Universitatea POLITEHNICA din București
Facultatea de Electronică, Telecomunicații și Tehnologia Informației

Sistem automat de reducere a reverberației și eliminare a zgomotului din semnale audio

Lucrare de diplomă

Prezentată ca cerință parțială pentru obținerea titlului de *Inginer*
în domeniul *Electronică, Telecomunicații și Tehnologia Informației*
programul de studii *Microelectronică, Optoelectronică și Nanotehnologii*

Conducători științifici
Prof. Dr. Ing. Corneliu Burileanu
As. Univ. Drd. Ing. Ana Neacșu

Absolvent
Răzvan-Paul Ciubotaru

Anul 2021

Universitatea "Politehnica" din Bucureşti
Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei
Program de studiu **MON**

Anexa 1

TEMA PROIECTULUI DE DIPLOMĂ

a studentului **CIUBOTARU F.T. Răzvan-Paul , 441E-MON**

1. Titlul temei: Sistem automat de reducere a reverberației și eliminarea a zgomotului din semnale audio

2. Descrierea temei și a contribuției personale a studentului (în afara părții de documentare):
Acest proiect are ca obiectiv eliminarea reverberației camerei și a zgomotului suprapus peste semnalele audio înregistrate în diferite încăperi. În acest scop vor folosi metode de învățare automată.
Baza de date de lucru se va forma prin convoluția unor semnale muzicale (înregistrate la orgă) cu funcțiile de transfer ale diferitelor camere, la care se adaugă zgomot gaussian.
Datele astfel obținute se vor prelucra utilizând metode de prelurare a semnalelor cunoscute (proprietăți statistice, transformate Fourier, Wavelet, etc.). Analiza se va face atât în domeniul timp, cât și în domeniul frecvență.
Ulterior, aceste date vor fi transferate către o rețea neurală ce va elimina reverberația și zgomotul. Metoda propusă va fi comparată cu alte metode standard prezente în literatura de specialitate.

3. Discipline necesare pt. proiect:
PDS, TAPDS, SS

4. Data înregistrării temei: 2020-11-24 13:29:58

Conducător(i) lucrare,
As. drd. Ing. Ana-Antonia NEACȘU

Prof. Corneliu BURILEANU

Student,
CIUBOTARU F.T. Răzvan-Paul

Director departament,
Prof. dr. ing. Claudius DAN

Decan,
Prof. dr. ing. Mihnea UDREA

Cod Validare: **526f67b18d**

Declaraţie de onestitate academică

Prin prezenta declar că lucrarea cu titlul *Sistem automat de reducere a reverberaţiei şi eliminare a zgomotului din semnale audio*, prezentată în cadrul Facultăţii de Electronică, Telecomunicaţii şi Tehnologia Informaţiei a Universităţii "Politehnica" din Bucureşti ca cerinţă parţială pentru obţinerea titlului de *Inginer* în domeniul Inginerie Electronică şi Telecomunicaţii/ Calculatoare şi Tehnologia Informaţiei, programul de studii *Microelectronică, Optoelectronică şi Nanotehnologii* este scrisă de mine şi nu a mai fost prezentată niciodată la o facultate sau instituţie de învăţământ superior din ţară sau străinătate.

Declar că toate sursele utilizate, inclusiv cele de pe Internet, sunt indicate în lucrare, ca referinţe bibliografice. Fragmentele de text din alte surse, reproduse exact, chiar şi în traducere proprie din altă limbă, sunt scrise între ghilimele şi fac referinţă la sursă. Reformularea în cuvinte proprii a textelor scrise de către alţi autori face referinţă la sursă. Înţeleg că plagiatul constituie infracţiune şi se sancţionează conform legilor în vigoare.

Declar că toate rezultatele simulărilor, experimentelor şi măsurătorilor pe care le prezint ca fiind făcute de mine, precum şi metodele prin care au fost obţinute, sunt reale şi provin din respectivele simulări, experimente şi măsurători. Înţeleg că falsificarea datelor şi rezultatelor constituie fraudă şi se sancţionează conform regulamentelor în vigoare.

Bucureşti, Iunie 2021.
Absolvent: Răzvan-Paul Ciubotaru

Cuprins

Lista figurilor . . . iii

Lista tabelelor . . . v
Lista acronimelor . . . vi

Introducere . . . 1
Motivație . . . 1
Aplicabilitate . . . 1
Obiective . . . 1

1. Noțiuni teoretice . . . 3
1.1. Clasificarea semnalelor . . . 3
1.1.1. Reprezentarea semnalelor în domeniul timp și în domeniul frecvență . . . 4
1.1.2. Semnale din domeniul audio . . . 6
1.2. Zgomot și reverberație . . . 7
1.2.1. Zgomotul aditiv . . . 7
1.2.2. Fenomenul de reverberație . . . 9
1.3. Rețele Neurale – Concepte generale . . . 10

2. Soluții propuse . . . 17
2.1. Deverberarea semnalelor . . . 17
2.2. Eliminarea zgomotului . . . 19
2.2.0.1. Analiza semnalelor prin ferestruire . . . 20
2.2.1. Metode clasice de eliminare a zgomotului . . . 21
2.2.1.1. Filtrarea Wiener . . . 21
2.2.1.2. Filtrarea adaptivă . . . 23
2.2.2. Eliminarea zgomotului folosind rețele neurale . . . 26
2.2.2.1. Rețele neurale complet interconectate . . . 26
2.2.2.2. Rețele capsulă . . . 27

3. Scheme de învățare folosind constrângeri . . . 31
3.1. Formularea problemei . . . 31
3.2. Constrângeri propuse . . . 32

4. Rezultate experimentale 37

Concluzii 43
Concluzii generale 43
Contribuții personale 43
Dezvoltări ulterioare 44
Bibliografie 45

Lista figurilor

1.1. Semnal analogic (a), eșantionat (b), cuantizat (c), discret (d). 4
1.2. Spectre de putere ale zgomotelor colorate: mov, albastru, alb, roz și maro. . . . 7
1.3. Funcția de autocorelație și densitatea spectrală de putere a zgomotului alb. . . . 8
1.4. Densitatea de probabilitate a zgomotului alb Gaussian. 9
1.5. Mărimile caracteristice filtrului pieptene 10
1.6. Arhitectura generală a unei rețele neurale. 11
1.7. Graficele funcțiilor *sigmoid* și *tanh* 13
1.8. Graficele funcțiilor *ReLU* și *Leaky ReLU* 14
1.9. Fenomenul de supraînvățare este ilustrat de către încadrarea făcută de linia verde. 16
2.1. Semnal reverberat și semnal corupt de zgomot. 18
2.2. Fereastră dreptunghiulară. 20
2.3. Fereastră Hanning. 22
2.4. Principiul estimării liniare al filtrului Wiener. 22
2.5. Schema de prelucrare folosind filtrul Wiener. 24
2.6. Principiul de funcționare al filtrării adaptive. 25
2.7. Configurația de identificare de sistem. 26
2.8. Arhitectura rețelei neurale interconectate. 27
2.9. Sistemul de eliminare a zgomotului folosind FCN. 28
2.10. Stratul propus pentru rețeaua capsulă. 28
2.11. Arhitectura propusă pentru rețeaua capsulă. 29
4.1. Comparație între densitățile spectrale de putere pentru baza de date de test. . . 40

Lista tabelelor

4.1. Comparație între metodele clasice și cea bazată pe rețea neurală interconectată . 38
4.2. Comparație între metodele clasice și cele bazate pe rețea capsulă. 39
4.3. Comparație între metodele clasice și cele bazate pe modele constrânse. 39
4.4. FCN vs. CapsNet – metrici de performanță 40

Lista acronimelor

AM = Amplitude Modulation
Adam = Adaptive Moment Estimation
BD = Blu-Ray Disc
CC = Cross Correlation
CD = Compact Disc
CE = Cross Entropy
CapsNet = Capsule Network
DFB = Dual Forward Backward
DVD = Digital Video Disc
FCN = Fully Connected Network
FFT = Fast Fourier Transform
FIR = Finite Impulse Response
FISTA = Fast Iterative Shrinkage-Thresholding Algorithm
FM = Frequency Modulation
ISTFT = Inverse Short-Time Fourier Transform
LMS = Least Mean Squares
MAE = Mean Average Error
MCCE = Multi Class Cross Entropy
MSE = Mean Squared Error
NLMS = Normalized Mean Squared Error
PSNR = Peak Signal to Noise Ratio
PnP = Plug and Play
RMN = Rezonanță Magnetică Nucleară
RMSE = Root Mean Squared Error
ReLU = Rectifier Linear Unit
SGD = Stochastic Gradient Descent
SLIT = Sistem Liniar și Invariant în Timp
SNR = Signal to Noise Ratio
STFT = Short-Time Fourier Transform

Introducere

Motivație

Scopul principal al acestei lucrări de diplomă constă în implementarea unui sistem automat de îmbunătățire a semnalelor audio, prin eliminarea efectelor reverberației și ale zgomotului aditiv suprapus peste semnalele audio utile. Acesta este un subiect complex, abordat de multe echipe de cercetare de-a lungul timpului [1], [2]. Odată cu evoluția sistemelor de învățare automată, noi metode de îmbunătățire au fost propuse frecvent, domeniul dezovltându-se într-un ritm alert [3], [4].

Aplicabilitate

Îmbunătățirea calității semnalelor audio este o problemă importantă în domeniul audio, cu aplicații în vorbire sau în domeniul muzical. Astfel de metode pot îmbunătăți experiența concertelor live, pot îmbunătăți calitatea înregistrărilor efectuate în medii zgomotoase sau pot spori inteligibilitatea semnalelor vocale transmise prin diferite medii (telefonie, înregistrări etc.) pentru a facilita comunicarea între oameni sau comunicarea om-mașină (spre exemplu, asistenții virtuali inteligenți).

Obiective

Proiectul își propune realizarea următoarelor obiective:

- Crearea bazei de date de lucru prin convoluția unor semnale muzicale (înregistrate la orgă) cu funcțiile de transfer ale diferitelor camere, la care se adaugă zgomot alb gaussian;
- Propunerea unui algoritm iterativ cu ajutorul căruia să se poată elimina efectul reverberației din semnalul audio;
- Eliminarea zgomotului aditiv Gaussian de medie nulă și dispersie constantă folosind sisteme bazate pe metode clasice de prelucrare a semnalelor (filtre digitale);
- Implementarea unor sisteme bazate pe rețele neurale ce vor elimina zgomotul aditiv Gaussian a căror performanță va fi comparată cu cea a metodelor clasice.

Capitolul 1
Noțiuni teoretice

1.1 Clasificarea semnalelor

Un semnal reprezintă o cantitate fizică care este funcție de una sau mai multe variabile independente, cum ar fi timpul, distanța, temperatura etc. Dacă semnalul este funcție de o singură variabilă independentă, este numit semnal unidimensional (1-D); dacă este funcție de mai multe variabile independente, este numit multidimensional (M-D). Semnalele pot fi clasificate în funcție de evoluția lor în timp ca fiind:

– **Deterministe**: Aceste semnale pot fi descrise în mod unic de o expresie matematică $s(t)$ sau de un tabel de valori.

– **Nedeterministe**: Aceste semnale se desfășoară indefinit în timp și sunt guvernate, cel puțin în parte, de legi probabilistice, ele numindu-se și semnale *aleatoare* [5].

Clasa semnalelor deterministe conține următoarele tipuri de semnale:

- *Semnale periodice*: Sunt complet descrise de o perioadă. Aceste semnale sunt descrise matematic astfel:
$$x(t) = x(t \pm kT), \quad k \in \mathbb{Z} \tag{1.1}$$
Aici, T reprezintă perioada fundamentală a semnalului $x(t)$, adică intervalul de timp cel mai mic după care valorile semnalului $x(t)$ se repetă.
- *Semnale cvasiperiodice*: Sunt formate din componente sinusoidale necorelate armonic.
- *Semnale aperiodice de durată finită*: Semnale de durată finită care variază pe un interval scurt și apoi descresc către o valoare constantă (spre exemplu, răspunsul la impuls al unei încăperi).
- *Semnale aperiodice care nu au durată finită*: Majoritatea semnalelor aleatoare provenite din surse naturale sau artificiale, de exemplu zgomotele.

Aceste semnale nu se întâlnesc în mod uzual în situațiile practice. De obicei, avem de a face cu semnale aleatoare, care nu sunt predictibile și pot fi descrise doar de termenii statistici ai acestora; caracterul aleator poate proveni din însăși natura fenomenului reprodus sau din modul de transmitere a lor pe un canal de comunicații, etc. Totuși, în multe aplicații concrete se poate utiliza un model matematic al semnalelor de interes, model care poate simplifica mult

analiza, fără a duce la micșorarea semnificativă a preciziei [5].
În telecomunicaţii, semnalele utile, purtătoare ale informaţiei de transmis, sunt aleatoare, e.g. semnalul vocal la ieşirea unui microfon, semnalul de imagine video, semnalul la ieşirea unui scanner etc. Pe de altă parte, sunt propuse şi semnale deterministe, cum sunt semnalele de test (de exemplu semnale sinusoidale), semnale de sincronizare (în TV) etc. [6]. De asemenea, semnalele pot fi clasificate în funcție de parametrii statistici ai acestora astfel:

- **Staționare**: Semnalele își păstrează proprietățile statistice în raport cu variația timpului.
- **Nestaționare**: Parametrii statistici ai semnalelor variază în raport cu timpul.

După modul în care evoluează în timp, semnalele utilizate în telecomunicaţii, deterministe sau aleatoare, pot fi: analogice (continue, cu nivel variabil), cuantizate, eşantionate sau eşantionate şi cuantizate, după cum se poate observa și în figura 1.1: [6]

1. Semnalele analogice au un număr infinit de valori în orice interval de reprezentare în domeniul timp. Aceste semnale sunt cel mai des întâlnite în practică, însă este necesară discretizarea acestor semnale pentru analiza lor;
2. Semnalele eşantionate (cuantizate sau discretizate în timp) au niveluri specificate (existente) numai în anumite momente care formează un şir discret;
3. Semnale cuantizate (discretizate în nivel, cu continuitate în timp), cu niveluri existente pe întreaga axă a timpului, dar care formează un şir de valori discrete în intervalul de nivele limită;
4. Semnale eşantionate şi cuantizate cu niveluri şi momentele de existenţă ce formează șiruri discrete.

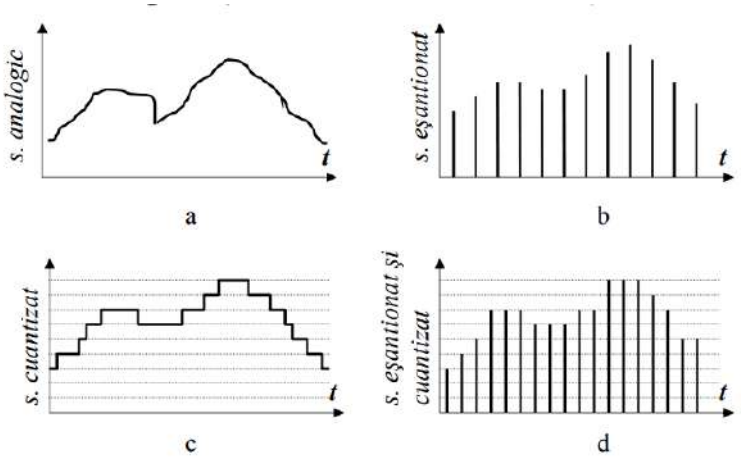


Figura 1.1: Semnal analogic (a), eșantionat (b), cuantizat (c), discret (d).

1.1.1 Reprezentarea semnalelor în domeniul timp și în domeniul frecvență

Orice semnal $x(t)$ poate fi caracterizat prin două reprezentări:

- Reprezentare în domeniul timp – numită de asemenea și *forma de undă a semnalului*;

- Reprezentare în domeniul frecvență – numită și *Spectrul de frecvențe al semnalului.*

Oricare dintre aceste două reprezentări caracterizează în mod univoc semnalul, adică unei reprezentări în domeniul timp îi corespunde o singură reprezentare în domeniul frecvenţă şi invers, unei reprezentări în frecvenţă îi corespunde o singură reprezentare în timp. Tranziția de la domeniul timp la domeniul frecvență și invers se face prin intermediul transformatelor.

Pentru reprezentarea spectrului de frecvențe a unui semnal periodic se preferă folosirea transformatei Fourier, iar pentru cele aperiodice se preferă folosirea fie a transformatei Laplace, fie a celei Fourier.

De asemenea, evoluţia unui semnal în timp poate fi mai uşor de analizat decât evoluţia sa în frecvenţă (care poate fi eventual "auzită" – dacă frecvenţele semnalului se situează în domeniul audio) [7].

Pentru a caracteriza un semnal în domeniul timp, definim secvența impuls-unitate:

$$\delta(n) = \begin{cases} 1, & \text{dacă } n = 0 \\ 0, & \text{dacă } n \neq 0 \end{cases} \tag{1.2}$$

Astfel, în reprezentarea discretă în domeniul timp, un semnal este constituit dintr-o sumă de astfel de impulsuri unitate, deplasate și ponderate dupa valorile semnalului la momentul respectiv:

$$x(n) = \sum_{k=-\infty}^{\infty} x(k)\delta(n-k) \tag{1.3}$$

În domeniul frecvență, un semnal discret poate fi privit ca o superpoziție de semnale sinusoidale ce denotă frecvențele din spectrul său:

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n}, \tag{1.4}$$

Tranziția în domeniul timp din domeniul frecvență se face după formula:

$$x(n) = \frac{1}{2\pi}\int_{-\pi}^{+\pi} X(e^{j\omega})e^{j\omega n}d\omega \tag{1.5}$$

În ecuațiile prezentate anterior, ω reprezintă frecvența digitală a spectrului, măsurată în radiani: $\omega = \frac{2\pi f}{f_s}$, unde f_s reprezintă frecvența de eșantionare a semnalului discret. De observat faptul că $e^{-j\omega n} = e^{-j(\omega \pm 2k\pi)n}$, de aceea $X(e^{jw})$ este o mărime periodică de perioadă 2π, de aceea este nevoie de o singură perioadă pentru a descrie spectrul de frecvențe al semnalului $x(n)$. Acesta este o mărime complexă ce caracterizează fiecare componentă spectrală a semnalului atât prin informația de modul $|X(e^{j\omega})|$, cât și prin informația de fază $arg\{X(e^{j\omega})\}$, date de coeficienții $Re\{X(e^{j\omega})\}$ și $Im\{X(e^{j\omega})\}$.

