Separarea surselor de semnal folosind măști de frecvență

Contents

[1. Introducere 3](#_Toc9762817)

[2. Motivația temei 4](#_Toc9762818)

[3. Tehnologii folosite 5](#_Toc9762819)

[3.1. Python 5](#_Toc9762820)

[3.1.1. Descriere 5](#_Toc9762821)

[3.1.2. Jupyter Notebook 5](#_Toc9762822)

[3.1.3. Pachetul “numPy” 5](#_Toc9762823)

[3.1.4. Pachetul “PyTorch” 5](#_Toc9762824)

[3.1.5. Pachetul “LibROSA” 5](#_Toc9762825)

[3.2. Transformata Fourier 6](#_Toc9762826)

[3.2.1. Semnalele și caracteristicile sunetului 6](#_Toc9762827)

[3.2.2. STFT (short-time frequency tranform) 7](#_Toc9762828)

[3.2.3. Spectrograma 10](#_Toc9762829)

[3.2.4. ISTFT (inverse short-time frequency transform) 11](#_Toc9762830)

[3.2.5. Măști de frecvențe în timp 12](#_Toc9762831)

[3.3. Articol îndrumător “Low-latency sound source separation using deep neural networks” 13](#_Toc9762832)

[3.3.1. NMF 14](#_Toc9762833)

[3.4. Deep learning 14](#_Toc9762834)

[3.4.1. Relația dintre dezvoltarea cognitivă și creierul uman 15](#_Toc9762835)

[3.4.2. Cum funcționează o rețea neuronală 15](#_Toc9762836)

[3.4.2.1. Ce este o rețea neuronală? 16](#_Toc9762837)

[3.4.2.2. Cum funcționează o rețea neuronală? 17](#_Toc9762838)

[3.4.2.3. De ce rețelele neuronale sunt capabile să învețe? 19](#_Toc9762839)

[3.4.3. Deep neural networks ( Rețele neuronale profunde) 20](#_Toc9762840)

[4. Descrierea soluției 21](#_Toc9762841)

[4.1. Obiective 21](#_Toc9762842)

[4.2. Implementare 22](#_Toc9762843)

[4.2.1. Construirea unui set de date de antrenare/validare/testare 22](#_Toc9762844)

[4.2.2. Încărcarea semnalelor și mixarea lor 25](#_Toc9762845)

[4.2.3. Strutura rețelei neuronale 26](#_Toc9762846)

[4.2.4. Workflow-ul rețelei neuronale 26](#_Toc9762847)

[4.2.5. Antrenarea rețelei 26](#_Toc9762848)

[4.2.6. Testarea rețelei 26](#_Toc9762849)

[4.3. Rezultate și performanțele rețelei 26](#_Toc9762850)

[5. Concluzii 26](#_Toc9762851)

[6. Index figuri 26](#_Toc9762852)

[1. Bibliografie și webografie 28](#_Toc9762853)

# Introducere

Separarea surselor de semnal folosind măști de frecvență urmărește faimoasa problemă cunoscută sub numele de „cocktail party”, sau „blind source separation”.

Efectul de „cocktail party” constă în abilitatea creierului de a se focusa preponderent doar asupra unei voci, surse de semnal sau conversație, într-o cameră plină de oameni, muzică și alte sunete de fundal, astfel încât să filtreze restul sunetelor care nu reprezintă un interes pentru interlocutor. Ascultătorii au abilitatea de a segrera multiple voci în surse diferite și, subconștiental, să decidă care dintre surse reprezintă interes într-un procentaj mai mare.

Lucrarea este compusă din cinci capitole, care împreună descriu pașii urmați pentru atingerea scopului propus.

Pentru început este prezentat capitolul „Introducere”, în care este descris, pe scurt, în ce constă proiectul, tema propusă și ce conține documentația actuală.

Al doilea capitol, „Motivația temei”, prezintă ce probleme sunt întâmpinate pentru oferirea unei separări cât mai corecte a celor două surse de semnal. .

Al treilea capitol, „Tehnologii folosite”, enumeră și detaliază ce tehnologii au ajutat la dezvoltarea proiectului curent. Tot aici este prezentat, pe scurt, articolul suport pe care s-a bazat proiectul.

În capitolul „Descrierea soluției”, este descris modul în care tehnologiile au fost îmbinate cu articolul îndrumător pentru realizarea aplicației finale. Sunt prezentate și etapele prin care trec sursele de semnal: de la sunetele inițiale, nealterate, se creează o mixtură între cele două semnale (voci), urmând antrenarea rețelei neuronale profunde din care va rezulta o mască. Această mască va fi aplicată pe mixtura celor două surse de semnal, rezultând două întregistrări în care cele două surse sunt separate una de cealaltă.

Acestă aplicație a fost realizată cu ajutorul domnului lector universitar doctor Gâlmeanu Honorius.

# Motivația temei

Problema de tip „cocktail” a fost populară din 1953, după publicarea unui articol pe această temă a lui Cherry, iar în ultima vreme a fost considerată a fi o mare provocare pentru rețelele neuronale. Aceasta constă în filtrarea sunetelor astfel încât să se audă doar ce este de interes pentru interlocutor. Problema, în mod intrinsec, este destul de dificilă, și mereu a fost arătat un interes deosebit de mare pentru modul în care oamenii reușesc să o rezolve, în mod natural, anatomic.

Problema nu este unică pentru oameni, multe alte specii se confruntă cu probleme similare. De exemplu, multe specii de broaște folosesc vocalizări, specifice speciei lor, pentru a-și localiza perechea noaptea, când vizibilitatea e scăzută și sunt foarte puține repere vizuale și multe alte broaște în jur.

Există două provocări distincte, din punct de vedere conceptual, pentru un ascultător în această situație.

Prima provocare este segregarea sunetului. Sunetele într-o scenă, toate unele peste altele, generează semnalul care intră mai departe în ureche. Totuși, această mixtură oferă în sine puțină informație pentru organism, care în sine este interesat într-o sursă de semnal particulară. Sistemul auditiv trebuie să derive proprietățile sunetelor individuale din mixtura care intra în ureche.

A doua provocare este alocarea atenției doar asupra sursei de semnal care prezintă interes, ignorând celelalte, și schimbarea atenției între surse, în timp ce se urmărește o conversație cu două surse de semnal diferite. Majoritatea proceselor noastre cognitive pot opera doar pe un singur lucru o data, în consecință alegem o singură sursă de semnal pe care să ne putem focusa.

Urechea umană face o descompunere a frecvențelor pe sursele de semnal, similar unei transformate Fourier, cu ajutorul căreia se poate verifica spectrograma frecvențelor sunetului.

Articolul întrumător acestei lucrări propune o abordare folosind o rețea neuronală profundă, formată din trei straturi de tip „fully connected”. În urma elaborării lucrării, s-a obervat că rețeaua nu este una din cele mai puternice pentru această problemă propusă, dar oferă un rezultat bun pentru surse de semnal înregistrate într-o cameră fără alte sunetele de fundal.

# Tehnologii folosite

## Python

### Descriere

### Jupyter Notebook

### Pachetul “numPy”

### Pachetul “PyTorch”

### Pachetul “LibROSA”

## Transformata Fourier

Din punct de vedere matematic, transformata Fourier reprezintă operația aplicată asupra unei funcții cu elemente complexe, și returnează tot o funcție cu elemente complexe. Ce este diferit la funcția returnată este că este reorganizată după frecvențele componente.

