritmului K-Means din SKLearn [Pedregosa et al., 2011].

5.4.1 Alegere număr clustere

O provocare este alegerea numarului de clustere aşa cum acestea influenţeaza performanţa si acurateţea algoritmului, o valoare optima trebuie gasita pentru a putea aproxima valoarea pe care o vom folosi pentru algoritm.

În tabelul 4 putem observa diferenţele de acurateţe între folosirea unui numar diferit de clustere, pentru aceasta testare s-au folosit descriptorii u-SURF şi clasificatorul SVC cu kernelul RBF.

5.4.2 Concluzie

După aceste rulări putem concluziona că valoarea optimă pentru numărul de clustere este direct proporțională cu numărul de categorii aceasta însă depinde și de mărimea imaginilor astfel o imaginile mai mare precum cele din Caltech necesită mai multe clustere pentru

dataset	K=15	K=50	K=300	K=1000
Corel 3 categorii	68.89%	73.3%	72.22%	72.22%
Corel 5 categorii	56.67%	66.0%	64.67%	61.33%
Corel 10 categorii	44.33%	46.0%	51.33%	46.0%
Caltech-101 3 categorii	88.46%	92.3%	92.3%	86.53%
Caltech-101 10 categorii	53.3%	56.6%	61.0%	n/a

Table 4: Acuratete K-Means

dataset	K=15	K=50	K=300	K=1000
Corel 3 categorii	6.2s	16.15s	67.2s	156.96s
Corel 5 categorii	15.26s	41.11s	158.37s	314.05s
Corel 10 categorii	36.82s	101.15s	486.46s	1867.43s
Caltech-101 3 categorii	25.96s	73.56s	293.86s	676.8s
Caltech-101 10 categorii	56.4s	245.32s	2027.85s	n/a

Table 5: Timp executie K-Means

un obiect pe când cele din corel au nevoie de mai puține.

Putem concluziona că numărul optim de clustere este în jur 20 de clustere pentru o categorie, pentru dataset-urile care conțin imagini mai mari dar și mai multe imagini vor mari această valoare iar pentru cele de dimensiuni mici îl vom micșora; o utilizare a unui număr prea mare de clustere nu doar înrăutățește acuratețea dar și impacteaza semnificativ timpul de execuție.

5.5 Clasificare

Pentru alegerea clasificatorului vom încerca o comparație a mai multor clasificatori enumerați anteriori la descrierea algorimului. Aceștia sunt implementați în librăria SKLearn [Pedregosa et al., 2011]. Voi testa următorii clasificatori: SVC, MLP,KNN și Bayes Naiv. Va fi folosit același algoritm de extragere a descriptorilor SURF iar descriptorii vor fi u-SURF, pentru clusterizare voi utiliza K-Means cu 20 de clustere pe categorie folosind dataseturile Corel și caltech 101 și GHIM.

5.5.1 Tipuri clasificatori

La bază clasificatorii pot fi de mai multe tipuri:

- Liniari, clasificatorii liniari încearcă să creeze o funcție de decizie bazându-se pe combinații liniare a caracteristicilor, exemplu Bayes Naiv.
- SVM se bazează pe elementele de frontieră pentru a crea o funcție de delimitare a claselor, SVM este deasemenea bun la a ignora elementele rebele.
- KNN clasificatorii de tip KNN atribuie elementelor de test clasa vecinilor lui.
- Rețele neuronale sunt un model compus din mai multe nivele unde fiecare nivel primește input-ul aplică pe acesta o funcție și îl trimite mai depare, fiecare output obține o greutate în funcție de corectitudinea clasificării.
- Arbori de decizie, aceștia separă elementele în funcție de caracteristicile acestuia încercând să izoleze pe o ramură cat mai puține clase. Arborii de decizie nu au oferit o acuratețe bună pentru acest tip de clasificare. (25% 10 categorii)

5.5.2 Permformanță clasificatori

Pentru a alege cel mai potrivit clasificator pentru această problemă vom îi vom testa pe toți în aceleași condiții folosind detectorul SURF și descriptorii aceluiași algoritm cu orientarea presetata. Deasemenea vom folosi un număr de 20 clustere per categorie cu excepție pentru datasetul GHIM-10K unde vom folosi un număr dublu de clustere.

dataset	SVC	MLP	KNN	Bayes Naiv
Corel-10k 5 categorii	66.67%	62.67%	53.3%	52.0%
Corel-10k 10 categorii	51.34%	41.0%	31.33%	38.3%
Caltech-101 5 categorii	76.93%	73.62%	65.95%	62.63%
Caltech-256 5 categorii	68.74%	73.48%	66.85%	36.46%
GHIM-10k 3 categorii	93.55%	92.67%	89.78%	83.56%

Table 6: Acurateţe clasificatori

dataset	SVC	MLP	KNN	Bayes Naiv
Corel-10k 5 categorii	70.55s	68.85s	84.3s	71.8s
Corel-10k 10 categorii	207.42s	210.72s	205.3s	221.6s
Caltech-101 5 categorii	268.92s	255.93s	285.48s	266.79s
Caltech-256 5 categorii	2198.62s	1908.58s	3015.43s	2171.27s
GHIM-10k 3 categorii	1915.02s	1919.44s	1966.90s	1994.57s

Table 7: Timpul de antrenare clasificator

Pentru un clasificator este foarte important și timpul în care acesta reușește să clasifice o imagine folosind un model gata antrenat, un timp bun poate face posibilă utilizarea acestuia pentru aplicație real-time, de aceia voi compară și timpul în care clasificatorii reușesc să asigneze categoria tuturor imaginilor de test pe modelul deja antrenat, putem observa comparația în tabelul 8.

dataset	imagini test	SVC	MLP	KNN	Bayes Naiv
Corel-10k 5 categorii	150	0.67s	0.7s	1.25s	0.96s
Corel-10k 10 categorii	300	1.5s	1.29s	1.46s	1.37s
Caltech-101 5 categorii	91	1.5s	1.39s	1.53s	1.71s
Caltech-256 5 categorii	181	11.14s	7.57s	10.76s	13.79s
GHIM-10k 3 categorii	450	8.97s	10.4s	9.22s	9.19s

Table 8: Timpul de clasificare per implementare

5.5.3 SVC

Clasificatorul SVC care este o implementare a clasificatorului SVM din libaria SKLearn [Pedregosa et al., 2011].

Asa cum SVC sa descurcat foarte bine pentru clasificarea imaginilor şi acesta poate fi configurat modificând multipli parametri voi incerca săi modific pentru a obţine o acurateţe mai bună.

SVC ne pune la dispoziție mai multe kernel-uri precum:

- Linear kernel, acesta va incerca să creeze drepte in jurul punctelor calasificate in un anumit fel.
- RBF este un kernel bazat pe functii RBF. O functie ϕ este considerată RBF dacă satisface urmatoarea egalitate: $\phi(x, y) = \phi(||x, y||)$

Mai sunt și alte funcții kernel, dar pentru cazul BOVW acestia s-au dovedit a fi cei mai potriviți.