Întrucât $X(e^{j\omega})$ este o mărime continuă, pentru analiza acestui spectru folosind tehnici software este necesară eșantionarea lui, astfel obținând mărimea $X(k)$, cu $k \in \{0, \ldots, N-1\}$, unde N este numărul de puncte. Operația poartă numele de transformată Fourier rapidă (*eng. FFT - Fast Fourier Transform*):

$$X(k) = \sum_{n=0}^{N-1} x(n)exp(\frac{-2\pi jkn}{N}), k \in \{0, \ldots, N-1\} \tag{1.6}$$

Astfel, pentru o rezoluție în timp T_s, în frecvență vom folosi o rezoluție $\Omega = \frac{1}{NT_s}$. Reprezentările frecvențiale folosesc doar informația de amplitudine $|X(k)|$, neglijând informația de fază.

Acest lucru se datorează faptului că urechea este, în mare măsură, insensibilă la modificări de fază [8].

Din moment ce se lucrează cu semnale discrete, este necesară eșantionarea acestora prin intermediul teoremei eșantionării (teorema lui Shannon):

Orice semnal $x(t)$, ce are o bandă de frecvenţă limitată (banda nu este infinită), este complet definit (univoc determinat) prin eşantioanele sale $x(nT_s)$, dacă perioada de eşantionare, T_s, îndeplineşte condiţia:

$$T_s \leq \frac{1}{2f_M} \tag{1.7}$$

Dacă $f_s = \frac{1}{T_s}$, atunci condiția de mai sus poate fi rescrisă astfel:

$$\frac{f_s}{2} = f_N \geq f_M, \tag{1.8}$$

unde f_N poartă numele de *frecvența Nyquist*. De aceea, condiția din relația (1.6) poartă numele de *condiția lui Nyquist* [7].

Altfel spus, este necesar ca frecvența minimă de eșantionare a unui semnal analog să fie *minim* dublul frecvenței maxime din banda semnalului pentru a evita fenomenul de împrăștiere (aliere) spectrală: componentele periodice din spectrul $X(e^{j\omega})$ se vor suprapune parțial, determinând apariția componentelor spectrale false, nedorite.

1.1.2 Semnale din domeniul audio

Semnalele cu spectrul în intervalul [20Hz ... 20KHz] sunt considerate semnale de audiofrecvenţă, deoarece sunt percepute de urechea umană și ele sunt date de variaţii ale presiunii aerului. Principalele categorii în care se regăsesc aceste semnale sunt semnale *vocale* și semnale *muzicale*. Semnalul vocal are spectrul extins de la 50 Hz la 7KHz (componentele din afara acestui interval transportă sub 0.1% din puterea totală). Semnalele muzicale au spectrul extins în gama de frecvențe [20Hz ... 15KHz], însă s-a constatat că fidelitatea audiției este îmbunătățită pentru intervalul de frecvențe 50-100Hz ... 10KHz [6].

Există mai multe clase de semnale audio ce diferă din punct de vedere al lărgimii de banda, a gamei dinamice și a calității oferite:

- semnale vocale de bandă îngustă: 300Hz - 3,4KHz; frecvențe specifice pentru aplicațiile de telefonie
- semnale vocale de bandă largă: 50Hz - 7KHz; utilizate în aplicații de recunoaștere a vorbirii sau a vorbitorului, precum și în sinteza vorbirii cu integibilitate și naturalețe foarte bune
- semnale audio de bandă intermediară: maximul benzii este 10KHz pentru semnale AM (*eng. Amplitude Modulation*), respectiv 15KHz pentru semnale FM (*eng. Frequency Modulation*)
- semnale audio de bandă largă și înaltă fidelitate, cu frecvențe în întreaga bandă audio, stocate pe CD, DVD, DAT, BD etc.

Pentru semnalele audio de bandă largă există 3 clase principale de prelucrare digitală: înregistrarea și redarea digitală, prelucrări diverse în studiouri profesionale și tehnici de compresie pentru stocare și transmisie [9].

1.2 Zgomot și reverberație

1.2.1 Zgomotul aditiv

Zgomotul reprezintă cea mai des întâlnită perturbație în prelucrarea semnalelor și a constituit dintotdeauna o problemă majoră în acest domeniu. Acesta este dat de obicei de o perturbație aleatoare, iar cauzele proveniență lui sunt de cele mai multe ori naturale. Există nenumărate tipuri de zgomote, iar discuția se poate face după domeniul în care acestea sunt întâlnite: spre exemplu, în cazul semnalelor electrice, zgomotul perturbator poate proveni de la rețeaua de alimentare.

În analiza semnalelor audio, cele mai des întâlnite zgomote ce perturbă semnalele sunt zgomotele *colorate*. Ele sunt numite astfel deoarece densitatea spectrală de putere a acestor zgomote este asemănătoare cu spectrul radiației luminoase a culorii după care ele sunt numite. Spectrele acestor zgomote sunt direct legate de frecvența lor: spre exemplu, în cazul zgomotului roz, densitatea spectrală de putere este direct proporțională cu $\frac{1}{f}$ sau în cazul zgomotului violet, unde densitatea este proporțioanlă cu f^2, iar f reprezintă frecvența componentelor spectrale. În figura 1.2 sunt prezentate diferite spectre de putere ale zgomotelor întâlnite în prelucrarea semnalelor audio:

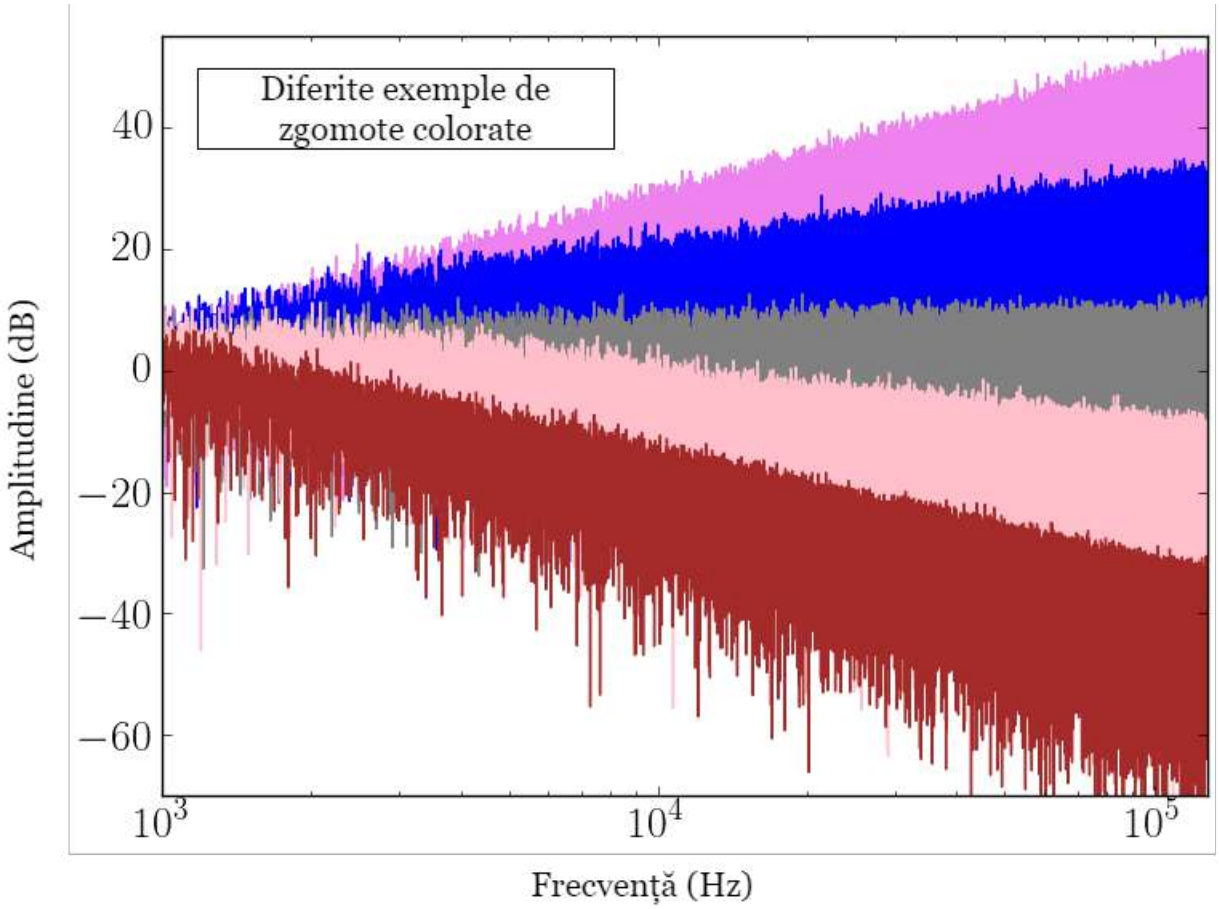


Figura 1.2: Spectre de putere ale zgomotelor colorate: mov, albastru, alb, roz și maro.

În această lucrare, zgomotul suprapus peste semnalele utile se presupune a fi zgomot alb aditiv. Zgomotul alb este un semnal staționar ce are toate eșantioanele complet necorelate între ele. Acesta are medie nulă ($\mu = 0$), variață constantă iar funcția sa de autocorelație este direct proporțională cu impulsul unitate:

$$R_{xx}(n) = \sigma^2 \delta(n) \tag{1.9}$$

unde σ^2 reprezintă varianța zgomotului. Pentru n = 0, autocorelația devine $R_{xx}(0) = \sigma^2$. Din moment ce valoarea autocorelației în origine este egală cu puterea semnalului, puterea zgomotului alb va fi dată chiar de varianța acestuia (acest lucru reiese și din faptul că media zgomotului este nulă).

Conform teoremei $Wiener - Khintchine$, denistatea spectrală de putere a zgomotului alb este:

$$S_{xx}(e^{j\omega}) = \sum_{n=-\infty}^{+\infty} R_{xx}(n)e^{-j\omega n} = \sum_{n=-\infty}^{+\infty} \sigma^2\delta(n)e^{-j\omega n} = \sigma^2 \tag{1.10}$$

În analogie cu spectrul luminos unde lumina albă conține toate lungimile de undă din spectrul vizibil, denistatea de putere a zgomotului alb are valoare constantă și conține toate frecvențele din banda disponibilă.

Zgomotul alb poate avea orice tip de densitate de probabilitate. În această lucrare, zgomotul va fi modelat de o densitatea de probabilitate normală de medie nulă și varianță constantă, fiind astfel vorba despre un zgomot alb Gaussian. Funcția ce modelează zgomotul este dat de densitatea de probabilitate a unei distribuții gaussiene de medie μ și deviație standard σ, dată de formula:

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} exp\left[\frac{(x-\mu)^2}{2\sigma^2}\right] \tag{1.11}$$

Zgomotul alb reprezintă un model teoretic deosebit de important în analiza statistică a semnalelor, întrucât peste semnalele utile se pot suprapune nu numai interferențe de bandă îngustă, ci și diferite semnale aleatorii nedorite de bandă largă, de exemplu: radiația electromagnetică. Pentru a modela comportamentul acestor fenomene și pentru a proiecta diverse sisteme de îndepărtare a lor se poate studia comportamentul zgomotului alb în astfel de situații [9].

Pentru a ilustra grafic funcțiile ce definesc statistic zgomotul alb, am presupus o distribuție de probabilitate gaussiană standard, adică de medie $\mu = 0$ și varianță $\sigma^2 = 1$. În figura 1.3 sunt prezentate funcția de autocorelație și densitatea spectrală de putere a zgomotului alb.

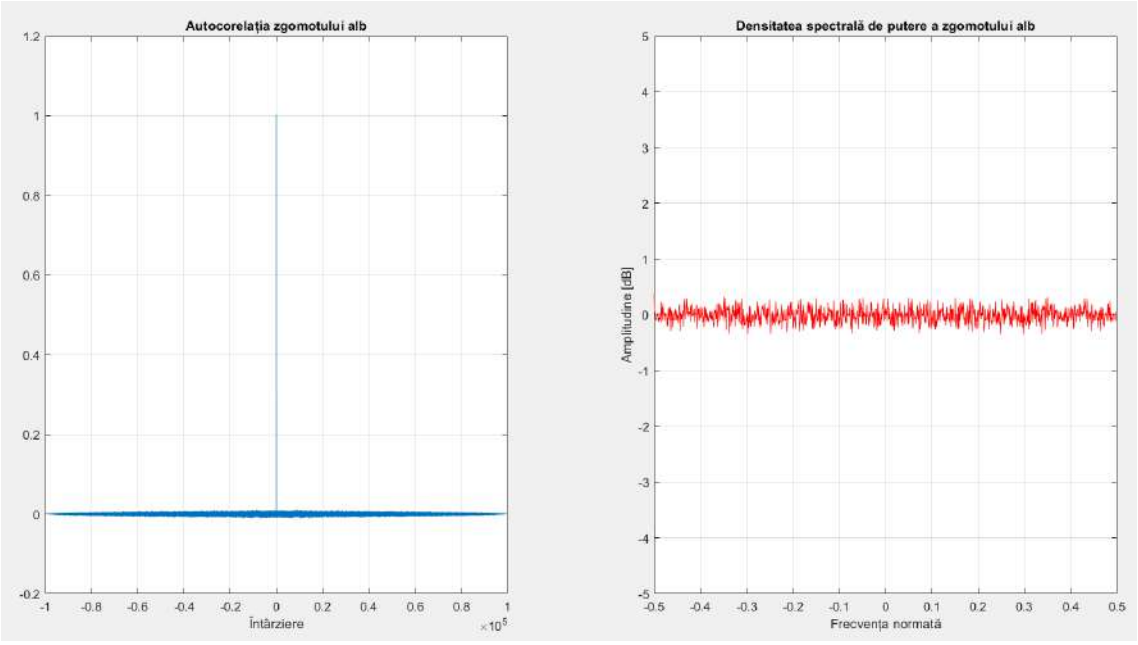


Figura 1.3: Funcția de autocorelație și densitatea spectrală de putere a zgomotului alb.

Am ilustrat de asemenea în figura 1.4 atât zgomotul simulat cu ajutorul `Matlab`, cât și

densitatea de probabilitate a funcției ce modelează zgomotul. Pentru a avea o reprezentare cât mai fidelă a densității de probabilitate, am simulat zgomotul pentru un număr $N = 100000$ eșantioane, astfel încât sa existe suficient de multe valori pentru a calcula probabilitatea ca X să se găsească în intervalul $[x, x + \Delta x]$: $p(x \leq X \leq x + \Delta x)$, unde X este variabila aleatoare ce reprezintă zgomotul iar x reprezintă valoarea amplitudinii la un anumit moment de timp.

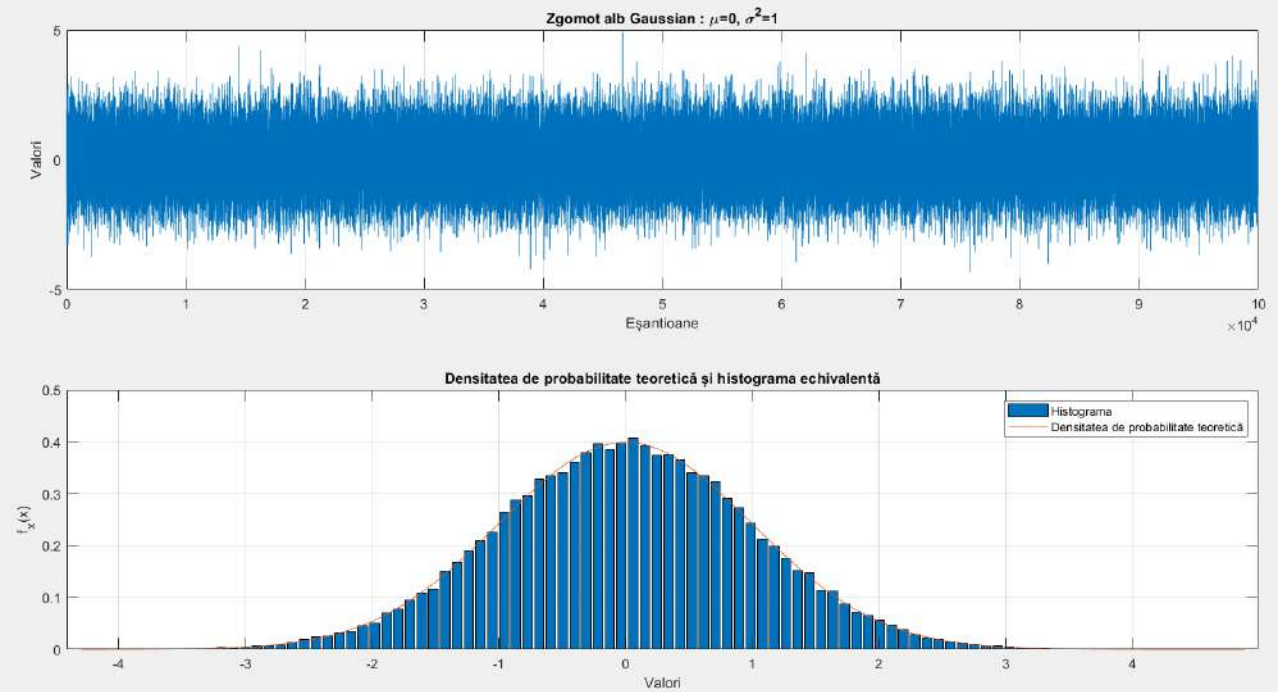


Figura 1.4: Densitatea de probabilitate a zgomotului alb Gaussian.

