PROIECT DE LICENŢĂ

Aplicaţie web pentru căutarea informaţiilor despre actori pe bază de recunoaştere facială

Cuprins

Capitolul 1

Introducere

În ciuda faptului că inteligența artificială s-a dezvoltat într-un pas foarte alert în ultimii ani, nu foarte multe

In această lucrare vom folosi o rețea neuronală convoluțională pentru a 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 în urma cautărilor pe platforma IMDB din anul 2017.

Rețelele neuronale convoluționale au avut un succes imens în ultima jumătate de deceniu, totul pornind de la rețeaua propusă de Alex Krizhevsky, [1]. După aceea, Facebook prin DeepFace, , și Google prin FaceNet, , au dus la un alt nivel recunoașterea facială, rețelele lor obținând o acuratețe mai bună decât cea umană.

Astfel, în capitolul 2 vom prezenta conceptele teoretice care stau la baza realizării recunoașterii faciale din imagini, vom discuta mai pe larg în ce constau cele 2 rețele propuse de Facebook și Google, iar apoi vom preciza rezultatele obținute folosind o rețea preantrenată pe setul de date al nostru, precum și o comparare cu un algoritm de bază.

În capitolul 3 prezentăm tehnologiile folosite în realizarea aplicației web,

Capitolul 4 este dedicat descrierii aplicației, unde vor fi prezentate cazuri de utilizare, diagrame de activități, structura bazei de date.

Capitolul 2

Recunoaşterea facială din imagini

2.1. Concepte teoretice

2.1.1. Reţele neuronale convoluţionale

Reţelele neuronale convoluţionale sunt o categorie a reţelelor neuronale care s-au dovedit să aibă un succes foarte mare în recunoaşterea si clasificarea feţelor si obiectelor din imagini. Faţă de reţelele normale, cele convoluţionale au avut un succes mai mare in aceste domenii datorită faptului că este păstrată spaţialitatea imaginii prin folosirea filtrelor.

Reţelele convoluţionale sunt foarte asemănătoare cu cele neuronale ordinare prin faptul că sunt compuse din neuroni în care sunt reţinute informaţii. Fiecare neuron primeşte ca date de intrare un vector de numere, realizează o operaţie de inmulţire cu vectorul de ponderi şi apoi este adunat un parametru numit bias. Astfel, formula arată în felul următor:

$$y = \sum_{i=1}^{n} x_i * w_i + b \tag{2.1}$$

De preferat, această operaţie ar trebui să fie urmată de o funcţie neliniară. Acest lucru este datorat faptului că dacă am înlanţui mai multe funcţii liniare precum cea din formula (2.1), tot acest lanţ ar putea fi înlocuit cu o singură funcţie liniara. De aceea, fiecare funcţie liniară este urmată de una neliniară, astfel evitând cazul prezentat. În final, formula va arată astfel:

$$y = f\left(\sum_{i=1}^{n} x_i * w_i + b\right) \tag{2.2}$$

De-a lungul anilor, au fost folosite mai multe funcţii neliniare, denumite şi funcţii de activare, în acest domeniu. Prima funcţie pe care o vom prezenta este funcţia **Sigmoid**, notată σ. Aceasta are următoarea formulă matematică şi are următoarea distribuţie în jurul punctului 0.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

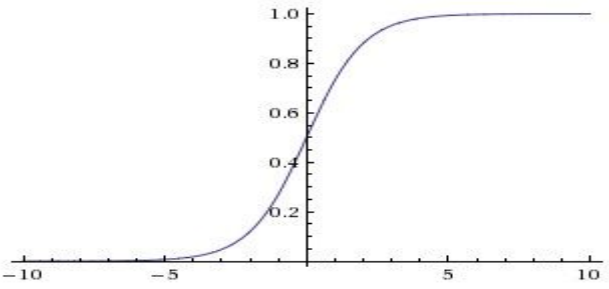

Figura 2.1: Reprezentarea grafică a funcţiei sigmoid

Funcţia sigmoid a fost folosită în trecut datorită faptului că reuşeste să restrângă variabila în intervalul [0, 1]. În practică, această funcţie este folosită foarte rar deoarece are câteva mari dezavantaje. Primul problemă este aceea că sigmoid saturează în jurul punctelor 0 şi 1, gradientul în aceste regiuni fiind aproape 0. Astfel, în propagarea înapoi a reţelei, semnalul care v-a trece prin neuron va fi foarte mic şi nu vor aparea schimbări prea mari în ponderile acestuia, reţeaua invaţând foarte puţin. De aceea, este necesar sa fim foarte atenţi când iniţializăm ponderile neuronilor care folosesc sigmoid pentru a evita saturarea.

Un alt dezavantaj este acela ca rezultatele funcţiei sigmoid nu sunt centrate în 0. Acest lucru nu este dorit datorită faptului că dacă toate datele de intrare ale unui neuron sunt pozitive, atunci gradientul ponderilor în pasul de propagare înapoi va avea toate valorile ori pozitive, ori negative, depinzând de gradientul total al funcţiei. In plus, funcţia exponenţială este scumpă din punct de vedere computaţional.

