Universitatea POLITEHNICA București
Facultatea Automatică şi Calculatoare
Departamentul Automatică şi Informatică Industrială

LUCRARE DE LICENŢĂ

Rețele neuronale pentru prelucrarea imaginilor faciale

Coordonator
Prof. Dr. Ing. Dorin Cârstoiu

Absolvent
Andrei-Daniel Ene

BUCUREȘTI
2018

Cuprins

CAPITOLUL I. Introducere

I.1. Introducere generală

Expresiile faciale sunt foarte importante pentru îmbunătățirea comunicării și interacțiunii umane. De asemenea, acestea pot fi utilizate pentru a studia comportamentul unei persoane, iar când acestea sunt abolite pot indica anumite afecțiuni medicale cum ar fi un accident vascular cerebral în urma căruia apare paralizia facială. Tehnicile de detectare a dispozițiilor bazate pe imaginile faciale pot oferi o abordare rapidă în determinarea unui comportament suspect al unei persoane.

Am ales această temă pentru lucrarea de licență deoarece expresia facială a partenerului de conversație îți poate oferi o informație a părerii lui despre subiectul discutat mult mai reală decât ceea ce el va exprima verbal.

Această lucrare mi-a stârnit interesul într-un mod deosebit și m-a determinat să studiez mai profund modul în care se pot determina anumite stări în funcție de anumite elemente ale feței, dar și importanța fiecărei distanțe în crearea unei rețele neuronale care va produce starea pe care o transmite o anumită persoană.

Dat fiind faptul că rețelele neuronale și inteligența artificială sunt unele din cele mai discutate subiecte în domeniul IT în ultimii ani, iar procesoarele neurale sunt din ce în ce mai populare și mai puternice, m-a încurajat să îmi doresc să înțeleg mecanismul de funcționare al lor.

Scopul acestui studiu a fost dezvoltarea unei aplicații inteligente pentru interpretarea imaginii faciale și încadrarea emoțiilor exprimate de o persoană într-una din cele șapte categorii folosind rețele neuronale.

Aplicația folosită împreună cu recunoașterea facială, poate deschide calea pentru realizarea de roboți umanoizi care să se exprime și vizual prin afișarea bucuriei, aprobării, dezgustului, etc. Aceste rețele neuronale pot fi de un mare ajutor în domenii din sfere diferite, ajutând la recunoașterea și interpretarea emoțiilor exprimate prin mimică.

Lucrarea presupune un studiu a mai multor cărți, articole și documente despre rețele neuronale, anatomia musculaturii faciale și despre emoțiile transmise de către o persoană prin intermediul mimicii. Am ales limbajul Python deoarece acesta este optimizat pentru analiza imaginilor și totodată am fost provocat să învăț să-l utilizez.

Lucrarea este structurată în șase capitole cuprinzând atât partea teoretică cât și partea aplicată. În următoarele capitole vor fi tratare aspecte teoretice legate de tehnologiile folosite în dezvoltarea aplicației, descrierea algoritmilor dar și modul în care este structurată baza de cunoștințe. În final voi prezenta rezultatele obținute, factori care pot influența sau îmbunătăți aceste rezultate dar și domenii în care aplicația poate fi utilizată și chiar îmbunătățită.

I.2. Prezentarea domeniului

Datorită informațiilor pe care le conțin, expresiile faciale joacă un rol important în interacțiunea omului cu diverse sisteme. Recunoașterea automată a expresiei feței se poate comporta ca un elmenet esențial în această interdependență om-mașină. O astfel de conexiune ar permite furnizarea automată a unor servicii bazate pe expresia interpretată în mod automat, ca de exemplu o negociere între o aplicație cu inteligență artificială și o persoană.

De asemenea, recunoașterea și interpretarea expresiilor faciale joacă un rol important și în domeniul securității, acolo unde folosind rețelele neuronale putem găsi anumite șabloane de comportament ciudate, care se datorează unei intenții de a face rău sau se pot descoperi eventuali teroriști.

Detectarea automată a expresiilor faciale a devenit un subiect foarte important în care se fac cercetări în ultimii ani. Acesta implică mai multe arii de studiu precum vederea artificială, machine learning și studiul comportamentului și poate fi folosit în diverse aplicații precum mașinile autonome care au nevoie de colectarea datelor despre persoane aflate în interiorul mașinii cât și în exteriorul ei, interacțiunea dintre om și mașină și nu în ultimul rând în securitate, acolo unde este folosit de multe ori pentru recunoașterea facială și a trăsăturilor specifice fiecărui individ. Îmbunătățiri semnificative s-au făcut deoarece s-a manifestat un interes deosebit pentru captura de imagini cu două dimensiuni dar și cu trei.

I.3. Descrierea problemei abordate

I.3.1. Din punct de vedere social

Problema abordată în această lucrare este încadrarea unei expresii faciale într-una din cele șapte categorii de expresii (neutru, furie, dezgust, frică, fericire, supărare și surprindere).

Analizând expresia facială a unei persoane și eventual o secvență de diferite expresii, pot fi demascate persoanele cu risc ridicat de a face rău în zonele aglomerate: aeroporturi, stații de tren sau autobuz , evenimente sportive sau culturale, unde se strâng mulți oameni.

De asemenea prin analiza expresiei faciale, sociologii pot evalua impactul pe care l-a avut un anumit discurs sau prezența unei persoane importante în mijlocul maselor, într-un timp relativ scurt, având în vedere faptul că un sondaj necesită un timp îndelungat, multe resurse consumate și disponibilitatea oamenilor de a participa la sondaj.

Analizarea expresiei faciale poate fi folosită și de liderii de campanii electorale în pregătirea unor discursuri care să împlinească așteptările și dorințele electoratului.

O altă aplicație poate fi folosită de către cadrele didactice în timpul testelor, tezelor, examenelor, atunci când pot fi depistați cei care doresc să înșele vigilența profesorilor.

I.3.2. Din punct de vedere medical

Victor Papilian relatează în lucrarea sa „Anatomia omului” [1], că musculatura facială este împărțită în musculatura mimicii și mușchii pieloși. Nervul facial inervează toți mușchii pieloși. Ei sunt distribuiți în jurul orificiilor feței având grade diferite de diferențiere. Funcția mușchilor pieloși este aceea de a executa acte fiziologice precum închiderea și deschiderea orificiilor, masticație și respirație.

Tot Victor Papilian susține că mimica este manifestarea stărilor psihice, în deosebi afective, cu ajutorul musculaturii faciale. Mișcările pantomimice sunt mișcările expresive ale membrelor și trunchiului, membrele superioare având un rol deosebit în gesticulație, pe când mimica este cel mai înalt grad de expresivitate a unui individ.

Acțiunile mușchilor pieloși, conform Victor Papilian, sunt producerea cutelor temporare, limbajul articulat mobilizând buzele, mobilizarea pleoapelor și a aripii nasului deformând șanțurile existente.

Șanțurile principale ale feței

- "Șanțul nazolabial este cel mai constant și caracteristic; este moștenit. Are formă de linie ușor ondulată, care pornește de lângă aripa nasului cu direcție oblică în jos și în afară spre comisura gurii.
- Șanțul mentolabial, cu direcție transversală, ușor curb, cu concavitatea în jos, separă bărbia de buza inferioară.
- Șantul jugal sau mentozigomatic coboară din regiunea zgomatică spre menton.
- Șanțul submental separă mentonul de eventuala "bărbie dubla".
- Șanțul palpebral superior separă porțiunea palpebrală de cea orbitară a pleoapei superioare.
- Șantul palpebral inferior are aceeași dispoziție la nivelul pleoapei inferioare.
- „Piciorul ciorii" este format din șanțuri radiare așezate în unghiul lateral al ochiului." [1, pp. 160,161]

Mușchii ce intervin în exemplificarea manifestărilor emoționale

- În timpul râsului intervin zigomaticul mare care trage comisurile gurii în sus și lateral, orbicularul care determină apropierea pleoapelor prin contracția sa. Șanțul nazolabial ia forma literei S.
- În timpul plânsului intervin coborâtorul unghiului gurii care trage comisurile gurii în jos, ridicătorii comun și propriu care ridică buza superioară, crăpătura bucală e cu concavitatea în jos, orbicularul care micșorează crăpătura palpebrală, sprâncenosul care determină coborârea sprâncenei.
- Atenția este determinată de mușchiul occipitofrontal care determină formarea cutelor transversale pe frunte.
- Gândirea se manifestă prin "micșorarea crăpăturilor palpebrale și coborârea sprâncenelor". [2]

- Disprețul se exteriorizează prin "coborârea colțului gurii și prin întinderea șanțului nazolabial sub formă aproape rectilinie (coborârea unghiului gurii)". [2]

Mușchii pleoapelor sunt enumerați de V. Papilian ca fiind următorii:

- orbicularul ochiului în principal, sprâncenos, coborâtor al sprâncenei, piramidal.
- Conform Jaroslov Kiss, mușchiul orbicular al ochiului care este format din trei porțiuni după cum urmează: palpebrală, orbitară și lacrimală. Porțiunea care declanșează închiderea ochiului este cea orbitară. "Laba de gâscă" apare atunci când la nivelul marginii externe a ochiului apar riduri provocate de contracția sa maximă. Închiderea forțată a ochiului este determinată de porțiunea tarsală.
- Mușchiul sprâncenos este mușchiul care exprimă atenția. Mușchiul piramidal acționează concomitent cu orbicularul ochiului. Fiind antagonistul frontalului determină cute transversale la rădăcina nasului exprimând amenințare sau durere. [3]

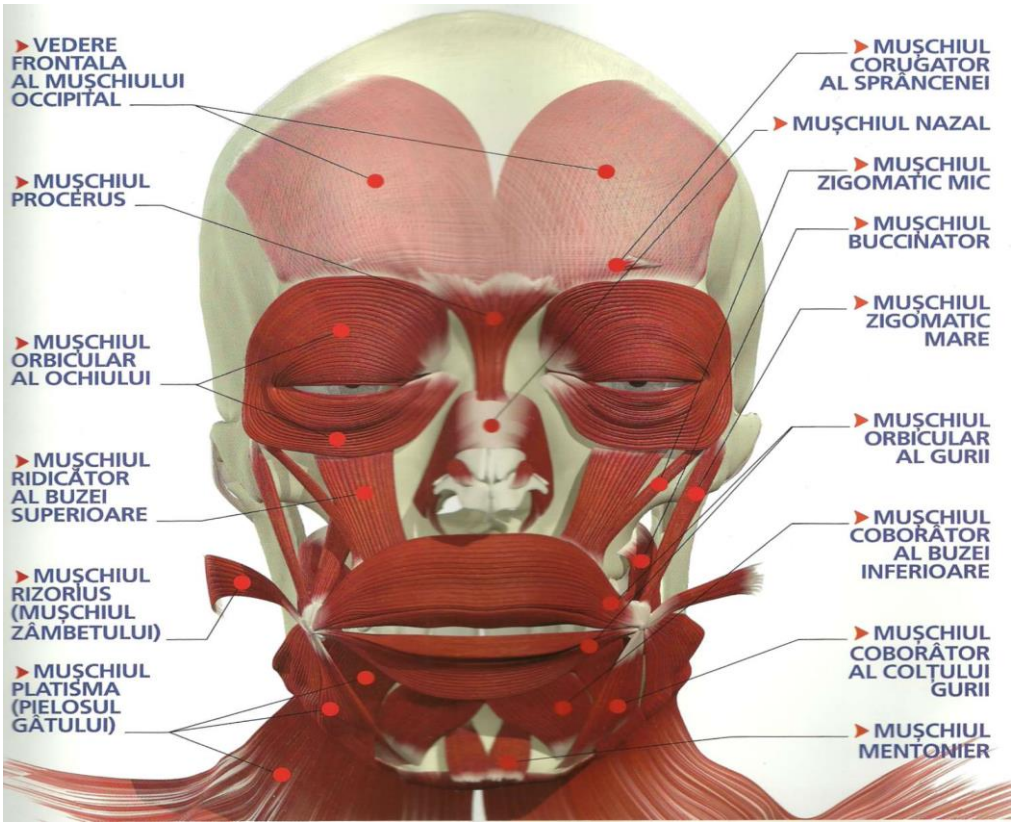
Nasul, conform lui V. Papilian [3] și J. Kiss [4], este compus din următorii mușchi:

- mușchiul nazal ce este constituit dintr-o porțiune transversă ce exprimă dezgustul și o porțiune alară ce determină dilatarea nărilor;
- mușchiul coborâtor al septului nazal care determină coborârea vârfului nasului și buza superioară.

Regiunea gurii, conform celor doi autori anterior menționați, este formată din următorii mușchi:

- orbicularul gurii ce determină închiderea gurii alcătuit dintr-o porțiune marginală și una labială,
- mușchiul buccinator fiind cel mai lat și mai profund mușchi determină comprimarea aerului când acesta se află în gură expulzându-l sub presiune,
- mușchiul ridicător al buzei superioare și al aripii nasului care ridică buza superioară și aripa nasului,

- mușchiul zigomatic mare care este mușchiul râsului trăgând comisura labială în sus și în exterior,
- mușchiul ridicător al unghiului gurii denumit mușchiul “canin” care trage comisura gurii în sus și ușor medial,
- mușchiul zigomatic mic care trage buza superioară în sus,
- mușchiul rizorius care este un mușchi slab, trage de comisura gurii în lateral,
- mușchiul coborâtor al unghiului gurii denumit și mușchiul triunghiular al buzelor sau “mușchiul suferinței” coboară comisura buzelor astfel șanțul nazolabial devine aproape rectiliniu,
- mușchiul coborâtor al buzei inferioare denumit și pătratul buzei inferioare trage buza inferioară în jos determinând disprețul,
- mușchiul mentonier ce determină împingerea înainte a buzei inferioare
- mușchiul transversal al mentonului ce coboară unghiul gurii [3], [4].



