**UNIVERSITATEA DIN BUCUREȘTI**

**FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ**

**DOMENIUL DE LICENȚĂ INFORMATICĂ**

**LUCRARE DE LICENȚĂ**

**COORDONATOR ȘTIINȚIFIC**

**CONF. DR. IONESCU RADU TUDOR**

**STUDENT**

**ENACHE ALEXANDRU-MĂDĂLIN**

**BUCUREȘTI**

**2018**

**UNIVERSITATEA DIN BUCUREȘTI**

**FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ**

**DOMENIUL DE LICENȚĂ INFORMATICĂ**

**LUCRARE DE LICENȚĂ**

**APLICAȚIE WEB PENTRU CĂUTAREA INFORMAȚIILOR DESPRE ACTORI PE BAZĂ DE RECUNOAȘTERE FACIALĂ**

**COORDONATOR ȘTIINȚIFIC**

**CONF. DR. IONESCU RADU TUDOR**

**STUDENT**

**ENACHE ALEXANDRU-MĂDĂLIN**

**BUCUREȘTI**

**2018**

**Cuprins**

[**1. Introducere**](#_n8pkrgyholim) **4**

[**2. Recunoașterea facială din imagini**](#_9yyvqphggk2p) **6**

[2.1. Concepte teoretice](#_2drgiqtebd1f) 7

[2.1.1. Rețele neuronale convoluționale](#_yxkmxbbcvj0z) 7

[2.1.2. Histograma gradienților orientați (HOG)](#_tukko73m3z1) 14

[2.1.3. Mașini cu suport vectorial (SVM)](#_fd7hk1lx5yf8) 16

[2.1.4. Metoda celor mai apropiați k vecini (KNN)](#_4ddvwmv502fo) 20

[2.2. Metode recente pentru recunoașterea facială din imagini](#_aowc2gsq7gjy) 22

[2.2.1. FaceNet](#_43h2w692o1gp) 22

[2.2.2. ResNet](#_ecas5j51xu6h) 23

[2.3. Algoritmul de recunoaștere facială implementat](#_8kw8gqs5plnu) 24

[2.4. Rezultate și experimente](#_oyz77oyeqcty) 26

[**3. Tehnologii folosite în dezvoltarea aplicației web**](#_gaduneybo7lr) **29**

[3.1. Java](#_kevqiicxksvv) 29

[3.2. Maven](#_hwe31px6iv33) 30

[3.3. Spring](#_13a984pqlpb2) 31

[3.4. Oracle](#_iunxnz4vfq20) 32

[3.5. MyBatis](#_ioo3tdc7wnp8) 33

[3.6. HTML](#_f8znkz540fm3) 33

[3.7. CSS](#_6p8z7jgt2fh4) 34

[3.8. JavaScript + JQuery + Ajax](#_wlxxkofzxusl) 34

[**4. Descrierea aplicației**](#_7qc261iczexn) **37**

4.1. [Diagrama relațională a bazei de date](#_fcupusvb3yum) 38

4.2. [Diagrama de activități a aplicației](#_mymnpljrl73v) 40

[**5. Concluzii și dezvoltări ulterioare**](#_c5s7794lc4gq) **47**

[**6. Bibliografie**](#_5f2gtbyzddgm) **49**

**Capitolul 1**

# **1. Introducere**

Rețelele neuronale convoluționale au avut un succes imens în ultima jumătate de deceniu, totul pornind de la rețeaua propusă în 2012 de Alex Krizhevsky, [1]. Deși prima versiune a rețelelor convoluționale a apărut înaintea anilor 2000, momentul respectiv a adus în atenție potențialul acestora și a făcut marile companii să investească în acest domeniu. Creșterea a fost una foarte rapidă, lucru văzut prin reușitele obținute de aceste rețele an de an în domeniul recunoașterii în imagini.

În ciuda faptului că inteligența artificială s-a dezvoltat într-un pas foarte alert în ultimii ani, nu foarte multe persoane au adăugat componente din acest domeniu în aplicațiile web. În momentul de față dezvoltatorilor le sunt puse la dispoziție multe surse de informații și modalități de a utiliza aceste rețele.

Motivația acestei lucrări este de a afla ce acuratețe poate avea un model creat de noi și de a le da posibilitatea unor persoane să folosească această funcționalitate de recunoaștere facială pentru aflarea informațiilor despre actori. Pentru a realiza acest lucru vom folosi o rețea neuronală convoluțională prin care vom clasifica un set de date creat specific pentru această aplicație, el conținând un număr de aproximativ 10600 imagini grupate în 100 categorii, reprezentând cei mai faimoși actori din anul 2017 în urma căutărilor pe platforma IMDB.

Astfel, în capitolul 2 vom prezenta conceptele teoretice care stau la baza realizării recunoașterii faciale din imagini, precizând principalele caracteristici ale rețelelor convoluționale. În continuare vom face o descriere a descriptorului HOG, acesta fiind folosit pentru a arăta diferențele de acuratețe între rețele și alte modele de recunoaștere facială, iar apoi vom prezenta 2 dintre cei mai folosiți clasificatori folosiți în inteligența artificială.

După aceea vom arăta particularitățile propuse de Google și Microsoft în două dintre cele mai recente rețele apărute în acest domeniu: FaceNet și ResNet. Capitolul se va încheia cu prezentarea algoritmului folosit pentru recunoașterea facială în aplicația web și enunțarea rezultatelor obținute pe setul de date nou folosind rețeaua preantrenată și descriptorul HOG: ResNet & SVM, ResNet & KNN, HOG & SVM și HOG & KNN.

Capitolul 3 ne oferă o descriere a tehnologiilor folosite în realizarea aplicației web. Pe lângă enumerarea lor, ne sunt prezentate și particularități ale acestora, inclusiv scopul pentru care au fost folosite în aplicație.

Capitolul 4 este dedicat descrierii aplicației. Aici vom discuta cum funcționează aplicația, vom arăta ce poate să facă fiecare utilizator în funcție de rolul său, măsurile luate în vederea protecției parolelor împotriva unor atacatori sunt prezentate, precum și modul în care este verificat dacă un utilizator are dreptul să execute o anumită acțiune.

Capitolul 5 ne prezintă concluziile în vederea numărului de poze necesare pentru fiecare actor astfel încât să obținem o acuratețe bună, dar și o listă de posibile dezvoltări ulterioare.

**Capitolul 2**

# **2. Recunoașterea facială din imagini**

## **2.1. Concepte teoretice**

### **2.1.1. Rețele neuronale convoluționale**

Rețelele neuronale convoluționale sunt o categorie a rețelelor neuronale care s-au dovedit să aibă un succes foarte mare în clasificarea, recunoașterea obiectelor și identificarea persoanelor din imagini după fețe. Față de rețelele normale, cele convoluționale au avut un rezultate mai bune datorită faptului că spațialitatea imaginii este păstrată prin folosirea filtrelor în straturile convoluționale.

Rețelele convoluționale sunt foarte asemănătoare cu cele neuronale ordinare prin faptul că sunt au la bază neuroni în care sunt reținute informații. Un neuron este o unitate de stocare, acesta actualizându-se prin înmulțirea unor ponderi cu datele de intrare și adunarea rezultatului cu o valoare b. Formula arată în felul următor:

y = (2.1)

De preferat, această operație ar trebui să fie urmată de o funcție neliniară. Acest lucru este dorit datorită faptului că dacă am înlănțui mai multe funcții liniare precum cea din formula (2.1), tot acest lanț ar putea fi înlocuit cu o singură funcție liniară. Fiecare funcție liniară trebuie urmată de una neliniară pentru a evita cazul precizat. În final formula va arată astfel:

y = f () (2.2)

De-a lungul anilor au fost folosite mai multe funcții neliniare, denumite și funcții de activare. Prima funcție pe care o vom prezenta este funcția **Sigmoid**, notată . Aceasta are următoarea formulă matematică și distribuție în jurul punctului 0.

(2.3)

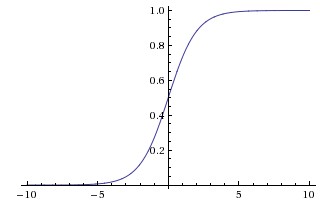


Figura 2.1: Reprezentarea grafică a funcției sigmoid

Sursă: CS231n [17]

Funcția sigmoid a fost folosită în trecut datorită faptului că are codomeniul în intervalul [0, 1]. În practică, această funcție este folosită foarte rar deoarece are câteva mari dezavantaje. Primul problemă este aceea că gradientul în jurul punctelor 0 și 1 este aproape 0. Astfel, în propagarea înapoi a rețelei, semnalul care v-a trece prin neuron va fi foarte mic și nu vor apărea schimbări prea mari în ponderile acestuia, rețeaua învățând foarte greu. Acest lucru poate fi evitat parțial prin inițializarea foarte atentă a ponderilor neuronilor care folosesc sigmoid. [17]

Un alt dezavantaj este acela ca rezultatele funcției sigmoid nu sunt centrate în 0. Acest lucru nu este dorit datorită faptului că dacă toate datele de intrare au același semn, gradienții în propagarea înapoi vor avea toți același semn, [17]. În plus, funcția exponențială este scumpă din punct de vedere computațional.

Funcția **Tangentă** este o funcție de activare care restrânge valorile în intervalul [-1, 1]. Tangenta este computațional scumpă, precum funcția sigmoid în (2.3), dar valorile rezultate sunt centrate în 0. De aceea, tangenta este mereu preferată față de sigmoid în practică. Un lucru interesat care poate fi observat este că tangenta poate fi formată dintr-o funcție sigmoid:

tanh(x) = 2 \* (2.4)

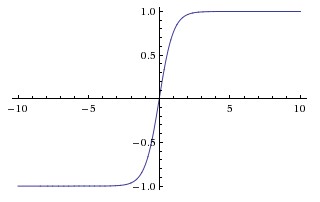


Figura 2.2: Reprezentarea grafică a funcției tangentă

Sursă: CS231n [17]

În continuare vom prezenta funcția de activare **ReLU**. Alex Krizhevsky a fost propus în [1] să folosim această funcție în rețelele neuronale în detrimentul tangentei, acuratețea rămânând aceeași în ciuda faptului că ReLU este mult mai rapidă. Formula și distribuția funcției apar în continuare:

f(x) = max(0, x) (2.5)



Figura 2.3: Reprezentarea grafică a funcției ReLU

Sursă: CS231n [17]

O mare problemă care poate apărea în antrenarea rețelelor neuronale este cazul în care rețeaua învață să clasifice foarte bine setul de date oferit la intrare, dar nu se descurcă bine pe unul nou. Pentru a evita acest lucru ne este propusă o tehnică foarte eficientă și simplă de ‘adormire’ a neuronilor în [2]. Aceasta ne spune ca în momentul antrenării să ‘adormim’ un neuron cu o probabilitate p. Astfel, la fiecare pas de propagare înainte, doar o parte din neuroni vor participa în acea etapă și doar ei vor fi actualizați la propagarea înapoi, restul fiind setați să aibă valoarea 0 în acea rundă.

Mai mulți neuroni din același nivel care conțin o funcție de activare formează un strat. O rețea neuronală este compusă din mai multe astfel de straturi complet conectate între ele și un strat final care ne va returna predicția.

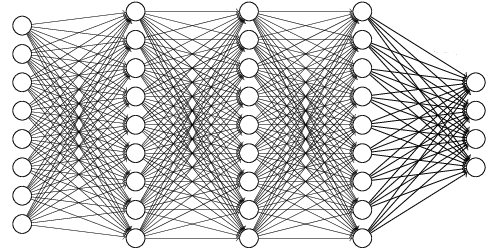


Figura 2.4: Reprezentarea grafică a unei rețele neuronale

Sursă: Google [19]

Ultimul strat al unei rețele este unul de decizie. Aici va avea loc prezicerea asupra cărei clase aparțin datele de intrare. Cea mai folosită funcție pentru a efectua acest lucru este **Softmax**. Ea primește la intrare un vector de dimensiune n și returnează un vector de aceeași dimensiune cu valori între 0 și 1, având suma totală egală cu 1 [15]. Astfel, valoarea vectorul de ieșire poate fi gândită ca ‘probabilitatea ca un element să aparțină acelei clase’. Funcția Softmax are următoarea formulă:

(2.6)

În final vom discuta despre funcția de cost. Această funcție ne spune cât de bine am aproximat rezultatul final. Astfel, putem vedea cât de aproape am fost de răspunsul corect și putem modifica valorile neuronilor. Rezultatul acestei funcții este numită eroarea acelui pas. Cea mai cunoscută funcție de cost este aceea în care calculăm pătratul diferenței dintre răspunsul corect și răspunsul prezis de noi pentru fiecare clasă, iar apoi le adunăm pe toate pentru a obține eroarea totală [15]. În funcție de tipul problemei pe care încercăm sa o rezolvăm pot fi folosite diferite funcții de cost.