Diferenta intre kernele RBF si liniar o reprezinta forma frontierelor, RBF permitand curbarea acestora.

În practica diferența între aceste kernele se poate manifesta diferit, pentru cazul recunoașterii de imagini folosind această abordare putem observa urmatoarele diferente de accuratețe prezentate în figura 9. Pentru testarea clasificatorului s-au folosit descriptorii u-SURF iar clusterizarea sa efectuat în 500 de clustere.

Pentru seturi mai mici de date diferența este mică însa pe măsura ce folosim un dataset cu mai multe categorii cu kernelul liniar obținem o acuratețe mai mică. Diferențe de performanță majore nu s-au înregistrat, ambele înregistrând aproximativ același timp de antrenare.

dataset	RBF	LinearSVC
Corel 3 categorii	71.1%	62.2%
Caltech-101 3 categorii	94.2%	94.2%
Caltech-101 5 categorii	74.7%	69.23%

Table 9: Acuratete kernel

Kernelul RBF din SVC este foarte costumizabil astfel modificând parametrul C în o valoarea reală mai mare de valoare implicită 1.0 shimbăm toleranța la erori a clasificatorului, o valoare mai mare va face clasificatorul mai intolerant asupra unor clasificari greșite a valorilor de antrenament, însa o valoare prea mare poate duce la over fitting.

În urma testării cu valori diferite pentru un set de imagini obținut din corel și caltech-101 sau obținut rezultate similare pentru C=1.0 si C=1000.0 [tabel 10]. Se observa o diferenta la timpul de antrentrenare de aproximativ 10% in favoarea unei valori mai mici pentru

C=1.0.

dataset	C=1.0	C=1000.0
Corel 3 categorii	74.34%	73.34%
Caltech-101 3 categorii	94.23%	94.23%

Table 10: Acuratete RBF parametrul C

5.5.4 Concluzie clasificatori

După analiza acestor clasificatori putem ușor observa că cele mai bune rezultate le obține clasificatorul SVC şi clasificatorul MLP bazat pe perceptroni, astfel SVC se descurcă mai bine pe seturi cu puține imagini și puține categorii pe când clasificatorul MLP este robust și pe un număr mai mare de imagini, din punct de vedere a vitezei de clasificare dar si verificare aceștia sunt foarte similari.

5.6 Testare

Testarea este foarte dependentă de antrenare, astfel se urmeaza același proces de extragere a descriptorilor care apoi pentru fiecare sunt asiociați celui mai apropiat cluster. Distribuția descriptorilor este introdusă in clasificator care prezice o categorie pentru imaginea de test.

5.7 Concluzie

După cum am observat cele prezentate în această secțiune cea mai optimă implementare va folosi descriptori SURF utilizând algoritmul de detecție a unghiurilor bazat pe matricea Hessiană, acești descriptori vor folosi orientarea pe care o au în poză, acești descriptori vor fi clusterizati folosind algoritmul K-Means cu un număr de aproximativ 20 de clustere per categorie, modul în care imaginile sunt fromate raportându-ne la clusterele create va fi învățat de clasificatorul SVC sau MLP în funcție de numarul imaginilor, o imagine de test va urma același proces până la clasificator care îi va atribui o categorie.

6 Arhitectura Aplicatiei

Pentru crearea acestui model de clasificare a imaginilor am folosit limbajul Python [van Rossum, 1995], librariile: numpy și scipy pentru calculul pe matrici [Jones et al., 01], iar pentru reprezentarea grafica a statisticilor am folosit libraria matplotlib [Hunter, 2007]. Pentru lucrul pe imagini am folosit libraria OpenCV [Bradski, 2000] iar pentru algoritmii de clasificare si clusterizare am folosit libraria SKLearn [Pedregosa et al., 2011].

Mi-am propus să creez un model robust și ușor costumizabil, un model modular care să poată utiliza diverși algoritmi chiar din aceiași rulare așa cum o rulare poate dura pană la câteva ore să nu fie necesar implicația utilizatorului pentru un nou antrenament. Deasemenea un model care să poată fi salvat pentru o altă utilizare.

Din motivele expuse anterior am ajuns la crearea unei clase responsabile pentru pornirea unei antrenări si livrarea de detalii necesare acesteia. Această clasă va pregati dataseul, îl va împarți in imagini de antrenament și de test cu un ratio configurabil și va porni clasa BOVW care se va ocupa de antrenare și testarea modelului.

Clasa BOVW reprezintă nucleul modelului aceasta primește ca input setul de imagini, clasificatorul, algoritmul de clusterizare si algoritmul de detectare a descriptorilor ca apoi utilizând acestea să creaze un model capabil de clasificare a imaginilor, la fel aceasta clasa este responsabilă și de testarea modelului folosind setul de imagini de test.

În figura 7 avem diagrama UML care prezintă aplicația creată.