Imaginea 1 Musculatura facială

I.3.3. Din punct de vedere software

Urmărirea feței

Multe dintre sistemele de recunoaștere a feței au ca și intrare o secvență video. De multe ori, aceste sisteme sunt nevoite să urmărească o anumită față, nu doar să o detecteze. Urmărirea feței este esențială în problemele de estimare a mișcării. Urmărirea feței poate fi efectuată folosind mai multe metode, de exemplu urmărirea capului, urmărirea caracteristicilor specifice ale feței, metode bazate pe urmărirea unei anumite imagini sau a unui model. Aceste metode sunt diferite moduri în care se pot clasa următorii algoritmi [5]:

- Urmărirea capului/Urmărirea trăsăturilor individuale. Capul poate fi urmărit ca și o entitate în întregime sau pot fi urmărite anumite caracteristici specifice.
- 2D/3D. Sistemele în două dimensiuni urmăresc o față într-o imagine și returnează două coordonate care reprezintă locația în fotografie/video a feței. Pe de altă parte, sistemele în trei dimensiuni pot crea un model 3D a feței. Această abordare ne poate permite să estimăm poziția la care se află subiectul dar și variația orientării lui.

Procesele fundamentale de urmărire a feței tind să localizeze o imagine într-o altă imagine. Apoi, acestea caută diferențele între două cadre consecutive și actualizează locația feței. Iluminarea incorectă, deformarea facială, astuparea de alte obiecte a anumitor trăsături ale feței sunt unele dintre problemele cu care aceste sisteme se confruntă.

Un exemplu de algoritm pentru urmărirea feței poate fi cel propus de Baek în lucrarea [6]. Vectorul de stări al feței include poziția centrului feței, mărimile dreptunghiului care va fi folosit pentru a încadra fața și culoarea medie a feței. Noile fețe candidate vor fi evaluate de estimatorul Kalman. În modul de urmărire a feței, dacă aceasta nu este nouă, fața preluată din cadrul anterior va fi folosită ca și șablon.

Când se recomandă utilizarea structurilor bazate pe cunoștințe?

Structurile bazate pe cunoștințe sunt folosite atunci când nu se poate determina o soluție algoritmică. Acesta este destul de greu de obținut deoarece datele de intrare pot varia într-un anumit interval și nu se poate stabili concret o legătură între datele de intrare și cele de ieșire.

Acestea sunt recomandate să fie utilizate atunci când există suficiente date acumulate în timp pe baza cărora s-a învățat un anumit șablon pentru a prezice clasa de apartenență pentru un set de date de intrare.

CAPITOLUL II. Tehnologii folosite

II.1. Limbajul Python

Imaginea 2 Logo Python

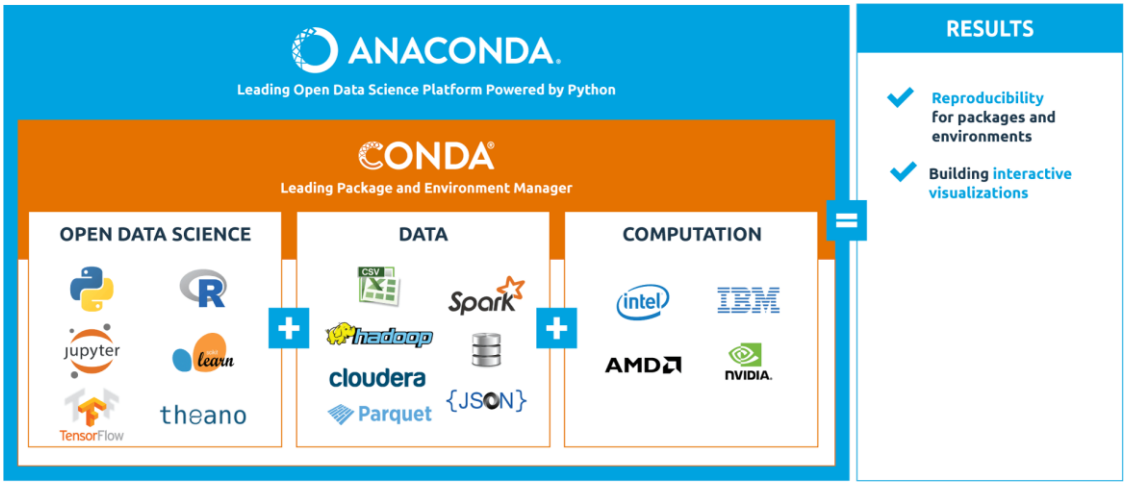
Python [Imaginea 2] este un limbaj de programare interpretat, interactiv și orientat pe obiecte. Acesta include module, excepții, tipuri de date foarte ridicate și dinamice și clase. Python combină o putere remarcabilă și complexă cu o sintaxă foarte clară și simplă. Este de asemenea folosit ca o extensie de limbaj pentru aplcațiile care folosesc interfețe programabile.

Python este un limbaj portabil, acesta putând rula atât pe toate sistemele de operare precum distribuții de Unix, Mac cât și Windows. De asemenea, acesta este folosit și pentru a scrie jocuri, a dezvolta aplicații web, a rezolva probleme de business și pentru a dezvolta unelte interne pentru a facilita productivitatea companiilor. [7]

Am ales acest limbaj deoarece se pliază foarte bine pentru dezvoltarea rețelelor neuronale, acesta dispune de o colecție mare de module pentru interpretarea imaginilor, iar codul este intuitiv de scris. Un alt motiv este sintaxa limbajului care ajută la scrierea unui cod foarte simplu, curat și ușor de înțeles pentru alți programatori deoarece se aseamănă cu limbajul uman.

II.2. Distribuția Anaconda

Distribuția Anaconda conține pachete gratuite pentru a rula cod scris în limbajul Python sau R, toate acestea fiind compilate într-un mediu securizat oferind astfel siguranță și optimizare în rularea algoritmilor de Machine Learning. Aceste pachete pot fi observate și în Imaginea 3.



Imaginea 3 Diagrama Anaconda

Anaconda Enterprise este software-ul gratuit oferit de cei de la Anaconda pentru organizații, pentru a avea siguranța și scalabilitatea datelor introduse în aplicațiile Python și R, dar și pentru machine learning. Folosind acest software, se pot crea modele specializate pentru machine learning care pot folosi clustere specializate în stocarea datelor precum Docker/Kubernetes, Hadoop și Spark.

Girish Modgil, directorul pe partea de date și analiză la General Electric afirma: „În procesarea datelor la General Electric, totul este despre colaborarea în timp real. Echipele noastre se află peste tot pe glob, iar noi ne dorim să colaborăm în timp real. Vrem să avem o palatformă comună pentru machine learning pentru ca toată lumea să poată utiliza o analiză modernă a datelor și să seteze standarde pentru ca oricine să o poată folosi. Anaconda Enterprise ne permite să facem acest lucru într-un mod optim.” [8]

De asemenea, distribuția conține peste 1400 de module care sunt gratuite pentru oricine dorește să își dezvolte o aplicație nouă. Combinat cu mediul virtual pus la dispoziție, aceasta poate rula același cod, cu aceeași reproducere atât pe sistemul de operare Linux, cât și pe Windows și MacOS.

II.2.1. Module folosite

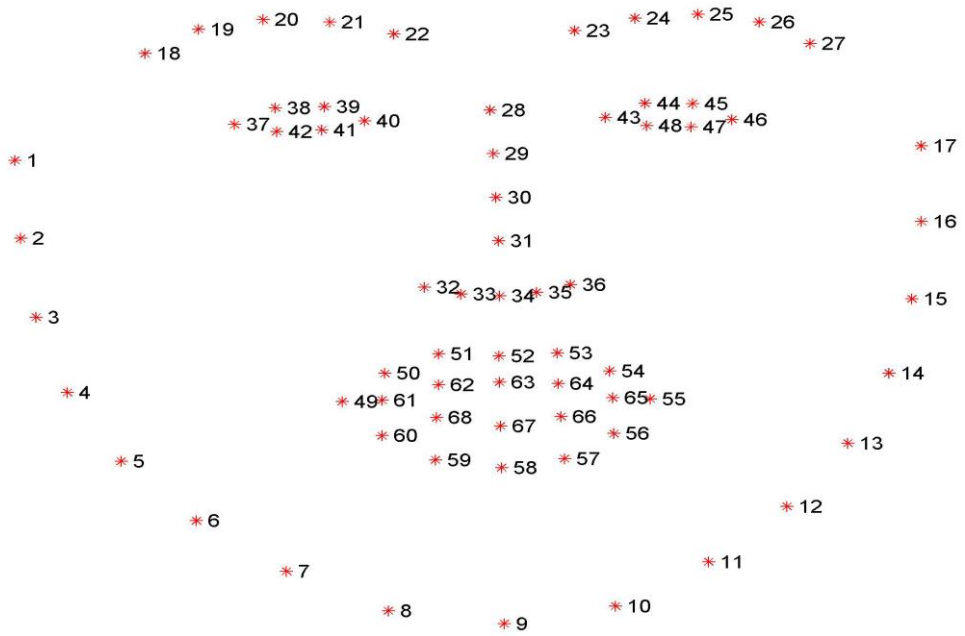
Limbajul Python conține diverse module pentru a ajuta programatorul să folosească funcții deja implementate, fără a rescrie toată logica, iar avantajul acestui limbaj este faptul că este open-source, însemnând că fiecare programator își poate crea propriul său modul pe care ulterior să îl publice astfel încât oricine are nevoie de o anumită funcție implementată de el să o poată folosi.

II.2.1.1. Dlib

Dlib este unul din modulele pe care le-am folosit pentru realizarea algoritmului de recunoaștere a expresiei faciale deoarece acesta conține metoda *„get_frontal_face_detector()”* care returnează un obiect de tip detector pe care l-am folosit pentru a identifica fețele prezente într-o anumită fotografie sau un anumit cadru pentru analiza video.

shape_predictor_68_face_landmarks

În modulul dlib am setat predictorul personalizat shape_predictor_68_face_landmarks pentru a întoarce coordonatele a 68 de puncte ale fiecărei fețe din imagine. Distribuția celor 68 de puncte este prezentată în Imaginea 4 .



Imaginea 4 Poziționarea celor 68 de puncte de pe față din baza iBUG 300-W

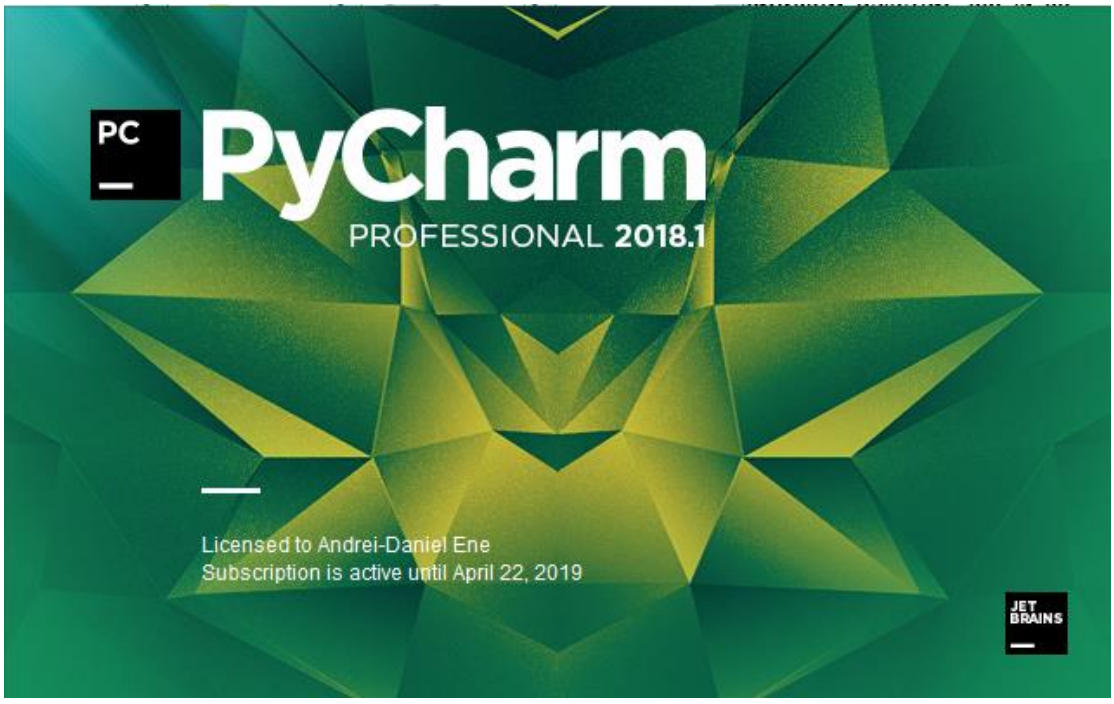
II.2.1.2. OpenCV

În anul 1999, Gary Bradsky a început dezvoltarea modulului OpenCV pentru Intel, prima sa versiune fiind lansată abia în anul 2000. În anul 2005, vehiculul care a câștigat DARPA Grand Challenge la Stanley a folosit acest modul. Acum, OpenCV este folosit în mulți algoritmi, în special în cei bazați pe vederea artificială și machine learning și se extinde în fiecare zi datorită faptului că orice programator poate aduce îmbunătățiri, codul fiind open-source. [9]

OpenCV suportă o multitudine de limbaje de programare precum C++, Python, Java, etc. și este disponibil pe majoritatea platformelor, printre care regăsim Windows, Mac OS, Android și Unix. [9]

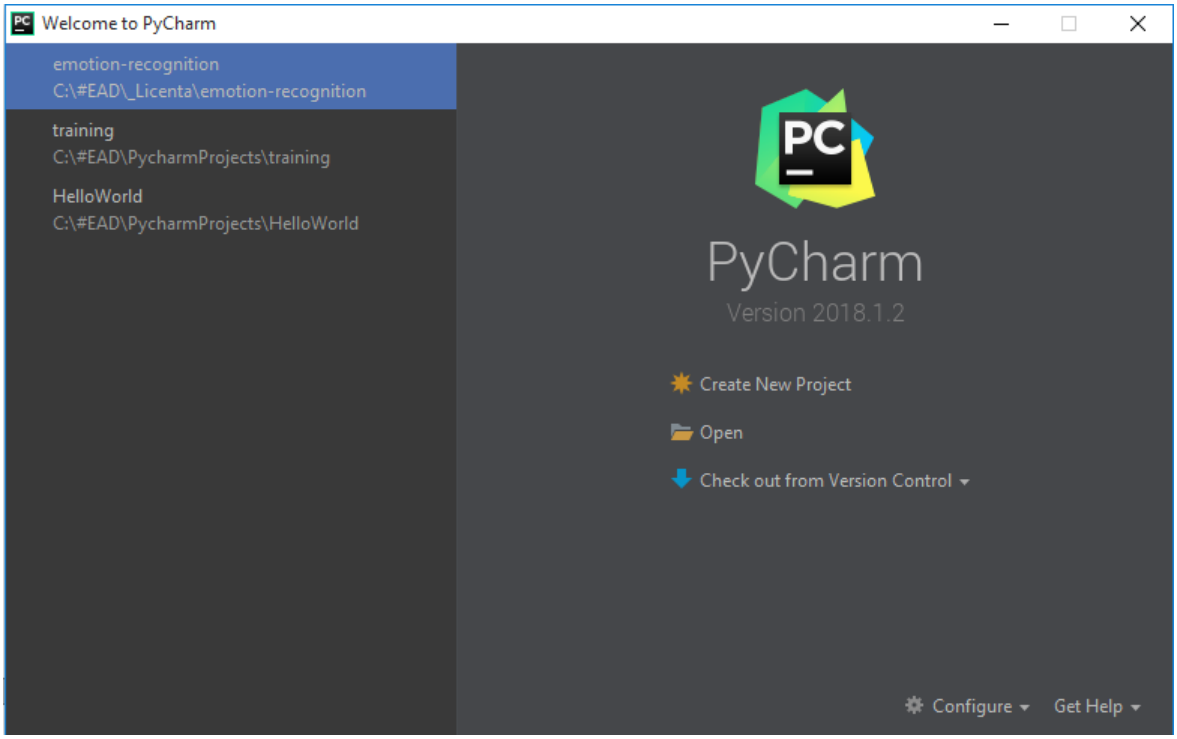
II.3. PyCharm IDE

În această lucrare voi folosi editorul PyCharm oferit de cei de la JetBrains pentru a scrie codul sursă.PyCharm este folosit atât ca platformă pentru aplicații desktop, cât și ca mediu de dezvoltare integrat (IDE) și prezintă o gamă largă de limbaje în care poate fi utilizat. PyCharm poate rula pe diferite sisteme de operare precum Windows, Linux și MacOS. În Imaginea 5 se poate observa pagina de încărcare a editorului.



