UNIVERSITATEA DIN BUCUREȘTI

FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

SPECIALIZAREA INFORMATICĂ

Lucrare de licență

METODE DE SEPARARE A VOCILOR SUPRAPUSE

Absolvent
Petre-Șoldan Adela

Coordonator științific
Conf. Dr. Rusu Cristian

București, iunie 2025

Rezumat

Lucrarea îşi propune analiza mai multor metode de a rezolva The Cocktail Party Problem într-un context simplificat, cu două persoane care vorbesc simultan pe înregistrări monofonice şi stereofonice, fără zgomot. Scopul este de a capta discursurile separate. Problema separării surselor este relevantă şi în alte situaţii, precum detectarea diverselor afecţiuni medicale din ecografii. Metodele din proiect sunt variate, unele lucrând direct pe domeniul timp iar altele pe spectrogramă. Unele sunt metode clasice (ICA, NMF, DUET), iar altele de învăţare automată (CNN, Wave-U-Net, Wave-U-Net cu Transformer simplu în stratul de maximă adâncime).

Evaluarea s-a realizat comparând rezultatele separării cu semnalele corecte, prin metricile SDR, SAR şi SIR. Metodele clasice nu au necesitat antrenare, dar au obţinut valori mai slabe, confirmând faptul că acestea au nevoie de implementare riguroasă şi eventual îmbinare cu învăţarea automată. Metodele moderne nu funcţionează bine fără o cantitate suficientă de date de antrenare.

Abstract

The thesis makes an analysis of different methods in order to solve The Cocktail Party Problem in a simplified context, with two people talking simultaneously on monophonic and stereophonic recordings. The purpose is to get the individual speeches. The source separation problem is relevant in other contexts as well, such as detecting diseases based on echographies. The methods included in the project vary in terms of approaches, some using time domain and the others using the spectrogram. Some of them are classical methods (ICA, NMF, DUET), and the others are machine learning techniques (CNN, Wave-U-Net, Wave-U-Net with a simple Transformer incorporated in the bottleneck).

The evaluation was made by comparing the separation results with the right answer, using the metrics SDR, SAR, SIR. Classical methods don't need to be trained, but they provide worse results, proving that they need an elaborated implementation and possibly to be combined with machine learning. Modern methods do not work very well without enough train data.

Cuprins

Capitolul 1

Introducere

1.1 Prezentarea generală şi motivaţia alegerii temei

Tema lucrării combină elemente de separare a surselor cu procesarea semnalelor audio. Separarea surselor este o tehnică de procesare a semnalelor, alături de aplicarea diverselor filtre, eliminarea zgomotului (care poate fi văzut tot ca o sursă) şi compresia.

Lucrarea presupune că nu avem în prealabil informaţii despre sursele pe care trebuie să le separăm. Singura informaţie o reprezintă semnalele amestecate, lucru întâlnit frecvent. Acest proces se numeşte separare oarbă a surselor (Blind Source Separation), o tehnică ce încă este de actualitate, fiind menționată frecvent în publicații care folosesc învățare automată[1].

Motivaţia se bazează pe faptul că procesarea semnalelor audio are numeroase aplicaţii. Îmbinată adeseori cu inteligenţa artificială, este utilizată în identificarea vocilor pentru autentificarea unui utilizator, crearea aparatelor auditive pentru persoane cu deficienţe de auz, crearea muzicii electronice şi a sunetelor realiste pentru jocuri şi filme. Este folosită şi pentru a monitoriza buna funcţionare a maşinilor, bazându-se pe sunetele emise de o maşină care are o anumită defecţiune.

Separarea surselor este utilă pentru a separa instrumentele muzicale şi vocile dintr-o piesă. Cazuri de separare care nu implică semnale audio includ determinarea formulei chimice a unei substanţe compuse, separarea umbrelor în imagini (vedere artificială) şi separarea undelor care sunt captate simultan în telecomunicaţii.

Scopul lucrării este de a prezenta mai multe metode de separare audio, explicând principiile din spatele acestora, limitările fiecăreia şi contextul în care funcţionează cel mai bine.

Varianta fără zgomot a fost aleasă pentru a studia strict dificultăţile legate de a diferenţia două voci.

1.2 Tehnologii folosite

Limbajul de programare folosit este Python, iar bibliotecile din Python folosite sunt:

- Numpy[3]: de exemplu pentru descompunerea in valori si vectori proprii cu functia numpy.linalg.eigh
- Matplotlib[4]: pentru plotarea graficelor şi rezultatelor
- Librosa[8]: pentru tranziţia din domeniul timp în domeniul timp-frecvenţă şi invers (librosa.stft si librosa.istft),
- PyTorch[10]: pentru straturile modelelor de învăţare automată
- Soundfile: pentru încărcarea şi salvarea semnalelor audio,
- mir_eval: cu functia mir_eval.separation.bss_eval_sources pentru calcularea metricilor SDR, SIR, SAR. Aceeaşi funcţie obţine şi permutarea corectă şi corectează decalările care apar de la întârzierile din procesul de mixare.
- time pentru măsurarea timpilor de testare şi antrenare

.

1.3 Contribuţii

Codul proiectului este disponibil pe GitHub.[1] Am implementat de la zero algoritmii clasici Fast Independent Component Analysis, Non-negative Matrix Factorization şi Degenerate Unmixing Estimation Technique.

Pentru implementarea şi antrenarea metodelor de învăţare automată am folosit, în plus, framework-ul PyTorch pentru a construi structura reţelelor neuronale. Aceste reţele sunt un CNN pe spectrogramă, un Wave-U-Net simplu si un Wave-U-Net cu self-attention. Pentru DUET am implementat un K-MEANS simplu.

În cod, înainte de aplicarea oricărui algoritm, am normalizat canalele la medie nulă scăzând din fiecare valoare media. Ne trebuie şi deviaţie standard 1, pentru ca algoritmii să dea aceeaşi importanţă ambelor canale. Pentru ICA aplicăm doar prima normalizare, de cealaltă ocupându-se algoritmul în sine.

Setul de date folosit este LibriMix[2], care generează doar canale mono. De aceea, pentru varianta stereo am făcut un program care generează 2 canale pe baza a două surse, asemănător modului de generare din LibriMix.

[1] Implementarea proiectului este disponibila la https://github.com/adelp13/Proiect-Licenta

[2] LibriMix este un set de date open-source pentru separarea surselor vocale, disponibil la https://github.com/JorisCos/LibriMix

1.4 Structura lucrării

Aici se află o scurtă descriere despre conţinutul câtorva secţiuni, împărţite pe capitole:

- Preliminarii: conţine informaţii teoretice generale
- Metode clasice:
  - Fast ICA: model pe semnal brut, presupune independenţa statistică şi non-gaussianitatea surselor. Foloseşte whitening cu descompunerea matricei de covarianţă în valori şi vectori proprii. Aproximează inversul matricei de amestecare prin determinarea direcţiilor perpendiculare care favorizează negentropia.
  - NMF: lucrează doar cu date pozitive, fiind nevoie de trecere pe spectrogramă. Pentru amplitudinea spectrogramei aproximează o descompunere în două matrici, una raportând sursele la timp, iar cealaltă sursele la frecvenţe. Pentru cele două spectrograme separate, calculează o mască prin raportul amplitudinilor separate şi amplitudinilor amestecului.
  - DUET: se bazează pe faptul că o sursă ajunge mai rapid la un microfon decât la altul (deci nu merge pe mono). Lucrează pe spectrogramă pentru a calcula întârzierea fazelor în secunde şi un raport de amplitudini, presupunând că dacă un semnal ajunge mai târziu are amplitudine mai mică. Foloseşte K-MEANS clustering pentru a împărţi punctele cu cele două dimensiuni în 2 grupuri, stabilind din ce sursă face parte fiecare.
- Metode de învăţare automată
  - CNN: model pe spectrogramă
  - Wave-U-Net: model pe semnal brut cu 5 nivele de adâncime pentru canale şi dimensiunea timp. Are strat convoluțional în bottleneck cu ieşire pe 512 canale. Foloseşte Batch Normalization şi Leaky ReLU în encoder, iar în decoder Instance Normalization şi PReLU. Foloseşte la ieşire activarea tanh pentru ca semnalele sunt între (-1, 1).
  - Transformer: stratul bottleneck din Wave-U-Net conţine componente de self-attention pentru aprofundarea relaţiilor globale. Conţine codificare poziţională, urmată de 2 straturi de self-attention şi feed forward cu două skip connections prin adunare.
- Evaluarea calităţii separării:
  - Antrenare şi testare: informaţii despre numărul de batch-uri şi de exemple

Capitolul 2

Preliminarii

Separarea surselor se realizează mai uşor atunci când avem cel puţin la fel de multe microfoane precum surse. În caz contrar, sistemul este subdeterminat, având mai multe necunoscute decât ecuaţii şi fiind nevoie de unele presupuneri sau aproximări pentru a obţine un rezultat.