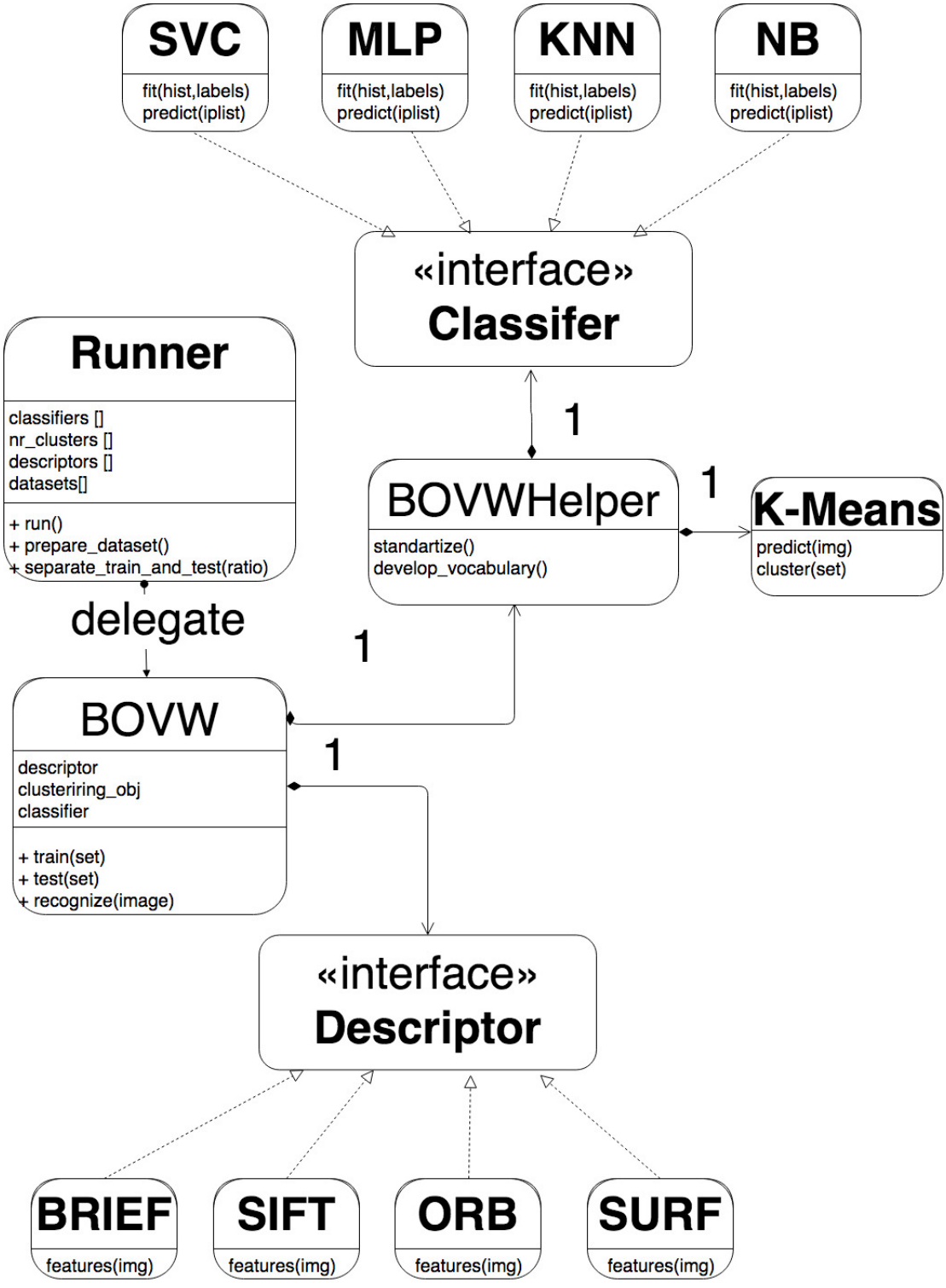


Figure 7: Arhitectura Aplicatiei

7 Analiza modelului obținut

Un model de clasificare a imaginilor poate avea diverse puncte slabe sau puternice, spre exemplu un model bazat pe rețele neuronale convoluționale va necesita un numar foarte mare de imagini de antrenament însă de obicei va oferi o acuratețe mai ridicată. Vom testa modelul obținut pentru a vedea care sunt vulnerabilitațile acestuia, astfel am testat modelul pentru următoarele cazuri posibile.

7.1 Numarul imaginilor de antrenament

Modelele de clasificare a imaginilor pot fi foarte diferite, de obicei un număr mai mare de date de antenament poate ajuta modelul să obțină o acuratețe mai bună, aceasta însă nu este general valabilă, de asemenea un numar mai mare de date de antrenament înseamnă un timp ridicat de antrenare, iar câteodată suntem nevoiți să dăm la schimb acuratețea pentru viteză. De asemenea obținerea unui set mare de date de antrenament poate fi dificilă și dorim un model care reușește să învețe de pe un set mic de date.

Am testat pe setul de imagini GHIM obținând statisticile care pot fi observate în figura 8, deasemenea am testat pe un set cu imagini mai mici si diferite, rezultatul poate fi observat în figura 9.

Pentru aceste statisitci am folosit clasificatorul bazat pe MLP și detectorul SURF care a avut limita pentru matricea Hessiană de 500.

Referitor la numarul imaginilor de antrenament putem concluziona faptul că mereu un numar mai mare de imagini va crea un model mai bun însă acest model oferă o acuratețe foarte buna chiar și pentru un numar foarte mic de imagini de antrenament, referitor la timpul de execuție putem observa ca un numar mai mare de imagini va creste timpul antrenament substanțial.

7.2 Diferența între numarul de imagini per categorie

Unele modele pot favoriza categoriile care conțin mai multe imagini de antrenament, aceasta se poate intampla și la modelul BOVW deoarece algoritmul K-Means are tendința

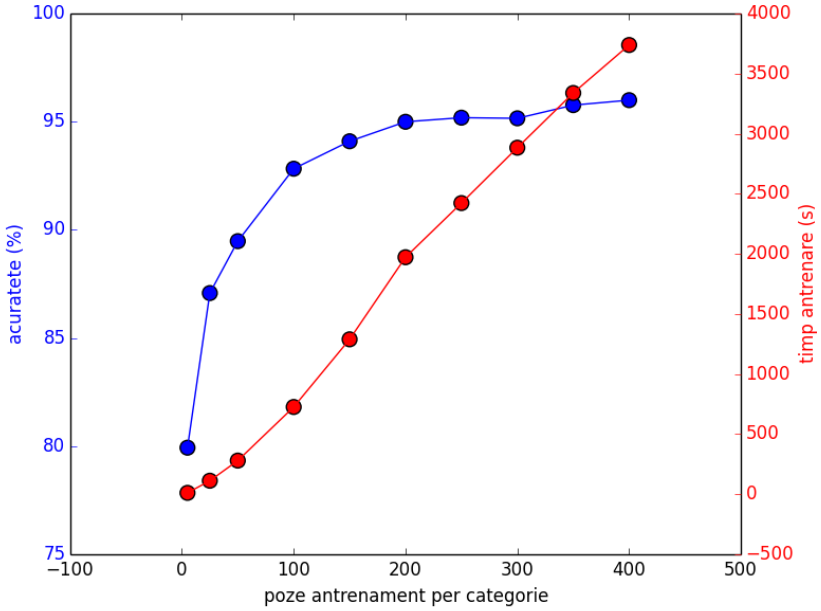


Figure 8: Acuratețea si timpul de antrenare în functie de numarului imaginilor GHIM

