UNIVERSITATEA “POLITEHNICA” DIN BUCUREȘTI

FACULTATEA DE ELECTRONICĂ, TELECOMUNICAȚII ȘI TEHNOLOGIA INFORMAȚIEI

SISTEM DE RECUNOAȘTERE VOCALĂ

IMPLEMENTAT PE ROBOTUL UMANOID NAO

Proiect de diplomă

prezentat ca cerință parțială pentru obținerea titlului de *Inginer* în domeniul Electronică și Telecomunicații

Programul de studii la licență: Microelectronică, Optoelectronică și Nanotehnologii

Conducători științifici

Prof. Dr. Ing. Corneliu BURILEANU

Ing. Ana Antonia NEACȘU

Ing. George CIOROIU

Absolvent

Ruxandra Ioana RĂDUCANU

București

2019

Universitatea "Politehnica" din București
Facultatea de Electronică, Telecomunicații și Tehnologia Informației
Departamentul **DCAE**

Anexa 1

TEMA PROIECTULUI DE DIPLOMĂ

a studentului **RĂDUCANU M.T. Ruxandra-Ioana , 442E**

1. Titlul temei: SISTEM DE RECUNOAȘTERE VOCALĂ IMPLEMENTAT PE ROBOTUL UMANOID NAO

2. Descrierea contribuției originale a studentului (în afara părții de documentare) și specificații de proiectare:
Lucrarea de față își propune dezvoltarea unui sistem de interacțiune om-mașină, prin intermediul vorbirii. Proiectul va fi alcătuit din două părți principale, însușind atât cunoștințe software cât și hardware. Scopul final este acela de a-i permite robotului umanoid Nao să recunoască un set de instrucțiuni transmise de către utilizator prin intermediul semnalului vocal și să genereze un răspuns în conformitate cu acesta.
În prima parte a proiectului se va dezvolta un sistem de recunoaștere automată a vorbirii (RAV), cu scopul identificării cuvintelor aparținând unui vocabular limitat. Acesta va fi implementat prin antrenarea unei rețele neurale profunde (DNN). Construirea și antrenarea rețelei se va realiza cu ajutorul librăriei Tensorflow, care reușește să ofere suport pentru dezvoltarea algoritmilor de inteligență artificială.
Partea hardware va consta în portarea rețelei pe robotul umanoid Nao și programarea acestuia pentru a executa o serie de acțiuni. Robotul va deveni astfel capabil să recunoască, în timp real, setul de cuvinte transmise de către vorbitor. Ca urmare a interpretării acestora, se va genera un răspuns motric sau vocal din partea robotului. Testarea va fi efectuată pe un grup neomogen de persoane, deoarece se dorește dezvoltarea unui sistem de recunoaștere vocală, independent de vorbitor.

3. Resurse folosite la dezvoltarea proiectului:
Limbaje de programare: Python-Tensorflow

4. Proiectul se bazează pe cunoștințe dobîndite în principal la următoarele 3-4 discipline:
Prelucrarea digitala a semnalelor, Decizie și estimare în prelucrarea informației, Programare obiect

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

6. Data înregistrării temei: 2018-11-28 17:22:52

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

Student,
semnătura:.....

Ing. Ana Antonia NEACSU, Ing. George CIOROIU
semnătura:..... cioroiu

Director departament,
Prof. dr. ing Claudius DAN
semnătura:.....

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

Cod Validare: **6d6305ffe5**

Declarație de onestitate academică

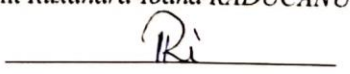
Prin prezenta declar că lucrarea cu titlul " *SISTEM DE RECUNOAȘTERE VOCALĂ IMPLEMENTAT PE ROBOTUL UMANOID NAO*", prezentată în cadrul Facultății de Electronică, Telecomunicații și Tehnologia Informației a Universității "Politehnica" din București ca cerință parțială pentru obținerea titlului de *Inginer* în domeniul *Electronică și Telecomunicații,* programul de studii *Microelectronică, Optoelectronică și Nanotehnologii* 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, *27.06.2019*

Absolvent *Ruxandra-Ioana RĂDUCANU*



(semnătura în original)

CUPRINS

LISTĂ DE FIGURI

LISTĂ DE TABELE

LISTĂ DE ACRONIME

A

ABS-PC – Acrylonitrile Butanide Stryrene Polycarbonate

C

CC – Creative Commons

CNN – Convolutional Neural Network

CPU – Central Processing Unit

D

DFT – Direct Fourier Transform

DNN – Deep Neural Network

F

FIR - Finite Impulse Response

FSR – Force Sensitive Resistors

H

HMM – Hidden Markov Model

HTTP – Hyper Text Transfer Protocol

I

IP – Internet Protocol

L

LED – Light Emitting Diode

M

MFCC – Mel Frequency Cepstral Coefficients

MLP – Multilayer Perceptron

P

PA-66 – PolyAmide 66

R

RAM – Random Access Memory

RAV – Recunoaștere Automată a Vorbirii

RISC – Reduced Instruction Set Computer

S

SDHC – Secure Digital High Capacity

SRV – Sistem de Recunoaștere a Vorbirii

T

TCP – Transmission Control Protocol

U

USB – Universal Serial Bus

CAPITOLUL 1
INTRODUCERE

1.1. MOTIVAȚIA TEMEI

Roboții umanoizi au devenit un subiect extrem de dezbătut în ultimii ani. Părerile oamenilor pot fi destul de împărțite în ceea ce îi privește, unele persoane privindu-i cu teamă, altele cu fascinație. Cu toate acestea, nimeni nu poate nega utilitatea acestora. Roboții pot fi folosiți cu succes în mediul industrial și se pot dovedi a fi indispensabili în zonele cu grad mare de risc.

Roboții umanoizi pot fi utilizați în scopuri care țin de îmbunătățirea calității vieții oamenilor. Fie că ne referim la utilizarea lor în domeniul de divertisment sau chiar medical, aceștia își dovedesc eficiența în situații diverse. Robotul umanoid NAO este utilizat în ședințele de terapie pentru copiii cu probleme de autism. Aceștia relaționează foarte bine cu robotul și urmează mișcările pe care acesta le execută, astfel încât poate fi utilizat pentru a-i învăța pe copii diverse jocuri și coregrafii. Ținând cont de acest lucru, este de dorit ca robotul să prezinte o autonomie cât mai mare.

Una dintre principalele forme de interacțiune ale oamenilor, atât între ei, cât și cu diverse mașini, o reprezintă interacțiunea prin intermediul vocii. Aceasta, pentru mine, reprezintă cea mai interesantă formă de interacțiune deoarece, în acest mod pot transmise informații diverse. Pe lângă informația concretă, verbalizată, mesajul vocal poate îngloba informații despre starea emoțională și chiar fizică a emițătorului.

Prin îmbinarea modelului de recunoaștere vocală, cu avantajele pe care le prezintă robotul NAO, sistemul pe care urmează să-l prezint pe parcursul acestei lucrări își poate găsi utilitatea în multe situații.

1.2. OBIECTIVELE LUCRĂRII

Scopul principal al lucrării de față a constat în dezvoltarea unui sistem de interacțiune cu robotul umanoid NAO, prin intermediul vorbirii. Sistemul îi permite robotului să recunoască un număr limitat de cuvinte, drept comenzi verbale transmise de către utilizator și să genereze un răspuns în conformitate cu acestea.

Pașii urmăriți pentru obținerea obiectivului principal, au constat în:

- Antrenarea unei rețele neurale profunde pentru recunoașterea vorbirii
- Testarea performanțelor modelului
- Captarea semnalului vocal, transmis de către utilizator, într-un fișier audio de tip ".wav", folosind resursele disponibile pe robot
- Crearea unei conexiuni de tip client-server între robot și laptopul pe care se află modelul antrenat
- Programarea robotului pentru a genera un răspuns motric sau vocal în urma transmiterii comenzilor de către utilizator
- Pornirea sistemului prin simpla atingere a unuia dintre butoanele robotului, astfel încât utilizatorul să interacționeze strict cu robotul NAO.

1.3. REZUMAT

Lucrarea de față cuprinde 6 capitole, care au ca scop prezentarea părții teoretice, dar și a părții practice, necesare pentru a realiza sistemul. Capitolul 2 prezintă, pe scurt, caracteristicile robotului umanoid NAO, precum și resursele pe care acesta le pune la dispoziția utilizatorilor săi. Capitolul 3 prezintă o parte dintre noțiunile de bază referitoare la funcționarea rețelelor neurale artificiale. Capitolul 4 prezintă, succint, arhitectura unui sistem de recunoaștere vocală, precum și câteva principii ale modelării semnalului vocal. În capitolul 5 sunt descrișii pașii care au fost urmăriți în realizarea proiectului de față, precum și rezultatele obținute în ceea ce privește sistemul de recunoaștere vocală. Capitolul final este destinat concluziilor pe care le-am putut sustrage în urma finalizării proiectului.

CAPITOLUL 2
ROBOTUL UMANOID NAO

2.1. ROBOȚII UMANOIZI

Roboții sunt sisteme automatizate, care pot acționa pe baza unui program de lucru stabilit, sau care pot reacționa ca urmare a anumitor influențe exterioare, dând impresia executării acțiunilor omenești[1]. Roboții umanoizi prezintă o particularitate în ceea ce privește aspectul lor, acesta amintind de conformația umană. În general, ei pot fi compuși din cap, trunchi, două mâini și două picioare. În plus, pot prezenta anumite caracteristici care pot aminti de ochii și urechile umane, degete ș.a.m.d. Roboții pot fi autonomi sau semi-autonomi și pot fi utilizați într-o gamă variată de aplicații, de la roboți industriali, la roboți utilizați în diverse forme de terapie.

2.2. UTILIZAREA ROBOȚILOR

În ciuda faptului că oamenii pot fi, în continuare, destul de sceptici în ceea ce privește crearea și utilizarea roboților, un prim exemplu al beneficiilor pe care aceștia le aduc vieții umane este acela că pot fi folosiți în medii și zone periculoase și pot îndeplini sarcini cu grad mare de risc.

Roboții pot fi utilizați în domenii de divertisment, un exemplu fiind TOPIO (TOPSY PING PONG PLAYING ROBOT), care a fost creat cu scopul de a juca tenis de masă împotriva adversarilor umani. De asemenea, aceștia se dovedesc a fi eficienți și în domeniul economic, fiind capabili să interacționeze în mod plăcut cu diverși clienți.

În domeniul medical, roboții sunt folosiți cu succes de către organizațiile care tratează persoanele cu probleme de autism. Aceștia pot fi utilizați și ca motivație pentru ca oamenii să practice diverse activități benefice sănătății lor, precum sport și yoga.

Asemenea oamenilor, aceste mașini trebuie învățate cum pot realiza toate activitățile menționate mai sus. Pe măsură ce este mărit gradul de autonomie al robotului, acesta devine capabil să execute

tot mai multe acțiuni, fără a fi necesară intervenția factorului uman. Din acest motiv, interesul pentru mărirea autonomiei roboților este din ce în ce mai mare.

2.3. NAO

Figura 2.3.1 : Robotul umanoid NAO [2]

2.3.1. ASPECTE GENERALE

Robotul umanoid NAO a fost lansat în anul 2008, fiind primul robot conceput de către firma ALDEBARAN. Acesta este unul dintre cei mai populari roboți umanoizi din lume, fiind vândut în peste 70 de țări. NAO a fost creat cu scopul de a fi utilizat în cercetare și procese didactice, reprezentând o unealtă ajutătoare pentru elevii care încearcă să numere, să învețe diverse coregrafii, sau chiar să descopere cum se programează un robot. Datorită înfățișării sale prietenoase, acesta poate crea legături empatice, atât cu elevii, cât și cu utilizatorii de toate vârstele. Din anul 2008 până în prezent au fost lansate mai multe variante ale robotului, ajungând în prezent la versiunea a șasea, fiecare generație aducând îmbunătățiri față de modelul precursor.

NAO poate fi personalizat în funcție de tipul aplicațiilor pentru care se dorește a fi utilizat. Acesta dispune de resurse hardware, precum diverși senzori și motoare, care pot fi manevrate de către utilizatori. De asemenea, dispune de un software ușor de programat, care realizează conexiunea între senzori și motoare.

2.3.2. DETALII TEHNICE ȘI RESURSE DISPONIBILE

DIMENSIUNI

Înălțimea lui NAO este de 574 mm, lățimea de 275 mm, grosimea de 311 mm și cântărește 5.4 kg, ceea ce îl face ușor de manipulat. Materialul din care este confecționat este ABS-PC și PA-66, acestea oferindu-i flexibilitate și protecție termică. Dimensiunile și capacitatea de a se mișca la 25 de grade, îi permit robotului să execute mișcări fără a-și pierde echilibrul și să recunoască pozițiile în care se află.

PLACA DE BAZĂ

Placa de bază este compusă din două procesoare principale:

- Primul procesor este un procesor Intel x68, ATOM Z530 și este plasat la nivelul capului. Modelul are un singur nucleu și este preferat pentru dispozitivele mobile, datorită consumului redus de energie. Viteza de ceas este de 1.6 GHz, setul de instrucțiuni fiind pe 32 de biți. Acesta are o memorie RAM de 1 GB și o memorie cache de 512 KB. Memoria Flash este de 2 GB, putând fi inserat un card Micro SDHC de maximum 8 GB. Arhitectura acestui procesor îi permite să execute 2 instrucțiuni într-un ciclu de ceas.

- Cel de-al doilea procesor este de tip ARM7TDMI și este localizat la nivelul trunchiului. Acesta este de tip RISC, setul de instrucțiuni fiind pe 32 b. Acesta oferă performanțe ridicate, având un consum mic de putere. Rolul acestuia este de a controla actuatorii și, deci, de a mișca robotul. Comunicarea între procesor și microcontrolerele locale aflate la nivelul modulelor actuatoare este realizată cu ajutorul a două magistrale RS-485, la o viteză de 460Kb/s. Aceste microcontrolere sunt de tipul Microchip ds-PIC pe 16b. [3]

BATERIE

Robotul dispune de o baterie Lithium-Ion, care are o capacitate nominală de 2.25Ah, oferindu-i acestuia o autonomie de până la 90 de minute, în funcție de intensitatea aplicațiilor la care este supus robotul. Tensiunea maximă de încărcare este de 25.2V, iar energia de 48.6Wh. Durata de încărcare a bateriei este de aproximativ 3 ore, însă robotul poate fi utilizat și atunci când este conectat la priză.

CONECTIVITATE

NAO poate fi conectat la internet prin protocoalele de comunicație Ethernet și Wi-Fi. Conexiunea Ethernet se realizează cu ajutorul mufei RJ-45, localizată la nivelul capului și este utilizată în special pentru a seta conexiunea prin Wi-Fi a robotului. Aceasta este compatibilă cu trei tipuri de adaptări, 10/100/1000 base T, având viteze de 10Mb/s, 100Mb/s sau 1000Mb/s. Robotul dispune și

de un port USB, aflat, de asemenea, la nivelul capului. Acesta asigură, printre altele, conexiunea la dispozitive precum Kinect, Asus 3D Sensor sau Arduino.

INTERACȚIUNE AUDIO

Robotul dispune de un sistem stereo de difuzare, alcătuit din două difuzoare aflate pe lateralele capului, amintind de cele două urechi umane. De asemenea, este dotat cu 4 microfoane, aflate tot la nivelul capului, având o sensibilitate de 20 mV/Pa +/-3dB la 1 KHz și o gamă de frecvențe între 300Hz și 8KHz.

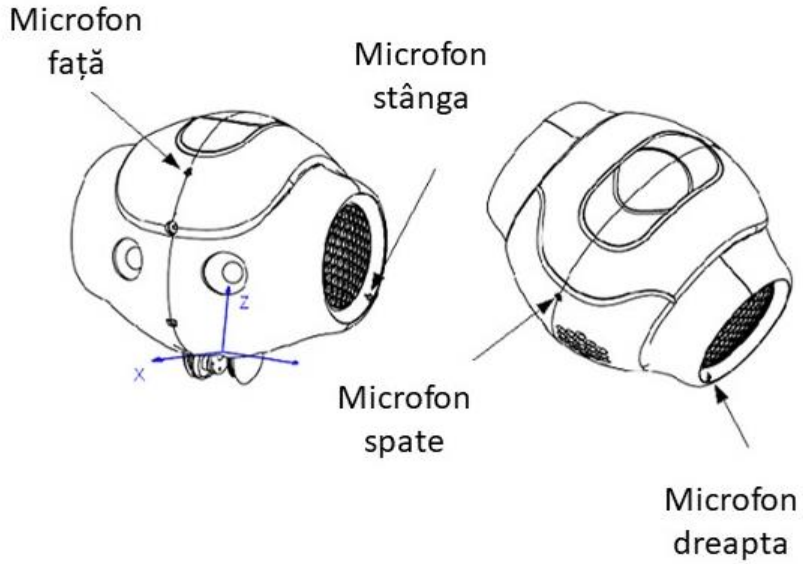


Figura 2.3.2.1 : Poziționarea microfoanelor [3]

INTERACȚIUNE VIDEO

NAO este prevăzut cu două camere video de tip SOC Image Sensor MT9M114, aflate în partea din față a capului robotului. Acestea sunt capabile să capteze imagini cu o rezoluție de 1280x960, la 30 de cadre pe secundă. Rezoluția celor două camere este de 1.22 Mp, cu format optic de 1/6 inch. Mărimea unui pixel este de 1.9 µm * 1.9 µm, iar gama dinamică este de 70 dB. În ceea ce privește câmpul vizual, acesta este de 70.6 ºDFOV, câmpul vizual pe orizontală având 60.9 ºDFOV, iar cel pe verticală având 47.6 ºDFOV.

