**ROMÂNIA**

**MINISTERUL APĂRĂRII NAŢIONALE**

**ACADEMIA TEHNICĂ MILITARĂ**

**„FERDINAND I”**

**FACULTATEA DE SISTEME ELECTRONICE ŞI INFORMATICE MILITARE**

**Specializarea: Calculatoare şi sisteme informatice pentru apărare**

**şi securitate naţională**



**SISTEM SPECIALIZAT DE ACŢIONARE COMANDAT CU SEMNAL VOCAL**

CONDUCĂTOR ŞTIINŢIFIC:

**Conf. dr. ing. Laurențiu MĂRGĂRIT**

ABSOLVENT:

**Stud. sg. maj. Radu-Ilie HEREȘANU**

Conţine \_\_\_\_\_\_\_\_\_\_\_ file Inventariat sub nr. \_\_\_\_\_\_

Poziţia din indicator: \_\_\_\_

Termen de păstrare: \_\_\_\_\_

**BUCUREŞTI**

**2023**

ABSTRACT

Speech is the most natural way of communication, and its fundamental purpose is to convey information. Human-to-human communication is very fast through speech, and such fast human-computer communication has been desired, so several voice-based human-machine interaction systems have been developed, such as automatic speech recognition systems, speaker recognition or systems for spoken language identification.

Speech processing is a field that has evolved greatly as computer systems have become more powerful. With the help of technology that has developed in the last 50 years, we can extract a lot of information from a spoken signal, even if it takes only a few milliseconds.

This work is motivated by all the things mentioned above and aims to create a voice recognition system that can identify a predefined set of words. This model will be trained to recognize and classify these words, enabling the automation of specific actions. The solution will be integrated onto a Raspberry PI 4 board and run in an infinite loop.

The main goal of this work is to apply supervised learning algorithms for speech recognition on devices with limited computing power. In total, three architectures are evaluated: VGG16, AlexNet, and SqueezeNet.

**REZUMAT**

Vorbirea este cel mai natural mod de comunicare, iar scopul ei fundamental este transmiterea de informații. Comunicarea de la om la om este foarte rapidă prin vorbire și s-a dorit o astfel de comunicare rapidă om-computer, astfel încât au fost dezvoltate mai multe sisteme de interacțiune om-mașină bazate pe voce, cum ar fi sistemele automate de recunoaștere a vorbirii, recunoașterea vorbitorului sau sistemele pentru identificarea limbii vorbite.

Procesarea vorbirii este un domeniu care a evoluat foarte mult pe măsură ce sistemele informatice au devenit mai puternice. Cu ajutorul tehnologiei care s-a dezvoltat în ultimii 50 de ani, putem extrage o mulțime de informații dintr-un semnal vorbit, chiar dacă durează doar câteva milisecunde.

Această lucrare este motivată de toate lucrurile menționate mai sus și își propune să creeze un sistem de recunoaștere a vocii care poate identifică un set predefinit de cuvinte. Acest model va fi instruit să recunoască și să clasifice aceste cuvinte, permițând automatizarea unor acțiuni specifice. Soluția va fi integrată pe o placă Raspberry PI 4 și va rula într-o buclă infinită.

Scopul principal al acestei lucrări este de a aplica algoritmi de învățare supravegheată pentru recunoașterea vorbirii pe dispozitive cu putere de calcul limitată. În total, sunt evaluate trei arhitecturi: VGG16, AlexNet și SqueezeNet.

CUPRINS

ABSTRACT..............................................................................................................6

REZUMAT...............................................................................................................7

LISTĂ DE ABREVIERI.........................................................................................10

LISTĂ DE FIGURI.................................................................................................11

LISTĂ DE TABELE.............................................................................................. 13

[1 Introducere 14](#_Toc133689280)

[1.1 Prezentare generală 14](#_Toc133689281)

[1.2 Importanța temei alese 15](#_Toc133689282)

[1.3 Utilitatea sistemului 15](#_Toc133689283)

[2 Instrumente folosite 16](#_Toc133689284)

[2.1 Visual Studio Code 16](#_Toc133689285)

[2.2 Python 16](#_Toc133689286)

[2.3 Raspberry Pi Imager 16](#_Toc133689287)

[2.4 Putty 17](#_Toc133689288)

[2.5 VNC Viewer 17](#_Toc133689289)

[2.6 WinSCP 17](#_Toc133689290)

[3 Concepte de bază 18](#_Toc133689291)

[3.1 ML 18](#_Toc133689292)

[3.2 Funcția step 21](#_Toc133689293)

[3.3 Funcția sigmoid 22](#_Toc133689294)

[3.4 Funcția tangentă hiperbolică 24](#_Toc133689295)

[3.5 Funcția ReLU 24](#_Toc133689296)

[3.6 Învățare profundă pentru recunoașterea vorbirii 27](#_Toc133689297)

[3.6.1 CNN 28](#_Toc133689298)

[3.7 Modele investigate 29](#_Toc133689299)

[3.7.1 VGG16 29](#_Toc133689300)

[3.7.2 AlexNet 31](#_Toc133689301)

[3.7.3 SqueezeNet 32](#_Toc133689302)

[3.8 Extragerea caracteristicilor 34](#_Toc133689303)

[3.8.1 Cepstru 34](#_Toc133689304)

[3.8.2 Short Time Fourier Transform 35](#_Toc133689305)

[3.8.3 Mel Frequency Cepstral Coefficients 36](#_Toc133689306)

[4 Echipamente folosite 40](#_Toc133689307)

[4.1 Raspberry Pi 4 40](#_Toc133689308)

[4.2 Microfon Trust GXT212 Mico 43](#_Toc133689309)

[4.3 Acumulator extern Romoss OEM Solo 5 43](#_Toc133689310)

[4.4 Componente electronice pasive 44](#_Toc133689311)

[4.4.1 Breadbord 44](#_Toc133689312)

[4.4.2 Fire Dupont 45](#_Toc133689313)

[4.4.3 Rezistor 45](#_Toc133689314)

[4.4.4 Led 46](#_Toc133689315)

[5 Implementarea practică a sistemului de programe 47](#_Toc133689316)

[5.1 Setul de date folosit 47](#_Toc133689317)

[5.2 Dependințe principale 49](#_Toc133689318)

[5.3 Colectarea datelor și extragerea caracteristicilor 50](#_Toc133689319)

[5.4 Adaptarea modelelor 57](#_Toc133689320)

[5.5 VGG16 59](#_Toc133689321)

[5.6 AlexNet 60](#_Toc133689322)

[5.7 SqueezeNet 60](#_Toc133689323)

[5.8 Conversia la Tensorflow Lite 61](#_Toc133689324)

[5.9 Utilizarea modelului pe Raspberry Pi 61](#_Toc133689325)

[6 Mediul de lucru 67](#_Toc133689326)

[6.1 Crearea environmentului 67](#_Toc133689327)

[6.2 Instalarea sistemului de operare pe Raspberry Pi 68](#_Toc133689328)

[6.3 Conectarea la Raspberry Pi 69](#_Toc133689329)

[6.4 Interconectarea componentelor hardware ale sistemului 70](#_Toc133689330)

[7 Rezultate experimentale și discuții 71](#_Toc133689331)

[8 Concluzii 75](#_Toc133689332)

[9 Bibliografie 75](#_Toc133689333)

[10 Anexe 78](#_Toc133689334)

[10.1 Anexa A 78](#_Toc133689335)

[10.2 Anexa B 79](#_Toc133689336)

[10.3 Anexa C 80](#_Toc133689337)

[10.4 Anexa D 80](#_Toc133689338)

LISTĂ DE ABREVIERI

ML – Machine Learning

DL – Deep Learning

ASR – Automatic Speech Recognition

NN – Neural Network

DNN – Deep Neural Network

NLP – Natural Language Processing

ReLU - Rectified Linear Unit

SGD - Stochastic Gradient Descent

FFT – Fast Fourier Transform

STFT – Short Time Fourier Transform

MFCC - Mel Frequency Cepstral Coefficients

IoT – Internet of Things

CNN – Convolutional Neural Network

RNN – Recurrent Neural Network

VGG16 – Visual Geometry Group 16

SBC – Single-board computer

GPIO – General Purpose Input/Output

OS – Operating System

ILSVRC - ImageNet Large Scale Visual Recognition Challenge

WAV - Waveform Audio File Format

SSH – Secure Shell

USB – Universal Serial Bus

LISTĂ DE FIGURI

[Fig. 3.1 Diferențele conceptuale dintre învățarea automată și programarea tradițională 19](#_Toc133340295)

[Fig. 3.2 Unitatea de bază a rețelei neuronale (neuron, nod) (McCulloch și Pitts) 21](#_Toc133340296)

[Fig. 3.3 Funcția treaptă unitate utilizată în arhitectura Perceptron 22](#_Toc133340297)

[Fig. 3.4 Plotare a funcției sigmoid 24](#_Toc133340298)

[Fig. 3.5 Graficul funcției tangentă hiperbolică și a funcției sigmoid 25](#_Toc133340299)

[Fig. 3.6 Graficul funției tangentă hiperbolică, sigmoid și derivatele acestora 26](#_Toc133340300)

[Fig. 3.7 Graficul funcției ReLU și derivata sa 27](#_Toc133340301)

[Fig. 3.8 Graficul funcției ReLU și Leaky ReLU 28](#_Toc133340302)

[Fig. 3.9 Procesul de extragere a caracteristicilor 37](#_Toc133340303)

[Fig. 3.10 Scala Mel 40](#_Toc133340304)

[Fig. 3.11 Banc de filtre Mel 40](#_Toc133340305)

[Fig. 4.1 Raspberry Pi 4 42](#_Toc133340306)

[Fig. 4.2 Tabelul codurilor de culori standard ale rezistenței 46](#_Toc133340307)

[Fig. 5.1 Crearea listelor cu monstre și etichetele acestora 51](#_Toc133340308)

[Fig. 5.2 Funcția Hann 53](#_Toc133340309)

[Fig. 5.3 Metoda calc\_mfcc\_delta 54](#_Toc133340310)

[Fig. 5.4 Definirea metodei extract\_features 56](#_Toc133340311)

[Fig. 5.5 Înlocuirea etichetelor pentru organizarea categoriilor 57](#_Toc133340312)

[Fig. 5.6 Redimensionarea tensorilor din 3D în 4D 58](#_Toc133340313)

[Fig. 5.7 Conversia etichetelor din numere întregi în vectori binari 59](#_Toc133340314)

[Fig. 5.8 Conversia modelului din Tensorflow în TensorflowLite 61](#_Toc133340315)

[Fig. 5.9 Parametri preliminari 62](#_Toc133340316)

[Fig. 5.10 Confiugurare buffer și model interpretor 63](#_Toc133340317)

[Fig. 5.11 Funția decimate 63](#_Toc133340318)

[Fig. 5.12 Prima parte a funcției sd\_callback 65](#_Toc133340319)

[Fig. 5.13 A doua parte a funției sd\_callback 66](#_Toc133340320)

[Fig. 6.1 Raspberry Pi Imager - Advanced Options 69](#_Toc133340321)

[Fig. 6.2 Putty - conectarea la Raspberry Pi 4 70](#_Toc133340322)

[Fig. 7.1 Parametrii de antrenament model 72](#_Toc133340323)

[Fig. 7.2 Antrenarea modelului și validarea sa 73](#_Toc133340324)

[Fig. 10.1 Diagrama de activitate a sistemului 78](#_Toc133340325)

[Fig. 10.2 Diagrama cazurilor de secvență 79](#_Toc133340326)

LISTĂ DE TABELE

[Tabel 1 Diferite configurații VGG 30](#_Toc133340336)

[Tabel 2 Dimensiuni arhitecturale AlexNet 32](#_Toc133340337)

[Tabel 3 Dimensiuni arhitecturale SqueezeNet 34](#_Toc133340338)

[Tabel 4 Numărul de monstre pentru fiecare categorie 47](#_Toc133340339)

[Tabel 5 Cuvinte cheie implementate în sistem 48](#_Toc133340340)

[Tabel 6 Procentele de abandon ale straturilor Dropout în VGG16 59](#_Toc133340341)

[Tabel 7 Pregătirea mediului de lucru și a pachetelor necesare 66](#_Toc133340342)

[Tabel 8 Performanța CNN-urilor pe setul cu 6 cuvinte de activare 72](#_Toc133340343)

[Tabel 9 Compararea timpilor pentru modelul cu 6 cuvinte de activare 73](#_Toc133340344)

[Tabel 10 Comparația dimensiunilor modelelor cu 6 cuvinte de activare 73](#_Toc133340345)

[Tabel 11 Performanța pe Raspberry Pi a modelelor cu 6 cuvinte de activare 73](#_Toc133340346)

# Introducere

În ultimii ani, odată cu avansul tehnologic, sistemele de învățare automată au fost aplicate la produse cu dimensiuni din ce în ce mai mici extinzând astfel aria de influență a IoT. În general, această dezvoltare continuă a tehnologiei de învățare automată are potențialul de a revoluționa modul în care interacționăm cu dispozitivele din jurul nostru și de a îmbunătăți experiența utilizatorului într-un mod semnificativ.

## Prezentare generală

Această lucrare de licență urmărește dezvoltarea unui sistem de învățare profundă (deep learning) aplicat pe un dispozitiv încorporat și conceput pentru a interacționa prin comenzi vocale.

Sistemul încorporat va răspunde la comenzi vocale simple formate dintr-un singur cuvânt și va genera o ieșire corespunzătoare cu inputul rostit de utilizator.

În total, sunt evaluate trei arhitecturi: AlexNet, VGG16 și SqueezeNet. După implementare, antrenare și testare, rezultatele confirmă că SqueezeNet este rețeaua cu cele mai mici cerințe de stocare și procesare. Pe de altă parte, VGG16 arată cea mai mare acuratețe a seturilor de testare, antrenament și validare, deși acest lucru vine cu prețul dimensiunii și complexității de calcul. În general, din evaluarea calitativă se poate spune că AlexNet este un compromis între performanță și resurse de calcul, cel puțin pentru dispozitivul investigat în acest studiu – Raspberry Pi 4.

## Importanța temei alese

Sistemele de recunoaștere vocală și de recunoaștere a vorbitorului sunt utilizate din ce în ce mai des în viața de zi cu zi, acestea fiind din ce în ce mai precise și sunt considerate a fi una dintre cele mai promițătoare tehnologii emergente în domeniul interacțiunii om-mașină. Aceste sisteme permit utilizatorilor să comunice cu dispozitivele și aplicațiile lor folosind doar vocea, ceea ce face interacțiunea mai naturală și mai intuitivă decât introducerea de text sau utilizarea altor forme de input. De la asistenți virtuali precum Siri și Alexa, până la sisteme de recunoaștere vocală în mașini sau în call centere, aceste tehnologii sunt deja folosite într-o gamă largă de aplicații și se așteaptă să continue să crească în popularitate și utilitate în viitorul apropiat.

## Utilitatea sistemului

Soluția propusă este utilă în situațiile în care se dorește asigurarea unui sistem vocal de acționare, personalizat, care să deservească pentru anumite scopuri o organizație sau un utilizator normal. De asemenea sistemul oferă o soluție simplă și portabilă, ce poate fi utilizată și pe echipamente cu dimensiuni reduse și putere de procesare mai scăzută decât un calculator normal.

În acest moment se găsesc diferite sisteme software integrate ce îndeplinesc funcția de recunoaștere vocală și care pot transmite comenzi către un dispozitiv extern cu scopul de a-l manevra de la distanță. Ceea ce le diferențiază este scopul de utilizare, aria de folosire, producătorul, tehnologiile folosite, performanțele acestora, complexitatea și dimensiunile. Sistemul de față se diferențiază de altele prin: libertatea de adaptare a acțiunilor generate – având acces total la pinii de GPIO ai SBC-ului Raspberry Pi 4, posibilitatea de creare a unor modele personalizate care să reacționeze numai la anumite cuvinte de activare și procesarea semnalului vocal realizându-se local. De asemenea, sistemul prezentat nu necesită o conexiune la internet și nu accesează servere la distanță în vederea procesării semnalului vocal ceea ce îl face mai sigur în ceea ce privește securitatea, mai rapid în timpul de răspuns și mai fiabil în ceea ce privește mobilitatea.

## Rezumatul lucrării pe capitole

Capitolul 1 prezintă scopul lucrării și importanța acesteia, urmând ca în capitolul următor să se prezinte soluțiile software folosite pentru a realiza sistemul. Urmează o scurtă descriere a unor concepte teoretice de bază ce aparțin de domeniul inteligenței artificiale. În capitolul 4 sunt prezentate echipamentele hardware utilizate în proiect. Capitolele 5, 6 și 7 sunt cele mai importante, acestea prezentând în mare modul în care a fost implementat sistemul, cum au fost interconectate componentele și rezultatele obținute. Capitolul 8 este dedicat concluziei finale rezultate în urma implementării soluției, iar capitolele 9 și 10 sunt destinate referințelor bibliografice și anexelor.

# Instrumente folosite

În cadrul acestui sistem au fost utilizate mai multe soluții software necesare realizării acestuia. În cele ce urmează sunt prezentate cele mai importante dintre acestea.

## Visual Studio Code

Soluția software a fost implementată în mediul de dezvoltare Visual Studio Code. Acesta este un editor de cod, dezvoltat de Microsoft și disponibil gratuit pentru sistemele de operare Windows, macOS și Linux. Acesta a devenit foarte popular printre dezvoltatorii de software datorită funcționalităților sale avansate, performanței bune și a extensibilității ce permite dezvoltarea de software într-o varietate de limbaje de programare precum: Java, JavaScript, C++, Python etc.