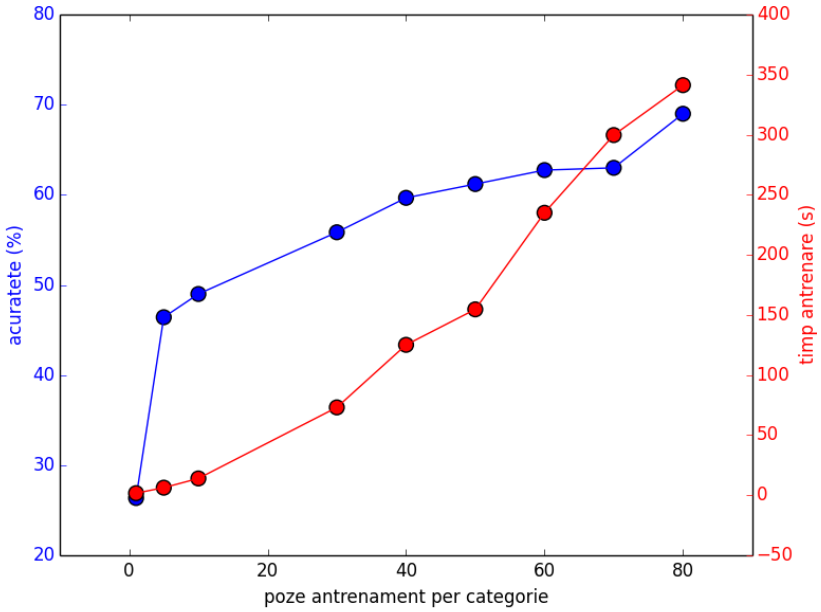
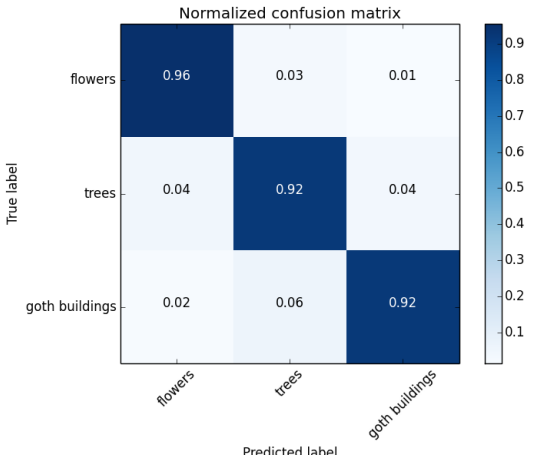
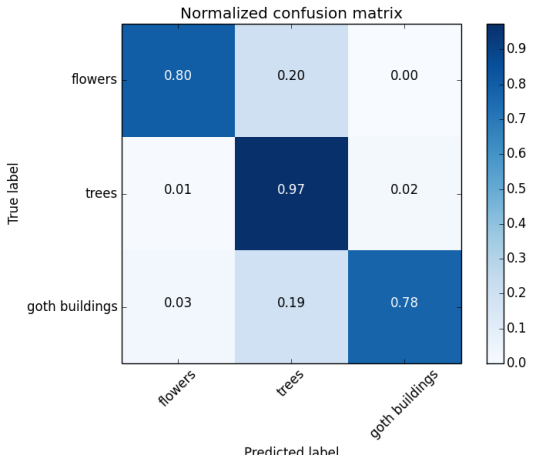


Figure 9: Acuratețea si timpul de antrenare în functie de numarului imaginilor Corel



(a) acelasi numar de imagini



(b) nr diferit de imagini

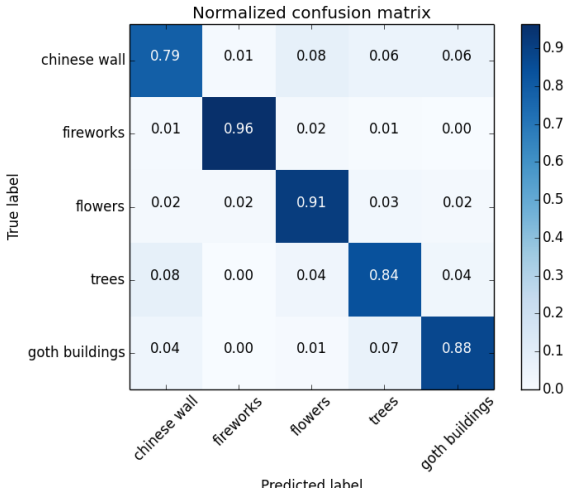
Figure 10: Acuratețea în funcție de numar de imagini per categorie, 3 categorii GHIM

să creeze clustere de marimi similare, astfel mai multe clustere cu descriptori specifici categoriei cu mai multe imagini sunt creați. Asa cum clasificatorul invață în funcție de procentajul distribuției descriptorilor, aceasta problemă ar putea afecta acuratețea întregului model fără să observat o mulțime de clasificari pentru categoria cu mai multe imagini.

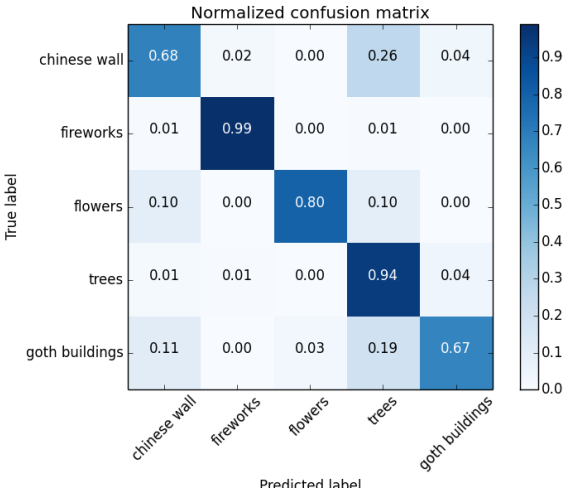
Am testat această posibilitate folosindun numar egal si variabil de imagini de antrenament modificând utilizînd setul de imagini GHIM.

Ca și acuratețe totală diferențele nu sunt foarte mari, pentru 3 categorii GHIM avem 93.22% acuratețe pentru numar egal de imagini per categorie și 92.96% pentru cazul în care avem un numar diferit de imagini per categorie, pentru 5 categorii cazul în care avem un numar diferit de imagini per categorie a subclasat cu 90.7% versus 87.67% celălalt caz.

Putem concluziona că modelul este puțin sensibil la numarul de imagini de antrenament per categorie, acesta însa nu va fi observabilă poate chiar benefica daca rația de distributie a imaginilor per categorie la antrenament este direct proporțională cu rația distribuției imaginilor de test.



(a) acelasi numar de imagini



(b) nr diferit de imagini

Figure 11: Acuratețea în funcție de numar de imagini per categorie, 5 categorii GHIM

7.3 Dimensiunile imaginilor

Imaginile pot avea marimi diferite, unii clasificatori pot fi foarte sensibili la această metrică, sau pot atinge un timp foarte mare de antrenare dacă imaginile sunt prea mari.

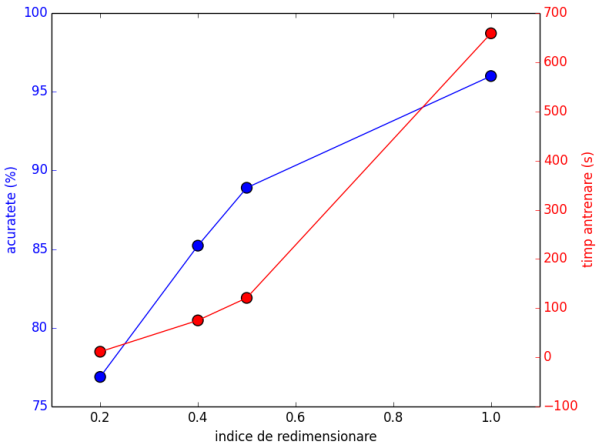
Modelul propus are o problemă cu dimensiunile imaginilor fiind incapabil să clasifice imagini prea mici (<40x40) deoarece nu se pot extrage mai mult de un descriptor care să nu se suprapună.

Folosind același set de imagini redimensionând imaginile am comparat rezultatele clasificatorului. Putem observa rezultatele în figura 12.