Imaginea 5 Pagina de start

De asemenea, PyCharm este foarte util pentru a accesa rapid proiectele anterioare oferind la lansarea în execuție a programului lista cu ultimele proiecte accesate [Imaginea 6].



Imaginea 6 Proiecte recente – PyCharm

PyCharm 2018.1.2 este cea mai nouă versiune la momentul actual, fiind un mediu de dezvoltare sau un instrument pentru programatori pentru scrierea, compilarea, testarea, depanarea, proiectarea și instalarea programelor.

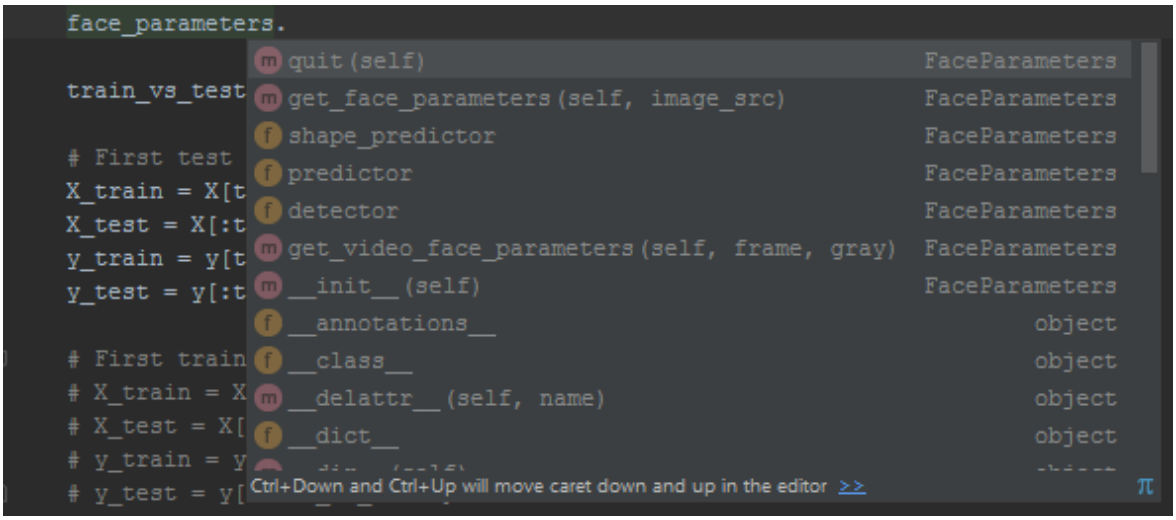
De asemenea, acesta este oferit în doua variante, atât cea premium cât și cea community. In Imaginea 7 sunt reprezentate diferențele dintre cele două.

CHOOSE EDITION

	PyCharm Professional Edition	PyCharm Community Edition
Intelligent Python editor	✓	✓
Graphical debugger and test runner	✓	✓
Navigation and Refactorings	✓	✓
Code inspections	✓	✓
VCS support	✓	✓
Scientific tools	✓	
Web development	✓	
Python web frameworks	✓	
Python Profiler	✓	
Remote development capabilities	✓	
Database & SQL support	✓	

Imaginea 7 Ediții PyCharm

Un alt avantaj al folosirii acestui editor de text, este faptul că oferă sugestii [Imaginea 8] pentru a scrie codul cât mai rapid și facilitează astfel productivitatea programatorului.



Imaginea 8 Sugestii de completarea a codului

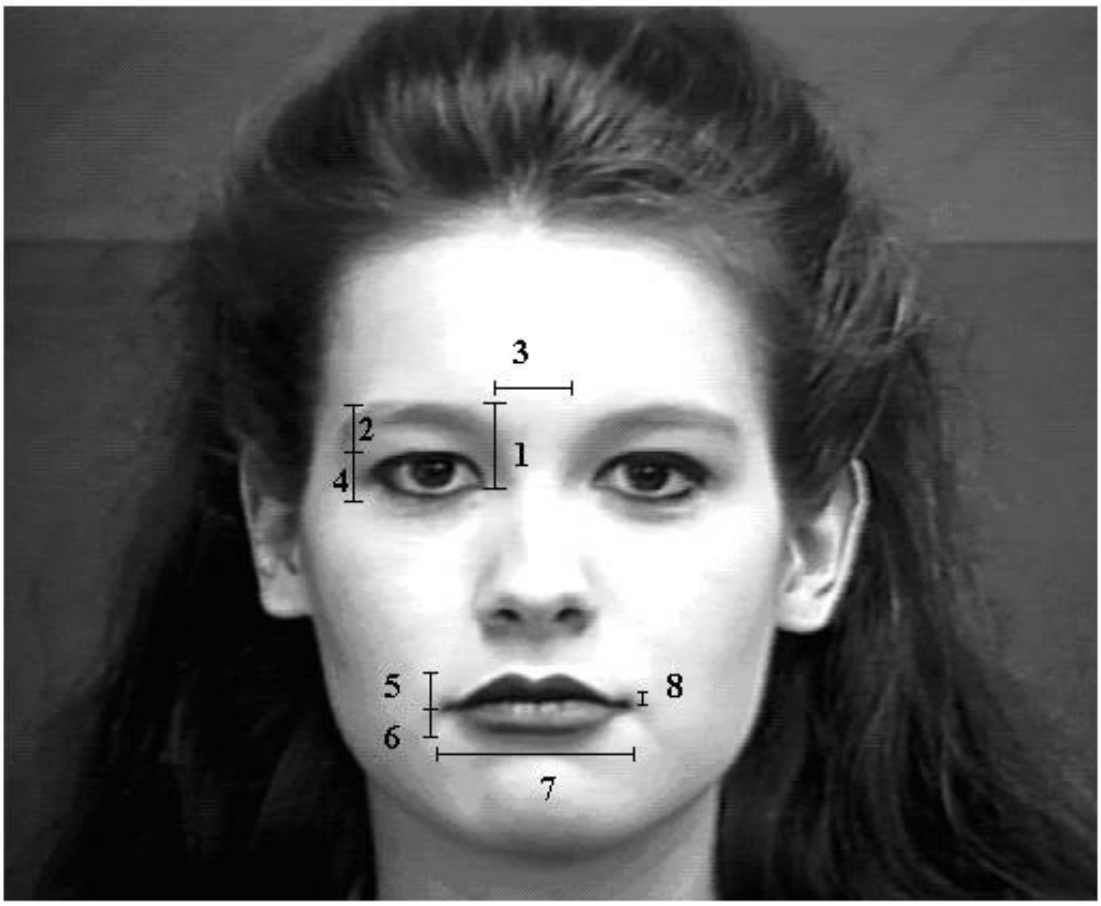
CAPITOLUL III. Arhitectura pentru rezolvare

III.1. Descrierea metodei de rezolvare propuse

În urma studiului acestei probleme, am găsit că orice expresie din cele șapte menționate anterior poate fi determinată prin extragerea a numai 15 parametri prezenți într-o expresie facială. Acești 15 parametri sunt împărțiți în două categorii și anume 8 dintre ei sunt distanțe între două puncte ale feței măsurate în pixeli și 7 parametri care au valorile doar adevărat/fals.

Cei 8 parametri care reprezintă valori exacte sunt:

1. **Distanța la care sunt ridicate sprâncenele** – această distanță este măsurată de la pleoapa inferioară a ochiului până la punctul cel mai înalt al sprâncenei
2. **Distanța pleoapei superioare față de sprânceană** – această distanță este măsurată de la pleoapa superioară până la punctul din mijlocul sprâncenei
3. **Distanța dintre sprâncene** – această distanță este măsurată între punctele de extrem ale sprâncenelor dinspre nas.
4. **Distanța de deschidere a ochiului** – această distanță este măsurată între punctele din mijlocul pleoapelor.
5. **Grosimea buzei superioare** – aceasta reprezintă distanța de la punctul care delimitează cele două buze până la punctul cel mai înalt al buzei superioare.
6. **Grosimea buzei inferioare** – aceasta reprezintă distanța de la punctul care delimitează cele două buze până la punctul cel mai de jos al buzei inferioare.
7. **Lungimea gurii** – aceasta reprezintă distanța dintre primul si ultimul punct care reprezintă locația gurii în frame
8. **Deschiderea gurii** – aceasta reprezintă distanța dintre suprafața inferioară a buzei superioare și suprafața superioară a buzei inferioare.



Imaginea 9 Valorile reale măsurate pe față pentru o expresie neutră.

Pe langă acești parametri, sunt necesari și alții 7 reprezentând valori de adevăr:

1. **Vizibilitatea dinților superiori** – Prezența sau absența vizibilă a dinților superiori.
2. **Vizibilitatea dinților inferiori** – Prezența sau absența vizibilă a dinților inferiori.
3. **Liniile frunții** – Prezența sau absența ridurilor în partea superioară a frunții
4. **Liniile sprâncenelor** – Prezența sau absența ridurilor în regiunea din zona sprâncenelor
5. **Liniile nasului** – Prezența sau absența ridurilor în zona de sub sprâncene și deasupra nasului.
6. **Liniile bărbiei** – Prezența sau absența cutelor în zona bărbiei
7. **Liniile pomeților** – Prezența sau absența cutelor în zona dintre pomeți și nas

Imaginea 10 Parametrii binari pentru crearea algoritmului

Cele 8 distanțe sunt măsurate în pixelii dintre două puncte specifice. În cazul distanțelor care sunt simetrice pe ambele părți ale feței s-a facut o medie a lor. Pentru a avea în bază valori similare pentru aceeași distanță, indiferent cât de aproape se află persoana de camera de fotografiat/filmat, s-au dublat numărul de pixeli măsurați și s-a împărțit la numarul de pixeli dintre cei doi ochi ai feței.

$$ValoareFinala = \frac{2 \cdot ValoareaMasurata}{DistantaDintreOchi}$$

Punctele specifice pentru a calcula distanțele au fost determinate folosind biblioteca **„dlib”** instalată alături de distribuția Anaconda de Python, ambele descrise în secțiunea II.1.

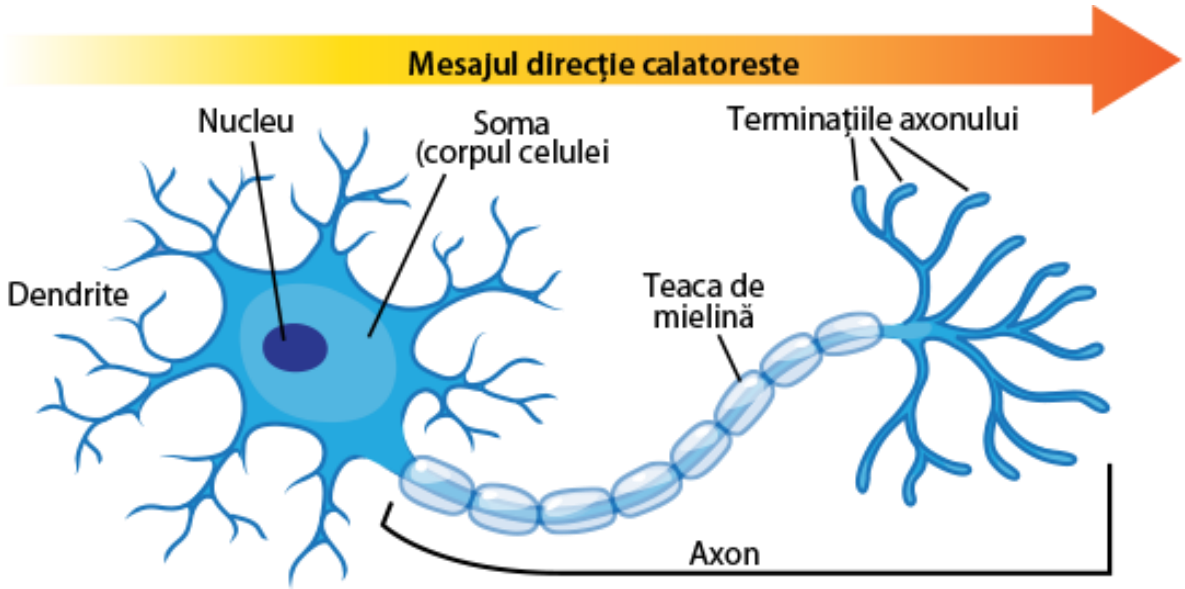
III.2. Rețeaua neuronală

Abilitatea de a învăța anumite clasificări este fundamentală pentru comportamentul inteligent. În ultimele două decenii, cercetătorii au dezvoltat un număr important de algoritmi pentru conceptul de achiziție automată a datelor.