Înregistrările din setul de date au rata de eşantionare de 16 kHz. Această rată este folosită foarte des în procesarea semnalelor vocale. Conform studiului publicat în PubMed Central[12], majoritatea frecvenţelor din vorbire sunt sub 7kHz.

Teorema Nyquist[11] spune că, pentru a reda semnale de frecvenţă f, avem nevoie de o frecvenţă de eşantionare $2f$. Acest lucru se poate înţelege prin faptul că, la bază, orice semnal este compus din unde sinusoidale. O astfel de undă de frecvenţă 1Hz are două întoarceri (puncte de maxim şi minim) în perioada sa într-o secundă, deci minim două eşantioane pentru a o reţine.

Pentru frecvenţele mai mari de f, va avea loc fenomenul de aliasing[9], acestea apărând ca frecvenţe mai mici. Luând o marjă de 1 kHz, reprezentăm frecvenţe de până la 8kHz cu o frecvenţă de eşantionare de 16kHz. Mai multe eşantioane ar fi crescut costul computaţional fără să fie absolut necesar.

Matricea de covarianţă a unei matrici de dimensiuni ($numar\_semnale$, $esantioane$) este simetrică faţă de diagonala principală. În afara diagonalei, pe poziţia (i, j), se află valorile de covarianţă dintre semnalele de pe liniile i şi j.

Formula este $Cov(i,j) = \frac{1}{N}\sum_{k=1}^{N}(\mathbf{s}_{ik} - \bar{\mathbf{s}_\mathbf{i}})(\mathbf{s}_{jk} - \bar{\mathbf{s}_\mathbf{j}})$, unde N este numărul de eșantioane. Matricea este și pozitiv semidefinită, iar varianta matriceală este $Cov(M) = \frac{1}{N}MM^\top$ (doar pentru medie 0).

Covarianţa determină dacă semnalele au valori care se mişcă liniar în acelaşi sens de-a lungul eşantionării (covarianţă pozitivă) sau dacă atunci când unul creşte celălalt scade (covarianţă negativă).

Covarianţa nulă înseamnă că nu există nicio astfel de legătură liniară, adică sunt independente liniar. Legături în creşterea sau scăderea valorilor pot exista în continuare, acestea fiind neliniare (de exemplu $s_i^n = s_j$, unde $n \neq 1$).

Capitolul 3

Metode clasice

3.1 Fast Independent Component Analysis

Există mai multe variante ale algoritmului ICA[5], cea mai des întâlnită fiind Fast ICA, cea pe care am implementat-o. Limitarea algoritmului este că nu poate face separarea pe un singur canal. De asemenea, ICA presupune că sursele iniţiale sunt statistic independente, adică ştiind ceva despre una dintre ele nu putem afla nimic despre cealaltă. ICA presupune şi că sursele au distribuţii non-gaussiene. Se potriveşte pentru voci, acestea neavând distribuţie gaussiană.

Teorema Centrală Limită[1] spune că amestecul va tinde spre o distribuţie gaussiană. ICA se bazează pe maximizarea non-gaussianităţii pentru a se apropia de sursele iniţiale, deci nu ar funcţiona dacă acestea sunt tot gaussiene.

Datele de intrare sunt cele două canale într-o matrice M. Scopul algoritmului este de a calcula o aproximare a inversei matricei de mixare, în cazul nostru ambele având 2 linii şi 2 coloane. Codul este generalizat pentru a aproxima n surse din n microfoane.

Primul pas al algoritmului se numeşte whitening. Scopul acestuia este de a elimina corelaţiile liniare dintre două canale, matricea de covarianţă devenind matricea unitate (pentru stabilitate toate variantele sunt 1). Mai întâi calculăm matricea de covarianţă pentru a vedea modul de variaţie a surselor între ele, după care o descompunem în produs de valori şi vectori proprii: $Cov(M) = ABA^\top$. O matrice simetrică are vectori proprii perpendiculari între ei şi doar valori proprii reale. Fiind şi pozitiv semidefinită, valorile proprii sunt pozitive din definiţie.

A = matricea vectorilor proprii, reprezintă direcţiile principale perpendiculare de variaţie.

B = matricea diagonală a valorilor proprii, reprezintă modulul direcţiilor.

Matricea de whitening W se calculează ca $W(M) = AB^{-\frac{1}{2}}A^\top$ și se aplică lui M astfel: $M_{nou} = WM$ (de la dreapta la stânga, rotim în sistemul de direcţii pentru a ajunge la

[1]Informații despre Teorema Centrală Limită la: https://en.wikipedia.org/wiki/Central_limit_theorem

valori proprii, normalizăm valorile proprii şi rotim înapoi).

Ca verificare, calculăm noua valoare a matricei de covarianţă, ţinând cont că $A^\top A = I$:

$\text{Cov}(\text{M}_{nou}) = \frac{1}{N}(WM)(WM)^\top = W(\frac{1}{N}MM^\top)W^\top = AB^{-\frac{1}{2}}A^\top ABA^\top A(B^{-\frac{1}{2}})^\top A^\top = AB^{-\frac{1}{2}}BB^{-\frac{1}{2}}A^\top = A^\top A = I.$

După whitening, mai trebuie să eliminăm corelaţiile neliniare. Fie S matricea de separare pe care o căutăm. O iniţializăm cu valori oarecare. Aceasta va reţine pe fiecare linie direcţia în care este proiectat canalul corespunzător. Proiecţia se calculează ca $pr = dir^\top M$, unde dir este direcţia. Luăm pe rând fiecare direcţie şi o modificăm iterativ până nu se mai schimbă semnificativ. Trebuie să micşorăm gaussianitatea proiecţiei, adică să maximizăm negentropia, care este o măsură a non-gaussianităţii.

Negentropia se poate scrie ca $Neg(pr) = entropie(prg) - entropie(pr)$, unde prg este o distribuție gaussiană cu aceeași varianță și medie ca pr. Negentropia este aproximată pentru a fi calculată mai ușor: $Neg(pr) \approx \mathbb{E}[G(pr)] - \mathbb{E}[G(prg)]$, unde o variantă a funcțiilor este $G(pr) = \log\cosh(pr)$ și $G'(pr) = g(pr) = \tanh(pr)$.

Pentru actualizarea direcţiei nu putem face urcare pe gradient simplă, avem nevoie de proiecţie pe sferă pentru că avem direcţii de normă 1. Folosim $dir_{\text{noua}} = \mathbb{E}[M\, g(pr))] - \mathbb{E}[g'(pr)]\, dir$. Primul termen e derivata față de dir a negentropiei aproximate. Intuitiv, este o medie ponderată pentru fiecare canal, fiecare eșantion primind o pondere în funcție de cât a contribuit la non-gaussianitate.

Al doilea termen este $\lambda \cdot dir$, λ venind dintr-o aproximare a multiplicatorului din Lagrangianul pentru constrângerea de normă 1: $L(dir) = \mathbb{E}[G(pr)] - \lambda\,(dir^T dir - 1)$.

Noi căutăm un optim, un punct critic, deci derivăm să obţinem gradientul şi egalăm cu 0: $L' = \mathbb{E}[Mg(pr)] - 2\lambda \cdot dir = 0$. De aici reiese $\lambda = \frac{1}{2} dir^T \mathbb{E}[M \cdot g(pr)]$, aproximat la $\frac{1}{2}\mathbb{E}[g'(pr)]$ pentru simplitate computațională. Din L'= 0, luăm $dir = \frac{1}{2\lambda}\,\mathbb{E}[M \cdot g(pr)]$.

După actualizare, trebuie să ne asigurăm că direcţia e perpendiculară pe direcţiile calculate deja, ele fiind deja perpendiculare între ele pentru o separare mai bună. Pentru a face vectorul direcţie a perpendicular cu direcţia b, aplicăm metoda Gram-Schmidt scăzând din a proiecţia lui pe b, proiecţia reprezentând partea lui a care e paralelă cu b.