Rolul principal al celor două camere video este acela de a îi permite robotului să identifice diversele obiecte aflate în jurul său. De asemenea, camera de jos este utilizată pentru detectarea spațiului în care robotul își execută mișcările.

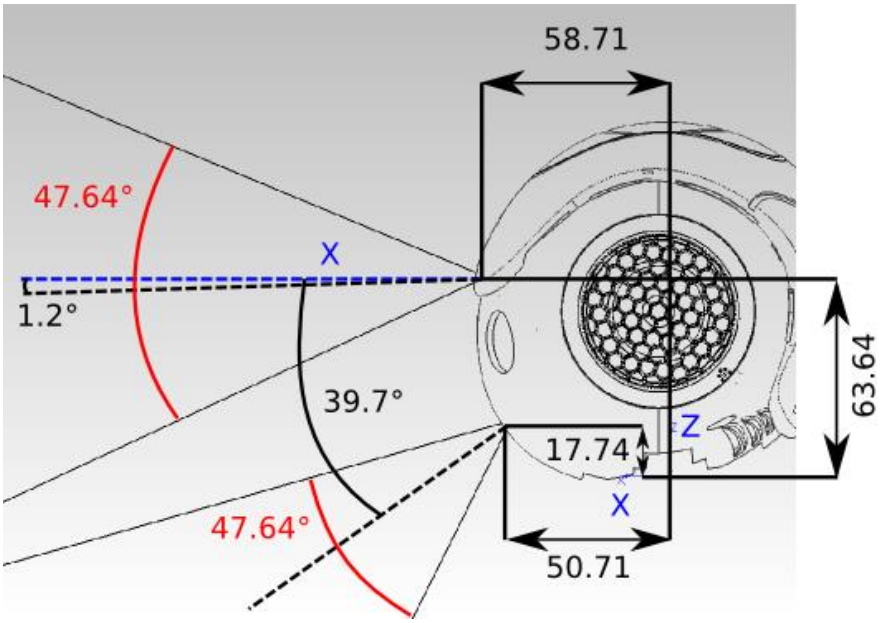


Figura 2.3.2.2 : Apertura camerei de sus [3]

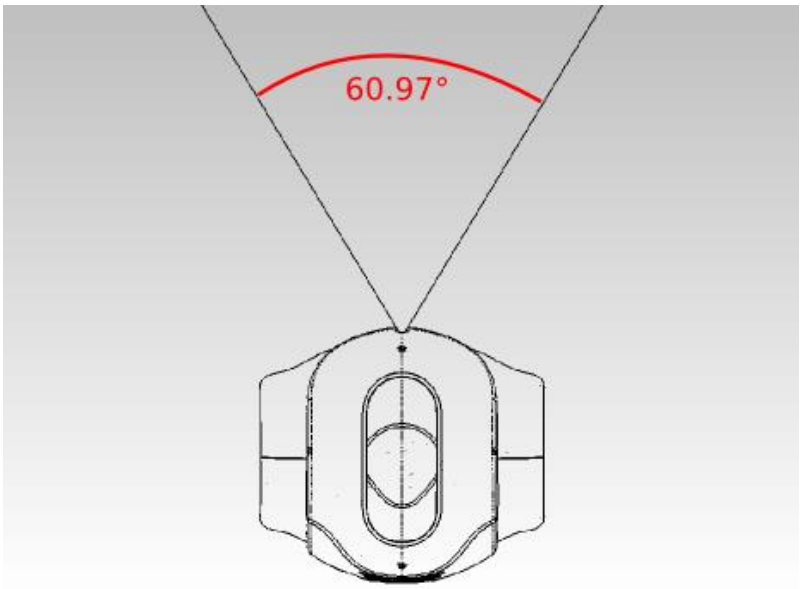


Figura 2.3.2.3 : Apertura camerei de jos [3]

LED-uri

NAO este prevăzut cu 51 de LED-uri, care au rolul de a oferi informații suplimentare legate de modul de funcționare al robotului. La nivelul ochilor există 16 LED-uri, care devin roși atunci când nivelul bateriei robotului devine scăzut. De asemenea, acestea se sting și se aprind la un anumit interval de timp, oferind impresia că robotul ar clipi. Acest artificiu contribuie la alura umană pe care NAO o posedă.

Cele 51 de LED-uri menționate anterior sunt poziționate în diverse locuri ale corpului robotului. Figura 2.3.2.2 împreună cu tabelul 2.3.2.1 prezintă localizarea acestora.

Figura 2.3.2.4 : Localizarea LED-urilor [3]

Secțiune	Localizare	Număr de LED-uri
A	CAP	12
B	URECHI	2x10
C	OCHI	2x8
D	PIEPT	1
E	PICIOARE	2x1

Tabelul 2.3.2.1 : Localizarea și numărul LED-urilor

LED-urile de la nivelul capului și al urechilor prezintă 16 nivele de albastru, iar cele de la nivelul ochilor, al pieptului și al picioarelor sunt LED-uri pe tot spectrul RVA. Intensitatea luminoasă a acestora poate fi variată de la 0 la 100%.

FSR

Senzorii rezistivi FSR măsoară valoarea rezistenței, atunci când este modificată presiunea aplicată pe aceștia. NAO dispune de 8 astfel de senzori, poziționați câte 4 pe fiecare picior, așa cum este sugerat în figura 2.3.1.3. Aceștia funcționează pentru o gamă de valori de la 0 N la 25 N.

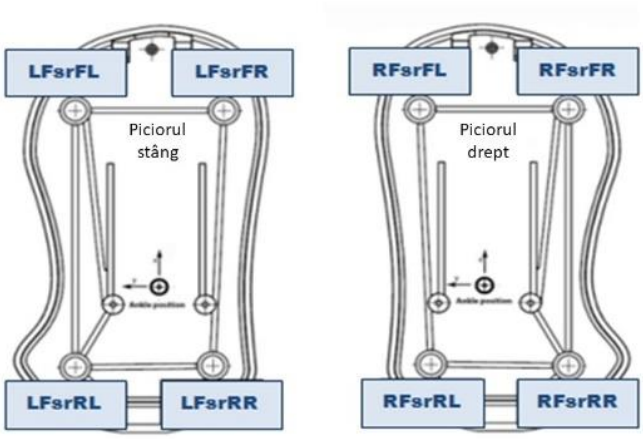


Figura 2.3.2.5 : Poziționarea senzorilor FSR [3]

UNITATEA DE INERȚIE

Aceasta este compusă din girometre și un accelerometru, fiind poziționată la nivelul trunchiului robotului, împreună cu propriul procesor. Girometrul este răspunzător pentru orientarea robotului, are o precizie de 5% și o viteză unghiulară de 500 °/s. Accelerometrul are o precizie de 1% și reprezintă punctul de referință pentru modul static. Când este detectată o mișcare, unghiul aferent este calculat utilizând girometrele.

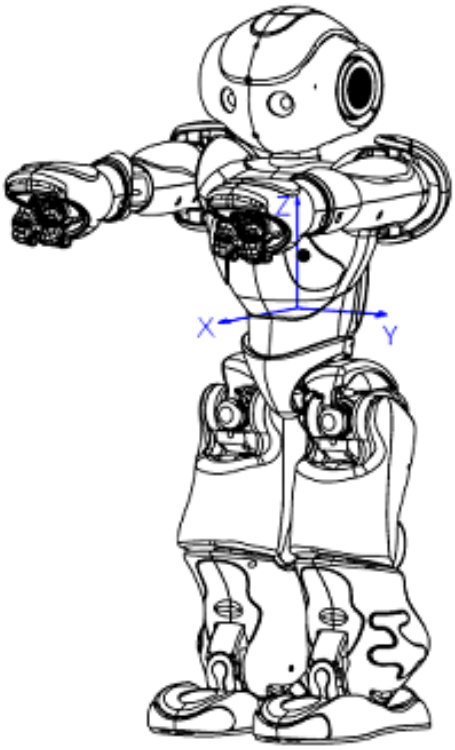


Figura 2.3.2.6 : Unitatea de inerție [3]

SONARE

NAO este prevăzut cu doi senzori ultrasonici (sonare), care îi permit acestuia să estimeze distanța până la obstacolele aflate în mediul său de deplasare. Acestea sunt alcătuite din două emițătoare și două receptoare, au o rază de detectare de 0.2m - 0.8m, o rezoluție de 1cm-4cm și funcționează la frecvențe de 40 kHz. Pentru valori mai mici de 20cm, nu se pot furniza informații legate de distanță, robotul înțelege doar că există un obiect în vecinătatea sa. Pentru valori mai mari de 80 cm, valoarea returnată este o estimare.

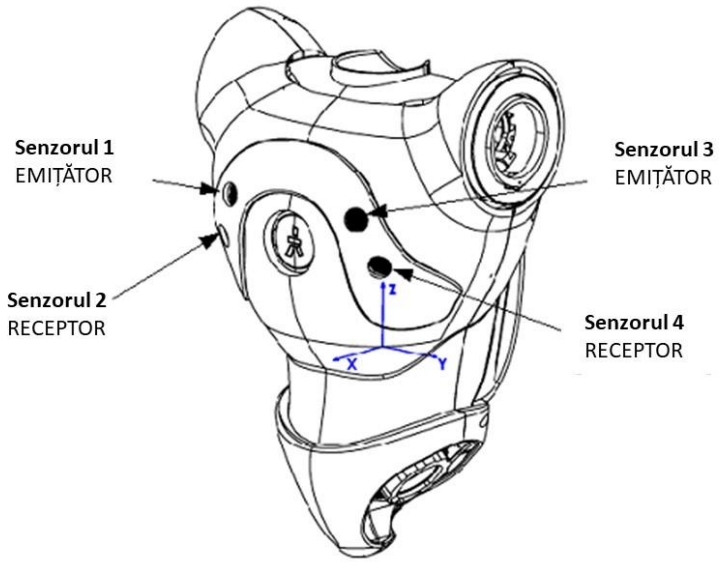


Figura 2.3.2.7 : Localizarea senzorilor ultrasonici [3]

SENZORI TACTILI ȘI DE CONTACT

Robotul dispune de trei senzori tactili, localizați la nivelul capului, care pot fi programați astfel încât, la atingerea lor, NAO să execute diverse acțiuni. Butonul de pe pieptul robotului este folosit pentru pornirea și oprirea acestuia. Cei trei senzori de la nivelul mâinilor, precum și cele două bare de la nivelul picioarelor au rol de protecție împotriva loviturilor.

MOTOARE

NAO dispune de 25 de motoare, aflate la nivelul articulațiilor, care îi oferă acestuia mobilitate și ajută la reproducerea cât mai fidelă a mișcărilor umane.

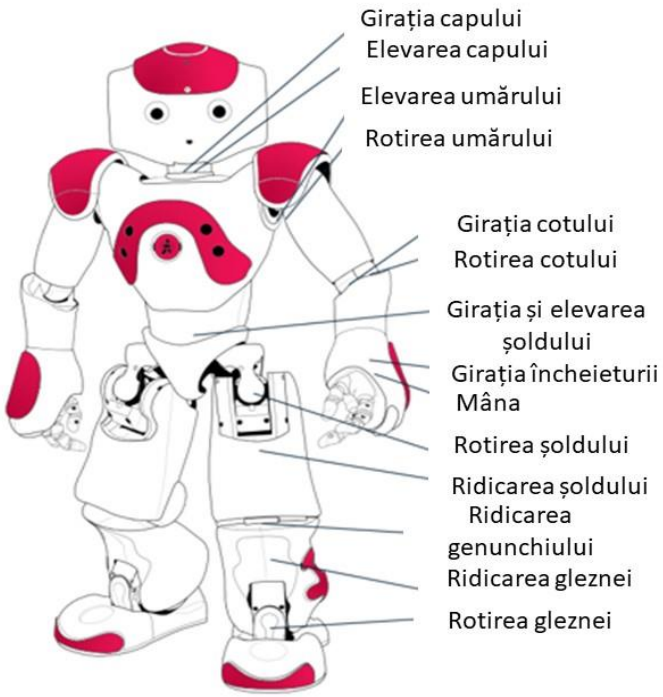


Figura 2.3.2.8 : Poziționarea și rolul articulațiilor [3]

Robotul dispune de 25 de grade de libertate, 11 pentru partea inferioară (pelvis și picioare) și 14 pentru partea superioară (cap, mâini și partea superioară a trunchiului). Acestea sunt responsabile pentru determinarea stării sistemului. Dispunerea gradelor de libertate ale robotului este realizată în felul următor:

- 2 grade de libertate la nivelul capului;
- 5 grade de libertate la nivelul fiecărui braț;
- 1 grad de libertate la nivelul pelvisului;
- 5 grade de libertate la nivelul fiecărui picior;
- 1 grad de libertate la nivelul fiecărei mâini.

Actuatorii prezintă perii de carbon, având un preț redus și o viteză care poate fi setată de către utilizator.

NAO deține 3 tipuri de motoare, fiecare având caracteristici diferite, așa cum este sugerat în tabelul 2.3.2.2. Acestea prezintă avantaje proprii, motiv pentru care sunt utilizate în zone diverse.

	Motor de tip 1	Motor de tip 2	Motor de tip 3
Model	22NT82213P	17N88208E	16GT83210E
Viteză	8 330 rpm ±10%	8 400 rpm ±12%	10 700 rpm ±10%
Cuplu	68 mNm ±8%	9.4 mNm ±8%	14.3 mNm ±8%
Cuplu nominal	16.1 mNm	4.9 mNm	6.2 mNm

Tabelul 2.3.2.2 : Detalii tehnice ale motoarelor [3]

CAPITOLUL 3
REȚELE NEURALE ARTIFICIALE

3.1. MODELUL BIOLOGIC

Creierul uman cuprinde aproximativ 10^{11} neuroni și 10^{15} sinapse, care rețin toate funcțiile acestuia. Modul de funcționare al rețelei este determinat de plasarea neuronilor și de calitatea legăturilor dintre aceștia. Rețeaua neuronală are proprietatea de a se modifica pe parcursul vieții unui om, prin procesul de învățare, care poate determina apariția, dispariția sau modificarea din punct de vedere chimic a conexiunilor dintre neuroni.

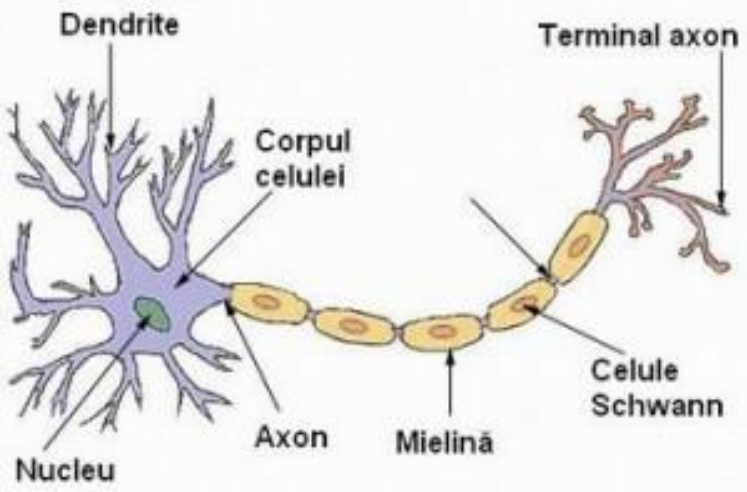


Figura 3.1.1. : Structura neuronului biologic [4]

Neuronul biologic primește semnalele de intrare prin dendrite. Acestea generează diferențe de potențial pe membrana celulară, potențial care este transmis către axon. De-a lungul axonului este transmis un tren de impulsuri electrice, care, ajunse la terminațiile axonului, generează transferul de

informații către dendritele altui neuron. Legătura dintre terminațiile axon și dendrite formează o sinapsă.

3.2. PRINCIPII DE BAZĂ

O rețea neurală artificială este un sistem adaptiv, care are capacitatea de a rezolva probleme necunoscute, dacă a fost antrenată în prealabil în situații similare. Rețelele neurale artificiale sunt inspirate din modelul rețelelor neuronale biologice, care sunt răspunzătoare pentru modul în care omul învață și rezolvă anumite probleme. Sistemul este compus din strat de intrare, straturi ascunse și strat de ieșire.

Rețelele neurale sunt formate din mai multe unități funcționale interconectate, având parametrii ajustabili. Aceste unități reprezintă un model simplificat al neuronului biologic și au rolul de a prelucra anumite date de intrare.

Primul model matematic care reușea să imite într-o oarecare măsură modelul biologic neuronal, a fost conceput în anul 1943 de către cercetătorii Warren S. McCulloch și Walter Pitts. După cum am menționat anterior, modelul biologic conține sinapse, care pot fi excitatorii sau inhibitorii. Pentru a determina caracterul conexiunilor în cazul rețelelor neurale artificiale este nevoie de introducerea parametrilor numiți ponderi. Astfel, dacă ponderea este pozitivă, conexiunea este excitatoare, respectiv dacă ponderea este negativă, conexiunea este inhibatoare.

Neuronului McCulloch-Pitts îi este asociată o valoare de prag, acesta fiind activat doar dacă intrările au o valoare mai mare decât valoarea pragului. Pentru valori mai mici decât valoarea pragului, neuronul rămâne inactiv (inhibat).

