

# **PROIECT DE LICENȚĂ**

**Aplicație web pentru căutarea informațiilor despre actori pe bază de  
recunoaștere facială**

# Cuprins

<b>Capitolul 1</b>	<b>3</b>
<b>Capitolul 2</b>	<b>5</b>
2.1. Concepte teoretice	5
2.1.1. Rețele neuronale convoluționale	5

# Capitolul 1

## Introducere

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

În 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 căutărilor 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

### Recunoașterea facială din imagini

#### 2.1. Concepte teoretice

##### 2.1.1. Rețele neuronale convoluționale

Rețelele neuronale convoluționale sunt o categorie a rețelilor neuronale care s-au dovedit să aibă un succes foarte mare în recunoașterea și clasificarea fețelor și obiectelor din imagini. Față de rețelele normale, cele convoluționale au avut un succes mai mare în aceste domenii datorită faptului că este păstrată spațialitatea imaginii prin folosirea filtrelor.

Rețelele convoluționale sunt foarte asemănătoare cu cele neuronale ordinare prin faptul că sunt compuse din neuroni în care sunt reținute informații. Fiecare neuron primește ca date de intrare un vector de numere, realizează o operație de înmulțire cu vectorul de ponderi și apoi este adunat un parametru numit bias. Astfel, formula arată în felul următor:

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

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

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

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

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

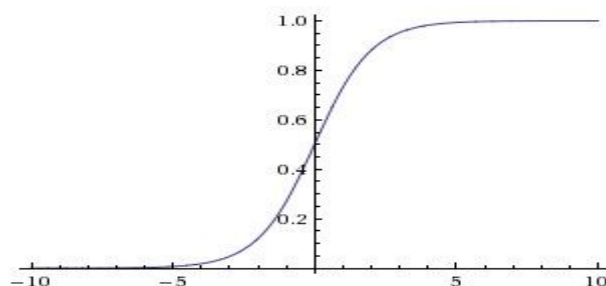


Figura 2.1: Reprezentarea grafică a funcției sigmoid

Funcția sigmoid a fost folosită în trecut datorită faptului că reușește să restrângă variabila în intervalul  $[0, 1]$ . În practică, această funcție este folosită foarte rar deoarece are câteva mari dezavantaje. Primul problemă este aceea că sigmoid saturează în jurul punctelor 0 și 1, gradientul în aceste regiuni fiind 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 puțin. De aceea, este necesar să fim foarte atenți când inițializăm ponderile neuronilor care folosesc sigmoid pentru a evita saturarea.

Un alt dezavantaj este acela că rezultatele funcției 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 funcției. În plus, funcția exponențială este scumpă din punct de vedere computațional.

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

$$\tanh(x) = 2 * \sigma(2 * x) - 1 \quad (2.4)$$

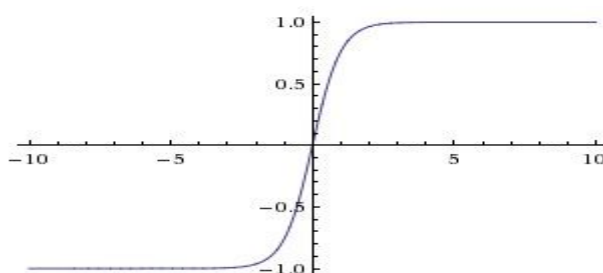


Figura 2.2: Reprezentarea grafică a funcției tangentă

În continuare, vom prezenta funcția de activare **ReLU**. Aceasta a fost propusă pentru a fi folosită în rețelele neuronale de către Alex Krizhevsky în [1]. Aici, ne este indicat să folosim ReLU în detrimentul tangentei, acuratețea rămânând aceeași, deși este de 6 ori mai rapidă. Această funcție are următoarea formulă:

