UNIVERSITATEA POLITEHNICA BUCUREŞTI

FACULTATEA DE ELECTRONICĂ, TELECOMUNICAŢII ŞI TEHNOLOGIA INFORMAŢIEI

Sistem de recunoaștere și clasificare a gesturilor mâinii utilizând rețele neurale convoluționale

PROIECT DE DIPLOMĂ

Prezentat ca cerinţă parţială pentru obţinerea titlului de Inginer în domeniul Electronică, Telecomunicaţii şi Tehnologia Informaţiei

Programul de studii: Electronică Aplicată (ETC-ELA)

Conducător ştiinţific:

Prof. Dr. Ing. Corneliu BURILEANU

Absolvent:

Chiriță Ștefan-Adrian

București

2019

Universitatea "Politehnica" din Bucureşti
Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei
Departamentul **EAII**

Anexa 1

TEMA PROIECTULUI DE DIPLOMĂ
a studentului **CHIRIŢĂ I.A. Ştefan-Adrian , 444B**

1. Titlul temei: Sistem de recunoaştere şi clasificare a gesturilor mâinii utilizând reţele neurale convoluţionale

2. Descrierea contribuţiei originale a studentului (în afara părţii de documentare) şi specificaţii de proiectare:
Lucrarea de diplomă presupune dezvoltarea unei aplicaţii de recunoaştere şi clasificare a gesturilor mâinii pe baza imaginilor, folosind algoritmi complecşi de inteligenţă artificială, pentru a obţine o acurateţe cât mai bună, ierarhizându-le în una dintre cele 5-7 clase de poziţii ale mâinii.

Clasificarea se realizează cu ajutorul unei reţele neurale profunde, care va primi la intrare o serie de parametrii extraşi din imagine şi va returna, în funcţie de probabilitatea de decizie, una dintre cele 5-7 clase de gesturi posibile. Sistemul va fi testat pe mai multe baze de date: o bază de date etalon standardizată, pentru a realiza o testare în condiţii optime, o bază de date realizată personal, pentru a testa performanţele sistemului în situaţii reale şi o bază de date preluată în timp real dintr-un flux video, pentru a pune in evidenţă funcţionarea în condiţii nefavorabile.

3. Resurse folosite la dezvoltarea proiectului:
Python, PyCharm, Keras, camera web/Kinect

4. Proiectul se bazează pe cunoştinţe dobîndite în principal la următoarele 3-4 discipline:
Cunoştinţe dobândite în cadrul laboratorului de cercetare SpeeD, cursul online "Convolutional Neural

5. Proprietatea intelectuală asupra proiectului aparţine: U.P.B.

6. Data înregistrării temei: 2018-11-29 18:14:50

Conducător(i) lucrare,
Prof. dr. ing. Corneliu BURILEANU
semnătura: .....

Neacsu Ana
semnătura: .....

Student,
semnătura:.....

Director departament,
Prof. dr. ing Sever PAŞCA
semnătura:.....

Decan,
Prof. dr. ing. Cristian NEGRESCU
semnătura:.....

Cod Validare: **3a07a0adde**

Declarație de onestitate academică

Prin prezenta declar că lucrarea cu titlul *" Sistem de recunoaștere și clasificare a gesturilor mâinii utilizând rețele neurale convoluționale"*, prezentată în cadrul Facultății de Electronică, Telecomunicații și Tehnologia Informației a Universității "Politehnica" din București ca cerință parțială pentru obținerea titlului de *Inginer* în domeniul *Electronică și Telecomunicații,* programul de studii *Electronică Aplicată (ETC – ELA)* este scrisă de mine și nu a mai fost prezentată niciodată la o facultate sau instituție de învățământ superior din țară sau străinătate.

Declar că toate sursele utilizate, inclusiv cele de pe Internet, sunt indicate în lucrare, ca referințe bibliografice. Fragmentele de text din alte surse, reproduse exact, chiar și în traducere proprie din altă limbă, sunt scrise între ghilimele și fac referință la sursă. Reformularea în cuvinte proprii a textelor scrise de către alți autori face referință la sursă. Înțeleg că plagiatul constituie infracțiune și se sancționează conform legilor în vigoare.

Declar că toate rezultatele simulărilor, experimentelor și măsurătorilor pe care le prezint ca fiind făcute de mine, precum și metodele prin care au fost obținute, sunt reale și provin din respectivele simulări, experimente și măsurători. Înțeleg că falsificarea datelor și rezultatelor constituie fraudă și se sancționează conform regulamentelor în vigoare.

București, *Iunie, 2019*

Absolvent *Ștefan-Adrian CHIRIȚĂ*

(semnătura în original)

CUPRINS

Listă de figuri

Listă de tabele

Listă de acronime

RGB – Red Green Blue

CNN – Convolutional Neural Network

DNN – Deep Neural Netwok

ANN – Artficial Neural Network

HSV – Hue Saturation Value

HSL – Hue Saturation Lightness

W - Weights

B – Bias

GPU – Graphics Processing Unit

CPU – Cenral Processing Unit

ReLU - Rectified Linear Unit

SVM- Suport Vector Machine

SGD – Stochastic Gradient Descent

RMS -Root Mean Square

Introducere

Motivaţia lucrării

Comunicarea nonverbală este reprezentă de o varietate de mesaje care nu pot fi exprimate prin cuvinte și pot fi decodificate pentru extragerea înțelesului respectiv. Gesturile acestea pot repeta, contrazice, înlocui, completa sau accentua mesajul transmis prin cuvinte. Importanța comunicării nonverbale a fost demonstrată în anul 1967 de către Albert Mehrabian care, în urma unui studiu, acesta a ajuns la concluzia că numai 7% din mesaj este transmis prin comunicare verbală, în timp ce 38% este transmis pe cale verbală și 55% prin limbajul corpului, deci prin comunicarea nonverbală.

În zilele noastre, una dintre problemele de actualitate este reprezentată de incapacitatea comunicării verbale pentru persoane cu deficiențe de vorbire dar și în circumstanțe diferite, cum ar fi necesitatea interacțiunii cu roboți sau platforme robotice. Gesturile reprezintă unul din cele mai folosite tipuri de limbaj pe care le poate înţelege oricine. Utilitatea acestui limbaj este recunoscută universal, gesturile fiind autonome şi având posibilitatea de a înlocui în totalitate un cuvânt.

Sistemul realizat are ca scop facilitarea comunicării dintre un om cu deficiențe de vorbire sau auz cu altă persoană sau mediul înconjurător. Pe de altă parte, având în vedere contribuțiile viitoare, va fi folosit pentru executarea comenzilor de către un robot doar prin efectuarea unor gesturi ale operatorului uman.

Obiectivele lucrării

Lucrarea de faţă are ca scop dezvoltarea unei aplicații de recunoaștere și clasificare a gesturilor mâinii pe baza imaginilor, folosind algoritmi complecși de inteligență artificială, pentru a obține o acuratețe și precizie cât mai bună, ierarhizându-le în una dintre cele 10 clase de poziții ale mâinii. Pentru a permite utilizarea metodei pe orice calculator cu dotări medii fără a necesita echipamente suplimentare costisitoare, recunoașterea gesturilor trebuie să se realizeze exclusiv pe baza informației vizuale, achiziționate cu ajutorul unei camere web.

În mod concret, pentru realizarea acestor sarcini vor fi îndeplinite următoarele obiective:

1. Crearea unei baze de date (a unui set de antrenare) pentru fiecare gest al mâinii pentru antrenarea rețelei neurale (învățare de tip supervizat).
2. Preprocesarea imaginilor captate de camera web: redimensionare, filtrare și binarizare pentru a reduce complexitatea datelor de intrare în rețeaua neurală.

3. Implementarea unei metode pentru recunoașterea mâinii atât dintr-o imagine cât și dntr-un flux video.
4. Crearea unei Rețele Neurale Convoluționale (CNN) pentru a realiza clasificarea corectă a imaginilor în funcție de clasa de apartenență.

Structura lucrării

Capitolul 1 reprezintă o introducere teoretică ce are ca scop prezentarea fundamentelor ce stau la baza prelucrării de imagine. Sunt descrise operațiile care se utlizează pentru a ajunge la imaginea dorită dar și resursele utilizate.

Capitolul 2 este dedicat rețelelor neurale artificiale și convoluționale. Se prezintă funcționarea tipurilor de neuroni și o descriere a funcțiilor de activare alături de structura straturilor unei rețele neuronale convoluționale. Algoritmii de optimizare sunt prezentați pentru fiecare tip de rețea utlizat.

Capitolul 3 descrie etapele generale de realizare ale proiectului și de recunoaștere a gesturilor mâinii, urmărnd achiziția de date, preprocesarea imagini și dezvoltarea rețelei neurale convoluționale.

Capitolul 4 cuprinde parte de experimente și rezultate ce ilustrează performanțele rețelei din punct de vedere al testării dinamice cu ajutorul camerei web, dar și statice cu ajutorul bazei de date de testare.

Capitolul 1. Noțiuni teoretice privind prelucrarea imaginilor

Procesarea imaginilor este un domeniu relativ recent, care evoluează rapid. Aplicațiile sale se întâlnesc pretutindeni: armată, industrie, medicină sau acolo unde informația necesară este reprezentată sub formă de imagini. Principala aplicație o reprezintă înbunătățirea informației conținute de imagini având în vedere interpretarea de către un subiect uman sau pentru vederea artificială a unui robot sau platforme robotice.

Obținerea unei descrieri pentru imaginea supusă analizei presupune realizarea unei operații de segmentare a imaginii. Aceasta constă in descompunerea imaginii intr-un numar de regiuni disjuncte, fiecare având un anumit grad de omogenitate în raport cu o anumită proprietate: nivel de gri, culoare, textură, modulul gradientului funcției etc. Realizarea unei segmentări corecte si complete în care regiunile separate pot fi puse direct în corespondență cu un anumit obiect din imagine este dificil de realizat [2]. Pentru a descrie corect obiectele din imagine procesul de segmentare trebuie sa facă apel la o serie de informații suplimentare legate de tipul imaginii care trebuie segmentată și să coopereze cu alte module de nivel înalt utilizate ulterior segmentării în analiza imaginii. În cele mai multe cazuri însa, procesul segmentării imaginii poate fi dus la bun

sfârsit utilizând tehnici de procesare de nivel scăzut. Astfel de situații sunt cele în care imaginea constă dintr-un numar de obiecte situate pe un fond uniform fața de care exista o diferență mare de contrast. Pentru imagini de acest tip operația de segmentare poate fi realizata în mod global, fără a corela rezultatele obținute cu o serie de cunoștințe apriorice referitoare la obiectele din imagine [6] .

1.1 Prelucrarea imaginilor

Scopul acestor prelucrări îl constituie accentuarea (punerea în evidență) a unor caracteristici conținute în imagine pentru a putea fi observate mai ușor. Metodele utilizate în algoritmii de îmbunătățire a imaginilor amplifică anumite caracteristici fără a mări cantitatea de informație, aplicându-se operații punctuale sau spațiale.

1.1.1 Detectarea regiunii pielii:

Într-un sistem de viziune, condițiile de iluminare reprezintă unul dintre cei mai importanți factori ce afectează performanța sistemului. Utilizarea pragurilor de culoare pentru clasificarea culorii pielii și a culorii fundalului este comună în abordări convenționale, dar pragurile de culoare nu sunt suficiente pentru a descrie proprietățile statistice ale culorii pielii sub diverse condiții de lumină.

O metodă simplă este reprezentată de verificarea fiecărui pixel al pielii pentru a vedea daca intră într-un interval de culori sau valori definite în anumite coordonate ale unui spațiu de culoare. Există mai multe spații de culori precum RGB, HSV, YcbCr etc. care sunt utilizate pentru segmentarea culorii pielii [2]

Următorii factori trebuie luați în considerare pentru a determina intervalul de prag:

- Efectul iluminării în funcție de împrejurimi.
- Caracteristici individuale cum ar fi vârsta, sexul și părțile corpului.
- Variația tonului pielii cu privire la diferitele rase.
- Alți factori, cum ar fi culorile de fundal, umbrele și încețoșarea cauzată de mișcare.

1.1.2 Conversia RGB – HSV

HSV este una dintre cele mai utilizate reprezentări de coordonate cilindrice ale punctelor dintr-un model RGB. Acest tip de reprezentare presupune o rearanjare a geometriei RGB pentru a fi mai intuitivă și mai perceptivă decât reprezentarea carteziană. HSV, împreună cu HSL, au fost

dezvoltate în principal pentru aplicații grafice computerizate, dar sunt folosite astăzi în colectori de culori, editare de imagini, analiză de imagine și domeniul computer vision [9, 16].

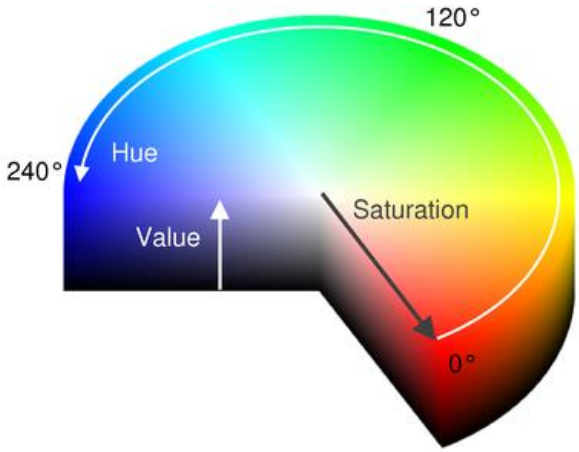

Figura 1.1. Reprezentare HSV Sursa [9]

În HSV, unghiul din jurul axei verticale reprezintă componenta "nuanță", distanța de la axă corespunde "saturației", iar distanța de-a lungul axei corespunde "valorii" sau "luminozității".

	Reprezentare RGB			**Reprezentare HSV**		
Cantitate culoare	Rosu	Verde	Albastru	Nuanță	Saturație	Valoare
Rosu Δ	130-255	0-162	0-155	0-9 / 151-180	1.00	1.00
Verde Δ	0-50	150-255	0-50	46-100	0.875	0.795
Albastru Δ	0-50	0-100	150-255	101-150	0.887	0.918
Galben Δ	0-207	200-255	0-150	16-45	0.467	0.998

Tabelul 1.1. Tabel Reprezentare RGB vs HSV

Spre deosebire de RGB, HSV separă „luma”, sau intensitatea imaginii, de „croma” sau informațiile de culoare. Acest lucru este foarte util în multe aplicații. De exemplu, dacă se dorește efectuarea unei egalizări de histograma a unei imagini color, probabil că dorim să o facem doar pe componenta de intensitate și lăsăm componentele de culoare singure. În caz contrar, se vor obține culori foarte diferite de ceea ce ne dorim.

În domeniul Computer Vision, se dorește schimbarea componentele de culoare de intensitate din diferite motive, cum ar fi robustețea la schimbările de lumină sau eliminarea umbrelor. Se poate folosi funcția cv2.cvtColor pentru conversia imaginii din RGB în HSV iar pentru binarizare funcția inRange cu parametrii specifici.

Figura 1.2. Binarizarea imaginilor folosind pragul

(a)Imagine HSV (b) Imagine binarizată

1.1.3 Metoda Otsu (metoda 2)

În cadrul proiectului a fost efectuată încă o metoda pentru a ajunge la o imagine finală binară și anume binarizarea de tip Otsu. Avem două grupuri de pixeli, care se întind pe domenii de intensităţi diferite (ex: obiecte şi fundal). Problema selecţiei unui prag este complicată de faptul că aceste domenii de intensitate uneori se suprapun parţial. Se doreşte minimizarea erorilor de clasificare a unui pixel obiect ca fundal, şi viceversa. Pentru realizarea acestui deziderat, încercăm să minimizăm aria de sub histograma unei regiuni, care pe baza pragului calculat va fi atribuită celeilalte regiuni. Vom considera aceste două regiuni ca două grupuri. Pragul va fi ales astfel încât cele două grupuri să fie cât mai "strânse", minimizând astfel suprapunerea. O măsură a omogenităţii grupului este varianţa. Un grup cu omogenitate mare are o varianţă mică, un grup cu omogenitate mică are varianţă mare[13].

Varianţa ponderată intra-clasă este:

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)$$

Probabilităţile asociate claselor sunt estimate ca:

$$q_1(t) = \sum_{i=1}^{t} P(i) \qquad q_1(t) = \sum_{i=t+1}^{t} P(i)$$

unde $$P(i) = \frac{H(i)}{N^2} \quad \text{și} \quad q_2 = 1 - q_1$$

Mediile claselor sunt definite ca:

$$\mu_1(t) = \sum_{i=1}^{t} \frac{iP(i)}{q_1(t)} \qquad \mu_1(t) = \sum_{i=1}^{t} \frac{iP(i)}{q_2(t)}$$

Varianţele individuale ale claselor sunt definite ca:

$$\sigma_1^2(t) = \sum_{i=1}^{t} [i - \mu_1(t)]^2 \frac{P(i)}{q_1(t)} \qquad \sigma_1^2(t) = \sum_{i=1}^{t} [i - \mu_1(t)]^2 \frac{P(i)}{q_1(t)}$$

În acest moment avem toate ecuaţiile necesare pentru a măsura varianţa ponderată intra-clasă. Am putea să verificăm fiecare posibilă valoare a pragului t, şi să o alegem pe cea care minimizează această varianţă[13].