Funcţia **Tangentă** este o funcţie de activare care restrânge valorile în intervalul [-1, 1]. Precum funcţia sigmoid, tangenta saturează şi este o computaţional scumpă, dar valorile rezultate sunt centrate în 0. De aceea, tangenta este mereu preferată faţă de sigmoid în practică. Un lucru interesat care poate fi observat este că tangenta poate fi formată dintr-o funcţie sigmoid:

$$\tanh(x) = 2 * \sigma(2 * x) + 1 \tag{2.4}$$

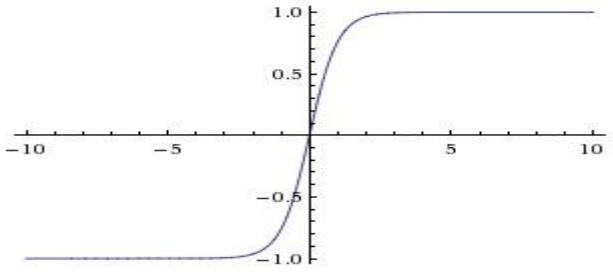

Figura 2.2: Reprezentarea grafică a funcţiei tangentă

În continuare, vom prezenta funcţia de activare **ReLU**. Aceasta a fost propusă pentru a fi folosită în reţelele neuronale de către Alex Krizhevsky în [1]. Aici, ne este indicat să folosim ReLU în detrimentul tangentei, acurateţea ramânând aceeaşi, deşi este de 6 ori mai rapidă. Această funcţie are urmatoarea formulă:

$$f(x) = \max(0, x) \tag{2.5}$$

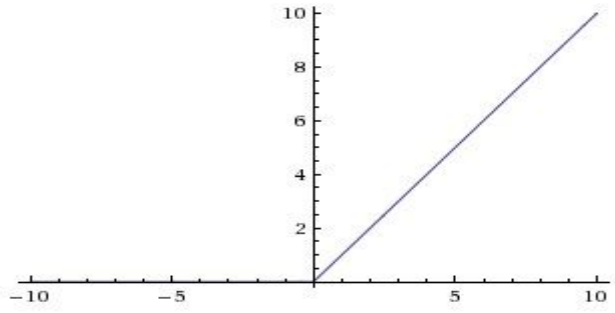

Figura 2.3: Reprezentarea grafică a funcţiei ReLU

O mare problemă care poate apărea în antrenarea reţelelor neuronale este cazul în care reţeaua învaţă 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 şi simplă de 'adormire' a neuronilor în [2]. Aceasta ne spune ca în momentul antrenării să avem o probabilitate pentru a 'adormi' un neuron. Astfel, la fiecare pas de propagare înainte, sunt decişi care neuroni vor partifica în acel pas şi doar ei vor fi actualizaţi la propagarea înapoi, restul fiind setaţi să aibă valoarea 0 în acea rundă.

Mai mulţi neuroni interconectaţi care conţin o funcţie de activare formează un strat. O reţea neuronală este compusă din mai multe astfel de straturi complet conectate între ele şi un strat final care ne va returna răspunsul.

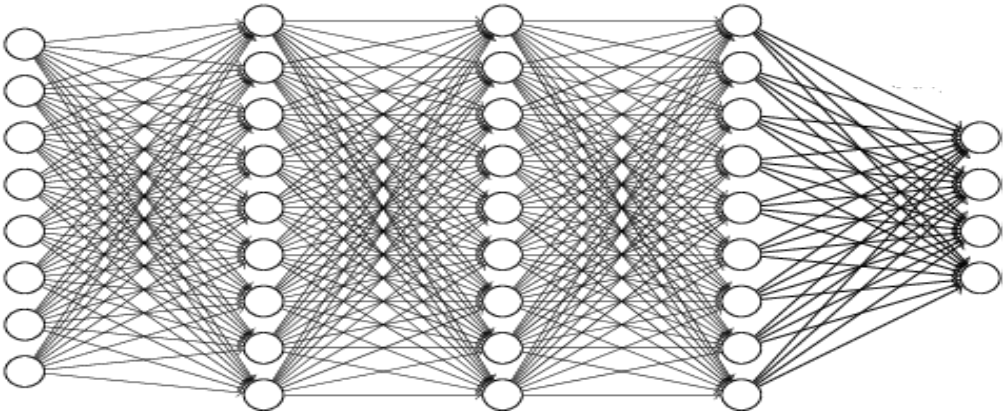

Figura 2.4: Reprezentarea grafică a unei reţele neuronale

Ultimul strat al unei reţele este unul de decizie. Aici va avea loc prezicerea asupra cărei clase aparţin datele de intrare. Cea mai populară funcţie este **Softmax**. Ea primeşte la intrare un vector de dimensiune n de numere reale şi returnează un vector de numere reale între 0 si 1, având suma totală egală cu 1. Astfel, valoarea vectorul de ieşire poate fi gândită ca 'probabilitatea ca un element să aparţină acelei clase'. Funcţia softmax are urmatoarea formulă:

$$\sigma(x)_j = \frac{e^{x_j}}{\sum_{k=1}^{n} e^{x_k}} \qquad (2.6)$$

În final, vom discuta despre funcţia de cost. Această funcţie ne spune cât de bine am aproximat rezultatul final. Astfel, putem vedea cât de aproape am fost de răspunsul corect şi putem modifica valorile neuronilor. Rezultatul acestei funcţii este numită eroarea în acel pas. Cea mai cunoscută funcţie de cost este aceea în care calculăm pătratul diferenţei dintre ce am prezis noi şi răspunsul corect pentru fiecare clasă, iar apoi le adunăm pe toate pentru a obţine eroarea totală. În funcţie de tipul problemei pe care încercăm sa o rezolvăm, pot apărea diferite funcţii de cost.