Visual Studio Code este utilizat într-o varietate de scenarii de programare cum ar fi: dezvoltarea aplicațiilor web, dezvoltarea de jocuri, dezvoltarea de aplicații ce utilizează Open MPI, OpenCL și permite conectarea la WSL(Windows Susbsystem for Linux).

## Python

Python este un limbaj de programare interpretat, codul sursă fiind tradus și executat în timp real de către un interpretor. De asemena, Python este un limbaj orientat obiect, ceea ce înseamnă că utilizează concepte precum: clase, obiecte, încapsulare, moștenire și polimorfism.

Acest limbaj a fost creat de Guido van Rossum și a fost lansat inițial în februarie 1991. În prezent, este unul dintre cele mai utilizate limbaje de programare.

Cel mai mare avantaj al limbajului Python este colecția imensă de biblioteci standard pe care le poți folosi pentru următoarele: învățare automată, procesarea de imagini, calcul științific, procesarea de text și multe altele.

## Raspberry Pi Imager

Raspberry Pi Imager este un utilitar software gratuit dezvoltat de echipa Raspberry Pi pentru a ajuta la instalarea sistemului de operare Raspberry Pi OS pe un card SD sau microSD. Este disponibil pentru Windows, macOS și Linux și poate fi descărcat de pe site-ul oficial Raspberry Pi. Acest utilitar software este foarte ușor de utilizat și urmând câțiva pași în interfața grafică a aplicației, sistemul de operare este pregătit pentru a fi folosit.

## Putty

Putty este un program de tip terminal emulator, folosit în principal pentru a oferi acces la shell-ul unui sistem de operare la distanță. Acesta este disponibil pentru sistemele de operare Windows și Linux și este folosit pentru conectarea la servere SSH, Telnet sau la alte servicii de rețea [1].

Putty suportă o serie de protocoale de rețea, cum ar fi SSH, Telnet, rlogin și TCP. De asemenea, oferă suport pentru criptarea datelor transferate prin intermediul SSL(Secure Socket Layer) și este capabil să transmită date cu viteze mari, ceea ce face din Putty un instrument util pentru administrarea și gestionarea sistemelor de operare la distanță.

## VNC Viewer

VNC înseamnă Virtual Network Computing. Este un sistem de partajare a ecranului pe mai multe platforme care a fost creat pentru a controla de la distanță un alt calculator. Aceasta înseamnă că ecranul, tastatura și mouse-ul unui computer pot fi utilizate de la distanță de către un utilizator de la distanță de pe un dispozitiv secundar.

VNC funcționează pe un model client/server. O componentă de server este instalată pe computerul de la distanță (cel care se dorește a fi controlat), iar un vizualizator VNC, sau client, este instalat pe dispozitivul de pe care se dorește a se face controlul. Remote Framebuffer, sau RFB, este protocolul care guvernează formatul datelor care trec între client și server în sistemul VNC.

## WinSCP

WinSCP este un program de transfer de fișiere gratuit și open-source, care este utilizat pentru a transfera fișiere între două calculatoare, fie local, fie remote. Acesta este adesea folosit în combinație cu SSH (Secure Shell) pentru a permite transferul de fișiere între un calculator local și un server remote prin intermediul unei conexiuni SSH securizate. De asemenea, poate utiliza protocolul SFTP (Secure File Transfer Protocol) pentru transferul de fișiere, care este un protocol securizat care utilizează criptarea (în acest caz algoritmul AES-256) pentru a proteja datele în timpul transferului [2].

# Concepte de bază

Înaintea implementării propriu-zise este necesară înțelegerea unor concepte teoretice fundamentale ce aparțin de domeniul inteligenței artificiale. În acest capitol voi prezenta o parte dintre aceste concepte ce au fost folosite pentru a rezolva problema de față.

## ML

Învățarea automată este un concept care permite computerelor să învețe din exemple și experiențe în mod automat și să imite oamenii în luarea deciziilor, fără a fi programate în mod explicit. Este o ramură a inteligenței artificiale care folosește algoritmi și tehnici statistice pentru a învăța din date și a crea modele ce sunt folosite pentru a rezolva sarcini concrete.

În ciuda faptului că este o ramură a informaticii, învățarea automată este conceptual diferită de programarea tradițională. După cum este reprezentat și în figura 3.1, programarea tradițională presupune faptul că noi avem posibilitatea să definim clar regulile modelului pentru un set de date de intrare și obținem răspunsurile corecte. Pe de altă parte, în cadrul învățării automate, datele procesate sunt transmise unui algoritm de ML și produc o ieșire tipizată – regulile modelului [3].

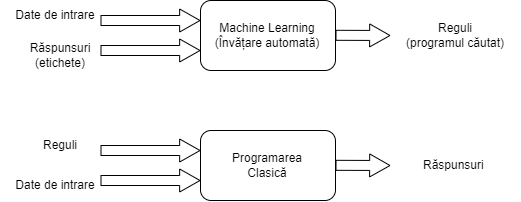


Fig. 3.1 Diferențele conceptuale dintre învățarea automată și programarea tradițională

Sistemele de învățare automată pot fi clasificate în patru mari categorii, corespunzător cu tipul de supraveghere efectuat în timpul antrenării:

1. Învățare supravegheată (Supervised learning) – Algoritmul este antrenat cu date care includ soluțiile dorite, numite etichete. Sarcinile tipice de învățare supravegheată sunt clasificarea (ex. recunoașterea animalului care apare într-o imagine dată) și regresia (ex. prezicerea valorii unei case având în vedere un set de caracteristici precum numărul de camere, vechimea clădirii,

locația). Procesul de antrenare al învățării supervizate implică de obicei divizarea setului de date într-un set de antrenament și un set de testare. Modelul este antrenat folosind setul de antrenament și este ajustat prin mărirea sau micșorarea ponderilor și a parametrilor până când predicțiile sale se apropie cât mai mult posibil de etichetele din setul de date de antrenament. Apoi, modelul este evaluat folosind setul de test, pentru a verifica dacă predicțiile sale sunt generalizabile la date noi care nu au fost incluse în setul de antrenament.

1. Învățare nesupravegheată (Unsupervised learning) - este un tip de învățare automată în care un model de învățare este antrenat pe un set de date de intrare fără etichete. Scopul este să se descopere structurile sau modelele din date, fără a se cunoaște etichetele asociate cu datele. Aplicații ale învățării nesupravegheate sunt: clusterizarea (ex. gruparea diferiților clienți corespunzător cu obiceiurile lor de cumpărare), reducerea dimensionalității

(ex. reducerea unui set de date neetichetat dintr-un spațiu cu dimensiuni mari într-un spațiu 2D sau 3D, pentru a permite reprezentarea și vizualizarea).

1. Învățare semisupravegheată (Semisupervised learning) - această clasă de algoritmi se ocupă cu un set de date de antrenament parțial etichetat. Un exemplu al acestei aplicații ML vine de la unele servicii de găzduire foto, cum ar fi Google Photos. În esență, partea nesupravegheată a algoritmului efectuează o sarcină de grupare, recunoscând și grupând imagini care arată aceeași persoană A. Apoi, cerând utilizatorului să spună cine este acea persoană A, algoritmul asociază eticheta dată tuturor imaginilor în care A este prezent.
2. Învățare prin consolidare (Reinforcement learning) – este cea mai unică categorie de algoritmi ML. Agentul (adică sistemul de învățare) poate observa mediul înconjurător (care poate fi real sau virtual), poate efectua acțiuni și obține recompense sau penalități (sub formă de recompense negative), în funcție de rezultat. Trebuie să învețe, de la sine, cea mai bună strategie pentru a obține cea mai mare recompensă în timp. Această strategie este denumită în mod obișnuit politică și constă în lista de acțiuni pe care agentul trebuie să le efectueze într-o situație dată. Exemple în acest sens pot fi găsite în robotică și în IA pentru jocuri video, în cazul în care un agent (respectiv un robot sau un NPC) folosește învățarea prin consolidare pentru a îmbunătăți o anumită sarcină, cum ar fi mersul pe jos, prinderea unui obiect sau câștigarea unui meci de joc.

Una dintre structurile de bază ale ML este rețeaua neuronală (uneori numită Artificial Neural Network), un sistem inspirat inițial de o reprezentare simplificată a arhitecturii bioligice a creierului. În zilele noastre, majoritatea NN-urilor sunt destul de diferite de definiția inițială dată de Warren McCulloch și Walter Pitts (1943) [4], dar rămân structura ideală pentru a aborda probleme complexe de ML.

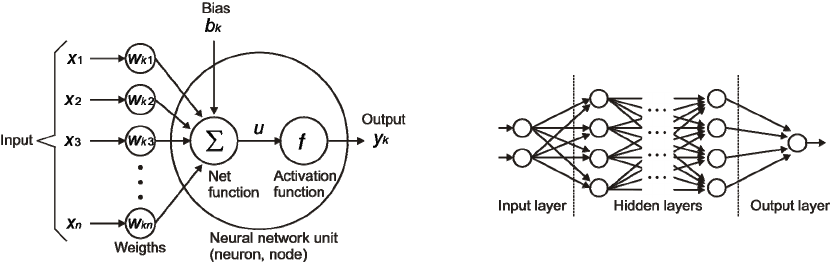


Fig. 3.2 Unitatea de bază a rețelei neuronale (neuron, nod) (McCulloch și Pitts)

Unitățile logistice de bază ale unei rețele neuronale sunt cunoscute ca neuroni (uneori denumiți neuroni artificiali, pentru a le distinge de omologii lor biologici). Ele sunt definite ca o structură de calcul simplă care primește intrări, le procesează și trimite rezultatele către unitatea următoare.

În termeni pur matematici, un neuron este containerul funcției de activare, adică regula conform căreia unitatea produce o ieșire determinată ca urmare a unui set specific de intrări. Alegerea acestor funcții este arbitrară, de obicei se bazează pe ipoteze făcute cu privire la tipul de caracteristici de antrenament și pe arhitectura modelului. Această funcție influențează modul în care sistemul aproximează comportamentul variabilelor sale dependente. În general, există patru funcții de activare importante: sigmoid, tangentă hiperbolică (tanh), ReLU și treaptă unitate (step).

## Funcția step

Funcția step (treaptă unitate) este una dintre cele mai simple funcții de activare, utilizată pe scară largă în primele arhitecturi ML, precum „Perceptronul” dezvoltat de Frank Rosenblatt (1957) și „ADALINE” realizată de Bernard Widrow și Ted Hoff (1960).

Funcția treaptă unitate sau treaptă Heaviside este o funcție discontinuă ce produce o ieșire binară sau discretă, în funcție de valoarea de intrare. Cele mai comune tipuri de funcții step sunt cea utilizată în Perceptron, definită în ecuația 3.1,

|  |  |
| --- | --- |
|  | (3.1) |

unde ieșirea este 0 dacă intrarea este strict mai mică decât 0 și 1 dacă intrarea este mai mare sau egală decât 0, și tipul implementat în ADALINE, definită în ecuația *3.2*,

|  |  |
| --- | --- |
|  | (3.2) |

unde ieșirea este -1 dacă intrarea este strict mai mică decât 0 și 1 dacă intrarea este mai mare sau egală decât 0. Această funcție este cunoscută și ca funcție semn.

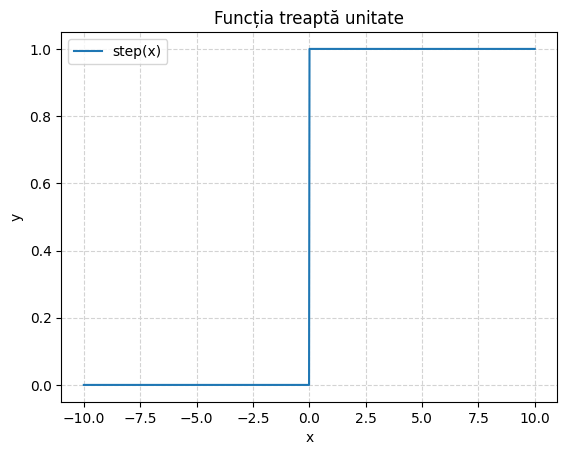


Fig. 3.3 Funcția treaptă unitate utilizată în arhitectura Perceptron

Deși această funcție a fost utilizată pe scară largă în primii ani ai rețelelor neuronale, prezintă multe probleme care, în zilele noastre, îi limitează utilizarea.

Primul dezavantaj major al funcției este faptul că nu este diferențiabilă în punctul de tăiere, adică la x = 0. Acest lucru duce la probleme în timpul antrenării rețelelor neuronale prin algoritmul de propagare inversă(eng. backpropagation) și algoritmul de coborâre în gradient(eng. gradient descent), care implică derivatele funcției de activare. Această indisponibilitate de a calcula derivatele în punctul de tăiere face dificilă optimizarea parametrilor rețelei.

Un alt dezavantaj important este că funcția treaptă unitate transformă orice intrare continuă sau discretă într-o singură valoare binară, limitând astfel capacitatea modelului de a aproxima funcții complexe. De exemplu, rețeaua neuronală nu ar putea distinge între două valori foarte apropiate, transformându-le în același rezultat de ieșire.

## Funcția sigmoid

Funcția sigmoid abordează cu succes problemele menționate mai sus cu funcția treaptă unitate. Graficul său este caracterizat de o curbă în formă de „S”, cunoscută și sub denumirea de curbă sigmoidă (vezi figura 3.4). În mijlocul acestui tip de funcții, cea mai populară pentru sarcinile ML este funcția logistică, definită în ecuația:

|  |  |
| --- | --- |
|  | (3.3) |

Ieșirea funcției tinde spre 0 când intrarea se apropie de −∞ și tinde spre 1 când intrarea se apropie de +∞.

Funcția sigmoid este diferențiabilă în fiecare punct, ceea ce face posibilă utilizarea ei în algoritmi de optimizare, cum ar fi coborârea în gradient, pentru a îmbunătăți performanța rețelei neuronale. În plus, funcția sigmoid este o funcție non-liniară, ceea ce permite rețelelor neuronale să modeleze relații complexe între intrări și ieșiri. De asemenea, se poate interpreta utilizarea funcției sigmoid ca o presupunere că distribuția probabilității de intrare urmează o distribuție Gaussiană, care apare frecvent în multe evenimente aleatorii. Această presupunere poate fi utilă pentru a ajuta rețelele neuronale să modeleze datele de intrare și să facă predicții mai precise.

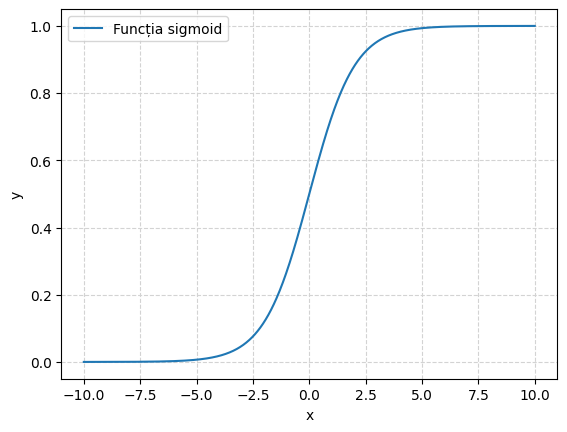


Fig. 3.4 Plotare a funcției sigmoid

## Funcția tangentă hiperbolică

|  |  |
| --- | --- |
|  | (3.4) |

Funcția tangentă hiperbolică este similară cu o funcție logistică, dar este redimensionată pentru a se potrivi cu intervalul de ieșire [−1, 1] în loc de [0, 1] (figura 3.5). Motivul principal pentru a utiliza o funcție tangentă hiperbolică în loc de o funcție sigmoidă este de a avea derivată mai mare. Totuși, în ciuda beneficiilor lor, ambele funcții prezintă așa-numita problemă a gradientului de dispariție [5] .

Problema gradientului de dispariție (în engleză "vanishing gradient problem") apare în rețelele neuronale profunde și se referă la faptul că gradientul (o măsură a ratei de schimbare a unei funcții) poate deveni foarte mic pe măsură ce se propagă prin rețeaua neuronală, ceea ce face ca actualizările ponderilor să fie minuscule și lent progresează în direcția corectă. Această problemă poate fi rezolvată prin utilizarea de funcții de activare care au derivate mai mari (cum ar fi ReLU).

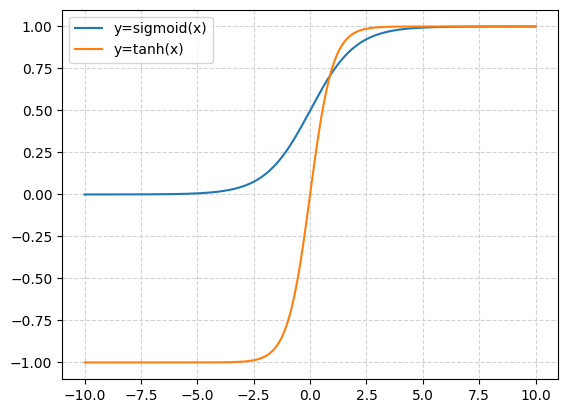


Fig. 3.5 Graficul funcției tangentă hiperbolică și a funcției sigmoid

## Funcția ReLU

Rectified Linear Unit (ReLU) este o funcție de activare utilizată în mod obișnuit în arhitecturile moderne ML. Funcția returnează 0 dacă intrarea este negativă, în timp ce pentru intrările pozitive returnează valoarea neschimbată. Este definită de ecuația *3*.*5*.

|  |  |
| --- | --- |
|  | (3.5) |

În plus, ReLU are o derivată constantă fie 0, fie 1, ceea ce înseamnă că derivata nu depinde de valorile de intrare. Această proprietate ajută la prevenirea problemei gradientului de dispariție, care poate să apară atunci când derivata funcției de activare devine foarte mică, determinând ca gradientul să se apropie de zero în timpul propagării inverse. Acest lucru poate face dificilă pentru rețea să învețe din straturile anterioare, ceea ce duce la o convergență mai lentă sau chiar la saturarea rețelei.