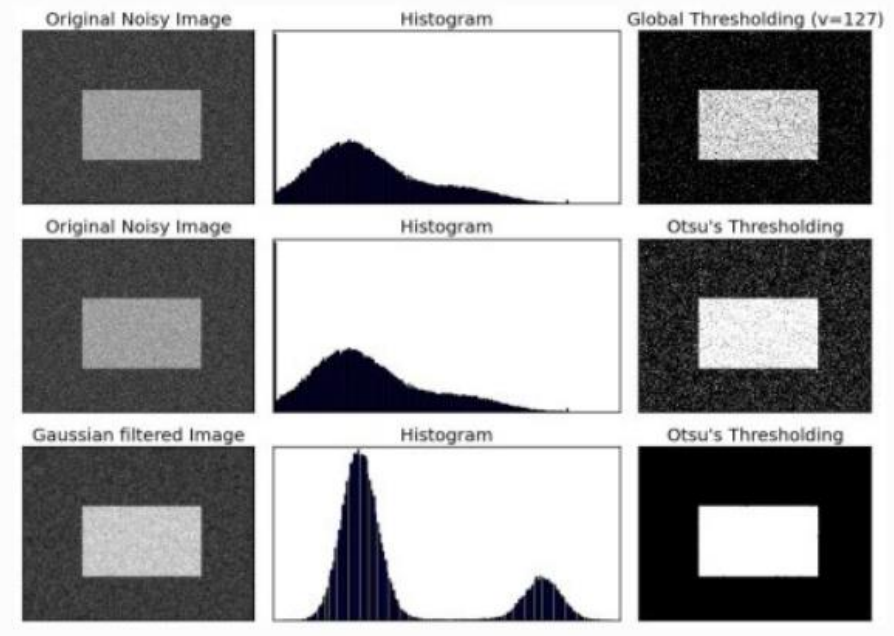

Figura 1.3. Histograme pentru imagine cu zgomot (primele 2) și pentru imagine cu filtru Gaussian și rezultatele obțiune cu binarizare normală si Otsu(Sursa [9])

În cadrul proiectului a fost folosită prima metoda datorită faptului că metoda Otsu întâmpina probleme majore în cazul asemănării între obiectul detectat (mâna) și fundal.

1.1.4 Eliminare zgomot prin dilatare si erodare

După cum am menționat in prima secțiune, unele părți din fundal ar fi segmentate și acestea inhiba procesul de recunoaștere. Deci, pentru a obține o recunoaștere perfectă este necesară eliminarea zgomotului nedorit. Pentru a obține o mai bună estimarea a mâinii, trebuie șterși pixelii

zgomotoși din imagine. Folosim operații morfologice care efectuează erodarea și dilatarea imaginii pentru a elimina zgomotul [6].

Bazele erodării:

- Erodează limitele obiectului din prim plan
- Folosit pentru a diminua caracteristicile unei imagini.

Funcționarea eroziunii:

- Un kernel (o matrice de dimensiuni impare (3,5,7)) este un operand în procesul de convoluție cu imaginea.
- Un pixel din imaginea originală (fie 1 sau 0) va fi considerat 1 numai dacă toți pixelii de sub kernel sunt 1, altfel este erodat (făcut zero).
- Astfel, toți pixelii din apropierea limitei vor fi aruncați în funcție de dimensiunea kernelului.
- Deci, grosimea sau mărimea obiectului din prim plan scade sau pur și simplu regiunea albă scade în imagine.

Bazele dilatării:

- Crește zona obiectului
- Folosit pentru a accentua caracteristicile

Funcționarea de dilatare:

- Un kernel (o matrice de dimensiuni impare (3,5,7) este un operand în procesul de convoluție cu imaginea.
- Un pixel în imaginea originală este '1' dacă cel puțin un pixel din kernel este '1'.
- Se mărește regiunea albă în imagine sau mărimea obiectului din prim plan

Matematic, erodarea poate fi definită ca: $A \ominus B = \bigcap_{b \in B} A_{-b}$

Matematic, dilatarea poate fi definită ca: $A \oplus B = \bigcup_{b \in B} A_b$

Utilizările eroziunii si dilatării:

Eroziune:

- Este utilă pentru eliminarea zgomotelor albe mici.
- Se utilizează pentru a detașa două obiecte conectate etc.

Dilatarea:

- În cazurile de eliminare a zgomotului, eroziunea este urmată de dilatare. Deoarece, eroziunea elimină zgomotele albe, dar și îngustează obiectul.
- Este, de asemenea, utilă în îmbinarea părților rupte ale unui obiect. [13]

Figura 1.4. a) Imagine binară b) Imagine dilatată c) Imagine erodată Sursa [19]

1.2 Biblioteca OpenCV

OpenCV (Open Source Computer Vision Library) este o bibliotecă software “open source” destinată recunoașterii computerizate a imaginilor (computer vision) și învățarea mașinilor (machine learning). Această librărie a fost construită pentru a oferi o infrastructură comună pentru aplicațiile de imagistică computerizată și pentru a accelera utilizarea percepției mașinilor în produsele comerciale. Fiind un produs cu licență BSD, OpenCV facilitează utilizarea și modificarea codului de către companii. [9]

Biblioteca are mai mult de 2500 de algoritmi optimizați, care includ un set vast de algoritmi de recunoaștere clasică și de ultimă generație a imaginilor oferite de calculator și de învățare a mașinilor. Acești algoritmi pot fi utilizați pentru a detecta și a recunoaște fețele, a identifica obiecte, a clasifica acțiunile umane în videoclipuri, a urmări mișcările camerei, a urmări obiectele în mișcare, a extrage modele 3D de obiecte, concatenarea unor puncte pentru producerea unei imagini de înaltă rezoluție, găsirea unor imagini similare dintr-o bază de date a imaginilor, îndepărtarea ochilor roșii din imaginile realizate cu ajutorul blițului, urmărirea mișcărilor ochilor, recunoașterea peisajelor și stabilirea marcatorilor pentru a suprapune cu realitatea augmentată etc [9].

Pentru proiectul meu, am folosit această bibliotecă pentru a detecta culoarea mâinii din imagine, conversia culorii în HSV,efectuarea de operații morfologice, conversia culorii în nivele de gri, binarizarea, precum și pentru a trasa chenarul de poziționare al mâinii.

Capitolul 2. Noțiuni teoretice privind rețele neurale artificiale și convoluționale

2.1 Analogie neuroni biologici și neuroni artificiali

Creierul uman este structura cea mai complexă la ora actuală și înțelegerea funcționării sale reprezintă una dintre cele mai mari, dificile și interesante provocări cu care știința se confruntă.[3]

Capacitatea de a învăța este considerată un semn distinctiv al vieții inteligente. Domeniul de „Machine learning" oferă capacitatea de învățare și extrapolare din seturi de date pentru a realiza sarcini complexe cum ar fi clasificarea fenomenelor necunoscute anterior.

Există atât similitudini surprinzătoare, cât și diferențe importante în ceea ce privește modul în care mașinile învață în raport cu oamenii. Folosind rețele neuronale biologice, învățarea reiese din interconexiunile dintre neuronii din creier. Interconexiunile acestor neuroni se schimbă deoarece creierul este expus la noi stimuli. Aceste modificări includ conexiuni noi, consolidarea

conexiunilor existente și eliminarea conexiunilor neutilizate. De exemplu, cu cât se repetă mai mult o anumită sarcină, cu atât este mai puternică legătura neurologică, până când această sarcină este considerată învățată.

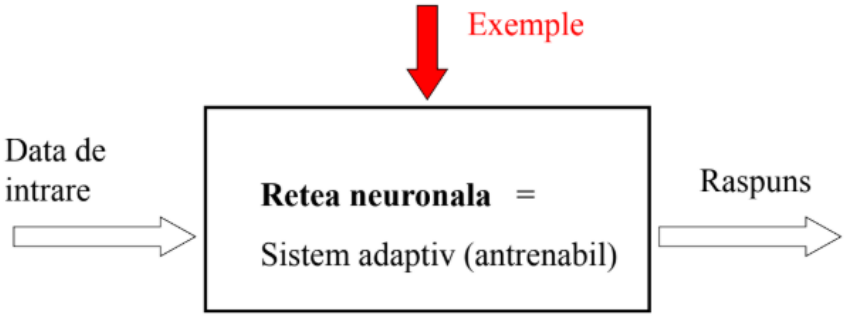

Figura 2.1. Funcționarea unei rețele neuronale artificiale (Sursa[4])

Neuronii pot procesa noi stimuli folosind reprezentări prestabilite din memorie și percepții bazate pe activarea unui mic set de neuroni. Pentru fiecare stimul, un subgrup diferit dintr-un grup mare de neuroni disponibili este activat în timpul învățării.

Acest design uimitor din biologie a inspirat cercetătorii de date în proiectarea modelelor de inteligență artificială.

Rețelele neuronale artificiale (RNA) își propun să imite acest comportament într-un sens abstract, dar pe o scară mult mai mică și mai simplă.

Aproximativ 86 miliarde de neuroni se găsesc în sistemul nervos uman și sunt conectați cu aproximativ 10^{14} - 10^{15} sinapse. Diagrama de mai jos ilustrează o reprezentare a unui neuron biologic (stânga) și un model matematic comun (dreapta). Fiecare neuron primește semnale de intrare de la dendritele sale și produce semnale de ieșire de-a lungul axonului său (unic), care în cele din urmă se conectează prin sinapse la dendritele altor neuroni. În modelul computațional al unui neuron, semnalele care se deplasează de-a lungul axonilor (de exemplu, x_0) interacționează în mod multiplicativ (de exemplu, w_0x_0) cu dendritele celuilalt neuron pe baza rezistenței sinaptice la acea sinapsă (de exemplu, w_0).[1] Ideea este că, forțele sinaptice (ponderile w) se pot învăța și controlează puterea de atracție între neuroni (și direcția: excitatoare (poderi pozitive) sau inhibitoare (ponderi negative)). În modelul de bază, dendritele transportă semnalul către corpul celular, unde fiecare se însumează. Dacă suma finală depășește un anumit prag, neuronul se activează, trimițând un impuls de-a lungul axonului său.

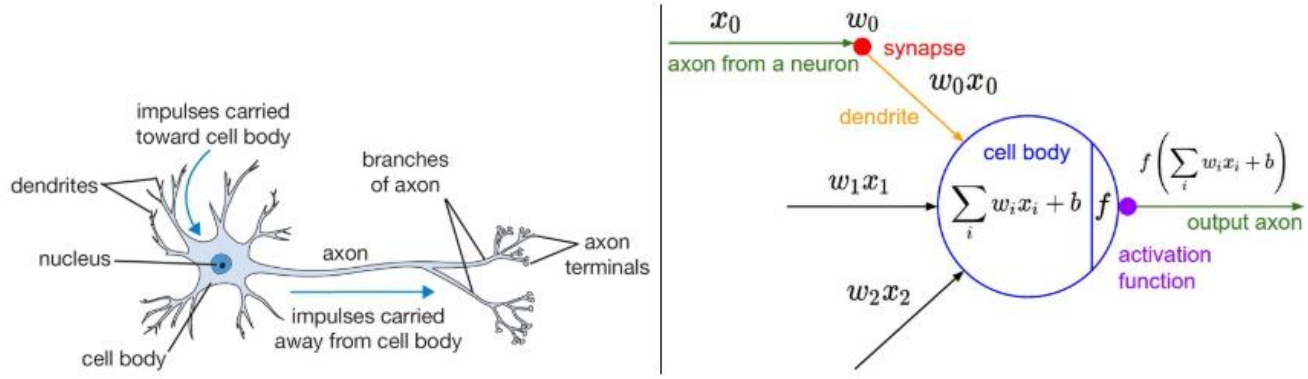

Figura 2.2. Neuron biologic(stânga) și model matematic(dreapta) (Sursa[1])

2.2 Modelul neuronal McCulloch-Pitts

Primul model de neuron a fost creat în anul 1943 de către neurofizicianul american Warren Sturgis McCulloch și matematicianul Walter Pitts, acesta reprezentând un instrument cu mai multe intrări, însoțite de ponderi specifice, instrument ce calculează suma intrărilor, ținându-se cont de ponderea fiecăreia dintre ele.

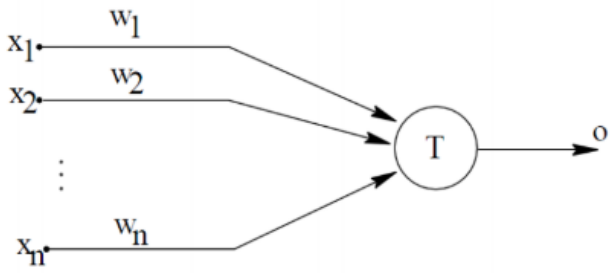

Figura 2.3. Modelul neuronului McCulloch-Pitts(Sursa[4])

Dacă suma totală calculată este superioară unui anumit prag stabilit, neuronul se activează și emite un semnal de ieșire, în caz contrar acesta rămânând neactivat. Acest model primar este cunoscut ca “poartă liniară cu prag” (Linear Threshold Gate). [5]

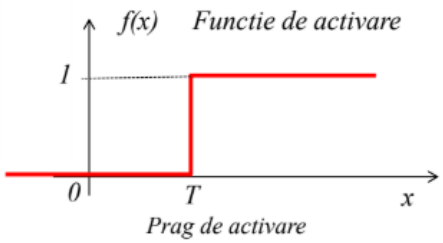

Figura 2.4. Funcția de activare a neuronului McCulloch-Pitts cu pragul T (Sursa[4])

Intrările x_i (i=1, 2, ….n) sunt 0 sau 1, în funcţie de absenţa sau prezenţa impulsului la momentul k (figura anterioara). Semnalul de ieşire al neuronului este „o”. Regula după care neuronul generează un semnal este:

$$o^k = \begin{cases} 1 \text{ daca } \sum_{i=1}^{n} w_i * x_i^k \geq T - \text{Neuron activat} \\ 0 \text{ daca } \sum_{i=1}^{n} w_i * x_i^k < T - \text{Neuron neactivat} \end{cases}$$

Iar pentru o sinapsă :

- excitatoare avem $w_i = 1$
- inhibitoare avem $w_i = -1$

Modelul McCulloch-Pitts al unui neuron este simplu, dar are un potențial semnificativ de calcul. De asemenea, are o definiție matematică precisă. Totuși, acest model este atât de simplist încât generează doar o ieșire binară, iar valorile ponderilor și de prag sunt fixate. Algoritmul de calcul neuronal are caracteristici diverse pentru diverse aplicații cum ar fi implementarea funcțiilor boolene de tip AND, OR, NOR, etc. Astfel, trebuie să obținem modelul neural cu caracteristici computaționale mai flexibile.

2.3 Perceptronul simplu

Mai târziu, în anul 1957, Frank Rosenblatt a dezvoltat algoritmul Perceptron, algoritm ce intenționa să fie o mașinărie, mai degrabă decât un program, utilizată pentru recunoașterea de imagini. Acest algoritm, spre deosebire de cel dezvoltat de McCulloch-Pitts, este unul supervizat, care ajustează ponderile astfel încât eroarea de clasificare să fie minimă.
Perceptronul este cea mai simplă formă a rețelelor neuronale utilizat pentru clasificarea pattern-urilor liniar separabile. Acesta conține un singur neuron cu sinapse ce pot fi ajustate prin modificarea ponderilor și a bias-ului. (Haykin November 28, 2008)

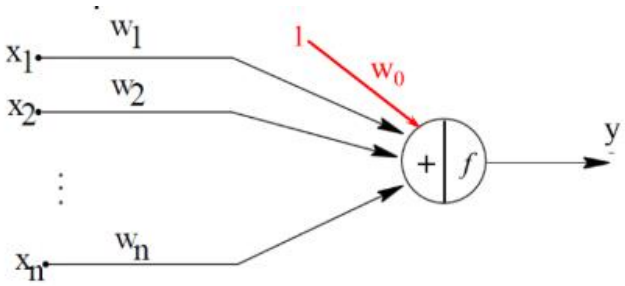

Figura 2.5. Arhitectura perceptronului simplu(Sursa[4])

Prin acest algoritm, Rosenblatt a demonstrat că dacă două clase sunt liniar separabile, atunci perceptronul converge către o suprafață de decizie de forma unui hiperplan între cele două clase. Deoarece perceptronul are un singur neuron, acesta este limitat la clasificarea vectorilor în doar două clase. Spre deosebire de algoritmul dezvoltat de McCulloch și Pitts, neuronul nu mai este reprezentat de un prag, ci acesta constituie o funcție de activare, de forma:

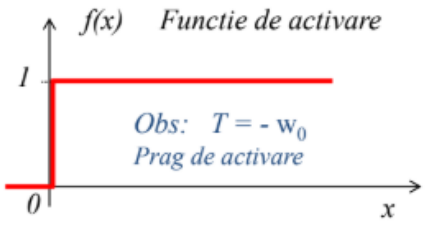

Figura 2.6. Funcția de activare a perceptronului simplu (Sursa[4])

$$\text{unde } f(x) = \begin{cases} 1 \text{ daca } x \geq 0 \\ 0 \text{ daca } x < 0 \end{cases}$$

În Fig. 2.5 este reprezentată arhitectura perceptronului simplu, unde X=(1,x1,x2,…,xn) , xi $\in \mathbb{R}$, reprezintă intrările în neuron, iar W=(w0,w1,…,wn) , $wi \in \mathbb{R}$, reprezintă ponderile neuronului.