Valoarea funcţiei de cost este influenţată de ponderile neuronilor. Astfel, pentru minimizarea erorii, ponderile sunt modificate în pasul de propagarea înapoi. Aceste modificări au loc folosind o funcţie de optimizare. În general, această funcţie calculează gradientul unui neuron, adică derivata funcţiei de cost în raport cu ponderile, iar apoi ponderile sunt modificate în direcţia opusă acestui gradient calculat.

În acest pas avem toate componentele necesare pentru a construi o reţea neuronală ordinară. Toate aceste lucruri vor apărea şi în reţelele neuronale convoluţionale.

Principala diferenţă între o reţea convoluţională şi una ordinară este faptul că ştim că la intrare vom primi o imagine. Fiecare imagine poate fi gândită ca o matrice de numere cuprinse între 0 şi 255. Un canal este termenul folosit pentru a ne referi la o anumită componentă a unei imagini. O imagine color are 3 canale (roşu, verde şi 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 şi 255, 0 reprezentând negru şi 255 alb.

O reţea convoluţională este formată dintr-o înşirare de straturi. Totuşi, spre deosebire de reţelele ordinare care folosesc doar straturi complet conectate între ele (fig. 2.4), cele convoluţionale mai introduc 2 tipuri de straturi: convoluţionale şi de unificare.

Stratul convoluţional este cea mai importantă componentă a unei reţele convoluţionale. Principalul rol al unei convoluţii este să extragă trăsături din imagine. Convoluţiile au proprietatea de a păstra legătura spaţială dintre pixeli, fiind folosite patrate de dimensiune mici, numite filtre. Filtrul va porni din partea stânga-sus şi va parcurge imaginea de la stânga la dreapta şi apoi de sus în jos. La fiecare pas, este efectuat produsul dintre filtru şi submatricea acoperită de filtru, iar apoi este efectuată suma componentelor matricei rezultate.

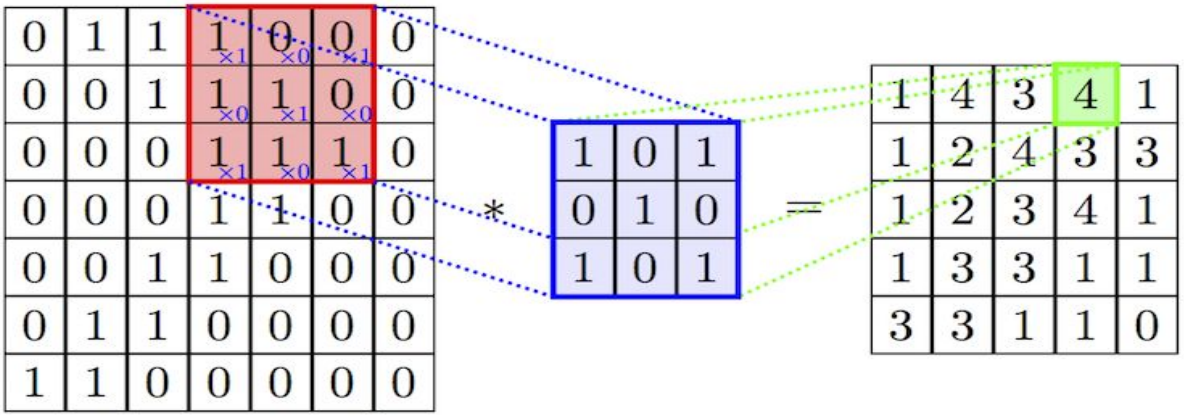

Figura 2.5: Reprezentarea grafică a unui pas de convoluţiei

Pentru fiecare strat convoluţional trebuie să decidem ce dimensiune va avea filtru, peste câţi pixeli sa săra la fiecare pas şi de unde să înceapă. Conform [3], filtrele ar trebui să aibă dimensiunea de 3x3, aceasta fiind cea mai mică dimensiune care păstrează noţiunile de stânga-dreapta, sus-jos şi centru. Filtrele de dimensiuni mari ar trebui să fie schimbate cu mai multe filtre de dimensiune 3x3 consecutive. Astfel, 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 funcţia de decizie să fie mai discriminativă. În al doilea rând, campul vizual al unui strat convoluţional 7x7 este acelaşi precum cel al 3 convoluţii consecutive 3x3. Nu în ultimul rând, un strat de dimensiune 7x7 ar avea nevoie de 7 x 7 x D x D parametri, D reprezentând numărul de canale, pe când cele 3 straturi 3x3 ar avea nevoie de doar 3 x (3 x 3 x D x D). Astfel, stratul 7x7 ar avea nevoie de 81% mai mulţi parametri decât cele de 3x3, ne spune [3].