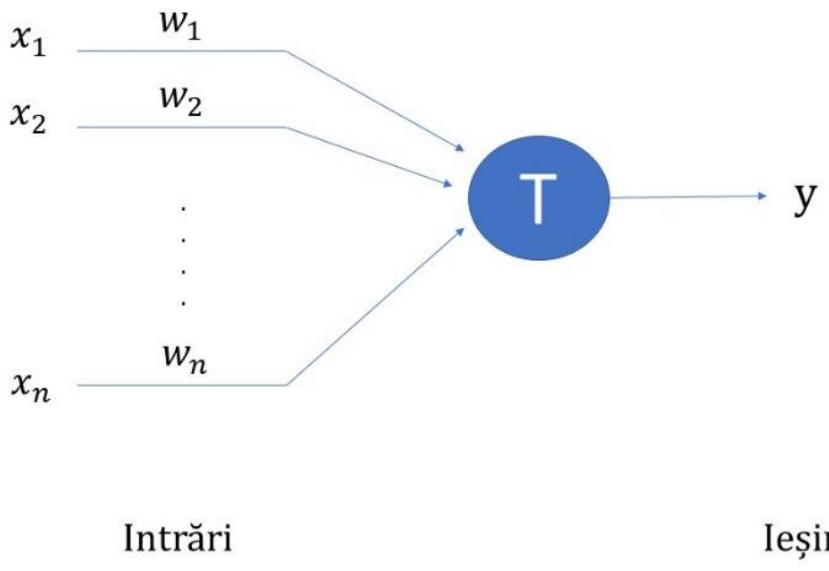


Figura 3.2.1 : Arhitectura neuronului McCulloch-Pitts

În figura 3.2.1, x_1, x_2,..., x_n reprezintă componentele vectorului de intrare, w_1, w_2,..., w_n ponderile, iar y reprezintă ieșirea. T reprezintă pragul menționat anterior. Formula care determină dacă neuronul este activat este următoarea:

$$\sum_{i=1}^{n} w_i \, x_i \geq T \; ; \; (1)$$

$$w_i \epsilon \{-1; 1\}$$

$$x_i \, \epsilon \{0; 1\}$$

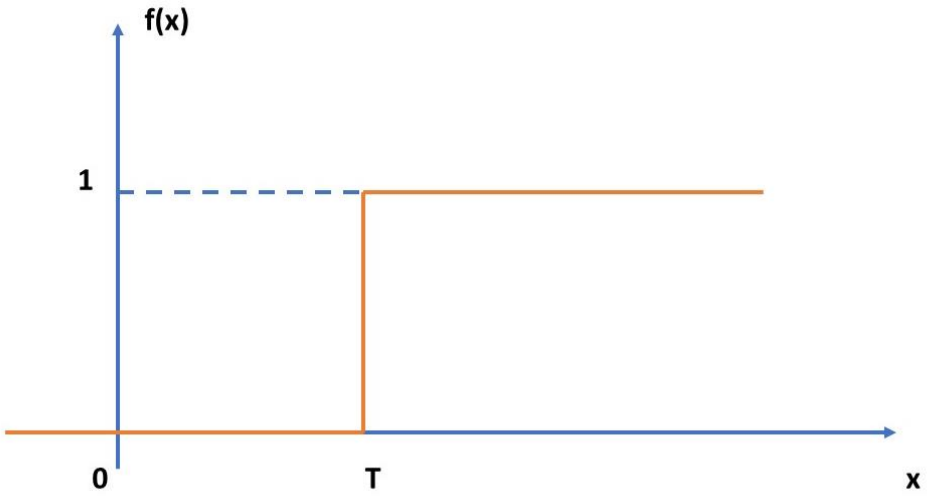


Figura 3.2.2 : Funcția de activare a modelului McCulloch-Pitts

Practic, neuronul se comportă ca un sistem de decizie. O analogie cu modelul creierului uman ar fi aceea că, atunci când trebuie luată o decizie, se iau în considerare toți factorii (intrările) și se cântărește importanța fiecăruia dintre ei (ponderile din figură). În funcție de cât de mult ne dorim să facem un compromis (pragul), putem lua o decizie (ieșirea).

3.2.1. PERCEPTRONUL

Perceptronul a fost conceput în anul 1957 de către F. Rosenblatt. Acesta reprezintă un algoritm pentru învățarea supervizată și are proprietatea de a clasifica valorile de intrare în două clase, care corespund valorilor de ieșire 0 și 1 (clasificare binară).

Învățarea supervizată utilizează un set de antrenare, format din perechi de intrare-ieșire, traduse sub formă de text. Vectorii de intrare sunt aplicați secvențial, iar valorile de ieșire ale rețelei neurale sunt comparate cu valorile dorite, din setul de învățare. Dacă răspunsul obținut nu este cel așteptat, se reglează ponderile astfel încât probabilitatea ca răspunsul să fie eronat, să scadă. Probabilitatea scade prin repetări succesive, până când vectorul este clasificat corect. Metoda de învățare prin penalizare este acea metodă prin care sistemul este penalizat atunci când generează un răspuns

greșit, astfel încât, la următoarea încercare, răspunsul să fie unul corect, sau mai aproape de cel corect.

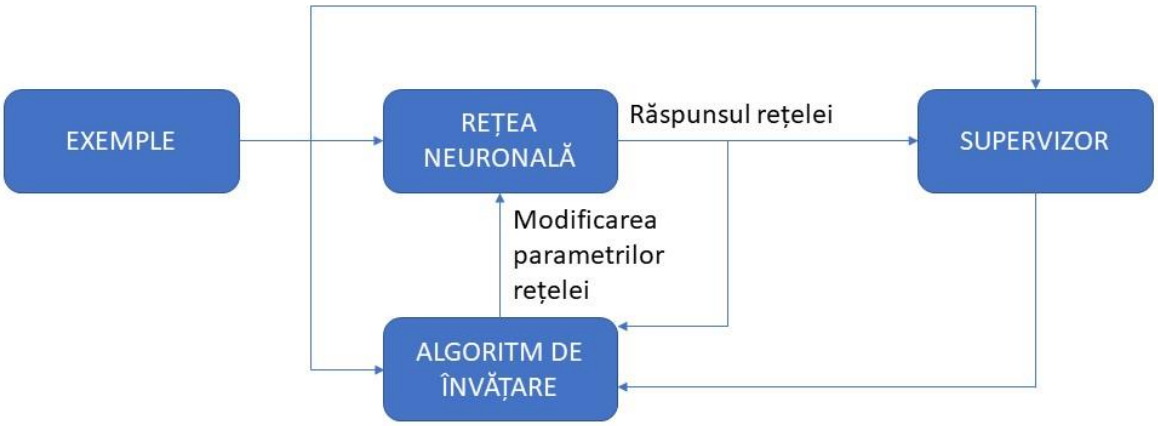


Figura 3.2.1.1 : Principiul învățării supervizate [5]

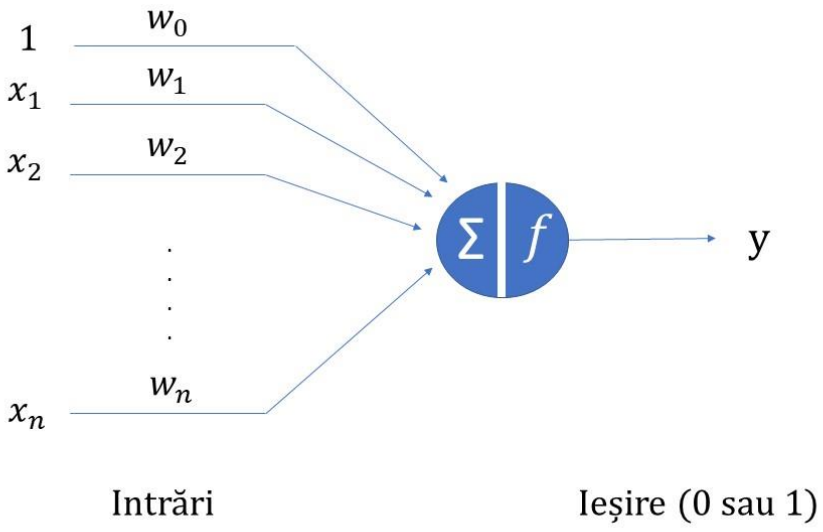


Figura 3.2.1.2 : Arhitectura perceptronului

În figura 3.2.1.1, intrările sunt notate cu x, ponderile cu w, y este ieșirea neuronului, iar f reprezintă funcție de activare. După cum se poate observa, perceptronul conține un singur neuron, ceea ce limitează clasificarea vectorilor la două clase. Neuronul reprezintă de această dată o funcție de activare f(k), unde k=$\sum_{i=1}^{n} w_i\, x_i$. Pentru valori ale lui k mai mici decât 0, f(k) ia valoarea 0, iar pentru valori ale lui k mai mari decât 0, f(k) ia valoarea 1. Ieșirea y a rețelei se calculează astfel:

$$y = f(w_0 + \sum_{i=1}^{n} w_i\, x_i); \quad (2)$$

$$x_i \in \mathrm{R},\ w_i \in \mathrm{R}$$

w_0 poartă numele de parțilaitate (bias în engleză). Acesta are rolul de a se asigura că neuronul se poate activa, chiar dacă toate intrările sunt nule.

Figura 3.2.1.3 : Funcția de activare a perceptronului simplu

După cum se poate deduce din figura 3.2.1.3, $f(x) = \begin{cases} 0, & dacă\ x < 0 \\ 1, & dacă\ x \geq 0 \end{cases}$

Pragul de activare T va lua valoarea $-w_0$.

TEOREMA DE CONVERGENȚĂ

Utilizând acest algoritm se poate demonstra faptul că, pentru două seturi de vectori liniari separabili, algoritmul perceptron converge către o suprafață de decizie de forma unui hiperplan între cele două clase [6]. Practic, în acest caz, algoritmul tinde să găsească o soluție stabilă după efectuarea unui număr limitat de pași.

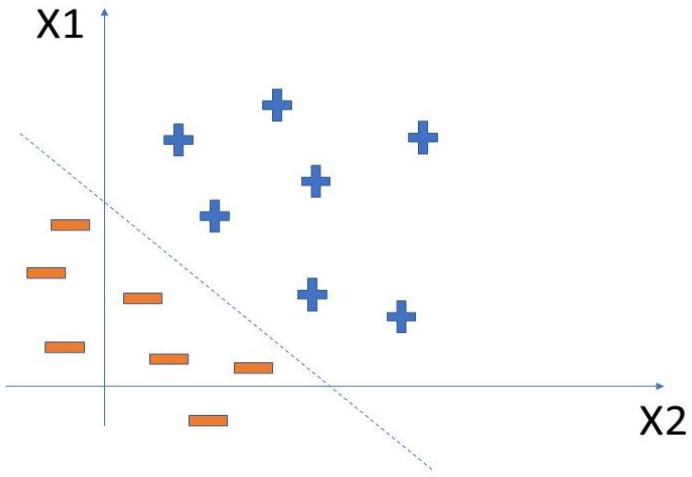


Figura 3.2.1.4 : Interpretarea geometrică a teoremei de convergență

În figura 3.2.1.4, linia punctată reprezintă suprafața de separație între cele două clase și este hiperplanul cu ecuația $w_0 + \sum_{i=1}^{n} w_i\, x_i = 0$.

ALGORITMUL DE ANTRENARE AL PERCEPTRONULUI SIMPLU

- Se inițializează ponderile la momentul t=0, cu valori aleatorii
- Se determină ieșirea $y_j = f(W^T X_j)$; j=1...N, unde N reprezintă numărul de vectori de intrare. W^T se referă la operația transpusă, care transformă vectorii linie în vectori coloană.
- Se compară ieșirea obținută y_j cu ieșirea dorită d_j și se determină eroarea $e_j(t) = d_j - y_j(t)$
- Se ajustează ponderile în vederea micșorării erorii $\Delta w_i(t) = \eta e_j(t)\, x_{ij}$, unde η reprezintă rata de învățare

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

- Se repetă pașii, exceptând primul pas, până când toate ieșirile sunt corecte (dorite)

3.2.2. REȚEAUA ADALINE

ADALINE (ADAptive Linear NEuron) a fost conceput de către Bernard Widrow și Marcian E. Hoff în anul 1960. Aceasta a fost prima rețea neurală utilizată pentru eliminarea ecoului în timp real. Din punct de vedere al structurii și al testării, aceasta nu prezintă nicio deosebire față de perceptron. Inovația constă în faptul că aceasta este antrenată cu ajutorul algoritmului LMS (least mean square).

Algoritmul LMS introduce un parametru numit funcție cost, care reprezintă eroarea pătratică medie dintre ieșirea dorită și ieșirea reală și care determină practic corectitudinea funcționării neuronului.

ALGORITMUL ADALINE

- Se inițializează ponderile la momentul t=0 cu valori aleatorii
- Se determină ieșirea rețelei $y_j = f(W^T X_j)$; j=1...N, unde N reprezintă numărul de vectori de intrare
- Se calculează eroarea folosind formula $E(W) = \frac{1}{N} \Sigma E_j(W)$; j=1...N

$$E_j(W) = \frac{1}{2}(d_j - y_j)^2 \quad (3)$$

- Se modifică ponderile în vederea micșorării erorii:

$$w_i(t+1) = w_i(t) + \Delta w_i(t) \quad (4)$$

- Se repetă pașii, exceptând primul pas, până când toate ieșirile sunt corecte (dorite)

3.2.3. MLP

Rețeaua MLP este o rețea de tip unidirecțional, care permite semnalului să se deplaseze de la intrare către ieșire. Aceste tipuri de rețele conțin un număr mai mare de neuroni, dispuși în straturi ascunse.

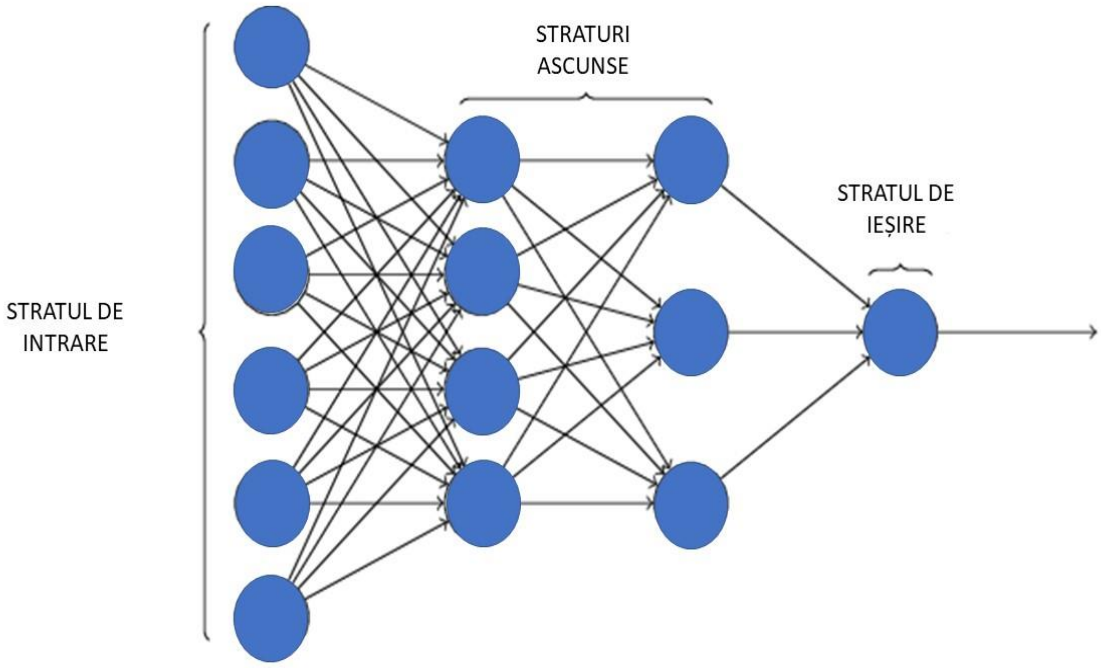


Figura 3.2.3.1 : Arhitectura unei rețele cu mai multe straturi

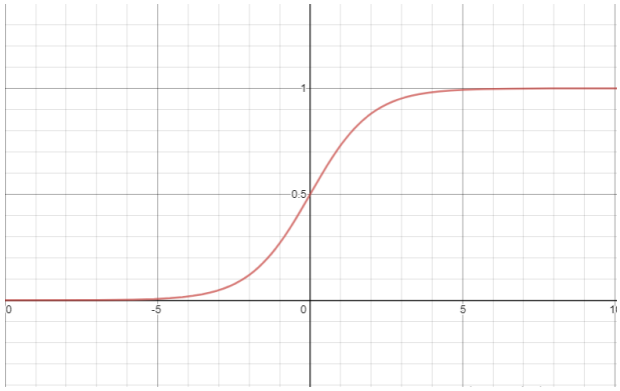
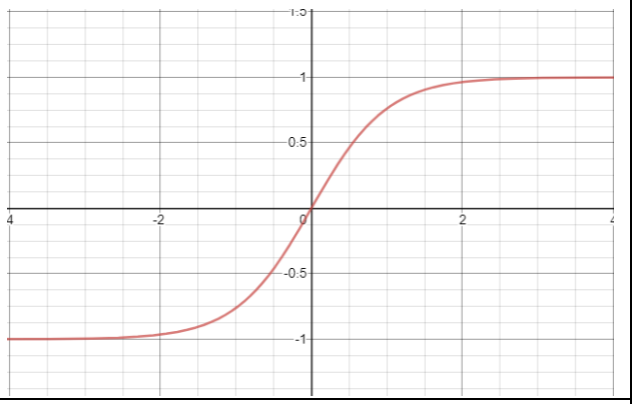
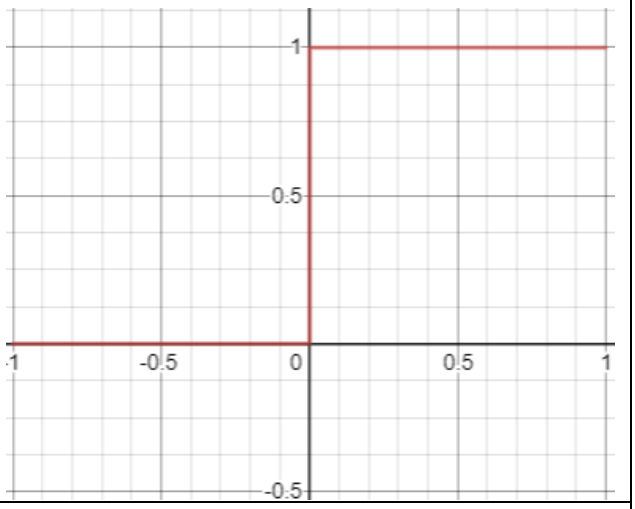
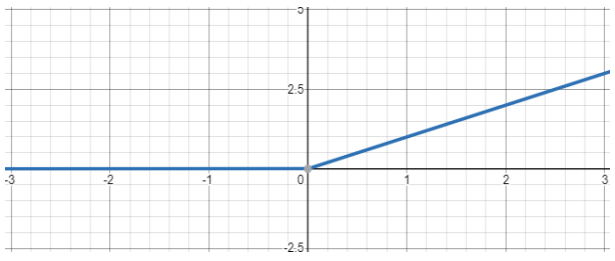
După cum se poate observa în figura de mai sus, rețeaua neurală este compusă dintr-un strat de intrare, un strat de ieșire și mai multe straturi ascunse. Primul strat este indispensabil pentru o rețea neurală deoarece el oferă informații referitoare la mediul exterior, informații fără de care sistemul nu ar putea funcționa. Straturile ascunse pot lipsi din arhitectura rețelei. Acestea sunt responsabile pentru efectuarea calculelor și transferul de informații de la intrare către ieșire. Stratul de ieșire este de asemenea responsabil pentru efectuarea calculelor. Prin intermediul lui se transmite răspunsul rețelei către mediul exterior.