Ieșirea neuronului reprezentată în Fig. cu litera y, este dată de ecuația:

$$y = f(w_0 + \sum \mathrm{w_i} * x_i)$$

2.3.1 Algoritmul perceptronului simplu

1. Inițializarea aleatoare a ponderilor la momentul t=0
2. Evaluarea ieșirii reale

$$y(j) = f(W^T X_j)$$

unde j=1:N, N – numărul de vectori de intrare

3. Compararea ieșirii reale yj cu ieșirea dorită dj

$$e_j(t) = d_j - y_j(t)$$

4. Ajustarea ponderilor cu o valoare care micșorează eroarea:

$$\Delta w_i(t) = \eta e_j(t) x_{ij}$$

$$w_i(t+1) = w_i(t) + \Delta w_i(t)$$

Unde η reprezintărata de învățare.

5. Revenire la pasul 2 până când toate corespondentele $\{(Xj,dj)\}$ sunt corecte.

2.3.2 Teorema de convergență

Se poate demonstra că:
Dacă se aplică pentru două seturi de vectori liniari separabili, algoritmul perceptron converge către o soluție stabilă într-un număr finit de pași.Suprafața de separație între clase este un hiperplan de ecuație:

$$w_0 + \sum \mathrm{w_i} x_i = 0$$

Două mulțimi de vectori se numesc liniar separabile dacă există cel puțin o suprafață liniară (de gradul I) care separă spațiul în două semispații în care clasele sunt disjuncte (nu se intersectează).

Una dintre problemele perceptonului este reprezentată de imposibilitatea găsirii unei soluții pentru probleme neseparabile liniar (XOR, par-impar). Mai mult, NU există o soluție de STOP în astfel de situații [7].

2.4 Adaline (Adaptive Linear Neuron)

Structura în etapa de funcționare a neuronului ADALINE este identică cu cea a perceptronului.

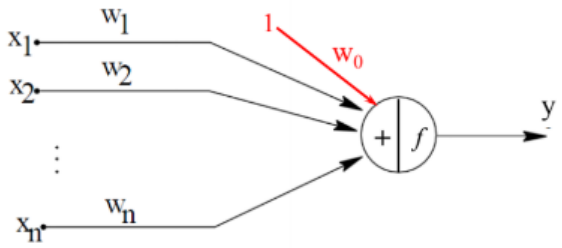

Figura 2.7. Arhitectura Adaline-ului (Sursa[4])

Diferența față de perceptron este modificarea funcției de activare în etapa de învățare:

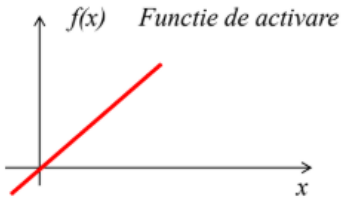

Figura 2.8. Funcția de activare Adaline-ului (Sursa[4])

f(x)= purelin(x)

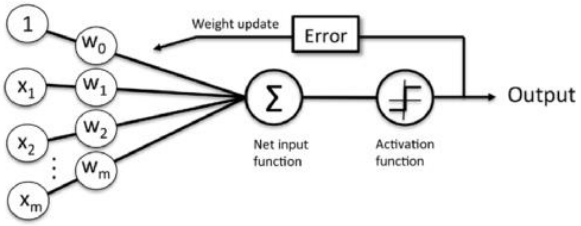

Figura 2.9. Perceptron simplu (Sursa[24])

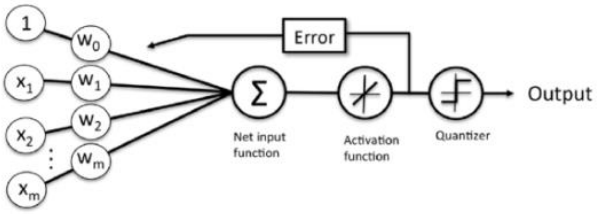

Figura 2.10. Adaline ((Sursa[24])

Noutatea Adaline-ului a constat în dezvoltarea unei noi filozofii în antrenarea rețelei neuronale. S-a introdus o funcție cost care măsoară performanța funcționării neuronului. Funcția cost pentru Adaline este eroarea pătratică între ieșirea dorită și cea reală.

2.4.1 Algoritm Adaline

1. Inițializarea aleatoare a ponderilor la momentul t=0
2. Evaluarea ieșirii reale

$$y(j) = f(W^T X_j)$$

unde j=1:N, N – numărul de vectori de intrare

3. Calcularea erorii:

$$E(W) = \frac{1}{N}\sum_{j=1}^{N} E_j(W)$$

$$E_j(W) = \frac{1}{2}(d_j - y_j)^2$$

4. Ajustarea ponderilor cu o valoare care micșorează eroarea:

$$\Delta w_i(t) = \eta \frac{\partial E}{\partial w_i}$$

$$w_i(t+1) = w_i(t) + \Delta w_i(t)$$

Unde η reprezintărata de învățare.

5. Revenire la pasul 2 până când eroarea medie pătratică pe setul de date scade sub o anumita valoare dorită.

2.5 Algoritmul Gradient Negativ

Algoritmul Gradientului Negativ este cel mai frecvent algoritm de optimizare în învățarea mașinilor și învățarea profundă. Este un algoritm de optimizare de ordinul întâi, fapt ce denota că ia în considerare doar prima derivată atunci când efectuează actualizările parametrilor. La fiecare iterație actualizăm parametrii în direcția opusă gradientului funcției obiectiv J (w) în raport cu parametrii în care gradientul indică direcția celei mai abrupte ascensiuni. Mărimea pasului pe care îl luăm la fiecare iterație pentru a ajunge la minimul local este determinat de rata de învățare η. Prin urmare, urmărim direcția pantei până ajungem la un minim local. [8]

Din motive de simplitate, să presupunem că modelul de regresie logistică are doar doi parametri: ponderea w și bias-ul b.

1. Inițializăm ponderea w și bias-ul b cu orice valoarea aleatoare.

2. Alegem o valoare pentru rata de învățare η. Rata de învățare determină cât de mare ar fi pasul pe fiecare iterație.

- Dacă η este foarte mică, ar fi nevoie de mult timp pentru a converge și a deveni computațional scump.
- Dacă η este mare, este posibil să nu reușească să conveargă și să depășească minimul.

Prin urmare, trebuie să expunem grafic funcția cost în raport cu diferite valori ale η și să alegem valoarea η care este chiar înaintea primei valori care nu converge astfel încât să avem un algoritm de învățare foarte rapid care converge.[10]

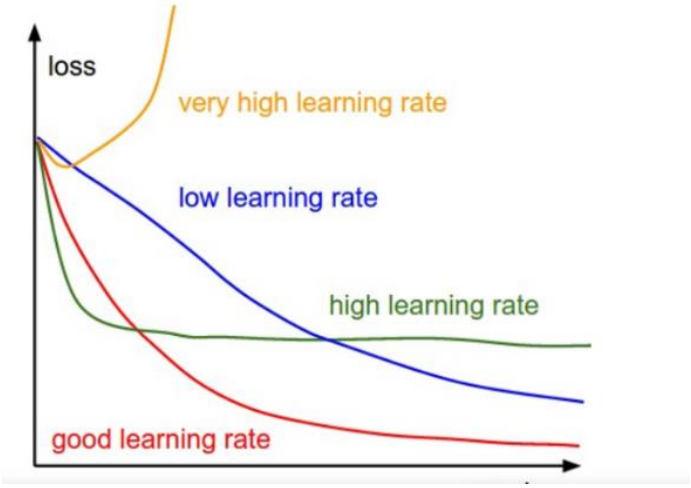

Figura 2.11. Gradient negativ pentru diferite rate de învățare (Sursa [1])

Cele mai folosite rate de învățare sunt: *0.001, 0.003, 0.01, 0.03, 0.1, 0.3*.

3. Scalarea datelor este foarte importantă dacă sunt pe o scară foarte diferită. Dacă nu scalăm datele, curbele de nivel (contururile) vor fi mai înguste și mai înalte, ceea ce înseamnă că va dura mai mult timp pentru convergență.

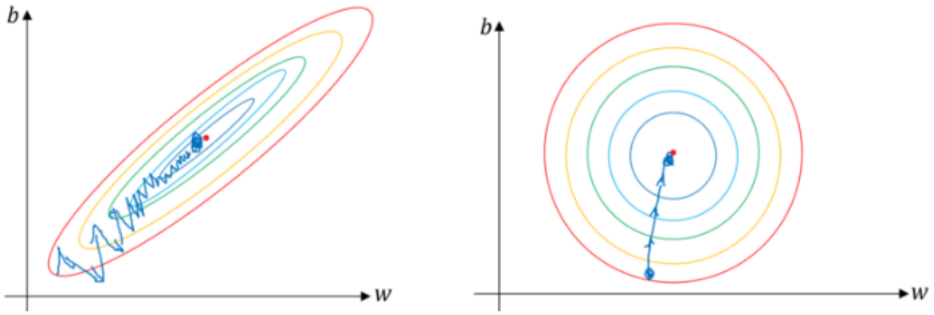

Figura 2.12. Gradient negativ: Curbele ne nivel nenormalizate(stânga) și normalizate(dreapta) (Sursa [23])

$$x_i \leftarrow \frac{x_i - \mu_i}{s_i}$$

Unde x_i = componenta i vectorului de intrare
μ_i = componenta i a vectorului medie
s_i = factor de scalare a compoentei i a vectorului de intrare, de obicei:

$$s_i = range(x_{i\,max} - x_{i\,min})$$

La fiecare iterație, trebui luată în calcul derivata parțială a funcției cost E(W) pentru fiecare parametru în parte:

$$\frac{\partial}{\partial w}E(W) = \nabla_w E \quad \text{Ecuațiile devin:} \quad w = w - \eta\nabla_w E$$

$$\frac{\partial}{\partial w}E(W) = \nabla_b E \qquad\qquad b = b - \eta\nabla_b E$$

Din motive de ilustrare, să presupunem că nu avem bias. În cazul în care panta valorii curente w>0, aceasta înseamnă că suntem în dreaptul optimului w*. Prin urmare, actualizarea va fi negativă și va începe să se apropie de valorile optime ale lui w *. Cu toate acestea, dacă este negativă, actualizarea va fi pozitivă și va crește valorile curente ale lui w pentru a convertge la valorile optime ale lui w*.

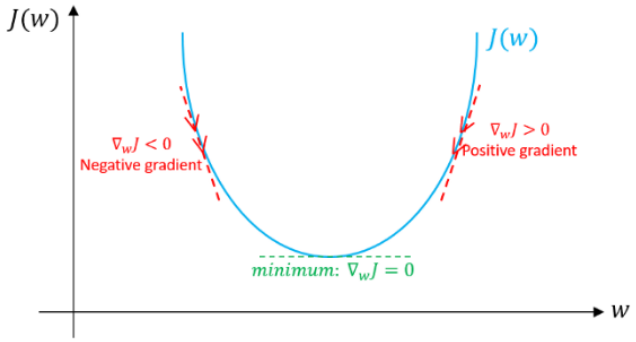

Figura 2.13. Gradient negativ. O ilustrare a modului în care algoritmul gradientului negativ al utilizează primua derivată a funcției cost pentru a urmări minimul local (Sursa [23])

Se continuă procesul până când funcția de cost converge. Acest lucru se întâmplă în momentul în care curba de eroare devine plată și nu se schimbă.
În plus, la fiecare iterație, pasul va fi în direcția care oferă o schimbare maximă, deoarece este perpendiculară la curbele de nivel la fiecare pas.

2.6 Algoritmul Adam

Adam este un algoritm ce poate fi privit ca o combinație dintre RMSprop și SGD varianta cu moment[15]. Acesta utilizează gradienți pătrați pentru a scala rata de învățare ca și RMSprop și profită de varianta cu moment folosind media mobilă a gradientului în loc de însuși gradientul ca și SGD varianta cu moment.

Adam este o metodă de adaptare a ratei de învățare, ceea ce înseamnă că rata de învățare este calculată individual pentru diferiți parametri. Numele ei derivă din estimarea momentului adaptiv și motivul pentru care se numește astfel este că Adam folosește estimări ale primului și

celui de-al doilea moment de gradient pentru a adapta rata de învățare pentru fiecare pondere a rețelei neuronale, unde momentul unei variabile aleatoare este definit ca valoarea așteptată a acelei variabile la puterea lui n [16]:

$$m_n = E[X^n]$$

Primul moment este mediu, iar cel de-al doilea moment este variația necentrată (adică nu se scăde media în timpul calculului varianței). Pentru estimarea momentelor, Adam folosește medii exponențial mobile, calculate pe un gradient evaluat pe un mini-lot actual:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\, g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

Unde m și v sunt medii mobile, g este gradientul pe mini-lotul actual și β - introduce hiper-parametrii noi ai algoritmului. Au valori implicite foarte bune de 0,9 și respectiv 0,999. Vectorii mediilor mobile sunt inițializați cu zerouri la prima repetare.

Pentru a vedea cum aceste valori se corelează cu momentul definit ca în prima ecuație, trebuie analizate valorile așteptate ale mediilor mobile. Deoarece m și v sunt estimări ale primului și al doilea moment, dorim să avem următoarea proprietate:

$$E[m_t] = E[g_t]$$
$$E[v_t] = E[g_t^2]$$

Dacă aceste proprietăți ar fi păstrate, ar însemna că avem estimatori imparțiali. Acum, vom observa că acestea nu sunt valabile pentru mediile noastre mobile [16]. Se extinde valoarea lui m, până când cele mai mici valori ale gradienților contribuie la valoarea globală, deoarece se înmulțesc cu β din ce în ce mai mici. Astfel, putem rescrie formula pentru mediul nostru mobil:

$$m_t = (1 - \beta_1) \sum_{i-0}^{t} \beta_1^{t-i}\, g_i$$

Si vom avea corecția bias-ului pentru estimatorul primului moment:

$$E[m_t] = E\left[(1 - \beta_1) \sum_{i-0}^{t} \beta_1^{t-i}\, g_i\right] = E[g_i](1 - \beta_1) \sum_{i-0}^{t} \beta_1^{t-i} + \zeta = E[g_i]\,(\beta_1^t) + \zeta$$

În primul rând, folosim noua formulă pentru media mobilă pentru a extinde *m* urmând să aproximăm g [i] cu g [t]. Deoarece are loc aproximarea, eroarea ζ apare în formula. În ultima linie folosim doar formula pentru suma unei serii geometrice finite.

Acum trebuie să corectăm estimatorul, astfel încât valoarea așteptată să fie cea dorită. [10] Acest pas este, de obicei, denumit corecția bias-ului. Formulele finale pentru estimatorul de corecție de bias pentru primul si cel de-al doilea moment vor fi după cum urmează:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

Singurul lucru de făcut este să utilizăm aceste medii mobile pentru a mări gradul de învățare individual pentru fiecare parametru. Modul în care se face în Adam este foarte simplu, pentru a efectua actualizarea ponderii:

$$w_t = w_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

2.7 Funcții de activare

Funcțiile de activare a rețelei neurale reprezintă o componentă crucială a învățării profunde. Acestea determină rezultatul unui model de învățare profundă, precizia acestuia și, de asemenea, eficiența computațională a formării unui model - care poate face sau distruge o rețea neurală pe scară largă. Funcțiile de activare au de asemenea un efect major asupra capacității rețelei neuronale de a converge, mai ales viteza de convergență , iar în unele cazuri, funcțiile de activare ar putea împiedica convergența rețelelor neuronale.

Fiecare funcție de activare (sau nelinearitate) are un singur număr și efectuează o anumită operație matematică fixă pe ea. Există mai multe funcții de activare pe care le putem întâlni în practică:

1.Sigmoidă $f(u) = \frac{1}{1+(1+e^{-x})}$

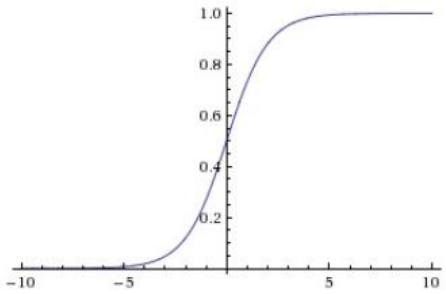

Figura 2.14. Funcția sigomidă (Sursa [1])

2.Tangentă hiperbolică $f(u) = \frac{e^u - e^{-u}}{e^u + e^{-u}}$

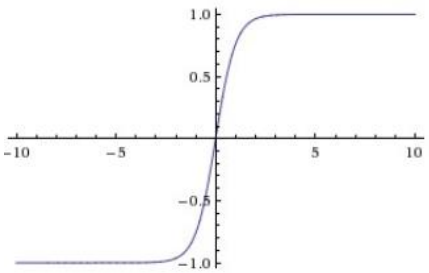

Figura 2.15. Funcția tangentă hiperbolică (Sursa [1])

3. ReLU (Rectified Linear Unit) $f(u) = \max(0, u)$

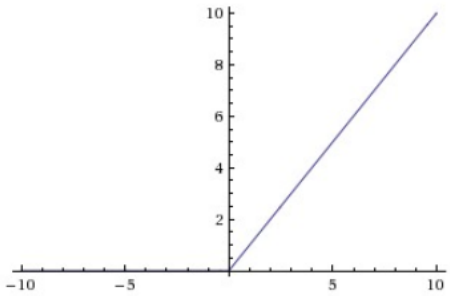

Figura 2.16. Funcția ReLU (Sursa [1])

2.8 Rețele neurale convoluționale

Rețelele neurale profunde (DNN) au prezentat îmbunătățiri semnificative în mai multe domenii, inclusiv viziunea pe calculator și recunoașterea vorbirii. În „Computer Vision", un anumit tip de DNN, cunoscut sub numele de rețele nurale convoluționale (CNN), a demonstrat rezultate de ultimă oră în recunoașterea obiectelor [11] și detecție [12].