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

Uneori ne va fi de folos să creştem dimensiunea ferestrei, iar acest lucru se poate realiza prin adăugarea de zerouri în jurul imaginii. Acest lucru ne permite să controlăm dimensiunea matricei de trăsături rezultată.

Când construim o reţea convoluţională, trebuie sa avem grijă la dimensiunea ferestrei care va rezulta după un strat convoluţional. Astfel, dimensiunea matricei de trăsături rezultate în urma stratului de convoluţie poate fi calculată astfel:

$$D = \frac{W - F + 2 * P}{S} + 1 \qquad (2.7)$$

, unde D reprezintă dimensiunea matricei de trăsături, W dimensiunea înălţimea/lăţimea ferestrei de intrare, F dimensiunea filtrului, P numărul de linii de zerouri adăugat, iar S dimensiunea pasului de deplasare.

Stratul de unificare este folosit pentru a reduce dimensiunea matricei de intrare, dar păstrand cea mai importantă informaţie. Conform [4], cea mai folosită metodă de unificare este cea a maximului. Prin aceast procedeu, matricea de trăsături este împărţită zone nesuprapuse şi este extras maximul din fiecare petic.

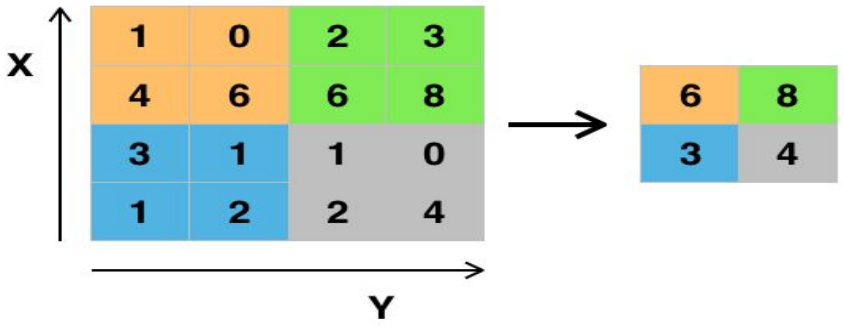

Figura 2.6: Strat de unificare prin extragerea maximului

Există şi alte metode de unificare, precum calcularea mediei sau a sumei. Precum în straturile convoluţionale, şi în cele de unificare trebuie să avem grijă la dimensiunea matricei de trasături rezultate. În [4] ne este prezentată formula care ne returnează dimensiunea matricei finale, aceasta fiind aceeaşi indiferent de tipul de unificare:

$$D = \frac{W - F}{S} + 1 \qquad (2.8)$$

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

Se poate observa ca formula 2.8 este un caz particular ar celei 2.7, atunci când P = 0. Acest lucru are sens, deoarece în straturile ne unificare nu sunt adăugare rezouri pentru a controla dimensiunea matricei de trăsături.

În final, o reţea neuronală convoluţională va fi compusă din mai multe straturi convoluţionale, printre care apar straturi de unificare şi la final unul sau mai multe straturi complet conectate între ele, încheiate cu un strat de softmax pentru a ne rezulta probabilităţile de apartenenţă pentru fiecare clasă.

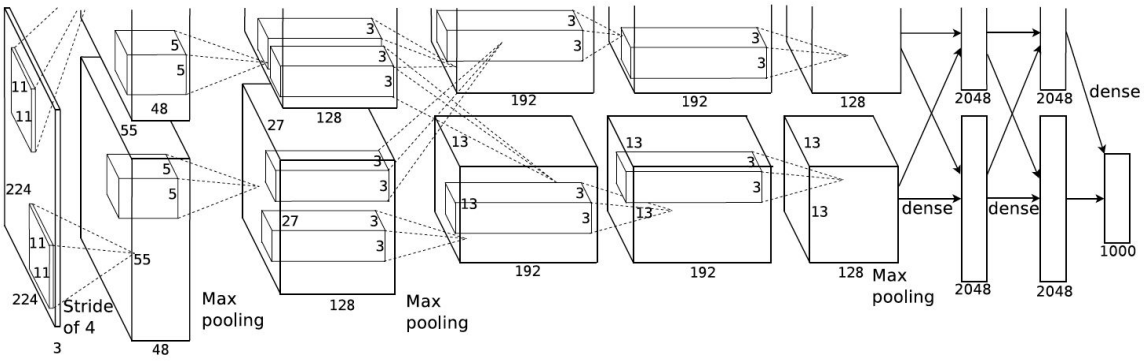

Figura 2.7: Reţeaua AlexNet, propusă de Alex Krizhevsky în [1]

Alex Krizhevsky, Ilya Sutskever şi Geoffrey Hinton au şocat lumea în 2012, când au prezentat o reţea neuronală convoluţională care a câştigat concursul ILSVRC, obţinând o eroare de doar 15.4% pe primele 5 predicţii, faţă de precedenta 26.2%. Acesta a fost primul model care a avut un astfel de succes pe setul de date foarte dificil ImageNet. Pornind de la ideile prezentate de cei 3 în [1], an de an au apărut reţele care au bătut recordul precedent. În 2015, Microsoft a ajuns la o eroare pe primele 5 predicţii de 3.56% prin reţeaua ResNet, prezentată în [5].