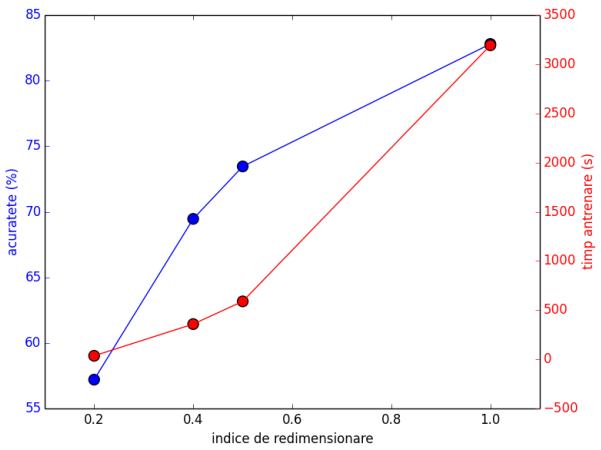
7.4 Dimensiuni diferite a imaginilor

Unele modele au nevoie de dimensiune fixă pentru a reuși clasificarea, acesta nu este cazul modelului BOVW însă utilizarea imaginilor cu dimensiuni variabile poate avea un anumit impact, pentru a testa modelul pentru această situație am folosit setul de imagini caltech-256 care cum am specificat anterior are un numar variabil de imagini per categorie cu dimensiuni variabile (200-500).

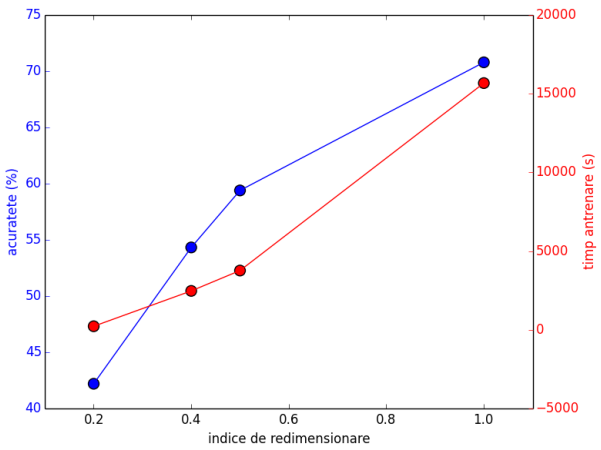
În figura 13 putem observa matricea de confuzie pe datasetul caltech-256, această rulare



(a) 3 categorii



(b) 5 categorii



(c) 10 categorii

Figure 12: Acuratețea si timpul de antrenare în functie de dimensiunea imaginilor, GHIM

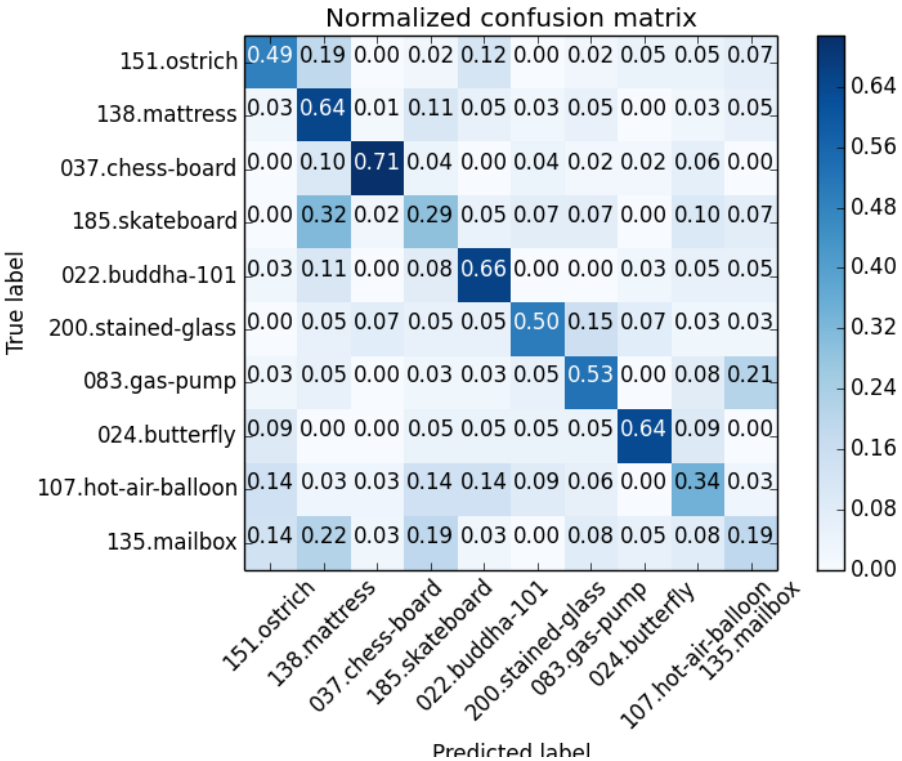


Figure 13: Acuratețea algoritmului pe imagini cu dimensiuni variabile

are acuratețea generală de 43.96% ceea ce comparativ cu acuratețea pe setul de date Corel (69.0%) este puțin, însă aceasta mai este influențată de modul în care sunt suprinse obiectele în acest set de imagini.

7.5 Testarea pe imagini din afara unui dataset

Pentru o testară mai detaliată și mai aproape de o situație reală am creat un set de date cu 10 categorii care conține în jur de 130 de imagini per categorie, imaginile sunt de dimensiuni diferite și poziționărea obiectului diferă, ele nu depășesc 400 de pixeli majoritatea dintre ele fiind redimensionate de la marimea lor originală. Setul a fost colecționat din imaginile returnate la cautarea categoriilor. Putem observa că modelul clasifică bine și acest tip de imagini. [figura 14]

Această antrenare a obținut acuratețea de 67.94% cu un timp de antrenare de 926.28s.

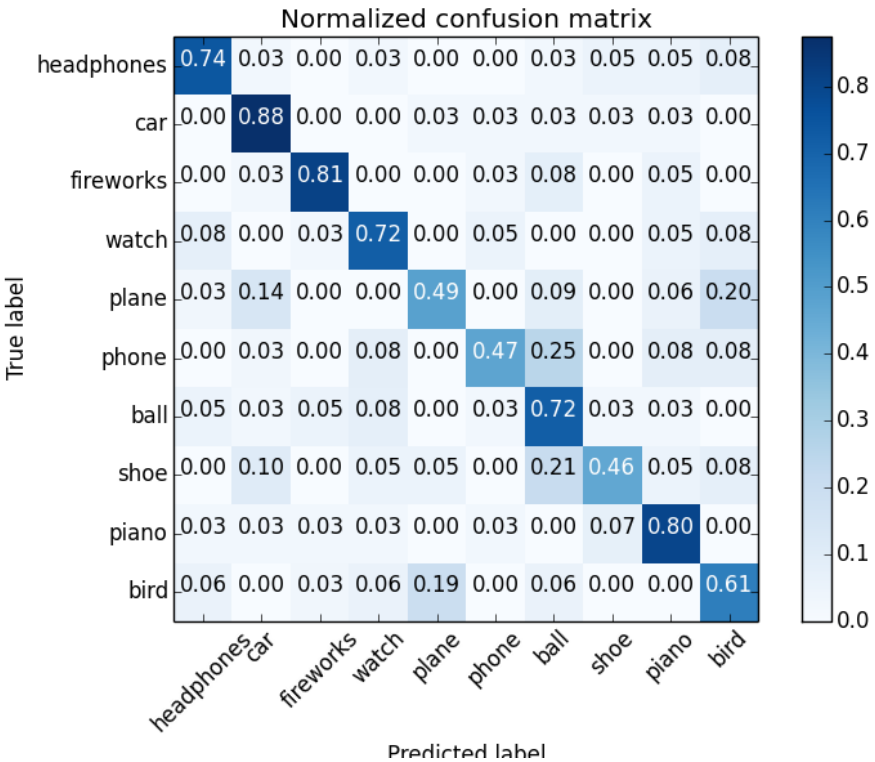


Figure 14: Acuratețea algoritmului pe imagini din surse aleatoare

8 Alți algoritmi de clasificare

Algoritmul BOVW este un algoritm vechi care are la baza idei simple dar destul de eficiente, începând cu 2012 acest model a început să fie înlocuit de rețelele neuronale care deși erau cunoscute deja de o perioada foarte mare au început abia atunci să fie folosite din 2 motive: puterea mare de procesare și volumul mare al datelor de antrenament.