O rețea neuronală este un suport hardware pentru un calcul neural ce poate fi folosit la rezolvarea problemelor de asociere deoarece se bazează pe o învățare prin a analiza suficient de multe exemple și pe extragerea unui model pe bază de exemple. Alte denumiri alternative pentru o rețea neuronală sunt neurocalcul, conexionism, sisteme adaptive, procesare paralelă distribuită și rețele cu auto-organizare.

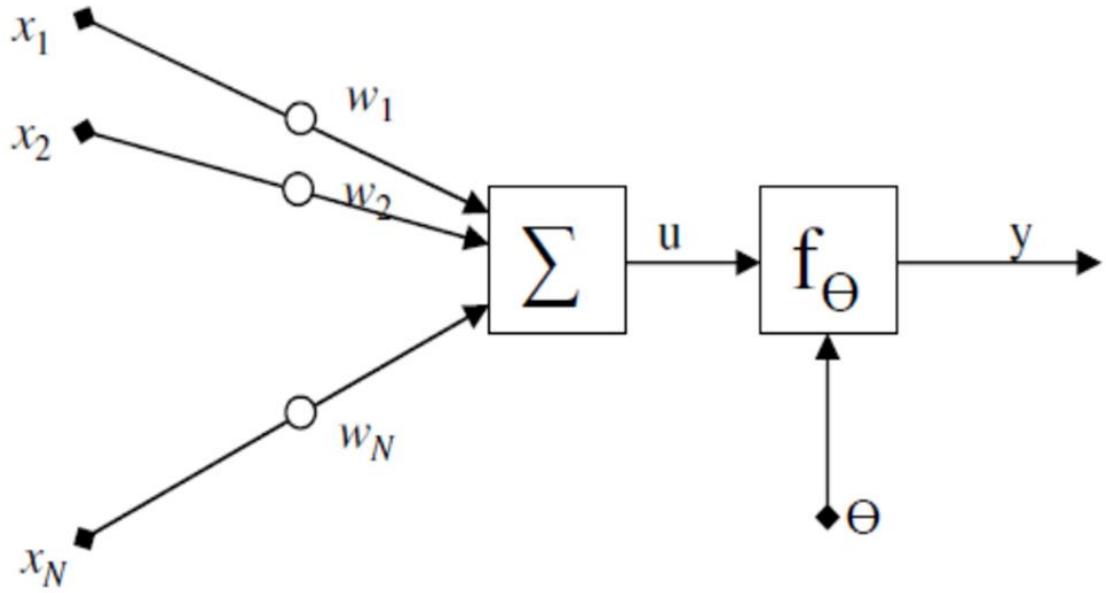
O definiție a unei Rețele neuronale artificiale este dată de către S. Haykin „Rețeaua neuronală artificială este un calculator distribuit, masiv paralel, care achiziționează noi cunoștințe pe baza experienței anterioare și le face disponibile pentru utilizarea ulterioară” [10].

Neuronul natural este prezentat în Imaginea 11 și reprezintă unitatea morfo-funcțională a sistemului nervos.

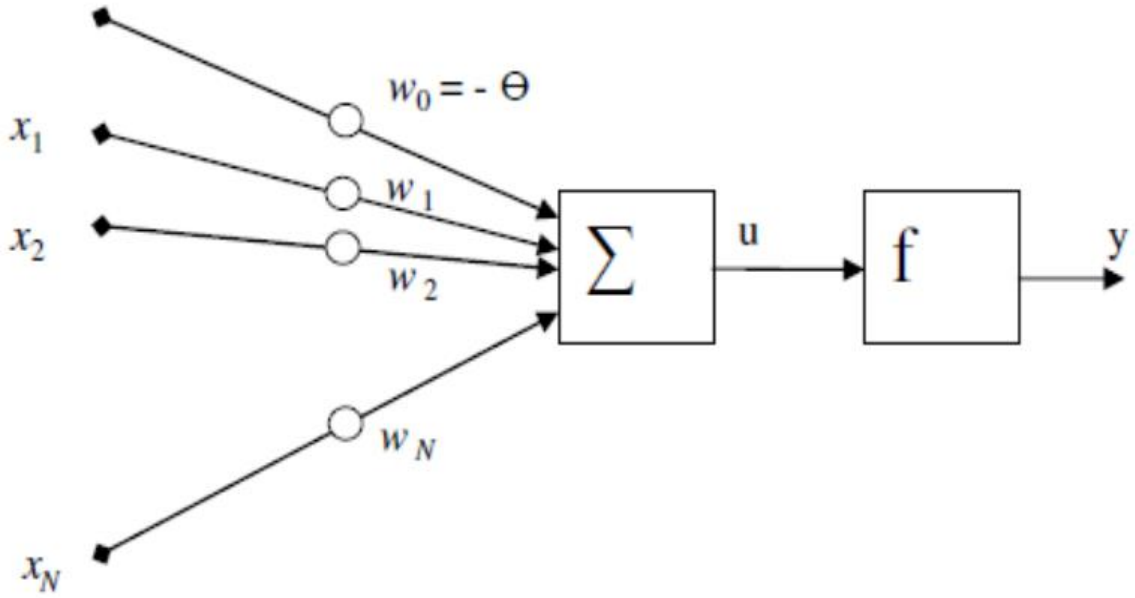


Imaginea 11 Anatomia neuronului

Neuronul artificial reprezintă o digitalizare a neuronului natural și încearcă să copieze comportamentul lui. Acesta este reprezentat în două modele: Modelul de bază McCullouch-Pitts [Imaginea 12] și Modelul derivat [Imaginea 13].



Imaginea 12 Model de bază McCullouch-Pitts (1943)



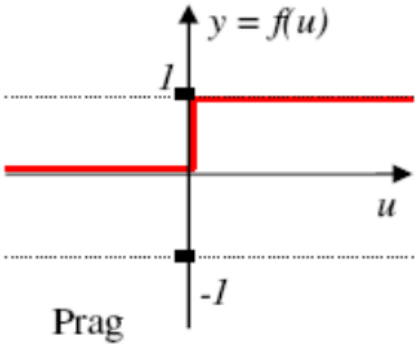
Imaginea 13 Model derivat

w – ponderi sinaptice

f – funcție de integrare/agregare/activare

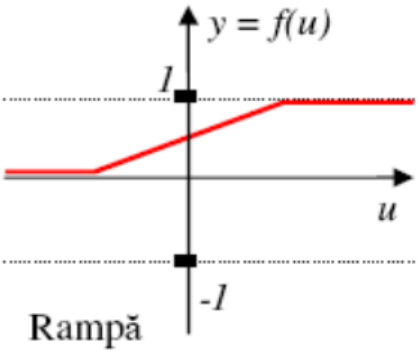
Θ – bias (polarizare)

Funcția de activare poate fi de patru tipuri: prag [Imaginea 14], rampă [Imaginea 15], sigmoidală [Imaginea 16] și liniară [Imaginea 17].



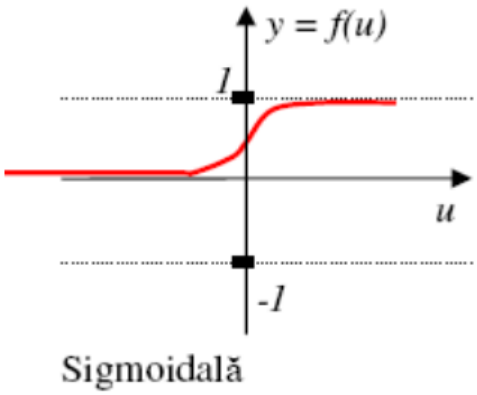
Imaginea 14 Funcția de activare - Prag

$$f(u) = \begin{Bmatrix} 0, u \leq 0 \\ 1, u > 0 \end{Bmatrix}$$



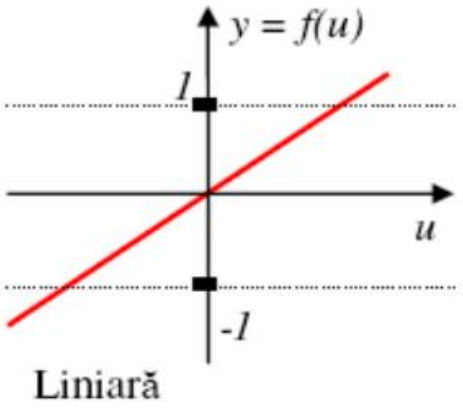
Imaginea 15 Funcția de activare – Rampă

$$f(u) = \begin{Bmatrix} 0, u \leq 0 \\ u/k, 0 < u < k \\ 1, k < u \end{Bmatrix}$$



Imaginea 16 Funcția de activare – Sigmoidală

$$f(u) = \frac{1}{1 + e^{-ku}}$$



Imaginea 17 Funcția de activare – Liniară

$$f(u) = ku$$

Deoarece rețelele neuronale sunt inspirate din trăsăturile creierului uman, acestea au capacitatea de a învăța din diferite exemple, capacitatea de a generaliza o anumită situație și să ia decizii corecte pe baza experienței acumulate și capacitatea de a sintetiza intrările și a elimia zgomotele care pot influența ieșirea. [11]

Procedura folosită pentru a realiza procesul de învățare se numește *Algoritm de învățare* și reprezintă funcția programului de a modifica și actualiza ponderea pe care un anumit nod o are în vederea obținerii rezultatului final [12], printr-o stimulare din partea unui expert sau a mediului, sau nesupervizat, prin analiza statistică a vectorilor de intrare.

Algoritmul de învățare/de instruire este un proces iterativ bazat pe o anumită regulă:

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n)$$

unde: *k* – neuron de la care „pleacă” ponderea;

j – neuron spre care „vine” ponderea;

n – momentul de timp;

$\Delta w_{kj}(n)$ – algoritmul de instruire.

Modificarea acestei ponderi reprezintă metoda tradițională pentru modelul de realizare a unei rețele neuronale. O astfel de abordare este apropiată de un sistem liniar adaptiv care este deja stabilit și aplicat cu succes în diverse domenii [13].

Pașii realizării unui astfel de algoritm sunt:

1. se inițializează ponderile
2. se calcululează neuronii de ieșire
3. se calculează cantitatea Δw cu care se modifică fiecare pondere
4. se actualizează ponderile: $w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n)$
5. se realizează recursiv pașii 2-4 până eroarea ajunge la un anumit coeficient stabilit la început.

Din punctul de vedere a realizării instruirii, rețelele neuronale pot fi împărțite în următoarele categorii:

- **Supervizate**: setul de instruire se realizează pe baza unor perechi intrare-ieșire;
- **Cu întărire**: se realizează online, folosind de asemenea și un semnal de întărire extern
- **Nesupervizate**: rețeaua poate fi atât online cât și offline, este cea mai rapidă deoarece este o combinație între primele două categorii și se bazează pe clasificarea statistică a intrărilor folosind o distanță euclidiană.

Această instruire a unei rețele neuronale se poate realiza prin minimizarea erorii sau pe baza gradientului.

- Instruire prin minimizarea erorii:

$$E(n) = \frac{1}{M}\sum_{k=1}^{M} e_k^2(n)$$

unde:

n – momentul de timp;

M – numărul de neuroni pe stratul de ieșire;

k – indicele neuronului de pe strat;

$e_k(n)$ – eroarea neuronului *k* la momentul de timp *n*.

Pentru neuronul *k*, eroarea de la momentul *n* se calculează folosind formula:

$$e_k(n) = d_k(n) - y_k(n)$$

unde: $d_k(n)$ este ieșirea dorită pentru neuronul *k* la momentul de timp *n*.

$y_k(n)$ este ieșirea neuronului *k* la momentul *n* calculată de către rețeaua neuronală

- Instruire bazată pe gradient:

$$\Delta w_k(n) = -\mu \nabla E(n) = -\mu \frac{\partial E(n)}{\partial w_k(n)}$$

unde: μ este pasul de instruire, subunitar;

Regula Widrow-Hoff sau regula Delta:

$$\Delta w_k(n) = -\mu \cdot x_k \cdot e_k$$

Așa cum neuronii din creierul uman pot muri și astfel se crează conexiuni noi, tot așa și pentru o rețea neuronală este posibil să își modifice propria topologie determinată pe baza seturilor precedente de antrenament.

Este evident că o rețea neuronală își dobândește puterea de calcul datorită structurii distribuită paralel și abilitatea de generalizare. Generalizarea reprezintă abilitatea unei rețele neuronale de a oferi un răspuns rezonabil pentru un set de date de intrare care nu s-au regăsit în setul de antrenament.

O rețea neuronală este caracterizată de următoarele proprietăți și capabilități:

- Neliniaritate
- Mapare intrare-ieșire
- Adaptabilitate
- Răspuns rațional
- Informație contextuală
- Toleranță la defecte
- Implementabilitate la scală mare

- Uniformitatea design-ului și analizei
- Analogie neurobiologică

III.3. Baza de date

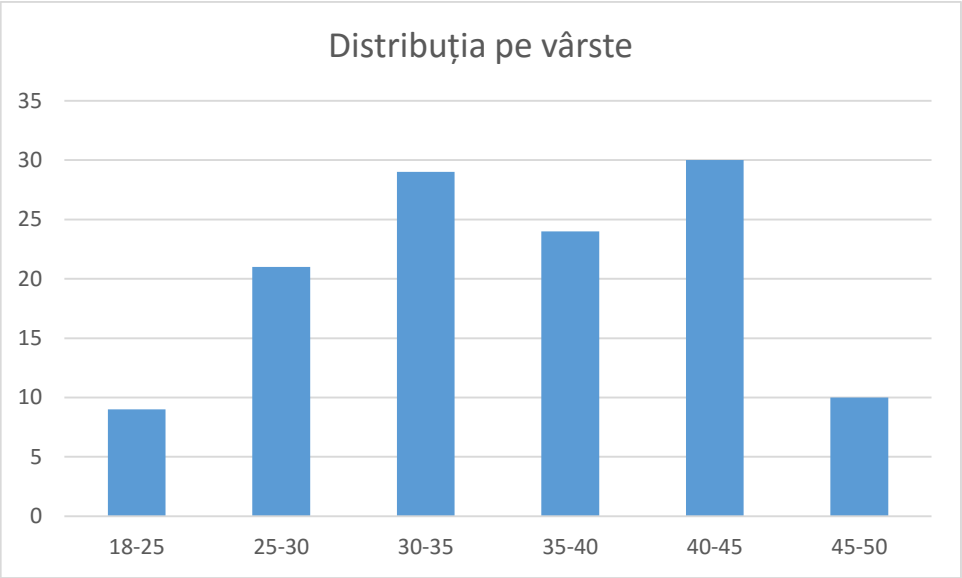
Baza de date ce conține asocierea dintre o anumită față a unei persoane și starea pe care aceasta o exprimă este bazata pe baza de date CK+ (Cohn-Kande Extins). Aceasta conține 10,727 de imagini care reprezintă 593 de secvențe de exprimare a unei emoții pentru 123 de subiecți voluntari.