1.2.2 Fenomenul de reverberație

Reverberația reprezintă fenomenul de persistență a unui sunet într-o încăpere închisă datorită reflexiilor repetate, după ce unda sonoră a încetat să mai emită.

Timpul în care se manifestă această persistență a sunetului se numește timp de reverberație, notat și *T60*. El este definit ca timpul în care nivelul presiunii sonore scade cu 60dB față de nivelul inițial. Cu cât timpul de reverberație este mai mic, cu atât percepția unei surse sonore de către un ascultător aflat în încăperea respectivă este mai fidelă, iar calitatea înregistrărilor sonore va fi mai bună. Se propune în această lucrare minimizarea efectului reverberației pentru asigurarea unei calități cât mai bune a înregistrărilor făcute într-o anumită încăpere.

Pe de altă parte, timpul optim de reverberație poate fi determinat ținând cont de tipul încăperii (ex. sală de concerte sau sală de conferințe). De aceea, nu ne dorim mereu ca reverberația să nu existe într-o anumită încăpere. Spre exemplu, vorbirea într-o cameră în care nu există aproape deloc reverberație oferă un sentiment nefiresc deoarece niciun sunet nu se întoarce la urechile vorbitorului, vocea devenind astfel foarte dificil de controlat. Astfel, reverberația adecvată poate modifica modul în care se aud sunetele, simulând un anume spațiu acustic, de aceea trebuie luată în considerare modelarea artificială a reverberației.

Fenomenul de reverberație este simulat cu ajutorul unor filtre digitale ce funcționează după următorul principiu: semnalul audio original este întârziat și sumat cu același semnal neîntârziat. Cele mai folosite astfel de filtre sunt filtrul pipetene (*eng. comb filter*) și filtrul trece-tot (*eng. all-pass filter*) [9].

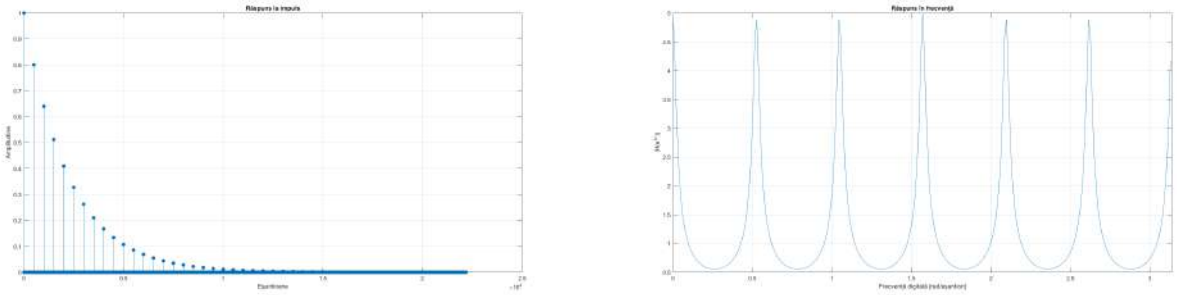
Filtrul pieptene constă în întârzierea semnalului de ieșire ce este readus la intrare. El elimină anumite frecvențe rezonante, simulând răspunsul la impuls al camerei. Filtrul este descris de

ecuația cu diferențe finite și de răspunsul său în frecvență:

$$y(n) = x(n-m) + gy(n-m), \quad |g| \leq 1 \tag{1.12}$$

$$H(e^{j\omega}) = \frac{1}{e^{j\omega m} - g} \tag{1.13}$$

Aceste mărimi sunt reprezentate grafic în figura 1.5, pentru $m = 500$ și $g = 0.8$. În ecuațiile de mai sus, m reprezintă întârzierea semnalelor dată de un număr de eșantioane iar g reprezintă câștigul subunitar al semnalului de la ieșire.



(a) *Răspunsul la impuls* (b) *Răspunsul în frecvență*

Figura 1.5: Mărimile caracteristice filtrului pieptene

1.3 Rețele Neurale – Concepte generale

Metodele tradiționale de prelucrare a semnalelor discrete în timp, precum filtrarea liniară optimală sau filtrarea adaptivă, se bazează pe anumite presupuneri: sistemele de realizat se pot reprezenta prin modele liniare, semnalele aleatorii sunt de obicei staționare sau pot fi considerate staționare pe o perioadă scurtă de timp, zgomotele ce se suprapun peste semnalele utile sunt considerate semnale staționare etc. Totuși, aceste metode parametrice sunt eficiente doar în anumite situații particulare și duc de obicei la o rezolvare matematică complicată. De asemenea, există multe situații practice în care aceste presupuneri nu sunt adevărate, de aceea este necesară găsirea unei soluții care să combată aceste probleme [9].

De aceea, se vor propune sisteme inteligente care prin învățare automată să își adapteze parametrii în funcție de problema ce trebuie combătută. Un sistem inteligent se definește ca un sistem ce se poate adapta continuu la condițiile date, prin învățarea din experiență. Adaptarea (învățarea) reprezintă modificarea parametrilor interni ai sistemului astfel încât spațiului mărimilor de la intrare să i se asocieze un spațiu al mărimilor de la ieșire. Matematic, acest lucru se traduce prin modelarea funcției f:

$$f : X \to Y$$

unde X și Y reprezintă spațiul de intrare, respectiv de ieșire.

Rețelele neurale (sau sisteme neurale artificiale) reprezintă astfel de sisteme inteligente,ele fiind capabile să estimeze funcții ce fac corespondența între perechea de date intrare – ieșire, sau extrag anumiți parametri caracteristici din spațiul datelor de intrare. Acestea nu reprezintă un concept nou, primele modele de astfel de sisteme fiind puse la punct în anii '40 [10]. În 1958, este implementat primul sistem de clasificare folosit în recunoașterea imaginilor [11]. Domeniul rețelelor neural ea început să se dezvole puternic la sfârșitul anilor '90 și începutul anilor 2000

datorită creșterii puterii de calcul a calculatoarelor. Astăzi, aceste sisteme dau dovadă de performanță sporită în diferite probleme ce presupun obiective matematice precum clasificarea datelor, regresie, optimizare etc.

Conceptul de învățare adâncă (*eng. Deep Learning*) a fost tot mai des întâlnit în ultimii ani în domeniul inteligenței artificiale și numeroase progerse au fost făcute în dezvoltarea rețelelor neurale adânci. Denumirea se datorează faptului că aceste sisteme sunt caracterizate de obicei de un număr foarte mare de parametrii. Arhitectura generală a unei rețele neurale este descrisă în figura 1.6.

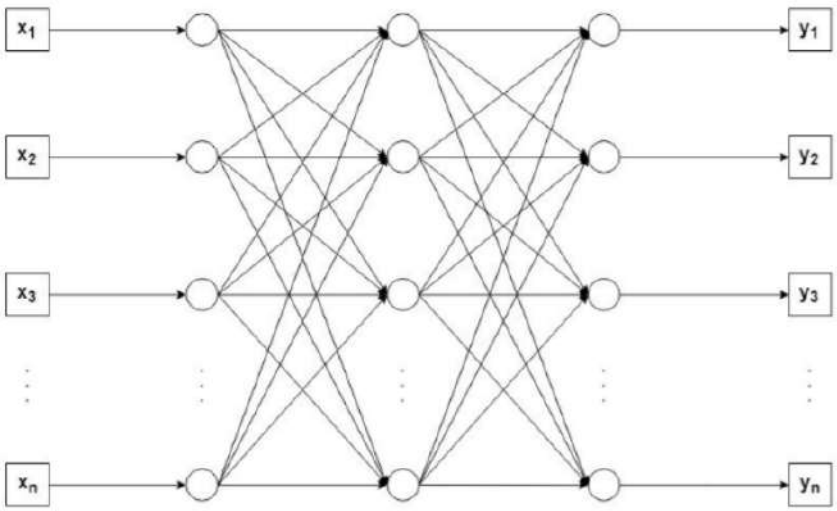


Figura 1.6: Arhitectura generală a unei rețele neurale.

Modul de lucru al acestor sisteme este asemănător cu cel al sistemelor de calcul, unitatea principală de prelucrare a datelor va fi dată de neuronul artifial. Unitățile sunt interconectate între ele, iar prin procesul de învățare ele sunt capabile să își modifice parametrii în scopul rezolvării unei anumite sarcini.

Modalitatea în care se realizează acest lucru este analog modului de funcționare al creierului. Creierul conține neuroni interconectați ce procesează informația, ei fiind capabili să creeze interconexiuni asfel încât să poată generaliza pe baza informației învățate: deci pentru anumite date de intrare se oferă un răspuns corespunzător. Pentru a putea învăța este necesar ca rețeaua să primească suficient de multe exemple pentru a putea generaliza pe un set cât mai extins de date.

Execuția nu este de tip secvențial, rețeaua fiind capabilă să exploreze simultan mai multe ipoteze datorită paralelismului și a interconectării neuronilor artificiali, legați prin intermediul unor ponderi ai căror parametrii se modifică în timpul procesului de adaptare. Putem spune astfel că proiectarea unei rețele neurale constă în definirea caracteristicilor nodurilor (alegerea funcției de activare potrivită), alegerea arhitecturii rețelei (numărul de straturi, noduri și tipul de conexiune) și specificarea algoritmului de optimizare. Acesta din urmă reprezintă un proces iterativ, ce stă la baza mecanismului de antrenare și se bazează pe minimizarea unei funcții de cost.

Principalele operații matematice fundamentale pentru care sunt folosite rețelele neurale sunt [9]:

- Clasificare: semnifică repartizarea într-un set de clase (predefinite la ieșirea rețelei) a vectorilor de intrare în rețea. Această operație poate fi privită ca o asociere de modele, fiind o grupare a mărimilor de la intrare în funcție de mărimile de la ieșire. Clasificarea este folosită în rețele antrenate în mod supervizat și presupune domeniu de ieșire discret;

- Autoasociere (*eng. Clustering*): rețeaua caută asemănări intre mărimile de la intrare și le grupează în funcție de trăsăturile similare pe care le au în comun. Se mai numește și clasificare nesupervizată, rețeaua fiind antrenată în mod nesupervizat;
- Aproximare funcțională (regresie matematică): rețeaua caută o funcție necunoscută pe baza perechii intrare - ieșire astfel încât să se creeze o legătură între aceste date, iar domeniul de ieșire este unul continuu;
- Predicție: modul în care rețeaua funcționează este asemănător cu filtrarea adaptivă, astfel se încearcă estimarea eșantionului la momentul de timp următor pe baza eșantioanelor primite la intrare;
- Îndepărtarea zgomotului: Similar cu sistemele clasice de filtrare a zgomotului, rețeaua încearcă să obțină la ieșire un semnal cu zgomot redus pe baza unui semnal afectat de zgomot de la intrarea rețelei;
- Optimizare: Eroarea dintre ieșirea rețelei și ieșirea dorită este minimă, problema reducându-se la minimizarea sau maximizarea unei funcții de cost.

Modul de prelucrare al datelor de către neuronii artificiali este constituit din 2 operații: una liniară, ce calculează produsul scalar dintre valorile de intrare și cele ale ponderii și produce o singură valoare la ieșire, urmată de o operație neliniară prin care această valoare este prelucrată de o funcție de activare:

$$y_j = f(\sum_{i=0}^{N-1} x_i w_{ij}) = f(\mathbf{x}\mathbf{w}_j^T + b) \tag{1.14}$$

unde $\mathbf{x} = (x_i)_{0 \leq i \leq N-1}$ reprezintă vectorul linie de intrare iar $\mathbf{w}_j = (w_{ij})_{0 \leq i \leq N-1}$ reprezintă vectorul coloană a ponderilor asociate neuronului j. Termenul b reprezintă componenta constantă de la intrarea neuronului, numit termenul *bias*. O rețea neurală va fi formată din mai multe straturi, fiecare având un număr specific de neuroni. Astfel, rețeaua va deveni o comupunere de funcții definite de ecuația (1.11). Performanțele rețelelor sunt strâns legate de modul în care se alege numărul de neuroni al fiecărui strat și de funcțiile de activare folosite. Depinde de problema abordată ce tip de operator de activare se alege. Rolul acestor funcții este de a aduce datele în aceeași gamă controlabilă de valori. Acesta reprezintă un pas important pentru ca rețeaua să reușească să învețe mai ușor exemple cu valori apropiate. De aceea, se preferă ca în etapa preprocesării datelor de antrenare să se facă *normalizarea* lor prin încadrarea valorilor pe o curbă gaussiană:

$$x \to \frac{x - \mu}{\sigma} \tag{1.15}$$

unde μ reprezintă media valorilor de intrare iar σ reprezintă deviația standard a acestora.

Majoritatea operatorilor de activare sunt neliniari, cei mai populari fiind funcția *sigmoid* și funcția *tangentă hiperbolică*. Acestea sunt funcții continue și diferențiabile.

Codomeniul funcției sigmoid are valori cuprinse în intervalul [0,1], deci valorile de intrare negative vor fi duse către 0, iar cele mari vor fi duse către 1. În figura 1.7a este ilustrat graficul acestei funcții cu următoarea formă:

$$\sigma : \mathbb{R} \to [0, 1], \quad \sigma(x) = sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{1.16}$$

Funcția tanh are forma graficului asemănătoare cu cea a funcției sigmoid, însă codomeniul ei este cuprins în intervalul $[-1, 1]$. Valorile negative vor fi duse către -1, iar cele nule vor fi duse

în vecinătatea lui 0. Performanțele sunt mai ridicate în ceea ce privește funcția *tanh*, întrucât gama de valori nu implică pozitivitatea datelor. Graficul funcției este ilustrat în figura 1.7b, iar formula funcției *tanh* este următoarea:

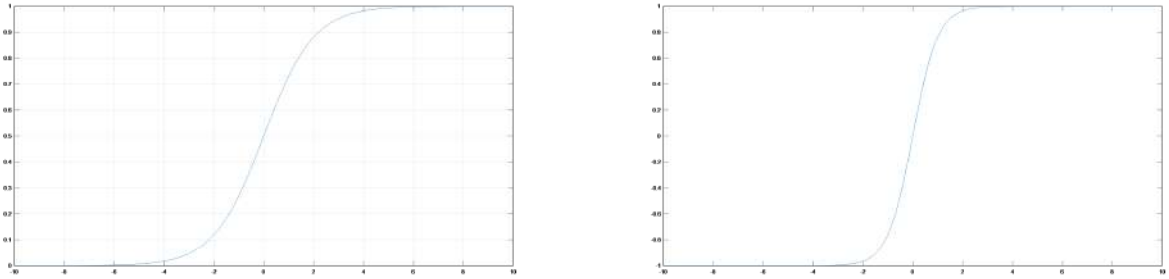
$$f : \mathbb{R} \to [-1, 1], \quad f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{1.17}$$

Un dezavantaj al acestor funcții este dat de faptul că valorile mici sunt mapate în -1 sau 0, iar cele mari se saturează la valoarea 1. Acest fapt împiedică îmbunătățirea performanței rețelei în sensul optimizării parametrilor ponderilor. De aceea, se propune o funcție de activare neliniară ce are codomeniul cuprins în intervalul $[0, +\infty)$: funcția *ReLU* (*eng. Rectified Linear Unit*). Deși problema saturației valorilor mari este rezolvată, această funcție impune pozitivitatea valorilor ca și în cazul funcției *sigmoid*, rețeaua având mai puține grade de libertate pentru a învăța. Funcția *ReLU* este folosită în majoritatea aplicațiilor rețelelor neurale, având performanțe ridicate în probleme de optimizare și de regresie, de aceea această funcție va fi preponderent folosită în această lucrare. Forma graficului este dată în figura 1.8a, iar formula ei este:

$$f : \mathbb{R} \to [0, +\infty), \quad f(x) = max(0, x) \tag{1.18}$$

Există o versiune modificată a funcției *ReLU* numită *Leaky ReLU*. Această lasă să treacă la ieșire valorile negative însă cu un gradient foarte mic, întrucât aceste valori sunt cu mult micșorate. Astfel, gradientul din cadrul procesului de optimizare nu va fi niciodată nul.

$$f : \mathbb{R} \to \mathbb{R}, f(x) = \begin{cases} x, & \text{dacă} \quad x \geq 0 \\ 0.01x, & \text{dacă} \quad x < 0 \end{cases} \tag{1.19}$$

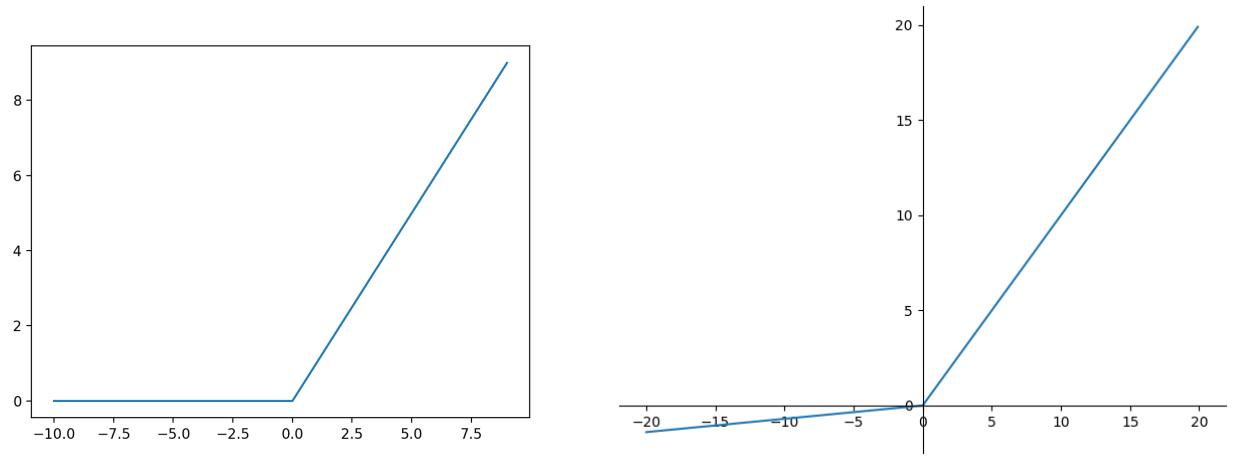


(a) *Sigmoid* (b) *Tanh*

Figura 1.7: Graficele funcțiilor *sigmoid* și *tanh*

Rolul funcțiilor prezentate este de a introduce neliniaritate în sistem, un pas important în învățarea anumitor funcții complexe. Fiecare strat va avea propria funcție de activare, toți neuronii folosind același operator.