2.1.2. Histograma gradienţilor orientaţi (HOG)

Histograma gradienţilor orientaţi a fost folosită pentru a extrage trăsăturile unui obiect din imagini. Spre deosebire de o reţea neuronală convoluţională care învaţă de una singură cum să extragă trăsături, această metodă are un algoritm predefinit pentru a calcula aceste date de ieşire. Motivul pentru care, în general, reţelele convoluţionale funcţionează mai bine decât descriptorul HOG este acela că reţelele învaţă cum trebuie să extragă trăsăturile pentru a obţine o acurateţe cât mai bună pe un anumit set de date, pe când descriptorul HOG are acelaşi mod de extragere a trăsăturilor, indiferent de input. De aceea, când avem un set de date cu multe imagini, este de preferat să antrenăm o reţea deoarece aceasta va avea acurateţe mai bună aproape de fiecare dată. Totuşi, dacă avem un set de date mic şi nu avem posibilitatea să creştem numărul de imagini, este posibil ca metoda histogramei gradienţilor orientaţi sa ofere o acurateţe mai bună.

Principala ideea din spatele histogramei gradienţilor orientaţi este aceea că aspectul şi forma locală a unui obiect poate fi descrisă de distribuirea intensităţii gradienţilor sau de orientarea marginilor, [6]. Imaginea este împarţită în regiuni conectate numite celule, iar pixelii din fiecare celulă vor determina o histogramă a direcţiei gradienţilor. Descriptorul este o concatenare a acestor histograme.

Definiţia 2.1: *Gradientul unei imagini este schimbarea direcţiei în intensitate sau culoare dintr-o imagine.*

Una dintre cele mai obişnuite utilizări a gradienţilor este în detectarea muchiilor unei imagini.

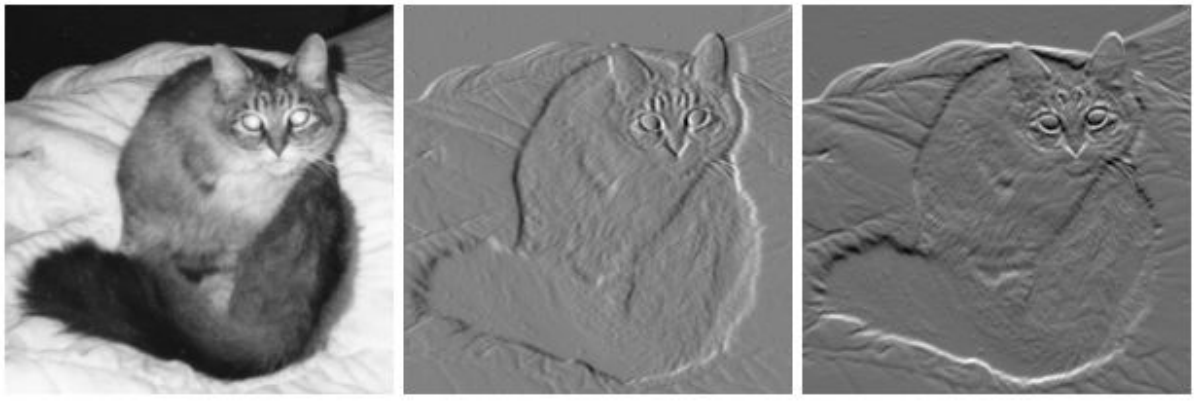

Figura 2.8: Gradienţii unei imagini

În partea stângă avem imaginea alb-negru a unei pisici. În centru avem imaginea gradienţilor în direcţia x, măsurând schimbările orizontale în intensitate. În partea dreaptă avem imaginea gradienţilor în direcţia y, măsurând schimbările verticale în intensitate. Astfel, pixelii gri au o valoare mică a gradienţilor, pe când pixeli albi şi negri au o valoare ridicată.

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

$$\nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}, \tag{2.9}$$

unde $\frac{\partial f}{\partial x}$ este derivata în funcţie de x, adică gradientul pe direcţia x, pe când $\frac{\partial f}{\partial y}$ este derivata în funcţie de y, reprezentând gradientul pe direcţia y.

Direcţia gradientului este dată de:

$$\theta = tanh^{-1}\left[\frac{g_x}{g_y}\right] \tag{2.10}$$

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

$$G = \sqrt{g_x^2 + g_y^2} \tag{2.11}$$

Una dintre cele mai simple metode de a găsi gradienţii unei imagini este prin folosirea unui filtru Sobel. Pentru găsirea gradientului pe direcţia x, filtrul Sobel va parcurge imaginea precum un filtru dintr-un strat convoluţional al unei reţele neuronale convoluţionale. Pentru găsirea gradientului pe direcţia y, vor aplica aceeaşi metoda, dar vom folosi transpusa filtrului Sobel. Pentru a obţine gradienţii pe ambele direcţii, este suficient să trecem cu primul filtru pe imaginea iniţială, iar apoi cu al doilea filtru pe imaginea rezultată după aplicarea primului filtru. Imaginea finală va fi folosită în construirea histogramei gradienţilor orientaţi. În acest pas este calculată direcţia şi magnitudinea gradienţilor.