3.2.4. FUNCȚII DE ACTIVARE

Așa cum a fost sugerat în figura 3.2.1, neuronul este format dintr-un sumator și o funcție de activare. Am menționat anterior faptul că, pentru a obține rezultatul dorit în urma obținerii unui rezultat eronat, trebuie modificate ponderile. Problema apare atunci când o modificare mică a ponderilor generează modificări ale parametrilor stratului următor, ceea ce generează o modificare

drastică a funcționării rețelei. Din acest motiv au fost introduse funcțiile de activare, care au rolul de a determina modul în care trebuie modificate ponderile, astfel încât rezultatul să fie cel dorit. Prin modificarea funcțiilor se înțelege variația lor în timp.

Cele mai importante funcții de activare, împreună cu descrierea lor, se regăsesc în tabelul 3.2.4.1.

DENUMIRE	DEFINIȚIE MATEMATICĂ	LIMITE	REPREZENTARE GRAFICĂ
Sigmoidă	$\sigma(x)=\frac{1}{1+e^{-x}}$	[0;1]	
Tangentă hiperbolică	$\tanh(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	[-1;1]	
Treaptă	$f(x)=\begin{cases}0; & x<0\\ 1; & x\geq 0\end{cases}$	[0;1]	
ReLU	f(x)=max(0,x)	[0; ∞)	

Tabelul 3.2.4.1 : Funcții de activare

3.3. ALGORITMUL DE ÎNVĂȚARE PRIN CORECTAREA ERORII

Învățarea prin corectarea erorii se folosește în general pentru rețelele care au la bază principiul învățării supervizate. Principiul pe care se bazează acest tip de învățare este următorul: pentru fiecare set de date de la intrare, rețelei i se oferă, de asemenea, răspunsul dorit (răspunsul considerat a fi cel corect). Acesta a fost notat pe parcursul lucrării cu litera d. Deoarece ieșirea y reală a rețelei diferă de valoarea așteptată d, se introduce o mărime de eroare $e = d - y$, cu ajutorul căreia se determină modul în care trebuie modificate ponderile rețelei, scopul final fiind acela de a reduce eroarea.

Există anumite aplicații pentru care metodele simple de estimare și prelucrare adaptivă devin ineficiente (de exemplu, aplicații în care se prelucrează semnale profund nestaționare etc.). Pentru acestea, o posibilă soluție este reprezentată de tehnica propagării inverse a erorilor (BKP). Prin utilizarea unei funcții de activare neliniare, derivabilă pe tot domeniul de definiție (de exemplu funcția sigmoid) împreună cu un mecanism de repartizare a erorilor în toate nodurile rețelei, aceasta reușește să generalizeze învățarea setului de date de intrare, către un set care deține anumite caracteristici comune. [5]

FUNCȚIA COST

Principiul de funcționare al acestui algoritm se referă la calculul derivatei parțiale a funcție cost C, în raport cu ponderile și parțialitatea rețelei: $\frac{\partial C}{\partial w}$, $\frac{\partial C}{\partial b}$. Funcția cost reprezintă eroarea pătratică medie și se calculează utilizând următoarea formulă:

$$C=\frac{1}{2n}\Sigma||y(x) - a^L(x)||^2 \quad (5)$$

- n reprezintă numărul de seturi de antrenare
- y(x) reprezintă ieșirea pe care o dorim din partea rețelei
- L reprezintă numărul de straturi ale rețelei
- $a^L(x)$ reprezintă vectorul valorilor de ieșire reale ale rețelei, atunci când se introduce x
- Suma este efectuată pentru toate intrările x ale rețelei [7]

În mod evident, ne dorim ca algoritmul să prezinte o acuratețe a predicțiilor cât mai mare. Din acest motiv, funcția cost trebuie minimizată cât mai mult, până la o valoare apropiată de 0. Așa cum putem deduce din formula prezentată anterior, C este întotdeauna pozitiv, ceea ce facilitează găsirea valorii minime a acesteia. De asemenea, C descrește odată cu apropierea valorilor de ieșire reale ale rețelei (a(x)) de valorile dorite. Algoritmul de minimizare al erorii se numește algoritmul gradientului negativ.

GRADIENTUL NEGATIV

Acest algoritm are ca scop găsirea minimului unei funcții. Așa cum am precizat anterior, se dorește obținerea unei valori a erorii cât mai scăzută. Principiul de funcționare al algoritmului este următorul: considerând o funcție definită de un set de parametri, se pornește de la un anumit set de parametri, aleși în mod aleator și se realizează o deplasare, în mod iterativ, către valoarea minimă a funcției. [8]

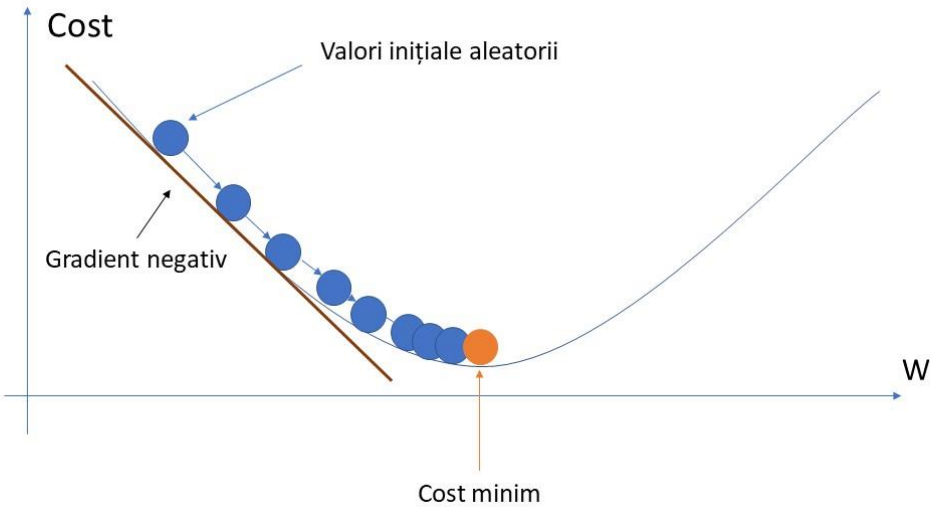


Figura 3.3.1 : Gradient negativ pentru funcția cost C(w)

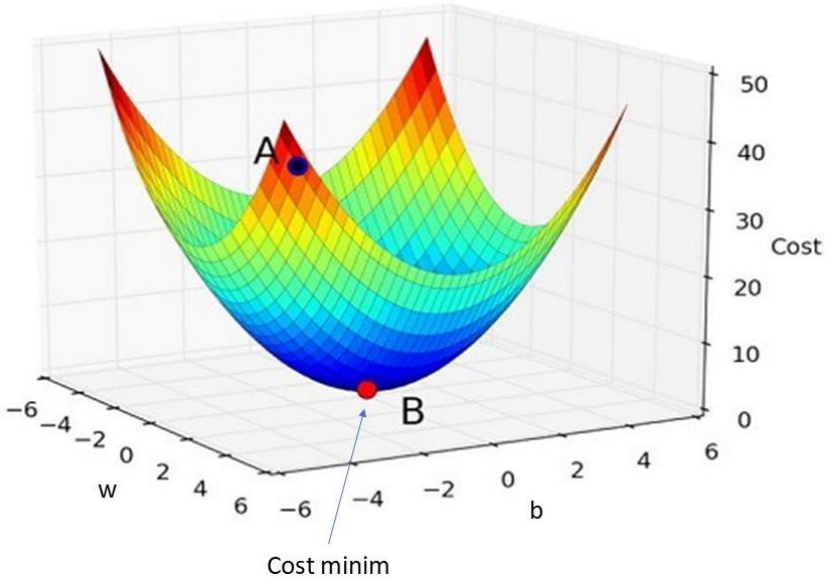


Figura 3.3.2 : Gradient negativ pentru funcția cost C(w,b) [9]

Figura 3.3.1 sugerează graficul funcției cost dacă aceasta este dependentă de o singură variabilă și anume de pondere. Graficul funcției cost în funcție de ponderi și parțialitate este reprezentat în figura 3.3.2.

Așa cum se poate observa în prima figură, gradientul poate fi considerat drept panta funcției. Dacă acesta are o valoare mare, panta este mai abruptă, ceea ce duce la o învățare mai rapidă a modelului.

După cum am menționat anterior și după cum se poate observa și din figura 3.3.1, modificarea ponderilor se face prin efectuarea pașilor în direcția descrescătoare gradientului.

$$w_{i+1} = w_i - \eta \frac{\partial C}{\partial w} = w_i - \eta \frac{1}{n} \Sigma (z(x_i, w, b) - y_i) x_i \quad (6)$$

$$b_{i+1} = b_i - \eta \frac{\partial C}{\partial b} = b_i - \eta \frac{1}{n} \Sigma (z(x_i, w, b) - y_i) \quad (7)$$

Unde $z(x_i, w, b) = wx_i + b$ iar η reprezintă rata de învățare. [10]

RATA DE ÎNVĂȚARE

Aceasta moderează viteza de apropiere de minimul dorit. Alegerea unei rate de învățare nepotrivite ridică două probleme esențiale:

- Dacă rata de învățare este prea mare, putem ajunge foarte aproape de valoarea minimă, dar fără să o atingem niciodată

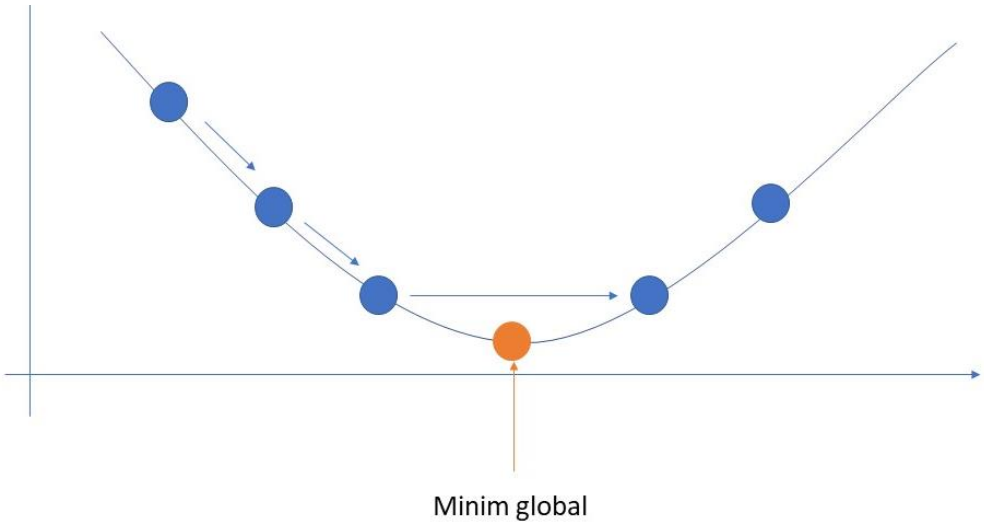


Figura 3.3.3 : Reprezentare grafică a ratei de învățare mare

Evident, o soluție a acestei probleme este scăderea ratei de învățare.

- Dacă rata este prea mică, există riscul de a rămâne blocați la valoarea unui minim local.

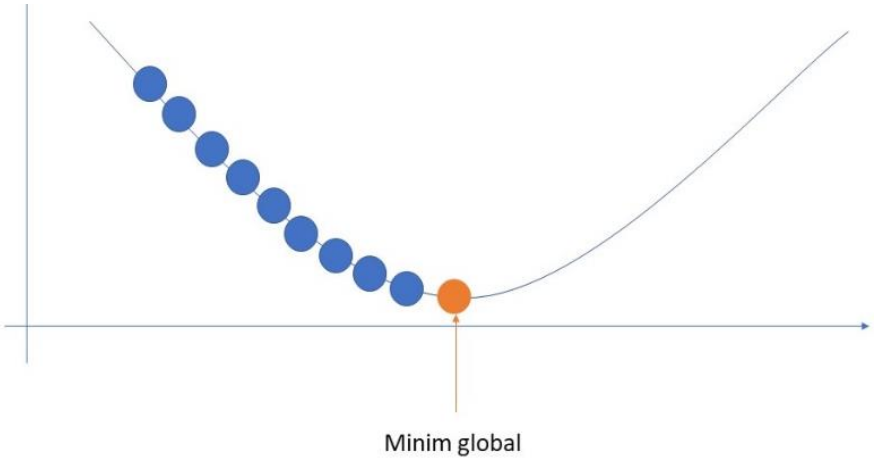


Figura 3.3.4 : Reprezentare grafică a ratei de învățare mică

Soluția acestei probleme poate fi reînceperea pasului de antrenare, pornind de la valori diferite ale ponderilor, sau introducerea unor perturbații aleatoare, care se adaugă ponderilor

ENTROPIA ÎNCRUCIȘATĂ

Pentru rezolvarea problemelor de aproximare, eroarea pătratică medie se dovedește a fi foarte utilă. Algoritmul gradientului prezintă însă anumite limitări, motiv pentru care s-a dezvoltat entropia încrucișată, pornind de la optimizarea funcției cost. Aceasta este o metodă mult mai potrivită atunci când se dorește rezolvarea problemelor de clasificare.

$$\frac{\partial C}{\partial w} = (a - y)\sigma'(z)x \quad (8)$$

$$\frac{\partial C}{\partial b} = (a - y)\sigma'(z) \quad (9)$$

- σ este funcția sigmoid
- $z = \Sigma w_{kj}^{L} a_{k}^{L-1} + b_{j}^{L}$ este intrarea ponderată

Așa cum este sugerat în tabelul 3.2.4.1, funcția σ(z) variază în intervalul [0;1]. Derivata acesteia va tinde, deci, spre 0 atunci când eroarea este mare, ceea ce face ca procesul de învățare să fie mult mai dificil. Apare necesitatea introducerii unei funcții cost diferite, care să nu depindă de acest comportament al funcției și care să grăbească procesul de învățare, chiar și atunci când variația sigmoidei este foarte mică. Vom nota ecuațiile scrise anterior, astfel:

$$\frac{\partial C}{\partial w} = (a_j - y_j)x \quad (10)$$

$$\frac{\partial C}{\partial b} = (a - y) \quad (11)$$

Utilizând regula produsului obținem:

$$\frac{\partial C}{\partial b} = \frac{\partial C}{\partial a}\frac{\partial a}{\partial b} = \frac{\partial C}{\partial a}\sigma'(z) \quad (12)$$

Cunoscând proprietatea $\sigma'(z) = \sigma(z)\big(1 - \sigma(z)\big) = a(1-a)$ a sigmoidei și introducând-o în relația scrisă anterior, obținem relația:

$$\frac{\partial C}{\partial b} = \frac{\partial C}{\partial a}a(1-a) \quad (13)$$

Utilizând relația (12) rezultă:

$$\frac{\partial C}{\partial a} = \frac{a-y}{a(1-a)} \quad (14)$$

Vom integra cu a pe ambele părți și vom obține forma finală a funcției cost, pe un singur neuron

$$C = -[ylna + (1-y)\ln(1-a)] + K \quad (15)$$

Unde K este o constantă.

Forma finală a funcției cost va fi determinată prin calcularea mediei pentru toate valorile din setul de învățare [10]:

$$C = -\frac{1}{n}\Sigma[ylna + (1-y)\ln(1-a)] + K \quad (16)$$

3.4. REȚELE NEURALE CONVOLUȚIONALE

Rețelele Neurale Convoluționale (CNN) reprezintă o clasă a rețelelor neurale profunde DNN, folosită preponderent pentru clasificarea obiectelor din imagini, însă care poate fi utilizată și pentru alte tipuri de aplicații, cum ar fi recunoașterea vorbirii.