Procesul de optimizare al parametrilor rețelei implică actualizarea lor pe baza derivatei (gradientului) funcției de cost în raport cu ponderile straturilor de-a lungul mai multor iterații. De aceea, este important să fie asigurată derivabilitatea operatorilor de activare folosiți. La fiecare iterație se caută actualizarea parametrilor în sensul minimizării funcției de cost după ponderi și după termenul *bias*: $\frac{\partial \mathcal{L}}{\partial w}$, $\frac{\partial \mathcal{L}}{\partial b}$, unde $\mathcal{L}$ reprezintă funcția de cost (*eng. loss function*, de aici și notația). Tehnica bazată pe micșorarea gradientului (*gradient descent*) funcției de cost este cea mai folosită în cadrul optimizării rețelelor. Mai multe detalii despre procesul de optimizare vor fi date în Capitolul 2.



(a) *ReLU* (b) *Leaky ReLU*

Figura 1.8: Graficele funcțiilor *ReLU* și *Leaky ReLU*

Totuși, pentru anumite funcții de activare, cum ar fi *sigmoid* sau *tanh*, gradientul funcției de cost tinde spre valoarea 0 în cazul valorilor foarte mari sau foarte mici, procesul de optimizare devenind ineficient (fenomen denumit în limba engleză *vanishing gradient descent*). De aceea, un alt avantaj al funcției *ReLU* este că elimină această problemă în cazul valorilor mari.

În general, pentru a eficientiza procesul de optimizare, se folosesc loturi (*eng. batch*) de date din numărul total de exemple pentru antrenare. Acest lot este trecut prin rețea de la intrare la ieșire, iar procesul propriu zis de optimizare are loc prin parcurgerea inversă a rețelei (*eng. backpropagation*). Există diferite tipuri de optimizatori pentru calculul gradientului, precum optimizatorul *SGD* (*eng. Stochastic Gradient Descent*) [12] sau optimizatorul *Adam* (*eng. Adaptive Movement Estimation*) [13]. Optimizatorul *SGD* folosește un singur exemplu din lotul curent ales aleator pentru a calcula gradientul întregului lot, astfel micșorând timpul de calcul. Deoarece *SGD* are probleme în găsirea minimului global pentru funcții de cost ce au o pantă mult mai abruptă pe o anumită dimensiune decât pe celelalte, caz în care optimizatorul va oscila pe panta respectivă deci gradientul va fi divergent, este nevoie de introducerea unui impuls (*eng. momentum*) pentru a accelera gradientul pe dimensiunea respectivă și a reduce oscilațiile. Acesta se bazează pe calculul gradientului bazat pe valori anterioare pentru actualizarea parametrilor. Cea mai folosită versiune de astfel de optimizator este *Adam*, ce folosește atât valori anterioare ale parametrului calculat, cât și valori precedente ale gradientului.

Există numeroase tipuri de funcții de cost, alegerea acestora făcându-se în funcție de problema abordată de rețea. Câteva exemple vor fi prezentate în cele ce urmează:

- **MSE - Mean Squared Error - Eroarea pătratică medie**: Se folosește cel mai des în aplicații de regresie liniară, se caută minimizarea erorii pătratice medii dintre datele de la ieșirea rețelei și datele dorite. Mai este numită și puterea erorii. Aceasta va reprezenta funcția de cost folosită în cadrul procesului de optimizare a rețelelor propuse în această lucrare:

$$\mathcal{L} = \frac{1}{N}\sum_{i=0}^{N-1}(\hat{y}^{(i)} - y^{(i)})^2 \tag{1.20}$$

  unde N este numărul de exemple, iar $y^{(i)}$ și $\hat{y}^{(i)}$ reprezintă valoarea ideală de ieșire, respectiv valoarea rețelei de ieșire pentru al i-lea exemplu. Diferența $\hat{y}^{(i)} - y^{(i)}$ se numește reziduu, iar aceste valori trebuie minimizate.

- **MAE - Mean Absolute Error - Eroarea medie absolută**: Funcție asemănătoare

cu MSE, însă mai robustă la deviații extreme față de media valorilor (*eng. outliers*):

$$\mathcal{L} = \frac{1}{N}\sum_{i=0}^{N-1}|\hat{y}^{(i)} - y^{(i)}| \tag{1.21}$$

- **RMSE - Root Mean Squared Error - Valoarea eficace medie pătratică**: Radicalul funcției MSE, reprezintă de fapt deviația standard a reziduurilor:

$$\mathcal{L} = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1}(\hat{y}^{(i)} - y^{(i)})^2} \tag{1.22}$$

- **CE - Cross Entropy**: Funcție folosită în principal în modelele de clasificare binară, însă există o variantă de clasificare pentru mai multe clase (*MCCE - eng. Multi-Class Cross-Entropy*):

$$\mathcal{L} = -\frac{1}{N}\sum_{i=0}^{N-1}(y^{(i)}log(\hat{y}^{(i)}) + (1 - y^{(i)})log(1 - \hat{y}^{(i)})) \tag{1.23}$$

În funcție de modul de învățare al rețelelor, există 3 mari cateogrii:

1. **Învățare supervizată**: Sistemului îi este furnizat un set de perechi intrare–ieșire cu ajutorul căruia se calculează eroarea în funcție de rezultatul real obținut și cel dorit (învățare cu corectarea erorii). Se minimizează această eroare prin ajustarea valorilor ponderilor.

2. **Învățare nesupervizată**: rețeaua extrage singură anumite caracteristici importante ale datelor de la intrare și formează reprezentări interne ale acestora. În acest caz, rețelele nu beneficiază de date de ieșire dorite pentru a evalua performanța, deci nu au în timpul antrenării informații despre ce înseamnă o abordare corectă sau greșită a problemei. Conceptul abordat de către rețea este cel de *învățare competitivă*: sunt modificate doar ponderile aferente neuronului care obține cele mai bune performanțe comparativ cu ceilalți neuroni, restul conexiunilor nefiind afectate

3. **Învățare mixtă**: abordează ambele tipuri de învățări menționate anterior: o parte din ponderi sunt determinate prin intermediul unei învățări supervizate, iar restul sunt obținute pe baza unei învățări nesupervizate. În această situație, rețeaua nu beneficiază de semnalul dorit, ci de un semnal ce oferă o informație calitativă asupra funcționării sistemului (informație binară, de tipul răspuns corect / greșit), astfel sistemul este încurajat să producă acțiunea care duce la un rezultat corect.

Scopul rețelelor neurale este să generalizeze pe un set extins de date cu trăsături comune setului de antrenare. Se dorește evitarea fenomenului de *supraînvățare* (*eng. overfit*), prin care rețeaua învață foarte bine pe un anumit set de date de intrare, dar performanțele ei folosind un set de date extins sunt mult mai slabe decât la antrenare. Rețeaua nu învață anumite trăsături reale dintre datele de intrare și cele de ieșire aferente, ci doar realizează legături între datele setului de antrenare dat, nereușind să generalizeze pe un set de date extins. Fenomenul de *overfitting* este detaliat în figura 1.9. Există, de asemenea, și fenomenul opus: *underfitting*. Rețeaua nu este destul de complexă pentru antrenarea anumitor date, rezultând în erori atât pe setul de antrenare, cât și pe cel de testare. În general, rezolvarea acestui fenomen se rezumă la

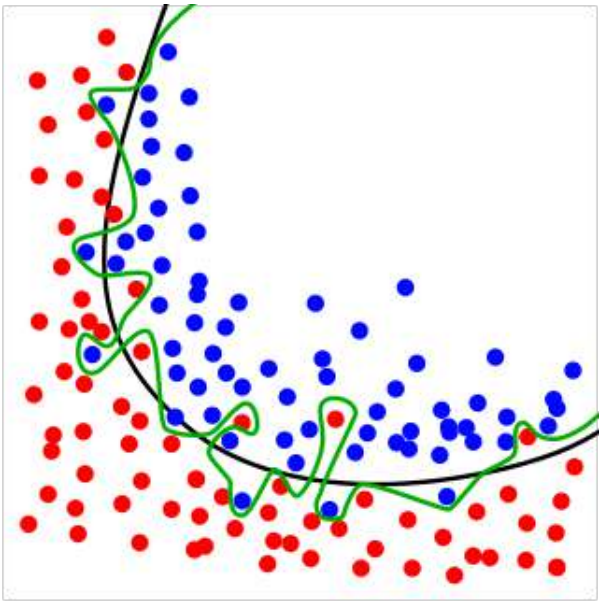


Figura 1.9: Fenomenul de supraînvățare este ilustrat de către încadrarea făcută de linia verde.

mărirea complexității sistemului, însă pentru a evita fenomenul de supraînvățare este necesară urmarea mai multor metode des întâlnite în domeniul de învățare artificială: mărirea setului de antrenare folosind mai multe date, adăugarea straturilor de *dropout* (un anumit procent din parametrii unei ponderi sunt egalați cu 0), memorarea ponderilor aferente momentului în care funcția de cost a atins un minim iar apoi valoarea acestia a crescut (*eng. early stopping*) etc.

Capitolul 2
Soluții propuse

2.1 Deverberarea semnalelor

În ultimii ani, reducerea reverberației din încăperi a devenit o problemă din ce în ce mai dezbatută de către diferite comunități de cercetare. Mulți oameni de știință au sugerat algoritmi pentru dereverberație care sunt adecvați pentru dispozitive *hands-free*, sisteme de recunoaștere automată a vorbirii și aparate auditive digitale.

Pentru evaluarea algoritmilor de dereverberație, se folosesc adesea răspunsurile la impuls din diferite încăperi ce au fost generate artificial pe baza parametrilor mediului acustic sau chiar au fost măsurate în încăperi reale [14].

Pentru evidențierea efectului de reverberație se vor propune 2 seturi de date:

1. Baza de date propusă pentru evaluarea experimentelor este alcătuită din melodii monofonice, ceea ce presupune că o singură notă muzicală este cântată la un anumit moment de timp. Melodiile sunt cântate la orgă electrică, fiind înregistrate în format `.wav`, ele fiind disponibile online[1]. Baza de date constă în 100 de exerciții muzicale și melodii scurte având o durată totală de o oră și 17 minute.

   Deoarece aceste semnale sunt deja în format digital, nu este nevoie de conversia analog-digital. Frecvența de eșantionare folosită pentru citirea și scrierea semnalelor este $f_s = 44100Hz$, frecvență standard folosită în domeniul audio.

   Întrucât baza de date conține semnale audio cu două canale (stereo), a fost făcută tranziția de la semnal stereo la semnal mono prin medierea valorilor celor două canale pentru a lucra mai ușor. Următorul pas constă în normarea amplitudinii semnalelor în intervalul $[-1, 1]$ pentru a aduce aceste valori în aceeași gamă de valori. Acest lucru se realizează prin împărțirea amplitudinilor la maximul valorii absolute ale acestora: $s(k) = \frac{s(k)}{max\{|s(k)|\}}$.

2. Raspunsurile la impuls care modelează reverberația din diferite camere, descrise în [14], sunt disponibile online, ele fiind selectate dintr-o bază de date publică[2]. Aceste semnale sunt clasificate atât în funcție poziționarea microfoanelor față de sursa sonoră, anume în funcție de distanță și unghi.

[1] `https://speed.pub.ro/downloads/music-datasets/`

[2] `https://www.iks.rwth-aachen.de/en/research/tools-downloads/databases/aachen-impulse-response-database`

Pentru modelarea efectului reverberației suprapus peste un semnal audio, vom presupune inițial un sistem liniar și invariant în timp (SLIT), astfel răspunsul la impuls al camerei poate fi utilizat pentru a descrie complet proprietățile acustice ale acesteia din punct de vedere al propagării sunetului și al modului în care acesta se reflectă pentru o anumită configurație sursă sonoră - microfon. În general, datele sunt achiziționate folosind J microfoane, fiecare microfon achiziționând un anumit răspuns la impuls al camerei respective: $(h_j)_{(1\leq j\leq J)}$ [14].

Dacă considerăm semnalul audio util $s(k)$ și zgomotul captat de fiecare microfon $n_j(k)$, atunci ecuația matematică ce descrie efectul reverberației este:

$$x_j(k) = s(k) * h_j(k) + n_j(k), \quad j = 1, \ldots, J \tag{2.1}$$

Semnalul x_j înregistrat cu al j-lea microfon este creat sintetic prin simpla convoluție dintre semnalul muzical s și răspunsul la impuls h_j. Întrucât zgomotul suprapus peste semnale este presupus a fi zgomot alb Gaussian de medie $\mu = 0$, puterea acestuia va fi egală cu varianța sa σ^2. Modelarea zgomotului se va face în funcție de raportul semnal-zgomot (*eng. SNR - Signal to Noise Ratio*), dat de formula:

$$SNR[dB] = 10lg\frac{P_{semnal\ util}}{P_{zgomot}} \tag{2.2}$$

Astfel, semnalele zgomotoase create artificial au fost modelate atribuind valori aleatorii varianței zgomotului astfel încât raportul SNR să fie cuprins între 5 dB și 30 dB. În figura 2.1 este oferită o comparație între semnalul reverberat și semnalul corupt de zgomot.

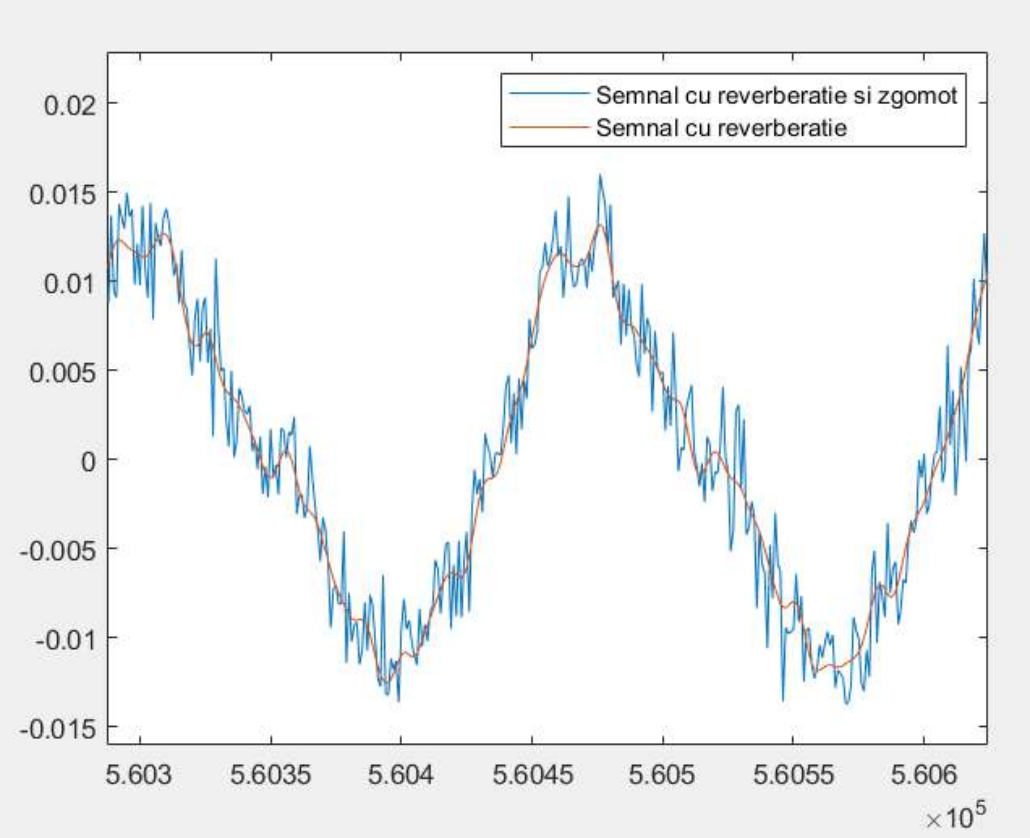


Figura 2.1: Semnal reverberat și semnal corupt de zgomot.