$$f(x) = \max(0, x) \quad (2.5)$$

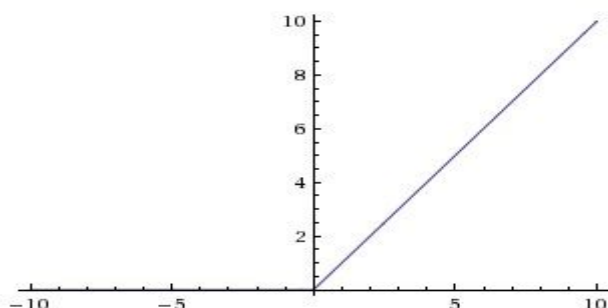


Figura 2.3: Reprezentarea grafică a funcției ReLU

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 avem o probabilitate pentru a ‘adormi’ un neuron. Astfel, la fiecare pas de propagare înainte, sunt decizi care neuroni vor participa în acel pas și doar ei vor fi actualizați la propagarea înapoi, restul fiind setați să aibă valoarea 0 în acea rundă.

Mai mulți neuroni interconectați 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 răspunsul.

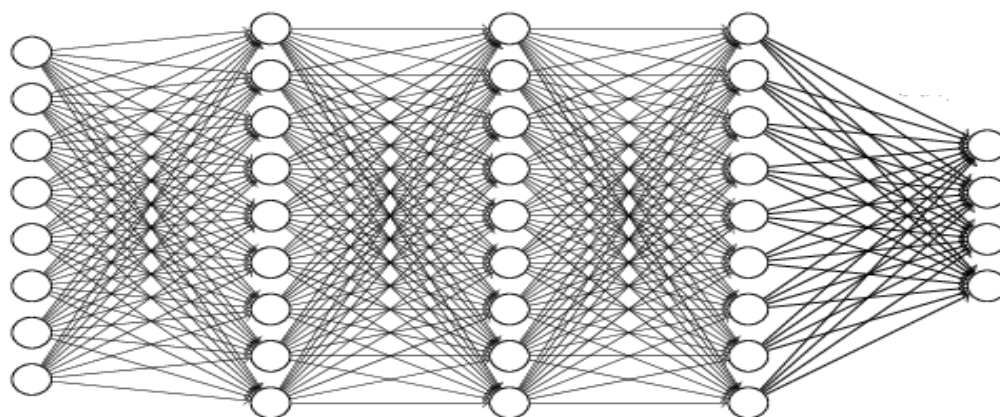


Figura 2.4: Reprezentarea grafică a unei rețele neuronale

Ultimul strat al unei rețele este unul de decizie. Aici va avea loc prezicerea asupra cărei clase aparțin datele de intrare. Cea mai populară funcție este **Softmax**. Ea primește la intrare un vector de dimensiune  $n$  de numere reale și returnează un vector de numere reale între 0 și 1, având suma totală egală cu 1. Astfel, valoarea vectorului de ieșire poate fi gândită ca ‘probabilitatea ca un element să aparțină acelei clase’. Funcția softmax are următoarea formulă:

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

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 în acel pas. Cea mai cunoscută funcție de cost este aceea în care calculăm pătratul diferenței dintre ce am prezis noi și răspunsul corect pentru fiecare clasă, iar apoi le adunăm pe toate pentru a obține eroarea totală. În funcție de tipul problemei pe care încercăm să o rezolvăm, pot apărea 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.

În acest pas avem toate componentele necesare pentru a construi o rețea neuronală ordinară. Toate aceste lucruri vor apărea și în rețelele neuronale 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 pentru a ne referi la o anumită componentă a unei imagini. 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, 0 reprezentând negru și 255 alb.

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. Principalul rol al unei convoluții este să extragă trăsături din imagine. Convoluțiile au proprietatea de a păstra legătura spațială dintre pixeli, fiind folosite patrate 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 apoi este efectuată suma componentelor matricei rezultate.