Rețelele convoluționale urmează principiul prezentat anterior în ceea ce privește rețelele neurale, fiind formate dintr-un strat de intrare, un strat de ieșire și mai multe straturi ascunse.

Straturile ascunse ale CNN sunt formate din straturi convoluționale, iar funcția de activare este funcția ReLU, care este reprezentată în figura 3.2.4.1. Stratul ReLU poate fi urmat de un strat de grupare (pooling). Etapa de grupare se referă la alegerea valorilor reprezentative dintr-o fereastră.

Imaginile sunt percepute de către calculator ca o matrice în care fiecare element reprezintă valoarea modelului RVA (roșu-verde-albastru) corespunzător fiecărui pixel al imaginii. Considerând ca exemplu o imagine cu dimensiunea 64x64, intrarea X a rețelei va avea dimensiunea egală cu rezultatul înmulțirii 64x64x3, adică 12288 (64x64 reprezintă dimensiunea imaginii, iar înmulțirea acestora cu 3 se datorează celor 3 canale de culoare RVA). Acest lucru poate reprezenta o problemă în cazul imaginilor cu un număr foarte mare de pixeli (o imagine mai clară va avea un număr mai mare de pixeli), dacă neuronii din straturi sunt conectați în totalitate cu cei din straturile anterioare, așa cum se întâmplă în cazul rețelelor neurale obișnuite.

Soluția problemei enunțate anterior o reprezintă operația de convoluție. Cu ajutorul acesteia, numărul de parametrii ai rețelei este redus. Acesta este definit prin 3 mărimi: lățimea imaginii, înălțimea și adâncimea acesteia. Pentru o vizualizare mai exactă al acestui aspect, se va face o comparație între o rețea neurală obișnuită, având 2 straturi ascunse (figura 3.4.1.) și o rețea neurală convoluțională (figura 3.4.2.).

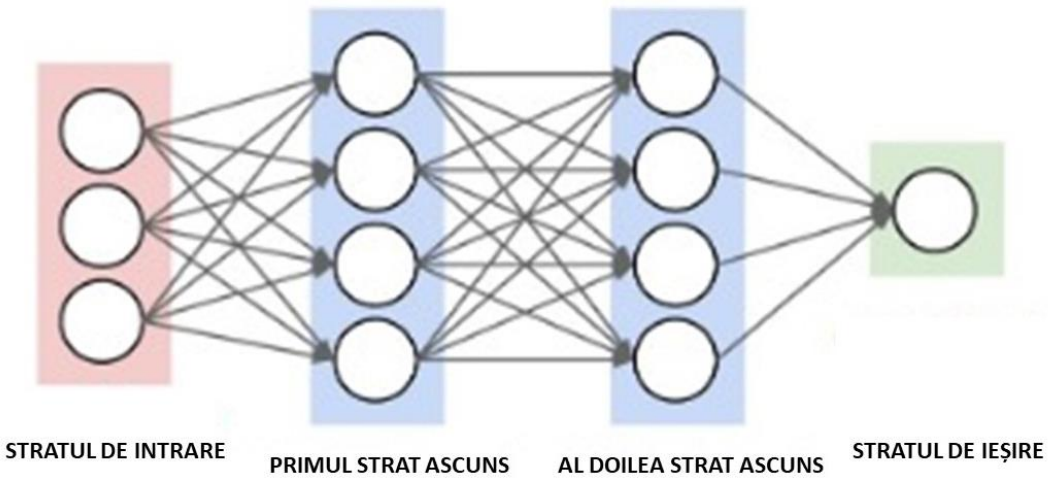


Figura 3.4.1 : Rețea neurală obișnuită [11]

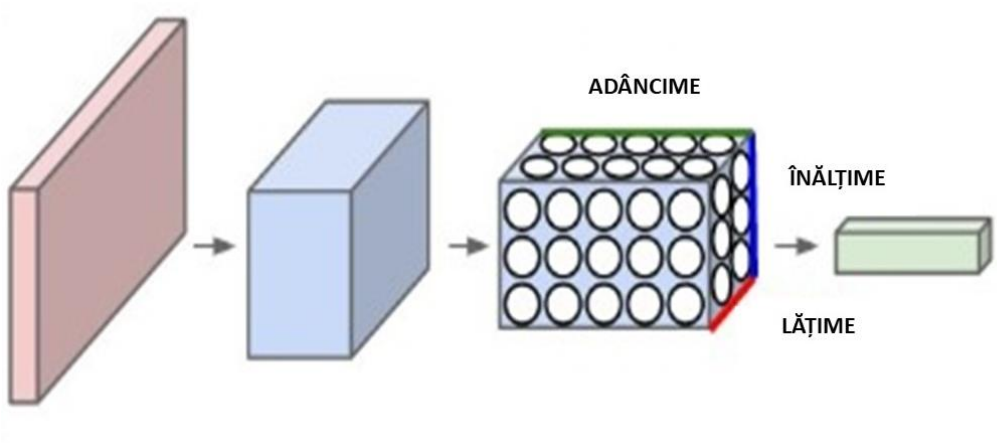


Figura 3.4.2 : Rețea neurală convoluțională [11]

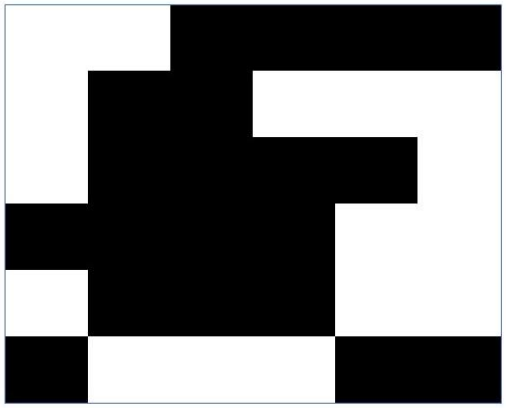
După cum se poate observa în imaginile de mai sus, în cazul CNN, neuronii aferenți unui strat sunt conectați doar într-o anumită regiune, spre deosebire de cazul rețelelor neurale obișnuite, unde neuronii sunt conectați complet.

Vectorul ponderilor și parțialităților poartă numele de filtru. În cazul unei rețele neurale convoluționale, un filtru poate fi folosit de un număr mai mare de neuroni. Operatorul de convoluție are rolul de a extrage caracteristici din imaginea de intrare, păstrând în același timp dependențele spațiale și temporale ale pixelilor imaginii, prin aplicarea unor filtre adecvate. Procesul de

convoluție este explicat în imaginile care urmează, fiind adaptat după un exemplu practic existent pe [12].

REALIZAREA CONVOLUȚIEI

Considerăm o imagine, având matricea următoare:



1	1	0	0	0	0
1	0	0	1	1	1
1	0	0	0	0	1
0	0	0	0	1	1
1	0	0	0	1	1
0	1	1	1	0	0

Figura 3.4.3 : Imaginea exemplu și matricea aferentă

După cum se poate observa în figura 3.4.1.1, imaginea pe care o voi folosi ca exemplu este o imagine alb-negru. În ceea ce privește matricea aferentă, aceasta are o dimensiune de 6x6. Cifra 1 corespunde culorii alb, iar cifra 0, culorii negru.

Considerăm mai departe un filtru cu dimensiunea 3x3, având forma:

F =

1	0	0
0	1	0
1	0	1

Procesul de convoluție presupune aplicarea filtrului pe fiecare porțiune a matricei aferente imaginii, așa cum este sugerat în figura 3.4.1.2. Aplicarea filtrului presupune înmulțirea elementelor acestuia cu elementele corespunzătoare porțiunii de matrice asupra căruia a fost aplicat. Mai departe, produsul acestor elemente se adună, iar rezultatul obținut va reprezenta un element al matricei de convoluție. Matricea de convoluție va avea dimensiunea $(n + f - 1)$x$(n - f + 1)$, unde n reprezintă dimensiunea imaginii și f este dimensiunea filtrului. În cazul de față 6-3+1=4, deci matricea obținută în urma filtrării va avea 4x4.

Primul pas al aplicării convoluției ar arăta în felul următor:

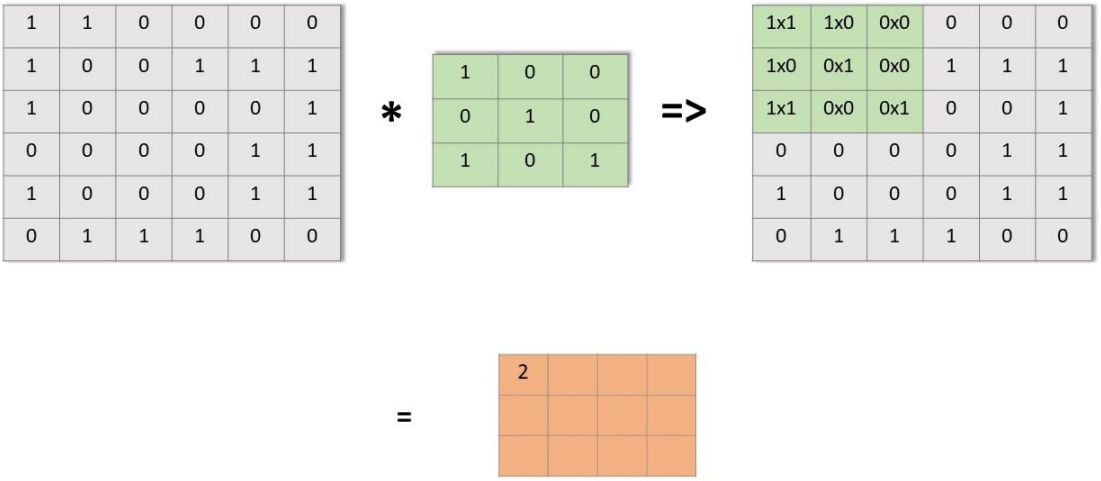


Figura 3.4.4 : Primul pas al procesului de convoluție

Urmând algoritmul, pasul următor va fi:

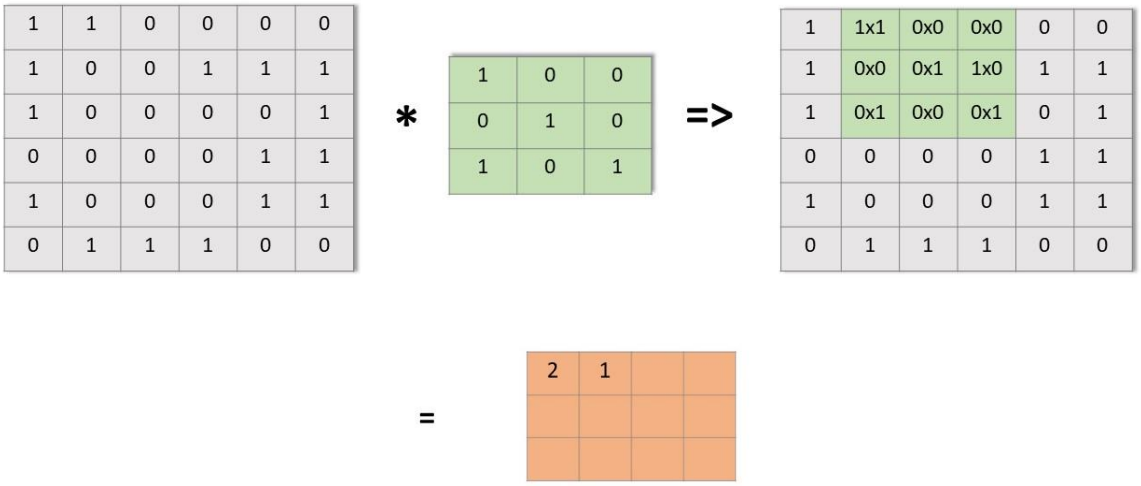


Figura 3.4.5 : Al doilea pas al procesului de convoluției

Se vor repeta pașii în continuare, iar rezultatul convoluției va fi:

2	1	1	3
1	0	1	2
1	2	1	2

Figura 3.4.6 : Rezultatul convoluției

CAPITOLUL 4
RECUNOAȘTEREA AUTOMATĂ A VORBIRII

Un sistem automat de recunoaștere a vorbirii (SRV) este un sistem informatic capabil să identifice informațiile pe care vorbitorul le transmite prin intermediul unui microfon. Cele mai avansate sisteme RAV se bazează pe modelele Markov ascunse (HMM). Într-o situație ideală, acest sistem ar fi capabil să realizeze recunoașterea în timp real, cu o precizie de 100% a tuturor cuvintelor rostite, independent de mărimea vocabularului, limba vorbită, sau factorii externi, precum zgomotul sau caracteristicile canalului prin care sunt transmise. [13]

Desigur, în aplicațiile reale, sistemul nu prezintă o acuratețe de 100%. Se întâmplă de multe ori ca oamenii să nu reușească să comunice între ei datorită diverșilor factori, precum zgomotul de fundal, dialectul diferit, etc. Răspunsul sistemului poate fi, de asemenea, influențat de o multitudine de elemente, cum ar fi zgomotul de fundal, dimensiunea vocabularului, gradul de familiarizare al sistemului cu interlocutorul, caracteristicile vorbitorului (dacă acesta are un anumit accent, dacă pronunță corect cuvintele, dacă vorbește sacadat, sau, dimpotrivă, mult prea rapid etc.).

Sistemele de recunoaștere automată a vorbirii prezintă o importanță deosebită și pot fi utilizate într-o gamă largă de aplicații. Implementate pe anumite mașini, precum roboți, pot ajuta la ușurarea vieții persoanelor cu diverse probleme, precum persoanele imobilizate la pat sau nevăzătoare. Acestea pot transmite comenzi roboților, care la rândul lor pot realiza manevre pe care persoanele cu nevoi nu le pot executa singure.

Intrarea sistemului este reprezentată de semnalul vocal. Mesajul vocal este compus din mai multe unități elementare, numite foneme. Acestea reprezintă sunete scurte, care se pot cataloga în trei categorii: vocale, semivocale și consoane. Alăturarea mai multor foneme formează cuvintele, care împreunate formează propoziții sau fraze. Procesul de recunoaștere vocală se referă, din punct de vedere probabilistic, la aflarea secvenței de cuvinte W^* care este cel mai probabil să apară în urma mesajului transmis X.

$$W^* = argmax\, P(W|X) \quad (17)$$

Aplicând regula lui Bayes, formula devine:

$$W *= argmax \frac{P(X|W)*P(W)}{P(X)} = argmax\, P(X|W) * P(W) \quad (18)$$

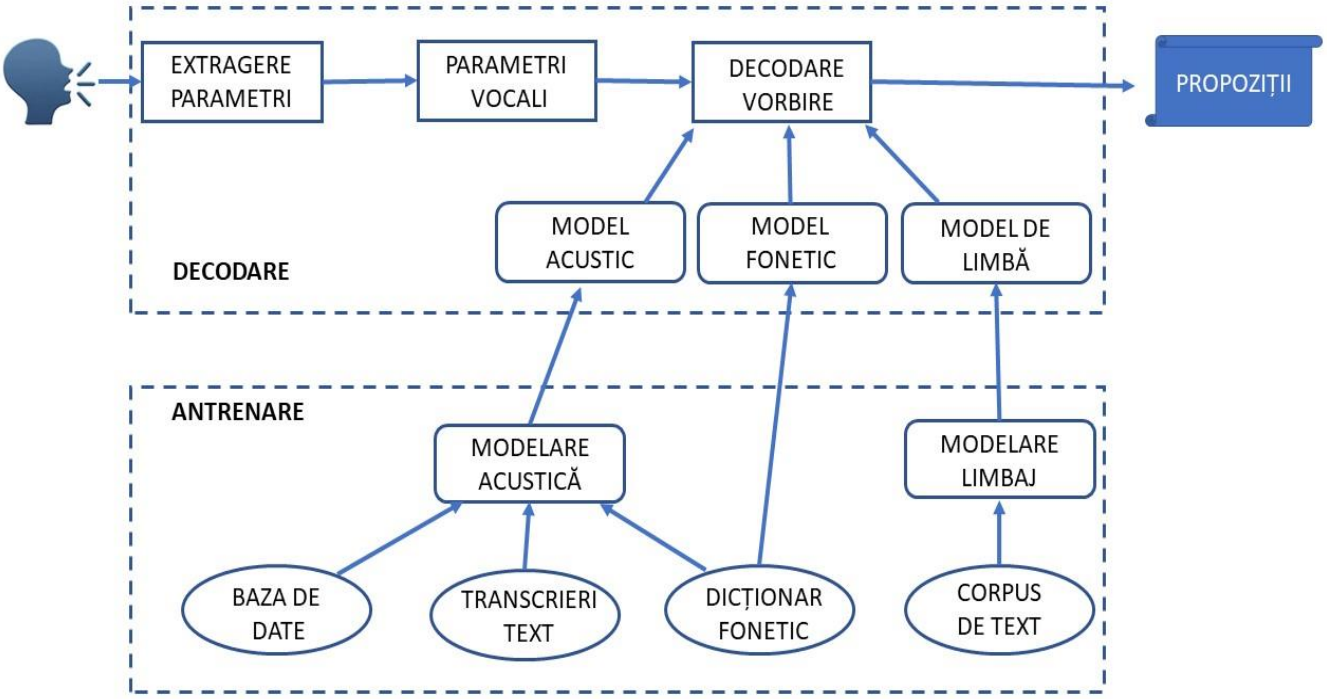


Figura 4.1 : Arhitectura unui SRV; după [14]

MODELUL ACUSTIC

Acesta este utilizat pentru estimarea probabilității cuvintelor ce urmează a fi rostite, în funcție de secvența deja enunțată P(X|W). Pentru ca modelul acustic să poată fi construit este nevoie de o bază de date cât mai amplă, care să conțină înregistrări audio, împreună cu o traducere scrisă a cuvintelor mesajelor exprimate.