Formula proiecţiei lui a pe b este: $pr_b a = \frac{a \cdot b}{\|b\|^2} \cdot b$. Raportul $\frac{b}{\|b\|^2}$ aplică lui b norma 1, dar în cazul nostru toate direcțiile sunt normalizate cu norma 1, deci formula proiecției devine $pr_b a = a \cdot b \cdot b$, adică produsul scalar $a \cdot b$ (care măsoară alinierea celor două direcţii) proiectat pe direcţia b. Pentru a fi independente, produsul scalar trebuie să fie nul, asta ducând şi la proiecție nulă.

Am ales nişte semnale sintetice simplificate pentru a se vedea clar etapele de separare, cu matricea de amestecare $\begin{bmatrix} 0.8 & 0.7 \\ 0.7 & 0.8 \end{bmatrix}$. Se observă în Figura 3.1 că amestecurile nu depăşeau intervalul (-1, 1), iar după whitening depășesc. ICA nu prea ţine cont de scală. De aceea, la final am normalizat împărţind la valoarea maximă. Înregistrările .wav au valori în acelaşi interval.

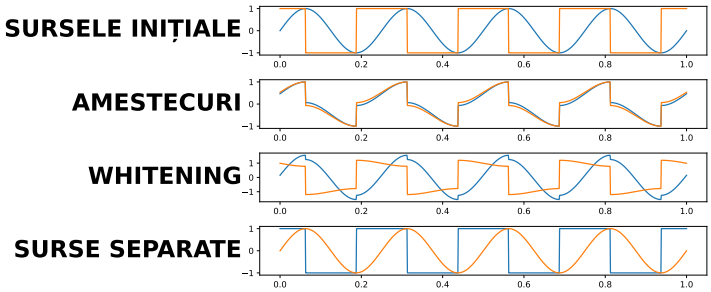

Figura 3.1: Pașii principali ai algoritmului

3.2 Non-negative Matrix Factorization

În general, NMF[6] e aplicat pe canal mono, dar există şi versiuni stereo. Implementarea din lucrare este pentru mono şi generalizează pentru n surse. Lucrează pe spectrogramă, care are doar valori pozitive, potrivit pentru algoritmul NMF.

Pentru trecerea în domeniul timp-frecvenţă, aplicăm Transformata Fourier pe câte 1024 eşantioane şi avem suprapunere de 512, adică 50%. Se poate folosi şi 256. Cu cât aplicăm Fourier pe mai multe eşantioane deodată, cu atât avem mai multă informaţie frecvenţială, dar mai puţină informaţie temporală. Un pas mai mic înseamnă mai multă informaţie temporală. Valorile de pe spectrogramă au două componente, amplitudinea unei frecvenţe la un anumit moment de timp şi faza corespunzătoare, adică decalajul faţă de începutul semnalului.

Notăm cu t numărul de eşantioane, cu f frecvenţele, cu s numărul de surse pe care le căutăm şi cu A matricea de amplitudini din spectrogramă. Algoritmul NMF caută două matrici M şi N de dimensiuni $(f,\ s)$ şi $(s,\ t)$ şi cu valori pozitive astfel încât $A \approx MN$. M reţine pe coloane pentru fiecare sursă câtă energie are din fiecare frecvenţă. N reţine când e fiecare sursă prezentă şi în ce măsură. Practic descompunem cele 2 informaţii date de matricea de amplitudini în două matrici.

Le iniţializăm cu valori oarecare. La fiecare iteraţie actualizăm valorile M si N prin metoda Lee and Seung: $N = N \odot \frac{M^\top A}{M^\top MN+\epsilon}$ si $M = M \odot \frac{AN^\top}{MNN^\top+\epsilon}$, unde ϵ este o valoare foarte mică pentru a evita împărțirea la 0. NMF minimizează o funcție care măsoară cât de bine aproximăm A: raportul $\frac{M^\top A}{M^\top MN+\epsilon}$ e aproape de 1 pentru aproximare bună. Dacă e subunitar, se reduc valorile din N. Înmulţirile şi împărţirile păstrează valorile pozitive.

Separarea s-a efectuat pe amplitudini. Pentru faze nu este la fel de uşor, având şi structura periodică. Parcurgem fiecare coloană din M cu linia corespunzătoare din N şi creăm prin produs extern o matrice de amplitudini $A'(f,\ t)$ doar pentru sursa respectivă. Acum trebuie să adăugăm la amplitudini şi faza. O metodă ar fi să folosim faza spectrogramei amestecului, dar rezultatele nu sunt foarte bune.

O altă opţiune este generarea unei măşti pe care să o aplicăm direct spectrogramei amestecului: $masca = \frac{A'}{MN+\epsilon}$. Astfel, normalizăm valorile lui A' faţă de matricea de

amplitudini a mixului pentru a afla unde în spectrogramă este prezentă sursa mai mult. La final, revenim în domeniul timp cu inversa Transformatei Fourier aplicată pe fiecare bucată şi normalizăm să nu existe valori în afara intervalului (-1, 1), la fel ca la ICA.

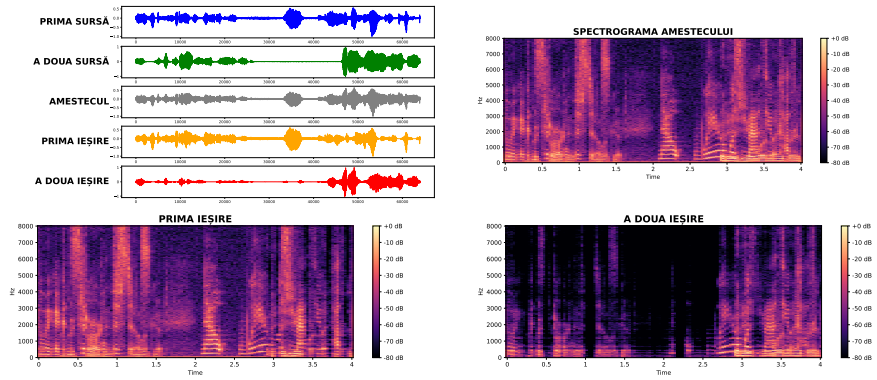

Figura 3.2: Exemplu din setul de date monofonic

În Figura 3.2, graficele spectrogramelor sunt pe axa y logaritmică pentru a se vedea mai clar diferenţele dintre frecvenţele joase. Se observă că algoritmul a reușit să separe atunci când prima sursă era foarte puțin activă (pe la eșantionul 3500) și nu a atribuit frecvențe ambelor rezultate.

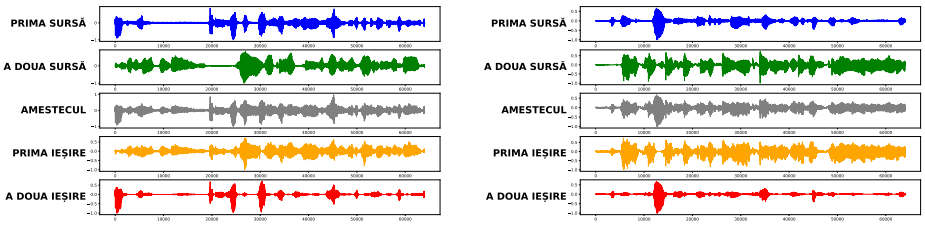

Figura 3.3: Alte două exemple

Se observă în Figura 3.3 că, în ambele cazuri, prima sursă separată este de fapt a doua. Algoritmii de separare returnează o permutare a surselor aproximate, identificarea surselor făcându-se ulterior.

3.3 Degenerative Unmixing Estimation Technique

DUET[17] lucrează tot pe spectrogramă. Se bazează în principal pe faptul că o sursă ajunge mai rapid la un microfon decât la celălalt. De aceea, nu funcţionează pe canal mono.

Calculează spectrogramele $S1$ şi $S2$ ale celor două canale şi apoi creează o hartă care reprezintă $S2$ raportat la $S1$. Pentru diferenţa de amplitudine, folosim $a = \left|\frac{S2}{S1}\right|$, adică valorile spectrogramei 2 raportate la prima, după care luăm doar amplitudinea.

Pentru compararea fazelor, abordarea este similară, doar că mai adăugăm o normalizare pentru a o exprima în secunde: $b = -\frac{(\angle \frac{S2}{S1})}{2\pi F}$, unde F sunt frecvențele din domeniul

frecvență (fiecare linie i din F conținând doar frecvența $f_i = f_s \frac{i}{N}, i = 0, 1, \ldots, \frac{N}{2}$), iar înmultite cu 2π sunt exprimate în radiani/secundă (viteza unghiulară, cât de repede e parcursă perioada unui semnal de frecvenţă f). ($\angle \frac{S2}{S1}$) este diferența de fază, iar prin acea împărțire obținem diferența în secunde b.