Analiza Fourier este considerată abecedarul analizei semnalelor. Aceasta tranformă semnalul nostru din domeniul timpului în domeniul frecvențelor. În timp ce domeniul timpului exprimă semnalul într-o secvență de probe/bucăți de semnal, domeniul frecvențelor reprezintă semnalul ca sinusoidale, cu diferite magnitudini, frecvențe, și decalaje de fază a semnalului.

### Semnalele și caracteristicile sunetului

Semnalul în timp este reprezentat printr-o anumită funcție, care de regulă dă forma de dependență a nivelului sau amplitudinii semnalului în raport cu timpul.

Semnalele pot fi de două feluri, periodice și neperiodice. Cele periodice sunt caracterizate printr-o perioada care se tot repetă în decursul semnalului, iar cele nepriodice nu conțin această perioadă de repetiție.

La rândul lor, semnalele perioduce sunt de două feluri, armonice și nearmonice.

Semnalul audio este un semnal neperiodic, de cele mai multe ori, fiind determinat de sunete neperiodice. Din punct de vedere electric, semnalul audio este reprezentat de variația în timp a tensiunii sau curentului electric.

Sunetul este caracterizat de patru elemente:

* amplitudinea – caracteristica undei sonore percepute ca volum
* frecvența/înălțimea sunetului – numărul de perioade ( T -> s) sau oscilații efectuate

( F -> HZ) într-un timp dat

* intensitatea sunetului/sonoră – este măsurată în decibeli
* timbrul – calitatea de bază a fiecărui sunet, determinat de componenta de armonici

În procesare semnalelor se ține foarte mult cont și sunt controlate frecvența și intensitatea sunetelor. În tema aleasă toate înregistrările sunt modificate la 16KHz, iar analiza pentru aflarea măștilor de frecvență se bazează pe ideea de „frecvențe în timp”.

### STFT (short-time frequency tranform)

Transformata Fourier pe termen scurt (STFT) este o transformare de tip Fourier care ajută la determinarea frecvenței și fazei sinusoidalei a secțiunilor locale ale unui semnal, având în vedere schimbarea lui în timp.

Pentru calculul STFT se împarte un semnal mai lung în segmente mai scurte, de dimensiuni egale, iar mai apoi se calculează tranformata Fourier, separat, pe fiecare segment mai scurt. La final, aceasta duce la aflarea spectrului Fourier pentru fiecare segment mai scurt. După se poate arăta spectrul, printr-un desen, ca o funcție a timpului.

Astfel, tranformata Fourier pe termen scurt este o secvență de transformări Fourier a unui semnal ferestruit. Pentru această transformare trebuie setați mai mulți parametrii, și anume:

* fereastra de analiză ( de exemplu 10/20/30 ms)
* valoarea de „overlap”/„hopping”, cât de mult se dorește suprapunerea ferestrelor în mutarea către următoarea fereastră ( de exemplu overlap de 50%, pentru dimensiunea ferestrelor de 10 ms se vor lua ferestre în felul următor:

0 ms → 10 ms, 5 ms → 15 ms, 10 ms → 20 ms, și așa mai departe)

* funcția de ferestruire ( de exemplu Hann, Gaussian, etc. )
* generarea segmentelor ferestruite
* aplicarea tranformatei Fourier pe fiecare segment ferestruit

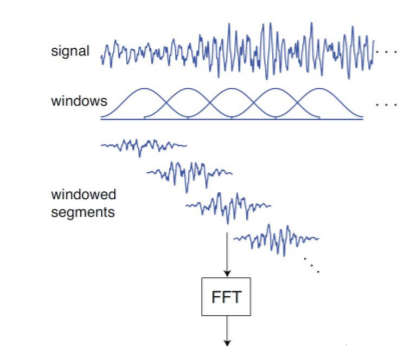


Figura 1 : Pașii aplicării transformatei Fourier pe un semnal

Aplicarea algoritmului de STFT constă în respectarea unor pași, și anume:

1. Se alege un segment de dimensiune mică din întregul semnal:

* se folosește conceptului de „sliding window” (fereastră care se plimbă) pentru a fi mai ușor de vizualizat ce se întâmplă
* datele dinăuntrul ferestrei reprezintă segmentul curent care va fi procesat
* dimensiunea ferestrei rămâne aceeași pe tot parcursul procesării datelor, dar punctul de pornire („offsetul”) se va schimba după fiecare pas din algoritm
* de obicei, schimbarea acestui „offset” constă într-o valoare mai mică decât dimensiunea ferestrei, pentru a face posibil fenomenul de „overlap”, astfel încât fereastra precedentă și cea curentă se vor suprapune

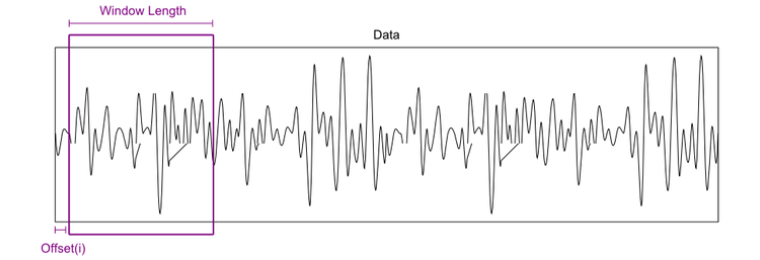


Figura 2 : Fereastră care se plimbă peste un semnal

1. Se înmulțește segmentul cu o funcție de jumătate de cosinusoidă:

* folosită pentru a evita problemele create de segmentarea unui semnal
* în momentul în care tăiem un semnal în bucățele mai mici, marginile creează o tranziție ascuțită care nu exista înainte în semnal
* înmulțind cu această funcție „jumătate cosinusoidală”, ajută la atenuarea semnalului, astfel încât tranzițiile la margini să nu afecteze transformata Fourier a datelor

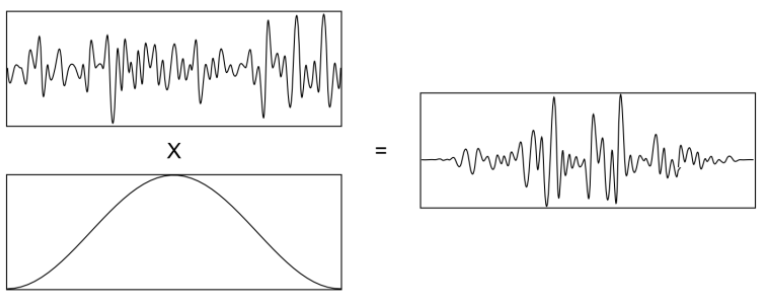


Figura 3 : Datele unui semnal după aplicarea funcției "half-cosine"

* alt atuu al acestei funcții, este că având un overlap de 50%, suma tuturor ferestrelor este o constată egală cu 1; acest lucru înseamnă că înmulțind datele noastre cu o anumită funcție, nu se vor adăuga diferențe în amplitudinea semnalului original

1. se adaugă zerouri la finalul segmentului: **TODO**
2. aplicarea transformatei Fourier asupra segmentului și normalizarea în frecvențe pozitive și negative:

* numărul the bucăți de date care intră este egal cu numărul de bucăți de frecvențe care ies
* datele date de transformarea Fourier trebuie să fie scalate cu numărul de bucăți din transformare pentru a menține egala energia semnalului ( în concordanță cu teorema lui Parseval)
* transformata Fourier returnează atât frecvențele pozitive, cât și cele negative

1. combinarea energiilor frecvențelor pozitive și negative la un loc și crearea unui grafic
2. scalarea spectrului rezultat în dB pentru a fi mai ușor de văzut:

* convetirea datelor în dB are efectul de a lungi vârfurile frecvențelor în jos, spre nivelul mediu al sunetului, și de a aduce valorile foarte joase în sus.
* acest lucru permite compararea conținutului la toate nivelele de amplitudine

1. înlăturarea zgomotului care se află peste o anumită limită:

* tot ce este sub -40 dB este sub limita minimă a frecvențelor, astfel încât se consideră ca este frecvențe sunt erori numerice din algoritm

### Spectrograma

„Spectrograma este reprezentarea vizuală a unui spectru (a densității spectrale) de frecvențe a unui sunet în raport cu timpul sau cu alte variabile [[1]](#footnote-1).”