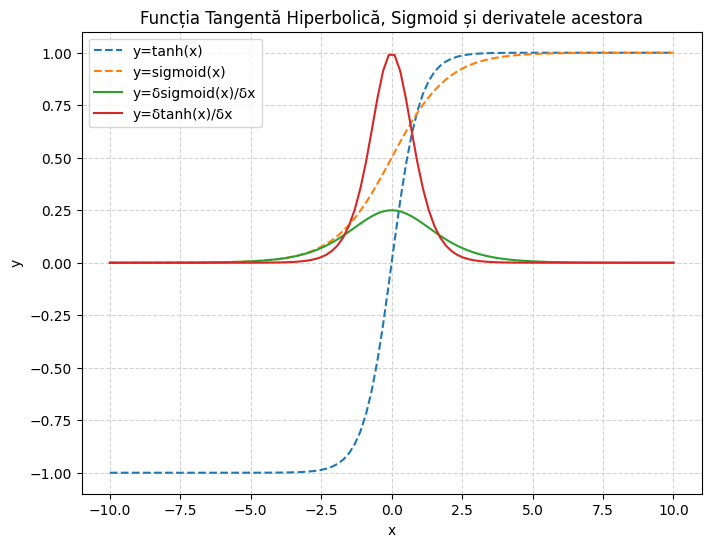


Fig. 3.6 Graficul funției tangentă hiperbolică, sigmoid și derivatele acestora

După cum se poate observa din figura 3.6, în cazul funcției sigmoid sau a funcției tangentă hiperbolică, derivata tinde spre zero atunci când intrarea x tinde spre .

În general, ReLU este mai puțin costisitoare din punct de vedere computațional decât funcțiile sigmoid și tangentă hiperbolică - figura. 3.7.

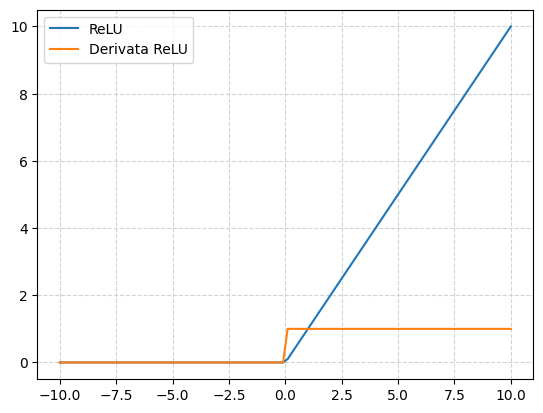


Fig. . Graficul funcției ReLU și derivata sa

Unele probleme cu această funcție de activare ar fi că nu este diferențiabilă pentru x = 0, ceea ce este cunoscut sub numele de problema Dying ReLU [6]. Deoarece jumătatea inferioară a curbei este constantă la y = 0, în timpul antrenamentului unii neuroni ReLU în esență „mor” și rămân inactivi, indiferent de intrarea furnizată. Dacă numărul de neuroni inactivi crește, performanța NN este grav afectată. Pentru a rezolva această problemă, este adesea folosită o variantă a acestei funcții: se numește Leaky ReLU și se comportă exact ca ReLU pentru valori pozitive, dar în loc să returneze y = 0 pentru toate intrările negative, are o pantă constantă cu un coeficientul a = 0, 01 - figura 3.8 .

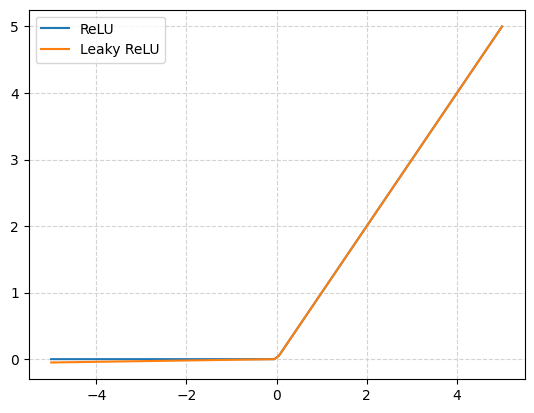


Fig. 3.8 Graficul funcției ReLU și Leaky ReLU

Cea mai mică și simplă rețea neuronală care poate fi proiectată constă dintr-un singur neuron și funcția sa de activare, dar o astfel de arhitectură este evident limitată și nu poate fi utilizată în nicio aplicație practică. Prin urmare, majoritatea NN-urilor conțin numeroși neuroni (de la zeci la mii) grupați în straturi.

Un strat este un set de neuroni care primește intrări ponderate, le procesează prin funcția de activare și apoi transmite aceste valori ca ieșire la stratul următor. Un strat este de obicei uniform, adică toți neuronii săi conțin un singur tip de funcție de activare, astfel încât să poată fi ușor clasificat și comparat cu alte părți ale rețelei. Primul și ultimul strat dintr-o rețea sunt numite straturi de intrare și, respectiv, straturi de ieșire, în timp ce toate straturile dintre acestea sunt definite ca straturi ascunse.

## Funcția de optimizare

Spre deosebire de funția de activare care este aplicată la ieșirile neuronilor pentru a produce o anumită activare în cadrul rețelei, o metodă de optimizare este folosită pentru a ajusta parametrii rețelei neuronale (greutățile și rata de învățare) în timpul procesului de antrenare cu scopul de a minimiza o funcție de cost. Aceste funcții de optimizare încearcă să găsească un minim local sau global al funcției de cost prin ajustarea parametrilor rețelei neuronale. Exemple de funcții de optimizare includ Stochastic Gradient Descent (SGD), Adam sau Adagrad și multe altele.

În cadrul acestei lucrări este utilizată funcția de coborâre a gradientului stocastic (SGD) care s-a potrivit cel mai bine pe cele trei arhitecturi neuronale fiind și optimizatorul inițial al acestora. A mai fost testat optimizatorul Adam, dar rezultatele obținute nu au fost cele așteptate, mai ales pentru arhitectura Alexnet a cărei acuratețe a fost alterată.

Coborârea gradientului este un algoritm iterativ, care pornește de la un punct aleatoriu al unei funcții și se deplasează în panta ei în pași, în funcție de rata de învățare, până când ajunge la punctul cel mai de jos al funcției respective [7]. Diferența cheie a SGD, în comparație cu coborârea gradientului standard, este că doar o singură parte din date este utilizată pentru a calcula pasul, iar datele sunt alese aleatoriu la fiecare pas.

## Funcția de cost

În ML, o funcție de cost este un parametru important care determină cât de bine funcționează un model de învățare automată pentru un anumit set de date. Acesta calculează diferența dintre valoarea așteptată(eticheta reală) și valoarea prezisă și o reprezintă ca un singur număr real [8]. În general, funcțiile de cost se clasifică în trei categorii: funcții de cost de regresie, de clasificare binară și de clasificare multiclasă. În contextul de față este realizată o clasificare multiclasă, iar funția folosită este entropia încrucișată categorică. Entropia încrucișată se bazează pe ideea de entropie din teoria informației și calculează numărul de biți necesari pentru a reprezenta sau transmite un eveniment dintr-o distribuție de probabilitate în comparație cu o altă distribuție de probabilitate [9].

## Învățare profundă pentru recunoașterea vorbirii

Termenul Deep Learning (DL) se referă la o subclasă de metode ML care pot învăța din ierarhii complexe de caracteristici neliniare sau pot detecta dependențe lungi de timp neliniare în datele secvențiale [10]. Pentru a aproxima modele și structuri profunde, o rețea neuronală profundă (DNN) are mai multe straturi ascunse de natură diferită, în funcție de aplicația specifică.

În recunoașterea automată a vorbirii, rețelele neuronale convoluționale (CNN) și rețelele neuronale recurente (RNN) sunt cele mai populare și eficiente arhitecturi de învățare profundă. Acestea sunt caracterizate de straturi personalizate concepute special pentru a învăța date spațiale și secvențiale. RNN-urile sunt de obicei mai bune decât CNN-urile pentru sarcini de recunoaștere a vorbirii în care intrarea audio este lungă și constă din mai mult de un cuvânt, datorită capacității lor de a stoca informații secvențiale temporale. În schimb, CNN-urile sunt mai bune în sarcinile de recunoaștere a imaginii, unde învățarea modelului spațial al fiecărui grup de pixeli este esențială pentru a înțelege natura imaginii și a returna rezultatul corect [11].

În această lucrare, problema de recunoaștere a vorbirii este tratată ca o problemă de recunoaștere a imaginii, deoarece toate caracteristicile extrase din fluxul audio pot fi reprezentate ca o singură imagine în tonuri de gri cu dimensiunea 32 × 98. Această abordare a arătat cele mai bune rezultate în ceea ce privește acuratețea detectării unui cuvânt, precum și reducerea timpului necesar pentru efectuarea deducției. Această abordare a fost exploatată pe scară largă în literatură și a fost utilizată cu succes de cercetători pentru a realiza recunoașterea automată a vorbirii.

### CNN

Rețelele neuronale convoluționale (CNN), numite și ConvNets, sunt un tip de rețele neuronale profunde special concepute pentru a procesa date care vin sub forma de matrici sau tablouri multiple. Aceste date pot fi, de exemplu, imagini, semnale audio, secvențe de limbaj sau spectrograme audio, sau chiar imagini video sau imagini volumetrice.

Structura unei rețele neuronale convolutionale este formată dintr-o serie de straturi, care sunt organizate în două tipuri principale de straturi:

* Straturile de convoluție: unitățile neuronale dintr-un astfel de strat sunt organizate în hărți de trăsături (feature maps), în cadrul cărora fiecare unitate este conectată la câmpuri receptive locale în harta de trăsături a stratului anterior.

Această conexiune se face prin intermediul unui set de ponderi numit bancă de filtre (sau filtru kernel), care este folosit de toate unitățile unei hărți de trăsături, dar hărțile de trăsături diferite din același strat folosesc bănci de filtre diferite. Apoi, rezultatul acestei convoluții discrete este procesat cu o funcție de activare non-liniară, cum ar fi ReLU sau Leaky ReLU.

* Straturile de pooling: deoarece straturile de convoluție necesită o cantitate mare de putere de calcul și memorie RAM (în special în timpul antrenării), sarcina acestor straturi este de a reduce dimensiunea fiecărei hărți de trăsături prin sub-eșantionare(sub-sampling). Formal, rolul lor este de a uni semantic o porțiune de trăsături similare într-una singură, astfel încât informațiile inutile sunt eliminate pe măsură ce straturile progresează.

După aceste straturi de convoluție și de pooling, hărțile de trăsături sunt transformate într-un tablou unidimensional pentru a fi introduse într-o rețea neuronală obișnuită, feedforward (informația circulă într-o singură direcție). Aceasta este de obicei compusă din câteva straturi complet conectate (cu o funcție de activare non-liniară), plus stratul de ieșire final care produce predicțiile estimate. În cazul problemelor de clasificare multiplă funția de activare a stratului final este, în general, Softmax care transformă valorile de ieșire într-o distribuție de probabilitate.

## Modele investigate

În această secțiune sunt prezentate arhitecturile model selectate pentru a fi antrenate, evaluate și utilizate pe placa integrată – Raspberry Pi 4. Trebuie remarcat faptul că, având în vedere diferențele în natura și dimensiunea datelor de intrare, nu a fost posibilă implementarea CNN-urilor fără modificări și îmbunătățiri pentru a le face potrivite pentru această aplicație specifică.

Aceste variații sunt detaliate într-o secțiune ce va urma, în timp ce aici modelele sunt analizate și revizuite așa cum au fost proiectate inițial.

### VGG16

Arhitectura VGG16 a fost introdusă pentru prima dată în 2015, în cadrul lucrării de cercetare "Very Deep Convolutional Networks for Large-Scale Image Recognition", scrisă de Karen Simonyan și Andrew Zisserman de la Universitatea din Oxford [12].

Arhitectura VGG16 a obținut performanțe remarcabile în domeniul recunoașterii de imagini și a fost utilizată în mai multe competiții de referință, printre care se numără:

1. ImageNet Large Scale Visual Recognition Challenge (ILSVRC) din anii 2014 și 2015, unde a obținut cel mai bun rezultat la clasificarea imaginilor.
2. Microsoft Common Objects in Context (COCO) din anul 2015, unde a fost folosită la detectarea obiectelor în imagini.

În ceea ce privește performanțele, arhitectura VGG16 a stabilit un nou standard în domeniul recunoașterii de imagini, cu o precizie (eng. accuracy) de 92,7% pe setul de date ImageNet în anul 2014. De asemenea, VGG16 a obținut rezultate impresionante și în alte seturi de date, precum CIFAR-10 și CIFAR-100.

Arhitectura VGG16 a fost o inovație în domeniul rețelelor neuronale convoluționale, cu o arhitectură profundă de 16 straturi care a permis obținerea de caracteristici complexe ale imaginilor și o învățare mai eficientă a acestora prin backpropagation. Deși VGG16 a fost depășită în performanță de alte arhitecturi ulterioare, aceasta rămâne o inovație importantă în domeniul viziunii artificiale.

În tabelul 1 sunt ilustrate diferite arhitecturi ale rețelei VGG [13]. Toate configurațiile urmăresc design-ul tradițional și diferă doar în adâncime: de la 11 straturi în rețeaua A (8 convoluționale și 3 complet conectate) la 19 straturi în rețeaua E (16 convoluționale și 3 complet conectate).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ConvNet Configuration | | | | | |
| A | A-LRN | B | C | D | E |
| 11 weight  layers | 11 weight  layers | 13 weight  layers | 16 weight  layers | 16 weight  layers | 19 weight  layers |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Input (224 x 224 RGB image) | | | | | |
| Conv3-64 | Conv3-64  LRN | Conv3-64  Conv3-64 | Conv3-64  Conv3-64 | Conv3-64  Conv3-64 | Conv3-64  Conv3-64 |
| maxpool | | | | | |
| Conv3-128 | Conv3-128 | Conv3-128  Conv3-128 | Conv3-128  Conv3-128 | Conv3-128  Conv3-128 | Conv3-128  Conv3-128 |
| maxpool | | | | | |
| Conv3-256  Conv3-256 | Conv3-256  Conv3-256 | Conv3-256  Conv3-256 | Conv3-256  Conv3-256  Conv1-256 | Conv3-256  Conv3-256  Conv3-256 | Conv3-256  Conv3-256  Conv3-256  Conv3-256 |
| maxpool | | | | | |
| Conv3-512  Conv3-512 | Conv3-512  Conv3-512 | Conv3-512  Conv3-512 | Conv3-512  Conv3-512  Conv1-512 | Conv3-512  Conv3-512  Conv3-512 | Conv3-512  Conv3-512  Conv3-512  Conv3-512 |
| maxpool | | | | | |
| FC-4096 | | | | | |
| FC-4096 | | | | | |
| FC-1000 | | | | | |
| Soft-max | | | | | |

Tabel 1 Diferite configurații VGG

### AlexNet

Când a fost publicată în 2012 de către Alex Krizhevsky și colaboratorii săi, această rețea neuronală convoluțională a depășit cu o marjă mare performanța anterioară. A câștigat atât sarcinile de clasificare, cât și de localizare ale concursului ILSVRC-2012 și este considerată printre cele mai influente arhitecturi în aplicarea învățării profunde în viziunea computerizată (computer vision).

Arhitectura modelului constă din opt straturi de învățare - cinci convoluționale și trei complet conectate (numite și straturi dense).

Stratul de intrare acceptă imagini RGB cu dimensiunea 224 × 224 (tensori 3D de formă 224 × 224 × 3), iar ieșirea este proiectată pentru a produce o distribuție pentru 1000 de clase diferite. Potrivit autorilor, cea mai importantă caracteristică a acestei rețele este utilizarea de funcții de activare ReLU ceea ce a permis modelului să se antreneze de câteva ori mai repede decât echivalentele cu unități tanh. O altă trăsătură importantă a acestui CNN este utilizarea de filtre kernel mai largi în primele straturi, în special 11 × 11 și 5 × 5.