Valoarea funcției de cost este influențată de ponderile neuronilor. Astfel, pentru minimizarea erorii ponderile sunt modificate în pasul de propagarea înapoi. Aceste modificări au loc folosind o funcție de optimizare. În general, această funcție calculează gradientul unui neuron, adică derivata funcției de cost în raport cu ponderile, iar apoi ponderile sunt modificate în direcția opusă acestui gradient calculat. [15]

În acest punct avem toate componentele necesare pentru a construi o rețea neuronală ordinară. Toate aceste lucruri vor apărea și în rețelele convoluționale.

Principala diferență între o rețea convoluțională și una ordinară este faptul că știm că la intrare vom primi o imagine. Fiecare imagine poate fi gândită ca o matrice de numere cuprinse între 0 și 255. Un canal este termenul folosit prin care ne referim la o componentă a imaginii. O imagine color are 3 canale (roșu, verde și albastru), putând fi exprimată ca 3 matrici suprapuse, câte una pentru fiecare canal. Imaginile alb-negru au un singur canal, deci vor fi reprezentate ca o matrice cu componentele între 0 și 255, de la alb la negru cu cât scade mai mult valoarea elementului.

O rețea convoluțională este formată dintr-o înșirare de straturi. Totuși, spre deosebire de rețelele ordinare care folosesc doar straturi complet conectate între ele (fig. 2.4), cele convoluționale mai introduc 2 tipuri de straturi: convoluționale și de unificare.

Stratul convoluțional este cea mai importantă componentă a unei rețele convoluționale. Convoluțiile au proprietatea de a păstra legătura spațială dintre pixeli, fiind folosite pătrate de dimensiune mici, numite filtre. Filtrul va porni din partea stânga-sus și va parcurge imaginea de la stânga la dreapta și apoi de sus în jos. La fiecare pas, este efectuat produsul dintre filtru și submatricea acoperită de filtru, iar răspunsul este reprezentat de suma componentelor matricei rezultate.

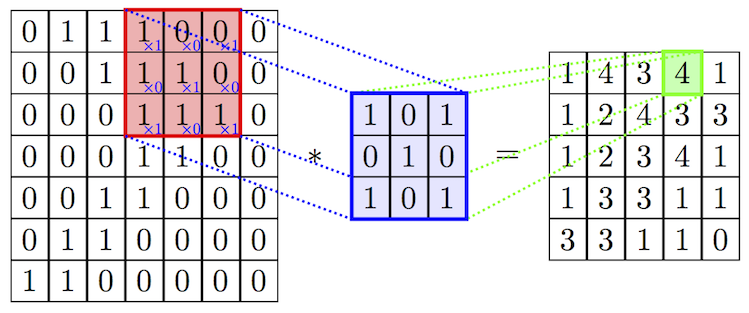


Figura 2.5: Reprezentarea grafică a unui pas de convoluției

Sursă: Google [19]

Pentru fiecare strat convoluțional trebuie să decidem ce dimensiune va avea filtru, peste câți pixeli sa săra la fiecare pas și de unde să înceapă. Conform [3], filtrele ar trebui să aibă dimensiunea de 3x3, aceasta fiind cea mai mică dimensiune care păstrează noțiunile de centru, sus-jos și stânga-dreapta. Filtrele de dimensiuni mari ar trebui să fie schimbate cu mai multe filtre de dimensiune 3x3 consecutive. De exemplu, putem înlocui un filtru de dimensiune 7x7 cu 3 filtre de dimensiune 3x3 cu deplasarea 1, iar faptul că sunt folosite mai multe filtre va face funcția de decizie să fie mai discriminatorie. Se observă că un strat convoluțional 7x7 are același câmpul vizual precum cel al 3 convoluții consecutive 3x3. În final se poate observa o micșorare al numărului parametrilor folosiți. Un strat de dimensiune 7x7 ar avea nevoie de 7 x 7 x C x C parametri, C reprezentând numărul de canale, pe când cele 3 straturi 3x3 ar avea nevoie de doar 3 x (3 x 3 x C x C). Astfel, stratul 7x7 folosește cu 81% mai mulți parametri decât cele 3 de dimensiune 3x3, [3].

Deplasarea reprezintă numărul de pixeli peste care va sări filtrul după fiecare pas. Când deplasarea este de dimensiune 1 filtrul va merge câte un pixel, iar când aceasta este 2 acesta va sări peste un pixel. Creșterea pasului de deplasare va genera o matrice de trăsături de dimensiune mai mică.

Uneori ne va fi nevoie să controlăm dimensiunea matricei de trăsături rezultată, iar acest lucru se poate realiza prin adăugarea de zerouri în jurul imaginii înaintea realizării unei convoluții.

Când construim o rețea convoluțională trebuie sa avem grijă la dimensiunea ferestrei care va rezulta după un strat convoluțional. Dimensiunea matricei de trăsături rezultate în urma stratului de convoluție poate fi calculată astfel:

D = (2.7)

, unde D reprezintă înălțimea/lățimea matricei de trăsături, W înălțimea/lățimea ferestrei de intrare, F înălțimea/lățimea filtrului, 2 \* P numărul de linii/coloane de zerouri adăugat, iar S dimensiunea pasului de deplasare.

Stratul de unificare este folosit pentru a reduce dimensiunea matricei de intrare, dar păstrând cea mai importantă informație. Conform [4], cea mai folosită metodă de unificare este cea a maximului. Prin acest procedeu, matricea de trăsături este împărțită în zone și este ales maximul din fiecare petic.

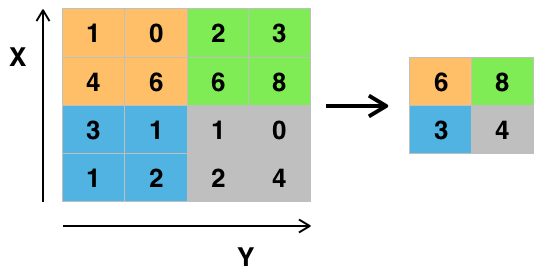


Figura 2.6: Strat de unificare prin extragerea maximului

Sursă: Wikipedia [15]

Există și alte metode de unificare, precum calcularea mediei sau a sumei. Când folosim straturi de unificare trebuie să avem grijă la dimensiunea matricei rezultate. În [4] ne este dată formula prin care putem calcula dimensiunea matricei finale, aceasta fiind aceeași indiferent de tipul de unificare:

D = (2.8)

, unde D reprezintă înălțimea/lățimea matricei de trăsături, W înălțimea/lățimea ferestrei de intrare, F înălțimea/lățimea filtrului, iar S dimensiunea pasului de deplasare.

Se poate observa ca formula (2.8) este un caz particular ar celei (2.7). Acest lucru are sens deoarece în straturile de unificare nu sunt adăugare zerouri pentru a controla dimensiunea matricei de trăsături.

O rețea neuronală convoluțională va fi compusă din mai multe straturi convoluționale, printre care apar straturi de unificare și apoi unul sau mai multe straturi complet conectate între ele, încheiate cu un strat de Softmax pentru a ne rezulta probabilitățile de apartenență pentru fiecare clasă.

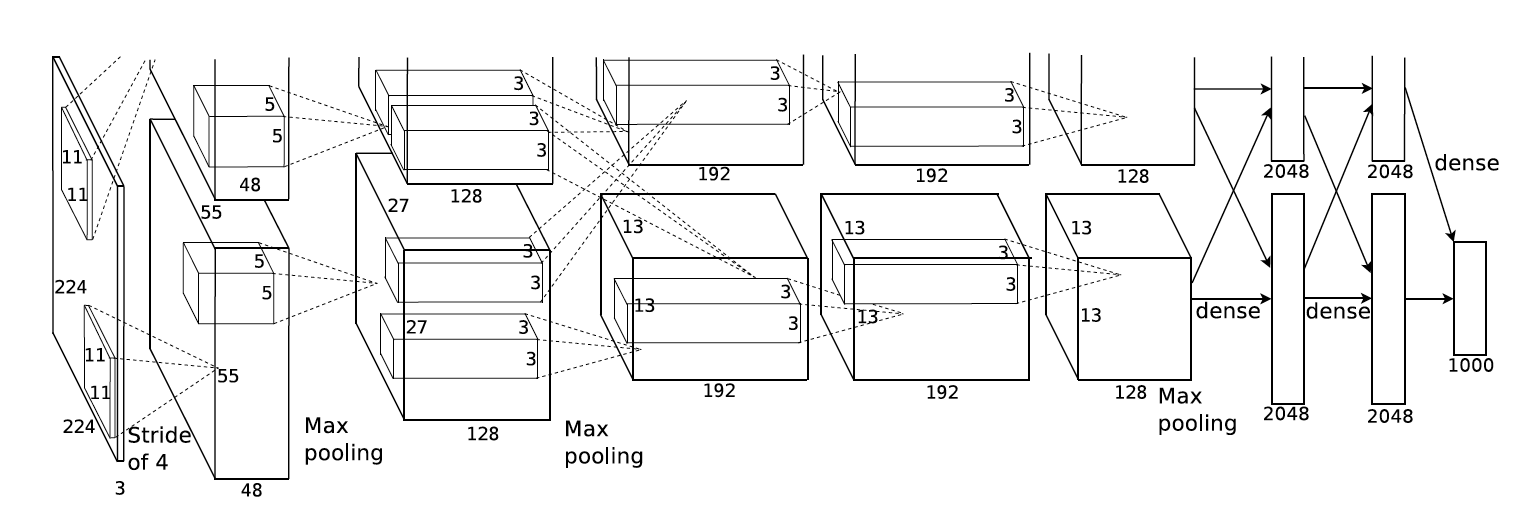


Figura 2.7: Rețeaua AlexNet, propusă de Alex Krizhevsky

Sursă: [1]

Alex Krizhevsky, Ilya Sutskever și Geoffrey Hinton au șocat lumea în 2012, prezentând o rețea neuronală convoluțională care a câștigat concursul ILSVRC și care a obținut o eroare de doar 15.4% pe primele 5 predicții [15]. Acesta a fost primul model care a avut un astfel de succes pe setul de date foarte dificil ImageNet, precedenta acuratețe fiind de 26.2%. Pornind de la ideile prezentate de cei 3 în [1], an de an au apărut rețele care au bătut recordul precedent. În 2015, Microsoft a ajuns la o eroare pe primele 5 predicții de 3.56% prin rețeaua ResNet, [5].

Rețelele convoluționale pot fi folosite la multe lucruri. De exemplu, în domeniul construcției de mașini care se conduc singure rețele convoluționale sunt foarte importante, fiind folosite în detectarea celorlalte mașini și a liniilor de pe stradă, precum și a semnelor de circulație.

### **2.1.2. Histograma gradienților orientați (HOG)**

Histograma gradienților orientați a fost folosită pentru a extrage trăsăturile unui obiect din imagini. Spre deosebire de o rețea neuronală convoluțională care deprinde de una singură cum să extragă trăsături, această metodă are un algoritm predefinit pentru a calcula aceste date de ieșire. Motivul pentru care, în general, rețelele convoluționale funcționează mai bine decât descriptorul HOG este acela că rețelele învață cum trebuie să extragă trăsăturile pentru a obține o acuratețe cât mai bună pe un anumit set de date, pe când descriptorul HOG are același mod de extragere a trăsăturilor. De aceea când avem un set de date cu multe imagini este de preferat să antrenăm o rețea deoarece aceasta va avea o acuratețe mai bună aproape de fiecare dată. Totuși, dacă avem un set de date mic și nu avem posibilitatea să creștem numărul de imagini, este posibil ca metoda histogramei gradienților orientați sa ofere o acuratețe mai bună.