Aceasta este de obicei asociată cu un „heat map”, o imagine cu insitatea dată de varietatea culorilor și a luminozității.

Formatul general este un grafic cu 2 dimensiuni plus una separată pentru amplitudine:

* o axă reprezintă timpului, cea orizontală
* o axă reprezintă frecvențele, cea verticală
* o a treia dimensine ar fi cea a amplitudinilor frecvențelor la un anumit timp, care sunt reprezentate de intensitatea de culoare a acelui punct

Există multe moduri pentru desenarea graficul spectrogramei, uneori axele orizontale și verticale sunt inversate, astfel încât timpul se desfășoară în sus și în jos, alteori graficul este de tip „waterfall” unde amplitudinea este reprezentată de înălțimea unei suprafețe 3D în locul intensității sau culorii. Axele de frecvență și amplitudine pot fi ori lineare, ori logaritmice, în funcție de scopul spectrogramei. Semnalele audio sunt reprezentate de cele mai multe ori cu o axă logaritmică a amplitudinii (decibeli, dB) și frecvențele ar putea fi liniare pentru a menține relația armonică, sau logaritmică pentru a menține relațiile muzicale/tonurile.

În continuare sunt prezentate trei spectrograme, folosite în aplicație. Prima spectrogramă este mixul dintre următoarele două. Spectrograma a doua și a treia sunt două voci diferite, una de femeie și una de bărbat. Cum este specificat mai sus, pe axa orizontală avem timpul, pe cea verticală sunt frecvențele, iar în dreapta avem o scară pentru măsurarea amplitudinii în dB.

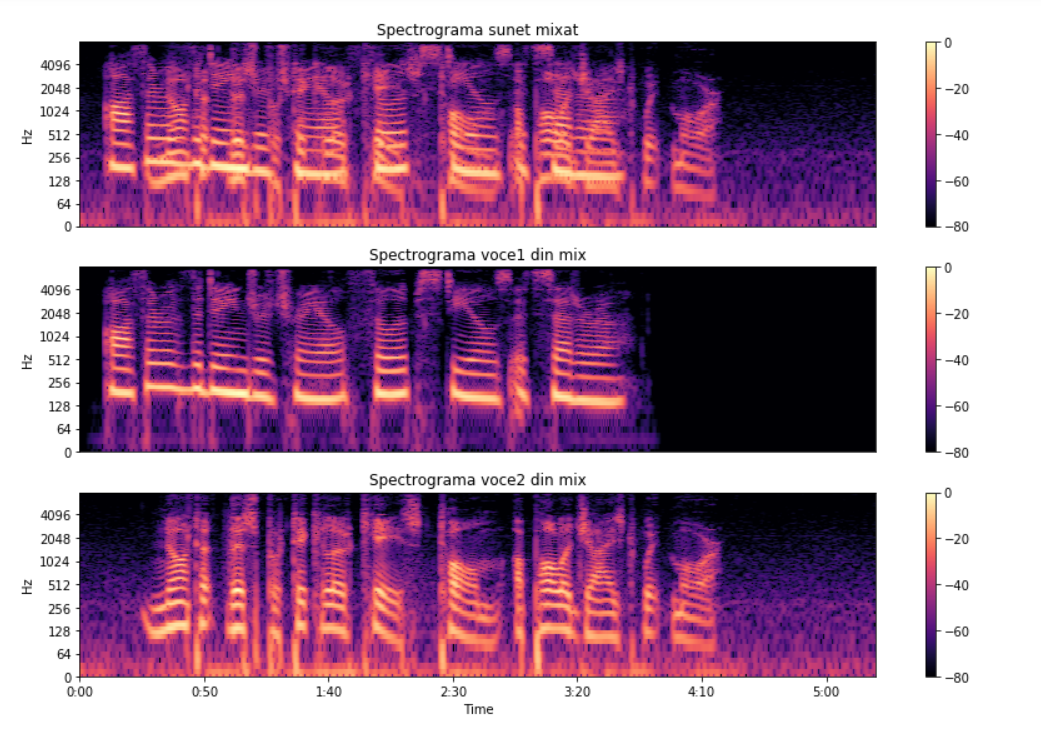


Figura 4 : Spectrograme folosite în aplicație pentru două voci și mixul lor

Pentru calculul spectrogramei se va calcula magnitudinea la pătrat, unde magnitudinea este valoarea absolută: spectrogram(f, t) = | STFT(t, f) |2.

### ISTFT (inverse short-time frequency transform)

STFT este o operație inversabilă, însemnând că după aplicarea transformării STFT asupra semnalului audio, acesta poate fi recuperat folosind operația de transformare inversă Fourier, dar doar atâta timp cât matricea respectivă mai conține elementele complexe (faza semnalului). Dacă se face transformarea matricii către spectrogramă/magnitudine, din acel punct transformarea inversă Fourier nu mai are niciun efect, deoarece elementele complexe ale semnalului s-au pierdut în momentul aflării valorii absolute pentru magnitudine.

Dacă se dorește să se modifice spectrul complex al semnalului, acesta mai poate fi recuperat folosind metoda „overlap – add (OLA)”.

### Măști de frecvențe în timp

Se presupune că avem două surse, un semnal pentru voce și un semnal pentru muzică. Presupunem că mixul celor două surse este risipit, adicpă majoritatea punctelor de tip frecvență – timp sunt nule.

Presupunem că primul semnal, muzica, are majoritatea punctelor de energie joasă, iar al doilea semnal are majoritatea punctelor de energie înaltă.

Se mai presupune și că cele două semnale sunt disjuncte, adică punctele lor de tip frecvență – timp nu se suprapun, sau se suprapun foarte puțin.

Având mixul risipit și disjuncția, cele două surse din mix se pot discrimina destul de ușor. Punctele care au probabilitatea să aparțină de o sursă sunt marcate cu valoarea constată „1”, iar restul cu „0”. Astfel rezultă o mască binară (varianta mai riguroasă a algoritmului).

După calcularea măștii, prin înmulțirea măștii cu sursa mixată, se va obține sursa pentru care s-a calculat masca.

Figura 5 : Calcularea măștii de frecvență cu respect pentru prima sursă de semnal

În majoritatea cazurilor, mixurile nu sunt foarte risipite și nici sursele disjuncte. Punctele din mix care sunt probabil să aparțină unei surse au valori aproape de valoarea 1, iar restul aproape de valoarea 0 ( varianta de „soft – masking”). Ecuația cu respect pentru prima sursa se semnal din mix, este următoarea :

Figura 6 : Calcularea măștii de frecvență cu respect pentru ambele surse de semnal

Având masca pentru o sursă, pentru cealaltă sursă masca se poate calcula fie în același mod, dar înlocuind numărătorul cu sursa pentru al doilea semnal, fie scăzând masca dintr-o matrice construită, care va avea aceeași dimensiune și va conține doar valori de 1.