Pentru a preveni supraadaptarea (overfitting-ul), au fost folosite straturi de dropare (Dropout) cu o probabilitate de 0,5 înainte de primul și al doilea strat dense [14]. În cele din urmă, instruirea a fost efectuată folosind un subset al ImageNet - un set de date care constă din peste 15 milioane de imagini de înaltă rezoluție etichetate aparținând aproximativ 22 000 de clase. A fost utilizat subsetul din ILSVRC-2010 deoarece a fost singura versiune pentru care erau disponibile etichetele setului de testare.

|  |  |  |  |
| --- | --- | --- | --- |
| Tipul stratului | Dimensiunea de ieșire | Dimensiune filtru/pas | Adâncime  (număr kerneluri) |
| Input | 224x224x3 | - | - |
| Conv2D | 55x55x96 | 11x11/4x4 | 96 |
| MaxPool | | | |
| Conv2D | 27x27x256 | 5x5/1x1 | 256 |
| MaxPool | | | |
| Conv2D | 13x13x384 | 3x3/1x1 | 384 |
| Conv2D | 13x13x384 | 3x3/1x1 | 384 |
| Conv2D | 13x13x256 | 3x3/1x1 | 256 |
| MaxPool | | | |
| Flatten | | | |
| Dense | 1x1x4096 | - | 4096 |
| Dense | 1x1x4096 | - | 4096 |
| Dense | 1x1x1000 | - | 1000 |
| Softmax | | | |

Tabel Dimensiuni arhitecturale AlexNet

După trecerea printr-o convoluție sau un strat de pooling maxim [15], dimensiunea imaginii intermediare de ieșire este produsă de relația *3*.*6*:

|  |  |
| --- | --- |
|  | (3.6) |

În această ecuație: n – dimensiunea dată, p – padding-ul adăugat de strat, f – numărul de kernel-uri, s – strides(pași), k – dimensiune kernel.

De exemplu pentru primul strat convoluțional parametrii vor fi - n=224, p=0, f=11, s = 4, k = 96, iar dimensiunile produse după acesta – (55,55,96).

După primul strat de convoluție urmează un strat MaxPooling, unde parametrii vor fi – n = 55, p = 0, f = 3, s = 2, k = 256, iar dimensiunile produse de acesta – (27,27,256). Se continuă și pentru următoarele straturi în aceeași manieră.

### SqueezeNet

SqueezeNet este o arhitectură de rețea neurală convoluțională (CNN) creată de Iandola și colaboratorii săi în 2016. Această arhitectură a fost dezvoltată cu scopul de a obține o acuratețe comparabilă cu modelele de referință precum AlexNet, dar cu un număr mult mai mic de parametri și o dimensiune mai mică a modelului (sub 0,5 MB). [16]

Pentru a atinge acest obiectiv, autorii au urmat trei strategii principale:

1. Au înlocuit filtrele 3x3 cu filtre 1x1 pentru a reduce numărul de parametri.
2. Au redus numărul de canale de intrare la filtre 3x3 folosind straturi de comprimare(squeeze).
3. Au mutat reducerea eșantionării(downsample) la sfârșitul rețelei pentru a păstra hărți mari de activare în straturile convoluționale și pentru a maximiza acuratețea cu un număr mic de parametri.

Pentru a implementa aceste strategii, a fost creat modulul Fire, care este un bloc de construcție al rețelei ce conține mai multe straturi și permite o implementare ușoară a rețelei. Modulul Fire este alcătuit dintr-o parte de comprimare "squeeze" (un strat de convoluțional 1 x 1) și o parte de "expand" (un strat de convoluțional format dintr-o combinație de filtre de 1 x 1 și 3 x 3). Acest modul poate fi ajustat prin intermediul a trei hiperparametri, care reprezintă numărul de filtre din fiecare strat: s1x1, e1x1 și e3x3. De asemenea, s1x1 trebuie să fie mai mic decât (e1x1 + e3x3), astfel încât stratul de "squeeze" limitează numărul de canale de intrare în filtrele de 3 x 3.

Arhitectura SqueezeNet conține 2 straturi de convoluție la capetele rețelei și 8 module Fire în mijloc. Această rețea primește imagini de dimensiune fixă 224 x 224 x 3, iar adâncimea modulelor Fire crește treptat către stratul de ieșire, care produce un vector de predicție peste 1000 de clase. O altă diferență față de modelele tradiționale de CNN este lipsa de straturi complet conectate, ceea ce a impus utilizarea unui strat de pooling 2D (AvgPool în tabelul 3) care calculează media valorilor dintr-o regiune a imaginii pentru a genera vectorul de ieșire dorit. În ciuda faptului că SqueezeNet are un număr mult mai mic de parametri decât modelele de referință, precum AlexNet, acesta poate atinge o acuratețe comparabilă cu acestea.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tipul Stratului | Dimensiunea de ieșire | Dimensiune filtru/pas | Depth | s1x1 | e1x1 | e3x3 |
| Input | 224x224x3 | - | - | - |  |  |
| Conv2D | 111x111x96 | 7x7/2x2 | 96 | - | - | - |
|  |  | MaxPool |  |  |  |  |
| Fire1 | 55x55x128 | - | - | 16 | 64 | 64 |
| Fire2 | 55x55x128 | - | - | 16 | 64 | 64 |
| Fire3 | 55x55x256 | - | - | 32 | 128 | 128 |
|  |  | MaxPool |  |  |  |  |
| Fire4 | 27x27x256 | - | - | 32 | 128 | 128 |
| Fire5 | 27x27x384 | - | - | 48 | 192 | 192 |
| Fire6 | 27x27x384 | - | - | 48 | 192 | 192 |
| Fire7 | 27x27x512 | - | - | 64 | 256 | 256 |
|  |  | MaxPool |  |  |  |  |
| Fire8 | 13x13x512 | - | - | 64 | 256 | 256 |
| Conv2D | 13x13x1000 | 1x1/1x1 | 1000 | - |  | - |
| AvgPool | 1x1x1000 | 13x13/1x1 | - | - |  |  |

Tabel 3 Dimensiuni arhitecturale SqueezeNet

## Extragerea caracteristicilor

În învățarea automată, termenul "caracteristică" se referă la un tip particular de informație pe baza căruia este antrenat un model. Procesul care transformă datele brute de intrare (de exemplu, eșantioane audio, imagini, videoclipuri) într-un format care poate fi procesat de o arhitectură de învățare automată se numește "extragerea caracteristicilor". În contextul recunoașterii automate a vorbirii, extragerea caracteristicilor se realizează prin eșantionarea unei forme de undă acustice în cadre (în general între 10 și 32 de milisecunde în lungime, în funcție de metoda folosită), care sunt apoi convertite în caracteristici spectrale și procesate pentru a elimina cât mai multe informații inutile. Rezultatul este un tensor care conține o reprezentare compactă a semnalului vocal dat.

### Cepstru

Cepstrul este o metodă matematică utilizată pentru analiza semnalelor și a spectrelor lor de frecvență. Termenul "cepstru" provine de la cuvintele "spectrum" și "cepstral", acesta din urmă referindu-se la transformarea unei transformate Fourier a unui semnal.

Cepstrul poate fi considerat o transformare de două ori Fourier a unui semnal, adică inversul transformatei Fourier a logaritmului transformatei Fourier a semnalului inițial. Rezultatul este o reprezentare a semnalului într-un domeniu numit domeniul cepstrului, care este util pentru analizarea semnalului în diferite moduri [17].

Cepstrul este utilizat într-o varietate de aplicații, cum ar fi prelucrarea vorbirii, analiza semnalului radar, analiza de spectru de frecvență și analiza semnalului biologic. Această tehnică poate fi utilizată pentru a extrage caracteristici utile din semnale și pentru a le analiza în mod eficient.

### Short Time Fourier Transform

Transformata Fourier de scurtă durată (*eng.* Short Time Fourier Transform – STFT) este o extensie a transformatei Fourier care poate fi utilizată pentru analiza semnalelor nestaționare, precum semnalul vocal. Aceasta presupune aplicarea de ferestre de analiză asupra cadrelor segmetate ale semnalului vocal, reflectând proprietățile variabile în timp ale formei de undă. STFT generează o matrice 2D de magnitudini (modulul spectrului unui semnal) și faze (cum semnalul este defazat în timp față de un semnal de referință), reprezentând spectrul semnalului în funcție de timp și frecvență.

Definiția transformatei Fourier de scurtă durată este dată de relația:

|  |  |  |
| --- | --- | --- |
|  |  | (3.7) |

Unde :

- x(m) reprezintă semnalul de intrare

- w(m) este fereastra de analiză, invariantă în timp și deplasată cu n eșantioane

STFT a semnalului, notată cu X(n,ω) este o funcție de două variabile, unde n reprezintă indicele în timp discret și ω, frecvența. Prin eșantionarea variabilei de frecvență ω la un spațiu al frecvențelor uniform distanțate, se poate obține și o discretizare a transformatei Fourier de scurtă durată.

Mărimea pătratului STFT este cunoscută ca reprezentarea spectrogramei timp-frecvență a semnalului.

### Mel Frequency Cepstral Coefficients

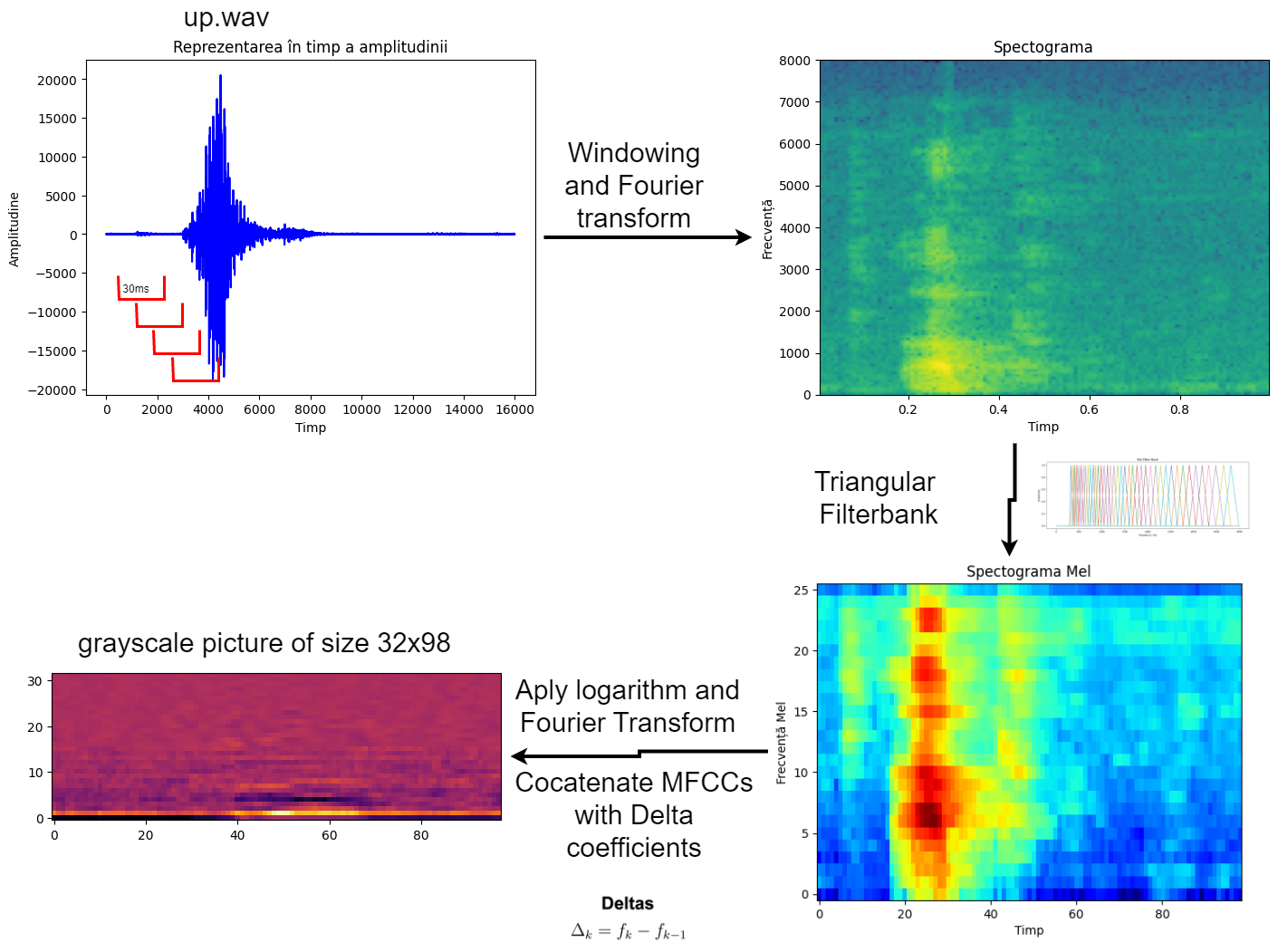


Fig. 3.9 Procesul de extragere a caracteristicilor

MFCC sau Mel Frequency Cepstral Coefficients reprezintă o tehnică folosită în prelucrarea semnalelor audio pentru a extrage caracteristici relevante. Acești coeficienți sunt o reprezentare compactă a spectrului de frecvență al semnalului audio. Coeficienții cepstrali de frecvență Mel conțin informații despre schimbările de frecvență în diferite benzi spectrale. Dacă un coeficient cepstral are o valoare pozitivă, majoritatea energiei spectrale este concentrată în regiunile de frecvență joasă, iar dacă are o valoare negativă înseamnă că majoritatea energiei este concentrată în frecvențele înalte.

Procesul de extracție a acestor caracteristici se realizeză în șase etape [18]:

În primul rând, semnalul audio este împărțit în cadre scurte de 20-40 milisecunde. Acest lucru este necesar pentru că frecvențele dintr-un semnal se schimbă în timp, deci nu are sens să facem transformata Fourier pe întregul semnal pentru că am pierde informații importante despre frecvență.

Pentru o rată de eșantionare de 8kHz (ceea ce înseamnă că luăm 8000 de eșantioane pe secundă pentru a reprezenta semnalul analogic), o dimensiune a cadrului de 0.03s = 3ms și un pas de 0.01s = 1ms (avem suprapunere între cadre), vom avea (1-0.03+0.01)/0.01 = 98 cadre, fiecare cu 0.03\*8000 = 240 eșantioane.

Ferestruirea (Windowing) este o tehnică folosită pentru a reduce scurgerile spectrale (leakage) în timpul calcului transformatei Fourier rapide (FFT). Aceste erori sunt cauzate de limitările impuse de eșantionarea finită și durata finită a semnalului, precum și de utilizarea unei ferestre de eșantionare. Aceste artefacte pot fi vizibile sub forma unor benzi laterale sau "loburi" în spectrul semnalului și pot fi confundate cu componente reale de frecvență.

Funcția de ferestruire este o funcție matematică care are valori nule în afara unei anumite ferestre de eșantionare și valori pozitive sau negative în interiorul acestei ferestre. Atunci când se înmulțește semnalul de intrare cu funcția de ferestruire, se produce o "tăiere" a semnalului în afara ferestrei, ceea ce reduce energia semnalului în acele porțiuni și minimizează efectul de "leakage". Câteva exemple folosite sunt: Hamming, Hanning, Gaussian, Blackman etc.

După aceea, se calculează transformata Fourier discretă în N puncte pe fiecare cadru pentru a obține spectrul de frecvență. Această transformare este numită și transformata Fourier rapidă în timp scurt (STFT). Numărul de puncte este de obicei 256 sau 512 (nfft=512). Apoi este calculat spectrul de putere ( periodograma – o estimare a densității spectrale a unui semnal). Periodograma se obține prin calcularea pătratului modulului transformatei Fourier a semnalului și împărțirea acestuia la lungimea semnalului.

În a patra etapă, se aplică o bancă de filtre Mel spațiate pentru a obține o reprezentare a semnalului audio în funcție de bandele de frecvență Mel. Aceasta etapă implică aplicarea a 20-40 filtre triunghiulare Mel spațiate în mod uniform pentru a obține 40 de coeficienți. Acești coeficienți reprezintă energia din fiecare filtru Mel.

Bancul de filtre triunghiulare vine sub forma a 40 de vectori de lungime 257 (presupunând că numărul de puncte ales în pasul anterior este 512). Fiecare vector este în mare parte zero, dar avem valori diferite de zero pentru o anumită secțiune a spectrului. Pentru a calcula energiile bancului de filtre, înmulțim fiecare filtru cu spectrul de putere, apoi adunăm coeficienții. Odată ce aceasta este efectuată, obținem 40 de numere care ne oferă o indicație despre câtă energie a fost în fiecare filtru banc.

Pentru a obține un filtru banc:

a) trebuie mai întâi să alegem o frecvență inferioară și superioară. Valorile potrivite sunt - 300Hz pentru frecvența inferioară și 4000Hz pentru frecvența superioară.

b) convertim frecvențele superioară și inferioară în Mels. În cazul nostru, 300Hz este de 401.25 Mels și 8000Hz este de 2834.99 Mels. Conversia de frecvență în MEL și invers este realizată folosind formulele:

|  |  |
| --- | --- |
|  | . |

|  |  |
| --- | --- |
|  | . |

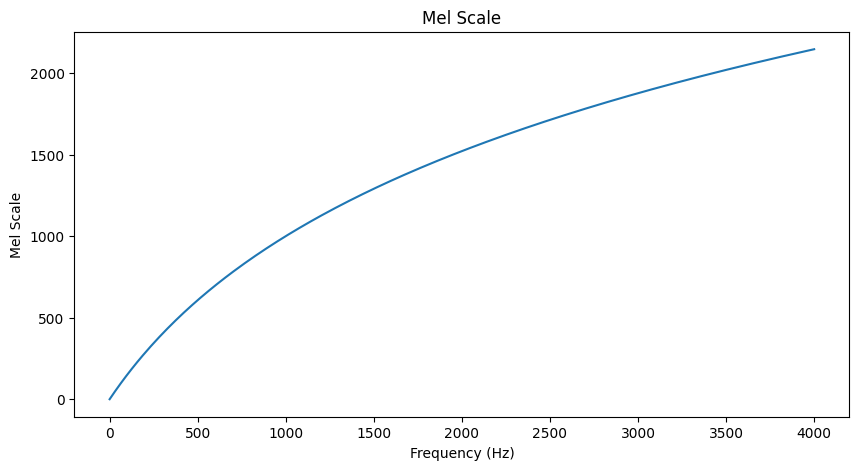


Fig. . Scala Mel

c) Acest lucru ne oferă 40 de coeficienți (în funcție de cerințe, poate fi orice număr), între intervalul selectat.

d) Acești coeficienți sunt apoi convertiți înapoi în hertzi

e) Și filtrul banc este trasat folosind aceste puncte - figura 3.11!