În Imaginea 18 este descrisă structura directoarelor din baza de antrenare:

```
Cohn-Kanade Database
├── cohn-kanade-images
│   ├── S005
│   │   ├── 001
│   │   │   ├── S001_001_00000001.png
│   │   │   ├── ...
│   │   │   └── S001_001_000000yy.png
│   │   ├── ...
│   │   └── 00n
│   │       ├── S001_00n_00000001.png
│   │       ├── ...
│   │       └── S001_00n_000000yy.png
│   ├── ....
│   └── Sxxx
│       ├── 001
│       │   ├── Sxxx_001_00000001.png
│       │   ├── ...
│       │   └── Sxxx_001_000000yy.png
│       ├── ...
│       └── 00n
│           ├── Sxxx_00n_00000001.png
│           ├── ...
│           └── Sxxx_00n_000000yy.png
├── Emotion
│   ├── S005
│   │   ├── 001
│   │   │   └── S001_001_000000yy_emotion.txt
│   │   ├── ...
│   │   └── 00n
│   │       └── S001_00n_000000yy_emotion.txt
│   ├── ....
│   └── Sxxx
│       ├── 001
│       │   └── Sxxx_001_000000yy_emotion.txt
│       ├── ...
│       └── 00n
│           └── Sxxx_00n_000000yy_emotion.txt
└──
```

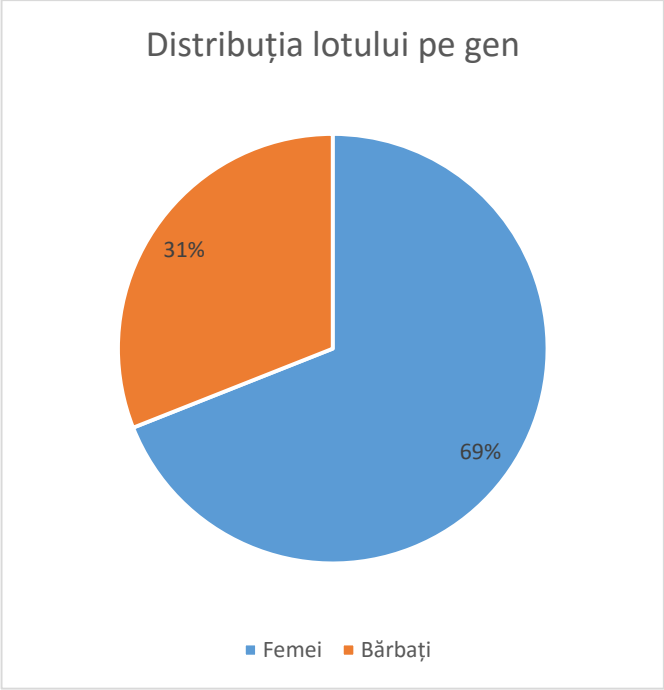
Imaginea 18 Structura de directoare a bazei de date

În cartea „Comprehensive database for facial expression analysis” [14], autorul J. Cohn descrie baza sa de date pentru determinarea expresiei faciale. Pentru fiecare persoană în parte, s-au realizat o serie de fotografii folosind o cameră Panasonic AG-7500, fiecare serie începând și terminând în poziția neutră a feței pacientului.



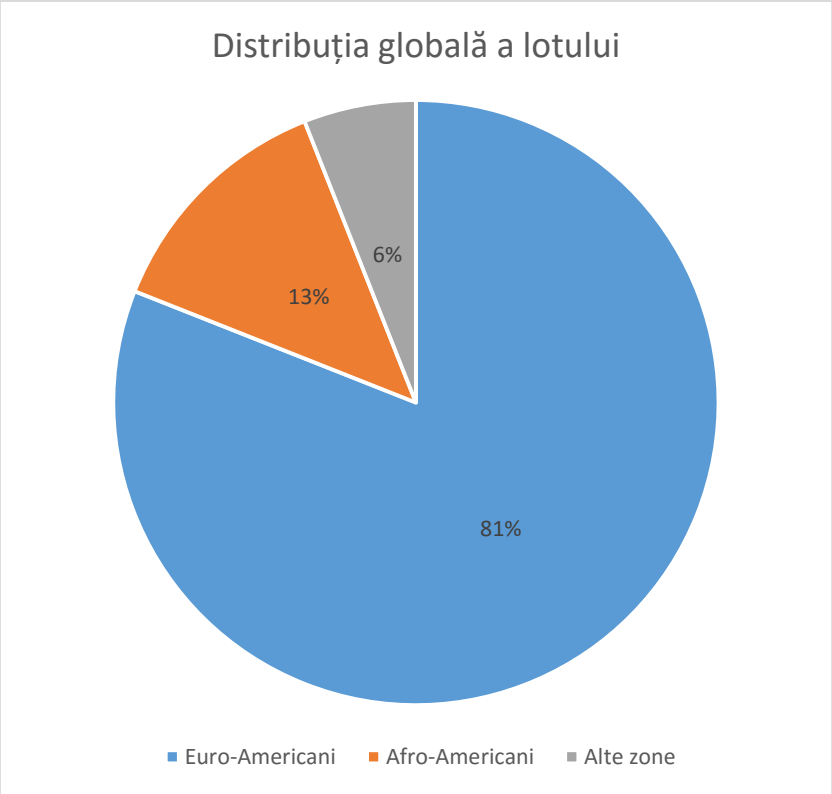
Tabelul 1 Distribuția lotului pe vârste

Participanții au vârste cuprinse între 18 și 50 de ani [Tabelul 1], 69% dintre ei fiind de sex feminin, iar 31% sunt bărbați [Tabelul 2].



Tabelul 2 Distribuția lotului pe gen

Din lot, 81% din persoane sunt Euro-Americani, 13% Afro-Americani și doar 6% din alte zone ale globului []. Subiecții au fost instruiți să efectueze o serie de 23 de imagini faciale. Această serie pentru imaginile frontale sau cu un unghi de aproximativ 30 de grade au fost digitalizate în imagini de dimensiuni 640x490 sau 640x480 pixeli având dimensiunea informațiilor per pixel de 8 biți pentru imaginile alb-negru sau 24 biți pentru cele color.



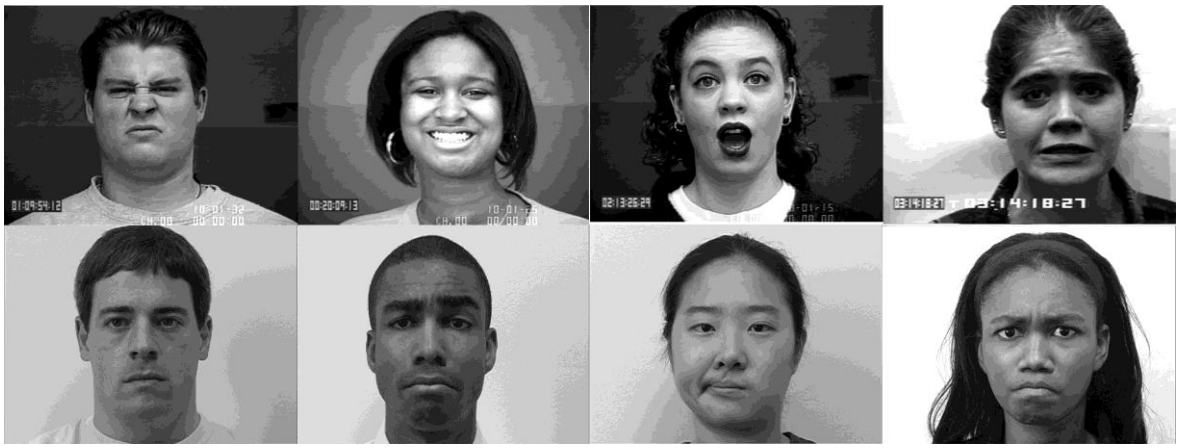
Tabelul 3 Distribuția pe glob a lotului de subiecți

În setul de date inițial oferit de baza CK sunt prezente doar 486 de secvențe de la 97 de indivizi. Pentru distribuția CK+ au fost realizate 593 de secvențe de la 123 de persoane, cu aproximativ 22% mai multe secvențe si 27% mai mulți subiecți. Secvența fotografiilor variază din punct de vedere a duratei, între 10 și 60 de cadre pe secunda și de asemenea conțin și informații de la expresii neutre.

În Tabelul 4 este reprezentată frecvența (N) anumitor caracteristici sau unități de acțiune (UA) ale feței pentru setul de imagini al bazei de date.

UA	Nume	N	UA	Nume	N
1	Fruntea interioară ridicată	173	18	Buzele strânse	9
2	Fruntea exterioară ridicată	116	20	Buze întinse	77
4	Fruntea lăsată	191	21	Mușchii gâtului încordați	3
5	Buza superioară ridicată	102	23	Buzele încordate	59
6	Obrazul ridicat	122	24	Buzele lipite	57
7	Orbicularul ochiului	119	25	Buzele despărțite	287
9	Cute ale nasului	74	26	Gura larg deschisă	48
10	Buza superioară ridicată	21	27	Zâmbet exagerat	81
11	Șanțul naso-labial	33	28	Buzele supte	1
12	Colțul gurii lăsat	111	29	Bărbia în piept	1
13	Obrazul căzut	2	31	Dinți strânși	3
14	Gropiță în obraz	29	34	Obraji umflați	1
15	Colțurile gurii căzute	89	38	Nări dilatate	29
16	Buza inferioară întoarsă	24	39	Nări ridicate	16
17	Bărbie încordată	196	43	Ochi închiși	9

Tabelul 4 Frecvența anumitor caracteristici prezente în setul de date al bazei CK+



Imaginea 19 Exemple de emoții din setul de date

Din setul întreg de imagini avute în baza de date, au fost extrase 7 categorii de emoții pe care persoanele le-au avut de exprimat: Dezgust (a), Fericire (b), Surprindere (c), Frică (d), Furie (e), Dispreț (f), Supărare (g) și dar și Neutru (h) [Imaginea 19]. Pentru fiecare emoție în parte avem asociate anumite caracteristici din tabelul anterior care sunt prezente în definirea expresiei faciale. Aceste criterii sunt prezentate în Tabelul 5.

Emoție	Criteriu
Furie	UA23 și UA24 trebuie să fie prezente
Dezgust	Atât UA9 cât și UA12 trebuie să fie prezente
Frică	Combinația dintre UA1+2+4 trebuie să fie prezentă, iar dacă UA5 este intensă, atunci UA4 poate fi absentă
Fericire	UA12 trebuie să fie prezentă
Supărare	UA1+4+15 trebuie să fie prezente. Excepție este combinația UA6+15
Suprindere	UA1+2 sau UA5 trebuie să fie prezente, dar intensitatea lui UA5 nu trebuie să fie mai mare decât cea a lui UA2
Dispreț	UA14 trebuie să fie prezentă

Tabelul 5. Descrierea emoțiilor pe baza caracteristicilor faciale

Pentru doar 327 de poze există o mapare exactă a emoției exprimate de către oameni, această mapare aflându-se în directorul „Emotion” și având calea identică cu cea a pozei pe care o reprezintă.

Pentru a determina un cod de emoție specific pentru imaginile folosite în setul de date de antrenament, acestea s-au etichetat conform Sistemului de Codificare a Acțiunii Faciale (SCAF) (FACS – „Facial Action Coding System”) dezvoltat de către W. Friesen și J. Hager în anul 2002 [15], rezultând un proces împărțit în trei pași [16]:

1. S-au comparat codurile din SCAF cu tabelul de predicție a emoției, ce conține configurațiile în termenii Unității de Acțiune (UA) pentru majoritatea emoțiilor cu excepția disprețului. Dacă o secvență satisface în mare măsură o emoție, aceasta este codificată provizoriu ca aparținând acelei categorii emoționale. În această primă etapă comparația cu tabelul de predicție a emoției s-a facut strict prin aplicarea regulii existenței UA în tabelă, ceea ce duce la omiterea altor unități intrate ulterior în standard.

2. După finalizarea primului pas, se efectuează o noua filtrare a unităților de acțiune prezente într-o anumită imagine în funcție de consistența lor în determinarea unei anumite emoții. De exemplu, UA4 poate fi considerată inconsistentă pentru a reprezenta surprinderea unei persoane (UA4 este o componentă a emoțiilor negative, nu a surprinderii). În contextul dezgustului, UA4 joacă un rol important alături de alte unități precum UA9. Similar se evaluează și unitățile care nu apar. În Tabelul 5 se poate observa o listă a criteriilor ce determină o anumită expresie în cazul lipsei unei anumite unități.

Dacă o UA lipsește, alte considerente sunt luate în considerare. De exemplu, în cazul UA20, aceasta nu trebuie să fie prezentă decât în cazul în care descrie frica. UA9 sau UA10 nu pot determina o altă emoție decât cea de dezgust, dar de asemenea pot fi prezente pentru a determina furia.

3. Al treilea pas este implicat în interpretarea unei anumite expresii, dacă aceasta se aseamănă sau nu cu categoria emoțională prezisă. Acest pas nu este complet independent de ceilalți doi precedenți deoarece expresiile care includ una dintre componentele necesare exprimării unei emoții sunt foarte posibile să facă parte totuși din altă categorie. Cu toate acestea, acest pas a fost introdus deoarece codurile SCAF descriu expresia doar la exprimarea ei maximă. Astfel, există riscul ca o anumită persoană să nu exprime o anumită emoție în proporție de 100%, iar de aici rezultă eroarea pe care aceste coduri o au în determinarea categoriei emoționale a persoanei.