O rețea neurală convoluțională este o clasă de rețele neuronale artificiale care utilizează straturi convoluționale pentru a filta intrări având în vedere obținerea de informații utile. Operația de convoluție implică combinarea datelor de intrare (harta caracteristicilor) cu un kernel (filtru) pentru a forma o hartă a caracteristicilor transformate. Filtrele din straturile convoluționale sunt

modificate pe baza parametrilor învățați pentru a extrage cele mai utile informații pentru o anumită sarcină. Rețelele convoluționale se ajustează automat pentru a avea cea mai bună funcționare bazată pe sarcina respectivă.

Rețelele convoluționale sunt compuse dintr-un strat de intrare, un strat de ieșire și unul sau mai multe straturi ascunse. O rețea convoluțională este diferită de o rețea neuronală artificială datorită faptului că neuronii din straturile sale sunt dispuși în trei dimensiuni (lățimie, înălțimie și adâncimie). Acest lucru permite CNN-ului să transforme volumul de intrare de trei dimensiuni într-un volum de ieșire. Straturile ascunse sunt o combinație de straturi de convoluție, straturi de grupare (pooling layers) și straturi conectate complet (fully connected). CNN-urile utilizează straturi convoluționale multiple pentru a filtra volumele de intrare la un nivel mai ridicat de abstractizare[16].

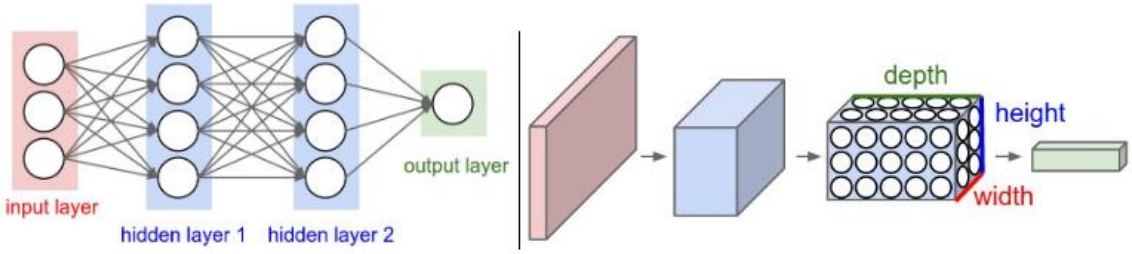

Figura 2.17. Arhitectura rețea neurală artificială (stânga) și Arhitectura rețea neurală convoluțională (dreapta) (Sursa [1])

2.9 Tipuri de straturi pentru CNN-uri

2.9.1 Strat Convoluțional

Stratul de convoluție este blocul central al CNN-ului. Acesta reprezintă partea principală a sarcinii computaționale a rețelei.

Parametrii stratului convoluțional constau într-un set de filtre aplicate. Fiecare filtru este de mici dimensiuni (de-a lungul lățimii și înălțimii), dar se extinde prin toată adâncimea volumului de intrare. De exemplu, un filtru tipic pe un prim strat al unui strat convoluțional
poate avea dimensiunea 5x5x3 (adică 5 lățime și înălțime 5 pixeli și 3 deoarece imaginile au adâncime 3, canalele de culoare). În timpul parcurgerii imaginii, se glisează fereastra (mai precis, se efectuează operația de convoluție) filtrului pe lățimea și înălțimea volumului de intrare și calculează produsele punctuale între intrările filtrului și intrarea în orice poziție. Pe măsură ce glisăm filtrul peste lățimea și înălțimea volumului de intrare, vom produce o hartă de activare bidimensională care oferă răspunsurile acelui filtru la fiecare poziție spațială. Intuitiv, rețeaua va învăța filtrele care se activează atunci când văd un anumit tip de caracteristică vizuală, cum ar fi o margine cu o anumită orientare sau o pată de culoare pe primul strat sau, eventual, modele întregi [1].

Având în vedere dimensiunea volumului de ieșire, avem 3 hiperparametrii: adâncimea, pasul și stratul de zerouri din jurul imagini (zero-padding)

Putem calcula dimensiunea spațială a volumului de ieșire ca funcție de mărimea volumului de intrare (W), dimensiunea câmpului receptiv a neuronilor stratului convoluțional (F), pasul cu care sunt aplicați (S) și cantitatea de zerouri de umplere folosită (P) pe margine. Formula pentru calcularea numărului de neuroni "potriviți" este [16] :

$$n = \frac{(\mathrm{W} - \mathrm{F} + 2\mathrm{P})}{\mathrm{S}} + 1$$

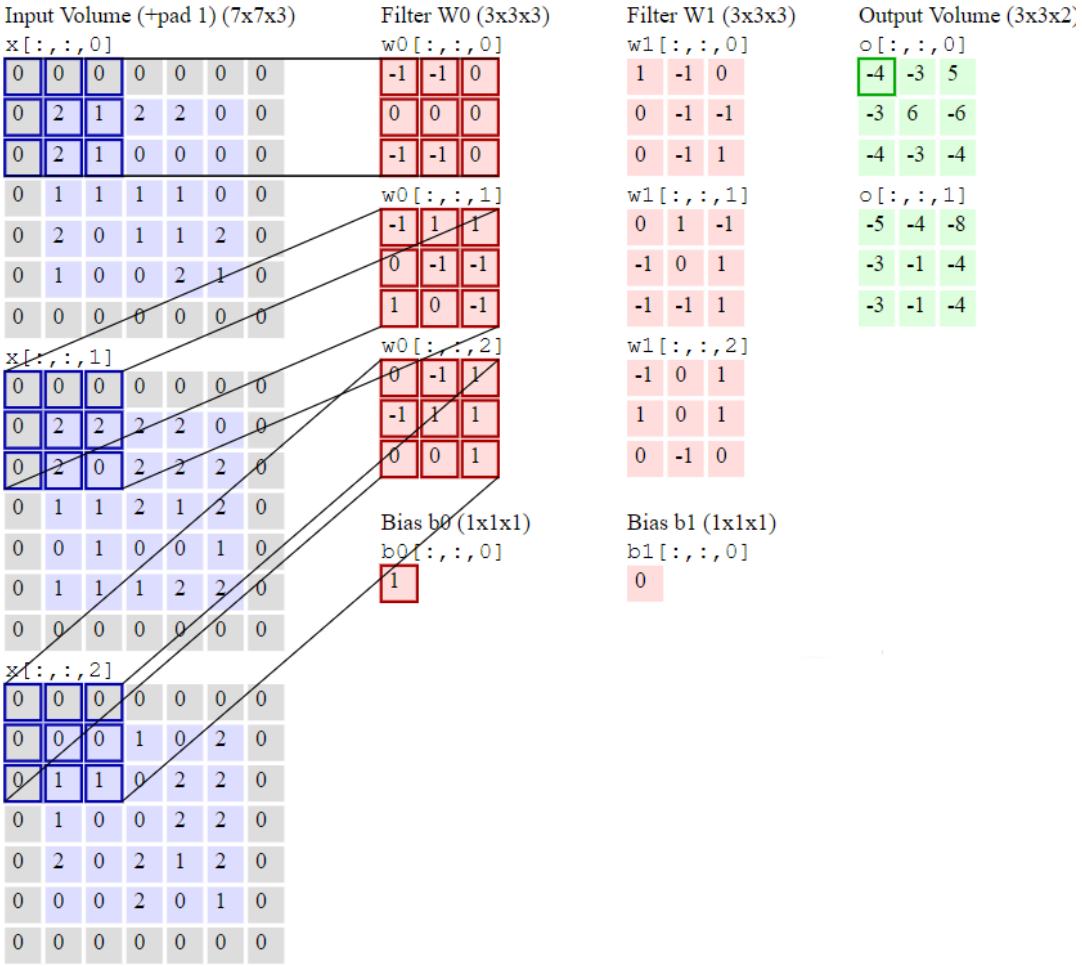

Figura 2.18. Volumul de ieșire (verde) rezultat prin înmulțirea intrării evidențiate (albastru) cu filtrul (roșu) (Sursa[1])

2.9.2 Strat de redimensionare (Pooling Layer)

Este comună introducerea periodică a unui strat Pooling între straturile consecutive convoluționale într-o arhitectură CNN. Funcția sa este de a reduce progresiv dimensiunea spațială, pentru a reduce cantitatea de parametri și de calcul în rețea și, prin urmare, de a evita supraantrenarea. Stratul „Pooling" funcționează independent la fiecare strat din adâncimea intrării și o redimensionează spațial, utilizând operația MAX. Forma cea mai comună este un strat de pooling cu filtre de dimensiune 2x2 aplicate cu un pas de 2, fiecare strat din adâncime de intrare de 2x2, eliminând 75% din activări. Fiecare operatie MAX ar lua in acest caz un numar de maxim 4 eșantioane (o regiune de 2x2). Dimensiunea adâncimii rămâne neschimbată având următoarele trăsături[1]:

Având un volum de intrare: W1 x H1 x D1 , spațiul filtrului F, Pasul S produce un volum de ieșire de dimensiune: W2 x H2 x D2 unde [16]:

$$W2 = \frac{(\mathrm{W1} - \mathrm{F})}{\mathrm{S}} + 1$$

$$H2 = \frac{(\mathrm{H1} - \mathrm{F})}{\mathrm{S}} + 1$$

$$D2 = D1$$

Stânga:Volumul de intrare de dimensiune [224x224x64] este reunit cu dimensiunea filtrului 2, pasul 2 cu volumul de ieșire de dimensiune [112x112x64]. Se observă că adâncimea volumului este păstrată. Dreapta: cea mai comună operațiune de downsampling este max, rezultând o reducere maximă, prezentată cu un pas de 2.

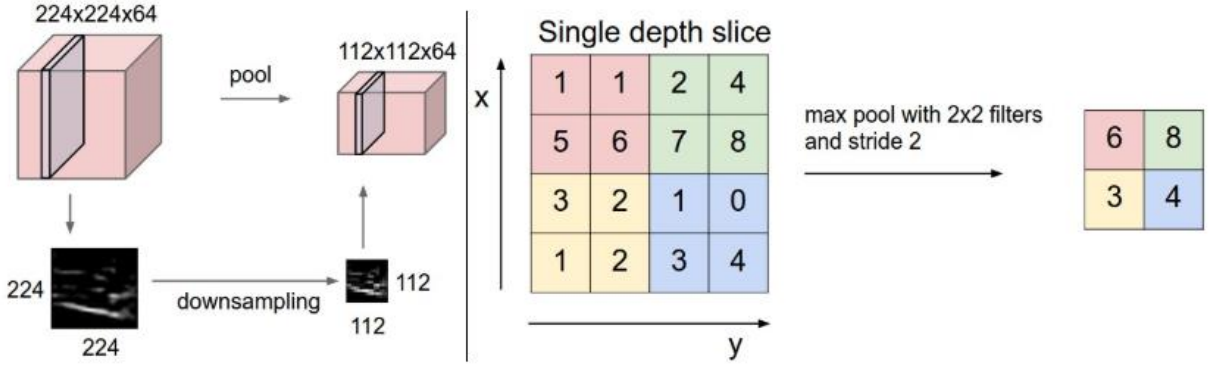

Figura 2.19. Exemplu Pooling (Sursa[1])

2.9.3 Strat complet conectat(Fully connected)

Neuronii dintr-un strat complet conectat au conexiuni complete la toate activările din stratul anterior, așa cum se observă în rețelele neuronale obișnuite (ANN). Activarea lor poate fi astfel calculată cu o multiplicare a matricei adăugându-se și bias-ul[1].

Capitolul 3. Recunoașterea gesturilor mâinii folosind o rețea neurală convoluțională

Recunoașterea automată a gesturilor este importantă pentru multe aplicații din lumea reală, cum ar fi recunoașterea limbajului semnelor și interacțiunea om-robot (HRI).

Lucrarea de fața are ca scop realizarea unei baze de date și a unei rețele neurale convoluționale în vederea prezicerii uneia dintre cele 10 clase de gesturi ale mâinii. Având în vedere acest lucru, au fost considerate 2 metode testare a funcționalității proiectului: prima metoda este repezentată de recunoașterea mâinii în timp real si cea de-a doua metoda prin testarea mai multor imagini dintr-o bază de dat de testare.

Etapele generale de realizare ale proiectului constau în patru părți principale:

1. Achiziția de date - creerea unei baze de date personle (set de date pentru antrenare și testare) pentru fiecare clasa de gesturi.

2. Un pas de preprocesare a imaginii gesturilor pentru a reduce complexitatea setului de date ce devine intrare în rețeaua neurală.

3. Antrenarea sistemului cu în cele 2 situații: antrenarea cu baza de date etalon standardizată si antrenarea cu baza de date creată pentru a demonstra capacitatea sistemului de clasificare în situații diferite.

4. Testarea sistemului în timp real și testarea statică ce implică clasificarea fiecărui gest.

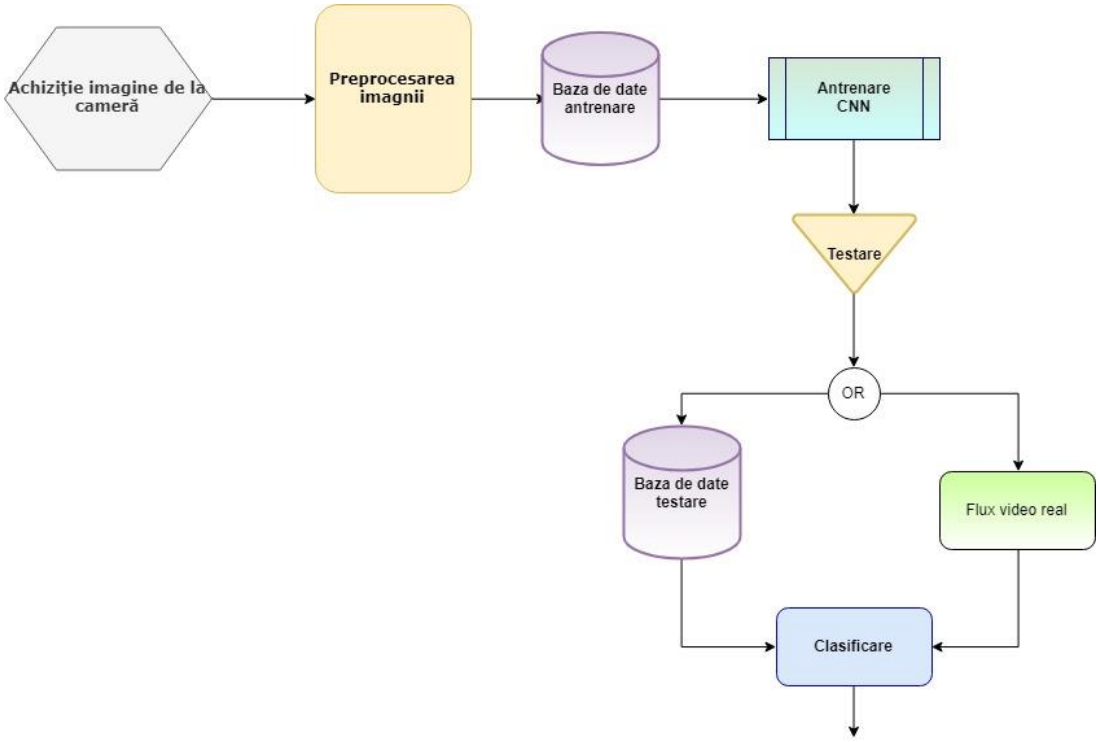

Figura 3.1. Schema bloc

3.1 Achiziția de date

În prima fază, achiziția datelor a fost efectuată cu ajutorul camerei web dintr-un flux video în care se realizau frame-uri în intervale de câte 5 secunde. Formatul bazei de date a fost realizat asemănător cu cel al bazei de date standard etalon pentru a efectua o comparație calitativă asupra rezultatelor ce se vor obține. Baza de date standard conține imaginii cu 10 gesturi diferite, imparțite in 2 categorii: imagini pentru antrenare si imagini pentru testare. Baza de date standard a imaginilor este compusă din 10.000 de imagini, 1000 pentru fiecare gest din cadrul antrenării [20] și 400 de imagini pentru testare. Baza de date creată conține 1200 de imagini: 1000 de imagini pentru antrenare, 100 pentru fiecare gest si aproximativ 150 de imagini pentru testare.

Seturile sunt stucturate in 2 foldere pentru fiecare baza de date:

Set 1:

Traindata

Gesture_0\
gest0_1.jpg\
gest0_2.jpg\
gest0_3.jpg\
…
gest0_1000.jpg\
Gesture_1\
Gesture_2\
….
Gesture_9\

Testdata

Testdata_0\
gest0_1.jpg\
gest0_2.jpg\
gest0_3.jpg\
…
gest0_1000.jpg\
Testdata_1\
Testdata_2\
….
Testdata_9\

Set 2:

mydb

Gest_0\
0.jpg\
1.jpg\
2.jpg\
…
100.jpg\
Gest_1\
Gest_2\
….
Gest_9\
Test_9\

mydbtest

Test_0\
0.jpg\
1.jpg\
2.jpg\
…
15.jpg\
Test_1\
Test_2\
….