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

$$\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}.$$

Figura 2.9: Componentele filtrului Sobel pe direcţia x

Alte filtre pot fi folosite pentru a obţine gradienţii unei imagini. În [7], histograma gradienţilor orientaţi a fost folosită pentru a detecta pietonii dintr-o intesercţie. Aici, filtrul care a dat cele mai bune rezultate a fost unul de dimensiune 1x3, având valorile [-1, 0, 1].

După aflarea gradienţilor şi împărţirea imaginii pe celule, urmează să construim histograma în fiecare dintre aceste zone. Matricea direcţiei gradienţilor va conţine valori cuprinse între 0 şi 179, reprezentând măsura unghiului dintre direcţia gradientului şi axa OX în grade. Astfel, histograma va împărţii aceste unghiuri în porţiuni, iar fiecare pixel îşi va aduna valoarea magnitudinii în componenta corespunzătoare direcţiei sale. După ce toţi gradienţii pixelilor unei celule au fost parcurşi, histograma acelei celule va fi completă. Histograma gradienţilor orientaţi ale unei imagini rezultă din concatenarea histogramelor tuturor celulelor acelei imagini.

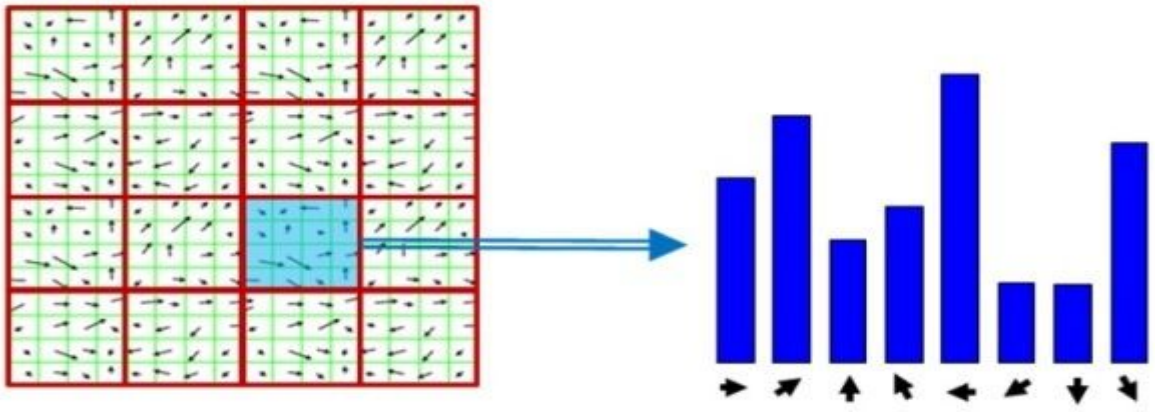

Figura 2.9: Reprezentarea grafică a histogramei unei celule

2.1.3. Maşini cu suport vectorial (SVM)

Maşinile cu suport vectorial sunt modele de învăţare supervizată folosite în clasificare şi regresie. Conform [10], maşinile cu suport vectorial implementează următoarea idee: vectorii primiţi ca date de intrare sunt mapaţi neliniar într-un spaţiu cu dimensiune foarte mare. În acest spaţiu, o suprafaţă liniară de decizie este construită. Proprietăţile speciale ale suprafeţei de decizie ne asigură o generalizare mare a maşinilor de învăţare.

Definiţia 2.2: *Învăţarea supervizată în inteligenţa artificială este problema învăţării unei funcţii care mapează datele de intrare pe unele de ieşire plecând de la mai multe exemple de intrare-ieşire.*

SVM-ul este printre cele mai bune(şi mulţi cred că este cel mai bun) algoritm de învăţare supervizată standard, [11]. Acesta este un clasificator discriminatoriu definit de un hiperplan separator.

Definiţia 2.3: *În geometrie, un hiperplan este un subspaţiu a cărui dimensiune este mai mică cu 1 decât a spaţiului ambiental.*

Dacă spaţiul este de dimensiune 3, atunci hiperplanele sale au dimensiunea 2, iar dacă spaţiul are dimensiunea 2, atunci hiperplanele sunt drepte de dimensiunea 1.

Definiţia 2.4: *Spaţiul ambiental este spaţiul care înconjură un obiect.*

Pornind de la un set de date de antrenare, fiecare exemplu aparţinând uneia dintre 2 clase, un algoritm de antrenare SVM crează un model care atribuie o nouă intrare la una dintre cele 2 clase, formând un clasificator binar liniar neprobabilistic. Un model de maşină cu suport vectorial este o reprezentare a datelor ca puncte în spaţiu, fiind mapate astfel încât cele aparţinând categoriilor diferite să fie despărţite de un gol clar care este cât mai mare posibil.

Pe lângă faptul că efectuează clasificări liniare, maşinile cu suport vectorial pot executa clasificări neliniare folosind ceea ce se numeşte trucul nucleului, mapând implicit datele sale de intrare într-un spaţiu al trăsăturilor cu dimensiune foarte mare.