Această bază de date este suficientă pentru începutul unei aplicații de recunoaștere a emoțiilor deoarece are în componență suficiente imagini distribuite uniform pe cele șapte emoții pe care aplicația construită le va recunoaște.

III.4. Algoritm utilizat

III.4.1. Descrierea algoritmului

Algoritmul folosit este unul de clasificare deoarece vom prelua 15 parametri distincți ai feței persoanei și în urma trecerii prin trei straturi intermediare în care se ajustează ponderea pentru fiecare parametru se va afla clasa din care face parte perechea de 15 parametri.

În primul rând, în metoda „main” din scriptul *„classifierNeuralNetwork.py”* voi declara un obiect de tipul FaceParameters pentru a încărca predictorul și detectorul o singură dată și să îl folosesc pentru a analiza fiecare poză din setul de date.

```
if __name__ == "__main__":
    # Initialize Face Parameters data
    face_parameters = FaceParameters()
```

```
def __init__(self):
    self.shape_predictor="shape_predictor_68_face_landmarks.dat"
    # initialize dlib's face detector
    print("[INFO] loading facial landmark predictor...")
    self.detector = dlib.get_frontal_face_detector()
    # create the facial landmark predictor
    self.predictor = dlib.shape_predictor(self.shape_predictor)
```

Un algoritm pentru o rețea neuronală trebuie să fie inițializat, și anume trebuiesc specificate numărul de layere intermediare, numărul de neuroni prezenți în fiecare layer, numărul maxim de iterații pe care algoritmul îl va parcurge pentru antrenarea rețelei, rata cu care se actualizează ponderea fiecărui neuron și toleranța. Constructorul clasei MLPClassifier are ca și parametri toate valorile menționate anterior, dar și altele care au o valoare predefinită în cazul în care se omite folosirea unei valori diferite.

```
def __init__(self, hidden_layer_sizes=(100,), activation="relu",
    solver='adam', alpha=0.0001, batch_size='auto',
    learning_rate="constant", learning_rate_init=0.001,
    power_t=0.5, max_iter=200, shuffle=True, random_state=None,
    tol=1e-4, verbose=False, warm_start=False, momentum=0.9,
    nesterovs_momentum=True, early_stopping=False,
    validation_fraction=0.1, beta_1=0.9, beta_2=0.999,
    epsilon=1e-8, n_iter_no_change=10):

    sup = super(MLPClassifier, self)
    sup.__init__(hidden_layer_sizes=hidden_layer_sizes,
        activation=activation, solver=solver, alpha=alpha,
        batch_size=batch_size, learning_rate=learning_rate,
        learning_rate_init=learning_rate_init, power_t=power_t,
        max_iter=max_iter, loss='log_loss', shuffle=shuffle,
        random_state=random_state, tol=tol, verbose=verbose,
        warm_start=warm_start, momentum=momentum,
        nesterovs_momentum=nesterovs_momentum,
        early_stopping=early_stopping,
        validation_fraction=validation_fraction, beta_1=beta_1,
        beta_2=beta_2, epsilon=epsilon,
        n_iter_no_change=n_iter_no_change)
```

Acest model de rețea neuronală optimizează eroarea astfel încât aceasta să scadă sub formă logaritmică folosind gradientul stohastic descendent. Parametrii de intrare sunt următorii:

- *hidden_layer_sizes*: este un nuplu de dimensiune nr_straturi_al_rețelei – 2, iar ca și valoare predefinită este 100. Este cu două elemente mai mic deoarece

stratul de intrare și cel de ieșire nu se adaugă aici. Elementul *i* din acest nuplu reprezintă numărul de neuroni pe care stratul *i* îi are in componență.

- *activation*: reprezintă modul în care arată funcția de activare descrisă în descrierea unei rețele neuronale [Capitolul III.2.]. Valoarea predefinită este *„relu”* – funcția liniară rectificată (*„rectified linear unit function”*) care este de forma $f(x) = \max(0, x)$.

- *solver*: Reprezintă metoda prin care se actualizează ponderea fiecărui neuron. Valoarea predefinită este *„adam”* și reprezintă metoda gradientului stohastic optimizată de către Kingma, Diederik și Jimmy Ba. Această metodă este utilizată în mod special atunci când se folosește o bază de date cu sufient de multe întregistrări. Altfel, poate fi folosită metoda *„lbfgs”* care reprezintă o optimizare a metodelor cvasi-Newton-iene.

- *alpha*: ajustarea pentru metoda gradientului.

- *batch_size*: mărimea lotului pentru optimizarea stohastică

- *learning_rate*: modul în care algoritmul ajustează ponderea pentru fiecare neuron. Valoarea predefinită este „constant”

- *learning_rate_init*: Rata folosită inițial pentru ajustarea ponderilor. Aceasta este luată în considerare doar dacă metoda de actualizare a ponderii este *„adam”*

- *power_t*: Exponentul pentru rata de învățare. Este folosit pentru a ajusta rata efectivă de învățare atunci când *„learning_rate”* este *„invscaling”* iar *solver*-ul este *„sgd”*

- *max_iter*: Numărul maxim de iterații pentru antrenarea rețelei neuronale. Dacă s-a ajuns la toleranța precizată, atunci numărul maxim de iterații nu mai este luat în considerare deoarece poate duce la o supra-învățare a rețelei

- *shuffle*: O variabliă de tip adevărat/fals, care precizează dacă setul de date să fie amestecat la fiecare iterație sau nu

- *tol*: Toleranța pentru optimizare. Atunci când în ultimele *„n_iter_no_change”* eroarea nu scade sub această toleranță, algoritmul se oprește

- *verbose*: Variabilă de tip binar care precizează daca progresul pentru fiecare iterație să fie afișat în consolă sau nu
- *warm_start*: Variabilă de tip binar pentru a transmite rețelei dacă să folosească antrenamentele anterioare sau să șteargă tot
- *momentum*: Folosit doar când *solver*-ul este *„sgd”* și reprezintă momentul pentru actualizarea gradientului descendent
- *nesterovs_momentum*: Indică când se folosește momentul Nesterov și este folosit doar când *momentum* este mai mare de 0 iar *solver*-ul este *„sgd”*
- *early_stopping*: Menționează rețelei o oprire anticipată atunci când scorul de validare nu se îmbunătățește
- *validation_fraction*: Proporția datelor de antrenament ce vor fi rezervate pentru validarea rețelei atunci când *„early_stopping”* este activ.
- *epsilon*: Valoarea pentru stabilitatea numerică atunci când *solver*-ul este *„adam”*
- *n_iter_no_change*: Numărul maxim de iterații succesive în care eroarea nu se îmbunătățește cu *„tol”*

Algoritmul pentru determinarea celor 15 parametri este aplicat pentru fiecare cadru preluat folosind camera web sau pentru fiecare fotografie încărcată la momentul antrenării. Se parcurg toate fețele prezente în acel cadru, și se crează două liste: una pentru cei 7 parametri care reprezintă distanțele, iar cealaltă pentru cei 8 parametri cu valori binare. În variabila *„shape”* salvez inițial cele 68 de puncte obținute folosind predictorul, iar apoi tot în aceeași variabilă salvez o lista de coordonate x și y a celor 68 de puncte. Lista de coordonate o creez în metoda *„shape_to_np”* folosind modulul NumPy. Toate aceste operații se realizează în metoda *„loop_over_faces”* ce primește ca variabile de intrare cadrul în care se face analiza, lista de fețe obținute anterior folosind detectorul inițializat în clasa *„FaceParameters”* și predictorul care este inițializat o singură dată la intrarea în aplicație.

```
def loop_over_faces(frame, faces, predictor):
    # loop over the faces
    print("[INFO] loop over faces...")
    distances = []
    binary_parameters = []
```

```
    for (i, face) in enumerate(faces):
        # determine the facial landmarks for the face region, then
        # convert the facial landmark (x, y)-coordinates to a NumPy
        # array
        shape = predictor(frame, face)
        shape = shape_to_np(shape)

        distances = get_distance(shape)
        binary_parameters = get_binary_parameters(shape)

        # draw (x,y)-coordinates for the facial landmarks
        # for (x, y) in shape:
        #     cv2.circle(frame, (x, y), 1, (0, 0, 255), -1)

    if len(distances) == 0:
        return None
    return distances, binary_parameters
```

Pentru a determina lista celor 7 distanțe, apelez metoda *„get_distance”* care primește ca parametru coordonatele celor 68 de puncte determinate.

Folosind informațiile despre poziționarea celor 68 de puncte ale feței descrise în *Imaginea 4 Poziționarea celor 68 de puncte de pe față din baza iBUG 300-W*, salvez în variabile poziția punctelor de interes. Deoarece lucrăm în Python, o listă începe de la elementul 0, astfel numerele ce reprezintă punctul din imagine sunt cu o unitate mai mici.

```
# Left eye
mid_eye_left = 39
down_eye_left = 41
up_eye_left = 38
up_eyebrow_left = 19

# Right eye
mid_eye_right = 42
down_eye_right = 46
up_eye_right = 44
up_eyebrow_right = 24

# Eyebrow mid
eyebrow_left = 21
eyebrow_right = 22
```

```
# Mouth
mid_mouth = 48
mid_mouth_right = 54
top_lip_up = 50
top_lip_down = 62
lower_lip_up = 66
lower_lip_down = 57
```

Pentru elementele feței care sunt simetrice, am calculat distanțele pentru ambele părți iar apoi am realizat o medie pentru o precizie mult mai bună a algoritmului. Distanțele le calculez fie între coordonatele x a punctelor dacă am nevoie de o distanță orizontală, fie între cele y atunci când distanța este pe verticală. Deoarece exista posibilitatea obținerii unor distanțe negative, ceea ce din punct de vedere fizic nu pot exista, am decis ca toate valorile salvate să fie în modul.

Un element din lista de coordonate poate fi accesat folosind sintaxa *shape.item(poziția_din_listă, coordonata_pe_care_o_dorim).*

Deoarece nu putem așeza fiecare subiect la o distanță fixă față de cameră, avem nevoie să raportăm toate distanțele la o distanță fixă pentru a transmite rețelei neuronale distanțe asemănătoare cu cele de antrenament. Această distanță la care voi raporta toate celelalte distanțe este cea dintre ochi calculată ca distanța orizontală dintre interiorul ochiului din dreapta și cel din stânga:

```
mid_eye_distance = np.absolute(shape.item(mid_eye_right, x) -
                               shape.item(mid_eye_left, x))
```

Așa cum am menționat anterior, pentru distanțele legate de ochii persoanei , voi calcula distanța pentru fiecare din cei doi ochi, iar apoi voi realiza media.

- Distanța la care sunt ridicate sprâncenele:

```
left_eye_distance1 = np.absolute(shape.item(up_eyebrow_left, y) -
                                 shape.item(mid_eye_left, y))
right_eye_distance1 = np.absolute(shape.item(up_eyebrow_right, y) -
                                  shape.item(mid_eye_right, y))
distances.append((left_eye_distance1 + right_eye_distance1)/2)
```

- Distanța pleoapei superioare față de sprânceană:

```
left_eye_distance2 = np.absolute(shape.item(up_eyebrow_left, y) -
                                 shape.item(up_eye_left, y))
```

```
right_eye_distance2 = np.absolute(shape.item(up_eyebrow_right, y) -
                                  shape.item(up_eye_right, y))
distances.append((left_eye_distance2 + right_eye_distance2)/2)
```

- Distanța de deschidere a ochiului:

```
left_eye_distance4 = np.absolute(shape.item(up_eye_left, y) -
                                 shape.item(down_eye_left, y))
right_eye_distance4 = np.absolute(shape.item(up_eye_right, y) -
                                  shape.item(down_eye_right, y))
distances.append((left_eye_distance4 + right_eye_distance4)/2)
```

Deoarece pe fața unui om gura există o singură data, distanțele caracteristice acesteia sunt calculate o singură data:

```
distances.append(np.absolute(shape.item(top_lip_up, y) -
                             shape.item(top_lip_down, y)))
distances.append(np.absolute(shape.item(lower_lip_up, y) -
                             shape.item(lower_lip_down, y)))
distances.append(np.absolute(shape.item(mid_mouth, x) -
                             shape.item(mid_mouth_right, x)))
distances.append(np.absolute(shape.item(top_lip_down, y) -
                             shape.item(lower_lip_up, y)))
```

Normarea distanțelor față de distanța dintre cei doi ochi se realizează parcurgând lista și împărțind fiecare element la distanța calculata în variabila *„mid_eye_distance”*:

```
# normalize distances
for i in range(0, len(distances)):
    distances[i] = distances[i] / mid_eye_distance
```

Folosind baza de date descrisă în secțiunea III.3. am parcurs toate subfolderele din path-ul database/training_dataset/emotion pentru a extrage toate fisierele text ce conțineau emoția pe care o exprimă subiectul. Această parcurgere este detaliată în codul următor:

```
for root, dirs, files in os.walk("./database/training_dataset"):
    for file in files:
        if file.endswith(".txt"):
            # Get emotion from subject
            file_path = os.path.join(root, file)
            # print("Emotion path: ", file_path)
            emotion = open(file_path, 'r').read()
```

Pentru a parcurge toate fișierele am folosit modulul OS integrat în distribuția Anaconda de Linux, si am specificat subfolder-ul în care metoda „walk” să realizeze căutarea de fișiere. În urma obținerii tuturor fișierelor și dosarelor, analizăm fiecare fișier, iar cu metoda „endswith” verificăm extensia. Dacă aceasta este de tipul „*.txt” atunci știu că în interiorul fișierului începând cu al treilea caracter se află codul emoției asociat unei fotografii. Fotografia reprezentativă pentru acea emoție se află în aceeași structură de dosare pornind de la dosarul „training_dataset” ca și fișierul ce conține codul emoției.

```
# Get image of subject with emotion
  image_path = file[:-12] + '.png'
  image_root = root.replace("emotion", "images")
  image_root = image_root.replace("\\", "/")
  image_root = image_root + '/'
  image_path = os.path.join(image_root[2:], image_path)
```

Pentru a crea calea specifică a fotografiei, preiau rădăcina proiectului care reprezintă locul unde se află proiectul salvat pe disc, salvată în variabila *„root”*, iar apoi concatenez cu numele fișierului text căruia îi elimin ultimele 12 caractere și adaug extensia specifică pentru fotografie.