Fiecare gest este descris in figura de mai jos:

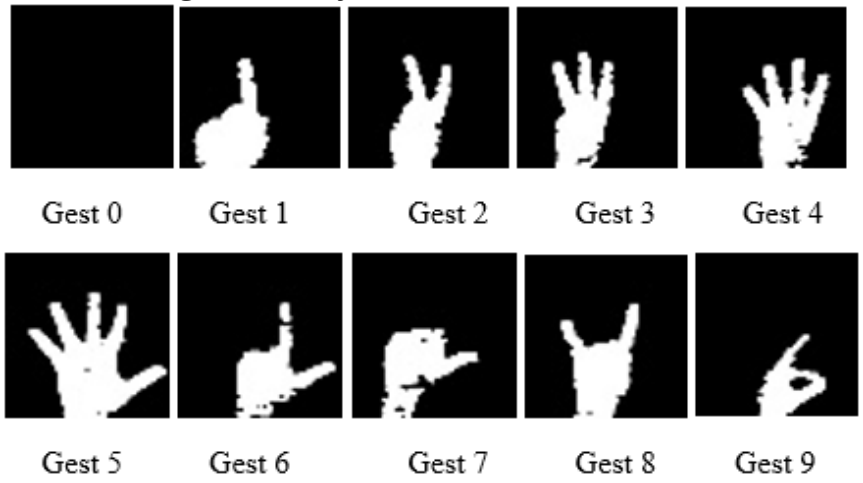

Figura 3.2. Ilustrarea celor nouă gesturi

Prelucrarea se face în trei etape: conversia spațiului de culoare pentru extracția zonei pielii, operațiile morfologice până la eliminarea zgomotului și truncherea imaginii pentru a facilita extragerea caracteristicilor.

Pentru a ajunge la imaginea finala din starea brută au fost necesare o serie de operații pentru a facilita în continuare buna desfășurare a programului. În primă instanță, aplicăm un filtru Gaussina de blurare pentru a reduce zgomotul și detaliile proeminente. Matematic, aplicând un filtru Gaussian de blurare are același efect ca și convoluția dintre imagine și funcția Gaussiană:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{\frac{x^2+y^2}{2\sigma^2}}$$

unde x este distanța de la origine la axa orizontală, y este distanța de la originea la axa verticală și σ este deviația standard a distribuției Gaussiane.

Se efectuează în continuare segmentarea, procesul de extragere a obiectelor de interes dintr-o imagine. În cazul nostru, obiectul de interes este mâna utilizatorului.

Există multe abordări posibile pentru a rezolva această problemă, fiecare cu o complexitate și precizie diferite. (Francesco 2018) Datorită lipsei unor constrângeri puternice asupra compoziției și iluminării scenei, a trebuit să excludem tehnicile de segmentare pe baza imaginilor cu praguri în tonuri de gri. Chiar dacă aceste tehnici sunt în general rapide și fiabile (cu imaginile potrivite), histograma tipică a imaginii nu are o separare distinctă a modurilor, prin urmare, analiza în tonuri de gri nu este o opțiune fiabilă. Am hotărât că, cea mai bună abordare pentru a rezolva problema, în termeni de complexitate și fiabilitate, este de a segmenta mâna pornind de la culoarea pielii utilizatorului.

Pentru a găsi culoarea pielii, imaginea este mai întâi convertită în spațiul de culoare HSV. După câteva teste, acest spațiu de culoare a rezistat cel mai mult la schimbările de nuanță și tonicitate a culorii, dar nu este singura opțiune. Cel mai important avantaj al utilizării HSV este abilitatea de a ignora umbrele de pe mâini, pur și simplu ignorând al treilea canal (V).Imaginile eșantionului sunt împărțite în cele trei canale: H, S și V. Pentru fiecare canal se calculează valoarea medie, de unde se calculează valori prag scăzute și înalte, cu excepția canalului "value" (V).

Pentru transformarile din RGB în HSV avem următoarele formule:

$$H = arccos \frac{0.5x((R-G)+(R-B))}{\sqrt{(R-G)^2+(R-B)+(G-B)}}$$

$$S = 1 - 3\frac{min(R,G,B)}{(R+G+B)}$$

$$V = \frac{1}{3}(R+G+B)$$

Binarizarea cadrului se face folosind funcția inRange din OpenCV. Operatorul convertește pur și simplu valorile în 1 toți pixelii care se află între pragul inferior și superior și în 0 toți ceilalți pixeli. După binarizare, imaginea rezultată este puțin zgomotosă, din cauza falsurilor pozitive. Pentru a curăța imaginea și a elimina acest lucru, se aplică un operator de deschidere, cu un element de structurare circular 11x11. De asemenea, se aplică o dilatare doar în cazul în care părțile mâinii au fost detașate după binarizare (de multe ori degetele sunt desprinse de la mână).

3.2 Structura arhitecturii rețelei neurale

Structura rețelei este implementată folosind PyCharm cu ajutorul librăriei Keras. Keras este un API pentru rețele neuronale de nivel înalt, scris în Python și capabil să ruleze peste TensorFlow, CNTK sau Theano. Ea a fost dezvoltată pentru a pune accentul pe facilitarea de a experimenta într-un mod rapid. Am utilizat Keras deoarece este o bibliotecă pentru învățarea profundă care permite prototipare ușoară și rapidă (prin simplitate, modularitate și extensibilitate), sprijină atât rețelele de convoluție, cât și rețelele recurente și se poate executa și pe CPU sau GPU.

Arhitectura de bază a CNN este compusă din trei straturi convoluționale, trei straturi de tip Pooling și două straturi complet conectate cu funcții de activare de tip ReLU .Ultimul strat complet conectat are funție de activare de tip softmax.

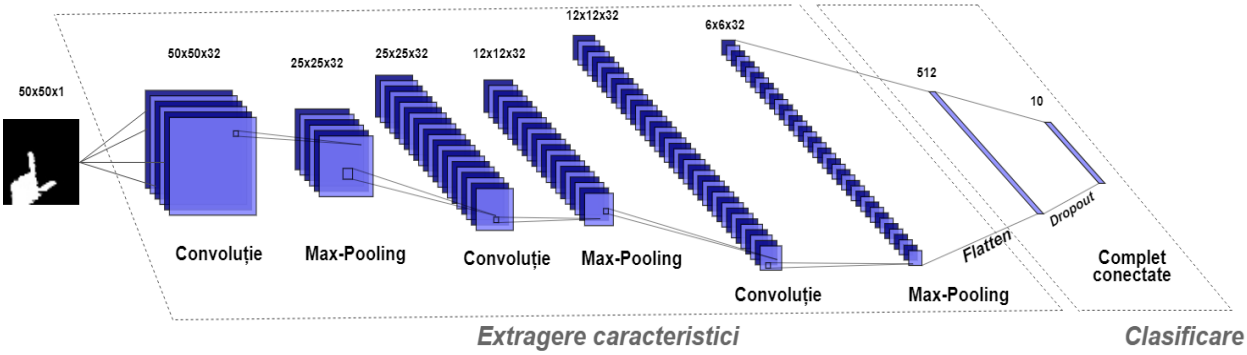

Figura 3.3. Arhitectura rețelei

Primul strat convoluțional are o dimensiune a nucleului de 3x3 pixeli care conține 32 de filtre (harți ale trăsăturilor) cu funții de activare ReLU. Acest strat are imagini cu valori de 50x50 pixeli ca intrare. Următorul strat este un strat de redimensionare (Pooling) care este configurat cu o dimensiune de 2×2 cu pas de 2. Stratul de redimensionare (Max-Pooling) utilizează valoarea maximă pentru a reduce progresiv dimensiunea spațială a reprezentării imaginii și de a reduce suma de hiperparametrii și calcul computațional al rețelei și, de semenea să controlăm fenomenul

de supraantrenare. Acest strat extrage regiuni de 2×2 pixeli ai hărții trăsăturilor, păstrând maximul și eliminând toate celelalte valori.

Următorul strat ascuns este un strat convoluțional care are dimensiunea nucleului de 3×3 pixeli și conține, de asemenea, 32 de filtre (harți ale trăsăturilor) și funcție de activare ReLU. Acest strat este urmat de un alt strat Max-Pooling asemănator cu cel descris anterior. Următoarele 2 straturi sunt asemănătoare cu cele descrise anterior,

Apoi, cu ajutorul funcției Flatten se convertesc datele matriceale bidimensionale într-un vector (o singură coloană), permițând ca ieșirea finală să fie procesată de stratul următor complet conectat (fully conected) pentru a obține straturile următoare.

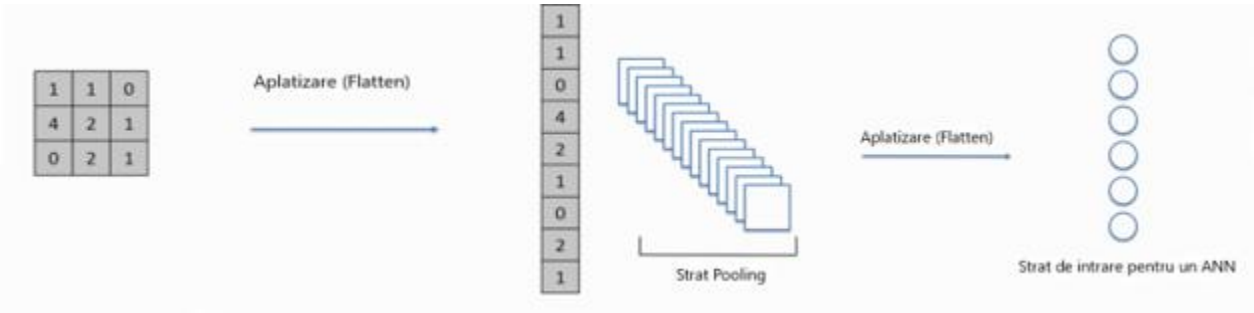

Figura 3.4. Aplicare funcției Flatten la nivel numeric(stânga) și arhitectural(dreapta) (Sursa [16])

Primul strat complet conectat cu funcția de activarea ReLU conține 512 de neuroni. Peste acest strat se aplică funcția Dropout. Aceasta este cea mai simplă metodă de regularizare a rețelelor neurale. Practic, în timpul antrenării, un sfert din neuroni de pe un anumit strat vor fi dezactivați (valoare aleasă de mine, se poate alege orice valoare). Acest lucru îmbunătățește generalizarea, deoarece forțează stratul să învețe cu diferiți neuroni același "concept". În timpul fazei de predicție, Dropout-ul este dezactivat. Pentru a implementa această dezactivare a neuronilor, creăm o mască (de zero și unu) în timpul propagării înainte. Această mască este aplicată ieșirilor stratului în timpul antrenării și memorată în cache pentru utilizare ulterioară în cazul unep propagări înapoi.

Ultima parte a structurii CNN este stratul de ieșire care cuprinde o funcție de activare Softmax și conține 10 neuroni, câte unul pentru fiecare clasă de recunoaștere a gesturilor mâinii. ieșire este maparea datelor la clasele finale pentru mână recunoașterea gesturilor.

Tipul stratului	Configurația stratului	Forma la ieșire	Parametrii
Input	-	(50, 50, 1)	0
Conv2D-1	32 filtre, nucleu 3x3, ReLU	(50, 50, 32)	320
Max-Pooling-1	nucleu 2x2, pas=2	(25, 25, 32)	0
Conv2D-2	32 filtre, nucleu 3x3, ReLU	(25, 25, 32)	9248
Max-Pooling-2	nucleu 2x2, pas=2	(12, 12, 32)	0
Conv2D-3	32 filtre, nucleu 3x3, ReLU	(12, 12, 32)	9248
Max-Pooling-3	nucleu 2x2, pas=2	(6, 6, 32)	0
Flatten	-	(1152)	0
Dense-1	512 neuroni	(512)	590336
Dropout	-	(512)	
Dense-2	10 clase, Softmax	(10)	5130

Tabelul 3.1. Structura rețelei

Număr total de parametrii: 614,282
Parametrii antrenați: 614,282
Parametrii neantrenați: 0

Pentru optimizare, am ales o altă metoda ce calculează o rată de învățare adaptivă pentru fiecare parametru si anume Adam (Adaptive Estimation Moment) [14]. Numele ei derivă din estimarea momentului adaptiv și motivul pentru care se numește în acest fel este pentru că Adam folosește estimări ale primului și celui de-al doilea moment de gradient pentru a adapta rata de învățare pentru fiecare pondere a rețelei neurale.

Capitolul 4. Experimente și rezultate

Recunoaștere automată a gesturilor mâinii cu ajutorul Computer Vision-ului este importantă pentru diferite aplicații din lumea reală, cum ar fi ca recunoașterea limbajului semnelor (SLR). Recent, datorită disponibilității pe scară largă a camerelor digitale, cercetarea în domeniul recunoașterii limbajului semnelor a crescut dramatic. Cu toate acestea, recunoașterea gesturilor mâinii încă se confruntă cu provocări din cauza unor factori precum complexitatea structurii gesturilor și a diferențelor între mărimea mâinii și postura acesteia sau variația de lumină și de fond, care pot influența performanța algoritmilor de recunoaștere [17]. De asemenea, există mai multe variante ale aceluiași gest, deoarece oamenii pot efectua același gest de mână în mod diferit; într-adevăr, aceeași persoană poate semna același gest de mână diferit. În plus, recunoașterea gesturilor bazate pe viziune este de asemenea susceptibilă la factorii de mediu, cum ar fi observabilitatea mâinii [18], [19].

Abordări anterioare privind recunoașterea gesturilor mâinilor utilizând algoritmi de prelucrare de imagini clasici, de obicei, constau în doi pași principali. Scopul primului pasul este

de a extrage caracteristicile principale și scopul celui de-al doilea pas este de a aplica un algoritm de învățare a mașinilor pentru a efectua clasificarea.

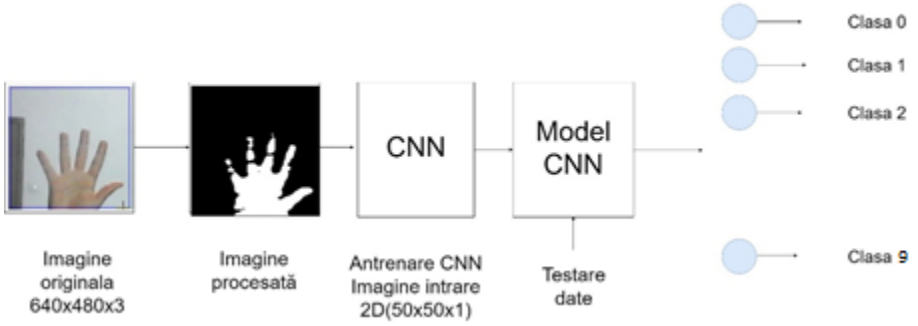

Figura 4.1. Pașii pentru realizarea clasificării

În continuare, voi evalua calitatea rețelei neurale convoluționale în funcție de prezicerea clasei de apartenență pentru setul de imagini de testare și gesturile propuse în timp real, cu care aceasta nu a interacționat niciodată. Pentru a demonstra acest lucru în diferite condiții, am considerat două ipoteze:

- Rețeaua este antrenată cu baza de date standard (etalon) verficându-se acuratețea cu baza de date de testare standrad (etalon) , baza de date de testare creată personal și predicția în timp real
- Rețeaua este antrenată cu baza de date creată personal verficându-se acuratețea în acelasi mod menționat anterior

4.1 Rețea antrenată cu baza de date standard (etalon)

În urma atrenării, am putut observa o creștere semnificativă a acurateții ce ajunge la 100% si o scadere a funcției cost până la 8,8517e-06. Calsificatorul Softmax are altă funcție cost față de SVM[3]. În clasificatorul Softmax, maparea funcțiilor $f(x_i;X)=Wx_i$ rămâne neschimbată, însă interpretăm aceste scoruri ca fiind probabilitățile logaritmice nenormalizate pentru fiecare clasă și înlocuiește „hinge loss-ul” cu o funcție cost de entropie încrucișată care are forma:

$$L_i = -log(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}) \text{ sau echivalent cu } L_i = -f_{y_i} + log \sum_j e^{f_j}$$

unde unde folosim notația f_j pentru a însemna elementul j al vectorului de clase f . Ca și înainte, pierderea totală pentru setul de date este media lui L_i pentru toate exemplele de antrenare alături de un termen de regularizare R (W)[18].

Precizia este definită ca find numărul corect de predicții împărțită la numarul total de intrări.

Voi prezenta în continuare starea antrenării în primele si ultimele trei epoci din cele cincizeci efectuate.

```
Epoch 1/50
2019-06-23 14:11:46.222220: I tensorflow/core/platform/cpu_feature_guard.cc:14
 - 19s - loss: 0.5187 - acc: 0.8487

Epoch 00001: loss improved from inf to 0.51868, saving model to m.h5
Epoch 2/50
 - 19s - loss: 0.0328 - acc: 0.9891

Epoch 00002: loss improved from 0.51868 to 0.03278, saving model to m.h5
Epoch 3/50
 - 19s - loss: 0.0123 - acc: 0.9951

Epoch 00003: loss improved from 0.03278 to 0.01227, saving model to m.h5
```

Figura 4.2. Primele 3 epoci din antrenarea rețelei cu baza de date standard (etalon)

```
Epoch 48/50
 - 22s - loss: 1.9324e-05 - acc: 1.0000

Epoch 00048: loss did not improve from 0.00001
Epoch 49/50
 - 22s - loss: 5.2806e-06 - acc: 1.0000

Epoch 00049: loss did not improve from 0.00001
Epoch 50/50
 - 22s - loss: 8.8517e-06 - acc: 1.0000

Epoch 00050: loss did not improve from 0.00001
```

Figura 4.3. Ultimele 3 epoci din antrenarea rețelei cu baza de date standard (etalon)

Se observa că, cel puțin din epoca 48, funția cost nu mai este înbunătățită semnificativ, fapt ce evidențiază ca antrenarea se putea efectua cu mai puține epoci. Din considerente de timp, dat fiind faptul ca antrenarea rețelei durează cel puțin o oră, comparațiile au fost efectuate cu 50 de epoci fiecare.