8.1 Rețele Neuronale Convoluționale CNN

Retelele neuronale sunt niste modele matematice care rezolva probleme de optimizare, CNN au la baza neuronii. Un neuron primește o informație de intrare (sa zicem x) si aplica niste calcule pentru a obține un raspuns y, acesta este apoi trimis la un alt neuron sau la o funcție de activare pentru a produce un raspuns final. [Abadi et al., 2015]

Rețele neuronale sunt compuse din mai multe nivele acestea reprezintă urmatorea structură dupa neuron de a rețelelor neuronale. Acestea sunt compuse din un set de neuroni care sunt la aceiași distanța de input. Primul nivel este numit nivelul de intrare iar ultimul nivelul de iesire, restul nivelelor sunt nivele ascunse.

Nivelele convoluționale prezintă o filtrare a imagini de intrare, o convulție folosește un filtru pentru o anumita zonă a imaginii având la output un numar. Nivelele convoluționale sunt folosite foarte des in clasificarea imaginilor.

8.2 State of The Art

"State of the art" în domeniul clasificării imaginilor este atins de rețelele neuronaleconvoluționale, această evoluție sa remarcat în 2012 când la competiția "Large Scale Visual Recognition Challenge" [Russakovsky et al., 2015] o abordare bazată pe rețele neuronale a demonstrat o performanță impresionantă.[figura 15]

În ediția din 2012 a competiției folosirea rețelelor neuronale cu un rezultat atât de bun și cel mai important acest rezultat putea fi îmbunatațit, astfel următoarele ediții au constat în mare parte din competitori care foloseau rețelele neuronale, până în 2015 pe setul de date propus de ImageNet acuratețea a depașit clasificarea realizată de om. [von Zitzewitz,

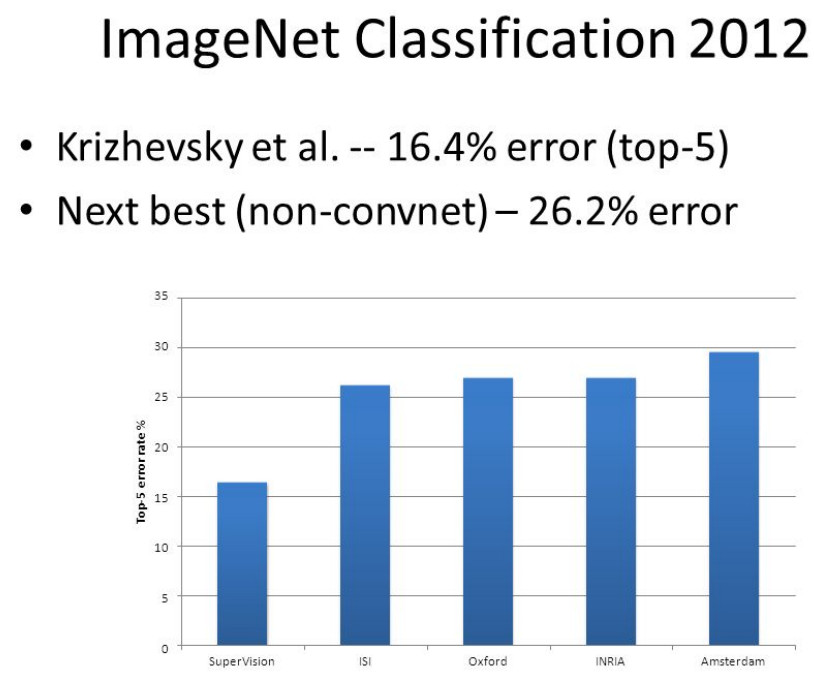


Figure 15: Cele mai bune rezultate ImageNet 2012 [Russakovsky et al., 2015]

2017] [figura 16]

8.3 Model CNN propus

Acestea însă sunt date antrenate pe calculatoare cu specificați extrem de bune, pentru o comparație corectă o să antrenez o rețea folosind aceleași date de antrenament, o să folosesc o arhitectură a rețelei propusă de Tatsuya Hatanaka [Hatanaka, 2017].

8.3.1 Arhitectura Rețelei

Rețeaua propusă este creată folosind librăria Keras [Chollet et al., 2015] folosind Tensorflow în backend [Abadi et al., 2015].

Rețeaua propusă este compusă din 4 nivele, 2 din acestea fiind convoluționale cu filtru de marimi de 32 și respectiv 64, și 2 sunt nivele simple unul cu 256 de neuroni iar celălalt este ultimul nivel care are numarul de neuronui egal cu numarul categoriilor, este aplicat un nivel "Dropout" cu parametru de 50% pentru combaterea învățării datelor de

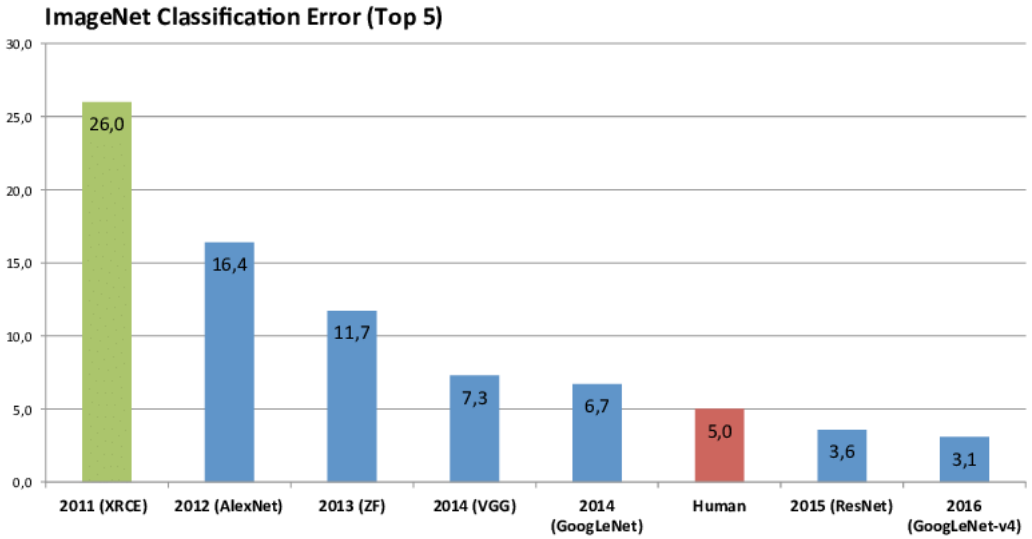


Figure 16: Cele mai bune rezultate ImageNet pe ani [Russakovsky et al., 2017]

antrenament, pentru nivele de activare s-a folosit funcția "relu"[3] și ultimul nivel folosește pentru activarea funcția "softmax" [4].