Minusul de la b vine din formula $diferenta_{faza} = -d * \omega$, unde d este întârzierea. Faza în radiani la un moment t este $faza(t) = faza(0) + \omega t$, similar cu formula distanței în metri. Pentru o întârziere d, avem $faza(t-d) = faza(0) + \omega t - d * \omega = faza(t) - d * \omega$, deci diferența de faze este $faza(t-d) - faza(t) = -d * \omega$. Am presupus că S_2 a întârziat față de S_1, dar matricea de întârzieri poate avea şi valori pozitive şi valori negative.

Se poate lucra direct cu diferenţa de faze, dar nu e la fel de eficient pentru DUET.

Pe hartă, dacă multe frecvenţe au aceeaşi pereche, se vor grupa în acelaşi loc, fiind semnale care ajung cu aceeaşi întârziere la cele 2 microfoane. Acolo poate fi o sursă. Vom ţine cont doar de frecvenţele cu amplitudine suficient de mare (>0.02).

Ne trebuie două măşti binare complementare pentru a le aplica pe cele două spectrograme iniţiale, similar ca la NMF. Pentru asta vom găsi două puncte de maxim de pe spectrogramă, câte unul pentru fiecare sursă separată. Dacă erau 3, am fi căutat 3. Fiecare punct de pe hartă va fi atribuit vârfului de care e mai apropiat.

Pentru a împărţi punctele în cele două grupuri, am implementat algoritmul clasic de clustering K-MEANS[7], generalizat pentru n dimensiuni. Acesta returnează coordonatele i şi j ale celor doi centroizi şi etichetele tuturor punctelor. Dacă o etichetă este 0, aparţine centroidului 0.

Inputul este o matrice $(n,\ 2)$, fiind numărul total de puncte de pe hartă, cu excepţia frecvenţelor prea mici. Iniţializăm aleator poziţiile centroizilor, iar apoi calculăm distanţa Euclidiană ($dist(a, b) = \|a - b\|2 = \sqrt{\sum_{i=1}^{\dim}(a_i - b_i)^2}$) a fiecărui punct faţă de centroizi. Se atribuie etichetele, după care fiecare centroid se mută în mijlocul punctelor din grupul său, calculând media fiecărei coordonate. Procesul se repetă până când $\|centroizi_{noi} - centroizi\|_2 < \varepsilon$.

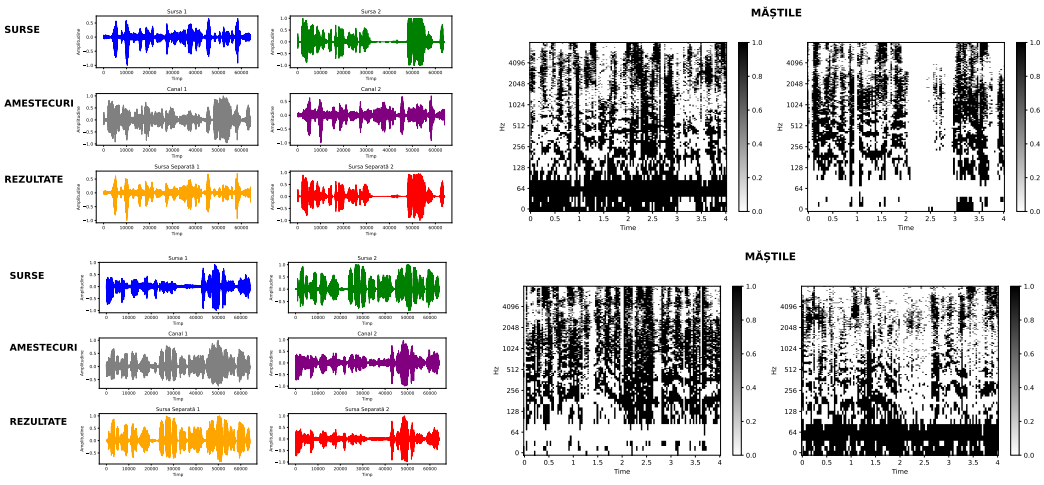

Figura 3.4: Exemple din setul de date stereofonic

Capitolul 4

Metode bazate pe învăţare automată

4.1 Reţea Neuronală Convoluţională Simplă (CNN)

Reţeaua poate primi unul sau două canale de dimensiuni(f, t), pentru cazurile mono şi stereo. Un canal de intrare reprezintă partea reală a spectrogramei din înregistrare. În ambele cazuri tensorul de ieşire va avea două canale, returnând câte o mască soft (cu valori între 0 şi 1) pentru fiecare sursă aproximată. Spectrograma permite analiza mai detaliată a detaliilor din înregistrare, cum ar fi tonalitatea, timbrul, viteza de vorbire. CNN[2] este o structură relativ simplă şi de aceea am ales să nu lucrez pe semnal brut.

Elementul principal este stratul convoluţional, care scanează cu o fereastră glisantă dimensiunile fiecărui canal pentru a găsi tipare din anumite regiuni. Cele mai des folosite filtre sunt cele pătratice, în special cel cu dimensiunile (3, 3), care nu este nici prea mare, nici prea mic şi are simetrie. Dacă aplicăm un astfel de filtru unei matrici (f, t), dimensiunile acesteia se vor micşora cu 2 unităţi de fiecare dată, deoarece fereastra se poate muta pe linie de $t-2$ ori şi pe coloană de $f-2$ ori. Pentru a nu pierde informaţie la fiecare convoluţie, bordăm matricea pentru a creşte lungimea şi înălţimea cu 2.

Reţeaua are două părţi principale, un encoder şi un decoder, amândouă având, printre alte straturi, 3 straturi convoluţionale. După primul strat convolutional, numărul de canale ajunge la 32, după care se dublează pentru fiecare strat convoluțional din encoder. Cu cât sunt mai multe canale, cu atât se aplică mai multe convoluţii simultan, fiecare pe o perspectivă diferită a datelor. Fiecare convoluţie se poate specializa pe a capta o anumită trăsătură. Deci cu fiecare strat convoluțional, modelul descoperă detalii şi reprezentări din ce în ce mai abstracte pentru a găsi corelaţii noi. Straturile convoluţionale din decoder au rolul de a genera un răspuns pe baza informaţiilor descoperite în decoder. Fiecare strat convoluțional din decoder înjumătăţeşte numărul de canale pentru a se întoarce la reprezentări mai simple şi a trage nişte concluzii.

După fiecare strat de convoluţie (cu excepţia ultimului) se aplică un strat de normalizare, urmat de un strat de activare. Stratul Batch Normalization ţine sub control

distribuţia datelor prin aplicarea mediei 0 şi deviaţiei standard 1. Pentru CNN, numărul de loturi (cazuri procesate simultan) este 8. După primul strat de convoluţie, tensorul va avea dimensiunea(8, 32, f, t). La normalizare, fiecare canal dintr-un lot este normalizat împreună cu canalele corespunzătoare din celelalte loturi, deoarece se specializează pe date asemănătoare. Această abordare este specifică stratului de Batch Normalization. Stratul de activare folosește $ReLu(val) = max(0, val)$, rolul unei funcţii de activare fiind de a introduce non-liniaritate. Fără aceasta, reţeaua ar face doar operaţii liniare (similar cu un SVM fără kernel), limitând capacitatea de învăţare. ReLu e eficientă pentru că e rapid de calculat şi generează multe valori 0, similar cu a dezactiva noduri. Normalizarea se face înainte de activare deoarece, normalizând, vom avea şi valori negative, pe care ReLu le elimină. În plus, după convoluţie valorile sunt instabile, ceea ce trebuie rezolvat rapid. Activarea schimbă distribuţia valorilor, pas care trebuie făcut după normalizare. Deoarece normalizarea face media să tindă spre 0, aceasta stabileşte care valori trebuie să fie negative şi care pozitive, urmând să fie filtrate de ReLU.

Reţeaua conţine şi un strat de Dropout cu probabilitate 30%, care dezactivează aleator noduri pentru a accelera procesarea şi a evita supraînvăţarea (un risc mai ales pe seturi mici de date[16], modelul nereuşind să se descurce pe teste şi contexte noi). Dropout e aplicat în encoder, când există 64 de canale, un număr destul de mare să aibă un efect, dar nu prea mare pentru a destabiliza.