4.1.1 Testarea cu baza de date de testare standard (etalon)

Am ales o matrice de confuzie pentru a descrie performanța modelului de clasificare (sau "clasificator") pentru seturile de date de testare pentru care sunt cunoscute valorile reale. În cazul bazei de date de tesare standard am obținut următoarele rezultate:

		Clasă prezisă									
		Gest 0	Gest 1	Gest 2	Gest 3	Gest 4	Gest 5	Gest 6	Gest 7	Gest 8	Gest 9
Clasă reală	Gest 0	40	0	0	0	0	0	0	0	0	0
	Gest 1	0	38	0	0	0	0	0	2	0	0
	Gest 2	0	9	19	4	0	0	0	0	8	0
	Gest 3	0	0	0	28	0	0	0	0	12	0
	Gest 4	0	0	0	5	35	0	0	0	0	0
	Gest 5	0	0	0	1	5	34	0	0	0	0
	Gest 6	0	0	0	0	0	0	39	0	1	0
	Gest 7	0	0	6	0	4	0	4	26	0	0
	Gest 8	0	0	0	0	0	0	0	0	40	0
	Gest 9	0	16	0	3	0	0	0	0	0	21

Tabelul 4.1. Matrice de confuzie în cazul testării bazei de date standard (etalon)

Precizia este o altă măsură care poate fi utilă atunci când problema are clase bine echilibrate și dorim să punem accentul pe potrivirile exacte. Din punct de vedere grafic, este suma elementelor de pe diagonala principală a matricei de confuzie împărțită la numărul total de predicții făcute:

$$Precizie = 80\%$$

Se observă ca există probleme asupra gestului 2, unde doar 19 imagini sunt clasificate corect(mai puțin de 50%), restul fiind clasificate greșit ca fiind din clasa gestului 1 și 8. De asemenea, gestul 9 este clasificat corect în proporție de 50%, facundu-se confuzie cu gestul 2.

4.1.2 Testarea cu baza de date de testare proprie

Pentru a putea realiza o comparație, vom evalua permformanța bazei de date create personal asemănător cazului anterior, cu o matrice de confuzie. De precizat faptul că, dimensiunea bazei de date este mai mică și anume 15 imagini de test pentru fiecare gest în parte.

		Clasă prezisă									
		Gest 0	Gest 1	Gest 2	Gest 3	Gest 4	Gest 5	Gest 6	Gest 7	Gest 8	Gest 9
Clasă reală	Gest 0	15	0	0	0	0	0	0	0	0	0
	Gest 1	0	15	0	0	0	0	0	0	0	0
	Gest 2	2	5	8	0	0	0	0	0	0	0
	Gest 3	0	0	7	8	0	0	0	0	0	0
	Gest 4	0	0	0	14	1	0	0	0	0	0
	Gest 5	0	0	0	0	0	15	0	0	0	0
	Gest 6	0	0	0	0	0	0	15	0	0	0
	Gest 7	0	0	0	0	0	0	1	14	0	0
	Gest 8	0	2	12	1	0	0	0	0	0	0
	Gest 9	0	3	2	2	0	0	0	0	0	8

Tabelul 4.2. Matrice de confuzie în cazul testării bazei de date create personal

$$Precizie = 66\%$$

Se observă că precizia a scăzut cu 14% , fiind de așteptat acest lucru, deoarece antrenarea a fost facută pe imagini cu un anumit mediu de iluminare, dar și la o anumită distanță față de camera, date ce nu puteau fi interpretate în momentul realizării bazei mele de date. Ca și în cazul anterior avem probleme de predicție, de aproximativ 50% prezicere corectă în cazul gestului 2,gestului 3 și gestului 3,dar iar în cazul gestului 8 și gestului 4 nu a fost nicio imagine clasificată corect, imaginile fiind clasificate ca aparținând clasei gestului 2, respectiv gestului 3.

4.1.3 Testarea în timp real

Pentru a testa în timp real funționalitatea programului am folosit camera web a laptop-ului cu rezoluția de 640 x 480 pixeli. Din considerente de simplitate, în fluxul video a fost trasat un chenar cu ajutorul funcțiilor din librăria OpenCV în care era poziționată mâna pentru a fi detectată. În stanga acesteia era afișat chenarul respectiv, cu funcțiile de procesare a imaginii aplicate și descrise în capitolul 1. În consolă este afișata predicția în timp real alături de timpul necesar pentru a prezice gestul efectuat. În imaginile de mai jos vor fi prezentate o serie de gesturi efectuate alături de fluxul video în timp real, imaginea gestului mâinii din interior chenarului prelucrată și în consolă predicția gestului.

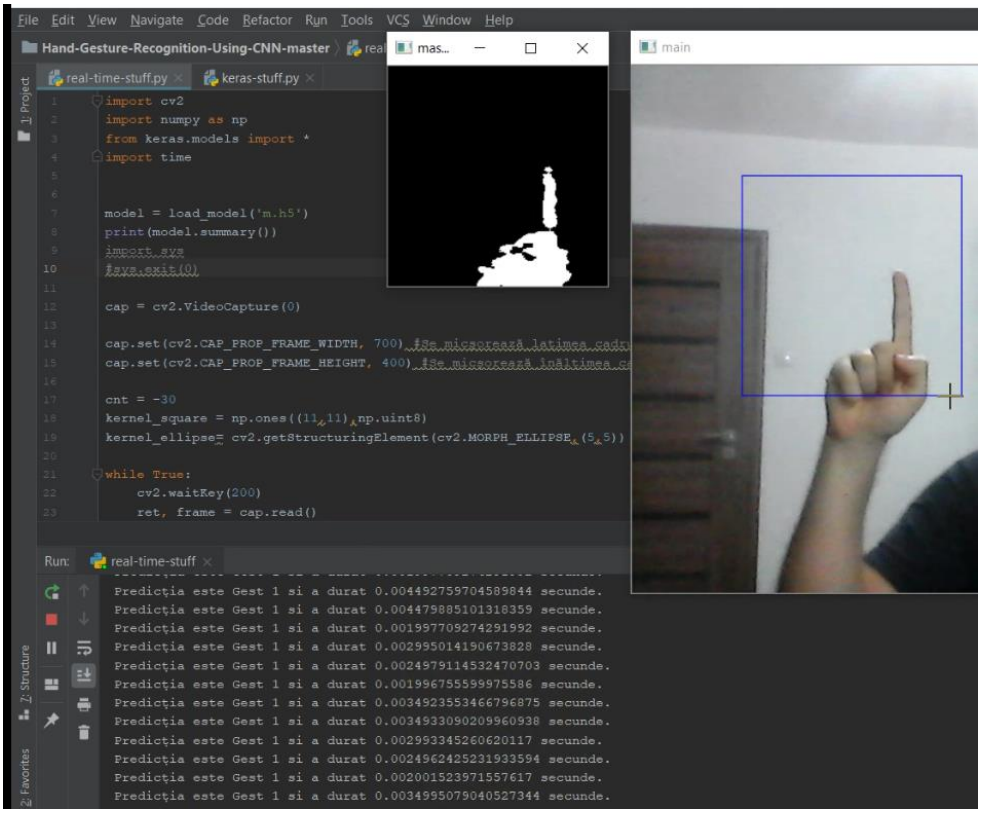

Figura 4.4. Realizare si predicție Gest 1

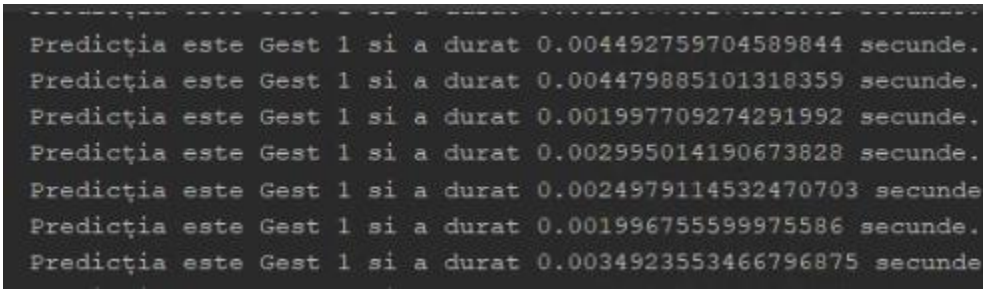

Figura 4.5. Predicția și durata de predicție Gest 1

Se observa că sistemul a detectat cu succes gestul 1. Nu a existat nicio detecție eronată deci putem spune că pentru această clasă avem o precizie de 100%.

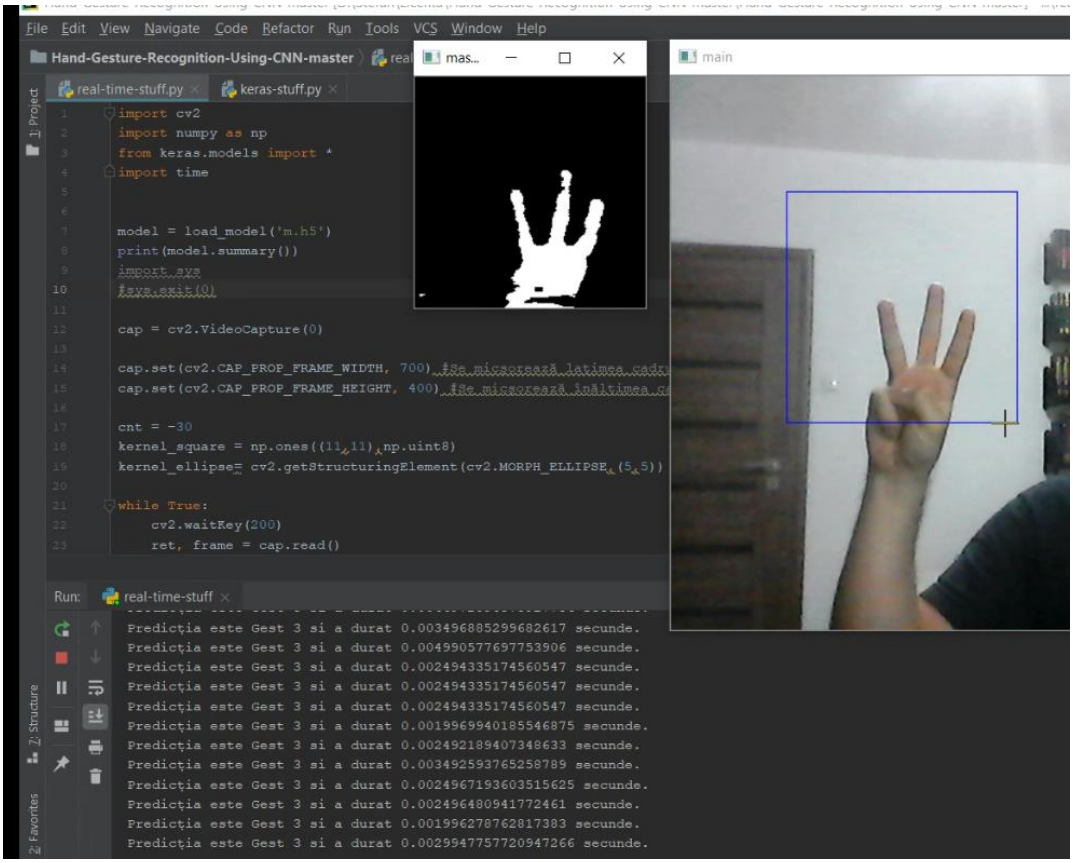

Figura 4.6. Realizare si predicție Gest 3

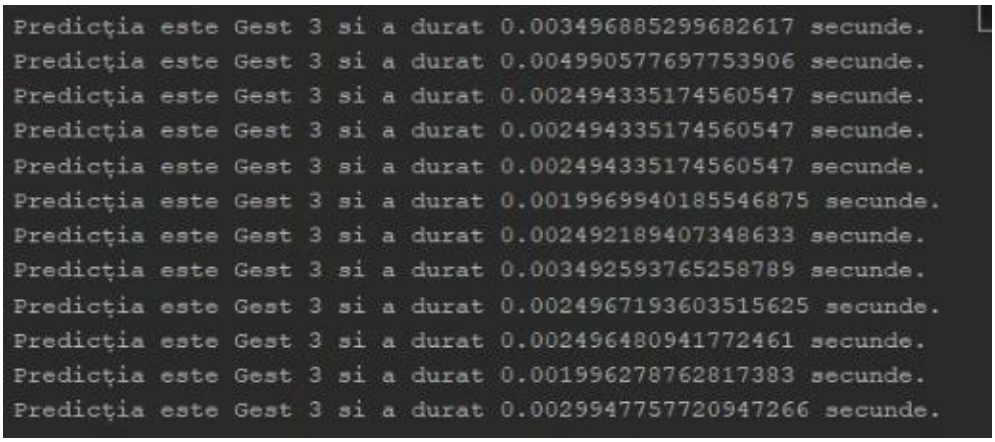

Figura 4.7. Predicția și durata de predicție Gest 3

Ca și în cazul anterior, gestul 3 este detectat cu succes fară a fi confundat cu alte gesturi din baza de date. Nici în acest caz nu avem predicții eronate.

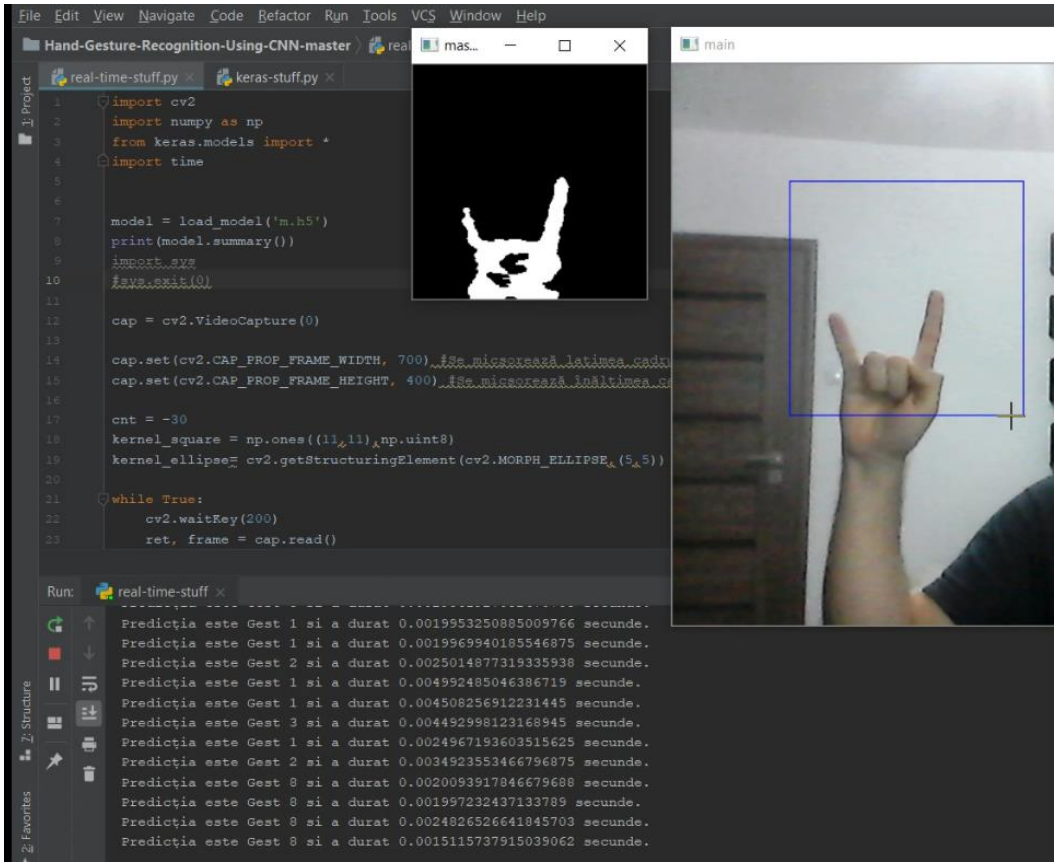

Figura 4.8. Realizare si predicție Gest 8

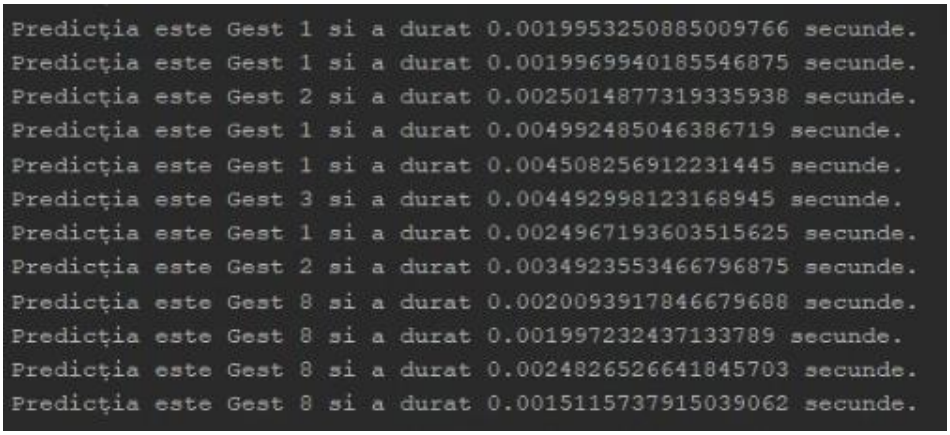

Figura 4.9. Predicția și durata de predicție Gest 8

În acest caz, după cum se poate observa, gestul 8 este foarte ușor confundat cu gesturile 1,2 și 3 (în special cu gestul 1). Acest fapt se datorează în principal prelucrării imaginii, deoarece masca binară a mâinii nu este foarte precisă, iar degetul mic devine foarte scurt. Totuși, observăm că sistemul a fost capabil în jumătate din predicții să îl recunoască corect.