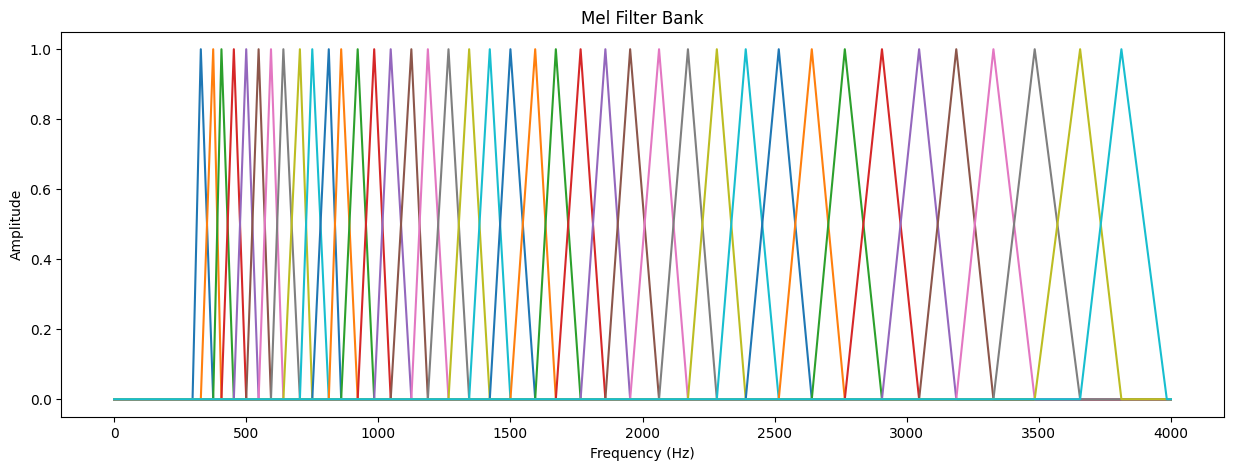


Fig. . Banc de filtre Mel

În etapa următoare, se aplică funcția logaritm valorilor spectrogramelor filtrate pentru a obține energiile filtrului Mel într-o formă logaritmică.

În final, coeficienții Filter Bank sunt decorelați prin aplicarea transformatei cosinus discrete (DCT) pentru a obține coeficienții Mel Frequency Cepstral. Acești coeficienți descriu doar încadrarea spectrală a puterii unui singur cadru de semnal audio.

În general, coeficienții MFCC descriu doar spectrul de putere al unui singur cadru al unui semnal audio, în timp ce coeficienții delta descriu variația acestor coeficienți în timp. Coeficienții delta sunt pur și simplu diferența dintre caracteristicile găsite în ferestrele individuale ale spectrogramei și pot fi considerate - într-un mod foarte simplificat - ca prima derivată a spectrogramei, deoarece descriu modul în care informația spectrală se modifică în timp.

# Echipamente folosite

Un sistem încorporat (eng. embedded system) este un sistem de calcul specializat care este proiectat să îndeplinească un set specific de funcții sau sarcini. De obicei, aceste sisteme sunt integrate în alte produse sau dispozitive și sunt proiectate să ruleze cu consum redus de energie și cu o dimensiune fizică mică. Sistemele încorporate sunt utilizate într-o gamă largă de aplicații, inclusiv în industria auto, în dispozitive medicale, în echipamente de comunicare, în sisteme de control industrial și în multe alte domenii. Acestea pot fi proiectate cu ajutorul unor microcontrolere, microprocesoare sau chiar cu circuite integrate specializate.

În această lucrare de licență s-a urmărit și crearea unui ansamblu electronic care să formeze un sistem încorporat de comandă. În cele ce urmează sunt prezentate componentele acestui sistem.

## Raspberry Pi 4

Un SBC este un computer complet funcțional, alcătuit dintr-o singură placă de circuit imprimat. Pe această placă sunt integrate toate componentele necesare funcționării unui computer, precum procesorul, memoria RAM, memoria de stocare, interfețele de comunicație (precum USB sau Ethernet). Un astfel de computer poate fi utilizat într-o varietate de aplicații, cum ar fi roboți, sisteme de control, monitoare, sau alte proiecte în care spațiul și consumul de energie sunt limitate. Exemple de single board computers sunt Raspberry Pi sau Arduino.

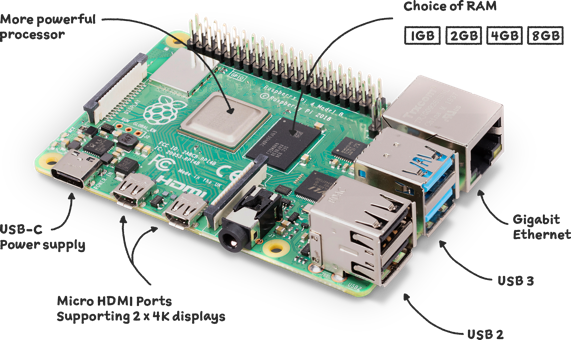


Fig. 4.1 Raspberry Pi 4

Raspberry Pi 4 model B este o placă de dezvoltare compactă, bazată pe arhitectura ARM, proiectată pentru a fi utilizată într-o varietate de proiecte hardware și software. Plăcuța Raspberry Pi 4 model B are următoarele specificații:

* Procesor Broadcom BCM2711 quad-core ARM Cortex-A72 (64-bit) cu o frecvență de 1.5 GHz
* 2GB, 4GB sau 8GB RAM, în funcție de varianta aleasă
* Conexiune Gigabit Ethernet
* Conexiune Wi-Fi 802.11ac dual-band și Bluetooth 5.0
* 2 porturi USB 2.0 și 2 porturi USB 3.0
* 2 porturi Micro-HDMI pentru conectarea la un monitor sau la televizor
* Slot pentru card microSD pentru stocare și boot
* Conectorul de alimentare USB-C de 5V / 3A
* Portul de afișaj MIPI DSI (Mobile Industry Processor Interface Display Serial Interface)
* Portul de camera MIPI CSI (Mobile Industry Processor Interface Camera Serial Interface)
* GPIO de 40 de pini pentru conectarea senzorilor și a altor componente electronice

Placa Raspberry Pi 4 suportă o varietate de sisteme de operare precum: Raspberry Pi OS (o distribuție Linux bazată pe Debian), Ubuntu, Kali Linux, Windows on Raspberry și multe altele.

Raspberry Pi 4 este o plăcuță puternică și versatilă, ideală pentru persoanele care doresc să construiască proiecte proprii, să invețe programare sau să-și transforme televizorul într-un smart TV. De asemenea, datorită dimensiunilor sale mici și a consumului redus de energie, poate fi utilizată în diverse proiecte de automatizare și de control al dispozitivelor inteligente din casă.

## Microfon Trust GXT212 Mico

Microfonul Trust GXT212 Mico este un microfon cardioid, ceea ce înseamnă că este mai sensibil la sunetele care vin din fața lui decât la sunetele care vin din spate sau din lateral. Acest lucru face microfonul potrivit pentru înregistrarea vocii sau a altor sunete în medii zgomotoase, unde ar fi mai dificil să se izoleze sunetul dorit de zgomotul de fond.

Specificațiile pentru microfonul Trust GXT212 Mico:

Frecvență de răspuns: 50 Hz - 16.000 Hz

Lungime cablu: 1.8 m

Conector: Jack de 3.5 mm

## Acumulator extern Romoss OEM Solo 5

În cadrul sistemului, alimentarea este asigurată de o baterie externă ce poate furniza o tensiune de 5V și un curent de 2.1A sau 1A.

Specificațiile acestui acumulator sunt:

* Capacitate: 10000mAh
* Tip baterie: Li-ION
* Intrare: Micro USB DC 5V/2.1A
* Ieșiri: USB 1: DC 5V/2.1A, USB 2: DC 5V/1A
* Dimensiuni: 14,2 x 6,5 x 2,2 cm
* Greutate: 296g
* Indicator LED pentru nivelul de încarcare al bateriei
* Tehnologie de protecție multiplă pentru siguranța dispozitivelor conectate

## Componente electronice pasive

O componentă electronică pasivă este un component electronic care nu are capacitatea de a furniza sau amplifica semnale electrice, ci doar reacționează la aceste semnale. În general, componente electronice pasive sunt cele care nu conțin surse de energie proprii și nu pot amplifica semnalele electrice, dar pot să ofere rezistență, capacități sau inductanță.

### Breadbord

Un breadboard (placă de conexiuni) este un dispozitiv care permite interconectarea ușoară a componentelor electronice fără a fi necesară utilizarea unui fier de lipit.

Breadbord folosit:

Dimensiuni: 16.5 x 5.5 cm - Acestea sunt dimensiunile plăcii de conexiuni. Aceasta este o dimensiune medie pentru un breadboard, oferind suficient spațiu pentru a conecta mai multe componente electronice.

Numărul de găuri: 830 - Acesta este numărul total de găuri disponibile pe placa de conexiuni, ceea ce permite utilizatorilor să conecteze un număr mare de componente și să creeze circuite complexe.

Linii de alimentare: 2 - Placa de conexiuni are două linii de alimentare, una pentru alimentare și alta pentru masă (GND). Acestea sunt de obicei marcate cu roșu și negru, respectiv.

Distribuția găurilor: găurile sunt distribuite în benzi verticale și orizontale, care permit utilizatorilor să creeze conexiuni între componente aflate la distanțe diferite.

Compatibilitate: Breadboard-ul este compatibil cu o gamă largă de componente electronice, inclusiv rezistențe, condensatoare, diode, tranzistoare, LED-uri și multe altele.

### Fire Dupont

Firele Dupont sunt o formă populară de cabluri jumper utilizate pentru a face conexiuni rapide și temporare între componente electronice și plăcile de conexiuni.

Culorile firelor: Firele Dupont cu 10 pini sunt disponibile în diferite culori, fiecare culoare reprezentând o anumită funcție. De exemplu, roșu este adesea folosit pentru alimentare, negru pentru masă, iar alte culori pot fi folosite pentru semnale de date sau pentru a identifica diferite componente ale circuitului.

### Rezistor

Un rezistor este o componentă electronică pasivă care are rolul de a opune rezistență la curentul electric ce trece prin el. Un rezistor are două terminale și este construit dintr-un material cu o rezistență electrică bine definită, cum ar fi carbonul sau metalul. Rezistența electrică este măsurată în ohmi și se poate regla prin utilizarea unor valori diferite de rezistență. Rolul principal al unui rezistor este de a limita curentul electric sau de a reduce tensiunea electrică.

Rezistorii sunt deseanați diferit în funcție de numărul de benzi colorate de pe corpul lor [19]. Benziile colorate reprezintă valorile nominale ale rezistenței și toleranța, iar numărul de benzi poate varia în funcție de precizia rezistorului. În figura. 4.2 este rezentat codul de culori standard utilizat pe rezistori. În cadrul sistemului de față sunt utilizate rezistențe de 10 ohmi.

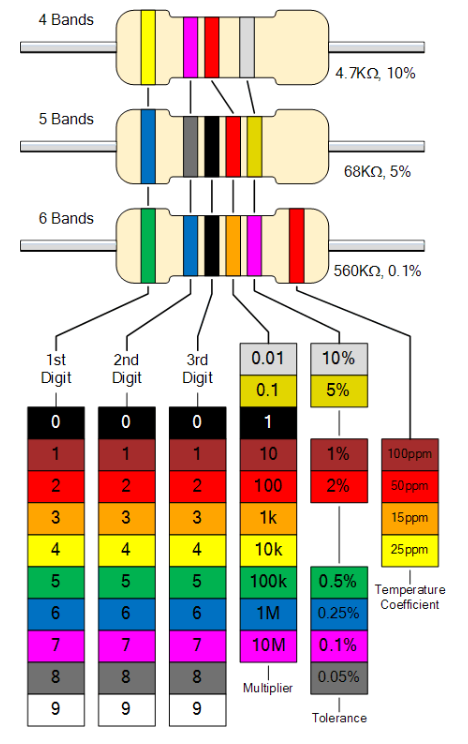


Fig. 4.2 Tabelul codurilor de culori standard ale rezistenței

### Led

LED (Light Emitting Diode) este o componentă electronică pasivă utilizată pentru a emite lumină. Este compus dintr-un material semiconductor care, atunci când este alimentat cu energie electrică, emite lumină vizibilă sau infraroșie, în funcție de materialul semiconductor utilizat. LED-urile sunt disponibile într-o varietate de culori, cum ar fi roșu, verde, albastru, galben, portocaliu și alb, și sunt utilizate în multe aplicații, cum ar fi semafoare, iluminat stradal, ecrane LED, dispozitive electronice și multe altele. Ele sunt preferate față de alte surse de iluminat, cum ar fi becurile incandescente sau fluorescente, datorită consumului redus de energie, duratei de viață mai lungi și a dimensiunilor mai mici.

# Implementarea practică a sistemului de programe

Sistemul este proiectat astfel: caracteristicile extrase din fiecare cadru al probelor audio sunt primii 16 coeficienți Mel cepstrali (MFCC) și coeficienții Delta corespunzători, rezultând un total de 32 de valori pe fiecare cadru. Acești vectori de caracteristici sunt utilizați pentru a antrena una dintre cele trei arhitecturi convoluționale. Codul este scris într-un fișier de notebook interactiv ipynb și sunt folosite mai multe librării ce vor fi explicate în cele ce urmează. După ce modelul este antrenat și evaluat, este convertit în format TensorFlow Lite și utilizat pe un Raspberry Pi 4. În cele din urmă, predicțiile făcute de modelul TFLite sunt utilizate pentru a trimite comenzi simple către un led.

Prima parte a codului este dedicată creării unui set de date adecvat dintr-un director de date brute, în a doua parte sunt definite modelele investigate, scriptul de antrenare și conversia la un model TFLite, iar în ultima secțiune este testat live modelul realizat.

## Setul de date folosit

Setul de date folosit pentru antrenare și validare este Speech Commands Dataset versiunea 0.02, lansat pe 11 aprilie 2018 [4].

Este o colecție open-source formată din 105 836 de mostre audio cu 35 de cuvinte diferite - plus un folder care conține zgomot de fundal -, stocate ca fișiere WAV pe 16 biți. Frecvența de eșantionare este de 16 kHz, în timp ce durata medie a mostrelor este de 1 secundă. Datele provin de la vorbitori diferiți, iar lista cuvintelor poate fi găsită în Tabelul 5.

|  |  |  |  |
| --- | --- | --- | --- |
| **Cuvânt** | **Număr monstre** | **Cuvânt** | **Număr monstre** |
| Backward | 1664 | No | 3941 |
| Bed | 2014 | Off | 3745 |
| Bird | 2064 | On | 3845 |
| Cat | 2031 | One | 3890 |
| Dog | 2128 | Right | 3778 |
| Down | 3917 | Seven | 3998 |
| Eight | 3787 | Sheila | 2022 |
| Five | 4052 | Six | 3860 |
| Follow | 1579 | Stop | 3872 |
| Forward | 1557 | Three | 3727 |
| Four | 3728 | Tree | 1759 |
| Go | 3880 | Two | 3880 |
| Happy | 2054 | Up | 3723 |
| House | 2113 | Visual | 1592 |
| Learn | 1575 | Wow | 2123 |
| Left | 3801 | Yes | 4044 |
| Marvin | 2100 | Zero | 4052 |
| Nine | 3934 | Background Noise | 7 |

Tabel 4 Numărul de monstre pentru fiecare categorie

Pentru sistemul prezentat, s-a folosit doar o fracțiune din colecția menționată mai sus. Motivul este dublu: în primul rând, etichetele pozitive – adică cuvintele de activare sunt doar zece, ceea ce reprezintă aproximativ un sfert din întreaga colecție de date. În al doilea rând, presupunând că toate enunțurile sunt utilizate, categoriile care nu sunt listate în tabelul 5 ar fi clasificate într-o singură categorie „negativă” (adică cuvintele care nu sunt de activare). Acest lucru înseamnă că raportul dintre exemplele pozitive și cele negative ar fi de 33,33% (cu fiecare clasă pozitivă cântărind mai puțin de 4% în setul de date). Prin urmare, dacă modelul antrenat ar furniza doar rezultate negative, ar fi corect în 66,57% din cazuri, ceea ce reprezintă deja un nivel destul de ridicat de acuratețe. Deci proporția datelor nu este una potrivită. Pentru a aborda această problemă și a furniza algoritmilor un set de antrenament mai echilibrat, s-a selectat aleatoriu doar 30% din exemplele din categoriile negative și s-au folosit împreună cu cele pozitive. Acest lucru schimbă numărul total de exemple la 52921, iar procentul fiecărei clase este listat în Tabelul 5. Dintre care 26447 cuvinte pozitive și 26474 cuvinte negative.

|  |  |  |
| --- | --- | --- |
| Etichetă | Număr de monstre | Procent |
| go | 3880 fișiere | 3.67% |
| stop | 3872 fișiere | 3.66% |
| left | 3801 fișiere | 3.59% |
| right | 3778 fișiere | 3.57% |
| visual | 1592 fișiere | 1.50% |
| bed | 2014 fișiere | 1.90% |
| up | 3723 fișiere | 3.52% |
| down | 3917 fișiere | 3.70% |
| follow | 1579 fișiere | 1.49% |
| marvin | 2100 fișiere | 1.98% |
| Cuvinte necunoscute | 26474 fișiere | 25,04% |

Tabel 5 Cuvinte cheie implementate în sistem

## Dependințe principale

Înainte de a continua cu examinarea codului, este importantă prezentarea bibliotecilor și modulelor de care depinde sistemul.