Principala ideea din spatele histogramei gradienților orientați este aceea că aspectul și forma locală a unui obiect poate fi descrisă de distribuirea intensității gradienților sau de orientarea marginilor, [6]. Imaginea este împărțită în regiuni conectate numite celule, iar fiecare celulă va determina o histogramă. Descriptorul este o concatenare a acestor histograme.

**Definiția 2.1: [15]** *‘Gradientul unei imagini este schimbarea direcției în intensitate sau culoare dintr-o imagine’.*

Detectarea muchiilor dintr-o imagine este una dintre cele mai întâlnite folosiri ale gradienților.

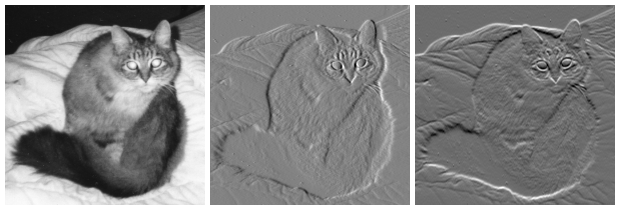
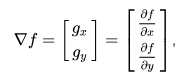


Figura 2.8: Gradienții unei imagini

Sursă: Wikipedia [15]

În partea stângă avem imaginea alb-negru a unei pisici. În centru avem imaginea gradienților în direcția x, măsurând schimbările orizontale în intensitate. În partea dreaptă avem imaginea gradienților în direcția y, măsurând schimbările verticale în intensitate. Astfel, pixelii gri din ultimele 2 imagini au o valoare mică a gradienților, pe când pixeli albi și negri au o valoare ridicată [17].

Matematic, gradientul unei imagini este un vector format din derivatele sale parțiale:

 (2.9)

unde este derivata în funcție de x, adică gradientul pe direcția x, pe când este derivata în funcție de y, reprezentând gradientul pe direcția y.

Direcția gradientului este dată de:

(2.10)

, iar magnitudinea gradientului, reprezentând cât de repede va crește intensitatea în acea direcție, poate fi calculată prin formula:

(2.11)

Una dintre cele mai simple metode de a găsi gradienții unei imagini este prin folosirea unui filtru Sobel. Pentru găsirea gradientului pe direcția x, filtrul Sobel va parcurge imaginea precum un filtru dintr-un strat convoluțional al unei rețele. Pentru găsirea gradientului pe direcția y vom aplica aceeași metoda, dar vom folosi transpusa filtrului Sobel. Pentru a obține gradienții pe ambele direcții este suficient să trecem cu primul filtru pe imaginea inițială, iar apoi cu al doilea filtru pe imaginea rezultată. Imaginea finală va fi folosită în construirea histogramei gradienților orientați. În acest pas este calculată direcția și magnitudinea gradienților.

Filtrul Sobel, numit după Irwin Sobel, este un filtru de dimensiune 3x3 și are următoarele componente:

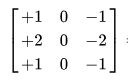


Figura 2.9: Componentele filtrului Sobel pe direcția x

Sursă: Wikipedia [15]

Alte filtre pot fi folosite pentru a obține gradienții unei imagini. În [7], histograma gradienților orientați a fost folosită pentru a detecta pietonii dintr-o intersecție. Filtrul care a dat cele mai bune rezultate aici a fost unul de dimensiune 1x3, având valorile [-1, 0, 1].

După aflarea gradienților și împărțirea imaginii pe celule, urmează să construim histograma în fiecare dintre aceste zone. Matricea direcției gradienților va conține valori cuprinse între 0 și 359, reprezentând măsura unghiului direcției gradientului într-un cerc trigonometric. Astfel, histograma va împărții aceste unghiuri în porțiuni, iar fiecare pixel își va aduna valoarea magnitudinii în componenta corespunzătoare direcției sale. După ce toți gradienții pixelilor unei celule au fost parcurși, histograma acelei celule va fi completă. Histograma gradienților orientați ale unei imagini rezultă din concatenarea histogramelor tuturor celulelor acelei imagini.

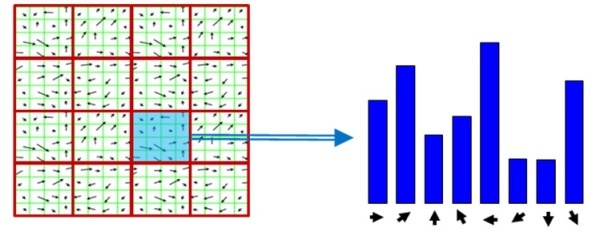


Figura 2.9: Reprezentarea grafică a histogramei unei celule

Sursă: Google [19]

### **2.1.3. Mașini cu suport vectorial (SVM)**

Mașinile cu suport vectorial sunt folosite în învățarea supervizată. Acestea mapează parametrii de intrare într-un spațiu de dimensiune foarte mare. În acest spațiu, o suprafață liniară de decizie este construită.

**Definiția 2.2**: **[15]** ‘*Învățarea supervizată în inteligența artificială este problema învățării unei funcții care mapează datele de intrare pe unele de ieșire plecând de la mai multe exemple de intrare-ieșire’.*

SVM-ul este printre cele mai bune( și mulți cred că este cel mai bun) algoritm de învățare supervizată standard, [11]. Acest clasificator definește un hiperplan pentru a separa clasele diferite.

**Definiția 2.3**: **[15]** ‘*În geometrie, un hiperplan este un subspațiu a cărui dimensiune este mai mică cu 1 decât a spațiului ambiental’.*

Conform definiției (2.3) hiperplanul într-un spațiu de dimensiune 2 este format dintr-o dreaptă sau un lanț de drepte.

**Definiția 2.4**: **[15]** ‘*Spațiul ambiental este spațiul care înconjură un obiect’.*

Un algoritm de antrenare SVM crează un sistem care, plecând de la un set de date notat corespunzător, atribuie unei noi intrări una dintre clase.

Un model de mașină cu suport vectorial poate fi gândit ca o adunare de mai multe puncte într-un spațiu, fiecare punct reprezentând una dintre datele de intrare. Acestea sunt adăugate astfel încât categoriile diferite să fie distanțate cât mai mult posibil una de cealaltă. Mașinile cu suport vectorial pot face și clasificări neliniare pe lângă cele liniare folosind așa-numitul truc al nucleului, trecând datele de intrare într-un spațiu cu dimensiune foarte mare, [15].

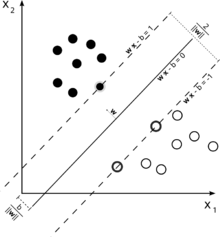


Figura 2.10: Reprezentarea grafică a unui SVM liniar

Sursă: Wikipedia [15]

Dreptele cu linie punctată se numesc vectori suport. Deși există multe linii care pot separa cele doua categorii din figura (2.10), hiperplanul optim este cel aflat la mijlocul distanței dintre cei 2 vectori suport și paralel cu aceștia.

Pornind de la un set de date format din n puncte de forma



, unde y reprezintă clasa din care face parte punctul x (1 sau -1), iar x este un vector de dimensiune m format din numere reale, trebuie găsit hiperplanul care desparte cele 2 grupe pentru care distanța minimă dintre acesta și oricare punct să fie maximă.

Putem scrie orice hiperplan ca:

 (2.12)

, unde w este vectorul normal al hiperplanului, [15].

**Definiția 2.5: [15] ‘***În geometrie, vectorul normal este vectorul perpendicular pe un obiect dat’.*

Construirea modelului SVM este influențată de 2 paremetrii:

* parametrul de **Regularizare** (notat cu C în librăria Scikit-Learn pentru Python) spune clasificatorului cât de mult îi permiți să clasifice greșit datele de antrenare. Pentru valori mari ale lui C, funcția de optimizare va alege un hiperplan format dintr-un lanț de drepte de dimensiuni mici pentru a face mai bună clasificarea punctele de antrenare. Pe de altă parte, o valoare mică a lui C va face funcția de optimizare să caute un hiperplan format dintr-o singură dreaptă, chiar dacă acel hiperplan va clasifica greșit câteva puncte, [16]. Problema cu alegerea unui C cu valoare mare este aceea că este posibil ca SVM-ul să nu prezică corect datele de testare, valoarea mică a lui C dând o generalizare mai mare a delimitărilor. De aceea, trebuie să descoperim valoarea variabilei C în funcție de fiecare set de date.
* parametrul **Gamma** definește care puncte influențează crearea hiperbolei de separare a claselor. Un gamma mic va face ca și punctele aflate la o distanță mai mare de posibila hiperbolă să fie luate în vedere în calcularea soluției. Pe de altă parte, doar punctele din apropiere vor fi folosite dacă valoarea lui gamma este mare, [16].

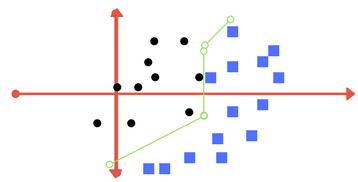
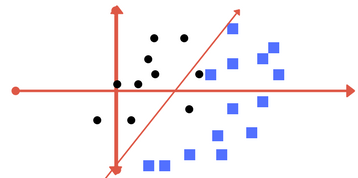


Figura 2.11: În stânga SVM-ul are valoarea parametrului de regularizare mică.

În dreapta SVM-ul are valoarea parametrului de regularizare mare.

Sursă: Mediul [16]

În continuare vom discuta despre noțiunea de margine în mașinile cu suport vectorial. SVM-ul încearcă să obțină o margine cât mai bună.

**Definiția 2.6**: **[15]** ‘*În inteligența artificială, marginea unui punct este definită ca distanța dintre acel punct și limita de decizie’.*

**Definiția 2.7**: **[15]** ‘*Limita de decizie este regiunea în care rezultatul unui clasificator este ambiguu’.*

Limita de decizie căutată trebuie să maximizeze marginea pentru a produce un model în ale cărui predicții sunt bune. Acest lucru va face clasificatorul să separe exemplele pozitive de cele negative eficient.

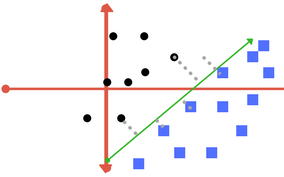
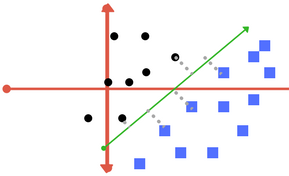


Figura 2.12: În stânga marginea este bună.

În dreapta marginea nu este construită bine.

Sursă: Mediul [16]

Marginea din imaginea stângă este bună deoarece aceasta a fost aleasă cât de egal s-a putut față de cele două părți. Pe de altă parte, imaginea din dreapta nu are marginea aleasă corect deoarece hiperbola este mai apropiată de clasa care conține pătratele albastre.

### **2.1.4. Metoda celor mai apropiați k vecini (KNN)**

Deși a fost introdusa acum mai bine de 60 de ani, metoda celor mai apropiați k vecini este încă un topic de discuție în inteligența artificială, [11]. Acest algoritm este folosit pentru clasificare trăsăturilor sau regresie.

În construirea clasificatorului KNN avem nevoie de 3 componente:

* setul de date pentru antrenare cu clasele corespunzătoare
* metoda prin care se va calcula distanța între 2 înregistrări. Cea mai folosită metrică în inteligența artificială este distanța Euclidiana, aceasta având următoarea formulă pentru 2 puncte p = (și q = (, [15]

(2.13)

* variabila k, reprezentând numărul de vecini care sunt luați în calcul în etapa de prezicere. Această variabilă este foarte importantă în vederea acurateții pe care o vom obține de la clasificatorul KNN. Dacă valoarea este prea mică, atunci algoritmul poate avea probleme în clasificarea setului de testare. Alegerea unui k mic face ca modelul să fie foarte sensibil la date de intrare introduse eronat. Pe de altă parte, o valoare prea mare a lui k poate duce la situația în care decizia este luată de puncte aflate la o distanță foarte mare de cel căutat. De exemplu, pentru un k = 21 și un punct X avem 7 vecini din clasa 1 la o distanță relativ mică și restul de 14 din clasa 0 la o distanță mai mare. Deși X este din clasa 1 și este înconjurat de clasa corectă în apropiere, prezicerea ne va spune că X este din 0 deoarece numărul vecinilor din 0 este de 2 ori numărul de 1.