O abordare variațională pentru această problemă de recuperare a semnalului original (prin asumarea aceluiași nivel de zgomot pentru fiecare microfon) este dată de minimizarea următoarei funcții obiectiv:

$$\underset{\tilde{s}}{\text{minimize}}\ \frac{1}{2}|H_j\tilde{S} - X_j|^2 + \varphi(\tilde{s}),$$

unde φ reprezintă funcția de cost, caracterizată prin norma ℓ_1 (este absolut sumabilă):

$$\varphi(\tilde{s}) = \lambda \sum_k |\tilde{s}(k)|$$

cu λ paramterul de regularizare ce trebuie setat. În ecuațiile de mai sus, H_j reprezintă funcția de transfer a camerei pentru al j-lea microfon, iar $\tilde{S}$ și X_j reprezintă transformata Fourier a semnalului deconvoluționat $\tilde{s}$, respectiv a semnalului înregistrat x_j

Problema poate fi rezolvată de algoritmul de gradient proximal. A n-a iterație a acestui algoritm se citește:

$$s_{n+1} = prox_{\gamma\lambda}(s_n - \mathcal{F}^{-1}\{\gamma \sum_{j=1}^{J} H_j^*(H_j S_n - X_j)\}), \tag{2.3}$$

unde H_j^* reprezintă imaginea complex conjugată a funcției de transfer, S_n reprezintă transformata Fourier a semnalului recuperat s_n la cea de-a n-a iterație, iar $\mathcal{F}^{-1}$ reprezintă operația de transformare Fourier inversă, iar γ este dat de ecuația:

$$\gamma = 1.9/(\sum_{j=1}^{J} \max |H_j|^2),$$

Argumentul operatorului proximal reprezintă operația propriu-zisă de deconvoluție prin intermediul căreia se dorește îndepărtarea efectului reverberației, iar operația de *soft-thresholding* cu pasul $\gamma\lambda$ reprezintă o metodă cunoscută în literatură de minimizare a efectului zgomotului aditiv suprapus peste semnale [15].

Semnalele deverberate folosind algoritmul descris vor constitui baza de date de lucru pentru sistemele ce elimină zgomotul.

2.2 Eliminarea zgomotului

Problema zgomotului aditiv a reprezentat dintotdeauna o problemă majoră în domeniul prelucrării semnalelor, cu precădere în domeniul semnalelor audio. Există numeroase metode ce propun eliminarea acestui fenomen nedorit, bazate atât pe metode clasice de filtrare a zgomotului (filtru Wiener, filtre adaptive etc.) cât și pe metode ce înglobează concepte de inteligență artificială.

În ultimii ani, au fost făcute numeroase demersuri în ceea ce privește eliminarea zgomotului suprapus peste semnale vocale folosind metode bazate pe machine learning sau învățare adâncă [16], [17]. Spre exemplu, acestea pot aduce îmbunătățiri în cazul asistenților virtuali inteligenți, precum Siri, dezvoltat de compania Apple. Se crede ca aceste îmbunătățiri pot fi aduse și în cazul semnalelor muzicale (spre exemplu, eliminarea zgomotului din înregistrări vechi [18]). Totuși, pentru a putea analiza semnalele zgomotoase, este necesară o preprocesare a lor.

Pașii inițiali de preprocesare au fost descriși în secțiunea precedentă: tranziția stereo - mono și normarea semnalelor. Operația de normare a fost strict necesară întrucât o rețea neurală va învăța mai ușor un set de date cu valori în aceeași gamă (în acest caz, intervalul [-1,1]), astfel ea va diferenția mai ușor între anumite trăsături ale semnalelor, ele fiind de asemenea în aceeași gamă de valori.

Semnalele muzicale sunt semnale aperiodice, nestaționare, cu parametrii statistici variabili,

fiind greu de analizat pe întreaga lor durată. De aceea, vom prespune că, pentru intervale *scurte* de timp, semnalul va avea un caracter cvasiperiodic și cvasistaționar. Astfel, următorii pași de preprocesare constau în segmentarea melodiilor în intervale scurte folosind diferite ferestre de analiză.

2.2.0.1 Analiza semnalelor prin ferestruire

A extrage dintr-un semnal de durată mare o anumită porțiune de o anumită durată de timp este echivalent cu a înmulți semnalul cu o fereastră de analiză *dreptunghiulară* notată $w_R(n)$, alcătuită din N impulsuri-unitate, evidențiată în figura 3.1. Această operație poartă numele de *tehnica ferestruirii*, des utilizată în multe aplicații de prelucrare digitală a semnalelor:

$$w_R(n) = \begin{cases} 1, & \text{dacă } 0 \leq n \leq N-1 \\ 0, & \text{în rest} \end{cases} \tag{2.4}$$

Astfel, prin înmulțirea semnalului $x(n)$ cu această fereastră, semnalul de analizat devine:

$$x_N(n) = x(n)w_R(n) = \begin{cases} 1, & \text{dacă } 0 \leq n \leq N-1 \\ 0, & \text{în rest} \end{cases} \tag{2.5}$$

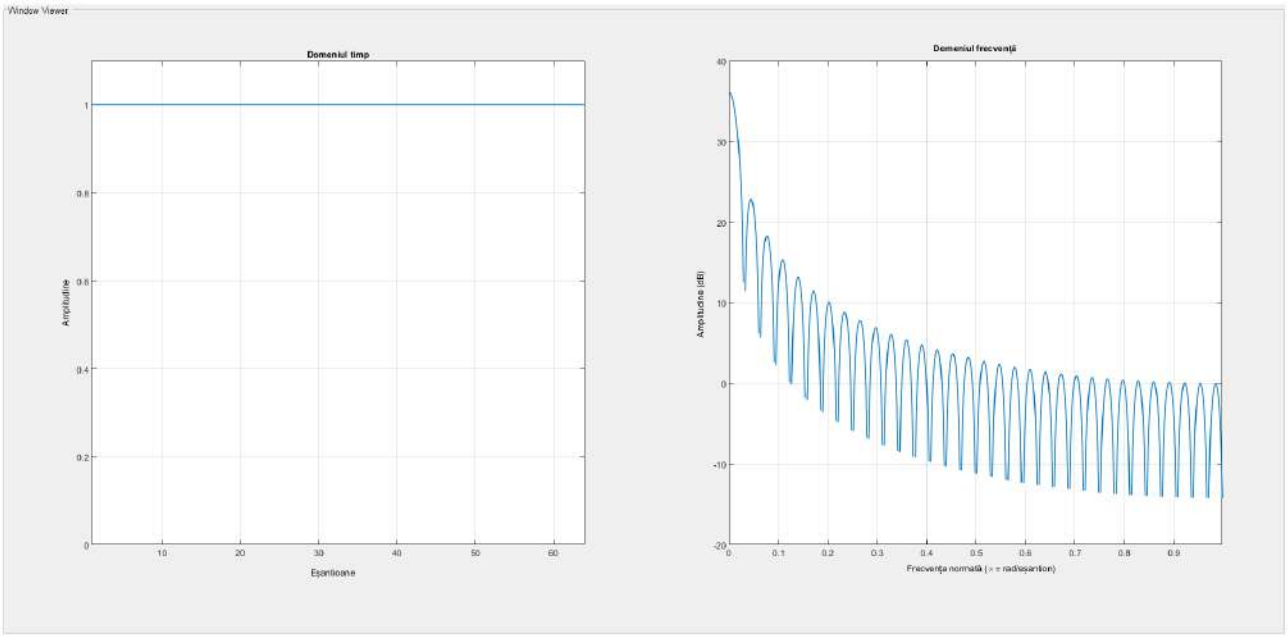


Figura 2.2: Fereastră dreptunghiulară.

În cazul transformării Fourier, produsul a două semnale în timp corespunde unei operații de convoluție în domeniul frecvență:

$$x_N(n) = x(n)w_R(n) \leftrightarrow X_N(e^{j\omega}) = X(e^{j\omega}) * W_R(e^{j\omega}), \tag{2.6}$$

unde $X_N(e^{j\omega})$, $X(e^{j\omega})$, $W_R(e^{j\omega})$ reprezintă transformatele Fourier ale semnalelor $x_N(n)$, $x(n)$ și respectiv $w_R(n)$. $W_R(e^{j\omega})$ mai poartă denumirea de fereastră spectrală, denumire adoptată pentru transformata Fourier a oricărui tip de fereastră de analiză. În cazul de față, spectrul $W_R(e^{j\omega})$ are forma unei funcții *sinc*, cu un lob central de lungime $2 \times \frac{2\pi}{N}$ și lobi laterali ce vor descrește progresiv în amplitudine către extremitățile intervalului unei perioade.

Cu alte cuvinte, spectrul real $X_N(e^{j\omega})$ va fi o convoluție dintre spectrul ideal $X(e^{J\omega})$ și o funcție de tip sinc. Acest lucru determină o deformare a spectrului ideal, însoțită de apariția unor ondulații în spectrul semnalului $x_N(n)$. Acest fenomen poartă numele de *dispersie* sau *împrăștiere* spectrală (*eng. leakage*).

Deoarece spectrul real $X_N(e^{j\omega})$ va fi eșantionat în frecvență pentru a obține coeficienții Fourier $X_N(k)$, unde k este indicele coeficientului discret, $k \in \{1, \ldots, N_{FFT}\}$, vor apărea erori în forma acestui spectru discret și în consecință în forma semnalului refăcut din eșantioane. Aceste erori sunt modificări ale amplitudinilor coeficienților spectrali $X(k)$, numite erori de amplitudine (*eng. picket-fence*), datorate faptului că anumite componente spectrale ale semnlului de interes pot să nu fie cele corecte, ele depinzând de poziția unde apar eșantioanele spectrului discret față de forma funcției de tip sinc. Alte erori posibile constau în apariția unor frecvențe false (din cauza lobilor laterali din spectrul ferestrei dreptunghiulare), sau pierderea unei informații de frecvență (nu mai pot fi puse in evidență componente de frecvențe foarte apropiate din spectrul unui semnal complex).

Astfel, s-a arătat că înmulțirea semnalului cu o fereastră de analiză dreptunghiulară are ca principale efecte în domeniul frecvență o dispersare a componentelor spectrale ale semnalului în benzile lor laterale și o modificare a amplitudinilor acestor componente. Aceste modificări pot duc la apariția în domeniul timp, având în vedere periodicitatea transformatei Fourier inverse, la discontinuități la capetele intervalului de analiză.

Soluția diminuării acestor efecte constă în utilizarea unor funcții fereastră cu o curbă mai "netedă" și cu un spectru care să permită micșorarea erorilor prezentate. Aceste funcții poartă denumirea de ferestre de ponderare (*eng. window weighting function*). Ele realizează o trunchiere mai puțin abruptă, ducând spre zero semnalul la capetele intervalului de N eșantioane. Spectrele acestor ferestre sunt asemănătoare cu cel al ferestrei dreptunghiulare, fiind compuse dintr-un lob central ce conține cea mai mare parte din energia semnalului și din lobi laterali cu amplitduini descrescătoare către capetele intervalului.

Astfel, aplicarea unei funcții fereastră (alta decât cea dreptunghiulară) asupra unui semnal determină, în timp, pierderea unei anumite cantități de informație spre capetele intervalului și duce la obținerea unui spectru cu vârfuri mai largi și de amplitudine mai mică decât în cazul ferestrei dreptunghiulare, apărând riscul ca anumite componente spectrale utile să fie "înghițite" de lobul central, dar se atenuează mult lobii laterali, deci se reduce semnificativ fenomenul de dispersie a spectrului [5].

În această lucrare, se va propune fereastra de analiză *Hanning* pentru analiza semnalelor în frecvență, astfel încât se vor reduce erorile de împrăștiere spectrală și de amplitudine. Această fereastră este cunoscută și ca fereastră *Hann*, pentru a nu se confunda cu fereastra de analiză *Hamming*. Fereastra *Hanning* este ilustrată în figura 2.3, alături de reprezentarea sa în domeniul frecvență, și are următoarea formă matematică:

$$w_{HAN}(n) = \begin{cases} \frac{1}{2} - \frac{1}{2}\cos\left(\frac{2\pi n}{N-1}\right), & \text{dacă } 0 \leq n \leq N-1 \\ 0, & \text{în rest} \end{cases} \tag{2.7}$$

2.2.1 Metode clasice de eliminare a zgomotului

2.2.1.1 Filtrarea Wiener

Multe probleme practice ce constau în reducerea zgomotului din semnale constau în estimarea valorilor semnalului util utilizând un set de date ce provin fie din semnalul corupt de zgomot, fie de la alte semnale ce au o legătură cu semnalul de interes.

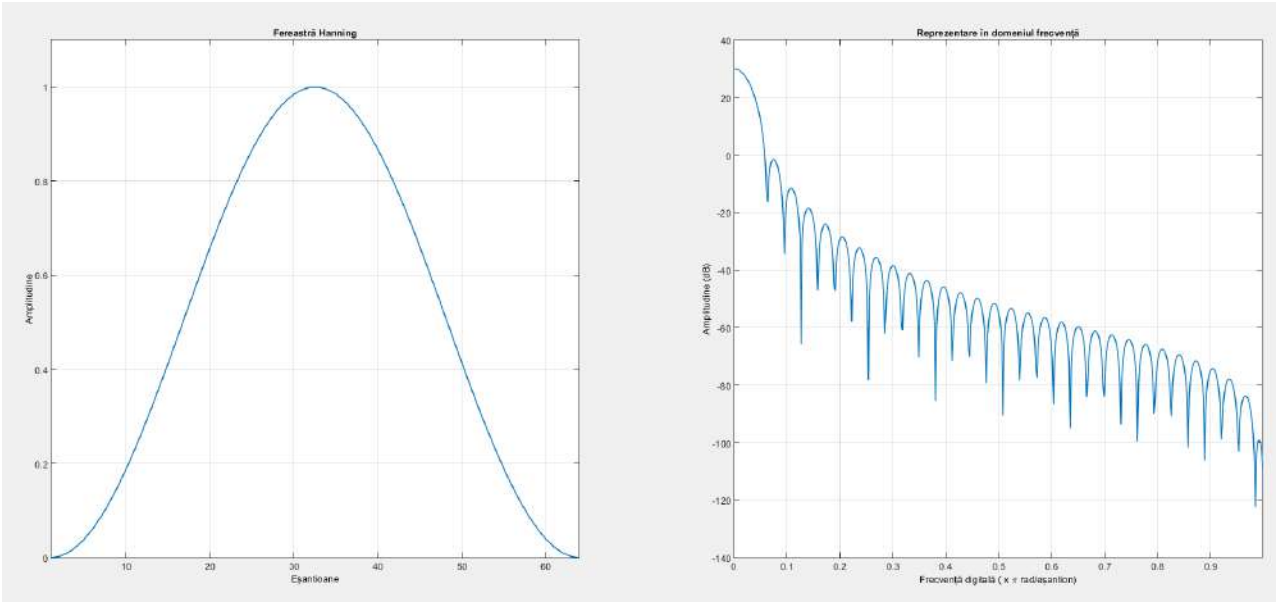


Figura 2.3: Fereastră Hanning.

Problema eliminării zgomotului aditiv se formulează în felul următor: Considerăm un semnal util, $d(n)$, care poate fi observat doar în prezența zgomotului, $x(n) = d(n) + v(n)$. Se presupune că semnalele $d(n)$ și $v(n)$ sunt semnale aleatorii, staționare în sens larg. Trebuie conceput un filtru care să ofere un estimat $y(n) = \hat{d}(n)$ pentru semnalul util astfel încât eroarea pătratică medie $[d(n) - \hat{d}(n)]^2$ să fie minimizată. Filtrul optimal care să rezolve această problemă se consideră a fi *filtrul Wiener*. Acest filtru este, în general, un filtru de tip *FIR* (*eng. Finite Impulse Response*), pentru stabilitatea nativă a acestor tipuri de filtre și pentru simplitatea calculelor determinării coeficienților săi [9]. Schema unui astfel de filtru este dată în figura 2.4.

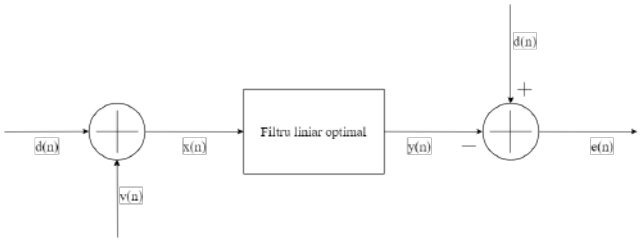


Figura 2.4: Principiul estimării liniare al filtrului Wiener.

Dacă presupuem un filtru cu L coeficienți, ecuația care stă la baza determinării coeficienților $h_{opt}(k)$ ai filtrului Wiener optimal este dată de versiunea discretă matriceală a ecuațiilor *Wiener-Hopf*:

$$\begin{bmatrix} R_{xx}(0) & R_{xx}(1) & R_{xx}(2) & \dots & R_{xx}(L-1) \\ R_{xx}(1) & R_{xx}(0) & R_{xx}(1) & \dots & R_{xx}(L-2) \\ R_{xx}(2) & R_{xx}(1) & R_{xx}(0) & \dots & R_{xx}(L-3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ R_{xx}(L-1) & R_{xx}(L-2) & R_{xx}(L-3) & \dots & R_{xx}(0) \end{bmatrix} \begin{bmatrix} h_{opt}(0) \\ h_{opt}(1) \\ h_{opt}(2) \\ \vdots \\ h_{opt}(L-1) \end{bmatrix} = \begin{bmatrix} R_{dx}(0) \\ R_{dx}(1) \\ R_{dx}(2) \\ \vdots \\ R_{dx}(L-1) \end{bmatrix}$$

R_{xx} reprezintă funcția de autocorelație a semnalului $x(n)$, cu următoarea formulă:

$$R_{xx}(l) = \frac{1}{L}\sum_{n=0}^{L-1} x(n)x(n+l), \quad l \in \{-L+1, \ldots, 0, \ldots, L-1\} \tag{2.8}$$

R_{dx} reprezintă intercorelația dintre semnalul util $d(n)$ și cel observat $x(n)$:

$$R_{dx}(l) = \frac{1}{L}\sum_{n=0}^{L-1} d(n)x(n+l), \quad l \in \{-NL+1, \ldots, 0, \ldots, L-1\} \tag{2.9}$$

Intercorelația este o măsură a similitudinii dintre două semnale în funcție de decalajul aplicat unuia dintre ele. În cazul unui singur semnal de analizat, această funcție devine practic funcția de autocorelație.