## Articol îndrumător “Low-latency sound source separation using deep neural networks”

Separarea sursei de semnal la o latență scăzută necesită ca fiecare input, o fereastră audio, să fie procesată la o temporizare cât mai mică, iar outputul să fie generat cât mai repede.

Articolul propune o rețea neuronală profundă, deep neural network (DNN), cu latență scăzută. Această metodă are avantajul de a folosi un context extins „din trecut”, mai apoi generându-se filtre/măști de tipul „soft time frequencies masks”, care mai apoi sunt aplicate peste sunetul mixat, astfel încât să fie oferită o performanță de separare mai bună, comparând cu „NMF baseline”.

### NMF

Separarea surselor poate fi văzută ca o problemă de factorizare a matricei, unde sursa mixtură este modelată ca o matrice care conține reprezentarea spectrogramei aferente.

Spectrogramele sunt de obicei folosite pentru vizualizarea timpului cu variația spectrală

a densității unei înregistrări audio. Semnalele audio pot, astfel, să fie reprezentate în întregime de o matrice cu :

* linii, axa orizontală ( reprezentarea timpul)
* coloane, axa verticală ( reprezentarea frecvențelor)
* valori ale elementelor, care corespund intensitatății sau culorii fiecărui punct din imagine ( indicând amplitudinea unei frecvențe particulare la un anumit moment „t”).

Spectrograma unui semnal x(t) poate fi estimată prin calculul magnitudinii la pătrat a

transformatei Fourier (prezentată anterior la secțiunea TODO), a semnalului x(t), astfel încât x(t) poate fi reconstruit din spectrogramă prin inversa transformatei Fourier (prezentată de asemenea în secțiunea TODO), după procesarea semnalului în domeniu spectral.

## Deep learning

Deep learningul (învățarea produndă), cunoscută și ca învățare ierarhică sau învățare profundă structurată, face parte din familia “machine learningului”, care se bazează pe rețele neuronale artificiale. Învățarea poate fi nesupervizată, supervizată sau semi-supervizată.

Arhitecturi ale învățării profunde precum rețele neuronale profunde, rețele neuronale recurente sau rețele neuronale convoluționale au fost aplicate de-a lungul timpului la recunoaștere vocală, procesare de limbaj natural, recunoaștere audio, filtrare a rețelelor sociale, traduceri, analiză medicală imagistică, inspecția meterialelor, și așa mai departe. Rețelele neurale au fost inspirate de procesarea informațiilor și comunicarea distribuită ale nodurilor din sistemele biologice. În mod natural, rețelele neuronale tind să fie statice și simbolice, în timp ce creierul biologic al diferitelor organisme în viață sunt dinamice și analogice.

Deep learningul este o clasă de algoritmi ai “machine learningului”, care folosesc mai multe straturi progressive pentru extragerea caracteristici de nivel tot mai înalt, din intrări de tip “raw”, adică neprelucrate. De exemplu, în procesarea de imagini, straturile de la început, sau de nivel mai jos, vor extrage margini sau colțuri, pe când straturile de nivel mai înalt vor extrage caracteristici care au o însemnătate pentru om, precum litere, cifre sau fețe.

### Relația dintre dezvoltarea cognitivă și creierul uman

“Deep learning” este în strânsă legătură cu o clasă de teorii a dezvoltării creierului (în special, dezvoltarea neocortexului) propusă de neurologi cognitivi în anii 1990. Aceste teorii ale dezvoltării au fost folosite în modele computaționale, creând astfel predecesorii sistemelor de “deep learning”.

Precum neocortexul, rețelele neuronale implică o ierarhie de filtre în straturi în care fiecare strat ia în considerare informația de la un strat anterior, iar mai apoi îl pasează mai departe ca rezultat la celelalte straturi. Acest proces este o stivă care se organizează singură.

O multitudine de moduri au fost folosite pentru investigarea plauzibilității modelelor de învățare profundă dintr-o perspectivă neurobiologică. Pe de o parte, câteva variante de algoritmi de “backpropagation” au fost propuși în așa fel încât să crească procesarea realistă a informației. Anumiți oameni de știință au argumentat că formele de învățare nesupervizată a învățării profunde, ca cele bazate pe modele generative ierarhice și grafice, sunt mai aproape de realitatea biologică.

Deși o comparație sistematică între organizația creierului uman și codificarea neuronală în rețele profunde nu a fost stabilită, câteva analogii au fost raportate. De exemplu, computația realizată de unitățile învățării profunde pot fi similar cu cea a neuronilor și populațiilor de neuroni. În mod similar, reprezentări dezvoltate de modele de învățare profundă sunt similare cu cele măsurate în sistemul visual primar, atât la nivel de unitate cât și la nivel de populație.

### Cum funcționează o rețea neuronală

În primă instanță, rețelele neuronale par un domeniu misterios. În timp ce există o cale intuitivă de a înțelege modelele liniare sau arborii de decizie, rețelele neuronale nu au explicații chiar atât de clare. La prima vedere, rețelele neuronale au un strat de intrare, care transferă datele către straturile ascunse, iar după puțină magie se poate observa informația dată de stratul de ieșire. În principiu, înțelegerea straturilor ascunse și a ceea ce ele realizează este cheia către implementarea și optimizarea rețelelor neuronale.

Rețelele neuronale reprezintă unii dintre cei mai puternici și folosiți algoritmi când vine vorba despre subdomeniul numit „deep learning” care aparține de „machine learning”.

Mai jos este prezentat arborele tuturor domeniilor și subdomeniilor ai inteligenței artificiale.

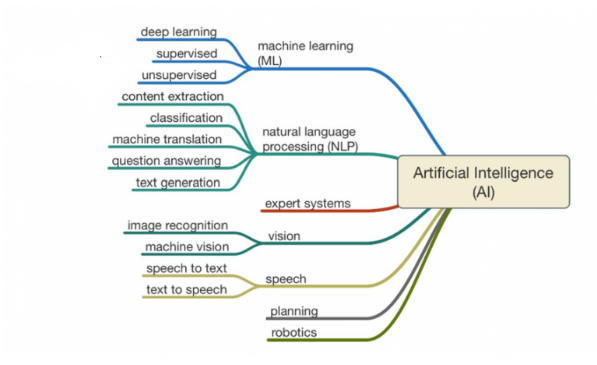


Figura 7 : Arbore structural al inteligenței artificiale și a tuturor subdomeniilor sale

#### Ce este o rețea neuronală?

Pe scurt, o rețea neuronală este un sistem computațional care constă într-un număr de elemente sau noduri simple dar cu un nivel de interconectibilitate foarte ridicat, numiți și neuroni, organizați pe straturi, acestea procesând informația și dând răspunsuri în stări dinamice (care se pot schimba în funcție de datele primite la intrare).

Algoritmul este extrem de folositor în aflarea modelelor caracteristice care sunt prea complexe pentru extragerea lor manuală, pentru ca mai târziu să fie date mașinii să învețe să recunoască diferențele dintre ele.

În contextul acestei structuri prezentate mai sus, șabloanele sau modelele caracteristice sunt introduse în rețeaua neuronală de stratul de intrare și sunt transmise în unul sau mai multe straturi ascunse prezente în rețea; straturile sunt numite „ascunse” doar pentru că nu constituie stratul de intrare sau de ieșire. Totuși, în aceste straturi ascunse, are loc toată procesarea datelor, printr-un sistem de conexiuni, caracterizat de ponderi („weights”) și valori de tip bias („biases”).