**Algoritm**

Fie x un punct pentru care trebuie să prezicem clasa în care aparține. Inițial, vom calcula distanțele de la x la toate celelalte puncte din setul de antrenare și vom sorta vectorul rezultat crescător. Dacă k = 1, atunci clasa lui x va fi aceeași cu clasa celui mai apropiat vecin. Pentru k > 1 decizia va fi luată de un vot majoritar. În cazul în care votul se termină cu remiză, atunci decizia poate fi luată fie prin alegerea clasei celui mai apropiat punct, fie aleator din clasele aflate la egal, [11].

Un dezavantaj al acestui algoritm simplu este faptul că este destul de costisitor din punct de vedere al timpului de prezicere, metoda trebuind să calculeze de fiecare dată toate distanțele și să afle care sunt primele k dintre acestea. Totuși, conform [11], alți algoritmi mai rapizi au fost propuși de Farago în [13] și de Belur Dasarathy în [12], ultimul fiind bazat pe arbori de căutare.

În momentul în care verifici un model KNN pe lângă unul SVM se pot observa următoarele:

* la antrenare KNN-ul termină imediat, neavând de executat nicio operație, pe când în SVM trebuie create hiperbolele de decizie.
* la testare SVM-ul se mișcă mai repede decât un model KNN pe același set.
* acuratețea celor 2 modele diferă de la set la set și în funcție de parametrii aleși. Astfel, dacă trebuie să alegi un clasificator, cel mai bine ar fi să compari mai multe modele și să verifici rezultatele obținute.

Un lucru interesant care se poate observa într-un model KNN cu k = 1 este acela că limitele de decizie generează o diagramă Voronoi.

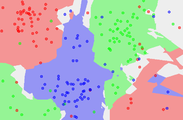
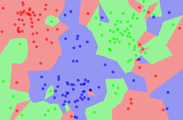


Figura 2.13: De la stânga la dreapta: setul de date, reprezentare grafică KNN pentru k = 1, reprezentare grafică KNN pentru k = 5.

Sursă: Wikipedia [15]

În partea stângă avem setul de date colorat diferit în funcție de clasa de care aparține. Imaginea din centru ne prezintă limitele de decizie ale clasificatorului KNN cu k = 1 folosind distanța euclidiană. În dreapta clasificatorul a fost calculat cu k = 5. Zonele albe au fost cauzate de remiză în calcularea votului majoritar ( de exemplu: 2 vecini roșie, 2 albaștrii și unul verde), [17].

## **2.2. Metode recente pentru recunoașterea facială din imagini**

### **2.2.1. FaceNet**

Rețeaua convoluțională FaceNet a fost propusă de cei de la Google în 2015. Spre deosebire de rețelele precedente care pentru a calcula acuratețea pe un set de date nou foloseau un start intermediar pentru a generaliza clasificarea, aceștia s-au gândit la o modalitate total diferită.

Conform [14], ei au construit o rețea care pentru o imagine trimisă prin rețea să returneze un vector de 128 numere reale. Vectorul rezultat din rețea are următoarea proprietate: distanța euclidiană dintre el și vectorul realizat dintr-o altă imagine a aceeași persoane este mai mică decât distanța față de vectorul rezultat din imaginea unei persoane diferite. Imaginile vor fi împărțite în grupuri într-un spațiu de dimensiunea 128. Acest lucru ne permite să antrenăm un clasificator elementar precum SVM sau KNN și să obținem rezultate comparabile cu alte rețele.

Spre deosebire de rețeaua FaceNet care returnează doar 128 octeți, celelalte rețele (pornind de la AlexNet în 2012 [1] și ajungând la VGG în 2014 [3]) au avut ultimul strat complet interconectat de dimensiune 1000.

Rețeaua FaceNet a fost antrenată folosind o funcție de cost triplă. La fiecare pas rețelei i-a fost dat ca date de intrare 2 imagini ale aceleiași persoane și una a altei persoane. Spre deosebire de rețelele precedente a căror funcții de cost le făceau să învețe să clasifice corect persoanele din imagini, funcția de cost triplă are rolul de a face ca vectorul rezultat să aproprie imaginile ale aceleiași persoane prin distanța euclidiană. Formula funcției de cost triplă este următoarea:

 (2.14)

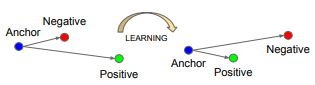


Figura 2.14: Antrenarea în FaceNet

Conform figurii 2.14 rețeaua primește câte 3 imagini. Funcția de cost triplă încearcă să apropie imaginea ancoră de imaginea pozitivă (adică cea a aceleași persoane) și să o depărteze de cea negativă.

### **2.2.2. ResNet**

Pornind de la rețeaua AlexNet care avea doar 8 staturi și ajungându-se la VGG, aceasta având 16 staturi, s-a pus problema numărului optim de staturi pe care trebuie să le aibă o rețea. În [5] s-a arătat că pentru a scoate o acuratețe mai bună nu este suficient doar să antrenăm o rețea cu un număr mare de straturi. Aceștia au obținut o eroare pe setul de antrenare mai mare pentru o rețea cu 56 staturi decât pe una cu 20.

Explicația acestui fenomen este următoarea: Deoarece există un număr mare de staturi în acea rețea, gradientul în pasul de propagare înapoi în primele staturi o să aibă o valoare foarte mică, posibil să ajungă chiar la 0, iar rețeaua nu va reuși să învețe.

Pentru a evita această situație, autorii articolului [5] au propus o învățare reziduală. Ei au decis să introducă o funcție de identitate (numit strat rezidual) care va sări unul sau mai multe staturi. Rețeaua în care au fost introduse aceste staturi reziduale nu ar trebui să producă o acuratețe mai slabă decât una fără acestea deoarece am putea folosi doar staturile inițiale și astfel prima rețea ar fi identică cu cea de-a doua, rezultând o acuratețe egală.

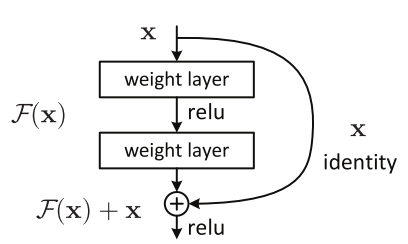


Figura 2.15: Reprezentarea unui strat rezidual

Prin antrenarea unei rețele cu astfel de straturi au observat că a învățat mult mai bine decât cea care nu le conținea. Acest lucru se datorează faptului că rețeaua poate decide acum dacă va antrena acele straturi sau va sări peste ele.

Prin acest procedeu Microsoft a reușit să construiască o rețea de 152 staturi prin care a câștigat ILSVRC 2015, obținând o eroare pe primele 5 predicții de 3,56%.

## **2.3. Algoritmul de recunoaștere facială implementat**

În implementarea algoritmului de recunoaștere facială am folosit o rețea preantrenată ResNet pentru a extrage trăsăturile din imagini și un clasificator simplu precum SVM sau KNN pentru a antrena modelul pe setul de date propriu. Motivul pentru care am fost nevoit să folosesc o rețea preantrenată și nu am antrenat una de început este acela că setul de date este foarte restrâns, conținând doar 10600 imagini și fiind împărțit în 100 clase, iar antrenarea unei noi rețele pe un astfel de set ar produce o acuratețe mai mică decât folosirea uneia deja antrenate.

Algoritmul de recunoaștere a actorilor a fost făcut în limbajul Python. Python este un limbaj de programare gratuit care poate fi rulat pe toate sistemele de operare. Popularitatea acestui limbaj a crescut imens datorită interesului care este acordat inteligenței artificiale, fiind principalul limbaj folosit în acest domeniu.

Pentru a obține trăsăturile dintr-o imagine am folosit libraria Face Recognition, [18]. Această librărie este disponibilă pe sistemele de operare macOS și Linux, dar există și ghiduri pentru instalarea sa pe Windows, deși acest sistem nu este suportat oficial. Aici, ne sunt oferite funcții pentru găsirea fețelor din imagini, existând atât o implementare naivă prin HOG+SVM, precum și una prin rețele convoluționale. Aceasta din urmă are nevoie de mai mult timp să dea un răspuns, deși acuratețea este mai mare. În plus ne este oferită o funcție de extragere a 128 trăsături dintr-o față.

Extragerea trăsăturilor se realizează folosind o rețea ResNet antrenată pe milioane de imagini și care are o acuratețe de 99,38% pe setul de date ‘Labeled Faces in the Wild’, [18]. Pentru antrenare a fost folosită o funcție de cost triplă precum cea propusă în [14]. A fost necesar doar să trecem acești vectori de trăsături printr-un clasificator SVM/KNN pentru a obține rezultate foarte bune.

Pentru clasificatorii SVM și KNN am folosit librăria Scikit-learn din Python. Aceasta este o librărie disponibilă pe toate sistemele de operare care oferă implementări pentru mai mulți algoritmi din domeniul inteligenței artificiale.

**Algoritm**

Pasul 1 este reprezentat de citirea setului de date și creerea celor 2 vectori: datasetX care va conține trăsăturile și datasetY care va reține clasa corespunzătoare. Toate imaginile din setul de date vor fi parcurse, iar pentru fiecare imagine vom extrage fețele prezente și vom trece aceste fețe prin rețea pentru a obține trăsăturile.

for director\_clase in os.listdir(director\_dataset):

for locatie\_imagine in image\_files\_in\_folder(os.path.join(director\_dataset, director\_clase)):

#incărcăm imaginea

imagine = face\_recognition.load\_image\_file(locatie\_imagine)

#extragem toate fețele din imagine

fețe = face\_recognition.face\_locations(imagine)

if len(fete) == 1:

#adăugăm în vector trăsăturile rezultate din rețea

datasetX.append(face\_recognition.face\_encodings(imagine, known\_face\_locations=fețe)[0])

datasetY.append(director\_clase)

La pasul 2 vom împărți vectorii datasetX și datasetY în seturi pentru antrenare și testare, iar apoi vom antrena modelul prin funcția trainSVM.

def trainSVM(model\_save\_path=None):

# se crează un clasificator SVM cu parametrii dați

svm\_classifier = svm.SVC(kernel='linear', C = 0.3)

# se antrenează pe SVM setul de date și se salveză modelul

svm\_classifier.fit(trainX, trainY)

if model\_save\_path is not None:

with open(model\_save\_path, 'wb') as f:

pickle.dump(svm\_classifier, f)

return svm\_classifier

În final vom crea funcția de prezicere și vom calcula acuratețea.

#funcție care prezice clasa unui vector de trăsături

def predictSVM(faces\_encodings):

return svm\_classifier.predict(faces\_encodings)[0]

#calcularea acurateții

total = 0

nrCorecte = 0

for i in range(len(testX)):

name = predictSVM([testX[i]])

total = total + 1

if name == testY[i]:

nrCorecte = nrCorecte + 1

print("Corecte KNN: " + str(nrCorecte))

print("Totale KNN: " + str(total))

print("Acc KNN: " + str(nrCorecte/total))

Librăria Flask din Python a fost folosită pentru a crea un serviciu REST care va primi de la aplicația web locația imaginii pe server și va returna clasa persoanei din imagine sau șirul vid dacă imaginea este incorectă.

## **2.4. Rezultate și experimente**

Setul de date de care am avut nevoie pentru această aplicație a trebuie să fie compus din imagini cu actori, aproximativ 100 imagini pentru fiecare din cei 100 actori aleși. Deoarece nu am reușit să găsesc un astfel de set, am luat decizia să-l construiesc de unul singur, ajungând astfel să testez cât de bine funcționează clasificarea pe un set nou de imagini. Actorii aleși au fost luați dintr-un top 100 a celor mai populari actori din anul 2017, realizat de platforma IMDB.

Pentru început, a trebuie să găsesc un program care să ma ajute să descarc imagini de pe internet. Am decis să folosesc o extensie de Google Chrome prin care mi se punea la dispoziție să aleg ce imagini să descarc de pe pagina în care sunt. Prima problemă pe care am avut-o a fost că Google Images îmi arăta și imagini în care actorul căutat era între mai multe persoane sau nu era în imagine. Din 500 rezultate rămâneam cu aproximativ 200 după prima fază. Aceste imagini erau apoi descărcate și verificate pentru a elimina duplicatele și pentru a decupa părțile care nu erau dorite, de exemplu o altă persoană.