Pentru preprocesare, am împărțit semnalele în ferestre de câte 68ms, reprezentând 3000 de eșantioane, folosind fereastra de analiză dreptunghiulară. În acest sens, am presupus că pe acest interval de timp semnalele sunt cvasistaționare, astfel încât să fie respectată ipoteza filtrului Wiener. Mai mult, am presupus că zgomotul aditiv și semnalul util sunt *necorelate*. Fiind un filtru FIR, este necesar un ordin mare pentru a avea performanțe bune, de aceea s-a ales ordinul $L = 100$. Schema de analiză folosind filtrul Wiener este dată în figura 2.5.

2.2.1.2 Filtrarea adaptivă

Scopul principal al unui filtru adaptiv este acela de a extrage sau îmbunătăți anumite informații utile dintr-un semnal oarecare, de aceea pentru implementarea unui astfel de sistem se va utiliza un filtru digital clasic căruia i se adaugă un algoritm adaptiv iterativ, prin care coeficienții filtrului sunt actualizați permanent în funcție de evoluția semnalelor disponibile și de un criteriu de performanță ales în funcție de situația în care este utliziat filtrul.

Ca și în cazul filtrului Wiener, criterul des utilizat este acela de a minimiza eroarea dintre semnalul de ieșire al filtrului și un semnal de referință dat. Schema de funcționare a unui astfel de filtru este dată în figura 2.6. Semnalul $d(n)$ este semnalul util, de obicei corupt de zgomotul aditiv $v(n)$, iar $x(n) = d(n) + v(n)$. Se dorește minimizarea erorii $e(n) = d(n) - y(n)$, acest lucru fiind realizat de algoritmul adaptiv. La fiecare iterație, semnalul de eroare este prelucrat de algoritm, în sensul găsirii coeficienților optimi ai filtrului adaptiv. Pentru simplitate, cele mai folosite configurații pentru filtere adaptive sunt filtrele FIR.

Cel mai eficient și utilizat algoritm adaptiv este algoritmul *LMS* (*eng. Least Mean Squares*), criteriul de performanță ales fiind dat de minimizarea erorii pătratice medii. Astfel, presupunând un filtru cu L coeficienți și notând coeficienții $h_k(n)$ la momentul n, cu $k \in \{0, 1, \ldots, L-1\}$, se caută minimzarea funcției de cost:

$$J = \frac{1}{N}\sum_{n=0}^{N-1} e^2(n) = \frac{1}{N}\sum_{n=0}^{N-1}(d(n) - y(n))^2 \tag{2.10}$$

Semnalul de la ieșirea filtrului este dat de simpla convoluție dintre semnalul zgomotos și răspunsul la impuls: $y(n) = h(n) * x(n)$, deci funcția de cost depinde direct de coeficienții lui $h(n)$. Astfel, minimizarea erorii implică calcularea Jacobienei la fiecare iterație în funcție de fiecare coeficient $h_k(n)$ în cadrul unui algoritm de gradient descendent. Actualizarea la momentul de timp n se citește după cum urmează:

$$h_k(n+1) = h_k(n) - \mu\frac{\partial J}{\partial h_k(n)} \tag{2.11}$$

Parametrul $\mu \in (0, 1)$ reprezintă rata de adaptare (sau învățare) a algoritmului și controlează

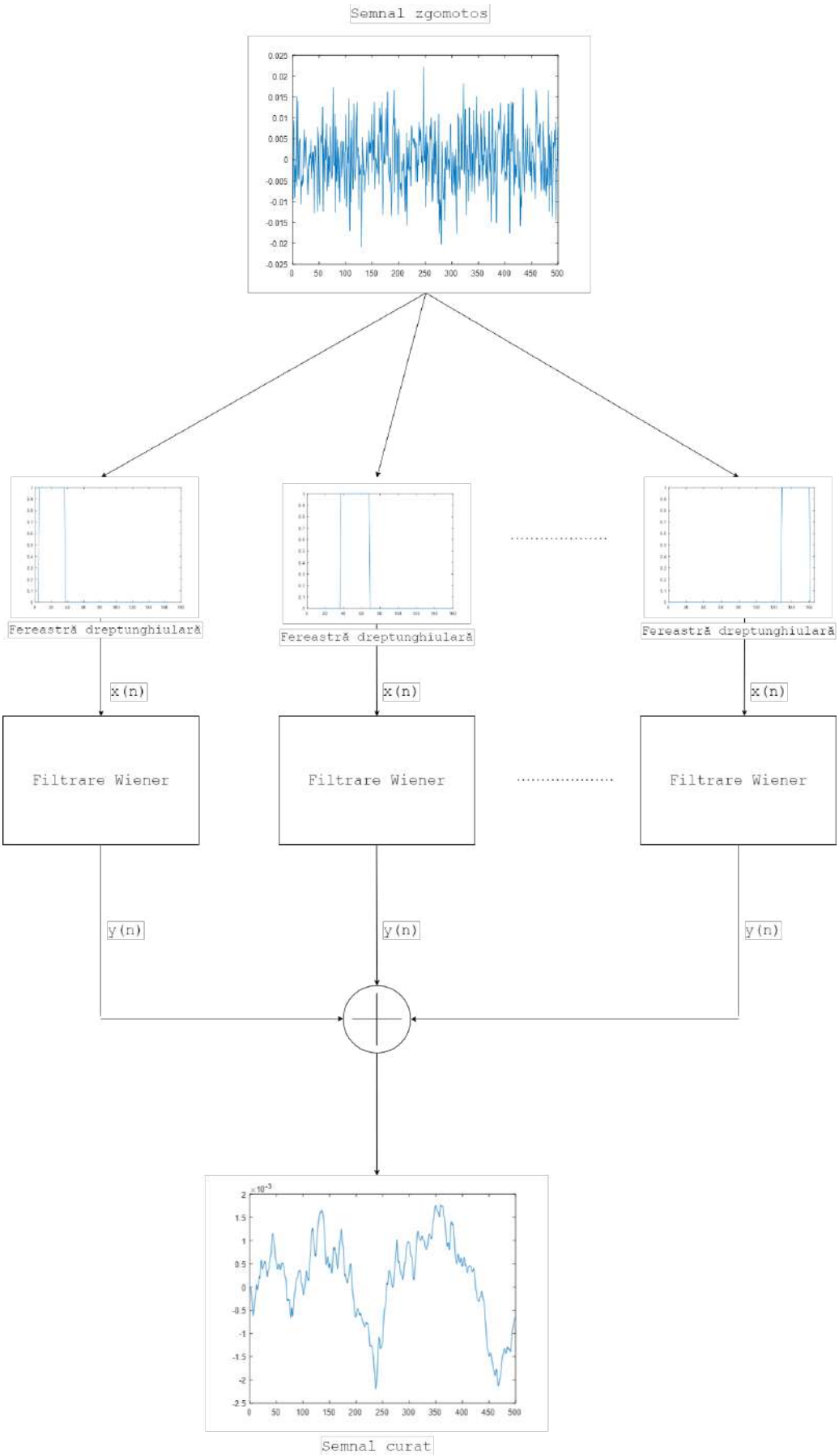


Figura 2.5: Schema de prelucrare folosind filtrul Wiener.

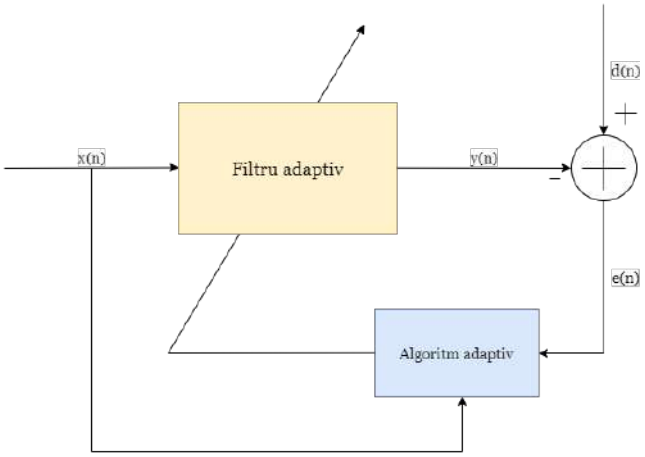


Figura 2.6: Principiul de funcționare al filtrării adaptive.

stabilitatea și viteză de convergență.

Există o versiune îmbunătățită a acestui algoritm, numită algoritmul *NLMS* (*eng. Normalised Least Mean Squares*) ce oferă un compormis mai bun între viteza de convergență, dezadaptarea minimului erorii pătratice medii și complexitatea calculelor. În acest caz, rata de învățare se normalizează cu energia semnalului de intrare, astfel:

$$\mu(n) = \frac{\tilde{\mu}}{2(\delta + \sum_{i=0}^{L-1} x^2(n-i))} \tag{2.12}$$

unde δ este o constantă aleasă așa încât numitorul fracției să nu devină 0 în cazul semnalelor cu energie foarte mică, iar $\tilde{\mu}$ este rata de învățare inițială. O versiune ușor modificată a algoritmului $NLMS$ se obține prin normarea ratei de învățare la puterea semnalului:

$$\mu(n) = \frac{\tilde{\mu}}{\delta + L[(1-\alpha)\sigma_x^2(n-1) + \alpha x^2(n)]} \tag{2.13}$$

Aici, σ_x^2 reprezintă puterea medie a semnalului $x(n)$. Normalizarea se face ținând cont atât de puterea medie a semnalului pana la momentul n, deci calculul se face folosind primele $n-1$ eșantioane și de puterea instantanee a acestuia la momentul n, iar α poartă numele de factor de corecție, ales deseori pentru simplitatea calculelor $\alpha = 0$. În general, parametrii $\tilde{\mu}, \delta$ și α sunt aleși empiric pentru a determina cea mai bună performanță a sistemului [9]. În această lucrare se va folosi această versiune a algoritmului *NLMS*.

Configurația în care vom aplica această filtrare este cea de *identificare de sistem*. Filtrul adaptiv este utilizat pentru a furniza un model liniar care să aproximeze cât mai fidel comportamentul sistemului necunoscut. În cazul de față, se dorește a determina un sistem care să îndeparteze cât mai bine zgomotul aditiv din semnalele utile. Configurația sistemului este dată în figura 2.7.

Procesarea semnalelor folosind filtarea adaptivă este aceeași cu cea folosită în cazul filtrării Wiener: segmentarea semnalului în ferestre de durată foarte scurtă de 86ms, filtrarea fiecărui segment și reconstrucția semnalului. Filtru adaptiv este un filtru FIR, de aceea am ales ordinul $L = 200$ pentru prelucrarea semnalului pentru a ne apropia de performanțele filtrării Wiener.

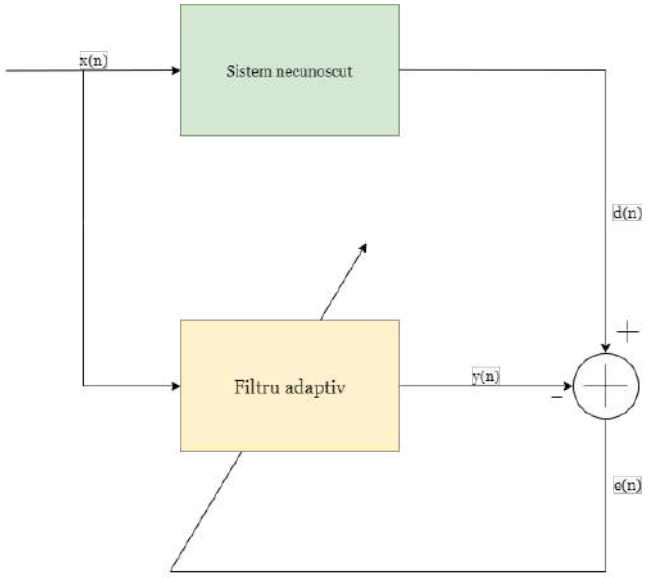


Figura 2.7: Configurația de identificare de sistem.

2.2.2 Eliminarea zgomotului folosind rețele neurale

2.2.2.1 Rețele neurale complet interconectate

Din moment ce performanțele metodelor clasice bazate pe filtre digitale sunt limitate de ordinul filtrului ales și de tăria zgomotului suprapus peste semnale, se vor propune metode de îndepărtare a zgomotului aditiv bazate pe rețele neurale. Spre deosebire de filtrele clasice unde este necesar să cunoaștem atât semnalul inițial, curat, cât și cel corupt de zgomot, rețelele neurale sunt capabile să reducă zgomotul fără a se cunoaște semnalul inițial.

În această lucrare se vor propune 2 tipuri diferite de arhitecturi: rețeaua neurală complet interconectată (*eng. FCN – Fully Connected Network*) și rețea neurală "*capsulă*" (*eng. CapsNet*).

Astfel, un FCN este alcătuit din mai multe straturi neurale, fiecare neuron dintr-un anumit strat fiind conectat cu toți neuronii stratului următor. În general, aceste rețele sunt folosite pentru a antrena orice tip de date, și este folosită în numeroase aplicații de machine learning, cu precădere în cele de regresie sau clasificare.

Pentru a construi setul de antrenare al rețelei, va fi necesară o preprocesare a semnalelor. Astfel, fiecare semnal va fi segmentat in ferestre de câte 23ms. Numărul de eșantioane corespunzător unui singur segment este $\lfloor (T_w \times fs) \rfloor$, unde T_w reprezintă lungimea în timp a ferestrei. Astfel, s-au obtinut $N = 1014$ eșantioane.

Se propune antrenarea rețelei folosind modulul coeficienților Fourier al fiecărui segment obținuți prin aplicarea transformatei Fourier rapidă (*eng. FFT - Fast Fourier Transform*). Întrucât performanțele calculului FFT sunt optime în cazul în care numărul de puncte este o putere a lui 2, se va calcula transformata în $N_{FFT} = 1024$ puncte. Pentru simplitatea calculelor, toată această operație se va realiza cu ajutorul operației STFT (*eng. Short Time Fourier Transform*). Operația presupune calculul FFT pe intervale mici de analiză. Formula pentru această transformată este:

$$STFT\{x(n)\}(m,\omega) = \sum_{n=-\infty}^{+\infty} x(n)w(n-m)e^{-j\omega n} \tag{2.14}$$

unde $w(m)$ reprezintă fereastra de analiză folosită, iar m reprezintă lungimea ei. Pentru a reduce erorile de amplitudine și de împrăștiere spectrală, a fost folosită fereastra de analiză

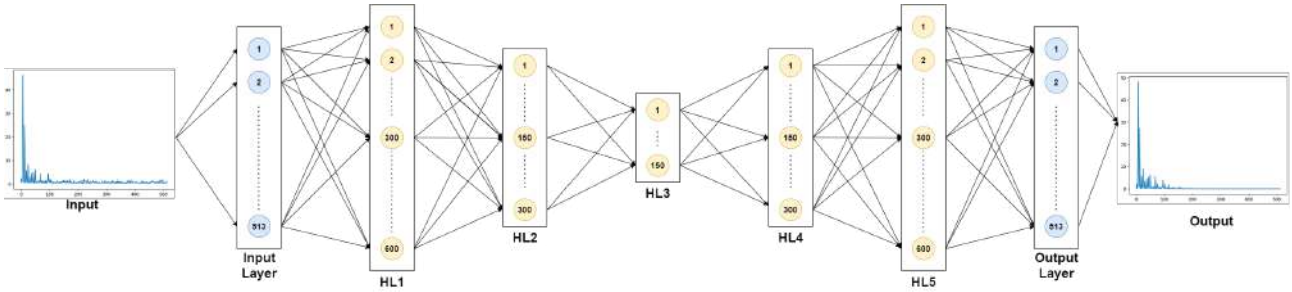


Figura 2.8: Arhitectura rețelei neurale interconectate.

Hanning. Segmentele sunt 50% suprapuse pentru a evita pierderea de informație de la capetele intervalului de analiză. Se vor extrage doar primii 513 coeficienți ai FFT-ului corespunzători frecvențelor mai mici decât $f_s/2$, ce vor reprezenta partea utilă a spectrului, coeficienții frecvențelor cuprinse între $f_s/2$ și f_s reprezentând de fapt partea virtuală a spectrului, oglindirea părții utile.

Din moment ce problema de minimizare a zgomotului poate fi privită ca o problemă de regeresie, scopul rețelei este acela de a găsi un estimat al modulului coeficienților Fourier $|\hat{S}_n|_{(1\leq n\leq \lfloor N_{fft}/2\rfloor+1)}$ cât mai apropiat de modulul spectrului semnalului original $|S_n|_{(1\leq n\leq \lfloor N_{fft}/2\rfloor+1)}$. Astfel, având aceeași dimensiune pentru vectorul de intrare și cel de ieșire, este de preferat ca dimensiunile primului și ultimului strat să fie identice și egale cu numărul de elemente din fiecare vector. Arhitectura folosită este structurată pe două module: codare și decodare (*eng. encoder-decoder*). Prin reducerea dimensiunilor, informația este încapsulată într-un vector de dimensiuni fixe și prin operația inversă de decodificare, readusă la dimensiunea inițială a vectorului de intrare. Arhitectura este evidențiată în figura 2.8.

Astfel, pentru straturile de intrare și de ieșire s-au ales 600 de neuroni pentru a avea un număr apropiat de numărul de trăsături de la intrare, respectiv ieșire. Treptat, dimensiunea a fost redusă la 150 de neuroni prin înjumătățiri pentru partea de codare, operația de revenrie la dimensiunea inițială fiind facută în ordine inversă pentru partea de decodare.