Datele de intrare sunt primite, neuronul calculează o sumă ponderată, adăugând și valoarea biasului, și în concordanță cu rezultatul și o funcție de activare ( cea mai comună este sigmoida, chiar dacă nu prea mai este folosită acum și există altele mai bune, precum ReLU, care vor fi prezentate în secțiunea „Descrierea soluției”) decide dacă trebuie să fie „fired” sau activat. După aceea, neronul transmite informația în continuare celorlalți neuroni conectați cu el, într-un proces numit propagare înainte ( „forward pass”). La finalul acestui proces, ultimul strat ascuns este conectat cu stratul de ieșire, care are câte un neuron pentru fiecare posibil rezultat dorit.

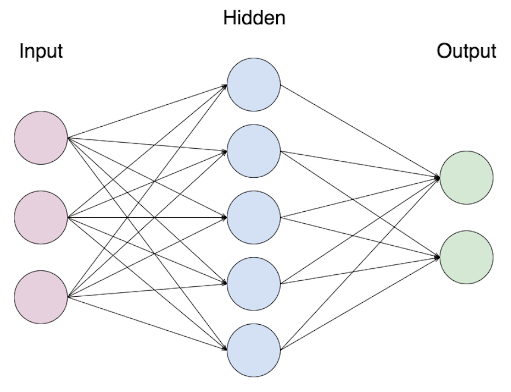


Figura 8 : Structura unei rețele neuronale

#### Cum funcționează o rețea neuronală?

Pentru a înțelege cum funcționează rețeaua trebuie înțeles mai întâi cum funcționează diferite tipuri de neuroni. Putem face o paralelă cu perceptronul, chiar dacă folosirea acestuia a scăzut în ultima perioadă, deoarece oferă o idee bine conturată despre cum funcționează neuronii mai moderni.

Un perceptron folosește o funcție pentru a învăța un clasificator binar mapând un vector de valori binare cu o singură valoare binară de ieșire și poate fi folosit de asemenea în învățarea supervizată. În acest context, perceptronul urmărește următorii pași:

* multiplică toate intrările cu ponderile lor, numere reale care exprimă importanța corespondenței dintre datele de intrare si cele de ieșire
* le adună într-o variabilă numită „sumă ponderată”
* se aplică o funcție de activare, în altă ordine de idei, se determină dacă suma ponderată este mai mare decât o valore prag impusă, unde acest prag este echivalent cu valoarea de bias, și i se asignează valoarea 1 sau 0 ca dată de ieșire

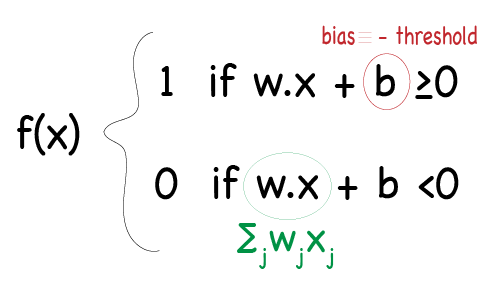


Figura 9 : Funcția perceptronul

Cel mai important lucru din acest algoritm este acela că putem varia ponderile și valoare biasului pentru a obține modele diferite de tipul „decision – making”. Putem asigna mai multă pondere (mai puternică) acestor intrări astfel încât daca ele sunt pozitive, vor favoriza ieșirea dorită de noi. De asemenea, pentru că valoarea de bias poate fi înțeleasă ca o măsură a nivelului de dificultate în aflarea ieșirii L, putem scădea sau crește valoare acestuia pentru a ni se oferi rezultatul dorit ca ieșire.

În consecință, un perceptron poate analiza diferite evidețe sau date și să ia o decizie având în vedere preferințele setului de date. Este posibilă crearea unor rețele mult mai complexe, incluzând mai multe straturi de perceptroni, unde fiecare strat primește ca intrare ieșirea stratului anterior, îl ponderează și reușeste să ia decizii din ce în ce mai complexe.

Unul dintre dezavantajele unei rețele care conține perceptroni este faptul că schimbări minore ale ponderilor sau valorii de bias, chiar dacă este doar într-un singur perceptron, poate schimba radical rezultatul, schimbând din valoarea 1 în 0 și vice versa.

Ce se dorește în domeniu este să se poată face schimbări minore asupra ponderilor sau valorii de bias, iar aici intervin tipurile moderne de neuroni, de exemplu neuronii de tip sigmoidă. Diferența majoră dintre un neuron sigmoidă și un perceptron este faptul că intrarea și ieșirea pot fi orice valoare continuă aflată îm intervalul deschis [0,1]. Ieșirea este obținută după aplicarea funcției sigmoide de activare asupra intrărilor, considerând ponderile și valoarea biasului.

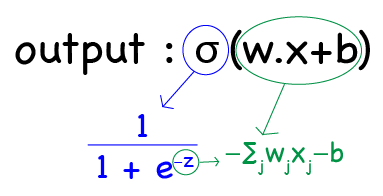


Figura 10 : Formula pentru aflarea valorii de ieșire

#### De ce rețelele neuronale sunt capabile să învețe?

Atuul principal al algoritmilor de „machine learning” este abilitatea lor de a învăța și a se îmbunătății de fiecare dată când prezic o ieșire. În contextul rețelelor neuronale, faptul că reușesc să învețe implică faptul că ponderile și valoarea de bias, care definesc conexiunea dintre neuroni, devin tot mai precise, adică ponderile și valoarea biasului sunt selectate în așa fel încât ieșirea din rețea aproximează valoarea reală a y(x) pentru toate intrările de antrenare.

Pașii prin care se trece într-o rețea neuronală sunt următorii:

* alegerea datelor de intrare : se ia un set de antrenare și se setează funcția de activarea „a” care corespunde stratului de intrare
* pasul de „feedforward” : pentru fiecare strat, se calculează funcția z = w \* a + b, unde a = σ(z)
* calculul erorii : se calculează eroarea folosindu-se formula :



Figura 11 : Formula de calcul al erorii

* pasul de „backpropagation” : aici se propagă înapoi eroarea; pentru fiecare strat, se calculează formula

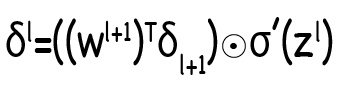


Figura 12 : Formula de calcul pentru propagarea înapoi

* afișarea rezultatelor : se calculează gradientul descent cu respect pentru fiecare pondere și bias folosindu-se formulele :

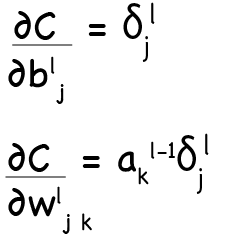


Figura 13 : Formulele de calcul pentru actualizarea ponderilor și a biasului

Desigur, există mult mai multe concepte, implementări și îmbunătățiri care pot fi făcute asupra rețelelor neuronale, care pot deveni din ce în ce mai folosite pentru mai multe domenii și mai puternice decât anii precedenți.

În „Descrierea soluției”, se va prezenta structura rețelei neuronale folosite pentru implementarea temei și vor fi explicate, pas cu pas, toate conceptele folosite în cadrul dezvoltării temei.