În final am ajuns la un total de 10654 de imagini distribuite uniform în 100 clase. Acest proces a durat peste 24 de ore. Consider că, deși crearea setului de date a fost destul de monotonă, procesul nu este unul deloc dificil.

În continuare am dorit să aflu cât de bine va reuși un model format dintr-o rețea preantrenată și un clasificator simplu să claseze un set nou de date. Setul a fost împărțit astfel: 7654 imagini au fost utilizate în antrenare, iar restul de 3000 au fost folosite pentru a calcula acuratețea programului.

Rețeaua convoluțională aleasă a fost un ResNet antrenat cu o funcție de cost triplă. Inițial, am folosit un clasificator SVM liniar și am obținut o performanță de aproximativ 99,66%. Totuși, prin reducerea parametrului C a clasificatorului la 0.3 am reușit să obțin o performanță de 99,7%. După aceea am verificat și cu un KNN, acesta reușind o acuratețe tot de 99,66% pentru k = 1, dar prin modificarea numărului de vecini la 5 am obținut o acuratețe de 99,73%.

În continuare, am decis să testez și un algoritm format din histograma gradienților orientați pentru a extrage atributele feței și un clasificator simplu. Pentru extragerea histogramei am folosit librăria OpenCV. Această librărie este destinată domeniului de viziune a calculatorului, oferind multe implementări de algoritmi elementari.

Folosind clasificatorul SVM și prin alegerea parametrilor, am ajuns la o acuratețe de 55,25%. Ținând cont că această metodă a fost introdusă acum mai bine de 10 ani și că dimensiunea setului de antrenare este una restrânsă, putem trage concluzia că acest model s-a descurcat onorabil, recunoscând corect persoana din imagine cel puțin o dată din două încercări. Totuși, folosind KNN nu am obținut un rezultat atât de bun, doar 27,6% pentru numărul de vecini egal cu 11.

Putem observa că prin folosirea unei rețele foarte bine antrenate cu o funcție de cost triplă se obțin rezultate foarte bune pe un set de date nou cu un număr relativ mic de imagini, pe când HOG nu s-a descurcat prea bine.

|  |  |  |
| --- | --- | --- |
| Metodă de extragere a trăsăturilor | Clasificator | Acuratețe |
| ResNet (Conv) | SVM | 99,7% |
| ResNet (Conv) | KNN | 99,73% |
| HOG | SVM | 55,25% |
| HOG | KNN | 27,6% |

Tabelul acurateții obținute în funcție de metodă

**Capitolul 3**

# **3. Tehnologii folosite în dezvoltarea aplicației web**

## **3.1. Java**

Aplicația web a fost realizată în **Java**. Java este un limbaj de programare de nivel înalt bazat pe clase făcut de cei de la Oracle în 1995. În prezent, Java a ajuns la versiunea 10, aceasta fiind lansată în Martie 2018.

O diferență interesantă între Java și C++ este lipsa variabilelor constante. Totuși, există o modalitate de a obține o astfel de variabilă folosind cuvintele cheie ‘static final’. Un câmp ‘static’ aparține clasei și nu obiectelor, însemnând că variabila constantă va fi disponibilă fără a crea un obiect. În plus, ‘final’ ne obligă să inițializăm variabila o singură dată. Astfel obținem o variabilă care va aparține tuturor obiectelor și care nu poate fi modificată.

În interiorul unei clase Java pot apărea unele construcții numite blocuri statice. Acestea sunt folosite pentru a inițializa variabilele statice și sunt executate doar la primul obiect creat din acea clasă. Ordinea de executare într-o clasă Java este următoarea: blocul static din supraclasă, blocul static din subclasă, constructorul din supraclasă și în final constructorul din subclasă.

Fișierele Java au extensia ‘.java’, iar acestea vor fi transformate în fișiere ‘.class’. Acestea din urmă conțin cod Java bytecode, care va fi compilat de mașina virtuală Java. La o singură rulare mașina virtuală Java încarcă mai mult de 20000 clase în memorie.

Java este împărțită în 3 mari componente:

* JavaEE, cuprinzând toate componentele necesare realizării unei aplicații desktop: sintaxă, tipurile de date, elementele POO, colecții, interacțiunea cu baza de date prin JDBC, manipularea fișierelor XML, JSON, lucrul pe mai multe fire de execuție, excepții etc.
* JavaSE, prin care putem crea aplicații și servicii web: HTTP Request/Response, HTTP Session, JSP, Servlet, JSTL, JAX-WS, JAX-RS.
* JavaME, prin care se pot realiza aplicații mobile

Aici poate fi precizat și Android-ul care, deși este făcut de cei de la Google, are la bază limbajul Java.

Interfața Throwable stă la baza erorilor și excepțiilor limbajului Java. Erorile (clasa Error care implementează Throwable) sunt problemele care apar la mașina virtuală Java, acestea rezultând în oprirea execuției. Excepțiile (clasa Exception care implementeză Throwable) sunt fie de tipul celor care trebui trecute în clauza ‘catch’ pentru a oferi o rezolvare în cazul în care apar, fie derivate din clasa RuntimeException (de exemplu NullPointerException).

În Java nu este nevoie să eliberăm memoria obiectelor create, acest lucru fiind realizare de mașina virtuală printr-un program demon care eliberează memoria folosită de obiectele care nu mai sunt folosite. Există 2 metode de a cere mașinii virtuale să ruleze acest program de eliberare al memoriei: ‘System.gc(); ’ și ‘Runtime.getRuntime().gc(); ’. Cu toate acestea nu există nici o modalitate să ne asigurăm ca mașina virtuală a ascultat de cererile noastre. Pot apărea situații în care trebuie să executăm anumite operații la distrugerea unui obiect. Un astfel de exemplu este închiderea conexiunii cu baza de date sau închiderea unui fișier deschis de constructorul clasei. Acest lucru se poate realiza suprascriind funcția ‘finalize()’ din clasa Object, cea mai importantă clasă din Java.

## **3.2. Maven**

**Maven** este un program realizat de Apache prin care dezvoltatorii Java pot introduce ușor dependințe în proiectele lor. Aceste ne oferă o serie de comenzi prin care putem: instala pachetele local pentru a le putea folosi în alte proiecte (‘mvn install), valida și compila codul, curăța artefactele vechi și lansa aplicația pe un server, [20]. În plus, proiectul ne este împărțit în ‘main’, unde vom păstra pachetele și tot ce ține de aplicație și în ‘test’, unde vor apărea testele unitare. Acestea pot fi scrise în JUnit și vor fi verificate la fiecare comanda ‘mvn clean install’, aceasta realizând succesiv cele 2 comenzi copil.

**Definiția 3.1**: *Testele unitare sunt o serie de verificări realizate de programatori prin care ne asigurăm că o anumită funcție va returna rezultatul corect chiar dacă implementarea acesteia s-a schimbat.*

Testele unitare ne permit să găsim mult mai ușor eventuale erori care pot apărea la anumite modificări ale funcțiilor existente. Se spune că o aplicație realizată corect ar trebui să aibă teste unitare pe mai mult de 80% din funcții.

## **3.3. Spring**

Spring-ul este o aplicație care are la bază principiul de proiectare numit ‘Inversiunea controlului’. Acest principiu ne spune că dezvoltatorul ar trebui să fie atent doar la părțile specifice ale unei aplicații, iar părțile comune să fie făcute de aceste aplicații precum Spring.

Fie o clasă Brad creată în Java. Pentru a obține un obiect cu proprietăți din acestă clasă este necesar să creăm unul nou folosind ‘new’ și să atribuim aceste proprietăți. Pe de altă parte, folosind modelul de programare al fabrici putem obține acest obiect apelând o metodă prin care trimitem proprietățile. Acest fenomen se numește căutarea dependinței. Totuși, prin injectarea dependinței obiectul este trimis direct. Acest lucru poate fi implementat în constructori, în funcțiile de setare și în proprietățile claselor.

Pentru implementarea injectării dependințelor prin constructori clasa care le conține trebuie să fie adnotată cu ‘@Configuration’. Această adnotare îi spune Spring-ului că acea clasă conține un bob. Un bob Spring este creat și administrat de containerul Spring, aceste boabe fiind compuse folosind unele fișiere de configurare oferite de către programator containerului, de exemplu folosind formatul XML. Crearea unui bob pentru clasa Brad se face în felul următor: <bean id=”brad” class=”Brad” />. Adnotarea ‘@Bean’ este folosită pentru a defini că o metodă este un bob. Dacă nu este specificat un nume personalizat, numele ales va fi numele metodei. Pentru realizarea injectării dependinței pe funcțiile de setare se folosește atributul ‘property’ în XML: <bean id=”brad” class=”Brad”> <property name=”nrRamuri” ref=”nrRamuri1” /> </bean>. Injectarea directă în proprietățile claselor se face folosind adnotarea ‘@Autowired’.

Spring folosește adnotări pentru afla tipul clasei, metodei sau proprietății. Astfel, toate clasele controlor din Spring trebuie adnotate cu ‘@Controller’. Adnotările ‘@RequestMapping’ sunt folosite pentru a mapa o adresă pe o anumită clasă sau metodă. Dacă clasa are o adnotare pe adresa ‘/url1’ și o metodă din interiorul ei ‘/url2’, atunci metoda va mapa adresa ‘/url1/url2’ în final. ‘@RequestParam’ este folosit pentru a lega un parametru al cererii web de o variabilă a funcției. Adnotarea ‘@ModelAtribute’ este una dintre cele mai importante în Spring. Prezența ei ne permită să legăm parametrii unei cereri de tip POST de o clasă fără să parsăm fiecare câmp individual. Acest fenomen este cunoscut ca legarea datelor în Spring.

Adnotarea specifică claselor de servicii este ‘@Service’, ele cuprinzând toată logica de business a aplicației (de exemplu funcțiile de validare). Acestea sunt principalele metode pe care trebuiesc făcute teste unitare, nu funcțiile care fac modificări pe baza de date. ‘@Repository’ este folosită pentru clasele de acces la baza de date, iar ‘@Component’ este o adnotare generică pentru clasele administrate de Spring, dar care nu sunt controlere, servicii sau clase care fac legătura cu baza de date.

## **3.4. Oracle**

Oracle este un sistem de gestiune al bazelor de date creat de firma cu același nume. Această bază de date este de tip SQL.

**Definiția 3.2**: *O baza de date reprezintă un mod prin care putem stoca și realiza operații cu milioane de înregistrări.*

Înregistrările bazei de date sunt reținute în tabele. Un tabel reprezintă o relație între mai multe mulțimi. O tabelă cuprinde o listă a câmpurilor, aici fiind specificate numele, tipul, dimensiunea și valoarea implicită unde este cazul. Pe lângă aceasta tabela conține și o listă de constrângeri ale datelor salvate precum unicitatea sau completarea obligatorie a câmpului. Fiecare tabelă trebuie să aibă o cheie primară, adică un atribut sau o combinație de atribute unică și diferită de nul pe acel tabel.

Oracle permite folosirea unui limbaj de tip SQL prin care putem declara și manipula datele. Pe lângă acesta ne este pus la dispoziție limbajul PL/SQL prin care putem folosi metode pentru a realiza operații mai complexe pe baza de date precum parcurgerea datelor selectate folosind un cursor sau construcția ‘pentru’. O particularitate a limbajului PL/SQL este aceea că putem crea unele funcții care să fie rulate automat înainte sau după executarea uneia sau mai multe operații de manipulare (creare, actualizare sau ștergere). Procedurile sunt de 2 feluri: vizibile doar în interiorul blocului PL/SQL sau globale, cele din urmă putând fi apelate folosind funcția ‘call’.

## **3.5. MyBatis**

MyBatis este un program folosit pentru a stabili legătura dintre aplicația Java și baza de date Oracle. Acesta folosește fișiere cu extensia ‘.xml’ pentru a scrie implementarea funcțiilor din interfețele pe care le mapează. Apelarea metodelor interfețelor se face creând un obiect cu adnotarea ‘@Autowired’.