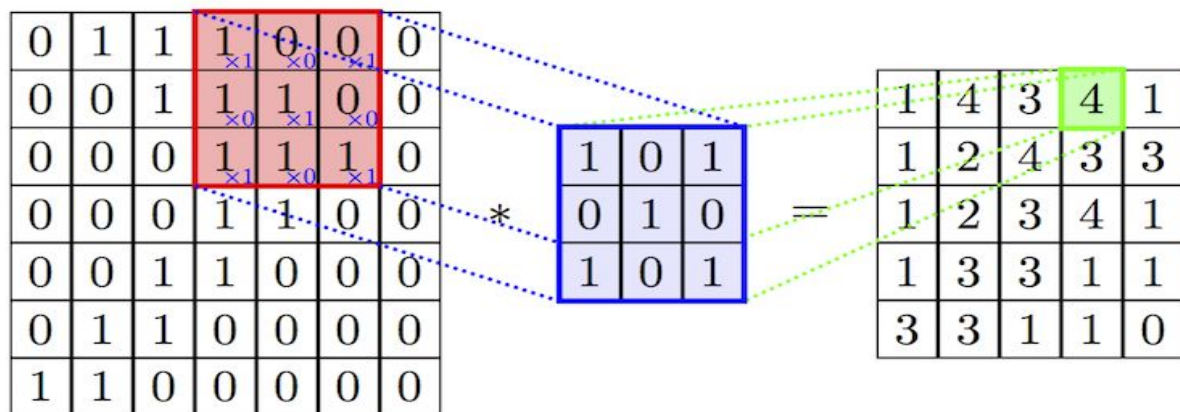


Figura 2.5: Reprezentarea grafică a unui pas de convoluției

Pentru fiecare strat convoluțional trebuie să decidem ce dimensiune va avea filtrul, peste câți pixeli să sara 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 stânga-dreapta, sus-jos și 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 funcția de decizie să fie mai discriminativă. În al doilea rând, campul vizual al unui strat convoluțional 7x7 este același precum cel al 3 convoluții consecutive 3x3. Nu în ultimul rând, un strat de dimensiune 7x7 ar avea nevoie de  $7 \times 7 \times D \times D$  parametri,  $D$  reprezentând numărul de canale, pe când cele 3 straturi 3x3 ar avea nevoie de doar  $3 \times (3 \times 3 \times D \times D)$ . Astfel, stratul 7x7 ar avea nevoie de 81% mai mulți 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. Creșterea pasului de deplasare va genera o matrice de trăsături de dimensiune mai mică.

Uneori ne va fi de folos să creștem 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 rețea convoluțională, trebuie să avem grijă la dimensiunea ferestrei care va rezulta după un strat convoluțional. Astfel, dimensiunea matricei de trăsături rezultate în urma stratului de convoluție poate fi calculată astfel:

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

, unde D reprezintă dimensiunea matricei de trăsături, W dimensiunea înălțimea/lățimea 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 păstrând cea mai importantă informație. Conform [4], cea mai folosită metodă de unificare este cea a maximului. Prin această procedură, matricea de trăsături este împărțită în zone nesuprapuse și este extras maximul din fiecare petec.

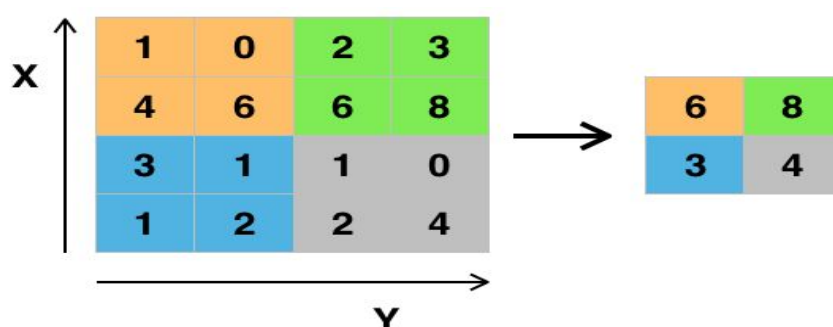


Figura 2.6: Strat de unificare prin extragerea maximului

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

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

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

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

În final, o rețea neuronală convoluțională va fi compusă din mai multe straturi convoluționale, printre care apar straturi de unificare și la final 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.