### Deep neural networks ( Rețele neuronale profunde)

O rețea neuronală profundă (DNN) este o rețea neuronală aritificială cu multiple straturi aflate între straturile de intrare – ieșire. Rețeaua neuronală profundă găsește manipularea corectă matematică pentru a transforma intrarea în ieșire, indiferent dacă este o relație lineară sau non – lineară. Rețeaua se plimbă printre straturi, calculând probabilitatea fiecărei ieșiri. De exemplu, o rețea neuronală profundă care este antrenată să recunoască rasele de câini va primi o imagine, și va calcula probabilitatea ca acel câine din imagine să aparțină unei anumite rase. Utilizatorul poate să vadă rezultatele și să selecteze ce probabilități ar trebui ca rețeaua să arate (mai mare de un anumit prag, etc.) și să returneze o etichetă propusă.

Fiecare manipulare matematică este considerată a fi un strat, iar rețelele neuronale profunde complexe au foarte multe straturi, de unde și numele de “profunde”. Scopul este ca, eventual, rețeaua să fie antrenată să descompună o imagine în caracteristici, să identifice trendurile care există printre toate mostrele și să clasifice imagini noi folosindu-se de similaritățile dintre ele, fără să aibă nevoie de ajutor uman.

Rețelele neuronale profunde pot modela relații non – lineare complexe. Arhitecturile rețelelor neuronale profunde generează modele compoziționale, unde obiectul este expus ca o compoziție stratificată de primitive. Straturile suplimentare permit compunerea de caracteristici din straturile inferioare, care pot modela date complexe cu mai puține unități decât o rețea superficială cu o performanță similară.

Arhitecturile profunde includ multiple variante. Fiecare arhitectură are perfomanțe bune în anumite domenii. Nu este întotdeauna posibilă compararea performanțelor a mai multor arhitecturi , decât atunci când ele au fost evaluate pe același set de date.

Rețelele neuronale profunde sunt, de obicei, rețele de tip “feedforward” (cu propagare înainte), în care datele sunt transmise de la stratul de intrare către stratul de ieșire, fără nicio buclă în sens invers. În primă instanță, rețeaua neuronală profundă creează o hartă a neuronilor virtuali și le asignează valori numerice aleatoare, sau așa numitele ponderi, pentru conexiunea dintre neuroni. Ponderile și valorile de intrare sunt multiplicate și returnează o valoare de ieșire în intervalul [0,1]. Dacă rețeaua nu recunoaște un anumit tipar în mod corect, un algoritm ar ajusta ponderile. Astfel, algoritmul poate face anumiți parametrii mai influenți, astfel încât manipularea matematică să fie cea corectă pentru procesarea datelor.

Rețelele neuronale recurente, în care datele se pot transfera în orice direcție, sunt folosite pentru aplicații precum modelarea limbajului. Pentru aceasta, foarte folositor este algoritmul “long short – term memory”.

Rețelele neuronale profunde convoluționale sunt folosite în “computer vision”. Aceste rețele au fost de asemenea aplicabile pe modelarea acustică pentru recunoaștere vocală.

# Descrierea soluției

## Obiective

Obiectivul principal al aplicației este separarea a două surse de semnal, care sunt mixate anterior. În principiu, se construiesc seturi de date pentru antrenare, validare și testare. Pentru fiecare set vor exista un număr de întregistrări atât pentru prima voce , cât și pentru a doua voce. Se va crea mixul pentru fiecare combinație de voci din fiecare set.

Rețeaua va primi un vector (tensor) alcătuit din mai multe ferestre de tip STFT (short time Fourier transformed). Aceste ferestre conțin date și din ferestrele anterioare, această abordare dând o îmbunătățire considerabilă a performanței în procesarea ferestrelor de dimensiuni mici (exemplu 20,10,5 milisecunde pe fereastră curentă, adică fără contextul din trecut incorporat, iar incorporând contextul din trecut fereastra va conține de 2 – 4 ori mai multe înregistrări decât cele din fereastra curentă).

Un alt obiectiv este testarea diferitelor valori date rețelei, astfel încât să se găsească rețeaua care se descurca cel mai bine, având acuratețea cea mai mare, fără a face „overfitting”, sau supra ajustare, adică rețeaua să învețe atât de bine setul de antrenare încât să nu știe deloc să prezică valorile pentru setul de validare sau testare.

Pentru testarea rețelei au fost modificați parametrii precum marja de eroare între masca prezisă de rețea și masca direct calculată cu formula pentru „soft time frequency masking”, numărul de epoci, rata de învățare, algoritmul de optimizare ( Adam, SGD), conținutul seturilor de antrenare, validare, testare.

În continuare vor fi prezentate modurile în care au fost construite seturile de date, fiecare în parte ce anume conține, cum au fost încărcate acestea și mixate, structura programului și a rețelei, ce se întamplă cu datele când trec prin rețea de la stratul de intrare, până la stratul de ieșire și performanțele rețelei.

## Implementare

### Construirea unui set de date de antrenare/validare/testare

Setul de date folosit este CMU ARCTIC Database[[2]](#footnote-2), unde se găsesc atât voci de femei, cât și de bărbați. Acestea au fost înregistrate la 16 bit 32 kHz, într-o cameră cu izolare fonică, stereo, iar un singur canal a fost sub formă de undă, iar celălalt EGG.

Acest set de date a fost folosit să se genereze mixurile folosite pentru testarea rețelei neuronale profunde. Au fost folosiți mai mulți vorbitori diferiți, atât femei cât și bărbați cu diferite accente.

Pentru generarea setului de antrenare, pentru fiecare vorbitor s-au folosit TODOnr înregistrări. Astfel setul de antrenare avea TODOnr mixuri generate, combinând fiecare înregistrare de la un vorbitor cu toate înregistrările celuilalt vorbitor. Toate înregistrările pentru setul de antrenare fac parte din setul A.

Setul de tastare a fost format din înregistrări din CMU Arctic set B, pentru a asigura faptul că setul de antrenare/validare și cel de testare sun disjuncte. Acesta conține TODOnr 10 înregistrări permutate similar ca pentru setul de antrenare, rezultând TODOnr 100 mixuri.

Înregistrărilor mai scurte de 5 secunde li se adăugau zerouri la final, până când devenea de 5 secunde, iar cele mai lungi de 5 secunde li se păstrau doar primele 5 secunde.

Toate înregistrările au fost încărcate folosind o rată de 16KHz, folosind biblioteca „LibRosa” din Python. Fiecare înregistrare va avez fix 5 secunde după procesare, și anume 80.000 de valori în vectorul de valori ale înregistrării.

Mixul se generează adunând efectiv valorile celor doi vectori ai înregistrărilor.

Fiecare înregistrare pentru mix, vocea 1 și vocea 2, sunt împărțite în ferestre mai mici de câte 20/10/5 milisecunde fiecare, aceasta reprezentând fereastra curentă (PF, processing frame) de acum încolo.

În rețeaua neuronală va intra un tensor, vector, care va conține un număr de ferestre de analizare (AF, analysis frame). Această fereastra de analizare este de 2-4 ori mai mare decât fereastra curentă. Ea este formată din fereastra curentă și ferestrele precendente. Pentru generarea ferestrei de analizare se folosește suprapunere de 50%. Acest proces este prezentat în Figura 14..