Librăria ne pune la dispoziție o modalitate de evitare a scrierii de mai multe ori a aceluiași cod folosind tagul <sql>, acesta putând fi folosit pentru a enumera toate câmpurile dorite atât în citirea tuturor datelor din tabel, cât și la recuperarea unei anumite intrări după identificator.

Structura ‘dacă’ poate fi folosită direct în fișierul ‘.xml’. Prin această structură putem construi interogări dinamice a bazei de date. Acest lucru este foarte avantajos când vrem să facem o retragere de date folosind un filtru cu foarte multe componente. Folosind condițiile ‘dacă’ putem scrie aceste interogări foarte ușor.

MyBatis-ul ne oferă posibilitatea să mapăm coloanele interogării bazei de date pe un obiect Java, spunându-i coloanei de ce proprietate să fie legată. Dacă o coloană nu este specificată, aceasta se va lega de proprietatea cu același nume.

## **3.6. HTML**

HTML este un limbaj folosit pentru aranjarea elementelor într-o pagina web. Acesta stă la baza oricărei aplicații web, fiind componenta de bază a acestor aplicații. Acesta ne propune o mare diversitate de elemente din care putem alege în funcție de ce avem de realizat. Distribuția elementelor unui fișier HTML este una ierarhică. Un document HTML are 2 componente principale: cap( head) și corp( body). Capul este folosit, în principal, pentru a specifica titlul paginii și pentru a încărca fișierele de stil și de script. Totuși, în ultimul timp tot mai multe persoane au început să adauge fișierele de script la finalul componentei de corp. Acest lucru face ca pagina să se încarce mai repede deoarece compilarea scriptului are loc după ce aceasta s-a afișat. Cu toate acestea, compilarea scriptului la final poate face probleme mari în cazul funcțiilor care se execută după încărcarea ferestrei în cazul în care internetul are o viteză mică.

JSP-ul este o tehnologie care ne ajută să creăm pagini web dinamice. Acesta ne oferă posibilitatea să scriem cod Java direct în paginile HTML. JSTL este o librărie care ne oferă suport pentru realizarea unor sarcini structurale folosind ‘dacă’, ‘alege dintre’ și ‘pentru fiecare’. În plus, ne sunt puse la dispoziție posibilități să formatăm ușor lucruri precum data, timpul și numere.

## **3.7. CSS**

Limbajul CSS este folosit pentru stilizarea elementelor HTML. Acesta ne oferă posibilitatea să alegem elementele după: identificatorul unic al elementului, clasele acestora sau tipul de element HTML. În plus, ne este pus la dispoziție și posibilitatea de a alege mai multe tipuri de elemente, precum și de a alege elemente într-un mod ierarhic ( de exemplu, toate paragrafele cu clasa X din div-ul cu identificatorul Y). Există 3 posibilități de a adăuga stil unei pagini HTML. Prima posibilitate este de a-l adăuga în componenta ‘style’ a tuturor elementelor HTML. Deși acest lucru nu este recomandat pentru proprietățile similare ale mai multor elemente, consider că este util să adaugăm aici acele caracteristici individuale elementului deoarece sunt mult mai ușor de găsit și modificat decât să cautam acel element după un identificator unic într-un fișier de stilizare. A doua posibilitate de folosite este prin introducerea în componenta ‘head’ al paginii web între tagurile ‘<style>...</style>’. Acest lucru ne permite să ne referim la mai multe elemente prin clase sau prin tip în modul următor: ‘.’ urmat de numele clasei este folosit pentru a identifica acea clasă, iar tipul nu este precedat de nimic. Totuși, nu ne dorim să avem stilul în paginile HTML în procesul de dezvoltare al aplicație. De aceea, pornind de la principiul de separare al conceptele s-a luat decizia de a pune stilul într-un fișier separat și de a-l include în HTML printr-un import.

Pe lângă posibilitatea de a modifica poziție, dimensiunea, culoarea și restul proprietăților unor obiecte, CSS-ul ne pune la dispoziție posibilitatea de a efectua tranziții ale obiectelor, animații sau transformări 2D și 3D la anumite acțiuni precum apăsarea unui buton de pe mouse folosind tagul ‘active’ sau planarea asupra componentei prin tagul ‘hover’.

## **3.8. JavaScript + JQuery + Ajax**

JavaScript este cel mai folosit limbaj de programare. Spre deosebire de limbajele Java, C, C++ care trebuiesc compilate, acesta este un limbaj de interpretare. În compilare, compilatorul prelucrează tot codul de la început, pe când la interpretare este aleasă prima line, este compilată și executată, iar apoi se trece la linia următoarea. La fel ca și CSS-ul, JavaScript-ul poate fi trecut atât în componenta ‘head’ a fișierului, cât și într-un fișier separată care apoi este importată în HTML pentru a respecta principiul separării conceptelor. De aceea vom avea 3 fișiere separate pentru interfața unei pagini web, fiecare având funcția sa proprie.

JQuery este o librărie de JavaScript folosită pentru a scrie cod mai rapid. Aceasta ne pune le dispoziție posibilitatea de a folosi selectorii CSS pentru a alege elementele, putând astfel să facem o selecție mai complicată de elemente mult mai ușor decât dacă am fi avut doar JavaScript. În plus codul scris este mai scurt, scăzând timpul de dezvoltare și fiind mai ușor de urmărit. De exemplu, pentru a căuta un element după identificatorul ‘actor’ în JavaScript ar trebui să scriem document.getElementById(‘actor’), pe când în JQuery același lucru se poate obține apelând $(‘#actor’).

Spre deosebire de JavaScript care lucrează direct cu proprietățile obiectelor, JQuery se folosește de funcții pentru a face același lucru: pentru găsirea informației se apelează funcțiile fără parametrii, iar pentru setare se trimite valoarea de atribuire apelând metodele cu parametru. Deși sunt diferite, funcțiile JQuery sunt asemănătoare cu proprietățile corespunzătoare în JavaScript: corespondenta proprietății ‘value’ din JavaScript este funcția ‘val()’ din JQuery, ‘innerHTML’ corespunde cu ‘html()’ etc. Conversia de la un obiect JavaScript la unul JQuery se face simplu folosind $: obJQ = $(obJS).

AJAX este folosit pentru a face cereri la server fără a reîncărca pagina web. La noi, AJAX a fost folosit pentru apela serviciul REST de actualizare a filmelor favorite ale utilizatorului conectat. Cererile la server prin AJAX pot fi făcute folosind atât JavaScript, cât și JQuery. Pentru a trimite parametrii la serviciu se poate folosi câmpul ‘data’ pentru cereri de tip POST, iar parametrii pentru cererile GET se pot trece la finalul adresei serviciului prin adăugarea unui ‘?’ și înșirarea parametrilor în felul următor ‘cheie=valoare’, fiind separați de caracterul ‘&’. Principala diferență între POST și GET este că GET-ul trece parametrii la finalul adresei, iar POST-ul îi salvează în interior. De aceea, conectarea prin parola sau alte operații care conțin informații cu caracter personal trebuie să fie executate folosind POST pentru a nu permite nimănui să vadă parola în clar în adresă.

Recent, dezvoltatorii web au preferat să folosească Angular sau Vue în detrimentul JavaScript-ului pur și JQuery-ului, acestea oferind o mai bună comunicare cu serviciile web decât cele din urmă. Construirea unei aplicații care folosește doar servicii web este atractivă pentru programatori datorită faptului că face posibilă comunicarea între limbaje total diferite. Spre exemplu, în aplicația noastră am creat un serviciu Rest în Flask care primea ca parametru locația imaginii și returna serverului Java numele actorului din imagine. Astfel, la crearea interfeței putem alege ce dorim să folosim, nefiind obligați de limbajul în care a fost scris serverul.

**Capitolul 4**

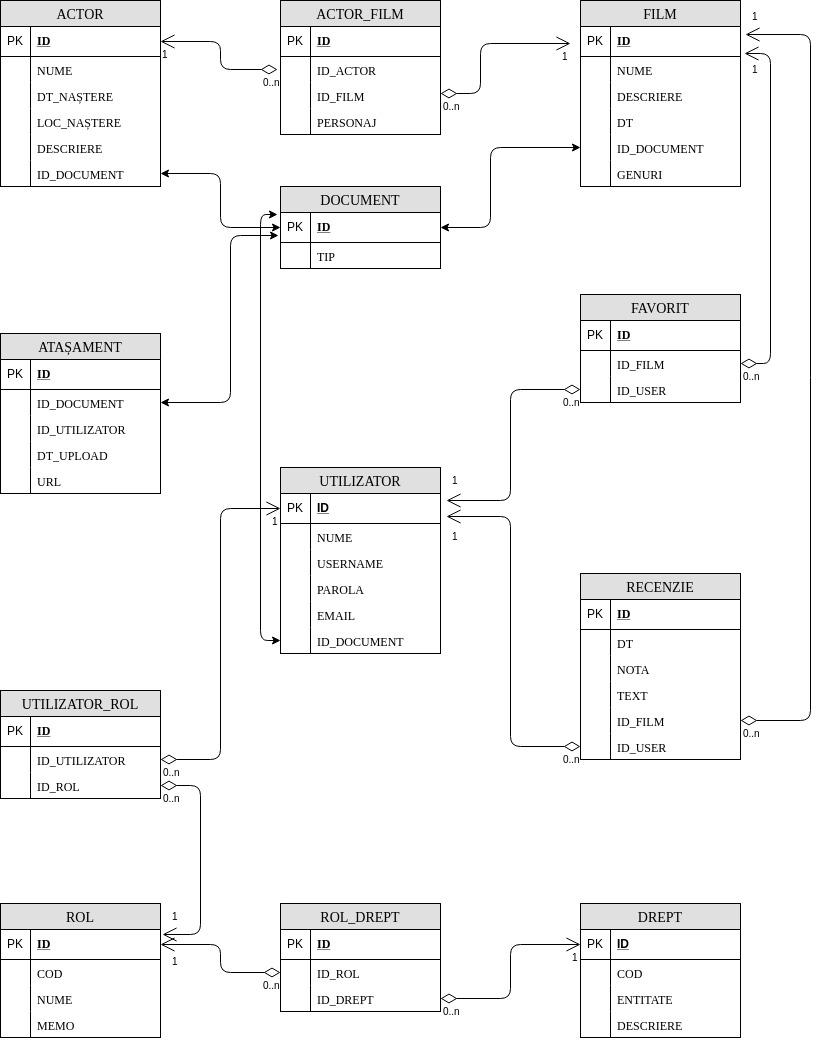
# **4. Descrierea aplicației**

Aplicația web a fost creată cu scopul de a pune la dispoziție un mod de a recunoaște un grup de actori din imagini și de a afla informații despre aceștia.

Aplicația este formată din 4 module care comunică folosind Maven:

* InfoAct. Aici sunt păstrate scripturile bazei de date, diagrama acesteia, o listă cu abrevieri folosită în instrucțiunile SQL pentru tabele și orice alte fișiere care nu conțin cod.
* InfoActApp. Cea mai importantă funcție a acestui modul este executarea operațiunilor pe baza de date. În directorul ‘resources’ oferit de Maven sunt salvate fișierele ‘.xml’ ale MyBatis-ului, aici fiind scrise instrucțiunile de manipulare ale datelor din bază. Pe de altă parte, directorul ‘src’ conține interfețele pe care sunt mapate fișierele din MyBatis, clasele prin care se verifică dacă informația care urmează să fie introdusă în baza de date este una corectă și serviciile, acestea cuprinzând o serie de funcții care sunt folosite de aplicație.
* InfoActCommon. Aici sunt păstrate obiectele de domeniu și cele care vor fi folosite în filtrarea rezultatelor după anumite criterii.
* InfoActWeb. Partea principală a aplicației web. Aici sunt păstrate controalele, serviciile Web (realizate folosind Jax-RS), fișierele de stilizarea, JavaScript și HTML, precum și fișierele de configurare ale Spring-ului.
* InfoActRecognition. Parte separată a aplicație care pune la dispoziție un serviciu web Rest realizat în librăria Flask din Python prin care are loc comunicarea dintre aplicația Java și funcția de recunoaștere.

Baza de date a fost realizată în Oracle, iar comunicarea între aceasta și serverul web a fost făcută folosind MyBatis. Informațiile prin care ne conectăm la bază au fost reținute într-un fișier de configurare, din care a fost creat mai târziu un Spring Bean pentru a recupera aceste informații. Diagramă relațională a bazei este prezentată în continuare.