Pentru a segmenta setul de date de antrenare, s-a folosit un lot (*eng. batch*) de 1024 de vectori de intrare, fiecare strat ascuns fiind urmat de o etapă de *normalizare a lotului* (*eng. batch-normalization*) pentru a aduce vectorii de intrare în aceeași gamă de valori și pentru a îmbunătăți performanțele și stabilitatea de antrenare a rețelei neurale. Pentru a evita fenomenul de *supra-învățare* (*eng. overfit*), fiecare strat va fi urmat de o etapă de *dropout*, i.e. un anumit procent din numărul de neuroni ai unui strat va fi egalat cu 0. În acest caz, s-a ales un procent de 50% pentru întreaga rețea, cu excepția straturilor de intrare și de ieșire. Funcția de activare folosită pentru fiecare strat cu excepția ultimului este *ReLU*, pentru ultimul strat am folosit activarea identitate (cu alte cuvinte, pentru ultimul strat nu am folosit un operator neliniar de activare ca și în celelalte cazuri), el fiind utilizat doar pentru a readuce vectorul ce parcurge rețeaua la dimensiunea inițială.

Odată estimat modulul transformatei STFT a semnalului zgomotos, acesta este înmulțit cu faza transformatei STFT a semnalului inițial, iar prin operația inversă *ISTFT*(*eng. Inverse Short-Time Fourier Transform*) semnalul este reconstruit. Întregul sistem de prelucrare a semnalelor zgomotoase folosind FCN este prezentat în figura 2.9.

2.2.2.2 Rețele capsulă

În ultimii ani, conceptul de rețea capsulă a fost tot mai des folosit în aplicații ce presupun antrenare cu numere complexe, precum aplicații de radar, aplicații în medicină precum analiza RMN, aplicații de prelucrare a imaginilor și a semnalelor audio etc. [19], [20]. În această lucrare

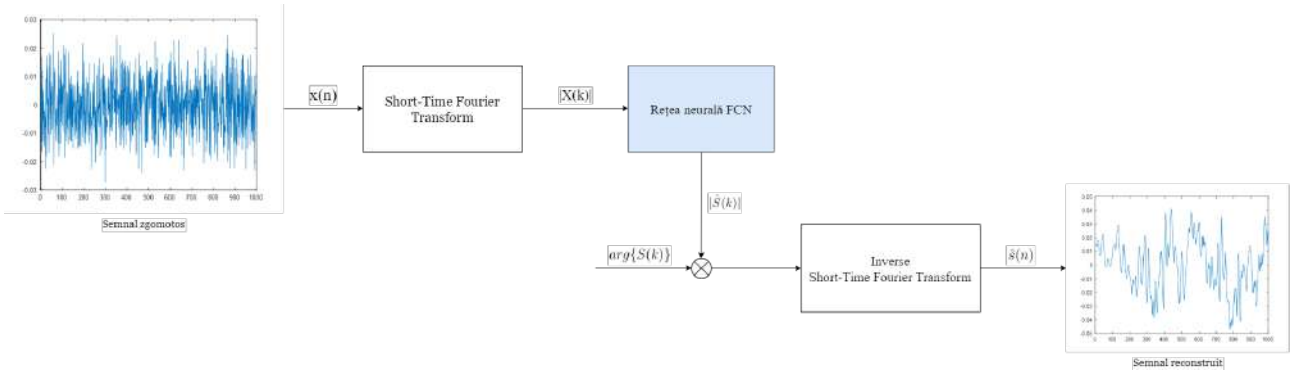


Figura 2.9: Sistemul de eliminare a zgomotului folosind FCN.

se va propune o rețea capsulă ce va trata problema de reducere a zgomotului din semnale, văzută în acest caz ca o problemă de regresie.

Pentru această rețea, se propune construirea setului de date de antrenare folosind părțile reale și imaginare ale coeficienților Fourier extrași cu ajutorul operației STFT. Procedura de extragere a coeficienților Fourier este identică cu cea descrisă în cazul FCN-ului, însă vectorul de intrare va fi format din concatenarea a 2 vectori ce conțin părțile reale, respectiv cele imaginare ale coeficienților, obținându-se 1026 de valori.

O capsulă reprezintă un grup de neuroni ce prelucrează anumite trăsături ale vectorului de intrare din stratul respectiv. Pentru implementarea rețelei se vor propune straturi ale căror ponderi vor fi împărțite în 2 capsule, fiecare capsulă prelucrând partea reală, respectiv partea imaginară a coeficienților Fourier. Stratul propus este prezentat în figura 2.10.

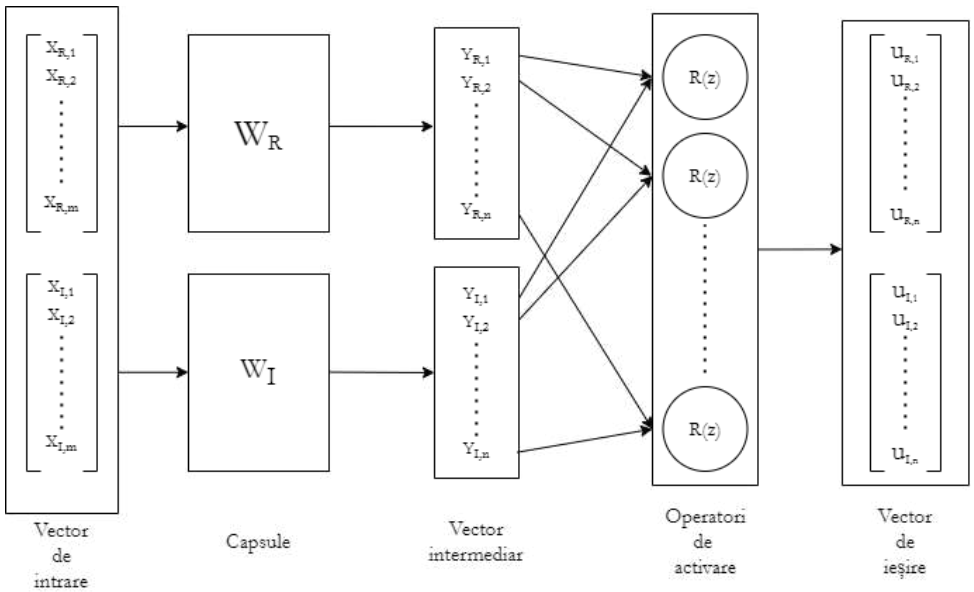


Figura 2.10: Stratul propus pentru rețeaua capsulă.

Astfel, datele de ieșire ale capsulelor sunt reordonate în perechi real-imaginar pentru a fi prelucrate de către funcția de activare, iar apoi ordonate astfel încât vectorul de ieșire al stratului să fie format din concatenarea vectorilor ce conțin coeficienții reali, respectiv pe cei imaginari. Dată fiind structura de prelucrare a datelor, pentru un strat cu n neuroni vom avea mereu la ieșire un vector de dimensiune $2n$.

Din moment ce se lucrează cu valori complexe, operatorii clasici de activare nu prezintă aceleași performanțe ca și în cazul rețelelor ce estimează modulul coeficienților Fourier. În această lucrare se vor propune 3 operatori concepuți pentru activarea numerelor complexe:

1.

$$R : z \to \frac{\mu|z|}{1+|z|^2}z,\ z \in \mathbb{C}, \tag{2.15}$$

unde $\mu = \frac{8}{3\sqrt{3}}$. Astfel, dacă presupunem aceleași notații ca în figura 2.10 și faptul că $z_i = y_{R,i} + jy_{I,i}$, avem:

$$u_{R,i} = \frac{\mu\sqrt{x_{R,i}^2 + y_{I,i}^2}}{1 + x_{R,i}^2 + x_{I,i}^2} y_{R,i} \tag{2.16}$$

$$u_{I,i} = \frac{\mu\sqrt{x_{R,i}^2 + y_{I,i}^2}}{1 + x_{R,i}^2 + x_{I,i}^2} y_{I,i} \tag{2.17}$$

2.

$$R : z \to \frac{z}{c + \frac{1}{r}|z|},\ z \in \mathbb{C} \tag{2.18}$$

unde $c, r \in \mathbb{R}_+$. Funcția a fost propusă prima dată în [21] pentru antrenarea rețelelor folosind numere complexe. Principala proprietate a funcției este aceea de a proiecta valoarea z în planul complex cu coordonatele în interiorul cercului $\{z : |z| < r\}$. În general, se aleg valorie $c = r = 1$.

3.

$$R : z \to \mathbb{C}ReLU(z) = ReLU(Re\{z\}) + jReLU(Im\{z\}), z \in \mathbb{C} \tag{2.19}$$

Se propune aplicarea activării $ReLU$ perechii real-imaginar aferentă fiecărui coeficient Fourier. Activarea a fost propusă și în [22] în scopul de a compara performanțele funcției $ReLU$ și a variantelor sale în diferite configurații pentru antrenarea numerelor complexe. Notația vine de la denumirea funcției *Complex ReLU*. Fiind una dintre cele mai folosite activări în antrenarea rețelelor neurale, s-a încercat aplicarea ei și în cazul valorilor complexe.

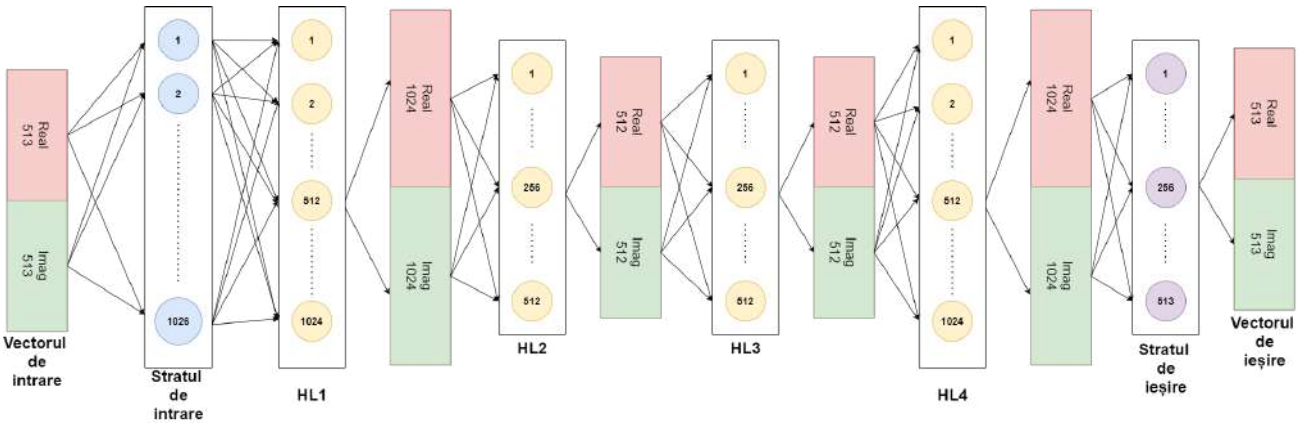


Figura 2.11: Arhitectura propusă pentru rețeaua capsulă.

Rețeaua FCN a fost antrenată folosind doar modulul coeficienților Fourier, fără a avea acces la informația de fază. Avantajul antrenării folosind argumentele carteziene ale coeficienților oferă informație și despre fază, nu doar despre modul, astfel că la reconstrucția semnalului nu va mai fi nevoie de faza semnalului inițial.

Arhitectura rețelei propuse este asemănătoare cu cea a rețelei FCN, întrucât s-a încercat păstrarea configurației de codor-decodor. Întrucât vectorul de intrare va avea 1026 de elemente și se preferă ca numărul de neuroni să fie putere a lui 2, se vor alege 1024 de neuroni pentru primul strat urmat de un strat de 512 neuroni. Pentru a respecta configurația propusă, celelalte 2 straturi vor avea 512, respectiv 1024 de neuroni. Ultimul strat de 513 neuroni este adăugat pentru a readuce datele din decodor la aceleași dimensiuni cu datele de ieșire. Arhitectura este evidențiată în figura 2.11.

Capitolul 3
Scheme de învățare folosind constrângeri

3.1 Formularea problemei

Rețelele neurale artificiale sunt tot mai des folosite în probleme ce presupun învățarea, modelarea și procesarea datelor. Aceste sisteme pot fi sensibile la diferite perturbații care, deși nu modifică sesizabil exemplele de intrare folosite pentru rețea, astfel încât eroarea dintre un exemplu curat și un exemplu perturbat să fie neglijabilă, ieșirile aferente acestor exemple diferă cu mult. Această problema a fost până acum abordată în aplicații de procesare și clasificare a imaginilor. [23]. Există diferite metode ce pot combate această problemă de robustețe a rețelei, iar metoda propusă în această lucrare este dată de controlul *constantei Lipschitz*.

Algoritmii de tip *Plug and Play* (*PnP*) au fost folosiți în probleme ce presupun înlăturarea zgomotului din imagini, iar noi credem ei pot fi folosiți și în probleme ce țin de zgomot în domeniul audio. Se dorește implementarea unui astfel de algoritm pentru asigura simultan deverberarea și scoaterea de sub zgomot a semnalelor audio. Pentru a asigura convergența acestui algoritm, trebuie asigurată și nonexpansivitatea sistemului.

Fie $f : A \to B, A \subseteq \mathbb{R}^m, B \subseteq \mathbb{R}^n$. Această funcție este lipschitiziană pe A dacă există o constantă θ așa încât:

$$\|f(x) - f(y)\| \leq \theta \|x - y\|, \quad \forall x, y \in A \tag{3.1}$$

unde $\|.\|$ reprezintă norma euclidiană iar θ reprezintă *constanta Lipschitz* a funcției f. Cea mai mică constantă θ poartă numele de *modul Lipschitz*. Dacă funcția f este continuă și diferențiabilă, atunci modulul Lipschitz poate fi calculat ca maximul normei spectrale a matricei jacobiene a funcției f:

$$\theta = \max_x \|\nabla f(x)\|_s \tag{3.2}$$

Dacă $\theta \leq 1$, spunem că funcția f este *nonexpansivă*.

În cazul rețelelor neurale, putem folosi această constantă pentru a controla nonexpansivitatea sistemului respectiv. Calculul precis al acestei constante s-a dovedit a fi o problemă de complexitate *NP* (*eng. NP-hard*). Se propune defnirea acestei constante, în primă fază, prin calculul produsului normelor spectrale ale ponderilor rețelei, iar mai apoi prin calculul normei spectrale a produsului ponderilor. Metode de calcul pentru această constantă au fost deja propuse în [24], [25], [26].

Se consideră T un sistem de învățare automată și $x \in \mathbb{R}^{N_0}$ datele de intrare în sistemul respectiv. Ieșirea acestui sistem va fi $T(x) \in \mathbb{R}^{N_m}$. Daca datele de intrare sunt alterate de o perturbație aleatore $z \in \mathbb{R}^{N_0}$, atunci putem caracteriza eroarea pe care acea perturbație o are asupra ieșirii prin inegalitatea:

$$\|T(x+z) - T(x)\| \leq \theta \|z\| \tag{3.3}$$

Precum a fost deja menționat în Capitolul 1, sistemul T poate fi scris ca o compunere de funcții:

$$T = R_n \circ (W_n x_n + b_n) \circ \cdots \circ R_1 \circ (W_1 x_1 + b_1) \tag{3.4}$$

În acest context, $\mathcal{H}_i$ reprezintă un spațiu Hilbert, $R_i : \mathcal{H}_i \to \mathcal{H}_i$ reprezintă un operator de activare, $W_i : \mathcal{H}_{i-1} \to \mathcal{H}_i$ reprezintă matricea pondere a fiecăuri strat, $x_i \in \mathcal{H}_{i-1}$ reperzintă vectorul de intrare al fiecărui strat iar $b_i \in \mathcal{H}_i$ reprezintă termenul de polarizare (*bias*), $\forall i \in \{1, \ldots, m\}$.

Dacă operatorii de activare ai straturilor sunt nonexpansivi, o aproximare grosieră a constantei Lipschitz a sistemului T este dată de formula:

$$\theta = \prod_{i=1}^{m} \|W_i\|_s \tag{3.5}$$

unde $\|.\|_s$ reprezintă norma spectrală a ponderii stratului i.

Astfel, se poate arăta că, în cazul unei rețele neurale, această constantă poate fi controlată impunând anumite constrângeri asupra normelor spectrale ale ponderilor. S-a arătat în [24] că majoritatea operatorilor folosiți în literatură sunt non-expansivi(spre exemplu, funcțiile *ReLU, Sigmoid*).

Mai mult, dacă este asigurată și condiția ca ponderile straturilor să fie pozitive: $W_i \in \mathbb{R}_+^{N_{i-1} \times N_i}$, atunci constanta Lipschitz minimă a rețelei este:

$$\theta = \|W_m \ldots W_1\|_s \tag{3.6}$$

Cu alte cuvinte, constanta Lipschitz a unei rețele neurale cu ponderi pozitive și operatori de activare non-expansivi este aceeași cu a unei rețele neurale căreia i-au fost eliminați operatorii de activare (o rețea neurală pur liniară). Aceasta este o aproixamre mai precisă a constantei Lipschitz, însă și cea mai greu de controlat întrucât rețeaua are mai puține grade de libertate pentru a-și optimiza parametrii față de cazul general în care nu sunt folosite constrângeri.

În această lucrare, se vor propune câteva metode de antrenare bazate pe constrângerea a ponderilor pentru controlul constnatei Lipschitz a unei rețele neurale, în contextul obținerii unui sistem de eliminare a zgomotului și a reverberației. De asemenea, se vor observa efectele constrângerilor asupra rețelelor neurale, comparativ cu cele neconstrânse.

3.2 Constrângeri propuse

Pentru a obține o rețea non-expansivă este necesar controlul constantei Lipchitz a acesteia. În cele ce urmează vor fi prezentate mai multe seturi de constrângeri ce pot fi aplicate în procesul

de antrenare spre obtinerea acestui deziderat.