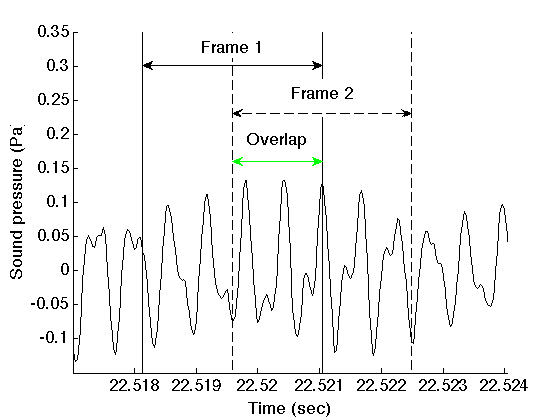


Figura 14 : Procesul de suprapunere a ferestrelor

După ce s-a creat ferestra de analiză, folosindu-se procesul de suprapunere, se aplică transformata Fourier pe termen scurt asupra AF-ului (analysis frame-ului), implementată în biblioteca LibRosa. Ca parametrii, funcția pentru STFT ( short time Fourier transform) primește fereastra pentru care se face transformarea, se alege metoda de „windowing” (în cazul nostru se folosește fereastra Hanning), dimensiunea ferestrei pentru transformata Fourier ( 1024 este valoarea folosită în aplicație), valoarea săriturii, adică numărul de coloane peste care sare pentru a trece la următoarea fereastră ( setat cu valoarea 8).

După calculul transformatei Fourier pe termen scurt, se calculează masca aferentă ferestrei curente și se salvează în lista de targeturi. Masca este calculată în felul următor:

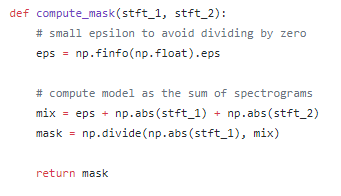


Figura 15 : Modul de calcul al măștii de frecvență pentru fereastra de procesare curentă

### Încărcarea semnalelor și mixarea lor

Semnalele sunt încărcate folosind biblioteca Python numită LibRosa, în felul următor :

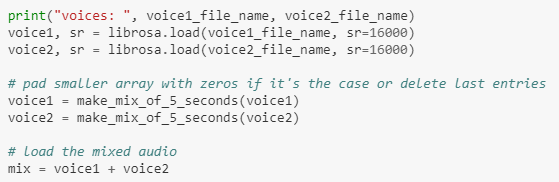


Figura 16 : Încărcarea semnalelor și mixarea lor

Mai departe, putem observa spectrogramele celor două voci și a mixului format din ele

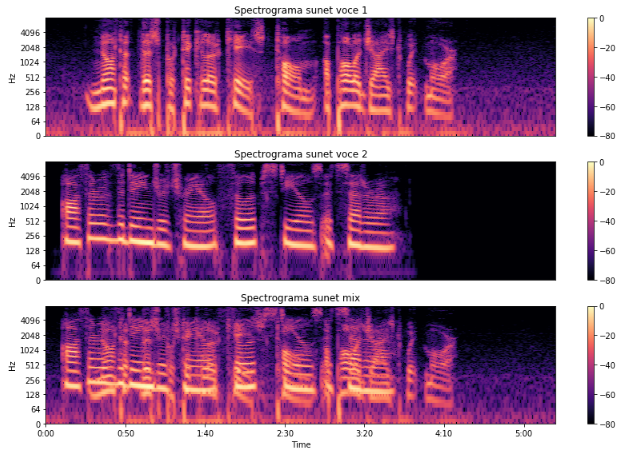


Figura 17 : Spectrogramele calculate pentru cele două surse de semnal și mixul format din ele

### Strutura rețelei neuronale

Rețeaua neuronală are trei straturi de tip „fully connected” ( dens conectate), iar după fiecare strat dens conectat se face o normalizare a datelor.

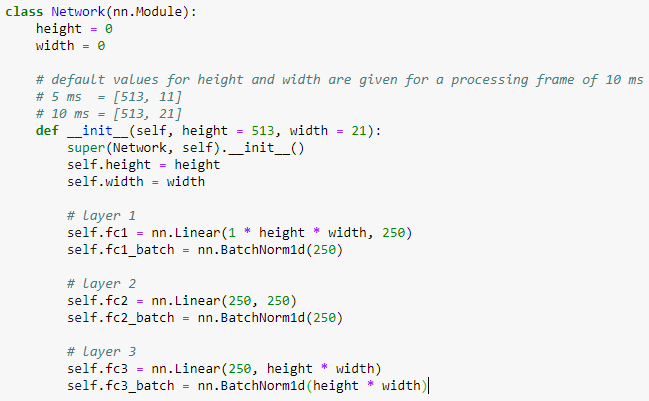


Figura 18 : Structura rețelei neuronale pe straturi

Valorile standard pentru numărul de linii și coloane sunt 513, respectiv 21, pentru o fereastra de procesare curentă de 10 milisecunde și o fereastră de analiză de 4 ori mai mare decât fereastra de procesare.

#### Straturile de tip fully connected (dens conectate)

O rețea neuronală profundă dens conectată constă într-o serie de straturi dens conectate. Aceste straturi dens conectate sunt reprezentate printr-o funcție de la ℝ (m) la ℝ (n). Fiecare dimensiune de ieșire depinde de dimensiunea intrării. Un strat dens conectat este reprezentat astfe:

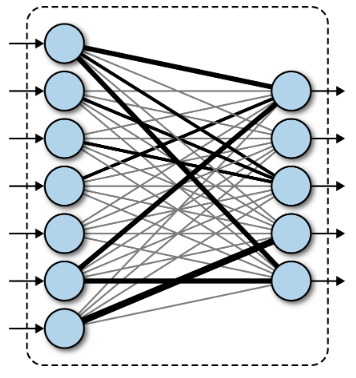


Figura 19 : Structrura unui strat dens conectat

Fie x ∈ ℝ cel care reprezintă intrarea pentru un strat dens conectat, iar y ∈ ℝ să fie a i-a ieșire din stratul dens conectat. Atunci yi ∈ ℝ este calculat astfel :

Aici, σ este o funcție neliniară ( sigmoida, care va fi prezentată mai târziu), iar wi sunt parametrii care învață din rețea. Ieșirea y este calculată astfel :

Este posibil să se stivuiască rețele dens conectate. O rețea cu multiple rețele dens conectate este de obicei numită „rețea profundă” și arată ca în Figura 20.

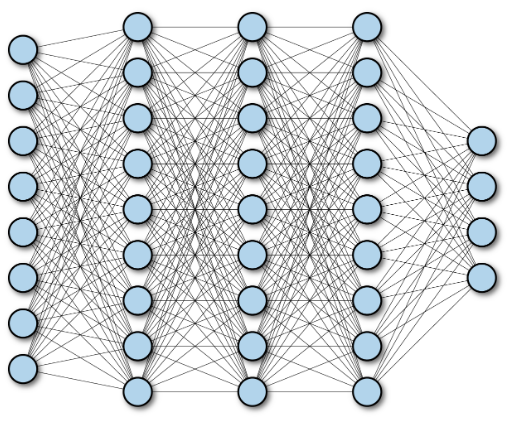


Figura 20 : Rețea neuronală profundă cu rețele dens conectate

Nodurile din rețelele dens conectate sunt numite de obicei neuroni. În consecință, oriunde în literatura de specialitate, rețelele dens conectate vor fi referite des ca fiind rețele neuronale.