În encoder există două straturi de MaxPooling, aplicate între secvenţele de convoluţie → normalizare→activare. Pe măsură ce modelul are mai multe canale şi perspective, nu mai este nevoie de o reprezentare la fel de detaliată a spaţiului, iar costul computaţional creşte rapid. Stratul MaxPooling folosit are kernel (2, 2), înjumătăţind dimensiunile f şi t. Din fiecare fereastră, stratul aplică funcţia max, luând nodul cel mai relevant. De aceea, informaţia importantă nu se pierde, ci e sintetizată. Stratul acesta nu are parametri, deci nu învaţă ceva nou spre deosebire de Batch Normalization, este doar o operaţie precum ReLu. Deoarece în output ne trebuie dimensiunile f şi t iniţiale, în decoder trebuie să refacem spaţiul prin două straturi de Upsample cu scalare 2, care interpolează valorile, având un efect opus cu MaxPooling. Metoda de interpolare este biliniară, care calculează o medie ponderată bazată pe distanţă (între fiecare 2 puncte mai apare unul). Este rapidă pentru că nu are parametri.

Deoarece măştile trebuie să aibă valori între (0, 1), trebuie să normalizăm valorile. După ultimul strat convoluțional aplicăm funcţia sigmoid: $sigmoid(var) = \frac{1}{1+e^{-var}}$. Pentru valori foarte mari, e^{-var} tinde la 0, deci funcţia tinde la 1. Pentru valori mici, va tinde la 0. Pe parcursul antrenării, modelul va învăţa ce valori să genereze pe ultimul strat pentru a le da ca parametri pentru sigmoid.

4.2 Wave-U-Net

Este bazat pe structura reţelelor U-Net, care lucrează pe dimensiuni 2D. Wave-U-Net[13] este adaptat pentru semnale unidimensionale. Este mai performant decât un CNN simplu, putând să lucreze pe semnal brut. Ieşirea reprezintă două canale de aceeaşi dimensiune, cele două surse separate. Nu mai e nevoie de procesare ulterioară pentru a aplica măştile şi a reveni în domeniul timp. Diferenţa principală faţă de CNN este că foloseşte skip connections, adică stratul i din encoder este concatenat cu stratul $n-i+1$ din decoder ($1 \leq i \leq n$). Concatenare înseamnă că stratul următor va conţine toate canalele celor două. Decoderul poate reface mai uşor informaţia primind date de la encoderul aflat la acelaşi nivel. Astfel, decoderul ştie ce informaţii să prioritizeze.

În modelul folosit, encoderul şi decoderul au câte 5 straturi convoluţionale, numărul acestora variind în funcţie de o listă de dimensiuni primită ca parametru. Implicit, dimensiunile sunt (16, 32, 64, 128, 256, 512). Primele 5 valori ordonate crescător reprezintă numărul de canale de ieşire ale straturilor convoluţionale din encodere. Ordonate descrescător reprezintă canalele de ieşire pentru convoluţiile din decoder. Un strat din encoder sau decoder conţine convoluţie urmată de normalizare, activare şi apoi pooling pentru encoder. Pentru decoder se face upsample înainte de convoluţie pentru a avea aceeaşi dimensiune t ca stratul cu care se concatenează. Concatenarea se face între upsample şi stratul convoluțional, deoarece convoluţia are cea mai mare nevoie de informaţii anterioare pentru a reface datele. Modelul e similar cu CNN-ul, doar că are concatenare, mai multe straturi şi realizează pooling/uspample pentru fiecare pentru a ajunge pe straturi mai adânci. Lucrând pe semnal brut, trebuie să abstractizeze mai mult informaţia pentru a învăţa.

Pentru CNN am aplicat dropout în encoder, dar pentru Wave-U-Net am vrut să încerc o altă abordare şi am aplicat straturi de dropout doar după activările din decoder. Encoderul găseşte detalii importante şi ar putea fi mai bine să nu dezactivăm neuroni pentru a nu pierde informaţii importante. Pentru decoder nu trebuie la fel de multă atenţie la detalii, mai ales că primeşte informaţii prin skip connections.

Pentru normalizările din decoder am folosit straturi de Instance Normalization. Spre deosebire de metoda Batch Normalization, aceasta normalizează fiecare canal separat. Astfel, modelul se poate concentra pe fiecare exemplu separat pentru reconstrucţie, fără influenţe din exterior. Altfel, poate exista un risc să apară unde nedorite în rezultat. Pentru encoder poate fi un avantaj să aibă acces la celelalte loturi, pentru a se familiariza mai rapid cu structurile semnalelor.

Pentru activările din encoder folosim Leaky ReLU, iar pentru decoder PReLU. Leaky ReLU are formula $LR(var) = \max(var, c \cdot var)$, unde c este o constantă pozitivă nenulă pentru panta valorilor negative. Este utilă deoarece nu dezactivează neuronii negativi precum ReLu, dar nici nu le permite să contribuie foarte mult. Dacă ar primi mereu

valori 0, nu ar mai reuşi să înveţe. Valoarea folosită este 0.1. Dacă ar fi prea aproape de 1, ne-am apropia prea mult de liniaritate, anulând scopul activărilor. Pentru valori pozitive rezultatul funcţiei este deja valoarea respectivă. Cu un c apropiat de 1, funcţia ar fi $LR(var) \approx \max(var, var)$, adică nu ar avea niciun efect semnificativ. Dacă c este prea mic, riscăm ca neuronii să moară.

ReLu este PReLU cu c nul. PReLU are aceeaşi formulă ca Leaky RelU, doar că parametrul c este învăţabil, nu este doar o operaţie simplă cu o constantă. De aceea, în cod folosim instanţe diferite pentru fiecare strat PReLu. Decoderul trebuie să fie mai creativ decât encoderul pentru a reconstrui informaţia învăţată de acesta. De aceea poate fi mai potrivit un parametru ajustabil, care permite flexibilitate. Encoderul doar sintetizează informaţia.

Ultima dimensiune din listă este pentru stratul de maximă adâncime (bottleneck), aflat între ultimul pooling şi primul upsample. Acolo este un strat convoluțional cu ieşire pe 512 canale, urmat de Batch Normalization şi activare ReLu. Este ca un punct de întoarcere. Fără acesta, prima concatenare ar fi avut loc între canale prea asemănătoare. De asemenea, creează un decalaj între numărul de canale, adică la fiecare nivel de adâncime decoderul se concatenează cu un encoder care are de 2 ori mai puţine canale. Astfel, decoderul este prioritar, dar ţine cont şi de encoder.

După decoder, trebuie să ne întoarcem de la 16 la 2 canale, deci aplicăm încă un strat convoluțional. Dimensiunea ferestrei este 1 pentru că nu vrem să mai captăm informaţii noi, doar să diminuăm numărul de canale.

4.3 Transformer

Acesta nu este un model de sine stătător în proiect, ci o îmbunătăţire la metoda Wave-U-Net prin adăugarea de self-attention[14]. Straturile de self-attention pot fi adăugate aproape oriunde. Modelul implementat conţine în bottleneck, după activare, un strat de codificare pozitională, urmat de mai multe straturi de self-attention şi feed-forward cu normalizări. Acesta este un loc potrivit pentru arhitectura Transformer, deoarece e un model solicitant computaţional, iar în cel mai adânc strat spaţiul timp este cel mai restrâns. Stratul self-attention este util pentru a capta relaţii pe termen lung, deci poate prelucra filtrele învăţate de encoder şi să stabilească relaţii mai strânse între ele. Filtrele din encoder se specializează pe câte o informaţie şi pot fi astfel prea izolate. Modelul transformer oferă o nouă perspectivă, sintetizând datele la un nou nivel. Astfel, decoderului îi va fi mai uşor să recreeze informaţia pentru că primeşte informaţii mai avansate. De asemenea, fiecare sursă are trăsături specifice prezente în mai multe locuri sau chiar pe toată durata semnalului, acestea putând fi valorificate prin transformer.

Structura tensorilor din Wave-U-Net este ($batch$, $canale$, t), iar pentru transformer este ($batch$, t, e), unde e este dimensiunea embedding-ului. Când ajungem la primul strat

din transformer (codificare poziţională), permutăm ultimele 2 straturi pentru a avea t pe poziţia percepută de transformer. Dimensiunea embedding-ului va fi 512, adică numărul de canale din Wave-U-Net. După parcurgerea transformerului, se realizează permutarea inversă şi se continuă reţeaua cu decodere. Embedding-ul este un nou spaţiu în care sunt sintetizate informaţii, în cazul acesta despre timp. Cele 512 canale au învăţat diferite perspective despre timp, deci pot reprezenta nişte valori de început pentru embedding foarte bune, nefiind nevoie de iniţializare aleatoare.