## Diagrama relațională a bazei de date

Coloana ID în fiecare tabelă este generată folosind o secvență care se incrementează cu 1. Acest lucru ne oferă un identificator unic pentru fiecare intrare care apoi poate fi folosit să ne referim la acel obiect fără a transmite toate informațiile, ci doar identificatorul. De obicei acesta este folosit în aplicațiile web pentru a exprima o acțiune pe acel obiect trecând-ul ca parametru în cererea către server. Pentru recuperarea informațiilor este suficient doar identificatorul deoarece le putem citi tot obiectul din baza de date prin acesta.

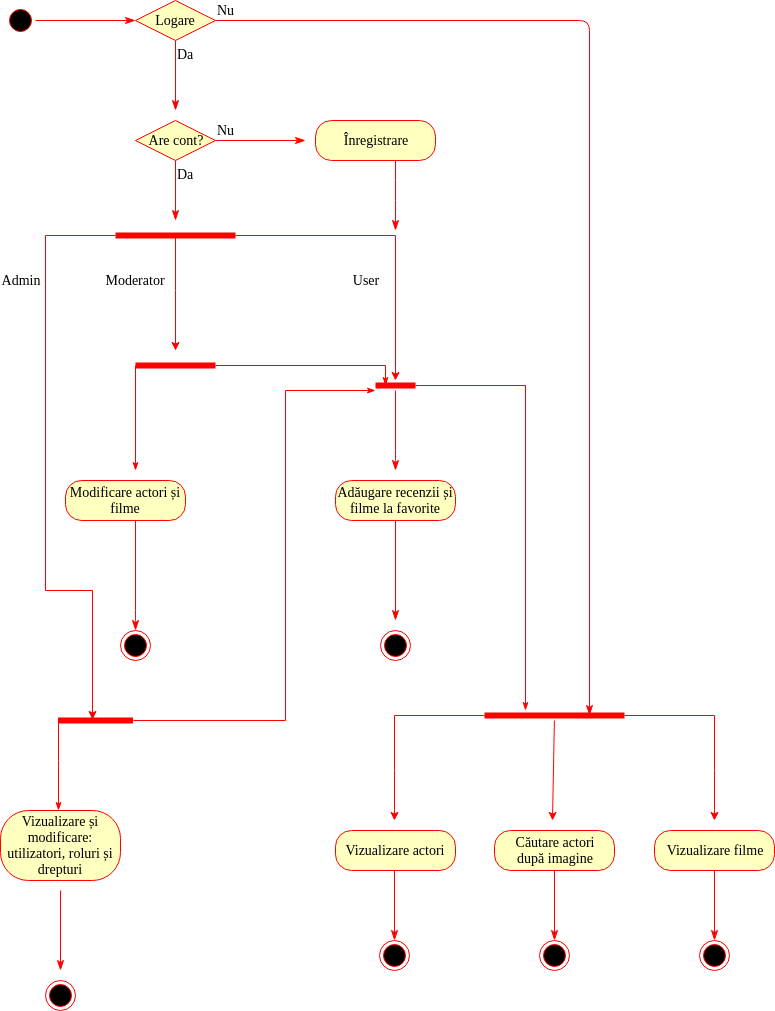
Tabelul ACTOR a fost folosit pentru a reține informațiile actorilor. Câmpul ID\_DOCUMENT crează o relație de 1 la 1 cu tabela DOCUMENT, aceasta fiind folosită la afișarea imaginilor actorilor. Între tabelele ACTOR și FILM există o relație ‘many to many’ deoarece actorul poate apărea în mai multe filme, iar un film poate avea mai mulți actori. Aceasta este folosită pentru a ne spune ce actor a jucat în ce film, precum și personajul pe care l-a interpretat. Tabelul FILM reține datele filmelor introduse de moderator.

Tabelul UTILIZATOR reține informațiilor utilizatorilor înregistrați în aplicație. Parola este salvată după trecerea prin funcția SHA-256 pentru a asigura protecția datelor în cazul în care un atacator reușește să afle informațiile reținute în bază. ATAȘAMENT-ul ne păstrează informații despre imaginile încărcate de moderatori pentru a fi afișate în paginile actorilor și filmelor corespunzătoare. Aici sunt păstrate informații despre cine și când a încărcat o imagine, precum și numele acesteia în fișierul de pe server. Tabela FAVORIT ne spune dacă un utilizator a introdus un film la favorit, astfel dând posibilitatea fiecărei persoane să își salveze un film pentru a-l urmări mai târziu. RECENZIE este folosit pentru a permite utilizatorilor să adauge o recenzie unui film. Folosind coloana NOTĂ din această tabelă putem crea o diagramă a notelor date de utilizatori unui film.

ROL-ul reprezintă rolurile utilizatorilor din aplicație, precum moderator, admin etc. Tabela UTILIZATOR-ROL ne permite să asigurăm unui utilizator mai multe roluri. DREPT și ROL\_DREPT ne permite să adăugăm drepturi de citire/scriere pe entități pentru fiecare rol.

Folosirea tabelelor de roluri și drepturi ne oferă posibilitatea să adăugam direct din aplicație noi roluri și să atribuim drepturi acestora. Într-o aplicație de dimensiuni mari acest lucru este obligatoriu deoarece mereu pot apărea funcționalități speciale unor tipuri de utilizatori, iar prin acest mod ne este foarte ușor să rezolvăm partea de acces.

În continuare o să vedem diagrama de activități a aplicației, iar apoi vom discuta mai pe larg ce funcționalități avem.



## Diagrama de activități a aplicației

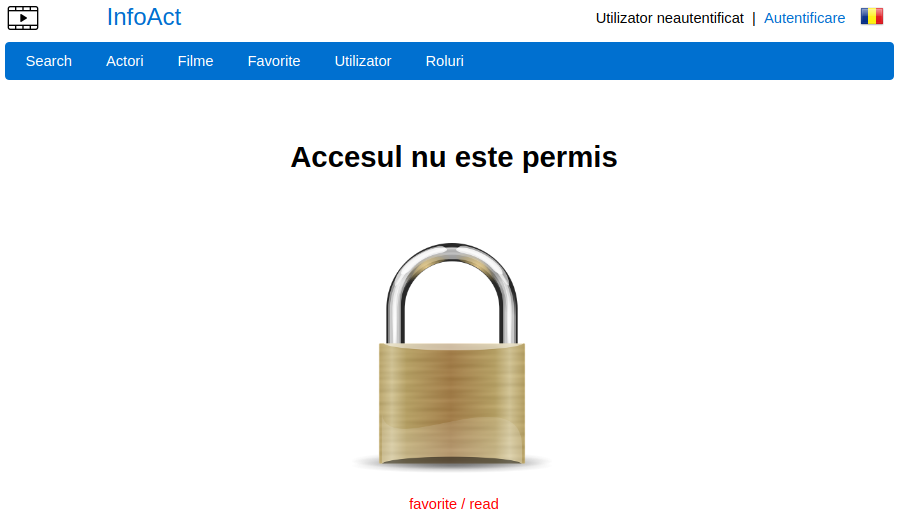
În momentul deschiderii aplicației, pe prima pagină apar câteva informații despre cum poate fi folosită și scopul acesteia. În partea dreaptă sus ne este pus la dispoziție un buton de logare. Dacă dorim să ne conectăm în aplicație, la acest punct ne apare un ecran în care putem introduce numele și parola. Lângă butonul de ‘Autentificare’ există și butonul de ‘Înregistrare’. La crearea unui cont ne sunt cerute următoarele informații: username-ul și parola prin care ne putem conecta în viitor, numele și adresa de email. Username-ul și email-ul trebuie să fie unice pentru fiecare utilizator din baza de date, de aceea există un verificator care te va anunța dacă acestea sunt deja folosite. În momentul înregistrării utilizatorul primește drepturile unui utilizator normal, urmând ca un moderator să îi acord un alt rol dacă este cazul. În momentul logării în aplicație este creată un HttpSession prin care vom reține utilizatorul. Sesiunea expiră după 30 de minute, deconectându-i pe cei care nu au executat nici o cerere la server în acest interval. Expirarea sesiunii este un mod simplu de a crește securitatea aplicației.

Pentru securitatea informațiilor utilizatorului, parolele din baza de date sunt ascunse folosind funcția de ‘hash’ SHA-256. Această funcție este urmașa lui SHA-1, cea din urmă fiind declarată nesigură deoarece cei de la Google au găsit o coliziune generată de acest algoritm, [10]. Găsirea unei ciocniri ne spune că un atacator se poate conecta chiar dacă nu știe parola corectă, ci folosind celălalt cuvânt din pereche. SHA-256 este mai sigură decât SHA-1 datorită dimensiunii mai mare a numărului de biți, 256 contra 160.

În momentul de față aplicația este gândită să cuprindă 4 tipuri de utilizatori: utilizator neconectat, utilizator înregistrat, moderator și admin. Astfel, un utilizator neconectat poate doar să caute un actor din imagine, să vadă informații despre acesta și despre filmele aflate în baza de date. Pe lângă toate cele prezentate, un utilizator înregistrat în aplicație are posibilitatea să salveze filmele favorite și să lase recenzii cu opinia. Moderatorul are dreptul și obligația de a modifica conținutul paginilor, iar adminul de a modifica drepturile și rolurile utilizatorilor.

Deoarece roluri au fost create dinamic, este foarte ușor să adăugăm unul nou. Un admin trebuie doar să intre în pagina de roluri, să adauge nume și să îi atribuie drepturile dorite. Fiecare ecran are doua drepturi, ‘read’ și ‘write’, care pot fi date sau revocate foarte ușor folosind acest panou.

Funcțiile din clasele controlor verificată drepturile utilizatorului conectat. De exemplu, pentru a vedea lista filmelor favorite, prima linie executată în controlor este o funcție care verifică dacă utilizatorul conectat în sesiune are dreptul de ‘read’ în ecranul ‘favorite’. Drepturile unui utilizator sunt citite din baza de date folosind legături între tabele pornind de la UTILIZATOR și ajungând la tabela DREPT. Dacă nu este găsit dreptul de ‘read’ în ecranul ‘favorite’, atunci este aruncată o excepție EroareAutorizare care va fi prinsă de Spring și ne va fi întoarsă următoarea pagină:



Ecranul de ‘EroareAutorizare’ al aplicației

În plus, aici apare motivul pentru care nu a fost lăsat să acceseze pagina, adică lipsa dreptului de ‘favorite / read’. Adminul poate decide să dea drepturi unui grup de utilizatori să facă o acțiune sau să le blocheze accesul.

Fiecărui utilizator al aplicației îi este permis să încarce o imagine pe server pentru a afla dacă un actor existent în baza de date a fost identificat. Încărcarea imaginilor pe server a fost realizată printr-o cerere de tip POST ‘multipart’. Imaginea este transferată în părți de câte 1024 octeți. Serverul primește cererea și încarcă imaginea într-un folder după ce numele este schimbat într-un UUID.

**Definiția 4.1**: *Un UUID este un identificator generat cu dimensiunea de 128 biți cu o probabilitate atât de apropiată de zero de a genera o coliziune încât este considerată neglijabilă.*

Locația folderului este stabilită într-un fișier cu extensia ‘.properties’ din care a fost creat un Spring Bean. După încărcarea imaginii în folder, serverul Java (port 8080) se conectează la serverul Python (port 8081), trimițându-i locația imaginii prin una din cele 2 servicii REST, prima folosind un model CNN+SVM pentru recunoaștere, pe când cea de-a doua un algoritm CNN+KNN. Modul de recunoaștere este stabilit înainte de către dezvoltator, apelând una din cele 2 metode puse la dispoziție. Ambele modele de recunoaștere încarcă imaginea primită și extrage fețele din imagine. Dacă numărul fețelor extrase este diferit de 1, este returnat ‘’, iar serverul Java va spune că imaginea introdusă nu este corectă.