În 1940, Warren S. McCulloch și Walter Pitts au publicat primul model matematic al creierului care argumenta că neuronii sunt capabili de calculul unor funcții arbitrare pe cantități boolene. Succesorii acestei munci au adus modificări mici asupra acestui model logic, făcându-i neuroni matematici, cu funcții continue care variau între 0 și 1. Dacă intrările acestor funcții cresc destul de mult, neuronul primește valoarea 1 și este marcat ca fiind „fired”, altfel este în repaus. Cu adăugarea ponderilor ajustabile, descrierea se potrivește cu ecuația precedentă.

În acest tip de strat, fiecare neuron din stratul curent este conectat cu fiecare neuron din stratul anterior și fiecare conexiune are propria pondere.

Aceste straturi dens conectate sunt cele care fac învățarea discriminantă într-o rețea neuronală profundă.

#### Batch normalization

Normalizarea este o metodă foarte cunoscută în antrenarea rețelelor neuronale profunde. Aceasta a fost introdusă de Sergey Ioffe și Christian Szegedy de la Google. Această normalizare se referă la normalizarea valorilor de activare ale unităților ascunse, astfel încât distrubuția acestor valori de activare să rămână aceleași pe timpul antrenării.

În timpul antrenării al oricărei rețele neuronale profunde, dacă actvarea ascunsă a distribuției se schimbă din cauza schimbărilor ponderilor și biasului în acel strat, ele pot cauză foarte repede schimbări în stratul următor, iar acest lucru încetinește foarte mult antrenarea.

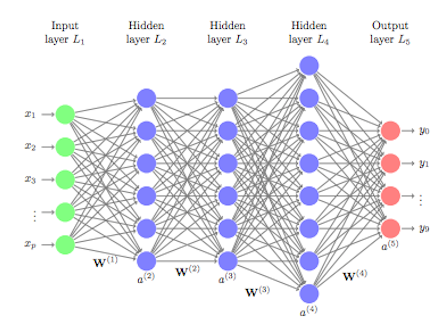


Figura 21 : Structura unei rețele neuronale profunde cu 3 straturi ascunse

În figura 21 avem o rețea neuronală profundă cu 3 straturi ascunse, pe lângă care un strat de intrare și încă unul de ieșire. Fiecare strat ascuns are propriile matrici de ponderi și vectori de bias, așa cum sunt prezentate și în figura 21. Intrarea la fiecare nivel trece prin transformări folosind matricea de ponderi și vectorul de bias. De exemplu, ieșirea din stratul ascuns L2 se comportă ca intrare pentru stratul ascuns L3. Valorile de activare ascunse ale stratului 2 (L2) sunt transformate înmulțind matricea de ponderi a stratului de 3 și adăungând valorile din vectorul de bias. Această ieșire este trecută mai departe printr-o funcție de activare precum sigmoida, reLU sau tanh, iar rezultatul pentru stratul ascuns 3 (L3) este obținut.

Acest proces se repetă la fiecare strat ascuns. Așa cum am observat, valorile de activare ale stratului 3 sunt de fapt afectate, influențate de valorile de activare ale stratului 2. Dacă distribuția valorilor de activare ale stratului 2 se schimbă rapid, atunci acest lucru va afecta eficiența în antrenarea rețelei neuronale profunde.

Mai departe se va descrie cum se realizează această normalizare asupra unui set de date. Normalizarea oricărei date se referă la aflarea mediei și varianței datelor și normalizarea lor, astfel încât datele să aibă 0 media și unitatea de varianță.

Considerăm că avem „d” unități ascunse într-un strat ascuns în rețeaua neuronală ascunsă. Se pot reprezenta valorile de activare ale acestui strat ca fiind . O unitate din aceast vector de valori de activare se poate normaliza folosind următoarea formulă :

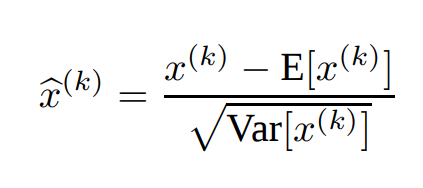


Figura 22 : Formula de normalizare a unei unități din stratul ascuns „d”

este media valorilor,iar este varianța unității k din stratul ascuns.

### Workflow-ul rețelei neuronale

### Antrenarea rețelei

### Testarea rețelei

## Rezultate și performanțele rețelei

# Concluzii

# Index figuri

[Figura 1 : Pașii aplicării transformatei Fourier pe un semnal 8](#_Toc9794500)

[Figura 2 : Fereastră care se plimbă peste un semnal 9](#_Toc9794501)

[Figura 3 : Datele unui semnal după aplicarea funcției "half-cosine" 9](#_Toc9794502)

[Figura 4 : Spectrograme folosite în aplicație pentru două voci și mixul lor 11](#_Toc9794503)

[Figura 5 : Calcularea măștii de frecvență cu respect pentru prima sursă de semnal 12](file:///C:\Users\Kate\Desktop\disertatie\sound-source-separation\Disertatie-doc.docx#_Toc9794504)

[Figura 6 : Calcularea măștii de frecvență cu respect pentru ambele surse de semnal 13](file:///C:\Users\Kate\Desktop\disertatie\sound-source-separation\Disertatie-doc.docx#_Toc9794505)

[Figura 7 : Arbore structural al inteligenței artificiale și a tuturor subdomeniilor sale 16](#_Toc9794506)

[Figura 8 : Structura unei rețele neuronale 17](#_Toc9794507)

[Figura 9 : Funcția perceptronul 18](#_Toc9794508)

[Figura 10 : Formula pentru aflarea valorii de ieșire 19](#_Toc9794509)

[Figura 11 : Formula de calcul al erorii 20](#_Toc9794510)

[Figura 12 : Formula de calcul pentru propagarea înapoi 20](#_Toc9794511)

[Figura 13 : Formulele de calcul pentru actualizarea ponderilor și a biasului 20](#_Toc9794512)

[Figura 14 : Procesul de suprapunere a ferestrelor 24](#_Toc9794513)

[Figura 15 : Modul de calcul al măștii de frecvență pentru fereastra de procesare curentă 24](#_Toc9794514)

[Figura 16 : Încărcarea semnalelor și mixarea lor 25](#_Toc9794515)

[Figura 17 : Spectrogramele calculate pentru cele două surse de semnal și mixul format din ele 25](#_Toc9794516)

[Figura 18 : Structura rețelei neuronale pe straturi 26](#_Toc9794517)

[Figura 19 : Structrura unui strat dens conectat 27](#_Toc9794518)

[Figura 20 : Rețea neuronală profundă cu rețele dens conectate 28](#_Toc9794519)

# Bibliografie și webografie

* „Blind Audio Source Separation Pipeline and Algorithm Evaluation” - Wisam Reid, Kai-Chieh Huang & Doron Roberts-Kedes

(<http://cs229.stanford.edu/proj2015/124_report.pdf>)

* <https://musicinformationretrieval.com/stft.html>
* https://en.wikipedia.org
* [https://librosa.github.io/librosa](https://librosa.github.io/librosa/)
* https://docs.scipy.org/doc/numpy
* “Time frequency masking”, Machine Perception of Music & Audio - Zafar Rafii, 2014 (<http://users.cs.northwestern.edu/~pardo/courses/eecs352/lectures/MPM14-Time-Frequency-Masking.pdf> )
* <https://towardsdatascience.com/understanding-neural-networks-what-how-and-why-18ec703ebd31>

1. [https://ro.wikipedia.org/wiki/Spectrogram%C4%83](https://ro.wikipedia.org/wiki/Spectrogramă) [↑](#footnote-ref-1)
2. Setul de date poate fi găsit la : <http://www.festvox.org/cmu_arctic/> [↑](#footnote-ref-2)