Straturile de self-attention permit eşantioanelor să interacţioneze între ele pentru a învăţa relaţii pe termen lung. Fiecare eşantion deţine următoarele informaţii: Q (ce informaţie caută de la celelalte), K (ce informaţie deţine), V (informaţia propriu-zisă). Aceşti tensori sunt iniţializaţi prin aplicarea unui strat complet conectat. După aceea, înmulţim Q cu transpusa lui K pentru a obţine scorurile de atenţie (dimensiune ($batch$, t, t)), adică pentru fiecare eşantion reţinem cât de importante sunt celelalte eşantioane. Rezultatului aplicăm funcţia softmax pe linii, care normalizează scorurile şi le transformă în probabilităţi, suma lor devenind 1. Deci fiecare eşantion îşi împarte un număr de procente din atenţie pentru altul. Înainte de aplicarea softmax, trebuie să ne asigurăm că valorile din matrice nu sunt prea mari şi cu diferenţe semnificative între ele, pentru a nu risca ca $softmax$ să calculeze câteva probabilităţi mari şi restul 0. Când calculam QK^{top}, fiecare element este o sumă de e produse. De aceea, împărţim fiecare valoare la $\sqrt{e}$. La final, înmulţim rezultatul cu V, pentru a obţine informaţiile de la celelalte eşantioane. Deci stratul de self-attention este atentie$(Q, K, V) = (\text{softmax}\left(\frac{QK^\top}{\sqrt{e}}\right))V$.

Deoarece operaţiile de self-attention sunt paralelizate prin înmulţire de matrici, primul pas al transformerului este de a integra în embedding informaţii despre poziţia fiecărui eşantion. Transformăm poziţiile fiecărui eşantion din indici în vectori de dimensiune e. Pentru asta, calculăm e factori de scalare ($\text{s}(2 \cdot i) = \text{s}(2 \cdot i + 1) = \frac{1}{10000^{\frac{2 \cdot i}{e}}}, i \in \overline{0, \frac{e}{2} - 1}$), apoi îi înmulţim cu poziţiile iniţiale, obţinând o matrice (t, e). Apoi aplicăm funcţia sin pentru elementele de pe coloane pare, iar pentru restul cos. Deci chiar dacă factorii de scalare sunt egali doi câte doi, prin aplicarea funcțiilor se face diferența. Acum fiecare rând reprezintă poziţia unui eşantion. Acesta este stratul de codificare poziţională, care se va aduna la stratul anterior.

După codificare poziţională, urmează 2 straturi de encoder. Aceste straturi sunt rezultatul normalizării sumei dintre stratul de self-attention şi un strat de feed- forward. Calcularea stratului de self-attention raportat la stratul anterior A are mai multe etape: calculăm atenţia bazat pe A, aplicăm dropout pentru că atenţia foloseşte straturi complet conectate care implică toţi neuronii, adunăm la self-attention A (skip connection pentru a nu pierde informaţiile anterioare), apoi normalizăm. Stratul de feed-forward are ca input stratul de self-attention şi presupune un strat complet conectat care măreşte dimensiunea embedding-ului pentru a permite operaţii mai complexe. După o activare ReLu, ne întoarcem la embedding-ul iniţial, iar apoi aplicăm dropout. Deci stratul A se transformă

în self-attention (care ţine cont şi de A) + feed-forward (care a aprofundat self-attention). Am fi putut păstra doar rezultatul din feed-forward şi să nu mai facem skip connection cu self-attention, dar feed-forward este prea abstract şi am pierde informaţii. Avem nevoie de skip connection pentru a atenua operaţiile care schimbă mult structura valorilor, păstrându-le în acelaşi timp efectul. Fără feed-forward, modelul ar merge în continuare, dar ar pierde din performanţă pentru că stratul are un rol important în accelerarea învăţării.

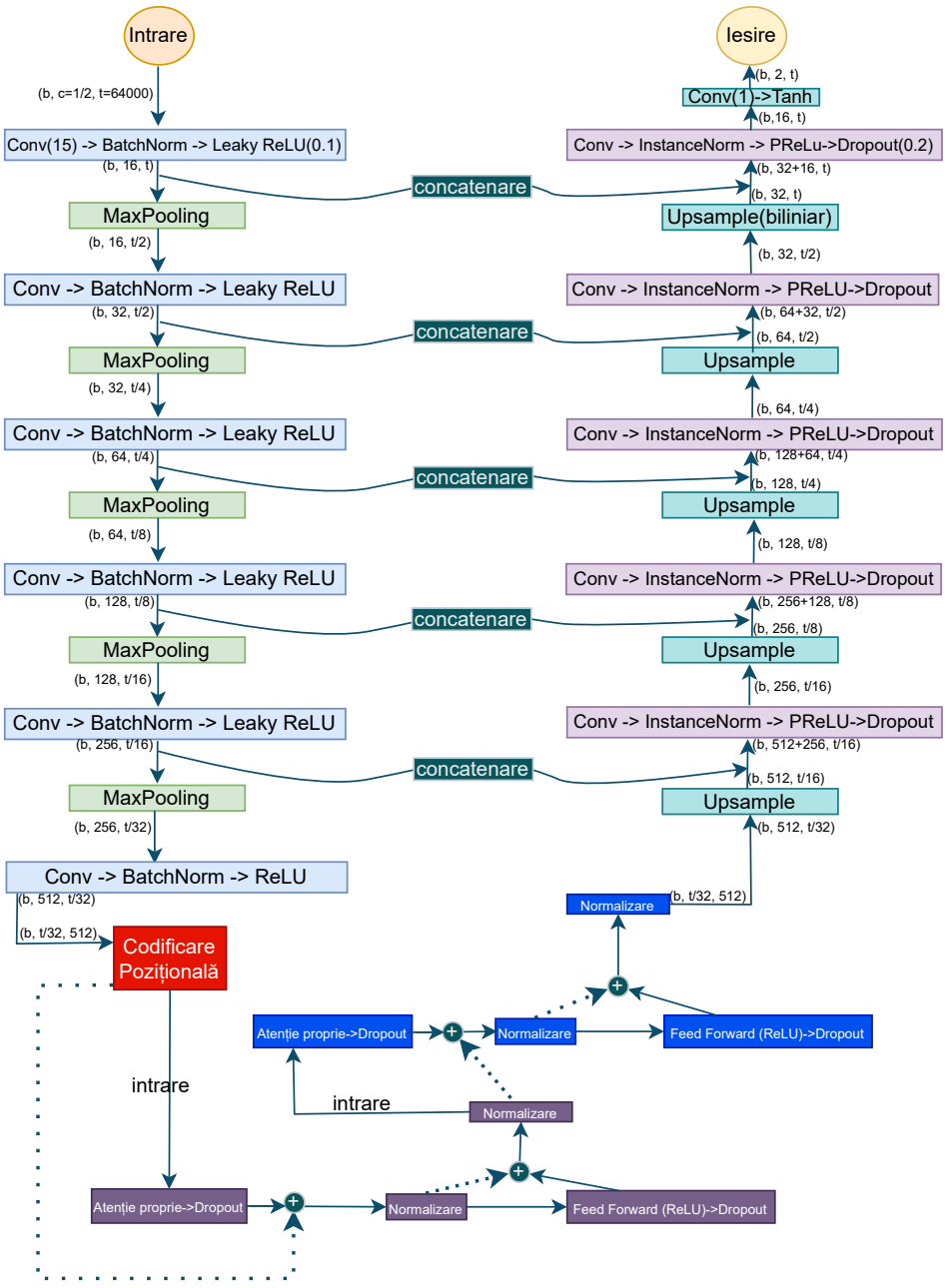

Figura 4.1: Structura Wave-U-Net cu self-attention

Capitolul 5

Evaluarea calității separării

5.1 Generarea setului de date

Întrucât setul Librimix nu are şi versiune stereo, pentru aceasta am luat aceleaşi perechi de înregistrări pentru care s-a creat versiunea mono. Pentru un mix fără zgomot sau reverberaţii, ne putem folosi de întârzierile celor două voci faţă de cele două canale. Acestea sunt generate aleator pentru fiecare amestec şi au valori pozitive şi negative. Întârzierea maximă e de 240 de eşantioane, adică 0.015 secunde. Întârzierile sunt relative la momentul când cele 2 microfoane au început să înregistreze.