Având construită astfel calea pentru fiecare fotografie, aplic algoritmul de extragere a celor 15 parametri pe care îi voi folosi pentru a antrena rețeaua neuronală.

```
# Get subject facial parameters
  facial_parameters=face_parameters.
                                    get_face_parameters(image_path)
```

Această extragere se reazlizează în metoda *„get_face_parameters()”* în care citesc fotografia aflată în locația primită ca și parametru de intrare. Pentru a vedea în consolă ce fotografie este procesată și evoluția analizării am folosit metoda *„print”* și am pus ca prim cuvânt afișat *„[INFO]”* pentru a găsi ușor detaliile dorite.

Folosind modulul OpenCV citesc fotografia dorită, iar apoi, pentru a optimiza procesul de recunoaștere a fețelor, transform imaginea în alb-negru. Aplic detectorul de fețe asupra imaginii filtrate, iar acesta returnează fețele prezente în imagine. Metoda *„loop_over_faces()”* descrisă anterior și declarată în scriptul *„util.py”* returnează cei 15 parametri necesari antrenării rețelei neuronale pe care îi returnez și îi salvez în variabila *„facial_parameters”* prezentată anterior.

```
def get_face_parameters(self, image_src):
    print("[INFO] load image: {}".format(image_src))

    # load the input image
    image = cv2.imread(image_src)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    # detect faces in image

    print("[INFO] detect faces...")
    faces = self.detector(gray)

    distances, binary_parameters = utils.loop_over_faces(image,
faces, self.predictor)

    print("[INFO] parameters get successfully")

    fac_param = numpy.concatenate((distances, binary_parameters))

    return fac_param
```

Setul de date de antrenament conține două liste, prima va avea ca element un nuplu de 15 parametri reprezentând cele 8 distanțe menționate în descrierea algoritmului și 7 parametri de tip boolean iar a doua va fi compusă din emoția caracteristică. Adăugarea în listă se face prin următoarea secvență de cod:

```
# add face parameters and emotion to training set
    training_data_set.append(facial_parameters)
    training_output_set.append(int(emotion[3:4]))
```

Antrenarea rețelei neuronale se realizează folosind metoda *„fit()”* din clasa *„MLPClassifier”*.

```
    mlp.fit(X_train, y_train)
```

După finalizarea antrenamentului, se așteaptă apăsarea tastei „Enter” pentru a activa camera web a unității pe care rulează codul python.

III.4.2. Alți algoritmi posibili

Clasificarea algoritmilor în funcție de apartenența unui eșantion de date de intrare la o anumită clasă de ieșire se face astfel:

- Clasificare exhaustivă: un anumit eșantion aparține exclusiv unui singur grup
- Clasificare cu suprapunere: un eșantion de date aparține în același timp mai multor grupuri. Această clasificare este specifică algoritmilor de tip fuzzy
- Clasificare ierarhică: toate eșantioanele aparțin inițial aceluiași grup, iar apoi pe baza anumitor proprietăți se formează un arbore de decizie.

III.4.2.1. Arbori de decizii

Un arbore de decizii este o reprezentare a unei proceduri decizionale pentru a determina o anumită clasă de apartenență pentru o instanță de intrare. Fiecare nod al unui astfel de arbore specifică fie un nume de clasă de apartenență, fie o anumită partiție care va asocia în final instanța primită cu o anumită clasă de ieșire. Fiecare sub-partiție corespunde unei sub-probleme de clasificare pentru un anumit subset din instanță care este rezolvat de un sub-arbore. Un arbore de decizii poate fi privit ca o strategie de tipul divide și cucerește pentru a clasifica un anumit obiect.

În mod formal, fiecare entitate dintr-un arbore de decizii poate fi definită astfel:

- Entitate/nod frunză sau nod de răspuns care conține unele clase de apartență
- Entitate/nod de decizie care conține un anumit atribut ce se va transforma în dată de intrare pentru un ulterior subarbore.

III.4.2.1.a. Algoritmul ID3

Ross Quinlan a dezvoltat inițial algoritmul ID3 [17] la universitatea din Sydney. Acesta a prezentat pentru prima dată algoritmul în cartea sa, „Machine Learning” [17] volumul 1, numărul 1 în anul 1975. Algoritmul unul de învățare supervizat care crează un arbore decizional pe baza unui set de date de antrenament. Arborele creat va fi folosit pentru a clasifica următoarele seturi de date de intrare.

Algoritmul ID3 se bazează pe entropia informațională dată pe colecția de exemple. De asemenea, acesta operează cu valori discrete nenumerice. Toate datele de intrare trebuie să fie complete. Nu se permit exemple în care un atribut să aibă valoare necunoscută.

Se calculează câștigul informațional pentru fiecare atribut după fiecare clasificator. Clasificatorul cu cel mai mare câștig informațional va constitui nodul rădăcină al arborelui de decizie și regulile acelui clasificator se pun la începutul bazei de cunoștințe, după care se utilizează ceilalți clasificatori în ordinea dată de cel mai mare câștig.

Singura limitare a algoritmului ID3 sunt clasificatorii, care nu suportă valori numerice continue.

Pseudocodul pentru acest algoritm este unul destul de simplu.

```
Intrari:
      R: mulțime de clasificatori
      C: mulțime de decizii
      S: setul de antrenament
Ieșiri:
      Arbore de decizie
Start
Inițializare arbore gol;
      DACĂ S este gol ATUNCI
RETURNEAZĂ un singur nod al arborelui care conține o eroare
      SFÂRȘIT DACĂ


      DACĂ R este gol ATUNCI
         RETURNEAZĂ un singur nod ce conține cea mai comună valoare
               găsită în setul de antrenament S
      SFÂRȘIT DACĂ
D ← atributul din R care are cel mai mare câștig informațional
G(D,S)
      {dj | j = 1, 2, ..., m}     // Valorile atributului D
{Sj | j = 1, 2, ..., m}    // Subset din datele de antrenement
constituit pentru atributele dj înregistrate în D
      RETURNEAZĂ
Un arbore a cărui nod rădăcină este D, arcele marcate prin d1, d2,
... dm și subarborii ID3( R-{D} C, S1), ID3( R-{D} C, S2) … ID3( R-
{D} C, Sm).
End
```

Pe scurt, dacă avem o distribuție de probabilități, algoritmul calculează entropia pentru fiecare atribut din setul de intrare folosind formula:

$$E(p) = -\sum_{i=1}^{n}(p_i \cdot \log_2 p_i)$$

Apoi, folosind entropia calculată anterior, se va calcula câștigul informațional pentru un anumit atribut cu distribuția *p*, folosind formula de mai jos:

$$G(p,T) = E(p) - \sum_{i=1}^{n-1}(p_i \cdot E(p_i))$$

unde valorile p_i sunt toate valorile posibile ale atributului T.

În urma determinării câștigului informațional al fiecărui atribut, baza de cunoștințe se organizează după acest câștig, simplificând astfel mecanismul interpretativ.

III.4.2.1.b. Algoritmul ID4

Acest algoritm a fost propus în anul 1993 tot de către Ross Quinlan în special pentru a depăși limitările algoritmului ID3.

Una dintre acestea este că algoritmul ID3 este foarte sensibil la lucrul cu numere foarte mari. Această sensibilitate se poate observa la codurile numerice personale ale persoanelor din România care sunt numere mari și în același timp unice.

Pentru a rezolva această problemă, algoritmul ID4 folosește câștigul informațional, care nu aduce nimic nou, dar este folosit pentru determinarea raportului de câștig.

Raportul de câștig este definit în felul următor:

$$GR = \frac{G(p,T)}{SI(p,T)}$$

unde

$$SI(p,Test) = \sum_{j=1}^{n} P'\left(\frac{j}{p}\right) \cdot \log\left(P'\left(\frac{j}{p}\right)\right)$$

$P'\left(\frac{j}{p}\right)$ reprezintă proporția prezenței elementului aflat la poziția *p* pentru setul de date *j*. La fel ca și în cazul algoritmului ID3 datele sunt sortate la fiecare nod astfel încât să se determine cât mai exact elementul în funcție de care se va face următoarea împărțire pentru noul subarbore.

Pentru algoritmul ID4, arborele de decizii se va construi pe același model ca și cel al algoritmului precedent folosind un set de date de antrenament.

Comparativ cu algoritmul ID3, algoritmul ID4 selectează cel mai important atribut din parametrii de intrare bazat pe entropia calculată și câștigul informațional pentru a determina arborele final de decizii. De asemenea, acesta aduce câteva îmbunătățiri precum posibilitatea utilizării unor date de intrare continue, posibilitatea de a folosi variabile necunoscute sau a nu le introduce deloc și abilitatea de a folosi atribute cu diferite grade de importanță.

III.4.2.2. Algoritm de clasterizare / Algoritmul K-means

Algoritmul K-means de clasterizare a fost prezentat de către Hartigan [18] în anul 1975. Scopul algoritmului este de a împărți M puncte formate din N dimensiuni în K clastere care pot fi bazate pe distanța dintre puncte, astfel încât va rezulta o minimizare a acesteia, sau pe similitudinea dintre puncte, care se dorește a fi cât mai mare.

Algoritmul are nevoie de o matrice de M puncte cu N dimensiuni ca și intrare, și o matrice a celor K centroizi cu N dimensiuni. Procedura generală este de a căuta pozițiile celor K centroizi făcând media pătratelor punctelor cu N dimensiuni aflate în interiorul clasterului respectiv și mutarea punctelor periferice în alte clastere mai apropiate.

Se definesc N = 10 puncte și K = 3 clastere [Imaginea 20]. C_1, C_2, C_3 – centroizii celor K clastere.

Imaginea 20 10 puncte grupate în 3 clastere

Pași ai algoritmului:

Pas 1. Se definesc aleator cei K centroizi și se calculează funcția obiectiv dintre fiecare punct și fiecare centroid definit

Pas 2. Fiecare punct este alocat grupului pentru care distanța la centroid este cea mai mică/mare (în funcție de construcția clasterelor: bazate pe distanță/similitudine)

Pas 3. În grupurile formate se calculează noul centroid dat de punctele ce aparțin centroidului; se repetă pasul 2 și 3 până când centroizii nu se mai mișcă.

Observație: Un punct aparține unui singur claster.

III.4.2.3. Rețele neuronale de tip fuzzy

În teorie, rețelele neuronale și sistemele fuzzy sunt echivalente prin faptul că ambele sunt convertibile, totuși, în practică, fiecare are propriile avantaje și dezavantaje. De exemplu, pentru rețelele neuronale informația este acumulată automat prin algoritmi de propagare inversă, dar procesul de învățare este unul lent, iar analiza rețelei antrenate este grea chiar imposibilă, asemănându-se unui proces de tip „cutie neagră”.

Sistemele fuzzy sunt de multe ori preferate datorită comportamentului lor care poate fi explicat bazat pe regulile fuzzy și prin urmare performanța lor poate fi optimizată ajustând aceste reguli la fiecare iterație.

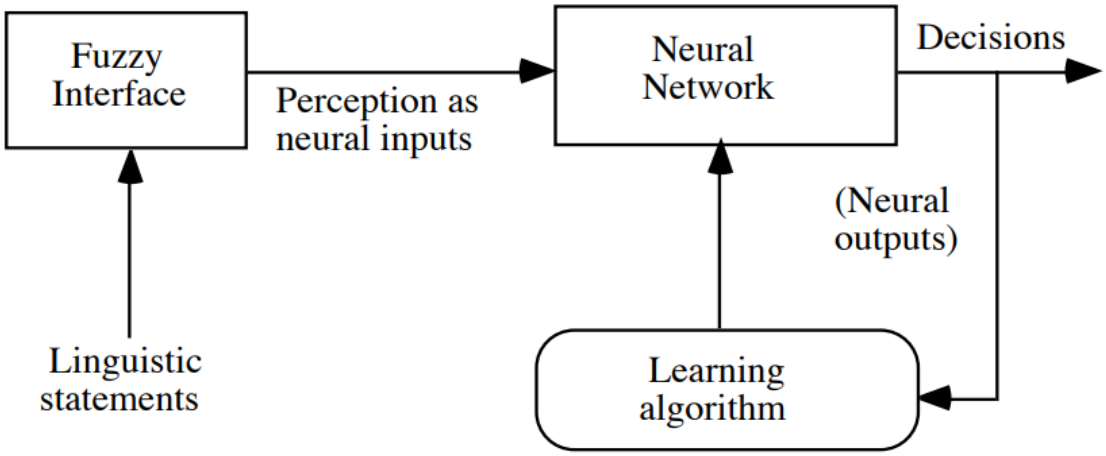
Un alt avantaj al folosirii rețelelor neuronale de tip fuzzy este abilitatea de a exprima numărul mare de ambiguități din gândirea umană și subiectivitatea într-o manieră comparativă.

Procesul de calcul proiectat pentru sistemele neuronale de tip fuzzy începe cu dezvoltarea unui „neuron fuzzy" bazat pe înțelegerea morfologiilor biologice neuronale iar apoi urmează mecanismul de învățare. Această proiectare duce la necesitatea respectării următorilor pași pentru a realiza un proces neuronal de tip fuzzy:

- Dezvoltarea unui model fuzzy bazat pe neuronii biologici
- Modelul conexiunilor sinaptice care să implementeze logica fuzzy în rețeaua neuronală
- Dezvoltarea unui algoritm de învățare folosind medota ajustării ponderii pentru fiecare legătură sinaptică.