MODELUL DE LIMBĂ

Modelul de limbă oferă o estimare a probabilității apriori a secvenței de cuvinte P(W). Acesta presupune utilizarea unui set de reguli, astfel încât alăturarea cuvintelor să formeze propoziții inteligibile.

MODELUL FONETIC

Pronunția cuvintelor urmează, de asemenea, anumite reguli. Modelul fonetic reprezintă dicționarul în care se regăsește modalitatea corectă de pronunție a cuvintelor.

4.1. MECANISMUL UMAN AL VORBIRII

Vorbirea reprezintă unul dintre mijloacele prin care oamenii își pot exprima gândurile, ideile și dorințele. Aceste gânduri sunt transpuse în cuvinte, care sunt transmise pe cale orală, astfel încât să poată fi înțelese de către ascultători. Cuvintele sunt formate din sunete, care sunt emise în urma unor mișcări voluntare ale aparatului respirator și masticator. Aparatul vorbirii cuprinde mai multe componente anatomice, care prezintă următoarele roluri în emiterea sunetelor:

Sistemul respirator este responsabil pentru asigurarea energiei necesare vorbirii. Undele de aer existente în plămâni sunt expulzate către laringe în urma aplicării presiunii de către mușchii diafragmei

Laringele este compus din 3 cartilaje și 2 seturi de mușchi și are rolul de a modula presiunea aerului. Cele două seturi de mușchi poartă numele de corzi vocale. Acestea sunt responsabile pentru modelarea sunetelor, reglând astfel volumul acestora. Vibrația cvasi-statică a corzilor produce sunetele vocalizate. Periodicitatea semnalului de excitație poartă numele de frecvență fundamentală. Valorile acesteia pot să difere de la persoană la persoană. În general, frecvența fundamentală în cazul bărbaților este de 132 Hz, iar în cazul femeilor este de 223 Hz. Frecvența semnalului vocal este dată de lungimea corzilor vocale. Astfel, dacă acestea sunt scurte, vocea este mai subțire, iar dacă sunt lungi, vocea este groasă. Între cele două seturi de corzi vocale se află glota, care atunci când este deschisă, permite aerului să circule dinspre plămâni către cavitatea bucală și nazală, sau în sens invers.

Tractul vocal are rolul de a filtra sunetele. Faringele modelează amplificarea sunetelor, iar cavitatea bucală este cea care determină pronunțarea acestora, fiind compusă din organe pasive (dinți, alveole etc.) și organe active (limba, buzele etc.). De asemenea, cavitatea nazală are și ea un rol important în pronunțarea sunetelor, aceasta fiind responsabilă pentru producerea sunetelor nazale.

4.1.1. CAPTAREA SEMNALELOR VOCALE

Sunetul este captat cu ajutorul unui dispozitiv numit microfon. Acesta reprezintă un traductor capabil să transforme sunetul în semnal electric. Acest semnal este ulterior eșantionat și cuantizat pentru a fi transformat într-un semnal digital. Prin eșantionare se înțelege procesul de obținere a valorilor semnalului analogic la momente de timp discrete. Cuantizarea se referă la conversia amplitudinii semnalului într-un număr binar discret, pentru fiecare moment la care s-a realizat eșantionarea. Conform teoremei lui Nyquist, frecvența de eșantionare trebuie să fie de cel puțin două ori mai mare decât frecvența maximă a semnalului.

$$F_{eșantionare} = 2 * F_{max} \quad (19)$$

Nerespectarea acestei condiții conduce la fenomenul cunoscut sub numele de aliere spectrală.

4.2. MODELELE MARKOV ASCUNSE

Lanțul Markov reprezintă un sistem matematic care își poate modifica starea în funcție de anumite reguli de probabilitate. Caracteristica unui astfel de lanț este faptul că stările viitoare au o natură fixă, indiferent de modul în care procesul a ajuns în starea sa actuală. Practic, probabilitatea trecerii la o anumită stare depinde doar de starea curentă și de timpul parcurs până la momentul respectiv. În cazul lanțurilor Markov, stările sunt observabile, având ca parametri unici probabilitățile tranzițiilor de stare. În cazul modelelor Markov ascunse, observatorul nu are acces la secvența de stări, de unde și numele de modele ascunse.

Modelele Markov ascunse (HMM) sunt automate probabilistice cu un număr finit de stări. Așa cum este sugerat în figura 4.2.1, aceste stări sunt conectate prin arce, reprezentând tranziții.

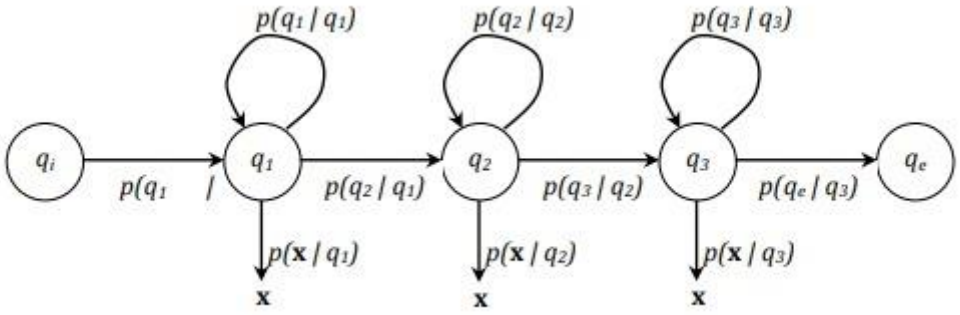


Figura 4.2.1 : Automat probabilistic de stări [14]

În cazul RAV, tranzițiile între stări au o importanță deosebită deoarece, pentru ca vorbirea să aibă sens, modelul trebuie să respecte anumite reguli:

Fonemul este unitatea lingvistică utilizată pentru recunoașterea vorbirii continue. Alăturarea mai multor foneme duce la formarea cuvintelor, alăturarea cuvintelor generează propoziții ș.a.m.d. Pentru a putea forma cuvinte corecte din concatenarea fonemelor, acestea trebuie să țină cont de un anumit lexic, așa cum, pentru a putea forma propoziții inteligibile, alăturarea cuvintelor este constrânsă de regulile gramaticale.

Semnalul vocal este unul nestaționar. Prin împărțirea acestuia în cadre scurte, de 20 ms – 30 ms, porțiunea din semnal este cvasi-staționară și, deci, se poate realiza analiza spectrală a acestuia. Pentru a putea discuta despre HMM, se consideră că semnalul vocal a fost segmentat în prealabil în secvențe de câteva zeci de milisecunde.

Un HMM este compus din:

- Secvența de stări a modelului, care este notată cu X și care este ascunsă
- Setul de simboluri de observație, notat cu $O = (o_1, o_2, \dots, o_T)$. Acesta este vectorul care conține caracteristicile extrase din semnal, corespunzătoare segmentelor în care acesta este

împărțit. Probabilitatea de apariție a acestora este dată de o funcție numită densitate de probabilitate.

- Set S de stări $Q = (q_1, q_2, \dots, q_s)$. Aceste elemente sunt singurele pe care le poate conține secvența X.

Cu A se notează matricea probabilităților de tranziție:

$$A = \begin{bmatrix} a_{11} & a_{12} \dots & a_{1S} \\ a_{21} & a_{22} \dots & a_{2S} \\ \vdots & & \\ a_{S1} & a_{S2} \dots & a_{SS} \end{bmatrix}$$

Unde $a_{ij} = P(i|j)$ reprezintă probabilitatea trecerii modelului din starea j în starea i. Deoarece tranziția este influențată doar de starea anterioară a modelului, nu de întreaga secvență de stări, vectorul probabilităților de stare poate fi calculat cu ajutorul formulei $S_{(t)} = AS_{(t-1)}$. Formula pentru calculul probabilității condiționate, corespunzătoare generării simbolurilor de observație O de către un anumit model, este următoarea:

$$P(O|M) = \Sigma a_{x(F)x(1)} \Pi b_{x(t)}(o_t) a_{x(t)x(t+1)} \quad (20)$$

Unde X(1) reprezintă starea inițială și X(F) reprezintă starea finală a modelului.

Pentru a putea implementa HMM în aplicațiile de recunoaștere vocală, trebuie efectuate trei etape:

- Etapa de antrenare a modelului presupune calculul elementelor matricei A, a densității de probabilitate a observațiilor și a probabilităților de stare inițiale pentru fiecare unitate de vorbire, segmentată și etichetată corespunzător
- Etapa de recunoaștere presupune calcularea probabilității ca un model antrenat în prealabil, să producă o anumită secvență de observații
- Etapa de evaluare constă în calculul probabilității $P(X|M)$, unde M reprezintă modelul dat.

4.2.1. ANTRENAREA HMM

Algoritmul Baum-Welch reprezintă metoda de antrenare a modelelor Markov ascunse. Etapa de antrenare este esențială deoarece se dorește ca parametrii modelului să corespundă unității lingvistice utilizate. Algoritmul este utilizat pentru obținerea unui maxim local al probabilității $P(O|M)$. Scopul final al etapei de antrenare este acela de a determina parametrii A (matricea probabilităților de tranziție), B (matricea probabilităților de observație) și π (vectorul de probabilități a stării inițiale), pornind de la setul de observații O și setul stărilor posibile al modelului.

Probabilitatea ca modelul dat să se afle în starea i la momentul t, respectiv în starea j la momentul (t+1) este dată de formula:

$$\xi_t(i,j) = P(q_t = i, q_{t+1} = j|O, M) = P\frac{(q_t=i, q_{t+1}=j \,|\, O, M)}{P(O|M)} \quad (21)$$

Pentru a putea calcula această probabilitate, trebuie să apelăm la algoritmul de înaintare și algoritmul de revenire.

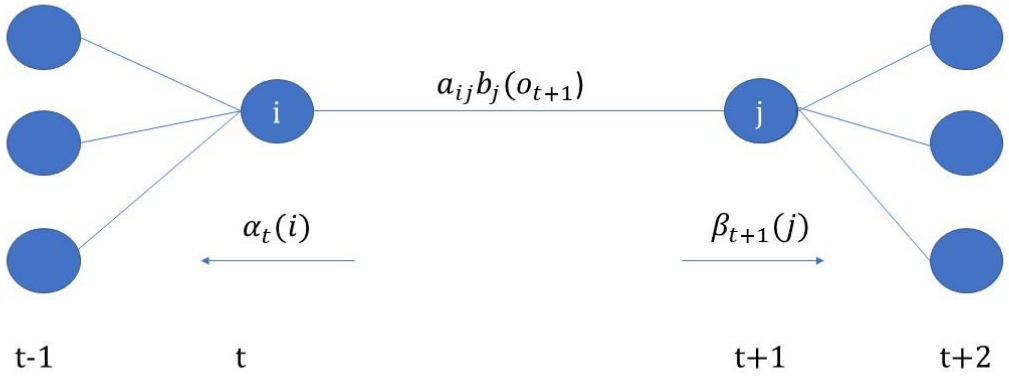


Figura 4.2.1.1 : Probabilitățile de înaintare și revenire

ALGORITMUL DE ÎNAINTARE

Algoritmul de înaintare prezintă avantajul faptului că acesta poate să stocheze rezultatele obținute, pe parcurs, în urma calculelor și să le utilizeze ulterior, scăzând astfel considerabil volumul de calcule efectuate.

Probabilitatea ca în momentul de timp t, modelul să ajungă la starea i în urma secvenței de observații parțiale $O_t(o_{1,}o_2, \dots, o_t)$ este notată cu $\alpha_t(i)$ și poartă numele de probabilitate de înaintare. Calculul acesteia presupune parcurgerea a trei etape:

- Etapa de inițializare $\alpha_1(i) = \pi(i)b_i(o(1))$
- Etapa de iterație $\alpha_{t+1}(j) = (\Sigma\alpha_t(i)a_{ij})b_j(o(t+1))$
- Etapa de finalizare P(O|M)=$\Sigma\alpha_T(i)$, unde T reprezintă momentul final de timp

ALGORITMUL DE REVENIRE

Algoritmul de revenire urmează un principiu asemănător celui descris mai devreme. Acesta își are originea la momentul de timp t+1 și se îndreaptă către momentul final de timp. Probabilitatea de revenire este notată cu $\beta_{t+1}(i)$. Calculul acesteia presupune următoarele etape:

- Etapa de inițializare $\beta_T(i) = 1$
- Etapa de iterație $\beta_t(i) = \sum_{j=1}^{N} a_{ij}\, b_j(o_{t+1})\beta_{t+1}(j)$

Utilizând cele două probabilități (de înaintare și de revenire), probabilitatea $\xi(i,j)$ devine:

$$\xi(i,j) = \frac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{P(O|M)} = \frac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)} \quad (22)$$

Definim probabilitatea ca la momentul t, modelul să se afle în starea i:

$$\gamma_t(i) = \sum_{j=1}^{N}\xi_t(i,j) \quad (23)$$

Numărul estimat al tranzițiilor din starea i reprezintă suma numărului total al tranzițiilor din starea i, respectiv numărul estimat al tranzițiilor din starea i în starea j reprezintă suma tuturor tranzițiilor din starea i în starea j. Rezultă astfel [15]:

$$\widehat{a_{ij}} = \frac{\sum_{t=1}^{T}\xi_t(i,j)}{\Sigma_{t=1}^{T}\Sigma_{j=1}^{N}\xi_t(i,j)} \quad (24)$$

Probabilitatea ca modelul să se afle în starea j la momentul de timp t este:

$$\gamma_t(j) = P(q_t = j|O,M) \quad (25)$$

Rezultă parametrii re-estimați:

$$\overline{\pi_j} = \gamma_1(j) \quad (26)$$

$$\overline{a_{ij}} = \frac{\sum_{t=1}^{T-1}\xi_t(i,j)}{\sum_{t=1}^{T-1}\gamma_t(j)} \quad (27)$$

$$\overline{b_j}(v_k) = \frac{\sum_{t=1\,t=v_k}^{T}\gamma_t(j)}{\sum_{t=1}^{T-1}\gamma_t(j)} \quad (28)$$

Probabilitatea de tranziție A și de observație B pot fi re-estimate dacă există deja estimări ale acestora, care reprezintă parametrii inițiali ai HMM pentru algoritmul de înaintare și retragere. Pașii care se realizează în mod iterativ sunt următorii:

- Se calculează γ și ξ din probabilitățile A și B anterioare
- Se calculează noile valori ale probabilităților A și B cu ajutorul lui γ și ξ.

4.3. COEFICIENȚII MEL CEPSTRALI

Un HMM necesită un bloc de extragere a unor parametrii pentru calcularea vectorilor de coeficienți, care sunt modelați ulterior de către modelul acustic. [14] Coeficienții Mel cepstrali (MFCC) sunt cei mai des utilizați în SRV-uri. Spre deosebire de coeficienții spectrali, coeficienții MFC prezintă avantajul de a fi necorelați. Pașii urmați pentru obținerea acestor parametrii sunt descriși în figura 4.3.1:

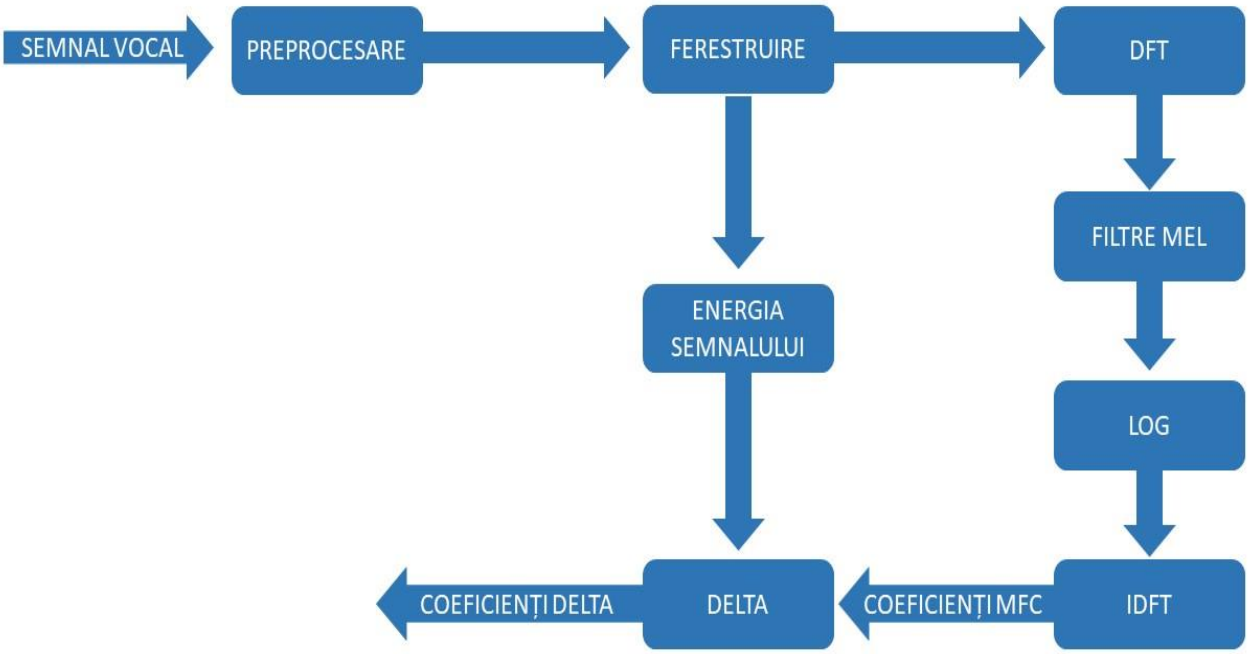


Figura 4.3.1 : Extragerea parametrilor Mel cepstrali

Blocurile prezentate în figura 4.3.1 vor fi analizate în continuare.