Bibliotecile și modulele utilizate de sistem sunt:

* os: bibliotecă pentru interacțiunea cu sistemul de operare pentru manipularea fișierelor și directoarelor.
* listdir și isdir din os: module pentru manipularea fișierelor și directoarelor.
* tarfile: bibliotecă pentru manipularea fișierelor tar.
* tensorflow și tensorflow.lite: biblioteci pentru construirea, antrenarea și evaluarea modelelor de învățare profundă și conversia acestora în format TFLite.
* keras.models din tensorflow.keras: modul pentru definirea modelului CNN.
* librosa: bibliotecă pentru analiza și prelucrarea semnalelor audio - încărcarea și redimensionarea fișierelor audio.
* python\_speech\_features: bibliotecă pentru extragerea de caracteristici din semnalele audio.
* random: modul pentru generarea de numere aleatoare pentru shuffle.
* numpy: bibliotecă pentru operații cu matrici și vectori.
* matplotlib.pyplot: modul pentru crearea de grafice și vizualizări.
* AudioSegment din pydub: bibliotecă pentru conversia și redimensionarea fișierelor audio.
* Audio din IPython.display: modul pentru redarea fișierelor audio în cadrul notebook-ului.

## Colectarea datelor și extragerea caracteristicilor

După afișarea numelor subdirectoarelor ce alcătuiesc setul de date și a numărului de fișiere incluse în fiecare subdirector, codul continuă cu declarearea unor variabile preliminare.

În special, următoarele necesită explicații suplimentare:

* target\_list – Fiecare subdirector din setul de antrenament este denumit după cuvântul rostit în fișierele audio din acel folder. Așadar, pentru a avea o listă a cuvintelor disponibile, se pot lua numele folderelor;
* val\_ratio, test\_ratio - Aceste valori sunt necesare pentru a împărți numărul de mostre audio alese aleatoriu în set de antrenament, validare și testare;
* len\_mfcc - Această variabilă depinde de alți parametri definiți în funcția cal\_mfcc\_delta. Este folosită pentru a verifica dimensiunea fiecărui vector MFCC și a găsi eventuale eșantioane corupte sau scurte. În cazul meu len\_mfcc are valoarea 98;
* neg\_labels\_percent - Pentru a evita un dezechilibru semnificativ al setului de antrenament, se păstrează un procent din numărul de mostre ce au etichete negative. Definim etichete negative, etichetele ce nu au fost alese în lista de cuvinte de activare – „wake\_words”.

În Fig. 6.1, este prezentat codul utilizat pentru a crea lista de nume de fișiere cu calea lor completă și vectorii de etichete, din setul de date.

Pentru fiecare subdirector din target\_list:

* Se construiește calea către subdirector utilizând os.path.join().
* Se creează o listă de nume de fișiere din subdirector folosind os.listdir().
* Numele fișierelor și vectorul de etichete pentru fiecare subdirector sunt adăugate la lista corespunzătoare, filenames și y.
* Dacă subdirectorul nu se află în lista „wake\_words”, se va păstra doar un procent din numărul total de fișiere și vectorul de etichete corespunzător, pentru a avea o distribuție echilibrată a datelor de antrenare.
* Pentru subdirectoarele care se află în lista de cuvinte de activare, se va adăuga doar întregul număr de fișiere și vectorul de etichete corespunzător.
* La final, listele – „filenames” și „y” sunt aplatizate la liste simple.

****

Fig. 5.1 Crearea listelor cu monstre și etichetele acestora

La acest punct, listele filename și y sunt încă independente una de alta. Prin urmare, înainte de a fi amestecate prin metoda shuffle, listele sunt unite în filenames\_y. Apoi, lista rezultată este împărțită din nou și din aceste liste sunt create seturile de antrenare, validare și testare

Definiția funcției pentru calcularea coeficienților MFCC și Delta dintr-un fișier audio este prezentată în Fig. 6.2. După cum se poate vedea din cod, această metodă primește ca argument calea către fișierul WAV, care este încărcat ca o serie temporală de numere în virgulă mobilă prin intermediul funcției librosa.load.

În primul rând, sunt calculați coeficienții MFCC și Delta folosind funcțiile python\_speech\_features.base.mfcc și python\_speech\_features.base.delta. O explicație detaliată a tuturor parametrilor disponibili pentru aceste funcții poate fi găsită în documentația oficială [20].

Cei mai relevanți parametri ai python\_speech\_features.base.mfcc sunt:

* winlen - Lungimea ferestrei de analiză, în secunde. Se utilizează o valoare de 0,03 s;
* winstep - Pasul dintre ferestrele succesive, în secunde. Se utilizează o valoare de 0,01 s;
* nfft - Dimensiunea FFT. Valoarea introdusă este de 512, ceea ce înseamnă că rezoluția frecvenței pentru fiecare linie spectrală este de 15,625 Hz2. Unde rezoluția\_frecvenței = frecvența\_eșantionare/dimensiune\_fft
* appendEnergy - Dacă este True, metoda înlocuiește coeficientul cepstral zero cu logaritmul energiei totale a cadrelor. Motivul este acela că cel mai mic coeficient poartă puține informații specifice vorbitorului și, prin urmare, este inutil pentru scopul acestui program;
* winfunc - Fereastra de analiză aplicată fiecărui cadru. S-a ales să se aplice o funcție Hann, deoarece produce mai puțină scurgere spectrală și, prin atingerea valorii zero la ambele capete ale ferestrei (Fig. 5.2), elimină toate discontinuitățile semnalului.

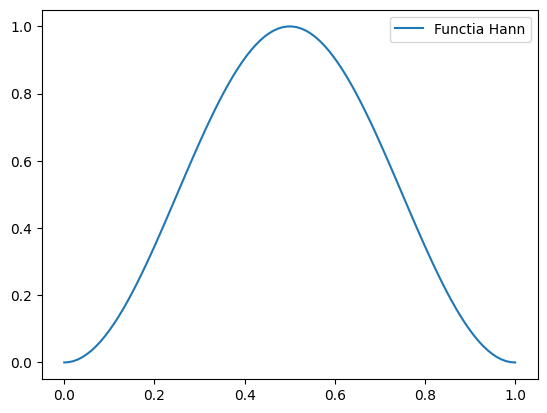


Fig. 5.2 Funcția Hann

În plus, metoda python\_speech\_features.base.delta are doar doi parametri: MFCC-urile extrase anterior și numărul de cadre precedente și următoare pe care trebuie calculate Delta. În cele din urmă, coeficienții MFCC și Delta ai eșantionului selectat sunt combinați într-un array NumPy, care este transpus și returnat de funcție.

Forma rezultată va avea dimensiunile - 32x98 (32 în acest caz, 16 MFCC-uri și 16 Delta, iar 98 reprezită numărul de cadre pentru fiecare eșantion audio).

****

Fig. 5.3 Metoda calc\_mfcc\_delta

În continuare, efectuez extragerea caracteristicilor și eliminarea monstrelor corupte. Dimensiunea fiecărui vector NumPy generat de funcția menționată mai sus este comparată cu variabila len\_mfcc, pentru a găsi fișiere audio corupte sau scurte în setul de antrenament. Dimensiunea corectă a matricei rezultate în urma calculării coeficienților MFCC și Delta depinte de lungimea și pasul ferestrei de analiză plus lungimea semnalului. Aceasta poate fi calculată prin rotunjirea prin adaos a valorii x obținute din ecuația *6*.*1*, unde lsignal este lungimea semnalului audio exprimată în secunde.

|  |  |
| --- | --- |
|  | 6.1 |

Metoda extract\_features, prezentată în Fig. 5.4, primește ca argumente două liste: una conținând numele fișierelor și cealaltă conținând etichetele fiecărui eșantion din setul de date brut. O buclă for parcurge listele și pentru fiecare enunț se execută următoarele acțiuni:

1. Definirea căii de fișier specifică prin unirea căii subfolderului cu numele fișierului.
2. Verificarea că fișierul este în format WAV. Dacă numele fișierului nu se termină cu .wav, bucla sare peste iterarea curentă.
3. Extracția coeficienților MFCC și Delta prin funcția prezentată în Fig. 5.3.
4. Verificarea că vectorul de caracteristici nou creat respectă cerința de dimensiune. Dacă cerința este satisfăcută, atât coeficienții, cât și eticheta sunt incluse în vectorii de ieșire corespunzători. Dacă nu, eșantionul este eliminat, se afișează o avertizare în consola de ieșire, iar prob\_cnt este incrementat cu 1.

În cele din urmă, metoda returnează un vector de caracteristici, un vector de etichete și numărul de fișiere audio eliminate în proces.

****

Fig. 5.4 Definirea metodei extract\_features

Următorul pas este necesar pentru a organiza mai bine categoriile în care sunt împărțite caracteristicile. Pentru a rezolva această problemă, celula de cod din Fig. 5.5 utilizează listele wake\_word\_index\_vec și not\_wake\_word\_index\_vec definite în Fig. 5.1 pentru a înlocui fiecare clasă pozitivă cu un număr întreg între 0 și „len(wake\_words) – 1”, iar toate clasele negative cu numărul „len(wake\_words)”. Vectorii de etichete care conțin clase ordonate facilitează procesul ulterior de optimizare și îmbunătățesc performanța CNN. Ultima celulă care trebuie executată în această secțiune salvează matricile într-un fișier .npz astfel încât seturile de antrenament, validare și testare să poată fi încărcate din nou în notebook fără a fi nevoie să se ruleze din nou celulele anterioare (cu excepția celor care conțin dependințe).

****

Fig. 5.5 Înlocuirea etichetelor pentru organizarea categoriilor

## Adaptarea modelelor

Această secțiune se ocupă de implementarea, antrenarea și evaluarea modelelor prezentate în 3.7. Înainte de definirea modelelor, sunt prezente mai multe operații suplimentare.

În primul rând, se încarcă și pregătește fișierul .npz, pentru a nu repeta procesul de extragere a caracteristicilor de fiecare dată.

În al doilea rând, se efectuează o prelucrare finală a seturilor de caracteristici. În special, clasa Conv2D importată din Keras (când este folosită ca prim strat într-o arhitectură de rețea) necesită argumentul cheie input\_shape. Acest argument trebuie formatat ca input\_shape = (înălțime, lățime, număr\_canale). Astfel matricilor de caracteristici li se vor adăuga o dimensiune în plus ce va reprezenta numărul de canale (Fig. 6.6).

****

Fig. 5.6 Redimensionarea tensorilor din 3D în 4D

Ultimul pas de optimizare, înainte de construirea rețelei convoluționale ce va fi antrenată și evaluată, este de a converti etichetele din clase numerice în vectori binari(one hot encoded). Așa cum am menționat anterior, vectorii de etichete y\_train, y\_val și y\_test conțin valori de la 0 la len(wake\_words).

Acest tip de etichete reduce drastic performanța arhitecturilor de învățare profundă atunci când se efectuează clasificarea multi-clasă. Motivul este că etichetele întregi stochează informații suplimentare care sunt absolut arbitrare (ca urmare a ordinii subdirectoriilor inițiale). Un exemplu pentru a clarifica conceptul este următorul: dacă procesul de extragere a caracteristicilor a convertit cuvintele "on", "off" și "bed" în clase numerice 1, 2 și 3, în timpul antrenării, algoritmul ar învăța să emită clasa 3 atunci când se spune cuvântul "bed". Dar, în plus față de asta, rețeaua ar cunoaște și faptul că clasa "bed" este mai mare decât "off", deoarece 3 este mai mare decât 2, și acest lucru este evident o informație incorectă și suplimentară.

Această problemă este rezolvată prin acordarea aceleiași "greutăți matematice" fiecărei clase, adică prin crearea unui vector binar care conține atâtea elemente câte clase este antrenat să le recunoască. Acest vector binar are valoarea „1” plasată într-o poziție diferită în funcție de categoria specifică reprezentată, în timp ce toate celelalte poziții conțin valorea „0”. Pentru efectuarea acestei operații folosesc funcția „to\_categorical” din biblioteca „utils” (Fig. 6.7).

****

Fig. 5.7 Conversia etichetelor din numere întregi în vectori binari

## VGG16

Modificările aduse modelului VGG16 au fost inspirate de Geifman [21], care a proiectat o variantă a acestui model pentru setul de date CIFAR-10, care este format din 60000 de imagini RGB de dimensiunea 32x32 împărțite în 10 clase diferite. Pe baza lucrării lui Geifman, modificările aplicate designului original propus de Simonyan și Zisserman [12] sunt următoarele:

* Forma de intrare a fost modificată de la 224 × 224 × 3 la 32 × 98 × 1;
* Dropout a fost adăugat după straturile convoluționale: 1, 3, 5, 6, 8, 9, 11, 12 și 13 (procentele de dropare pentru fiecare strat sunt listate în Tabelul 4);
* Primul strat complet conectat și următorul de dropout au fost eliminate pentru a simplifica modelul și a evita supraadaptarea(overfitting);
* Numărul de unități din al doilea strat dens a fost modificat de la 4096 la 512;
* Normalizarea lotului (batch normalization) a fost adăugată după fiecare strat, cu excepția stratului de ieșire;
* Lungimea stratului de ieșire a fost redusă de la 1000 la „len(wake\_words)+1”.

|  |  |
| --- | --- |
| Număr strat convoluțional | Procentul de dropare |
| 1 | 0,3 |
| 3 | 0,4 |
| 5 | 0,4 |
| 6 | 0,4 |
| 8 | 0,4 |
| 9 | 0,4 |
| 11 | 0,4 |
| 12 | 0,4 |
| 13 | 0,5 |

Tabel 6 Procentele de abandon ale straturilor Dropout în VGG16

## AlexNet

Pentru a adapta rețeaua originală creată de Krizhevsky și colaboratorii săi [14], au fost aplicate următoarele modificări:

* Forma de intrare a fost modificată din 224 × 224 × 3 în 32 × 98 × 1 pentru a se potrivi cu vectorii de caracteristici extrași cu metoda calc\_mffc\_delta(Fig. 6.3) ;
* Pentru a preveni o reducere excesivă a datelor, dimensiunea filtrelor cu kernel în primul strat de convoluție a fost redusă de la 11 × 11 la 7 × 7, iar pasul de la 4 × 4 la 2 × 2;
* Pentru același motiv ca și la punctul anterior, dimensiunea câmpului receptiv al tuturor straturilor max-pooling a fost redusă de la 3 × 3 la 3 × 2, în timp ce pasul rămâne neschimbat;
* Al patrulea strat de convoluție a fost eliminat pentru a preveni overfitting-ul pe setul de antrenare care este considerabil mai mic decât setul de date ImageNet;
* Pentru a reduce numărul de parametri al modelului, precum și pentru a preveni overfitting-ul, primul strat complet conectat și dropout-ul ulterior au fost eliminate;
* În loc de normalizarea răspunsului local (Local response normalization) a fost folosită normalizarea loturilor (Batch normalization);
* Lungimea stratului de ieșire a fost redusă de la 1000 la „len(wake\_words)+1”, pentru a se potrivi cu categoriile clasificate de sistem.

## SqueezeNet

Pentru modelul SqueezeNet, arhitectura propusă de Iandola [16] are deja un număr limitat de parametri și o amprentă mică de memorie. Cu toate acestea, la fel ca celelalte modele, fiind concepută pentru a învăța din imaginile stocate în ImageNet implică faptul că rețeaua așteaptă date mai mari și mai numeroase.

O adaptare a SqueezeNet creată pentru CIFAR-10 de către Hancock [22] a fost utilizată fără mari modificări, cu execepția dimensiunii formei de intrare și a lungimii stratului de ieșire. În plus a fost crescut procentul de dropare din 0.5 în 0,6 și a fost adăugată regularizarea L2 în stratul convoluțional pentru a reduce overfitting-ul.

## Selecția modelului și antrenarea propriu-zisă

Pentru a facilita selectarea, compilarea și antrenarea rețelelor convoluționale,

reprezentative sunt celule de cod prezentate în figura 7.1 și figura 7.2. Funția de cost (eng. loss function) utilizată pentru calcularea erorii în scorurile rețelei este entropia încrucișată categorică (eng. categorical cross entropy), definită în ecuația 7.1, unde y\_truei este eticheta adevărată și y\_predi este eticheta prezisă.

|  |  |
| --- | --- |
|  | (7.1) |

Cele trei arhitecturi folosesc funția de optimizare SGD cu moment Nesterov pentru a estima valoarea gradientului în avans. Alți parametri ai acestui optimizator sunt:

* Rata de învățare ce controlează cât de mult greutățile modelelor sunt actualizate la fiecare pas. Cu o rată de învățare mai mică, modelele sunt actualizate mai lent, dar pot ajunge la un minim local mai precis și, cu o rată de învățare mai mare, modelele sunt actualizate mai rapid, dar pot sări peste minimul local optim.
* Rata de descompunere a ratei de învățare(englezescul decay) - în timpul antrenării, poate fi util să se reducă treptat rata de învățare pentru a ajunge la un minimum local mai precis.
* Momentum - parametru ce ajută la accelerarea vitezei de convergență a funcției de cost prin adăugarea unei fracțiuni din actualizările anterioare la noua actualizare a ponderilor.