Pentru intarziere am tăiat eşantioanele de la stânga sau dreapta (în funcţie de semn) şi am adăugat zerouri în cealaltă parte.

Ponderile sunt generate tot aleator, cu valori între (0.2, 1.3). Avem valori peste 1 pentru că unele microfoane pot amplifica semnalul. Într-un caz trivial de mixare, matricea de amestecare are doar aceste valori, ele reprezentând cu ce amplitudine ajung frecvenţele la microfoane. De obicei ponderea ar fi invers proporţională cu întârzierea pozitivă, deoarece dacă un semnal ajunge mai târziu la un microfon îşi poate pierde din amplitudine. Totuşi, pentru diversitatea setului de date şi presupunând că unele microfoane pot amplifica semnalul, nu am mai implementat această legătură. Înregistrările $s1$ şi $s2$ au aceeaşi durată, fiind şi durata înregistrărilor mixate. Formulele pentru microfoanele m1 şi m2 rezultate din sursele $s1$ şi $s2$ la momentul de timp t sunt:

$$m_1(t) = pondere_{11}\, s_1(t - intarziere_{11}) + pondere_{12}\, s_2(t - intarziere_{12})$$

$$m_2(t) = pondere_{21}\, s_1(t - intarziere_{21}) + pondere_{22}\, s_2(t - intarziere_{22})$$

Chiar dacă o diferenţă între ponderi poate genera canale suficient de diferite să uşureze separarea, dacă una dintre surse are ponderi mai mici decât cealaltă poate fi mai greu de detectat. Informaţiile despre ea nu au aceeaşi amplitudine raportat la sursa dominantă. În setul de date avem înregistrări de lungimi diferite (4s, 6s, 15s). Pentru uniformitate

în antrenare şi testare, le-am împărţit în înregistrări de câte 4 secunde. Cu o frecvenţă de 16 000 eşantioane pe secundă, ajungem la 64 000 de eşantioane. Pentru modelele clasice nu contează durata la fel de mult, dar am ales 4 secunde pentru că este o valoare rezonabilă pentru CNN şi Wave-U-Net. De asemenea, trebuie să fie o valoare divizibilă cu 32, deoarece în Wave-U-Net axa temporală se înjumătăţeşte de 5 ori.

5.2 Metrici de măsurare a erorilor

Evaluarea separării se poate face prin plotarea formei semnalelor, dar de multe ori semnalele pot arăta aproape identic iar separarea să nu fie calitativă. O voce înregistrată are multe semnale şi frecvenţe, nu se pot observa toate clar în forma undei. Afişarea spectrogramelor ar părea mai potrivită, dar pentru a decide dacă separarea funcţionează trebuie să testăm pe multe date. Deci cel mai bine folosim o formulă care aproximează numeric calitatea[15].

Semnalul aproximat poate avea mai multe tipuri de erori comparat cu semnalul corect, adică unde care nu ar trebui să apară: artefactele (apărute în procesul de separare, nu de la altă sursă), interferenţă (părţi din altă sursă prezente, separarea nu a reuşit să le elimine), părţi lipsă din semnalul corect şi zgomot.

SDR (Signal to Distortion Ratio) măsoară toate tipurile de erori şi calculează raportul dintre energia semnalului corect şi energia erorii totale. Energia unui semnal este pătratul normei de ordin 2 şi măsoară cât de intens e un semnal: $E(s) = \|s\|^2 = \sum_{i=1}^{n} s[i]^2$. Este preferată în locul normei şi pentru eficienţa computaţională. SIR masora eroarea venită de la interferenţe prin raportul dintre energia semnalului corect şi energia erorii de interferenţă. SAR măsoară eroarea artefactelor şi consideră că eroarea de interferenţă face parte din semnal (pentru că este un semnal corect, chiar dacă nu trebuie să apară).

5.3 Antrenare şi testare

Antrenarea a avut loc pe 3000 de exemple pentru fiecare model. În cazul reţelelor neuronale, am antrenat două modele separate pentru fiecare (pentru mono şi stereo). Pentru CNN, fiind un model mai simplu, am folosit loturi de câte 16 exemple, pentru Wave-U-Net simplu 8, iar pentru Wave-U-Net cu Transformer 4. Aş fi putut să folosesc numere diferite pentru acelaşi model în funcţie de numărul de canale de intrare, dar în arhitecturile construite numărul de canale de ieşire din straturile convoluționale este fix, nedepinzând de numărul canalelor de intrare. Deci tensorii nu vor avea dimensiuni mai mari pentru cazul stereo. Dublarea datelor de intrare poate creşte complexitatea, dar nu are un impact semnificativ. Optimizatorul folosit este Adam pentru că este rapid si are stabilitate.

Funcţia de pierdere folosită este MSE (Mean Squared Error). Pentru a determina care este permutarea corectă, calculăm pentru fiecare pereche cele două valori MSE, le facem media şi alegem valoarea mai mică. MSE este norma de ordin 2 a diferenţei la pătrat şi penalizează mai mult diferenţele mari, fiind în formă pătratică (unitate la pătrat): $\text{MSE}(\hat{s}, s) = \frac{1}{N}\sum_{i=1}^{N}(\hat{s}_i - s_i)^2$

Testarea pentru fiecare algoritm a fost făcută pe 200 de tupluri (sursele iniţiale, canalul/canalele, rezultatul), calculând timpul mediu şi media metricilor de măsurare SIR, SAR, SDR. Pentru modelele nesupravegheate, timpul de testare include si timpul de antrenare, deoarece la fiecare exemplu trebuie sa se antreneze pe loc. Pentru un exemplu, valorile metricilor sunt medie artimetica dintre erorile pentru cele doua separari.

5.4 Rezultate

5.4.1 Valorile metricilor

Model	$\overline{SDR}$ (dB)	$\overline{SAR}$ (dB)	$\overline{SIR}$ (dB)
NMF	1.95	4.86	7.80
CNN	2.88	5.14	5.43
Wave-U-Net	4.47	6.35	7.41
Wave-U-Net cu Transformer	5.20	7.24	7.97

Tabela 5.1: Cazul monofonic

Model	$\overline{SDR}$ (dB)	$\overline{SAR}$ (dB)	$\overline{SIR}$ (dB)
ICA	0.41	3.90	4.25
DUET	1.37	7.03	1.57
CNN	3.75	5.29	6.31
Wave-U-Net	4.60	6.74	7.82
Wave-U-Net cu Transformer	5.56	7.83	8.17

Tabela 5.2: Cazul stereofonic

Valorile pentru SDR sunt cele mai mici pentru că măsoară toate tipurile de erori. De aceea este cea mai potrivită pentru compararea metodelor. DUET nu generează prea multe artefacte, dar masca binară folosită cauzează interferenţe. Şi NMF foloseşte mască, dar este soft şi este calculată în două etape.

Din rezultatele ascultate, artefactele nu se aud deloc chiar dacă există. Problema principală sunt interferenţele, uneori auzindu-se şi cealaltă voce (în special la DUET, după cum demonstrează Tabela 5.2). În imagini, semnalele pentru DUET păreau separate foarte bine, dar s-a dovedit că forma undei nu este cel mai bun criteriu.

5.4.2 Timpi de testare

Model	$\overline{Timp}$ (s)
ICA stereo	5.05
NMF mono	0.81
DUET stereo	0.55
CNN mono	3.32
CNN stereo	3.39
Wave-U-Net mono	5.34
Wave-U-Net stereo	5.63
Wave-U-Net cu Transformer mono	25.76
Wave-U-Net cu Transformer stereo	28.57

Tabela 5.3: Timpi de testare

ICA este mai lent deoarece prelucrează pe rând direcţiile din matricea de separare, nu în paralel. La fiecare actualizare foloseşte multe înmulţiri de matrici, iar ortogonalizarea direcţiilor este costisitoare. DUET este rapid deoarece K-MEANS converge rapid şi calculează doar nişte distanţe. Lucrând pe spectrogramă, are mai multe informaţii disponibile, iar Transformata Rapidă Fourier are complexitate $\mathcal{O}(n \log n)$, unde n este numărul de eşantioane. În cazul NMF, actualizările efectuează şi operaţii element cu element, mai rapide decât înmulţirile de matrici. Reconstruirea surselor are paşi simpli, fără iteraţii pentru fiecare ca în cazul ICA. Un alt motiv este că lucrează cu un singur canal, deci nu putem spune clar că este mai rapid din rezultate.