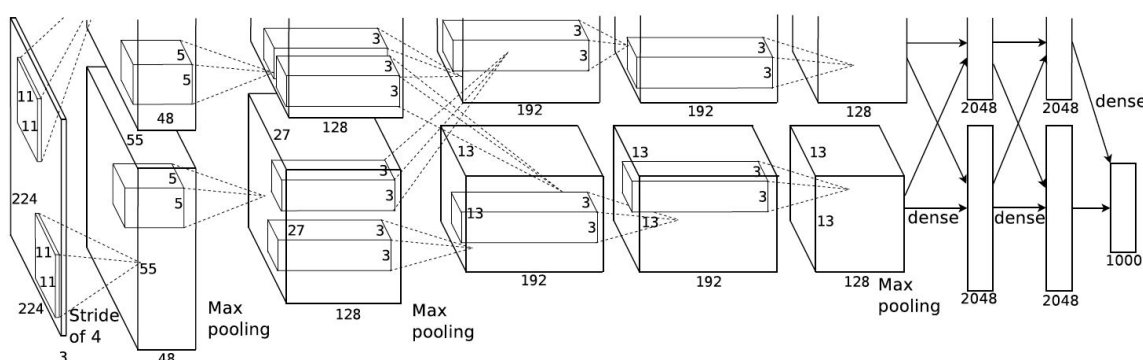


Figura 2.7: Rețeaua AlexNet, propusă de Alex Krizhevsky în [1]

Alex Krizhevsky, Ilya Sutskever și Geoffrey Hinton au șocat lumea în 2012, când au prezentat o rețea neuronală convoluțională care a câștigat concursul ILSVRC, obținând o eroare de doar 15.4% pe primele 5 predicții, față de precedentă 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 rețele care au bătut recordul precedent. În 2015, Microsoft a ajuns la o eroare pe primele 5 predicții de 3.56% prin rețeaua ResNet, prezentată în [5].

### 2.1.2. Histograma gradientelor orientați (HOG)

Histograma gradientelor orientați a fost folosită pentru a extrage trăsăturile unui obiect din imagini. Spre deosebire de o rețea neuronală convoluțională care învață 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, indiferent de input. De aceea, când avem un set de date cu multe imagini, este de preferat să antrenăm o rețea deoarece aceasta va avea 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 gradientilor orientați să ofere o acuratețe mai bună.

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

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

Una dintre cele mai obișnuite utilizări a gradientilor este în detectarea muchiilor unei imagini.