****

Fig. 7.1 Alegerea modelului, setarea metodei de optimizare și a funcției de cost

Mai departe, după antrenarea modelului, rezultatele rețelelor neuronale convoluționale sunt evaluate folosind precizia(acuratețea) antrenamentului, a validării și a testului. Precizia antrenamentului și a validării sunt calculate ca medie a ultimelor 250 de epoci ale procesului corespunzător. Modelul este antrenat folosind metoda fit() pe setul de date de antrenament și cu un lot de dimensiune 128. Validarea se realizează folosind setul de validare corespunzător. Parametrul steps\_per\_epoch specifică numărul de pași de antrenament pe epocă și este calculat ca x\_train.shape[0] // 128.

****

Fig. 7.2 Antrenarea modelului și validarea sa

## Conversia la Tensorflow Lite

După ce un modelul este construit, antrenat și evaluat, procesul de conversie din TensorFlow în TensorFlow Lite este simplu, așa cum este prezentat în Fig. 5.8. Procesul de conversie constă în încărcarea modelului TensorFlow care a fost salvat anterior , crearea unui obiect TFLiteConverter (în acest caz, dintr-un model Keras) și convertirea modelului în format TFLite. Următorul pas este de a copia modelul în folderul programului de pe Raspberry Pi 4 și de a configura TFLite Interpreter.

****

Fig. 5.8 Conversia modelului din Tensorflow în TensorflowLite

## Utilizarea modelului pe Raspberry Pi

Fișierul rpi\_word\_recognition.py conține partea principală a codului pentru a efectua testarea modelului simplificat pe dispozitiv. Începe cu importul dependențelor necesare și apoi este definit un set de parametri (Fig. 5.9):

* debug\_time, debug\_acc - doi întregi folosiți ca variabile boolene, declarați în scopuri de depanare;
* word\_threshold - probabilitatea pragului care trebuie depășită de predicția CNN pentru a declanșa o comandă;
* sample\_length, rec\_duration, window\_stride - prima valoare este durata (în secunde) a probelor audio acceptate de model, a doua este lungimea blocurilor ulterioare înregistrate și procesate de program, în timp ce a treia este pasul ferestrei. În configurația sa implicită, sistemul înregistrează blocuri de 0,5 secunde cu un pas de 0,5 secunde;
* sample\_rate, resample\_rate - rata de eșantionare a înregistrării și rata de eșantionare necesară modelului. Este necesar un proces de reeșantionare deoarece microfonul funcționează mai bine când se înregistrează cu 48.000 de eșantioane pe secundă;
* num\_channels - numărul de canale înregistrate. Sistemul explorat funcționează cu probe audio mono, deci această variabilă este setată la "1";
* num\_mfcc - această valoare trebuie să corespundă parametrului setat în timpul antrenării;
* model\_path - variabilă de tip șir de caractere care conține numele modelului TFLite care trebuie încărcat și utilizat pentru a efectua deducția.

****

Fig. 5.9 Parametri preliminari

Trecând la Figura 5.10, buffer-ul utilizat pentru a stoca jumătate din coeficienții necesari este definit ca o matrice NumPy de zerouri, un obiect TFLiteInterpreter este creat și modelul antrenat este încărcat în software. Metodele apelate după declarația interpretorului sunt necesare pentru a configura resursele pentru detecția pe dispozitiv. Când este convertit în formatul TFLite, un model pre-antrenat este transformat într-un buffer binar care conține informații despre arhitectură și greutăți, dar nu suportă cele mai multe dintre operațiile TensorFlow care automatizează sarcini precum detecția. De exemplu, alocarea de memorie pentru tensorii de intrare și ieșire trebuie declanșată prin apelarea metodei allocate\_tensors, în timp ce get\_input\_details și get\_output\_details sunt utilizate pentru a stoca informații despre straturile de intrare și ieșire ale modelului.

****

Fig. 5.10 Confiugurare buffer și model interpretor

Software-ul care rulează pe Raspberry Pi este format din două funcții: decimate, sd\_callback. Prima metodă este prezentată în Fig. 5.11 și este folosită pentru a reduce frecvența de eșantionare a fluxului audio provenit de la microfon printr-un factor întreg. Mai precis, metoda SciPy apelată în corpul funcției atenuează frecvențele nedorite rezultate în urma conversiei din analog în digital, după aplicarea unui filtru anti-aliasing, iar acest proces este cunoscut formal sub numele de decimare. După procesare, funcția returnează semnalul redus la noua sa dimensiune și rată de eșantionare.

****

Fig. 5.11 Funția decimate

Metoda a doua se numește msgSwitcher și este o implementare în Python a unei declarații switch-case: primește ca argument o valoare întreagă și returnează șirul de caractere corespunzător. Dacă valoarea de intrare nu se află în intervalul 0-5, șirul returnat este "Unknown".

În cele din urmă, funcția sd\_callback (prezentată în Fig. 5.12 ) gestionează procesarea semnalului, extragerea caracteristicilor și rulează detecția. Este apelată ciclic de modulul sounddevice, care introduce un flux audio împărțit în blocuri de dimensiune – sample\_rate ∗ rec\_duration. Pentru fiecare bloc audio, funcția ia un timestamp - necesar pentru depanare, comprimă înregistrarea într-un vector 1D, efectuează decimarea și extrage jumătate din matricea de coeficienți (așa cum s-a menționat anterior, un bloc audio durează doar jumătate de secundă).

Este important de menționat că acest proces este rezultatul unor teste preliminare pe sistem. Într-o versiune anterioară a codului, s-a inițializat un buffer pentru a stoca o secundă de semnal audio (este încă vizibil ca și comentariu în Fig. 5.10) – ceea ce era mai lent. Soluția a fost să mut "operația de glisare" a ferestrei după extragerea caracteristicilor. Concret, un bloc audio A care sosește este procesat și coeficienții săi sunt stocați în mfccs\_new. Apoi, la sosirea blocului audio ulterior B, matricea de coeficienți a lui A este asignată la mfccs\_old, eliberând variabila mfccs\_new. În acest fel, doar 0,5 s de flux audio sunt transformate pentru fiecare ciclu, reducând timpul de procesare la jumătate.

****

Fig. 5.12 Prima parte a funcției sd\_callback

După ce au fost generați coeficienții MFCC și Delta din eșantionul audio, metoda sd\_callback continuă prin crearea unui tensor de intrare - adică o versiune remodelată a matricei de coeficienți, așa cum se procedează și în Fig. 5.6 - și setează valoarea acestuia în funcție de detaliile de intrare ale modelului. În TFLite, detecția este rulată prin invocarea obiectului TFLiteInterpreter, iar vectorul de predicție este stocat ca parte a detaliilor de ieșire.

În cele din urmă, valoarea maximă de predicție (exprimată ca un număr între 0 și 1) și poziția sa în vectorul de ieșire (corespunzătoare clasei recunoscute) pot fi extrase și utilizate pentru a decide ce comandă să fie trimisă către pinii de GPIO. Pentru a efectua această sarcină, o instrucțiune „if” verifică dacă valoarea maximă prezisă este mai mare decât pragul ales, ca măsură a cât de sigur trebuie să fie modelul în predicția sa pentru a declanșa o comandă. Dacă procentul este mai mare decât pragul, comanda corespunzătoare clasei prezise este selectată și trimisă. Mai mult, dacă indicatorii de depanare definiți în Fig. 5.9 sunt activați, funcția afișează anumiți parametri, cum ar fi timpul de execuție, acuratețea și întregul vector de ieșire.

Fișierul rpi\_word\_recognition.py se încheie cu bucla principală care, prin metoda InputStream, înregistrează blocurile audio și invocă sd\_callback.

****

Fig. 5.13 A doua parte a funției sd\_callback

# Mediul de lucru

## Crearea environmentului

Crearea unui environment înseamnă crearea unui mediu izolat pentru dezvoltarea și rularea aplicațiilor Python. Un mediu de lucru îți permite să instalezi și să gestionezi separat versiuni de pachete Python și să eviți conflictul între pachetele instalate în diferite proiecte.

Anaconda este o platformă de distribuție și gestionare a pachetelor pentru limbajele de programare Python și R. Acesta include o serie de pachete pre-încărcate și o interfață grafică utilizator (Anaconda Navigator) care permite utilizatorilor să instaleze, să actualizeze și să gestioneze pachete și dependințe fără a utiliza linia de comandă.

Pentru acest proiect, mediul izolat necesar a fost realizat utilizând exclusiv linia de comandă și salvat sub forma unui fișier YAML. O listă a comenzilor folosite pentru a instala pachetele necesare este expusă în tabelul de mai jos:

|  |
| --- |
| conda create --name gputf python=3.10.6 |
| conda install jupyter |
| conda install -c conda-forge tensorflow-gpu |
| conda install -c conda-forge tensorflow |
| conda install -c conda-forge librosa |
| conda install -c conda-forge keras |
| conda install -c conda-forge pydub |
| conda install -c conda-forge pyaudio |
| conda install -c conda-forge playsound |
| conda install -c conda-forge graphviz |
| conda install -c conda-forge pydot |
| conda install -c bricew python\_speech\_features |
| pip install sounddevice |
| conda env export > environment.yml |
| conda env create --name myenv -f environment.yml |

Tabel 7 Pregătirea mediului de lucru și a pachetelor necesare

În urma problemelor întâmpinate cu instalarea diferitelor pachete, au fost găsiți o serie de pași ce au dus la rezolvarea acestora:

* Verificarea că pachetul este compatibil cu versiunea de Python instalată.
* De multe ori, problemele de instalare pot fi rezolvate prin curățarea cache-ului conda folosind comanda "conda clean -tipy", care elimină toate pachetele temporare și inutile din cache-ul local. Apoi se poate încerca reinstalarea pachetului.
* Crearea unui environment nou cu o altă versiune de Python.

## Instalarea sistemului de operare pe Raspberry Pi

Pentru a putea folosi placa de dezvoltare – Raspberry Pi 4, este necesară instalarea unui sistem de operare. În cazul proiectului meu, am ales să instalez sistemul de operare Raspberry Pi OS deoarece este un sistem de operare open source, eficient și ușor de folosit. Sistemul de operare Raspberry Pi OS vine pre-instalat cu un set de aplicații esențiale, inclusiv un browser web, un editor de text, un player multimedia și multe alte instrumente utile pentru programatori și utilizatori obișnuiți. De asemenea, acesta include suport pentru numeroase limbaje de programare, inclusiv Python, C++, Java și Ruby.

Cum placa de bază nu are memorie internă, sistemul de operare va fi scris pe un card MicroSD folosind aplicația Raspberry Pi Imager. MicroSD-ul va fi folosit și pentru bootare și ca memorie internă.

Vom alege din interfața oferită de acest utilitar: sistemul de operare, dispozitivul de stocare și câteva opțiuni suplimentare de configurare în ceea ce privește conectarea la internet și credențialele de conectare prin Putty.

În figura. 6.1 sunt evidențiați parametrii ce au fost setați preferențial pentru conectarea prin Putty și conectarea la o rețea Wi-Fi cunoscută. De asemenea este aleasă ora standard și limba tasturii.

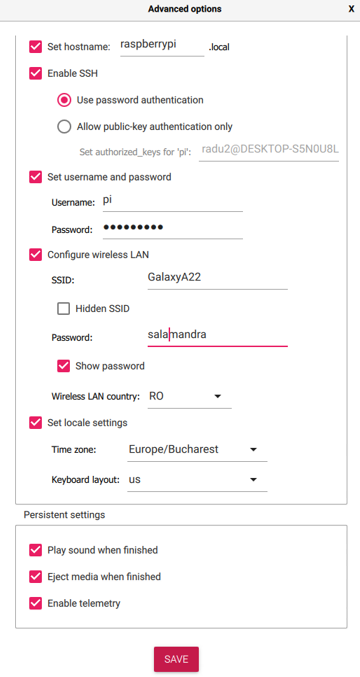


Fig. 6.1 Raspberry Pi Imager - Advanced Options

## Conectarea la Raspberry Pi

Conectarea Raspberry Pi 4 la un calculator poate fi realizată prin mai multe modalități, iar una dintre cele mai utilizate este conectarea prin intermediul unui cablu Ethernet (RJ45). După ce ambele dispozitive au fost pornite și interconectate, se deschide Putty, se introduce HostName-ul – raspberrypi.local, se selectează protocolul SSH și se apasă butonul Open.

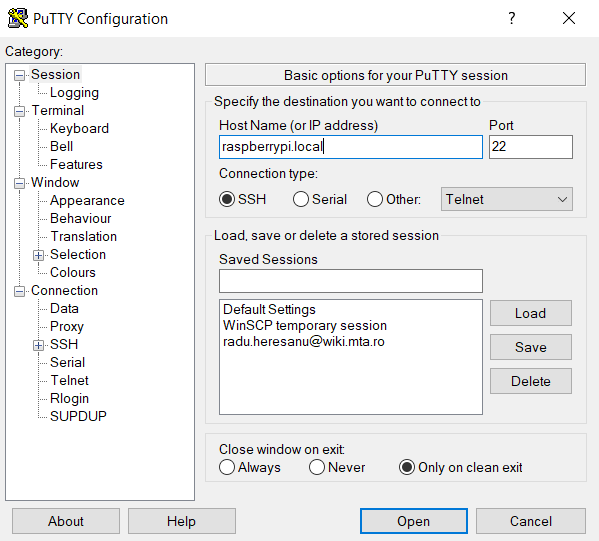


Fig. 6.2 Putty - conectarea la Raspberry Pi 4

## Interconectarea componentelor hardware ale sistemului

Interconectarea componentelor sistemului presupune în primul rând alimetarea de la o sursă de energie. Placa integrată – Raspberry pi 4 poate fi racordată la o sursă de alimentare cu o tensiune nominală de 5.1V și un curent maxim de 3A, folosind o mufă tip C. În acest sens, pot fi folosite trei surse de energie: un alimentator omologat, o baterie externă, laptop-ul personal (folosind un cablu USB-C).

Microfonul este conectat cu placa integrată folosind unul dintre cele patru porturi USB, de preferat unul din cele două porturi USB 3.0 datorită vitezei de transfer e și a puterii maxime ce o poate furniza superioare față de porturile USB 2.0. Fizic, componentele de până acum pot fi vizualizate în Anexa A.

Comenzile generate în urma procesării semnalului audio sunt transmise folosind pinii GPIO ai plăcuței și fire Dupont, sub forma de impulsuri electrice(3,3 V, 16mA), către breadbord.

Mai departe pe breadbord poate fi montat un led sau alte dispozitive hardware care să fie acționate.

# Rezultate experimentale și discuții

În general, evaluarea a evidențiat mai multe probleme comune tuturor configurațiilor. Sistemul nu răspunde întotdeauna corect în situațiile în care este prezent doar zgomotul de fond și are puțină încredere în clasificarea ferestrei ca aparținând categoriei not\_wake\_word. Acest lucru se datorează faptului că nu există suficiente mostre audio care să conțină zgomot de fundal.

Nici performanța de recunoaștere a cuvintelor nu este ideală pe Raspberry Pi 4. Cred că tonul vocii utilizatorului și felul în care pronunță cuvintele joacă un rol important. Folosind un microfon și un dispozitiv de procesare mai performante, rezultatele ar putea fi direct proporționale.

Totuși, așa cum reiese și din tabelul 8, metricile obținute sunt mulțumitoare. Compararea metricilor rezultate în urma antrenării, validării și testării fiind și primul experiment realizat. Se poate observa că arhitectura AlexNet prezintă cea mai bună acuratețe pe setul de date de antrenare (99,5%), iar diferența dintre predicțiile modelului și valorile adevărate (funcția de cost – eng. Loss) este de asemenea cea mai mică pentru acest model. VGG16 concurează cu arhitectura Alexnet în ceea ce privește acuratețea de validare și testare, valorile fiind destul de apropiate. De asemenea, SqueezeNet generează cele mai robuste modele în timpul cel mai scurt. Din tabelele 9 și 10 se poate observa că modelul SqueezeNet folosit pe placa Raspberry Pi 4 are dimensiunea de numai 485 kB, modelul fiind realizat într-un timp de 5 ori mai mic decât modelul generat cu VGG16, iar resursele computaționale folosite pe target sunt de asemenea cele mai bune - Tabel 11.