4.3.1. PREPROCESAREA SEMNALULUI

Acest pas este necesar pentru obținerea unui spectru având componentele de amplitudini egale. Se folosește un filtru FIR de ordinul întâi, având funcția de transfer:

$$H(z) = 1 - az^{-1} \quad (29)$$

Unde $0 \leq a \leq 1$.

$x'(n) = x(n) - ax(n-1)$, formula în domeniul timp aferentă preprocesării,

a - poartă numele de constantă de preprocesare și ia valori apropiate de 1 (ex. 0,95). [15]

În urma preprocesării, semnalul este împărțit în segmente de câteva zeci de milisecunde. Pentru aceste segmente scurte, semnalul poate fi considerat cvasistaționar, ceea ce facilitează extragerea caracteristicilor.

4.3.2. FERESTRUIREA SEMNALULUI

Analiza unui semnal de durată mare se poate traduce drept o trunchiere a semnalului. Acest lucru poate conduce la apariția fenomenului Gibbs, adică la apariția riplului în cazul răspunsului în frecvență.

Soluția acestei probleme o reprezintă utilizarea unei funcții fereastră. Aceasta va fi înmulțită cu un cadru al semnalului, ceea ce în domeniul timp se traduce printr-o convoluție.

$$x_t(n) = w(n) * x_t'(n) \quad (30)$$

Dezavantajul utilizării acestor funcții fereastră este acela că, odată cu aplicarea lor, se poate pierde din semnal o anumită cantitate de informație. Prin aplicarea acestora, deși se reduce considerabil fenomenul de dispersie, se obține de asemenea un spectru cu lobul central mai larg. Din acest motiv, alegerea funcției fereastră se face în funcție de aplicația dorită, luând în considerare avantajele și dezavantajele pe care acestea le prezintă.

Cele mai cunoscute funcții fereastră, împreună cu avantajele și dezavantajele lor sunt următoarele:

Fereastra dreptunghiulară

- Performanțe slabe

Fereastra triunghiulară (Bartlett)

- Calcule simple
- Lob central îngust
- Lobi centrali destul de mari

Fereastra Hanning

- Calcule relativ simple
- Lobul central îngust
- Primul lob lateral este mai mare, ceilalți micșorându-se rapid

Fereastra Hamming

- Primul lob lateral este destul de atenuat, dar lobii următori au o descreștere mai lentă

Fereastra Blackman

- Lob central mai larg, comparativ cu cel al ferestrei Hamming
- Atenuarea lobilor laterali foarte bună [5]

În general, cea mai utilizată fereastră este cea Hamming

$$w_{HAM}(n) = \begin{cases} \alpha - (1-\alpha)\cos\left(\frac{2\pi n}{N-1}\right), \\ 0, \qquad\qquad \text{în } rest \end{cases}$$

$0 \leq n \leq N-1$; N – numărul de eșantioane

4.3.3. TRANSFORMATA FOURIER DISCRETĂ

Transformata Fourier pentru un semnal aperiodic este definită prin:

$$X(e^{j\omega}) = \Sigma_{n=0}^{N-1} x(n) e^{-j\omega n} \quad (31)$$

Transformata Fourier discretă (DFT) reprezintă varianta practică de calcul a transformatei Fourier pentru un semnal discret în timp. [5]

Pentru calculul DFT, trebuie să înlocuim ω cu o variabilă discretă. Considerăm $\omega = \frac{2k\pi}{N}$, iar formula devine:

$$X_t(k) = X_t\left(e^{\frac{j2k\pi}{N}}\right), k = 0, \dots, N-1 \quad (32)$$

Complexitatea algoritmului este de $Nlog(N)$, motiv pentru care numărul de eșantioane se alege putere a lui 2.

4.3.4. FILTRAREA PE SCALA MEL

Urechea umană este mai puțin sensibilă la frecvențele înalte, motiv pentru care putem considera auzul ca fiind reprezentat pe scară logaritmică. Aproximarea pe care dorim să o facem este filtrarea pe scala Mel, parametrii vocali fiind astfel grupați în jurul frecvențelor joase. Transformarea frecvențelor în Mel este liniară pentru frecvențele de sub 1000Hz și logaritmică pentru frecvențele mai mari decât aceasta. Formula de transformare este:

$$F_{MEL} = 1127 \ln\left(1 + \frac{f}{700}\right) \quad (33)$$

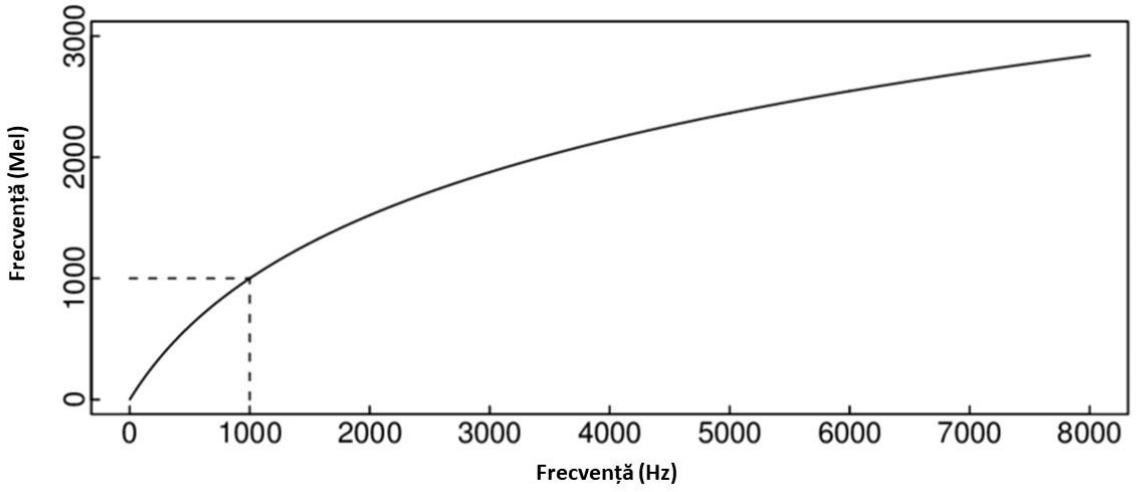


Figura 4.3.4.1 : Scala Mel în funcție de scala în Hz [16]

Filtrarea Mel se realizează prin calculul convoluției dintre semnalul vocal și filtrul Mel, realizat pentru fiecare cadru în parte. Această filtrare are ca scop netezirea spectrului obținut în urma aplicării DFT. Filtrarea este realizată cu ajutorul unui set de filtre triunghiulare Mel (figura 4.3.4.2). Aceste filtre sunt alese astfel:

- Se calculează spectrul în jurul frecvenței centrale aferente fiecărei benzi
- Se aleg filtrele, în funcție de lărgimea benzii corespunzătoare frecvenței centrale a ferestrei.

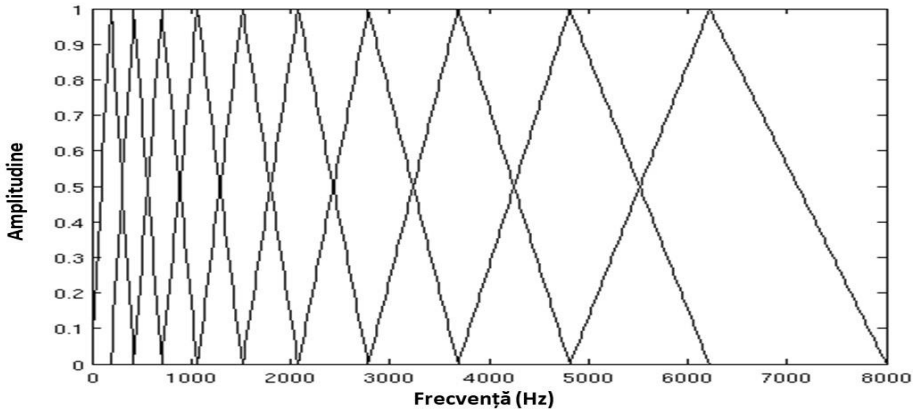


Figura 4.3.4.2 : Set de filtre triunghiulare Mel [17]

4.3.5. CALCULUL MFCC

Pentru convertirea spectrului logaritmic pe scala Mel înapoi în domeniul timp, se folosește transformata cosinus discretă. Rezultatul obținut reprezintă coeficienții Mel cepstrali, 13 dintre aceștia fiind folosiți în general pentru recunoaștere. Anumite prelucrări sunt necesare pentru îmbunătățirea acestor parametrii, precum scalarea acestora și scăderea mediei cepstrale.

4.3.6. CALCULUL ENERGIEI ȘI A COEFICIENȚILOR DELTA

După cum am menționat anterior, în urma calculelor se obțin 13 coeficienți cepstrali folosiți în recunoașterea vocală. Putem obține un total de 39 de parametri dacă luăm în considerare și derivatele de ordinul I și II ale acestora, derivate calculate în funcție de timp. Derivatele de ordinul I determină parametrii delta (numiți și coeficienți de viteză), iar derivatele de ordinul II determină parametrii delta-delta (numiți și coeficienți accelerație).

CAPITOLUL 5
DEZVOLTAREA PRACTICĂ A SISTEMULUI

5.1. BAZA DE DATE UTILIZATĂ

Baza de date utilizată a fost colectată de către Google și este distribuită sub licență CC. Aceasta conține 105.000 de fișiere de tip WAVE cu înregistrări ale diferitor oameni rostind câte un cuvânt. Vocabularul este compus 30 de cuvinte diferite, precum „go”, „left”, „down”, „off” etc.

Baza de date este utilizată astfel:

- Aproximativ 80% este folosită pentru partea de antrenate a rețelei
- 10% este folosită pentru partea de validare, care are rolul de a determina acuratețea pe parcursul etapei de antrenare
- 10% este utilizată pentru partea de testare și are rolul de a prezenta acuratețea modelului, odată cu încheierea fazei de antrenare.

Această catalogare a rețelei este extrem de importantă deoarece rețeaua poate tinde să memoreze intrările în etapa de antrenare. În acest fel, ne putem asigura că, în etapa de validare, rețeaua lucrează cu fișiere pe care nu le-a mai utilizat până la momentul respectiv. Etapa de testare este, de asemenea, foarte importantă deoarece ne asigură că rețeaua poate fi folosită pentru un set mare de intrări.

Ne dorim ca sistemul de recunoaștere vocală să poată fi utilizat pentru aplicațiile din viața reală. Așa cum am mai menționat pe parcursul acestei lucrări, există mai mulți factori care pot influența în mod negativ rezultatele SRV. Luând în considerare acest aspect, înregistrările au fost realizate cu ajutorul mai multor dispozitive și în medii diverse, astfel încât sunetul să nu fie perfect. Mai mult decât atât, baza de date conține un fișier numit “_background_noise_” cu înregistrări ale zgomotului traficului, al activităților de uz casnic și al diverselor tipuri de mașinării. Fragmente ale acestor

înregistrări sunt suprapuse în mod aleator peste înregistrările cuvintelor, la diverse intensități ale volumului, astfel încât să imite mediul de propagare al cuvintelor în viața reală.

5.2. LIBRĂRIA TENSORFLOW

TensorFlow este o librărie software, care oferă suport pentru dezvoltarea algoritmilor de inteligență artificială. Acesta a fost dezvoltat de către echipa Google Brain în anul 2015 și a fost folosit inițial doar în interiorul companiei Google, ulterior devenind o librărie care poate fi utilizată de către oricine.

Librăria are capacitatea de a efectua diverse calcule. Operațiile matematice sunt reprezentate prin noduri, iar între nodurile grafurilor se află tensorii, de TensorFlow. Acesta poate rula pe mai multe tipuri de procesoare, fiind compatibil de asemenea cu sistemele de operare Windows, Linux, macOS și platformele mobile Android și iOS.

5.3. MODELUL UTILIZAT

Modelul rețelei neurale utilizat este dezvoltat utilizând librăria TensorFlow [18] și este bazat pe arhitectura rețelelor neurale convoluționale. În general, acest tip de rețele este folosit pentru clasificarea obiectelor din imagini. Pentru a transforma semnalul continuu într-o imagine 2-Dimensională, este nevoie de efectuarea unui set de prelucrări, astfel încât să obținem ceea ce se spectograma semnalului.

Spectograma este o reprezentare grafică a densităților spectrale ale unui semnal. Forma cea mai uzuală a acesteia este reprezentată de un grafic bidimensional, care conține frecvențele semnalului în funcție de timp. În cazul de față, se alege o fereastră de timp suficient de mare încât să poată cuprinde, pe rând, toate cuvintele rostite din baza de date. Semnalul audio din fereastra respectivă va fi împărțit în secvențe scurte. Frecvențele semnalului corespunzătoare acestor segmente vor fi dispuse în funcție de timp și vor forma spectograma semnalului. De menționat faptul că, datorită așezării în memoria librăriei Tensorflow, timpul nu este dispus pa axa Ox, ci pe axa 0y, fiind parcurs de sus în jos. Frecvențele, pe de altă parte, sunt dispuse pe axa Ox, fiind parcursă de la stânga la dreapta.

Așa cum am menționat în capitolul 4.3, urechea umană percepe sunetele într-un mod logaritmic. Din acest motiv este necesară o prelucrare suplimentară, care transformă spectograma într-un set de coeficienți Mel cepstrali.

5.3.1. ETAPA DE ANTRENARE A REȚELEI

Modelul a fost antrenat pe un laptop cu procesor Intel Core i7-6700HQ CPU @ 2.60GHz pe 64b, având memorie RAM de 8GB. Procesul de antrenare a durat aproximativ 20 de ore. Pe parcursul acestei etape au fost afișate mai multe informații:

- Numărul curent al pașilor efectuați: au fost necesari 18.000 de pași pentru ca procesul de antrenare să fie complet
- Rata de învățare: la începutul procesului aceasta avea valoarea de 0.001. Pe parcurs, aceasta a fost redusă, iar la final avea valoarea de 0.0001
- Acuratețea: reprezintă procentul claselor prezise corect de către rețea. În mod evident, valoarea de început a acesteia a fost destul de mică, însă a crescut pe parcursul antrenării. După aproximativ 10.000 de pași, aceasta a ajuns la 40%, iar la final avea valoarea de 89%
- Entropia încrucișată: valoarea de început a acesteia a fost de aproximativ 2.5, dar a fost micșorată pe parcursul antrenării. Astfel, după aproximativ 10.000 de pași, aceasta avea valoarea de 1.5, iar la final, valoarea sa era de aproximativ 0.4.

5.3.2. MATRICEA DE CONFUZIE

Matricea de confuzie este un artificiu extrem de util și des folosit pentru algoritmii de inteligență artificială, în special pentru algoritmii de învățare supervizată. Aceasta are rolul de a oferi un răspuns vizual al performanțelor algoritmului și poate fi utilizată pentru îmbunătățirea acestora.

La modul general, liniile metricei pot reprezenta numărul de obiecte clasificate de către rețea ca aparținând unei anumite categorii, iar coloanele pot reprezenta clasele reale ale respectivelor obiecte. Pentru a înțelege mai bine cum funcționează o astfel de matrice, vom considera exemplul următor:

Considerăm un sistem care a fost antrenat să diferențieze imaginile unui delfin față de cele ale unui rechin. Presupunând un set de date care conține10 imagini ale animalelor (6 delfini și 4 rechini), tabelul realizat pe baza matricei erorilor ar putea fi:

		CLASIFICAREA REALĂ	
		DELFIN	**RECHIN**
CLASIFICAREA REȚELEI	**DELFIN**	**4**	1
	RECHIN	2	**3**

Tabelul 5.3.2.1 : Exemplu de tabel realizat pe baza matricei de confuzie

După cum se poate observa, din cei 6 delfini, 2 au fost clasificați drept rechini, iar din cei 4 rechini, 1 a fost clasificat drept delfin.

Figura 5.3.2.1 reprezintă matricea de confuzie obținută la finalul etapei de antrenare a rețelei:

Figura 5.3.2.1 : Matricea de confuzie obținută

Urmând exemplul descris anterior, o matrice de confuzie ideală ar avea doar diagonala principală diferită de 0, restul termenilor fiind nuli. Se poate observa în figura de mai sus că matricea obținută nu este una ideală, însă numărul fișierelor clasificate greșit este totuși mic, comparativ cu numărul fișierelor clasificate în mod corect.

Toate fișierele TensorFlow sunt bazate pe repetoare de protocol, denumite protobuff. Structurile de date sunt definite în fișiere text, iar uneltele protobuf generează clase în diverse limbaje de programare, printre care se numără și Python. Acestea pot fi accesate ulterior într-un mod facil. Pentru realizarea calculelor, TensorFlow utilizează obiectul graf. Acesta este format dintr-o rețea de noduri, care reprezintă operații, conectate între ele. În urma creării unui graf, acesta este salvat drept obiectul GraphDef, aparținând librăriei ProtoBuf.

După terminarea antrenării modelului, acesta este convertit într-un graf binar, denumit GraphDef. Pentru a realiza în mod practic recunoașterea cuvintelor transmise de către vorbitor, fișierele audio sunt trecute prin graful antrenat, iar predicțiile în urma trecerii fișierului prin rețea sunt afișate în consolă, sub formă de text.