În cazul reţelelor neuronale, se observă o creştere atunci când intrarea este stereo, deoarece modelul are de procesat date mai complicate, chiar dacă tensorii intermediari au aceleaşi dimensiuni. Odată cu complexitatea structurii modelului, creşte şi timpul de execuţie, mai ales după aplicarea straturilor de self-attention.

Dezavantajul modelelor de învăţare supravegheată este necesitatea datelor de antrenare în prealabil şi timpii mai mari pentru modelele mai complexe. Totuşi, acestea au obţinut scoruri mai bune pe fiecare tip de eroare faţă de majoritatea metodelor clasice. Excepţie face CNN mono, care a obţinut un SIR de 1.43 ori mai mic decât NMF. Acelaşi scor pentru Wave-U-Net este de 1.05 ori mai mic decât scorul lui NMF. Structura simplă a lui CNN nu îi dă un avantaj major în faţa metodelor clasice. Wave-U-Net cu self-attention a obţinut de departe cele mai bune rezultate peste tot. Pare cea mai potrivită alegere, dar timpul de procesare a 4 secunde de conţinut este destul de mare. Dacă este nevoie de a separa o înregistrare mai lungă, diferenţa se va vedea din ce în ce mai mult. În plus, contează puterea de procesare a fiecărui dispozitiv, pe un telefon putând fi nevoie de optimizări (de exemplu micșorarea preciziei numerelor). O rezolvare ar fi procesarea de secvențe mai mici și concatenarea ulterioară a rezultatelor.

Capitolul 6

Concluzii

După modificarea CNN-ului adăugând skip connections şi mai multe convoluţii şi normalizări, performanţa a crescut cu 55.21% pentru mono şi 22.67% pentru stereo (pe baza SDR). După introducerea arhitecturii Transformer, performanţa Wave-U-Net a crescut cu 16.3% pentru mono şi 20.87% pentru stereo.

Dintre metodele clasice, NMF a dat cele mai bune rezultate la erori. Chiar dacă a fost testată pe alt set de date decât DUET şi ICA, este relevant faptul că a reuşit să obţină scoruri mai mari deşi primeşte doar un canal. Simplul fapt că NMF nu este condiționată de minim 2 canale este un avantaj semnificativ. Multe înregistrări, chiar dacă sunt stereo, pot avea microfoanele foarte apropiate.

6.1 Posibilităţi de extindere

Metodele clasice nu sunt foarte eficiente în cazuri realiste. Pentru NMF am putea iniţializa matricile M şi N prin factorizare SVD (Singular Value Decomposition). O altă îmbunătăţire ar fi combinarea cu învăţarea automată. Spre exemplu, după separarea amplitudinilor în NMF, avem nevoie de o mască pentru a ţine cont şi de faze. Putem folosi un CNN pe spectrogramă asemănător cu cel din lucrare pentru a separa acele faze. Pentru DUET folosim masca binară, dar o mască soft poate fi mai utilă pentru că nu separă la fel de dur.

Algoritmii de separare trebuie să primească numărul de surse care trebuie separate, deci ar putea fi adăugată o metodă de a afla câte surse sunt active la un moment dat.

Ar putea fi adăugată şi o parte de diarizare vocală, aplicată după separarea surselor pentru că aşa funcţionează mai eficient. Diarizarea vocală recunoaşte, de exemplu, că vorbitorul din primele secunde e acelaşi cu cel de la finalul înregistrării.

Testarea şi antrenarea pe date cu zgomot ar trata cazuri mai realiste. Combinând cu diarizarea, un model de recunoaştere automată a vorbirii (ASR) şi antrenarea pe mai mult de două voci ar putea funcţiona ca un sistem de transcriere a conversaţiilor.

Bibliografie

[1] Sam Ansari, Abbas Saad Alatrany, Khawla A. Alnajjar, Tarek Khater, Soliman Mahmoud, Dhiya Al-Jumeily și Abir Jaafar Hussain, „A survey of artificial intelligence approaches in blind source separation", în *Neurocomputing* 561 (2023), p. 126895, ISSN: 0925-2312, DOI: 10.1016/j.neucom.2023.126895, URL: https://www.sciencedirect.com/science/article/pii/S0925231223010184.

[2] Ian Goodfellow, Yoshua Bengio și Aaron Courville, *Deep Learning*, MIT Press, 2016, pp. 326–366, URL: http://www.deeplearningbook.org.

[3] Charles R Harris, K Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith et al., „Array programming with NumPy", în *Nature* 585.7825 (2020), pp. 357–362, DOI: 10.1038/s41586-020-2649-2.

[4] J. D. Hunter, „Matplotlib: A 2D graphics environment", în *Computing in Science & Engineering* 9.3 (2007), pp. 90–95, DOI: 10.1109/MCSE.2007.55.

[5] Aapo Hyvärinen, „Fast and Robust Fixed-Point Algorithms for Independent Component Analysis", în *IEEE Transactions on Neural Networks* 10.3 (1999), pp. 626–634, DOI: 10.1109/72.761722.

[6] Daniel D. Lee și H. Sebastian Seung, „Learning the parts of objects by non-negative matrix factorization", în *Nature* 401.6755 (1999), pp. 788–791, DOI: 10.1038/44565, URL: https://doi.org/10.1038/44565.

[7] James MacQueen, „Some methods for classification and analysis of multivariate observations", în *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, University of California Press, 1967, pp. 281–297, URL: https://projecteuclid.org/euclid.bsmsp/1200512992.

[8] Brian McFee, Colin Raffel, Dawen Liang, Daniel P. W. Ellis, Matt McVicar, Eric Battenberg și Oriol Nieto, „librosa: Audio and Music Signal Analysis in Python", în *Proceedings of the 14th Python in Science Conference*, 2015, pp. 18–25, URL: https://librosa.org/.

[9] Bogdan Mihai, „Sampling rate and aliasing on a virtual laboratory”, în *Journal of Electrical and Electronics Engineering* 2 (2009), URL: https://www.researchgate.net/publication/40422576_Sampling_rate_and_aliasing_on_a_virtual_laboratory.

[10] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga et al., „PyTorch: An Imperative Style, High-Performance Deep Learning Library”, în *Advances in Neural Information Processing Systems*, vol. 32, Curran Associates, Inc., 2019, pp. 8024–8035, URL: https://papers.nips.cc/paper_files/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

[11] C.E. Shannon, „Communication in the Presence of Noise”, în *Proceedings of the IRE* 37.1 (Ian. 1949), pp. 10–21, DOI: 10.1109/jrproc.1949.232969, URL: https://doi.org/10.1109/jrproc.1949.232969.

[12] David R. R. Smith, Roy D. Patterson, Richard Turner, Hideki Kawahara și Toshio Irino, „The processing and perception of size information in speech sounds”, în *The Journal of the Acoustical Society of America* 117.1 (2005), pp. 305–318, URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4059169/.

[13] Daniel Stoller, Simon Ewert și Simon Dixon, „Wave-U-Net: A Multi-Scale Neural Network for End-to-End Audio Source Separation”, în *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018, URL: https://arxiv.org/abs/1806.03185.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser și Illia Polosukhin, „Attention is All You Need”, în *Advances in Neural Information Processing Systems*, vol. 30, 2017, URL: https://arxiv.org/abs/1706.03762.

[15] Emmanuel Vincent, Rémi Gribonval și Cédric Févotte, „Performance measurement in blind audio source separation”, în *IEEE Transactions on Audio, Speech, and Language Processing* 14.4 (2006), Open access version available at INRIA HAL, pp. 1462–1469, URL: https://inria.hal.science/inria-00544230v1.

[16] Ying Xue, „J. Phys.: Conf. Ser. 1168 022022”, în *Journal of Physics: Conference Series* 1168.2 (2019), p. 022022, DOI: 10.1088/1742-6596/1168/2/022022, URL: https://iopscience.iop.org/article/10.1088/1742-6596/1168/2/022022/pdf.

[17] Ozgur Yilmaz și Scott Rickard, „Blind Separation of Speech Mixtures via Time-Frequency Masking”, în *IEEE Transactions on Signal Processing* 52.7 (2004), pp. 1830–1847, DOI: 10.1109/TSP.2004.828896, URL: https://doi.org/10.1109/TSP.2004.828896.