8.3.2 Detalii de antrenare

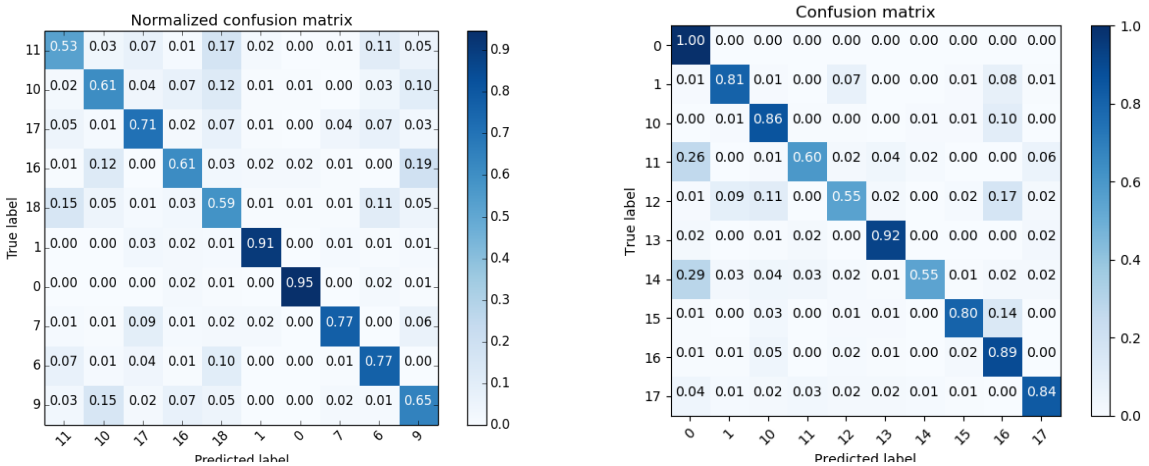
Setul de imagini Pentru compararea clasificatorilor în aceleași condiții cu modelul BOVW voi antrena acest model pe aceleași seturi de date: GHIM, Corel, Caltech.

Hardware -ul va fi diferit așa cum rețelele neuronale necesită o putere computațională superioară preferabil, astfel este preferat un GPU mai puternic, specificațiile calaculatorului:

- Procesor: Intel Core i7 2.5-3.5 GHz cu 4 nuclee, 8 nuclee virtuale
- GPU: Nvidia GTX 860m 2Gb GDDR5
- Memorie internă: 8 Gb 1600Hz
- Memorie nevolatilă: viteză citire/scriere 450Mb/s

[3] $f(x) = log(1+\exp\ x)$

[4] $\sigma(z)_j = \frac{e_j^z}{sum_{k=1}^{K} e_j^z}$



(a) BOVW (b) CNN

Figure 17: Matricea de confuzie pentru GHIM 10 categorii (BOVW vs CNN)

Această mașina a fost utilizată și pentru modelul BOVW înregistrând o viteză mai mare însă nesemnificativă față mașina folosită pentru statisticile anterioare de aproximativ 5-10%.

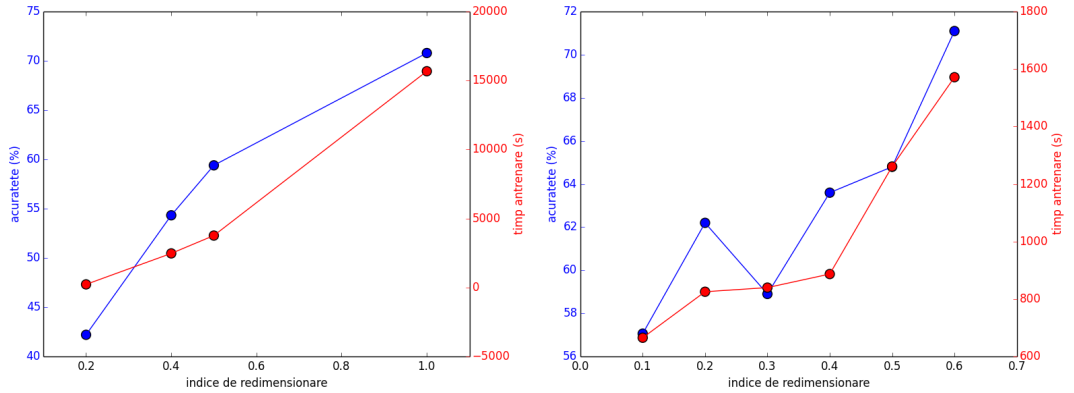
Din cauza limitărilor hardware (memorie GPU) dimensiunea maxima a imaginilor este de 180x180.

8.3.3 Comparație acuratețe

Am testat pe setul GHIM astfel putem observa în figura 17, putem observa că tendința rețelei e să învețe unele categorii mai bine. Acuratețea pentru această rulare este de 70.08% pentru modelul BOVW și 71.36% pentru CNN rețeaua însă a învățat semnificativ mai rapid 839.7s comparativ cu 15679.61s a modelului BOVW.

8.3.4 Mărimea imaginii

La fel pe setul de date GHIM am testat toleranța rețelei la dimensiunile imaginilor, în figura 18 putem observa rezultatele, astfel putem observa că rețeaua este mai tolerantă la schimbarile de mărime înregistrând fructuații mai mici la acuratețe la fel și o creștere semnificativ mai mică a vitezei de antrenare.



(a) BOVW (b) CNN

Figure 18: Acuratețea și timpul de antrenare dimensiunea imaginii (BOVW vs CNN)

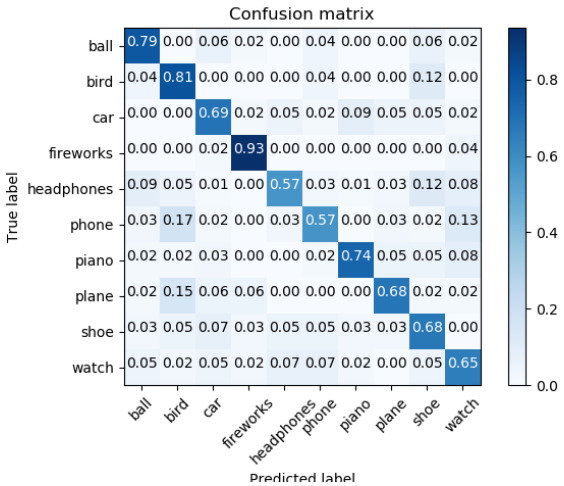


Figure 19: Matrice confuzie CNN pentru imagini de mărimi diferite

8.3.5 Mărimi diferite a imaginilor

Rețeaua neuronală prezentată nu acceptă imagini nepătratice de dimensiuni diferite între ele, astfel abordarea va fi redimensionare acesteia la o marime stabilită deoarece decuparea până la o valoare pătratică împactează negativ acuratețea. Am folosit datasetul colectat de mine, în figura 19 putem observa matricea de confuzie.