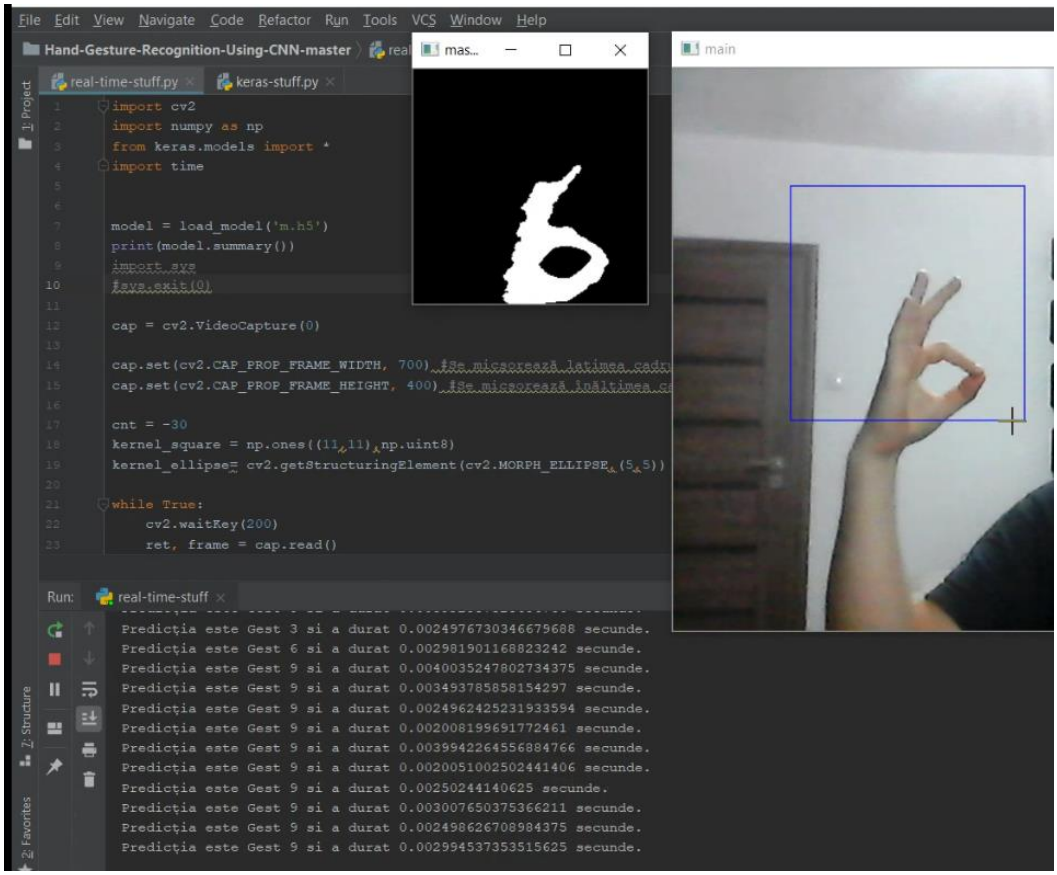

Figura 4.10. Realizare si predicție Gest 9

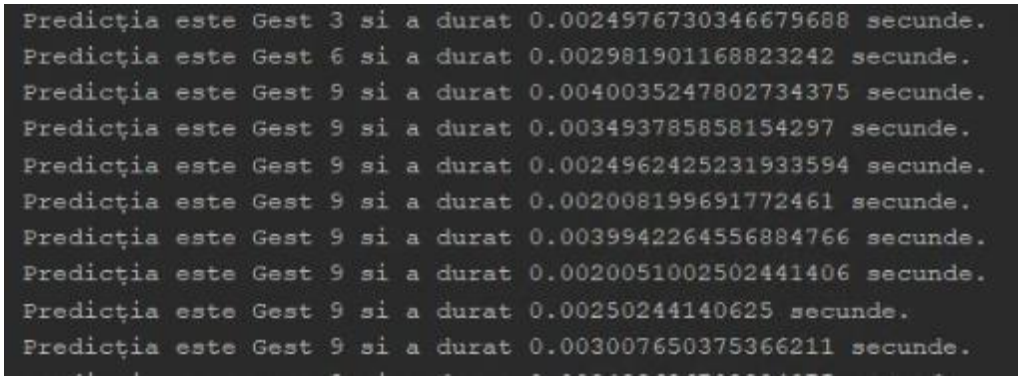

Figura 4.11. Predicția și durata de predicție Gest 9

Pentru gestul 9 avem o situație asemănatoare cu cea anterioară, însă în acest caz sistemul prezice corect in 80% din cazuri. Acest gest este confundat cu gestul 3 deoarece, în urma binarizării, unele imagini pot apărea ca fiind o mână cu 3 degete ridicate, pierzându-se din forma inițială a mâinii. De asemenea, se mai confundă și cu gestul 6.

4.2 Rețea antrenată cu baza de date personală

Ca și în cazul prezentat anterior, voi evidenția funcția cost și precizia din timpul antrenării ilustrând primele și ultimele trei epoci:

```
Epoch 1/50
2019-06-23 22:44:48.280241: I tensorflow/core/platform/cpu_feature_guard.cc:1
 - 3s - loss: 2.1216 - acc: 0.2334

Epoch 00001: loss improved from inf to 2.12165, saving model to m.h5
Epoch 2/50
 - 2s - loss: 1.5063 - acc: 0.5420

Epoch 00002: loss improved from 2.12165 to 1.50634, saving model to m.h5
Epoch 3/50
 - 2s - loss: 1.0232 - acc: 0.6815

Epoch 00003: loss improved from 1.50634 to 1.02324, saving model to m.h5
Epoch 4/50
 - 2s - loss: 0.6126 - acc: 0.8111
```

Figura 4.12. Primele trei epoci antrenată cu baza de date personală

```
Epoch 00047: loss did not improve from 0.00016
Epoch 48/50
 - 3s - loss: 3.3840e-04 - acc: 1.0000

Epoch 00048: loss did not improve from 0.00016
Epoch 49/50
 - 3s - loss: 3.0767e-04 - acc: 1.0000

Epoch 00049: loss did not improve from 0.00016
Epoch 50/50
 - 2s - loss: 2.0689e-04 - acc: 1.0000

Epoch 00050: loss did not improve from 0.00016
```

Figura 4.13. Ultimele trei epoci antrenată cu baza de date personală

Comparativ cu situația precedentă, pornim de la prima iterație cu o funcție cost mai mare de 2,1216 și o precizie mai mică, dar la cea de-a cincizecea epocă funcția cost este destul de mica (2,0689e-4) și precizia de 100% pentru a putea realiza o comparație calitativă. De menționat faptul că durata de antrenare a sistemului a fost cu mult mai mică decât în cazul anterior datorită dimensiunii de 10 ori mai mică față de baza de date standard.

4.2.1 Testarea cu baza de date de testare standard (etalon)

Am realizat matricea de confuzie pentru a determina precizia clasificării rețelei neurale convoluționale antrenată cu baza de date creată personal:

$$Precizie = 51\%$$

		Clasă prezisă									
		Gest 0	Gest 1	Gest 2	Gest 3	Gest 4	Gest 5	Gest 6	Gest 7	Gest 8	Gest 9
Clasă reală	Gest 0	40	0	0	0	0	0	0	0	0	0
	Gest 1	0	10	0	0	0	0	0	0	0	30
	Gest 2	0	0	27	4	0	0	0	0	0	13
	Gest 3	0	17	0	0	0	0	6	0	0	17
	Gest 4	0	2	0	0	0	0	22	16	0	0
	Gest 5	0	0	0	0	2	37	0	1	0	0
	Gest 6	0	4	0	0	0	0	32	4	0	0
	Gest 7	0	0	0	0	0	0	0	40	0	0
	Gest 8	0	15	4	0	0	0	0	0	0	21
	Gest 9	0	22	0	0	0	0	0	0	0	18

Tabelul 4.3. Matrice de confuzie în cazul testării bazei de date standard (etalon)

Se observă cum acuratețea clasificării a scăzut semnificativ. Toate imaginile de testare ce aparțin gesturilor 3, 4 și 8 au fost clasficate greșit. Problemele principale ce au dus la un asemenea rezultat pot fi: condiții de iluminare diferite, algoritm diferit de procesare a imaginii RGB sau distanța si poziția mâinii în chenar. Dimensiunea setului de date de antrenare reprezintă de asemenea o cauză, însă acest lucru îl vom demonstra în secțiunea următoare.

4.2.2 Testarea cu baza de date de testare creată personal

		Clasă prezisă									
		Gest 0	Gest 1	Gest 2	Gest 3	Gest 4	Gest 5	Gest 6	Gest 7	Gest 8	Gest 9
Clasă reală	Gest 0	15	0	0	0	0	0	0	0	0	0
	Gest 1	0	13	0	0	0	0	0	0	0	2
	Gest 2	0	0	15	0	0	0	0	0	0	0
	Gest 3	0	0	11	4	0	0	0	0	0	0
	Gest 4	0	0	0	15	0	0	0	0	0	0
	Gest 5	0	0	0	0	0	15	0	0	0	0
	Gest 6	0	0	0	0	0	0	14	0	0	1
	Gest 7	0	0	0	0	0	0	4	11	0	0
	Gest 8	0	0	9	0	0	0	0	0	6	0
	Gest 9	0	0	0	1	0	0	0	0	0	14

Tabelul 4.4. Matrice de confuzie în cazul testării bazei de date create personal

$$Precizie = 71.33\%$$

Precizia crește cum era de așteptat, dat fiind faptul ca baza de date de testare si antrenare au fost realizate în aceleași condiții de mediu. Totuși, aceasta este mai scăzută față de cazul antrenării – testării cu baza de date standard, deci dimensiunea mai mică a setului de date reprezentând un factor important pentru o clasificare corectă.

4.2.3 Testarea în timp real

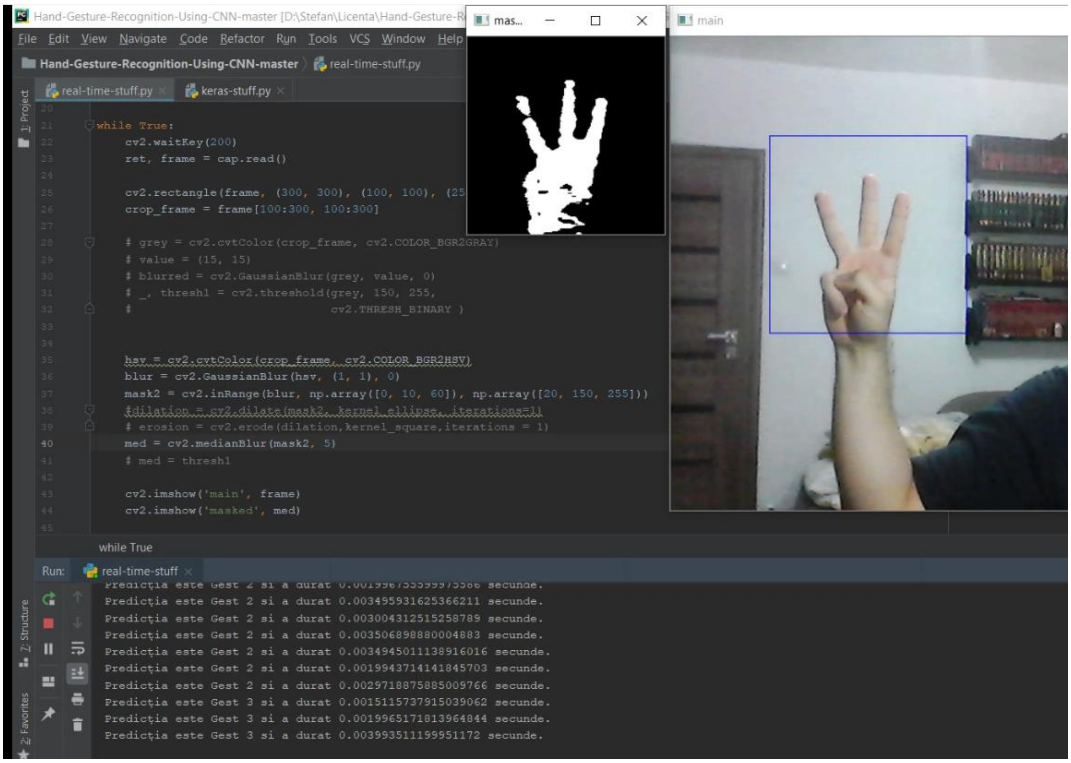

Figura 4.14. Realizare si predicție Gest 3

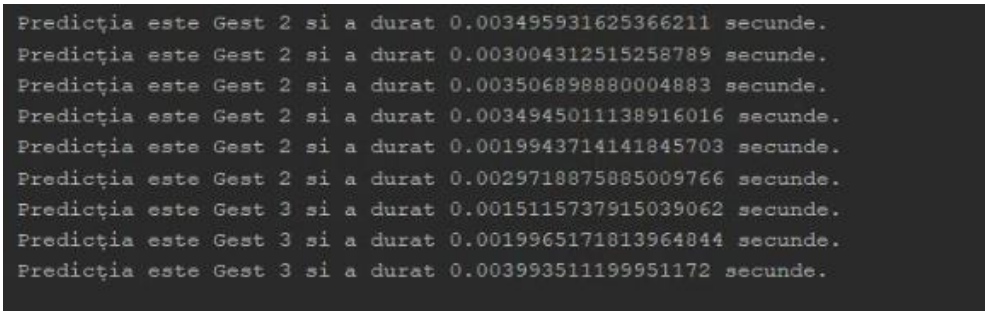

Figura 4.15. . Predicția și durata de predicție Gest 3

Putem observa în figurile de mai sus o faptul că sistemul confunda foarte ușor gestul 3 cu gestul 2. Deși în prima parte acest gest era recunoscut cu o precizie de 100%, se pare că baza de date realizată personal nu are un randament la fel de bun. Numărul de imagini mult mai mic a avut un impact considerabil asupra performanțelor sistemului.

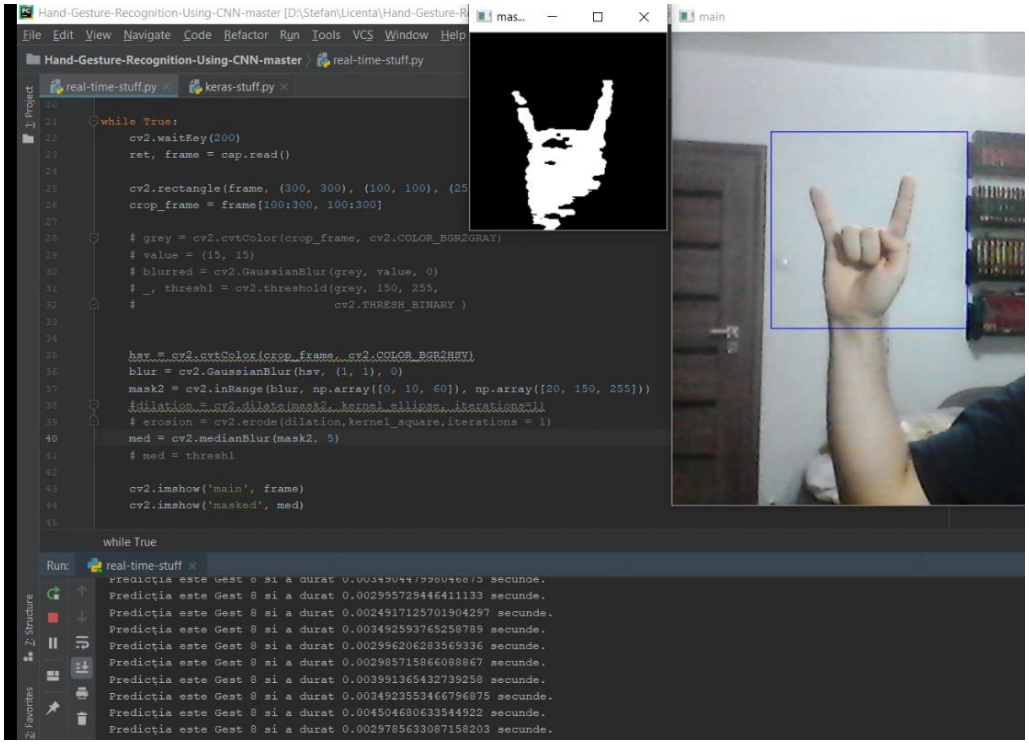

Figura 4.16. Realizare si predicție Gest 8

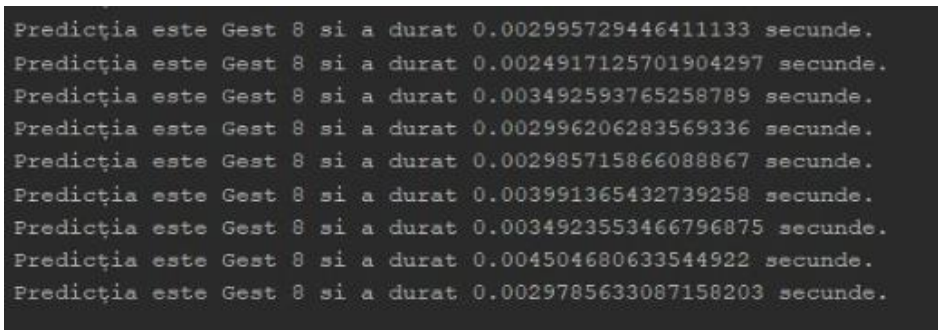

Figura 4.17. . Predicția și durata de predicție Gest 8

Dacă în cazul anterior am observat o scădere radicală a performanțelor, în cazul gestului 8 putem observa chiar contrarul. Acesta este detectat cu o precizie de 100%, nefiind confundat cu alte gesturi. În cazul bazei de date etalon, acest gest nu era recunoscut întotdeauna, fiind ușor confundat cu gesturile 1, 2 și 3.