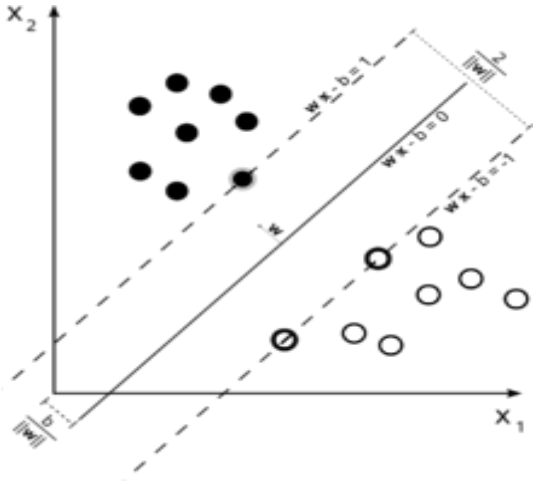

Figura 2.10: Reprezentarea grafică a unui SVM liniar

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

Învaţarea hiperplanelor într-un SVM liniar este efectuată folosind algebră liniară.

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

$$(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)$$

, 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. Vrem să aflăm hiperplanul care împarte grupul punctelor pentru care y = -1 de grupul punctelor cu y = 1 astfel încat distanţa dintre hiperplanul ales şi cel mai apropiat punct x din oricare dintre cele 2 grupe este maximă.

Orice hiperplan poate fi scris ca un set de puncte satifăcând urmatoarea relaţie:

$$\vec{w} \cdot \vec{x} - b = 0, \tag{2.12}$$

, unde w este vectorul normal al hiperplanului.

Definiţia 2.5: *În geometrie, vectorul normal este vectorul perpendicular pe un obiect dat.*

Construirea modelului SVM este influenţată de 2 paremetri:

- parametrul de **Regularizare** (notat cu C în libraria Scikit-Learn pentru Python) spune clasificatorului cât de mult vrei să te fereşti de clasificarea greşită a fiecărei date de antrenare. Pentru valori mari ale lui C, optimizatorul va alege un hiperplan de dimensiune mai mică dacă acesta va face o treabă mai bună în a clasifica punctele de antrenare corect. Pe de altă parte, o valoare mică a lui C va face optimizatorul să caute un hiperplan de lungime mai mare, chiar dacă acel hiperplan va clasifica greşit câteva puncte. Problema cu alegerea unui C cu valoare mare este aceea că este posibil ca SVM-ul să nu prezică corect datele de testare. De aceea, trebuie să descoperim valoarea variabilei C în funcţie de fiecare set de date.
- parametrul **Gamma** defineşte care puncte influenţează crearea hiperbolei de separare a claselor. Un gamma mic va face ca şi punctele aflate la o distanţă mai mare de posibila hiperbolă să fie luate în vedere în calcularea soluţiei. Pe de altă parte, un gamma mare va spune să fie folosite doar punctele din apropiere.

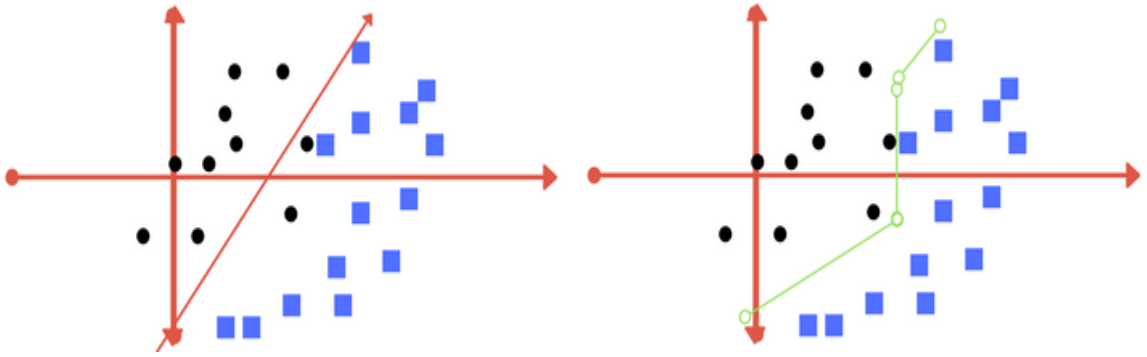

Figura 2.11: În stânga SVM-ul are valoarea parametrului de regularizare mică.
În dreapta SVM-ul are valoarea parametrului de regularizare mare.

În continuare vom discuta despre margine în maşinile cu suport vectorial. SVM-ul încearcă să obţină o margine cât mai bună.

Definiţia 2.6: *În inteligența artificială, marginea unui punct este definită ca distanța dintre acel punct și limita de decizie.*

Definiţia 2.7: *Limita de decizie este regiunea în care rezultatul unui clasificator este ambiguu.*

Pornind de la un set de date, este natural să căutăm o limită de decizie care să maximizeze marginea deoarece acest lucru va produce un set de predicţii despre care avem mai mare încredere că sunt corecte, [10]. În special, acest lucru va face clasificatorul să separe exemplele pozitive de cele negative eficient.