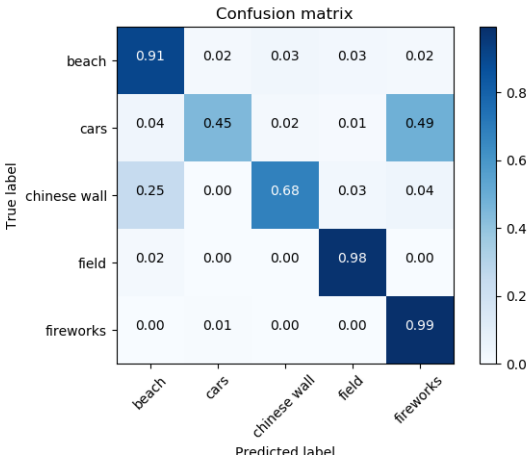


Figure 20: Matrice confuzie CNN număr variabil de imagini de antrenament

Acuratețea rețelei a fost de 69.6% antreându-se în 647.12s, astfel putem concluziona că rețeaua învață bine inclusiv din imagini redimensionate cu ratio lungime-lățime diferit.

8.3.6 Număr variabil de imagini per categorie

Deja știm ca modelul BOVW este dependent de această variabilă, aceasta însă poate fi benefică dacă imaginile de test au aceiași rație ca imaginile de antrenament, la CNN am observat deja că învățarea unor categorii poate fi favorizată, dorim să aflăm dacă o distribuire inegală a imaginilor per categorie va afecta acuratețea [Figura 20].

Acuratețea nu este impactată de un numar diferit de imagini de antrenament, deși putem observa că unele categorii nu au fost învățate foarte, aceasta sa putut observa și atunci când numarul de imagini de antrenament per categorie este același, este doar o problemă când rețeaua nu reușește să învețe o categorie din diverse motive.

9 Concluzii

BOVW este un model matur de clasificarea a imaginilor, oferă o acuratețe decentă indiferent de dimensiunile imaginilor, distribuția pe categorii, dimensiunile variabile a imaginilor.

Punctele puternice ale acestui model sunt: rapiditatea acestuia, pe un numar mic de categorii poate fi antrenat aproape instant indiferent de hardware iar acuratețea va fi destul de bună.

Modelul BOVW este destul de departe de "State of The Art" când vine vorba de acuratețe aceasta fiind cauzată prepoderent de faptul că modelul nu poate profita de un volum mare de date, înregistrăm o creștere semnifivă a timpului de antrenare atunci când numarul imaginilor este mărit fără ca acuratețea să se marească semnificativ la fel și un numar mare de categorii va impacta negativ acuratețea.

Folosind metode clasice de clasificare a datelor modelul BOVW poate fi ușor învațat de persoane cu cunoștințe de bază de învațare automată. Astfel modelul este recomandat pentru clasificarea unor imagini pentru cazul în care setul de antrenament conține puține date de antrenament, puține categorii iar hardware-ul este limitat, deasemenea un model de tip BOVW este ușor de configurat așa cum acesta nu are foarte mulți parametri.

Când acuratețea este principala țintă o abordare de tip CNN este recomandată, aceasta însă va necesita o configurare mai detaliată pentru setul de date, un hardware performant și un set mare de date de antrenament.

Pentru o cale de mijloc se poate utliza BOVW cu clasificatorul MLP acesta va oferi o acuratețe mai bună pentru un număr mai mare de categorii însă diferența față de SVM nu este semnificativă.

9.1 Direcții viitoare

Modelul BOVW folosește foarte puțin din informația oferită de o imagine, acesta nu folosește culorile, forma obiectelor în imagine, etc. Utilizarea unui model care să-l combine cu alte metode utilizând ponderi pentru fiecare ar putea funrniza informații de o calitate mai înaltă.

BOVW poate fi extins și spre alte direcții spre exemplu acesta ar putea utiliza descriptorii mai întâlniți pentru a face detecția obiectelor, chiar dacă descriptorii nu conțin informații cu privire la localizarea lor este posibilă o asemenea abordare, o altă direcție este recrearea imaginilor din descriptori Hiroharu Kato și Tatsuya Harada [Kato and Harada, 2014] au reușit să obțină rezultate impresionante folosind această abordare.

Bibliografie

(2014). *The OpenCV Reference Manual*. Itseez, 2.4.9.0 edition.

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Agrawal M., Konolige K., B. M. (2008). Censure: Center surround extremas for realtime feature detection and matching.

Berthier Ribeiro-Neto, R. B.-Y. (1999). Modern information retrieval.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Chollet, F. et al. (2015). Keras. https://keras.io.

Damian Eads, D. H. and Rosten, E. (2013). Fusing points and lines for high performance tracking.

Davidson, M. W. and Abramowitz, M. (2002). Molecular expressions microscopy primer: Digital image processing – difference of gaussians edge enhancement algorithm.

Ethan Rublee, Vincent Rabaud, K. K. and Bradski, G. R. (2011). Orb: An efficient alternative to sift or surf.

Fei-Fei Li, M. A. and Ranzato, M. A. (2003). Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories.

G.Lowe, D. (2004). Distinctive image features from scale-invariant keypoints.

Griffin, G. Holub, A. P. P. (2006). The caltech 256.

Guang-Hai Liu, J.-Y. Y. (2013). Content-based image retrieval using color difference histogram, pattern recognition.

Guang-Hai Liu, J.-Y. Y. (2015). Content-based image retrieval using computational visual attention model, pattern recognition.

Harris, C. and Stephens, M. (1988). A combined corner and edge detector.

Harris, Z. (1954). Distributional structure.

Hatanaka, T. (2017). CNN Image Classifier.

Herbert Bay, T. T. and Gool, L. V. (2006). Surf: Speeded up robust features.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95.

Jones, E., Oliphant, T., Peterson, P., et al. (2001–). SciPy: Open source scientific tools for Python. [Online; accessed <today>].

Kato, H. and Harada, T. (2014). Image reconstruction from bag-of-visual-words. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 955–962.

Michael Calonder, Vincent Lepetit, C. S. and Fua, P. (2010). Brief: Binary robust independent elementary features.

Moravec, H. (1980). Obstacle avoidance and navigation in the real world by a seeing robot rover.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2017). Imagenet lsvrc challenge.

van Rossum, G. (1995). Python tutorial. Technical Report CS-R9526, Centrum voor Wiskunde en Informatica (CWI), Amsterdam.

von Zitzewitz, G. (2017). Survey of neural networks in autonomous driving.