Anterior era folosită funcția de optimizare Adam pentru Alexnet, dar rezultatele nu erau la fel de bune așa cum sunt cu SGD.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Antrenare** | | **Validare** | | **Testare** | |
| **Arhitectură** | Loss | Acc. | Loss | Acc. | Loss | Acc. |
| **VGG16** | 0.4999 | 0.9745 | 0.6525 | **0.9425** | 0.3920 | **0.9649** |
| **AlexNet** | 0.0154 | **0.9950** | 0.3237 | **0.9474** | 0.2186 | **0.9641** |
| **SqueezeNet** | 0.1017 | 0.9694 | 0.5534 | 0.8941 | 0.5113 | 0.9218 |

Tabel 8 Performanța CNN-urilor pe setul cu 6 cuvinte de activare

|  |  |  |  |
| --- | --- | --- | --- |
| **Arhitectură** | **Timp Antrenare** | **Timpul unei epoci** | **Timpul unui pas** |
| **VGG16** | 5780s | 23s 123ms | 0.109 s/step |
| **AlexNet** | 2010s | 8s 33ms | 0.0286 s/step |
| **SqueezeNet** | 1005s | 4s 23ms | 0.0191 s/step |

Tabel 9 Compararea timpilor pentru modelul cu 6 cuvinte de activare

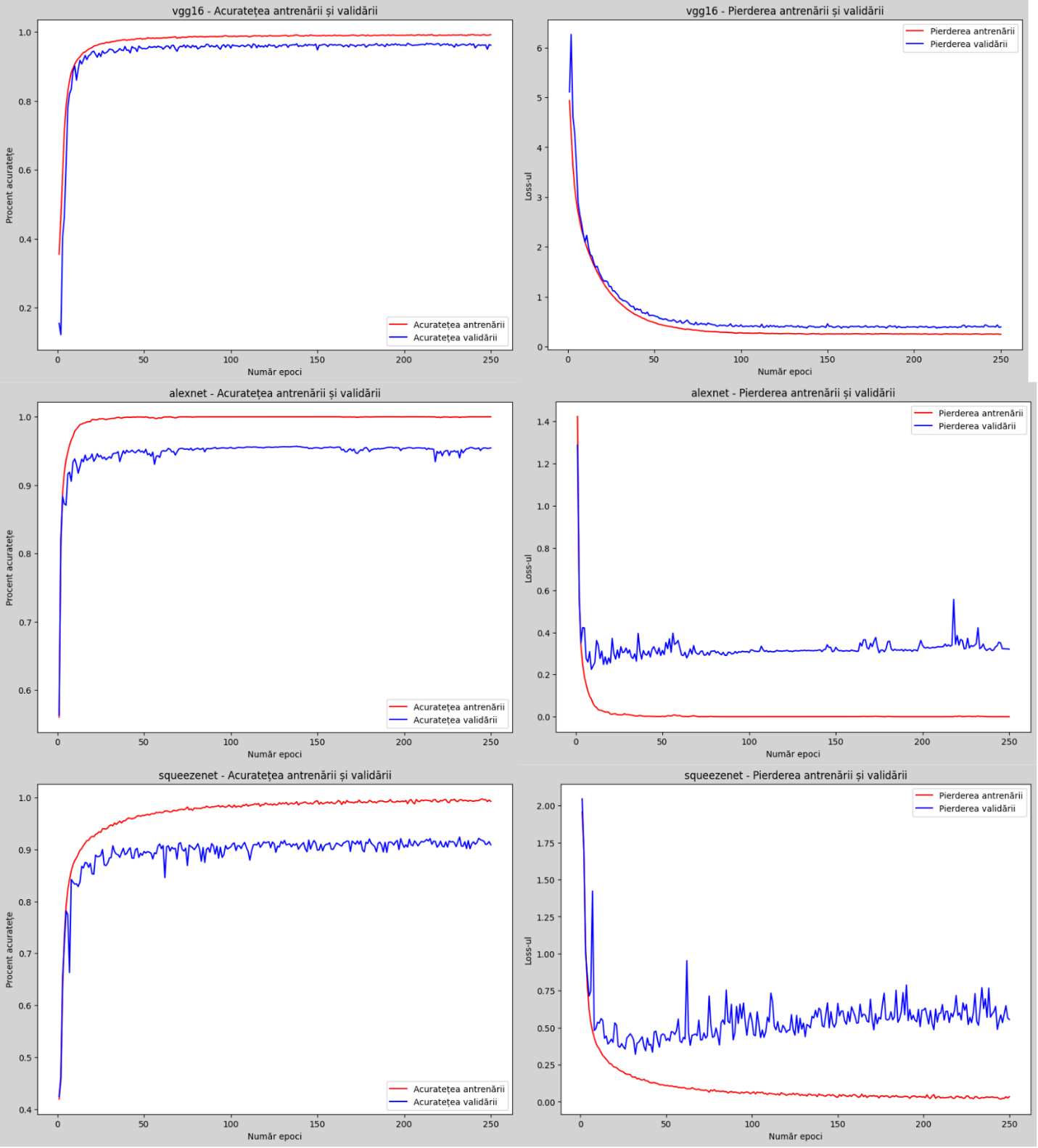
|  |  |  |
| --- | --- | --- |
| **Arhitectură** | **Dimensiune model TF** | **Dimensiune model TFLite** |
| **VGG16** | 117 MB | 58 MB |
| **AlexNet** | 65 MB | 21 MB |
| **SqueezeNet** | 2 MB | **485 kB** |

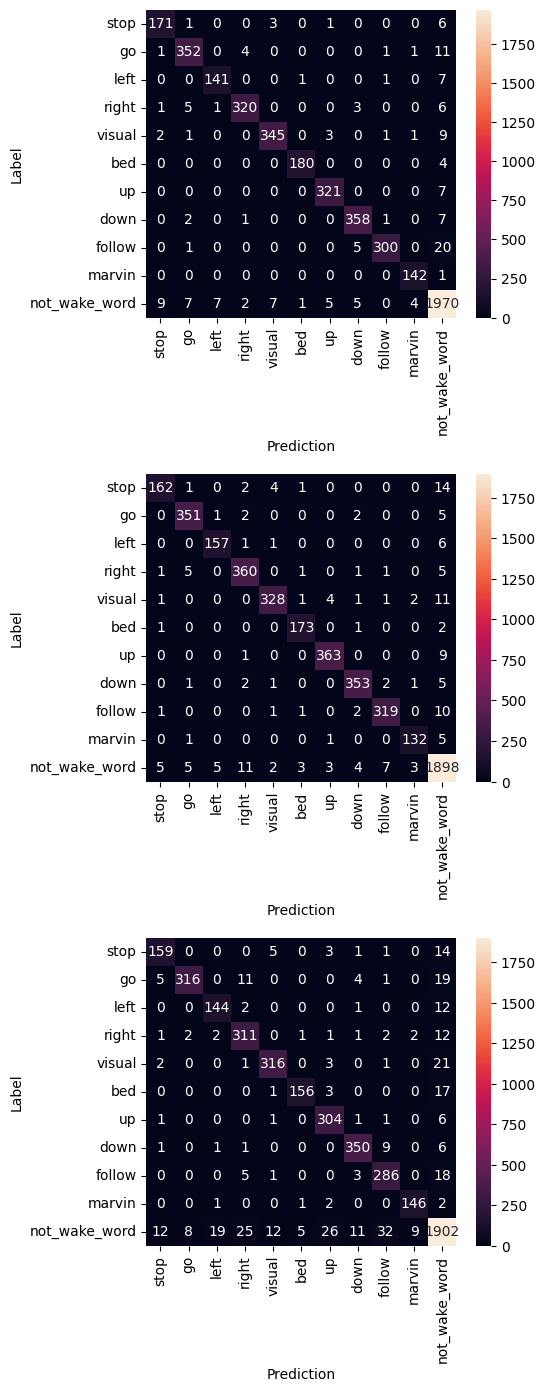
Tabel 10 Comparația dimensiunilor modelelor cu 6 cuvinte de activare

|  |  |  |
| --- | --- | --- |
| **Arhitectură** | **CPU (%) Raspberry Pi – quad core** | **RAM (%)** |
| **VGG16** | 125 | 5.6 |
| **AlexNet** | 106 | 3.5 |
| **SqueezeNet** | 101 | 2.4 |

Tabel 11 Performanța pe Raspberry Pi a modelelor cu 6 cuvinte de activare

Un lucru important de luat în considerare atunci când se proiectează un sistem care procesează date vizuale sau audio este posibilitatea implicațiilor etice și legale legate de confidențialitatea utilizatorului. Spre deosebire de asistenții virtuali inteligenți precum Siri, Alexa, Google Assistant, sistemul propus nu trimite nicio informație la internet, iar toate procesările se fac pe dispozitivul utilizatorului. Mai mult, după cum se poate observa din codul sursă, software-ul în discuție înregistrează doar ferestre de jumătate de secundă care sunt continuu suprascrise. Prin urmare, fiecare înregistrare audio are o durata de jumătate de secundă și o durată de viață totală de o secundă, ceea ce face imposibil ca utilizatorii să fie victime ale furtului de informații sensibile.





# Concluzii și direcții viitoare de cercetare

Rezultatele acestei lucrări arată că rețelele de învățare profundă pot fi dezvoltate cu succes pentru o gamă largă de sarcini, inclusiv aplicații cu consum redus și dispozitive portabile. Este important să găsim compromisul potrivit între performanța rețelei și costul de calcul și să ne asigurăm că setul de date utilizat pentru antrenare și validare este reprezentativ, diversificat și de dimensiune adecvată.

O direcție viitoare de dezvoltare poate presupune îmbunătățirea uneia dintre cele trei arhitecturi și realizarea unui set de date de cuvinte în limba română ce va fi folosit pentru a antrena modele ce vor fi folosite în automatizarea aprinderii unui bec inteligent și modificarea intensității și culorii acestuia.

# Bibliografie

|  |  |
| --- | --- |
| [1] | „Wikipedia,” 2023 Martie 2023. [Interactiv]. Available: https://en.wikipedia.org/wiki/PuTTY. [Accesat 01 Aprilie 2023]. |
| [2] | „WinSCP,” 23 Noiembrie 2019. [Interactiv]. Available: https://winscp.net/eng/docs/file\_encryption. [Accesat 01 Aprilie 2023]. |
| [3] | L. Moroney, „Intro to Machine Learning (ML Zero to Hero - Part 1),” 30 August 2019. [Interactiv]. Available: https://www.youtube.com/watch?v=KNAWp2S3w94. [Accesat 25 Martie 2023]. |
| [4] | P. Warden, „Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition,” 2018. [Interactiv]. Available: https://arxiv.org/pdf/1804.03209.pdf. [Accesat 2023]. |
| [5] | P. Antoniadis, „Baeldung,” 23 Martie 2023. [Interactiv]. Available: https://www.baeldung.com/cs/sigmoid-vs-tanh-functions. |
| [6] | N. Kumar, „The Professionals Point,” 06 Iunie 2019. [Interactiv]. Available: http://theprofessionalspoint.blogspot.com/2019/06/dying-relu-causes-and-solutions-leaky.html. [Accesat 30 Aprilie 2023]. |
| [7] | „lwvworc.org,” [Interactiv]. Available: https://lwvworc.org/ro/do-stochastic-gradient-descent. [Accesat 08 Mai 2023]. |
| [8] | „javaTpoint,” [Interactiv]. Available: https://www.javatpoint.com/cost-function-in-machine-learning. [Accesat 08 Mai 2023]. |
| [9] | J. Brownlee , „Machine Learning Mastery,” 21 Octombrie 2019. [Interactiv]. Available: https://machinelearningmastery.com/cross-entropy-for-machine-learning/. [Accesat 08 Mai 2023]. |
| [10] | T. Dettmers, „Nvidia Developer Blog,” 03 Noiembrie 2015. [Interactiv]. Available: https://developer.nvidia.com/blog/deep-learning-nutshell-core-concepts/. [Accesat 24 Martie 2023]. |
| [11] | Z. Zixuan și L. Xuejiao , „Speech Command Recognition with Convolutional Neural,” *cs229.stanford.edu Final Project,* 2017. |
| [12] | K. Simonyan și A. Zisserman, „Very Deep Convolutional Networks for Large-Scale Image Recognition,” *ICLR 2015,* p. 14, 10 Aprilie 2015. |
| [13] | M. Hassan, „Neurohive,” 20 Noiembrie 2018. [Interactiv]. Available: https://neurohive.io/en/popular-networks/vgg16/. |
| [14] | A. Krizhevsky, I. SutsKever și G. Hinton, „ImageNet classification with deep convolutional neural networks,” *Communications of the ACM,* p. 7, 24 Mai 2012. |
| [15] | A. Rastogi, „blog.devgenius.io,” Mai 2022. [Interactiv]. Available: https://blog.devgenius.io/alexnet-the-net-that-surpassed-cnns-5d551ba1b901. [Accesat 26 Martie 2023]. |
| [16] | F. Iandola, S. Han, M. Moskewicz, K. Ashraf, W. Dally și K. Keutzer, „SqueezeNet: AlexNet-level accuracy with 50x,” p. 13, 4 Noiembrie 2016. |
| [17] | „Wikipedia,” 06 Aprilie 2020. [Interactiv]. Available: https://en.wikipedia.org/wiki/Cepstrum. [Accesat 01 Mai 2023]. |
| [18] | T. Singh, „Medium,” 15 Iunie 2019. [Interactiv]. Available: https://medium.com/@tanveer9812/mfccs-made-easy-7ef383006040. [Accesat 04 Aprilie 2023]. |
| [19] | „Electronics Tutorials,” [Interactiv]. Available: https://www.electronics-tutorials.ws/resistor/res\_2.html. |
| [20] | J. Lyons, D. Y.-B. Wang, H. Shteingart, E. Mavrinac, Y. Gaurkar și W. Watcharawisetkul, „jameslyons/python\_speech\_features,” 14 Ianuarie 2020. [Interactiv]. Available: https://zenodo.org/record/3607820#.ZCMOYcpBxPZ. |
| [21] | Y. Geifman, „cifar-vgg,” 27 Martie 2018. [Interactiv]. Available: https://github.com/geifmany/cifar-vgg. [Accesat 29 Martie 2023]. |
| [22] | Z. Hancock, „SqueezeNet\_vs\_CIFAR10,” 13 Ianuare 2019. [Interactiv]. Available: https://github.com/zshancock/SqueezeNet\_vs\_CIFAR10. |
| [23] | F. Chollet, Deep Learning With Python, 2017. |
| [24] | A. Verma, „towardsdatascience,” 28 Iulie 2020. [Interactiv]. Available: https://towardsdatascience.com/alexnet-8b05c5eb88d4. [Accesat 26 Martie 2023]. |
| [25] | „A Guide to Mel Frequency Cepstral Coefficients (MFCCs),” [Interactiv]. Available: http://www.practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/. |
| [26] | P. C. Loizou, Speech Enhancement: Theory and Practic, 2013. |
| [27] | B. Ehrismann și H. Drück, „ResearchGate,” [Interactiv]. Available: https://www.researchgate.net/figure/Basic-neural-network-unit-neuron-node-McCulloch-and-Pitts-1943\_fig3\_267122431. [Accesat 29 Aprilie 2023]. |
| [28] | N. Kumar, „The Professionals Point,” 6 Iunie 2019. [Interactiv]. Available: http://theprofessionalspoint.blogspot.com/2019/06/dying-relu-causes-and-solutions-leaky.html. [Accesat 30 Aprilie 2023]. |
| [29] | N. Kumar, „TheProfessionalsPoint,” 06 Iunie 2019. [Interactiv]. Available: http://theprofessionalspoint.blogspot.com/2019/06/dying-relu-causes-and-solutions-leaky.html. [Accesat 30 Aprilie 2023]. |

# Anexe

## Anexa A



## Anexa B

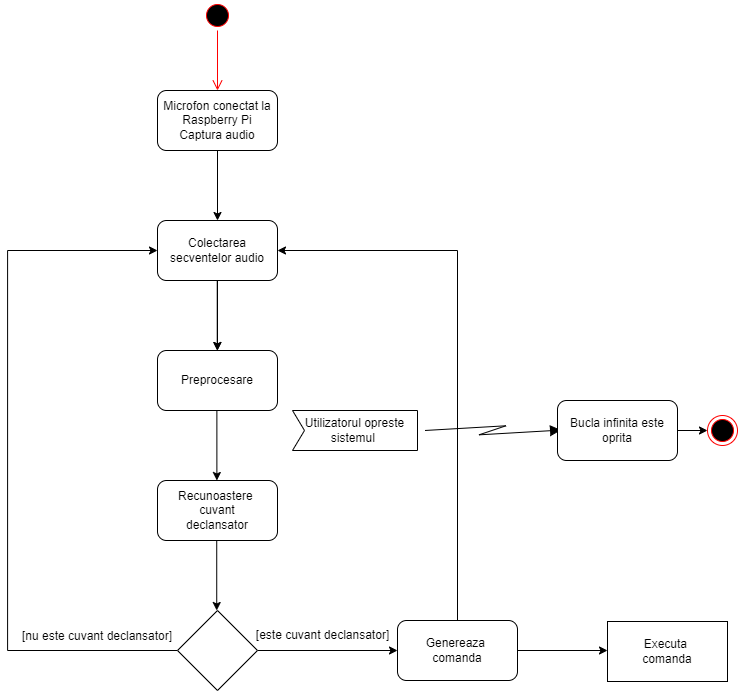


Fig. 10.1 Diagrama de activitate a sistemului

## Anexa C

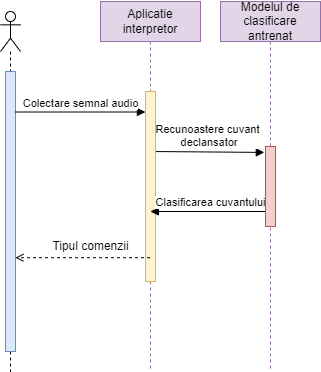


Fig. . Diagrama cazurilor de secvență

## Anexa D