Dacă numărul fețelor este 1, atunci sunt extrase cele 128 trăsături folosind librăria Face Recognition, iar apoi acest vector este trecut printr-un clasificator SVM sau KNN, în funcție de serviciul apelat. Clasificatorul ne returnează numele actorului din imagine, iar serverul Java va căuta actorul cu acel nume în bază de date și îl va trimite utilizatorului. Pentru a evita situația în care există 2 actori cu același nume, ar fi de preferat să antrenăm un model care întoarce identificatorul actorului respectiv din baza de date. În cazul acestei antrenări trebuie să avem mare grijă în momentul în care ștergem un anumit actor din oricare motiv (eroare de soft, greșeală a unui moderator), deoarece la reintroducerea acestuia trebuind să-i schimbăm câmpul ‘id’ din baza de date sau să antrenăm din nou rețeaua.



Ecranul de recunoaștere a actorului din imagine

Imaginea introdusă de utilizator apare în partea stângă. Aceasta este o imagine a lui Hugh Jackman din filmul Logan descărcată de pe Google . În partea dreaptă apare imaginea actorului luată de pe server și jos ne este spus numele acestuia, putând de aici să mergem pe pagina sa pentru a afla mai multe informații. Pentru a obține rezultate bune în căutare, imaginea feței actorului ar trebui să aibă o dimensiune de cel puțin 50x50, dar 90x90 ar fi recomandat pentru a pune cu adevărat în valoare rețeaua convoluțională. Fiecare față care trece prin model este redimensionată înainte de a intra în rețea, dar redimensionarea de la o rezoluție mică la una mai mare face ca imaginea să piardă din claritate, rezultând în rezultate mai proaste ale rețelei. Fețele care au fost folosite pentru a antrena rețeaua au avut o rezoluție de cel puțin 50x50.

În ecranul de ‘Actori’, primul lucru pe care îl poate vedea un utilizator este o listare a actorilor ordonați în ordinea alfabetică după nume. Fiecărui actor îi este precizat numele, o mică descriere și o imagine a acestuia. Pentru a afla mai multe informații, utilizatorul trebuie să intre pe pagina personală a actorului, acest lucru fiind realizat printr-o singură apăsare a mouse-lui pe nume. Toate rezultatele sunt paginate, fiind citite din baza de date doar cel mult 10 rezultate pentru a nu crește timpul de răspuns al aplicației. Pentru a afla mai multe rezultate, utilizatorul poate să folosească filtrul aflat în partea de sus a paginii pentru a căuta un actor după nume, precum și meniul de navigare prin pagini aflat în partea de jos. În plus, moderatorii au posibilitatea de a intra pe pagina de adăugare a unui nou actor de aici.

În pagina pagina personală a unui actor se regăsesc mai multe informații publice despre acesta precum: numele, o mică descriere, data și locul nașterii. În final apar o serie de imagini ale acelui actor încărcate pe site și o listă cu personajele pe care le-a jucat în filme. Acestea din urmă pot fi modificate de un moderator fără a trebui să șteargă actorul și să-l introducă din nou. Funcțiile de ștergere, adăugare și editare a unui actor sunt puse la dispoziție doar moderatorilor, aceștia trebuind să aibă grijă ca pagina să aibă un conținut la curent. Orice alt utilizator care va încerca să folosească aceste funcții va primi ca răspuns un mesaj de eroare.

Pagina de listare a filmelor este asemănătoare cu cea a actorilor. În ecranul de detaliu al unui film apar, pe lângă detaliile și actorii participanți, o diagramă a notelor date în recenzii, o listă a recenziilor și o steluță prin care putem adăuga un film la favorite. Spre deosebire de celelalte cereri la server, pentru adăugarea unui film la favorit am folosit o funcție JavaScript cu Ajax pentru a apela un serviciu RESTful în care are loc adăugarea sau a scoaterea un film de la favorit pentru un utilizator. Un serviciu asemănător este folosit și pentru a adăuga filme în care a participat un actor sau actori care au jucat într-un film. Ecranul de favorite este asemănător cu cel de filme, singura diferență fiind lista filmelor care apar.

Orice utilizator conectat are dreptul să scrie o recenzie unui film în care își poate exprima părerea despre acesta și îi poate acorda o notă. Adăugarea recenziilor ajută utilizatorii noi să aleagă un film pe care nu l-au văzut până atunci. Utilizatorul are dreptul să-și șteargă recenzia, dar nu poate adăuga mai mult de una pentru același film. Moderatorul are dreptul de a șterge orice recenzie dacă este de părere că are un conținut neadecvat precum cele în care sunt descrise scene din filmul respectiv, acestea deranjând persoanele care nu l-au văzut până atunci.

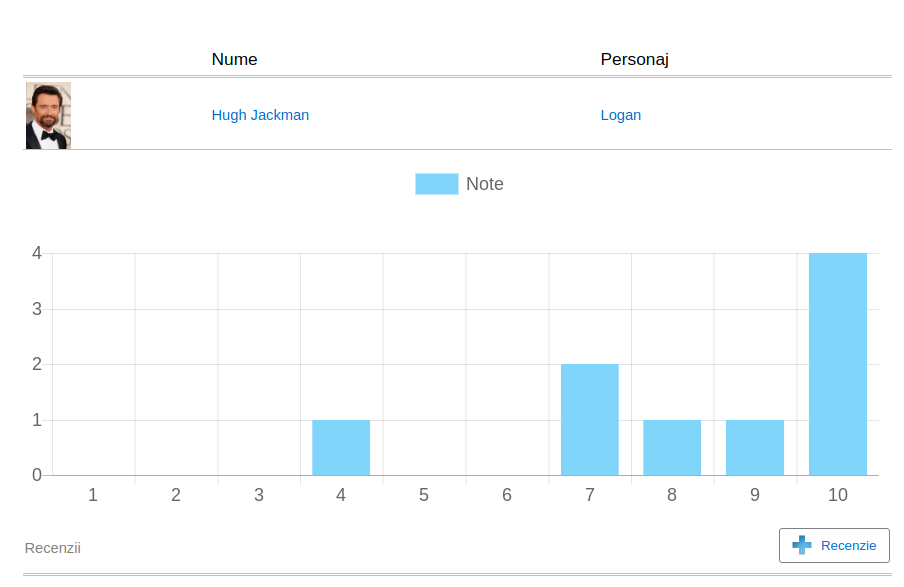


Diagrama notelor unui film

Diagrama notelor a fost realizată folosind librăria JavaScript Chart.js [21]. Deoarece această metodă a avut nevoie de un obiect JSON pentru crearea acestei diagrame, în funcția de vizualizare din clasa de control specifică filmelor a trebuit adăugată această operație. Toate notele au fost trecute într-o listă, care apoi a fost transformată într-o listă JSON folosind dependința Jackson și a fost trimisă în partea de interfață. JavaScript-ul a preluat această listă și a desenat o diagramă de tip bară albastră peste un obiect HTML Canvas.

**Capitolul 5**

# **5. Concluzii și dezvoltări ulterioare**

Am arătat că este posibil să obținem un model de recunoaștere facială cu o acuratețe foarte mare pornind de la un set de date construit special pentru această aplicație. Folosirea unei rețele convoluționale antrenate pe un alt set de date folosind o funcție de cost triplă și trecerea rezultatului printr-un clasificator elementar precum SVM sau KNN au dat rezultate foarte bune în antrenarea unui model diferit. În plus, am observat că sunt suficiente doar 70 poze pe actor pentru a obține acuratețea de 99,73%, restul de 30 fiind folosite pentru testare.

Primul lucru care trebuie îmbunătățit în dezvoltările ulterioare este numărul relativ mic de actori pe care aplicația îi poate recunoaște. Am arătat că pentru a obține o acuratețe bună ne sunt suficiente cel puțin 70 pe poze pe individ. Găsirea acestui număr de poze pentru o persoană celebră este destul de simplă, deși toată munca care trebuie depusă pentru a adăuga un număr mare de actori este destul de consumabilă din punct de vedere al timpului și monotonă. În plus, adăugarea unei prag pe care nu ar trebui să îl depășească distanța dintre imaginea propusă și persoana prezisă îi oferă modelului posibilitatea de a spune dacă actorul introdus pentru a fi recunoscută apare în setul de antrenare sau nu. Decât o predicție proastă este mai bine să-i spunem utilizatorului că nu putem recunoaște acel actor. Totuși, valoarea acestui prag trebuie găsită astfel încât acuratețea rețelei să nu scadă prea mult, introducerea unei noi clase (persoană necunoscută) influențând și predicțiile pe setul testat în trecut.

Un alt lucra care ar putea fi luat în calcul este numărul relativ mic de funcționalități ale aplicației web. Pe viitor se pot implementa lucruri care pot care să asigure o mai bună comunicare între utilizatori, acest lucru fiind atrăgător pentru cei care doresc să discute cu mai multe persoane pe seama unor filme. În acestă direcție se poate crea un forum în care utilizatorii pot pune întrebări, raporta eventuale probleme ale aplicației sau discuta anumite subiecte. În plus, funcția de editare a profilului, adăugarea pozelor de avatar și a descrierilor, precum și posibilitatea de exportare a listei filmelor favorite oferă utilizatorilor posibilitatea să petreacă mai mult timp pe aplicație și să distribuie ușor lista preferințelor altor persoane.

Funcția de favorit a filmelor poate fi împărțită în 2: filme care doresc să le văd în viitor și filme pe care le-am văzut și vreau să le am salvate în cazul în care vreau să le revăd în viitor. Un alt ecran care ar trebui adăugat este unul în care utilizatorul poate vedea o listă a recenziilor date de el.

Nu în ultimul rând, o versiune mobilă a aplicației ar putea crește numărul utilizatorilor cu mult. Un exemplu este acela în care observi o reclamă la un televizor și vrei să afli cine este persoana aceea. Atunci ai posibilitatea să faci o poză cu camera video a telefonului și să o încarci în aplicație pentru a afla identitatea actorului.

# **6. Bibliografie**

[1] Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. *ImageNet Classification with Deep Convolutional Neural Networks*. Proceedings of NIPS, pp. 1106–1114, 2012.

[2] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. *Dropout: a simple way to prevent neural networks from overfitting.* J. Machine Learning Res. 15, 1929–1958, 2014.

[3] Simonyan, K. & Zisserman, A. *Very deep convolutional networks for large-scale image recognition*. In Proc. International Conference on Learning Representations http://arxiv.org/abs/1409.1556, 2014.

[4] Vincent Dumoulin & Francesco Visin. *A guide to convolution arithmetic for deep learning.* arXiv:1603.07285, 2016.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. *Deep Residual Learning for Image Recognition*. arXiv:1512.03385, 2015.

[6] G. Tsai. *Histogram of oriented gradients*. University of Michigan, 2010.

[7] Navneet Dalal, Bill Triggs. *Histograms of Oriented Gradients for Human Detection*. Cordelia Schmid and Stefano Soatto and Carlo Tomasi. International Conference on Computer Vision & Pattern Recognition (CVPR ’05), Jun 2005, San Diego, United States. IEEE Computer Society, 1, pp.886–893, 2005, . <10.1109/CVPR.2005.177>.

[8] F. Suard, A. Rakotomamonjy, A. Bensrhair, and A. Broggi. P*edestrian detection using infrared images and histograms of oriented gradients*. In IEEE Intelligent Vehicles Symposium, 2006.

[9] CORINNA CORTES & VLADIMIR VAPNIK. *Support-Vector Networks*. 1995 Kluwer Academic Publishers, Boston, 1995.

[10] https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html

[11] Radu Tudor Ionescu & Marius Popescu. *Knowledge transfer between computer vision and text mining*. *Similarity-based Learning Approaches.* Springer, 2016.

[12] Dasarathy BV. *Nearest neighbor (NN) norms: pattern classification techniques.* IEEE Computer Society Press, Los Alamitos, 1991.

[13] Faragó A, Linder T, Lugosi G. *Fast nearest-neighbor search in dissimilarity spaces*. IEEE Trans Pattern Anal Mach Intell 15(9):957–962, 1993.

[14] Florian Schroff, Dmitry Kalenichenko, James Philbin. *FaceNet: A Unified Embedding for Face Recognition and Clustering*. arXiv:1503.03832, 2015.

[15] https://en.wikipedia.org

[16] https://medium.com/

[17] http://cs231n.stanford.edu/

[18] https://pypi.org/project/face\_recognition/

[19] http://www.google.ro

[20] https://maven.apache.org

[21] https://www.chartjs.org/