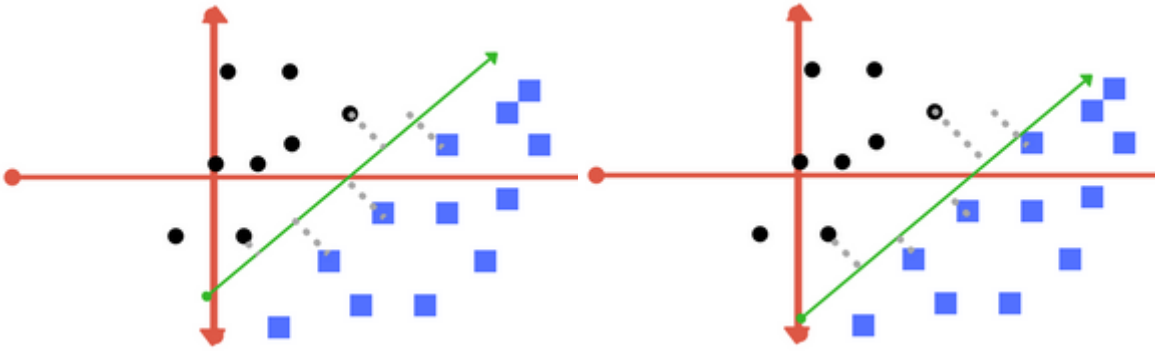

Figura 2.12: În stânga marginea este bună.
În dreapta marginea nu este aleasă bine.

Marginea din imaginea stângă este bună deoarece aceasta a fost aleasă cât de echidistant s-a putut faţă de cele două părţi. Pe de altă parte, imaginea din dreapta este foarte apropiată de pătrate.

2.1.4. Metoda celor mai apropiaţi k vecini (KNN)

Deşi a fost introdusa acum mai bine de 60 de ani, metoda celor mai apropiaţi k vecini este un topic activ în inteligenţa artificială, susţinând faptul că metodele simple sunt mereu atractive, [11]. Aceast algoritm este folosit pentru clasificare şi regresie.

În construirea clasificatorului KNN avem nevoie de 3 componenete:

- setul de date pentru antrenare cu clasele corespunzătoare
- metoda prin care se va calcula distanţa între 2 înregistrari. Cea mai folosită metrică în inteligenţa artificială este distanţa Euclidiana, aceasta având următoarea formulă pentru 2 puncte p = $(p_1, p_2, ..., p_n)$ şi q = $(q_1, q_2, ..., q_n)$

$$d(p,q) = d(q,p) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \qquad (2.13)$$

- variabila k, reprezentând numărul de vecini care sunt consideraţi în prezicere. Această variabilă este foarte importantă în vederea acurateţii pe care o vom obţine de la clasificatorul KNN. Dacă valoarea este prea mică, atunci acesta poate avea probleme să clasifice corect setul de testare. În plus, alegerea unui k mic face ca modelul să fie foarte sensibil la presupuse date de intrare eronate. Pe de altă parte, dacă alegem un k prea mare putem ajunge la situaţia în care decizie să fie luată de vecinii aflaţi la o distanţă mare faţă de punctul curent, aceştia fiind irelevanţi pentru etapa curentă.

Algoritm

Fie x un punct pentru care trebuie să prezicem clasa în care aparţine. Iniţial, vom calcula distanţele de la x la toate celelalte puncte din setul de antrenare şi vom sorta vectorul rezultat crescător. Dacă k = 1, atunci clasa lui x va fi aceeaşi 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.

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 distanţele şi să afle care sunt primele k dintre acestea. Totuşi, alţi algoritmi mai rapizi au fost

propuşi de Belur Dasarathy în [12], bazat pe arbori de căutare multidimensionali care partiţionează spaţiul şi ghidează cautarea, precum şi de Farago în [13].

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

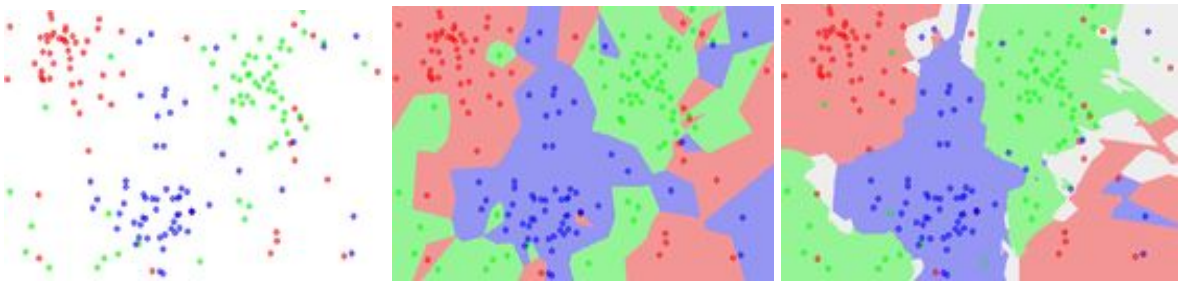

Figura 2.13: De la stânga la dreapta: setul de date, reprezentare grafică KNN pentru k = 1, reprezentare grafică KNN pentru k = 5.

În partea stângă avem setul de date colorat diferit în funcţie de clasa de care aparţine. Îmaginea din centru ne prezintă limitele de decizie ale clasificatorului KNN cu k = 1 folosind distanţa 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 au clasa rosie, 2 au clasa albastră şi ultimul are clasa verde).

2.2. Metode recente pentru recunoaşterea facială din imagini