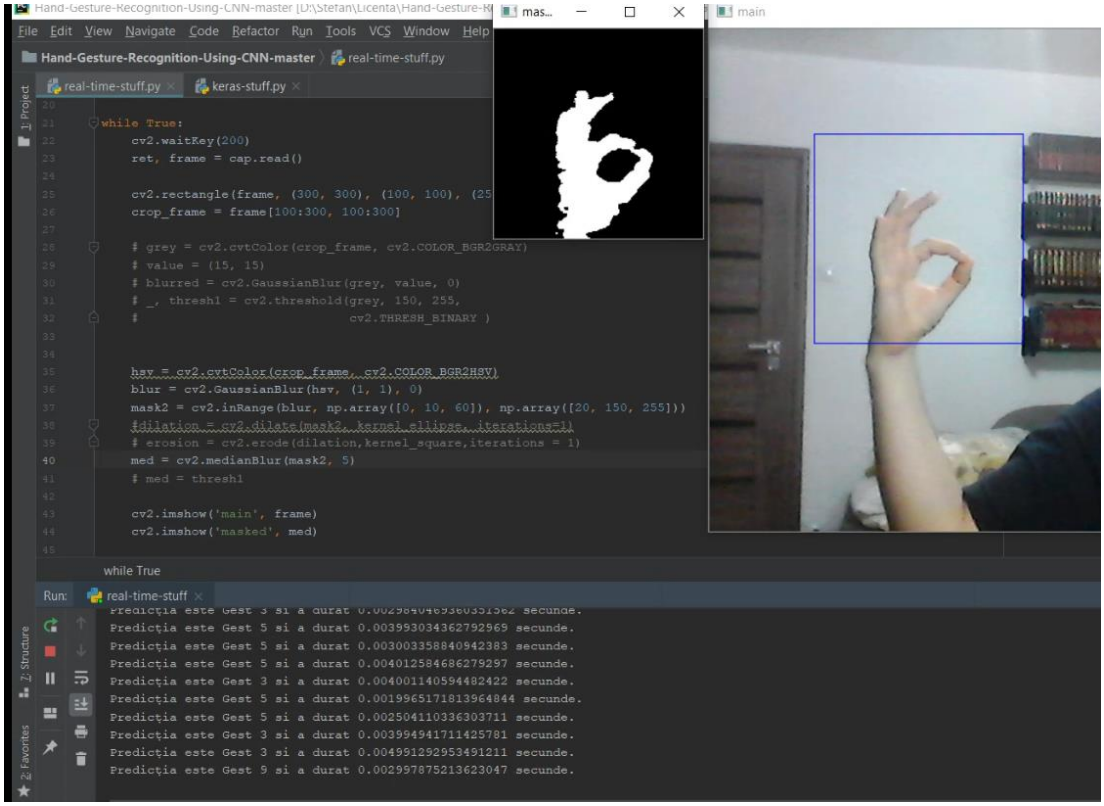

Figura 4.18. Realizare si predicție Gest 9

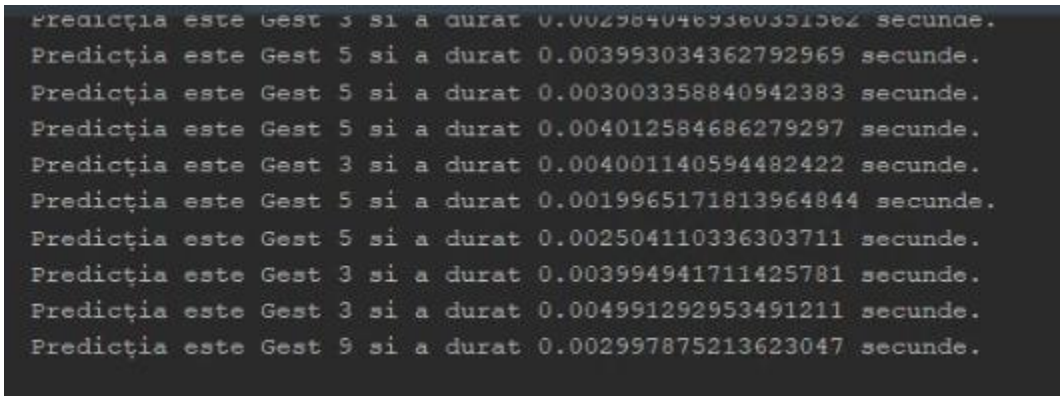

Figura 4.19. . Predicția și durata de predicție Gest 9

În cazul gestului 9 avem o scădere a performanței, acesta fiind recunoscut doar în 10% din cazuri. Este foarte ușor confundat cu gesturile 3 și 5. În cazul anterior acest gest era recunoscut în proporție de 80%, fiind confundat foarte rar cu gestul 3 sau 6.

CONCLUZII ŞI DEZVOLTĂRI ULTERIOARE

Concluzii generale

Obiectivul principal al proiectului de diplomă a constat în dezvoltarea unei aplicații adecvate pentru clasificarea și recunoaștere gesturilor mâinii pe baza imaginilor statice dar și întru-un flux video în timp real. Pentru realizarea acestui lucru s-au folosit 2 baze de date: baza de date standard (etalon) și baza de date creată personal ambele fiind antrenate de rețeau neurlă convoluțională creată.

Plecând de la premisa menționată în introducere, scopul principal al ansamblului descris este de a facilita comunicare cu persoanele cu dizabilități de vorbire, iar pentru a demonstra acest lucru am vrut să testez precizia sistemului în condiții diferite . Limitările actuale ale aplicaţiei se datorează rezoluției slabe a camerei web dar mai ales lista exhaustivă privind condițiile diferite ale mediului: variații diferite ale orientării mâinii, ocluzie, nivelul de lumină din cameră, variații între clase sau confuzie între fundal și regiunea de interes.

Rețeaua neurală convoluțională creată a interpretat cu succes ambele baze de date oferind rezultate buna din punct de vedere al predicției gesturilor în timp real cât si static. Rezultate por fi îmbunătățite semnificativ mărind baza de date de antrenare dar și optimizând algoritmii de prelucrare a imagini.

În concluzie, lucrarea de de diplomă a prezentat o serie de soluţii pentru a construi o aplicaţie capabilă să recunoască gesturi dinamice simple. Prin experimentele efectuate și rezultatele obţinute se arată că trăsăturile pot modela o varietate de gesturi, iar tehnicile de determinare a lor, permit o implementare eficientă, făcând posibilă înglobarea acestuia în sisteme mai mari.

Contribuţii personale

În cadrul acestei lucrări, am reușit să îndeplinesc obiectivele propuse aducând următoarele contribuții personale:

- Implementaea algoritmului de achiziție de imagini
- Implementarea unui algoritm de prelucrare a imaginilor pentru detectarea culorii pielii
- Crearea unei baze de date cu 1000 de imagini pentru antrenarea rețelei și 150 de imagini pentru testarea acesteia
- Implementarea unui model de rețea neurală convoluțională și testarea acesteia pentru clasficarea gesturilor
- Evaluarea acurateții și preciziei sistemului
- Analiza rezultatelor, depistarea posibilelor surse de eroare

Dezvoltări ulterioare

Având în vedere interesul și progresul imens în domeniul inteligeței artificiale și al învățării profunde, mi-am propus să conturez ideea de control al roboților cu ajutorul gesturilor mâinii. Pentru a spori gradul de utilizare a acestora cum ar fi în domenii de interes precum securitate, medicină sau construcții trebuie să se efectueze o detecție cu o precizie de peste 98% pentru a efectua sarcinile corect, unde fiecare gest trebuie să aibă un înțeles specific și să reprezinte o funcție diferită, de exemplu, "un deget ridicat" poate însemna „mișcare înainte", "cinci degete ridicate" poate însemna "oprire" și așa mai departe.

Tot în cadrul dezvoltărilor viitoare se va avea în vedere și crearea unei interfețe ce permite utilizatorilor experimentați sau neexperimenați să coopereze și să interacționeze cu un robot de asistență care operează într-un mediu domestic sau industrial.

BIBLIOGRAFIE

[1] CS231n: Convolutional Neural Networks for Visual Recognition, http://cs231n.stanford.edu/2018/, accesat la data:14..02.2019

[2] Weeks, A. R. (1996). Fundamentals of electronic image processing (pp. 316-414). Bellingham: SPIE Optical Engineering Press.

[3] Bishop, C. M. (1994). Neural networks and their applications. Review of Scientific

Instruments, 65(6), 1803–1832. doi:10.1063/1.1144830

[4] Ovidiu Grigore, Curs Rețele neuronal și sisteme fuzzy ”http://www.ai.pub.ro/content/RNSF.htm”, accesat la data: 12.03.2019

[5] Victor Neagoe, Curs Recunoașterea formelor de inteligență artificială, 2014-2015

[6] Weeks, A. R. (1996). Fundamentals of electronic image processing (pp. 316-414). Bellingham: SPIE Optical Engineering Press.

[7] Ovidiu Grigore, Laborator Rețele neuronale și sisteme fuzzy ”http://www.ai.pub.ro/content/RNSF.html”, accesat la data: 12.03.2019

[8] Yoshua Bengio, Nicolas Boulanger-Lewandowski, and Razvan Pascanu. Advances in Optimizing Recurrent Networks. 2012.

[9] [Open CV] “Color Conversion”, https://docs.opencv.org/3.1.0/de/d25/imgproc_color_conversions.html, accesat la data: 02.06.2019

[10]”Negative Gradient Algorithm, ” https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3, accesat la data: 12.03.2019

[11] 2. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)

[12] Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems. (2015) 91–99

[13] „Pocesarea Imaginilor Curs nr. 6 Prelucrari pe imagini multinivel (grayscale) (I) Proprietati statistice ale imaginilor grayscale si aplicatii”, http://users.utcluj.ro/~tmarita/IPL/IPCurs/C6.pdf, accesat la data: 28.05.2019

[14] G. X.Ritter, J.N. Wilson, Handbook of computer vision algorithms in image algebra - 2nd ed, 2001 CRC Press: Cap. 4.7. Threshold Selection by Maximizing Between-Class Variance.

[15] [l] P. J. Burt, “Fast filter transforms for image processing,” Compur. Graph. Image Processing, vol. 16, pp. 20-51, 1981.

[16] [Machine learning] Machine learning in computer vision, Fei-Fei Li, http://www.cs.princeton.edu/courses/archive/spr07/cos424/lectures/li-guest-lecture.pdf, accesat la data: 19.06.2019

[18] L. Pigou, S. Dieleman, P.-J. Kindermans, and B. Schrauwen. Sign language recognition using convolutional neural networks. In ECCVW, 2014.

[19]” Hand Gesture Recognition Database with CNN” ,

https://www.kaggle.com/benenharrington/hand-gesture-recognition-database-with-cnn, accesat la data: 21.06.2019

[20] Hand Gesture Recognition using CNN, https://github.com/raj-shah14/Hand-Gesture-Recognition-Using-CNN, 15.03.2019

[21] Francesco, Pier. 2018. Handy, hand detection with OpenCV. 4 Jun. , https://medium.com/@soffritti.pierfrancesco/handy-hands-detection-with-opencv-ac6e9fb3cec1,

accesat la data: 14.05.2019

[22] Haykin, Simon. November 28, 2008. Neural Networks and Learning Machines. Prentice Hall.

[23]”Gradient Descent Algorithm and it’s variants” https://imaddabbura.github.io/post/gradient_descent_algorithms/, accesat la data: 12.05.2019

[24] „Machine Learning”, https://sebastianraschka.com/faq/docs/diff-perceptron-adaline-neuralnet.html, accesat la data: 20.03.2019

Anexe

Anexa 1: Cod sursă pentru antrenarea și crearea rețelei și testarea bazelor de date

```
from keras.layers import *
from keras.models import *
from keras.optimizers import *
import cv2
import numpy as np
import os
from keras.utils import *
from keras.callbacks import *


def create_model_1():
    il = Input(shape=(50, 50, 1))

    hdn = Conv2D(filters=32, kernel_size=3, padding='same', strides=1, use_bias=True,
activation='relu')(il)
    hdn = MaxPooling2D(pool_size=2, strides=2)(hdn)


    hdn = Conv2D(filters=32, kernel_size=3, padding='same', strides=1, use_bias=True,
activation='relu')(hdn)
    hdn = MaxPooling2D(pool_size=2, strides=2)(hdn)


    hdn = Conv2D(filters=32, kernel_size=3, padding='same', strides=1, use_bias=True,
activation='relu')(hdn)
    hdn = MaxPooling2D(pool_size=2, strides=2)(hdn)


    hdn = Flatten()(hdn)

    hdn = Dense(512, activation='relu')(hdn)
    hdn=Dropout(0.25)(hdn)

    ol = Dense(10, activation='softmax')(hdn)

    opt = Adam(lr=1e-3)
    model = Model(inputs=il, outputs=ol)
    model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])

    return model


def train():
    model = create_model_1()

    x = []
    y = []

    for k in range(10):
```

```
        fldr = 'mydb/Gest %d' % k
        for file in os.listdir(fldr):
            im = np.expand_dims(cv2.imread('%s/%s' % (fldr, file), cv2.IMREAD_GRAYSCALE) /
255.0, axis=-1)
            x.append(im)
            y.append(k)

    x = np.asanyarray(x)
    y = to_categorical(np.asanyarray(y), 10)

    print(x.shape)
    print(y.shape)

    model.fit(x, y, epochs=50, batch_size=128, verbose=2,
              callbacks=[ModelCheckpoint(filepath='m.h5',    monitor='loss',    verbose=1,
save_best_only=True)])


def test():
    model = load_model('m.h5')
    cm = np.zeros((10, 10), dtype=np.int)
    fldr = 'mydbtest/test_'
    for k in range(10):
        for file in os.listdir('%s%d' % (fldr, k)):
        # file = 'mydb/Gest 9/8.jpg'
            im = cv2.imread('%s%d/%s' % (fldr, k, file), cv2.IMREAD_GRAYSCALE) / 255.0
            im = np.expand_dims(im, axis=-1)
            im = np.expand_dims(im, axis=0)
            #print(im.shape)
            pred = model.predict(im)
            ##print(pred.shape)
            pred = pred[0]
            pred = np.argmax(pred, axis=-1)
            cm[k, pred] += 1
    print("Matrice de confuzie:")
    print(cm)
    print("Precizie:")
    print(100 * np.sum(np.diag(cm)) / np.sum(cm))



def main():
    test()


if __name__ == '__main__':
    main()
```

Anexa 1: Cod sursă pentru prelucrarea imaginii și prezicerea în tim real

```
import cv2
import numpy as np
from keras.models import *
import time


model = load_model('m.h5')
print(model.summary())
import sys
#sys.exit(0)
cap = cv2.VideoCapture(0)

cap.set(cv2.CAP_PROP_FRAME_WIDTH, 700)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 400)

cnt = -30
kernel_square = np.ones((11,11),np.uint8)
kernel_ellipse= cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(5,5))

while True:
    cv2.waitKey(200)
    ret, frame = cap.read()

    cv2.rectangle(frame, (300, 300), (100, 100), (255, 0, 0), 0)
    crop_frame = frame[100:300, 100:300]

    # grey = cv2.cvtColor(crop_frame, cv2.COLOR_BGR2GRAY)
    # value = (15, 15)
    # blurred = cv2.GaussianBlur(grey, value, 0)
    # _, thresh1 = cv2.threshold(grey, 150, 255,
    #                            cv2.THRESH_BINARY + cv2.THRESH_OTSU)

    hsv = cv2.cvtColor(crop_frame, cv2.COLOR_BGR2HSV)
    blur = cv2.GaussianBlur(hsv, (1, 1), 0)
    mask2 = cv2.inRange(blur, np.array([0, 10, 60]), np.array([20, 150, 255]))
    #dilation = cv2.dilate(mask2, kernel_ellipse, iterations=1)
    #erosion = cv2.erode(dilation,kernel_square,iterations = 1)
    med = cv2.medianBlur(mask2, 5)
    # med = thresh1

    cv2.imshow('main', frame)
    cv2.imshow('masked', med)

    st = time.time()
    newimg = cv2.resize(med, (int(50), int(50)))

    ##if cnt >= 0:
      ##  newimg = cv2.resize(med, (int(50), int(50)))
       ## cv2.imwrite('mydbtest/%d.jpg' % cnt, newimg)
    ##cnt += 1

    im = np.expand_dims(np.expand_dims(cv2.resize(med, (50, 50)) / 255.0, axis=-1), axis=0)
    pred = np.argmax(model.predict(im)[0], axis=-1)
    dur = time.time() - st
```

```
print('Predicția este Gest %s si a durat %s secunde.' % (pred, dur))
```