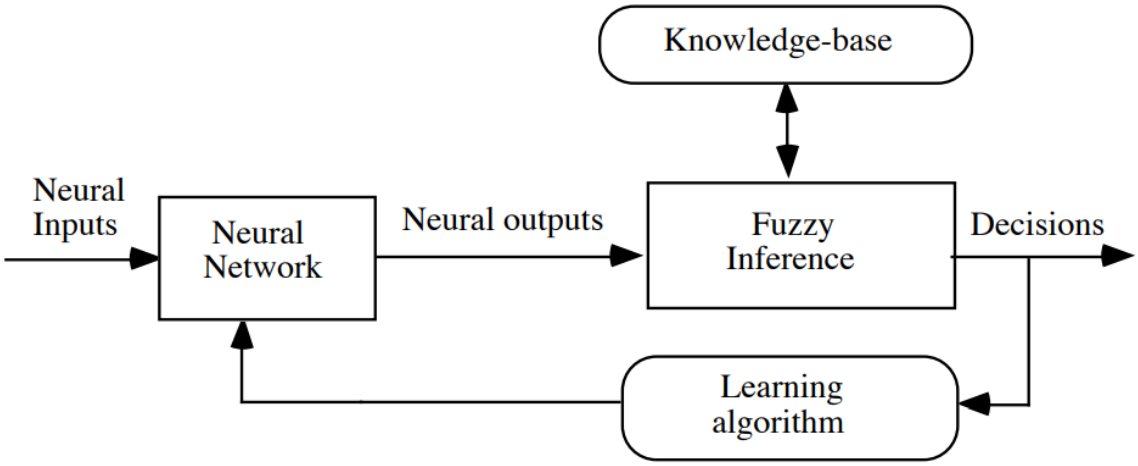
Astfel, cele două modele pentru sistemele neurale de tip fuzzy:

- Ca un răspuns al declarărilor lingvistice, blocul de interfață fuzzy furnizează un vector de intrare către o rețea neuronală multi-strat. Rețeaua neuronală poate fi antrenată pentru a obține decizia corectă [Imaginea 21].



Imaginea 21 Primul model de sistem fuzzy

- O rețea neuronală cu mai multe straturi conduce la un mecanism de interfață pentru logica fuzzy [Imaginea 22].



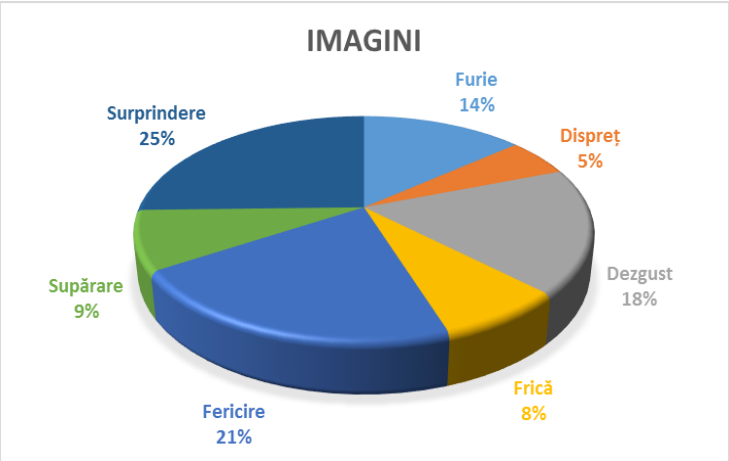
Imaginea 22 Al doilea model de sistem fuzzy

CAPITOLUL IV. Rezultate obținute

În urma parcurgerii setului de date, obținem o distribuție relativ echilibrată a emoțiilor, după cum se poate vedea și în tabelele următoare [Tabelul 6,Tabelul 7]:

	Imagini
Furie	45
Dispreț	18
Dezgust	59
Frică	25
Fericire	69
Supărare	28
Surprindere	83
Total	327

Tabelul 7 Numărul emoțiilor din setul de date



Tabelul 6 Distribuția emoțiilor

Pentru primul set de teste nu am luat în considerare fețele cu emoție neutră și am parcurs o singură dată setul de date. Am impărțit setul complet de date în 80% date de antrenament și 20% date de test. Am considerat ca datele de test să fie primele din setul complet. Pentru setul de date de test (75 de înregistrări) am prezis folosind rețeaua neuronală antrenată emoția reprezentată de persoana din poza și am salvat această predicție într-o listă.

Ulterior, am comparat această listă obținută cu lista care conține exact emoțiile transmise de persoană folosind o transmise de persoană folosind o matrice de confuzie și am obținut următoarele rezultate reprezentate în

reprezentate în

Tabelul 8 :

	Furie	Dispreț	Dezgust	Frică	Fericire	Supărare	Surprindere	Total
Furie	10	0	2	0	0	2	0	14
Dispreț	0	0	0	0	0	0	0	0
Dezgust	1	0	12	0	1	0	0	14
Frică	0	0	0	6	0	0	0	6
Fericire	0	0	1	1	14	0	0	16
Supărare	2	2	0	0	0	1	0	5
Surprindere	0	0	0	0	1	0	19	20
Eroare	3	2	3	1	2	2	0	

Tabelul 8 Matrice de confuzie pentru 75 date de test

Am făcut un raport al celor 75 de clasificări efectuate în zona de test și a rezultat o precizie totală de 82%, o sensibilitate de 84%, scorul F1 care reprezintă media armonică între precizie și sensibilitate de 0.82. Toate aceste date se pot vedea în Tabelul 9.

	Precizie	Sensibilitate	Scor F1	Imagini
Furie	0.80	0.86	0.83	14
Dispreț	0.00	0.00	0.00	0
Dezgust	0.76	0.93	0.84	14
Frică	0.75	1.00	0.86	6
Fericire	0.93	0.81	0.87	16
Supărare	0.00	0.00	0.00	5
Surprindere	1.00	0.95	0.97	20
Medie	0.82	0.84	0.82	75

Tabelul 9 Raportul clasificărilor pentru 75 date de test

Antrenând din nou rețeaua neuronală, dar de această dată trecând de două ori prin întreg setul de date, și creând setul de test din 130 de valori aleatoare din setul de antrenament obținem următoarea matrice de confuzie reprezentată în Tabelul 10 :

	Furie	Dispreț	Dezgust	Frică	Fericire	Supărare	Surprindere	Total
Furie	22	0	1	0	0	0	0	**23**
Dispreț	3	4	0	1	0	2	0	**10**
Dezgust	3	0	13	0	0	1	0	**17**
Frică	0	0	0	10	3	2	0	**15**
Fericire	0	0	0	5	26	0	0	**31**
Supărare	1	0	0	0	0	7	0	**8**
Surprindere	0	0	0	0	0	0	26	**26**
Eroare	**7**	**0**	**1**	**6**	**3**	**5**	**0**	

Tabelul 10 Matrice de confuzie pentru 130 date de test

Se obțin astfel valorile medii: precizie 85%, sensibilitate 83%, scor F1 0,83, descrise în Tabelul 11.

	Precizie	Sensibilitate	Scor F1	Imagini
Furie	0.76	0.96	0.85	23
Dispreț	1.00	0.40	0.57	10
Dezgust	0.93	0.76	0.84	17
Frică	0.62	0.67	0.65	15
Fericire	0.90	0.84	0.87	31
Supărare	0.58	0.88	0.70	8
Surprindere	1.00	1.00	1.00	26
Medie	**0.85**	**0.83**	**0.83**	**130**

Tabelul 11 Raport clasificare pentru 130 date de test

IV.1. Factori care pot influența rezultatele

Un prim factor care poate influența rezultatele este calitatea camerei de filmat. Atunci când rezoluția nu este suficient de bună, distanța dintre doi pixeli este mult mai mică, iar pozițiile punctelor de interes ale feței sunt eronate. O astfel de eroare poate duce la crearea greșită a setului de date de intrare către rețeaua neuronală, iar aceasta va returna un rezultat neconcludent.

Un alt factor este lumina din încăpere care poate afecta citirea corectă a feței. În momentul în care două puncte ale feței sunt destul de aproape, umbra apărută în urma luminii proaste poate astupa acel punct ceea ce conduce la o citire inconsistentă a datelor.

Absența unei componente a feței este un alt factor care poate influența rezultatele. Un astfel de comportament poate genera un rezultat precum „no face” în cazul aplicației mele. Această eroare este returnată atunci când nu se pot citi în întregime toate punctele necesare interpretării faciale.

Afecțiuni medicale anterioare precum paralizia de nerv facial sau anumite traumatisme ale feței pot împiedica aplicația să genereze un rezultat exact. Din cauza acestor afecțiuni, subiectul nu își poate exprima subiectele prin mimică.

CAPITOLUL V. Concluzii și dezvoltări ulterioare

În concluzie, algoritmul realizează o recunoaștere a emoțiilor exprimate prin intermediul mimicii în proporție de 80% folosind un set de date de antrenament de aproximativ 300 de înregistrări. Aceste rezultate se datorează minimizării numărului de puncte faciale pe care le analizez și clasificării emoțiilor în strict șapte clase.

Un element neprevăzut care ar putea apărea în momentul utilizării rețelelor neuronale și în special la folosirea algoritmilor de tip „învățare aprofundată” („deep learning”) este momentul la care se oprește rețeaua neuronală din procesarea setului de antrenament. Depășirea acestui moment poate duce la setarea unor ponderi greșite a nodurilor rețelei neuronale. Fiecare pondere setată greșit este regăsită amplificat în ultimul strat al rețelei.

Datorită proprietății de învățare și adaptare a rețelei neuronale aceasta poate fi aplicată în majoritatea domeniilor de studii, inclusiv în domeniul interpretării expresiei faciale. Alegerea folosirii unei rețele neuronale s-a dovedit a fi alegerea optimă pentru procesarea unui set de date de dimensiuni relativ mici, iar implementarea folosind limbajul Python a ajutat la crearea acestei aplicații.

Rezultatele obținute pot fi îmbunătățite prin acumularea tuturor datelor înregistrate și antrenarea frecventă a rețelei folosind aceste noi imagini preluate de camera de filmat.

V.1. Domenii de aplicabilitate

Expresia feței este un element esențial pentru conversațiile dintre persoane, de aceea această aplicație, cu ajustările necesare poate fi utilizată în toate domeniile unde există o cameră de filmat.

De exemplu, în domeniul interogărilor suspecților de către polițiști sau procurori, această aplicație poate fi o sursă de informare excactă a stărilor emoționale ale accestor suspecți prin stabilirea veridicității afirmațiilor lor.

V.2. Domenii în care aplicația poate fi îmbunătățită

Aplicația poate fi îmbunătățită prin încărcarea pe un server cloud a datelor înregistrate de toate locurile unde aceasta rulează, ca de exemplu în piețe publice, unde poate înregistra un flux foarte mare de persoane sau în aeroporturi unde sunt prezente persoane cu diferite trăsături ale feței.

Un set mai mare de date de antrenament poate duce la o precizie a predicției mult mai bună datorită varietății subiecților. Orice algoritm de inteligență artificială, cu cât are mai multe date de antrenament, cu atât este mai eficient.

Microexpresiile întâmpină o dificultate în a fi citite și interpretate de o cameră de filmat obișnuită. De aceea, este necesară o cameră de filmat tridimensională pentru crearea unui model 3D al feței care ar îmbunătăți rezultatele.

Un alt mod în care rezultatele predicției emoțiilor poate fi îmbunătățit este prin folosirea unui test poligraf care constă în măsurarea anumitor parametri fiziologici precum pulsul, tensiunea arterială, transpirața și respirația în timp ce subiectului uman i se pun anumite întrebări. Din păcate, această modalitate de îmbunătățire nu poate fi aplicată în locurile publice unde dorim analiza expresiilor faciale fără a intra în contact cu persoanele.

CAPITOLUL VI. Bibliografie

[1] V. Papilian, „Aparatul locomotor,” *Anatomia omului,* vol. I, 2003.

[2] V. Papilian, „Aparatul locomotor,” *Anatomia omului,* vol. I, p. 161, 2003.

[3] V. Papilian, „Aparatul locomotor,” *Anatomia omului,* vol. I, pp. 162-168, 2003.

[4] J. Kiss, „Fiziokinetoterapia și recuperarea medicală în afecțiunile aparatului locomotor,” în *Fiziokinetoterapia și recuperarea medicală în afecțiunile aparatului locomotor*, 2007, p. 191.

[5] R. C. A. R. a. P. P. W. Zhao, „Face recognition: A literature survey. ACM Computing Surveys,” în *Face recognition: A literature survey. ACM Computing Surveys*, 2003, pp. 399-458.

[6] K. Baek, „AI 2005,” în *AI 2005*, Heidelberg, Springer-Verlag, 2005, pp. 1229-1232.

[7] E. Matthes, „Python Crash Course,” în *A Hands-On, Project-Based Introduction to Programming*, San Francisco, no starch press, 2016, p. xxxi.

[8] „What is Anaconda?,” 1 Mai 2018. [Interactiv]. Available: https://www.anaconda.com/what-is-anaconda/. [Accesat 2 Iunie 2018].

[9] Open Source Computer Vision, „OpenCV documentation,” doxygen.com, 23 Februarie 2018. [Interactiv]. Available: https://docs.opencv.org/3.4.1/d0/de3/tutorial_py_intro.html. [Accesat 15 Iunie 2018].

[10] S. Haykin, Neural Networks and Learning Machines, PEARSON Prentice Hall, 1994.

[11] Ș. d. i. L.-N. IVANCIU, „Sisteme inteligente de suport decizional,” Facultatea de Electronică, Telecomunicații și Tehnologia Informației, Cluj.

[12] S. Haykin, „Neural Networks and Learning Machines,” în *Neural Networks and Learning Machines*, PEARSON Prentice Hall, 2009, p. 2.

[13] S. Haykin, Widrow and Stearns, 1985.

[14] J. C. Y. T. T. Hande, „Comprehensive database for facial expression anaysis,” în *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, 2000, pp. 46-53.

[15] W. F. J. H. P. Ekman, Facial Action Coding System; Research Nexus, Salt Lake City, UT, USA: Network Research Information, 2002.

[16] University of Pittsburgh, The Extended Cohn-Kanade Dataset (CK+): A complete dataset for action unit and emotion-specified expression, Pittsburgh, 2010.

[17] J. R. Quinlan, „Induction of Decision Trees," în *Machine Learning 1*, 1986, pp. 81-106.

[18] J. A. Hartigan, Clustering Algorithms, New York: Wiley, 1975.

[19] A.Mehrabian, „"Communication without words"," în *Psychology today, vol. 2, nr. 4*, 1968, pp. 53-56.

[20] B. &. M. T. M. Buchanan, „Pattern-directed inference systems," în *Model-directed learning of production rules. In Waterman & Hayes-Roth*, New York, 1978.

[21] Python Software Foundation, „General Python FAQ," 30 Mai 2018. [Interactiv]. Available: https://docs.python.org/3.6/faq/general.html. [Accesat 30 Mai 2018].

[22] N. P. R. S. H. Saket S Kulkarni, „Facial expression (mood) recognition from facial images using committee neural networks," 5 August 2009. [Interactiv]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2731770/. [Accesat 27 Ianuarie 2018].

Cuprinsul tabelelor prezente în lucrare

Cuprinsul imaginilor prezente în lucrare