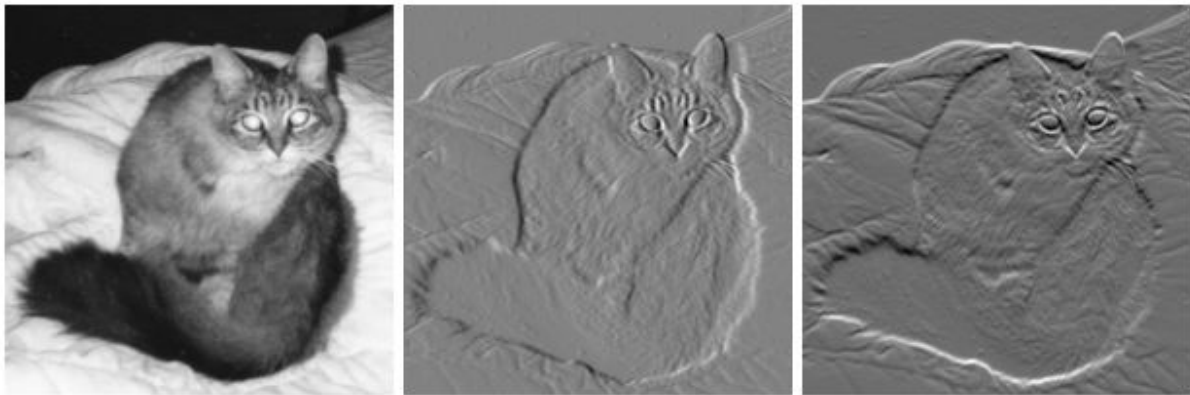


Figura 2.8: Gradientii unei imagini

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

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

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

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

Direcția gradientului este dată de:

$$\theta = \tan^{-1} \left[ \frac{g_x}{g_y} \right] \quad (2.10)$$

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

$$G = \sqrt{g_x^2 + g_y^2} \quad (2.11)$$

Una dintre cele mai simple metode de a găsi gradientii 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 neuronale convoluționale. Pentru găsirea gradientului pe direcția y, vor aplica aceeași metoda, dar vom folosi transpusa filtrului Sobel. Pentru a obține gradientii pe ambele direcții, este suficient să trecem cu primul filtru pe imaginea inițială, iar apoi cu al doilea filtru pe imaginea rezultată după aplicarea primului filtru. Imaginea finală va fi folosită în construirea histogramei gradientilor orientați. În acest pas este calculată direcția și magnitudinea gradientilor.

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

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

Figura 2.9: Componentele filtrului Sobel pe direcția x

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

După aflarea gradientilor și împărțirea imaginii pe celule, urmează să construim histograma în fiecare dintre aceste zone. Matricea direcției gradientilor va conține valori cuprinse între 0 și 179, reprezentând măsura unghiului dintre direcția gradientului și axa OX în grade. Astfel, histograma va împărți aceste unghiuri în porțiuni, iar fiecare pixel își va aduna valoarea magnitudinii în componenta corespunzătoare direcției sale. După ce toți gradientii pixelilor unei celule au fost parcurși, histograma acelei celule va fi completă.

Histograma gradientilor orientați ale unei imagini rezulta prin concatenarea histogramelor tuturor celulelor acelei imagini.

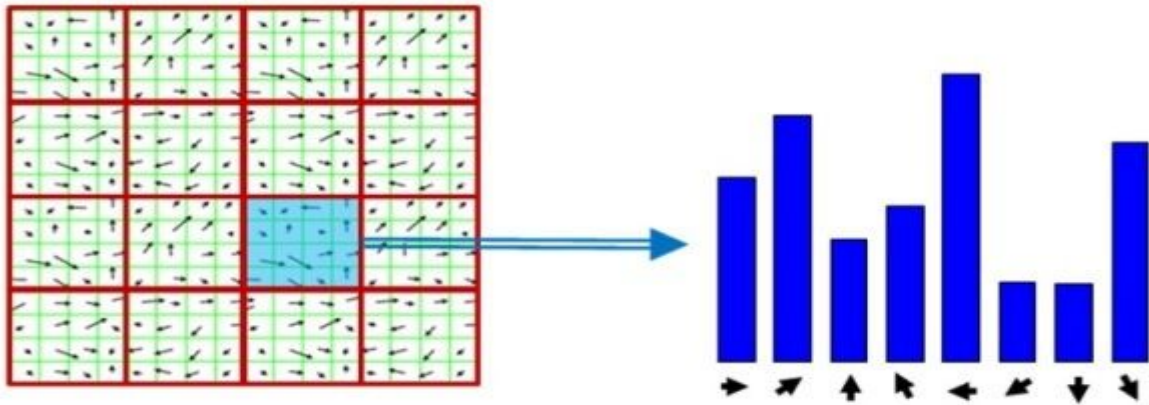


Figura 2.9: Reprezentarea grafică a histogramei unei celule

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