5.3.3. REZULTATE OBȚINUTE ÎN URMA TESTĂRII MODELULUI

Modelul este antrenat să recunoască 10 cuvinte, acestea fiind: “go” , “up”, “down” , “stop”, “on”, “off” , “left” , “right” , “yes” , “no”. Cuvintele diferite de acestea vor fi catalogate drept “unknown”, sau, în cazul în care nu este rostit niciun cuvânt, “silence”. În prima fază, testarea modelului a fost realizată prin înregistrarea unor fișiere de tip WAVE, folosind microfonul aferent laptopului pe care s-a realizat etapa de antrenare. Aceste fișiere au fost reeșantionate la o frecvență de 16 kHz, pentru a coincide cu frecvența fișierelor din baza de date. Rezultatul predicției este

prezentat prin afișarea în consolă a 3 răspunsuri. Acestea reprezintă predicția modelului, în mod ierarhic, primul răspuns fiind cel cu scorul cel mai mare. Scorul poate lua valoarea maximă 1 și sugerează încrederea rețelei în predicția făcută.

În tabelul 5.3.3.1 sunt prezentate o parte dintre rezultatele obținute. De menționat faptul că fișierele ".wav" au fost înregistrate în condiții reale, de zgomot. Modelul a fost testat cu ajutorul fișierelor audio înregistrate, atât în condiții de zgomot, cât și în condiții de absență a acestuia. În mod evident, rezultatele obținute în condiții de liniște au fost mult mai precise, scorul având în aceste cazuri valori de aproximativ 0.8. Totuși, am ales să prezint aceste rezultate deoarece, așa cum am menționat anterior, îmi doresc ca sistemul să poată fi folosit în aplicații reale.

CUVÂNTUL ROSTIT	REZULTATUL PREDICȚIEI
left	left (score = 0.48138) yes (score = 0.26522) no (score = 0.12129)
right	right (score = 0.45226) _unknown_ (score = 0.38339) go (score = 0.06209)
on	on (score = 0.75311) _unknown_ (score = 0.22758) off (score = 0.00640)
stop	stop (score = 0.58696) up (score = 0.36988) _unknown_ (score = 0.02486)
up	up (score = 0.53881) stop (score = 0.41561) _unknown_ (score = 0.02276)
go	go (score = 0.67365) stop (score = 0.14224) no (score = 0.09557)

Tabelul 5.3.3.1 : Rezultate obținute

5.4. IMPLEMENTAREA SISTEMULUI

Scopul final al proiectului de față este acela de a putea interacționa cu robotul umanoid NAO, prin intermediul vocii. Primul pas pentru a realiza acest lucru a fost programarea robotului pentru a înregistra fișiere audio de tip WAVE, de durată de 3 secunde, care să conțină comanda transmisă de către utilizator. Am specificat ca frecvența de eșantionare să fie de 16 kHz, pentru a coincide cu frecvența de eșantionare a fișierelor audio din baza de date.

Următorul pas a fost acela de a testa sistemul de recunoaștere vocală realizat prin antrenarea rețelei neurale, folosind fișierele audio înregistrate cu ajutorul robotului. În acest pas am întâmpinat mai multe limitări. Sistemul de operare al robotului este OpenNAO, aceasta fiind o distribuție Linux, bazată pe Gentoo, creată special pentru cerințele de funcționare ale robotului. TensorFlow prezintă dependențe care nu sunt compatibile cu acest sistem de operare. De asemenea, după cum este specificat în capitolul 2.3.2, robotul are o memorie RAM de doar 1 GB, aceste două probleme forțându-mă să mă îndrept către o soluție diferită, mai precis, către realizarea unei conexiuni client-server.

Serverul reprezintă un tip de aplicație prin intermediul căreia se pot transmite servicii către alte aplicații sau dispozitive, care poartă numele de clienți. Aceștia comunică, în general, prin intermediul Internetului, însă pot avea și un suport fizic comun. Serverul este entitatea care își pune la dispoziție resursele pentru client, acesta din urmă fiind cel care are rolul de a iniția apelul către server. Conexiunea dintre cei doi este, de obicei, limitată în timp și poartă numele de sesiune. Pentru a putea realiza comunicația, clientul trebuie să cunoască adresa serverului, portul și protocolul de comunicație utilizat de server. Pentru realizarea acestei conexiuni am utilizat librăria Flask, care este o librărie scrisă în limbajul de programare Python, bazată pe setul de instrumente Werkzeug. Acest set de instrumente este capabil să implementeze operațiuni de tip interogare și alte funcții cu rol important. Metoda HTTP utilizată este metoda POST, capabilă să trimită date către server.

Am folosit o astfel de aplicație de tip client-server pentru a transmite fișiere audio înregistrate cu ajutorul robotului, către laptop. Aceste fișiere sunt ulterior introduse în rețea și reprezintă comenzile rostite de vorbitor. După recunoașterea comenzii, rețeaua oferă răspunsul predicției, așa cum s-a putut observa în tabelul 5.3.3.1. Cuvântul cu acuratețea cea mai mare este transmis către robot sub formă de șir de caractere. Acesta este perceput de către robot, care va executa anumite acțiuni predefinite, în funcție de comanda dată de către vorbitor. Întreg sistemul este reprezentat în figura 5.4.1.

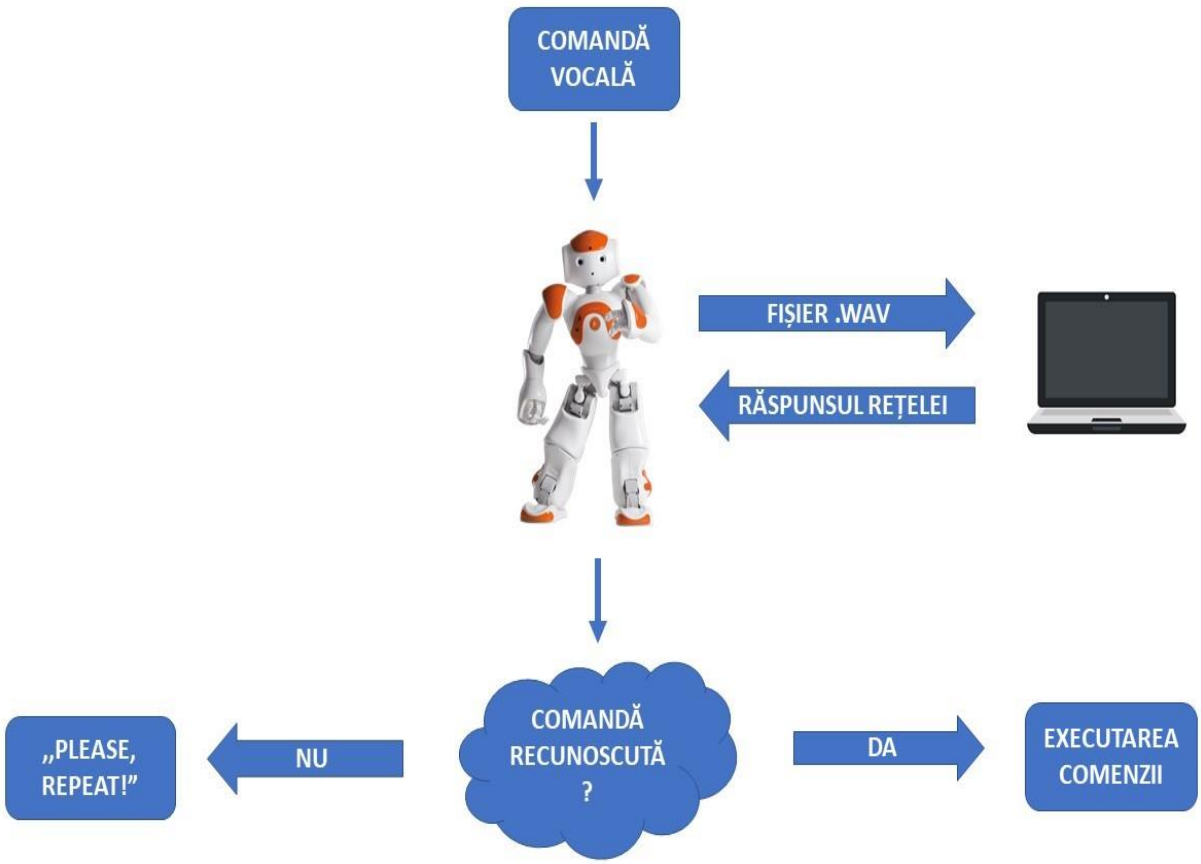


Figura 5.4.1 : Diagrama sistemului

5.4.1. EXECUTAREA COMENZILOR DE CĂTRE ROBOT

În urma transmiterii răspunsului rețelei către robotul NAO, acesta a fost programat să execute o serie de acțiuni, în funcție de comanda transmisă de către vorbitor. Limbajul de programare utilizat este Python. Librăria ALProxy este o librărie dezvoltată de către Aldebaran Robotics, care pune la dispoziția utilizatorului diverse funcții cu ajutorul cărora poate fi programat robotul. Am utilizat această librărie și am corelat cuvintele recunoscute de rețea, cu răspunsuri motrice și verbale din partea robotului. În figura 5.4.1.1. sunt sugerate toate comenzile recunoscute de NAO, împreună cu acțiunea pe care robotul o execută la primirea acestora.

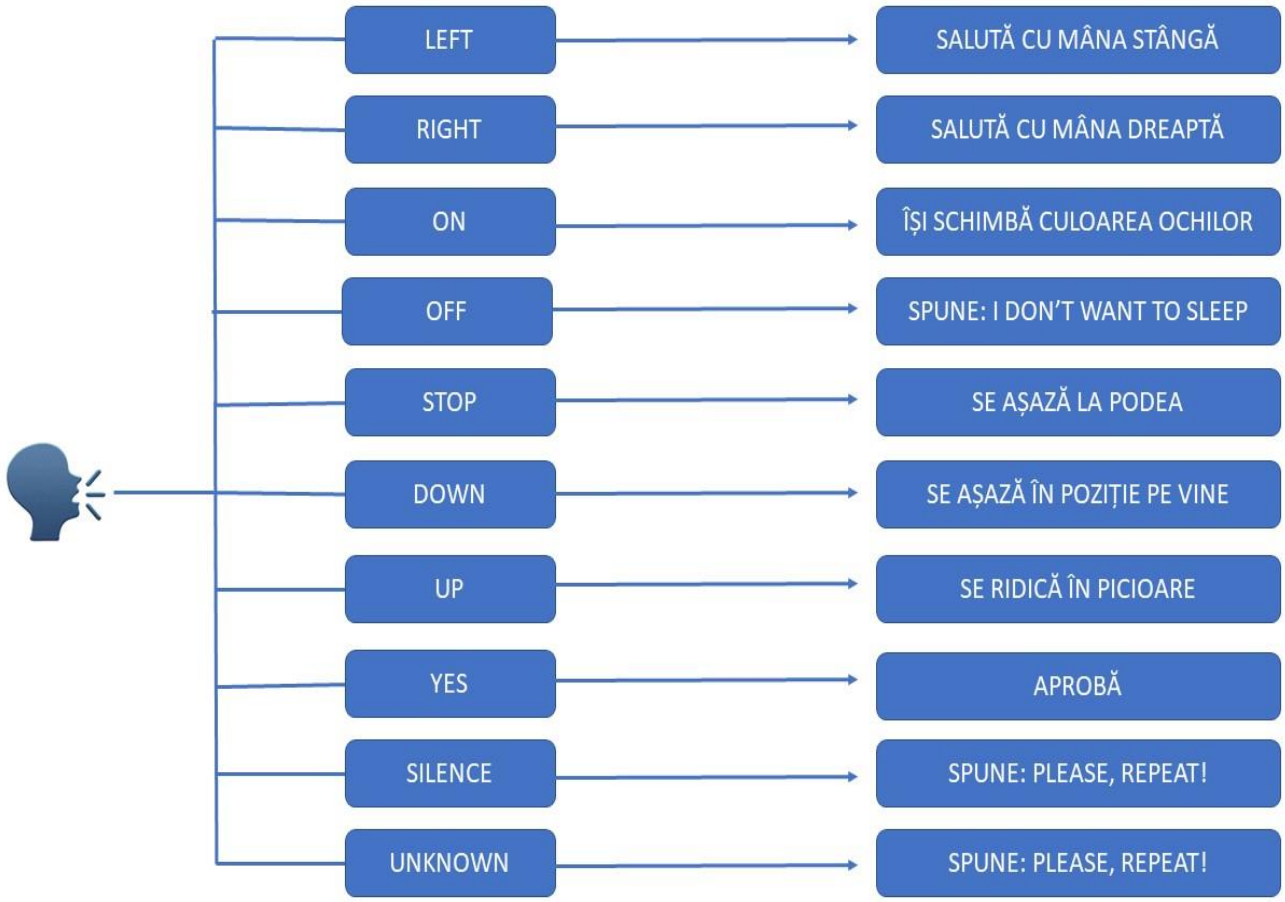


Figura 5.4.1.1 : Comenzile recunoscute de robot și executarea acestora

Un ultim detaliu în ceea ce privește implementarea practică a sistemului a constat în utilizarea unuia dintre butoanele tactile menționate în capitolul 2.3. Dintre cele 3 butoane aflate pe creștetul robotului, cel din mijloc a fost programat ca, la atingere, să înceapă procesul de înregistrare a comenzii și, practic, să pornească întregul sistem prezentat în figura 5.4.1. Acest detaliu a fost realizat cu scopul ca sistemul să fie cât mai ușor de folosit de către utilizatori.

CAPITOLUL 6
CONCLUZII

Scopul principal al proiectului a fost acela de a realiza un sistem de interacțiune prin voce, cu robotul umanoid NAO, acesta fiind realizat cu succes.

Sistemul de recunoaștere vocală a fost implementat utilizând arhitectura rețelelor neurale convoluționale. Performanțele acestuia sunt influențate de diverși factori care țin de calitatea înregistrărilor audio, precum zgomotul de fundal și calitatea microfoanelor care captează semnalul vocal. Modelul este independent de vorbitor, ceea ce face ca sistemul să poată fi utilizat de orice persoană. La momentul acesta, robotul are capacitatea de a recunoaște un vocabular format din 10 cuvinte în limba engleză. În cazul în care utilizatorul nu rostește nicio comandă, sau comanda transmisă nu este cunoscută de către robot, NAO generează un răspuns vocal în care îi cere utilizatorului să repete ceea ce a spus.

O limitare importantă a performanțelor sistemului o reprezintă utilizarea microfoanelor disponibile pe NAO, deoarece fișierele audio înregistrate cu ajutorul acestora prezintă o cantitate destul de mare de zgomot. De asemenea, o altă limitare este reprezentată de incompatibilitatea sistemului de operare al robotului cu librăria TensorFlow. Datorită acestui aspect, sistemul este dependent de două componente de tip hardware: robotul NAO și laptopul care conține modelul antrenat al rețelei neurale artificiale. Cu toate acestea, comunicația de tip client-server între robot și laptop se realizează rapid, motiv pentru care, din punct de vedere al timpul de răspuns, aspectul menționat anterior nu reprezintă un inconvenient.

6.1. CONTRIBUȚII PERSONALE

Contribuțiile personale la dezvoltarea acestui proiect sunt:

- Antrenarea rețelei neurale artificiale pentru recunoașterea vocală
- Programarea robotului să înregistreze fișiere de tip WAVE, care să conțină comanda transmisă de către utilizator

- Realizarea conexiunii client-server între robotul NAO și laptopul care conține modelul antrenat
- Programarea robotului să execute diverse manevre în funcție de comanda dată de către utilizator
- Setarea rulării sistemului prin atingerea unuia dintre butoanele tactile ale robotului, astfel încât utilizatorul să interacționeze strict cu robotul NAO.

REFERINȚE

[1] Dicționarul Explicativ al Limbii Române

[2] https://www.indiamart.com/proddetail/nao-humanoid-robot-17671491230.html

[3] http://doc.aldebaran.com/2-1/home_nao.html ,accesat la data: 24.05.2019

[4] *Cum funcționează neuronul biologic?* https://www.descopera.org/cum-functioneaza-neuronul-biologic/ ,accesat la data: 18.06

[5] Burileanu D., Curs de Tehnici Avansate de Prelucrare Digitală a Semnalelor

[6] Grigore O., Curs de Rețele Neuronale și Sisteme fuzzy

[7] *How the backpropagation algorithm works* http://neuralnetworksanddeeplearning.com/chap2.html , accesat la data: 19.06

[8] *An introduction to linear descent and Linear Regression* https://spin.atomicobject.com/2014/06/24/gradient-descent-linear-regression/ , accesat la

data: 19.06.2019

[9] *Intuition of Gradient Descent for Machine Learning* https://medium.com/abdullah-al-imran/intuition-of-gradient-descent-for-machine-learning-49e1b6b89c8b ,accesat la data: 19.06.2019

[10] *Introducere în rețele neuronale – Teorie și aplicații* https://www.code-it.ro/introducere-in-retele-neuronale-teorie-si-aplicatii/ 19.06.2019

[11] *Convolutional Neural Networks* https://cs231n.github.io/convolutional-networks/ ,accesat la data: 19.06.2019

[12] *A comprehensive Guide to Convolutional Neural Networ*ks https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 , accesat la data: 20.06.2019

[13] Burileanu C., ”Speech Recognition Experimental Results for Romanian Language”, 2019

[14] Cucu H., *Proiect de cercetare-dezvoltare în tehnologia vorbirii*

[15] Domokos J. *CONTRIBUTII LA RECUNOASTEREA VORBIRII CONTINUE SI LA PROCESAREA LIMBAJULUI NATURAL*

[16] *Cepstrum and MFCC* https://wiki.aalto.fi/display/ITSP/Cepstrum+and+MFCC , accesat la data: 21.06.2019

[17] *Mel Frequency Cepstral Coefficient (MFCC) tutorial* http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/ , accesat la data: 21.06.2019

[18] https://www.tensorflow.org/