1. **Pozitivitate** – O primă condiție primordială care permite calculul constantei Lipschitz este aceea de a asigura pozitivitatea ponderilor [27]. Astfel, se propun următorele seturi de constrângeri:

$$\mathcal{D}_i = \{W_i | W_i \geq 0, \quad \forall i \in \{1, \ldots, m\}\} \tag{3.7}$$

2. **Norma ponderilor** – Cea mai simplă metodă prin care se poate impune o constrângere asupra produsului normelor este de a constrânge fiecare normă în parte prin limitarea ei la o valoare maximă. Astfel, se propun următoarele seturi:

$$\mathcal{C}_i^1 = \{W_i | \, \|W_i\|_s \leq \theta, \quad \forall i \in \{1, \ldots, m\}\} \tag{3.8}$$

3. **Produsul normelor** – Odată asigurată constrângerea fiecărei norme, se poate face o aproximare a constantei Lipschitz prin constrângerea produsului normelor. Aceasta este o metodă de aproximare grosolană, din moment ce rețeaua are mai multe grade de libertate prin care își învață parametrii:

$$\mathcal{C}_i^2 = \{W_i | \, \prod_{i=1}^{m} \|W_i\|_s \leq \theta, \quad \forall i \in \{1, \ldots, m\}\} \tag{3.9}$$

4. **Norma produsului ponderilor** – Prin constrângerea produsului normelor ponderilor se asigură și un control cât mai precis a constantei Lipschitz, în același timp respectând condițiile deja menționate (nonexpansivitatea operatorilor de activare și pozitivitate):

$$\mathcal{C}_i^3 = \{W_i | \, \| \prod_{i=1}^{m} W_i\|_s \leq \theta, \quad \forall i \in \{1, \ldots, m\}\} \tag{3.10}$$

Trebuie menționat faptul că implementarea acestor constrângeri se va face în cadrul procesului de antrenare a rețelei, mai exact în partea de optimizare, prin care rețeaua își actualizează parametrii odată cu minimzarea funcției de cost.

Indiferent de optimizatorul folosit, procesul de actualizare a parametrilor rețelelor neurale este evidețiat matematic astfel:

$$W_{i,n+1} = W_{i,n} - \gamma_n \nabla \mathcal{L}(W_{i,n}), \quad \forall i \in \{1, \ldots, m\} \tag{3.11}$$

În acest context, $\nabla \mathcal{L}(W_{i,n})$ reprezintă gradientul funcției de cost în funcție de stratul i la a n-a iterație iar $\gamma_n \in [0, 1]$ reprezintă rata de învățare a sistemului. Această rată poate ramâne constantă sau poate fi ajustată pe parcursul procesului de antrenare. Gradientul funcției de cost este actualizat de la ultimul strat la primul strat print-un algoritm de propagare inversă.

Pentru a aplica constrângerile menționate mai sus, este necesară proiecția ponderilor actualizate prin procesul de optimizare descris de ecuația 3.11 pe un spațiu $\mathcal{S}$ descris de constrângerile $\mathcal{C}_i^{1-3}$

$$W_{i,n+1} = \mathsf{P}_{\mathcal{S}}(W_{i,n} - \gamma_n \nabla \mathcal{L}(W_{i,n})) \tag{3.12}$$

Pentru a asigura această proiecție, este necesar ca setul C să fie închis și convex. Totuși, pentru a valida ecuația 3.6, este necesar să fie respectate simultan condiția de constrângere a normelor ($\mathcal{C}_i^{1-3}$) și cea de pozitivitate ($\mathcal{D}$). În acest caz, spațiul $\mathcal{S}$ devinde spațiul $\mathcal{C}_i \cap \mathcal{D}$.

Astfel, problema se rezumă la calculul proiecției ponderilor pe intersecția a 2 spații convexe, o problemă destul de dificil de rezolvat, din moment ce deși seturile de constrângeri menționate sunt convexe, intersecția lor nu mai respectă această proprietate.

În lucrarea [28] au fost propuse câteva metode pentru calulul proiecției pe intersecția a 2 spații convexe. În această lucrare propunem 2 metode de calcul al proiecției:

1. **Proiecție aproximativă** – Se pot considera pe rând proiecțiile celor 2 spații, întâi proiectând pe spațiul descris de $\mathcal{D}$, iar apoi pe cel descris de $\mathcal{C}_\rangle$:

$$\tilde{\mathsf{P}}_{\mathcal{C}_i \cap \mathcal{D}} \simeq \mathsf{P}_{\mathcal{D}} \circ \mathsf{P}_{\mathcal{C}_i}$$

   Deși această metodă pentru determinarea proiecției nu este una precisă, ea este ușor de implementat și nu necesită putere computațională ridicată.

2. **Proiecție exactă** – Propunem o metodă de a aproxima mai precis proiecția prin introducerea algoritmului *DFB (eng. Dual Forward-Backward)* ce va fi descris ulterior.

Astfel, pentru a impune constrângerile $\mathcal{C}_i^{1-3}$, se propune următorul set convex și închis:

$$\mathcal{C}_{i,n} = \{W_i \in \mathbb{R}^{N_i \times N_{i-1}} | \ \|A_{i,n} W_i B_{i,n}\|_s \leq \theta\} \tag{3.13}$$

În acest context, matricile $A_{i,n}$ și $B_{i,n}$ reprezintă produsul ponderilor anterioare, respectiv posterioare stratului i la a n-a iterație. Dacă se impune ca $A_{i,n} = Id$ pentru $i = m$ și $B_{i,n} = Id$ pentru $i = 1$, definim:

$$A_{i,n} = W_{m,n} \dots W_{i+1,n}$$
$$B_{i,n} = W_{i-1,n} \dots W_{1,n}$$

Înainte de a descrie algoritmul *DFB*, propunem proiecțiile pentru spațiile C și D. Astfel, pentru $W \in \mathbb{R}^{m \times n}$ și dacă considerăm $D = [0, +\infty]^{m \times n}$, atunci:

$$\mathsf{P}_{\mathcal{D}}(W) = (\tilde{W}_{i,j})_{(1 \leq i \leq m, 1 \leq j \leq n)}, \tag{3.14}$$

unde pentru $i \in \{1, \dots, m\}, \quad j \in \{1, \dots, n\}$, avem:

$$\tilde{W}_{i,j} = \begin{cases} W_{i,j}, & \text{dacă } W_{i,j} \geq 0 \\ 0, & \text{în rest} \end{cases} \tag{3.15}$$

Considerăm $\mathcal{B}(0, \theta)$ sfera închisă de centru 0 și rază θ definită astfel:

$$\mathcal{B}(0, \theta) = \{W \in \mathbb{R}^{m \times n} | \quad \|W\|_s \leq \theta\}$$

Pentru fiecare $(W_{i,j})_{(1 \leq i \leq m, 1 \leq j \leq n)} \in \mathbb{R}^{m \times n}$ considerăm următoarea descompunere în valori singulare (SVD): $Y = U \Lambda V^T$, unde $U \in \mathbb{R}^{m \times r}$ și $V \in \mathbb{R}^{n \times r}$ sunt matrici ce respectă condițiile: $U^T U = Id$ și $V^T V = Id$, $r = min\{p, q\}$ iar $\Lambda = Diag(\lambda_1, \dots, \lambda_r)$, unde $\lambda_i, \quad i \in \{1, \dots, r\}$ reprezinta valorile singulare ale matricei Y. Atunci:

$$\mathsf{P}_{\mathcal{B}(0,\theta)}(Y) = U \tilde{\Lambda} V^T \tag{3.16}$$

unde $\tilde{\Lambda}$ este definit astfel:

$$\tilde{\Lambda} = Diag(\tilde{\lambda_1}, \dots, \tilde{\lambda_r})$$

$$\tilde{\lambda_i} = \begin{cases} \lambda_i, & \text{dacă} \quad \lambda_i \leq \theta \\ \theta, & \text{în rest} \end{cases} \qquad \forall i \in \{1, \ldots, r\} \tag{3.17}$$

În cazul unei matrici $A \in \mathbb{R}^{n \times n}$, considerăm $x \in \mathbb{R}^n$ și $\lambda \in \mathbb{R}$. Spunem că x se numește vector propriu a matricei A și λ valoare proprie asociată vectorului x dacă și numai dacă $Ax = \lambda x$. Date fiind aceste noțiuni, putem considera norma spectrală a unei matrici $A \in \mathbb{R}^{p \times q}$ ca fiind valoarea sa singulară maximă: $\|A\|_s = \lambda_{max}$ [29].

Algoritmul *DFB* [30] prin care sunt controlate constrângerile este următorul:

Let:$Y_0 \in \mathbb{R}^{N_m \times N_0}$;
Set:$\epsilon \in [0, 1/(\|A_{i,n}\|_s \|B_{i,n}\|_s)^2]$;
for *l=0,1,...* **do**
  Set $\gamma_l \in [\epsilon, 2/(\|A_{i,n}\|_s \|B_{i,n}\|_s)^2 - \epsilon]$
  $V_l = \mathsf{P}_{\mathcal{D}}(W_i - A_{i,n}^T Y_l B_{i,n}^T)$
  $\tilde{Y_l} = Y_l + \gamma_l A_{i,n} V_l B_{i,n}$
  $Y_{l+1} = \tilde{Y_l} - \gamma_l P_{B(0,\theta)}(\gamma_l^{-1} \tilde{Y_l})$
end
retrun V_l

La fiecare iterație, algoritmul proiectează ponderea W_i întâi pe spațiul $\mathcal{D}$, apoi pe spațiul descris de sfera $\mathcal{B}(0, \theta)$. Aceasta este o soluție ce oferă proiecția precisă a ponderilor în sensul constrângerilor $\mathcal{C}_i^{1-3}$, în cazul constrângerilor $\mathcal{C}_i^1$ și $\mathcal{C}_i^2$ fiind necesară condiția $A_{i,n} = B_{i,n} = Id$.

Se propune o aproximare de complexitate redusă pentru calculul proiecției pe sfera $\mathcal{B}(0, \theta)$, din moment ce calculul descompunerii în valori singulare necesită complexitate mare de calcul, mai ales în cazul matricilor de dimensiuni mari:

$$\gamma_l P_{B(0,\theta)}(\gamma_l^{-1} \tilde{Y_l}) = \frac{\theta}{\|\tilde{Y_l}\|_s} \tilde{Y_l} \tag{3.18}$$

În sensul constrângerii $\mathcal{C}_i^1$, este necesară aplicarea operațiilor 3.14 și 3.18 pentru fiecare pondere, fără a mai fi necesară parcurgerea algoritmului *DFB*. Aceleași operații vor fi aplicate și pentru calculul constrângerii $\mathcal{C}_i^3$, însă va fi necesară parcurgerea algoritmului menționat anterior.

Am folosit de asemenea o variantă accelerată a algoritmului *DFB* asemănătoare algoritmului *FISTA* [30], [31]:

Let:$Y_0 \in \mathbb{R}^{N_m \times N_0}$;
Set:$\gamma \in [0, 1/(\|A_{i,n}\|_s \|B_{i,n}\|_s)^2]$;
Set:$\alpha \in [2, +\infty]$;
for *l=0,1,...* **do**
  $\eta_l = \frac{l}{l+1+\alpha}$
  $Z_l = Y_l + \eta_l (Y_l - Y_{l-1})$
  $V_l = P_D(W_i - A_{i,n}^T Z_l B_{i,n}^T)$
  $\tilde{Y_l} = Z_l + \gamma_l A_{i,n} V_l B_{i,n}$
  $Y_{l+1} = \tilde{Y_l} - \gamma_l P_{B(0,\theta)}(\gamma_l^{-1} \tilde{Y_l})$
end
return V_l

Capitolul 4
Rezultate experimentale

Pentru a îmbunătăți calitatea semnalelor audio am aplicat secvențial algoritmul de deverberare și sistemele implementate pentru a înlătura zgomotul. Agloritmul de deconvoluție a fost implementată folosind 200 de iterații. Metoda de eliminare a zgomotului folosind clasica operație de *soft-thresholding* implementată în cadrul algoritmului de deverberare s-a dovedit a fi ineficientă. Prin folosirea acestor două metode menționate au fost introduse perturbații zgomotoase în semnal la toate frecvențele, înrăutățind criteriile de performanță ce vor fi menționate ulterior.

Semnalele rezultate din algoritmul de deverberare au fost aplicate sistemelor de eliminare a zgomotului discutate în această lucrare. Inițial, au fost folosite metodele bazate pe filtrele clasice: filtrul Wiener și filtrul adaptiv. Ordinele filtrelor au fost alese așa încât să avem criterii de performanță sporite: ordinul 100 pentru filtrul Wiener, respectiv 200 pentru filtrul adaptiv. Performanța sistemelor de îndepărtare a zgomotului bazate pe rețele neurale au fost comparate cu cele ale ale filtrelor.

Pentru a soluționa problema antrenării rețelei, este necesară împărțirea melodiilor conținute în baza de date menționată în Capitolul 2. Astfel, acestea vor fi delimitate în 3 categorii:

- **Setul de antrenare**: Este setul ce conține datele care vor folosite pentru a antrena parametrii rețelei (ponderile și termenii *bias*). Rețeaua învață din aceste date. Setul de antrenare conține 90 de melodii cu durata totală de 67 de minute.
- **Setul de validare**: Este folosit pentru evaluarea modelului, pe baza acestor date sunt modelați *hiperparametrii* rețelei (rata de învățare, numărul de unități de pe fiecare strat etc.). Aceste date nu participă în mod direct la antrenarea rețelei, ele făcând parte din procesul de optimizare. Setul de validare conține 9 melodii cu durată totală de 9 minute.
- **Setul de testare**: Este folosit doar după ce procesul de antrenare este finalizat și pe baza acestuia sunt evaluate performanțele rețelei. Datele de testare conțin cât mai multe trăsături asemănătoare celor din setul de antrenare și validare. Acest set va conține o singură melodie cu durata de un minut și 17 secunde.

Pentru antrenarea rețelelor, au fost folosiți coeficienții Fourier ai semnalelor de antrenare extrași cu ajutorul operației *STFT*. În cazul în care pentru antrenare este folosit modulul, la reconstrucție, datele prezise sunt înmulțite cu o exponențială complexă ce conține ca argument faza semnalului inițial. Pentru reconstrucția semnalului din părțile reale și imaginare ale coeficienților, au fost reconstruiți coeficienții Fourier prin simpla sumare a părților: $z = Re\{z\} + jIm\{z\}$.

Am evaluat rezultatele obținute de sistemele implementate pe trei criterii de performanță diferite:

- **SNR – Signal to Noise Ratio**: Raportul semnal-zgomot, măsurat în decibeli (dB), calculat pe baza puterii semnalului inițial, respectiv pe cea a zgomotului:

$$SNR = 10lg\frac{P_{semnal}}{P_{zgomot}} \tag{4.1}$$

  Puterea unui semnal oarecare $x(n)$ este dat de raportul dintre energia semnalului și durata sa și se calculează astfel:

$$P = \frac{1}{N}\sum_{i=0}^{N-1} x^2(n) \tag{4.2}$$

- **PSNR – Peak Signal to Noise Ratio**: Raportul semnal-zgomot de vârf, calculat ca raportul dintre maximul valorii instantanee a puterii semnalului util și puterea zgomotului:

$$PSNR = 10lg\frac{max\{x^2(n)\}}{P_{zgomot}} \tag{4.3}$$

- **CC – Cross Corelation**: Coeficientul de corelație dintre semnalul util inițial și cel reconstruit. Dacă considerăm două variabile aleatoare A și B, coeficientul de corelație $\rho(A, B)$ este definit astfel:

$$\rho(A, B) = \frac{1}{N}\sum_{i=0}^{N-1}\frac{(A_i - \mu_A)(B_i - \mu_B)}{\sigma_A\sigma_B} \tag{4.4}$$

  unde μ_A și μ_B reprezintă media variabilelor A și B, iar σ_A și σ_B reprezintă deviațiile lor standard.

Operatorul de activare ales pentru rețeaua interconectată este dat de funcția *ReLU* datorită avantajelor menționate în Capitolul 1, iar optimizatorul folosit este *Adam*. Pentru a evalua performanța rețelei neurale în comparație cu metodele clasice de îndepărtare a zgomotului s-a folosit ca nivel de referință semnalul deverberat.

	Model	SNR	PSNR	CC
Semnal zgomotos	$s * h + n$	5	21.30	0.87
Semnal de referință	Deverb	3.13	19.39	0.72
Metodele clasice	Filtrul Wiener	9.23	25.49	0.94
	Filtrul adaptiv	8.12	24.40	0.92
Rețea neurală	FCN	**16.82**	**33.07**	**0.99**

Tabela 4.1: Comparație între metodele clasice și cea bazată pe rețea neurală interconectată

Se observă în tabelul 4.1 că performanțele rețelei neurale sunt superioare metodelor clasice, obținând un câștig de 13.69dB în comparație cu performanțele filtrului Wiener unde câștigul este de 6.1dB.

Se va face de asemenea o comparație între rețelele capsulă și metodele clasice de eliminare a zgomotului folosind trei modele diferite, câte un model pentru fiecare funcție de activare propuse în Capitolul 2.

	Model	Activare	SNR	PSNR	CC
Referință	$s * h + n$	-	5	21.3	0.87
	Deverb	-	3.13	19.39	0.72
Metode clasice	Filtrul Wiener	-	9.23	25.49	0.94
	Filtrul adaptiv	-	8.12	24.40	0.92
Rețele neurale	CapsNet 1	$f(z) = \frac{8}{3\sqrt{3}} \frac{\|z\|}{1+\|z\|^2} z$	**12.24**	**28.49**	**0.97**
	CapsNet 2	$f(z) = \frac{z}{1+\|z\|}$	12.01	28.25	0.97
	CapsNet 3	$f(z) = \mathbb{C}ReLU(z)$	11.98	28.22	0.97

Tabela 4.2: Comparație între metodele clasice și cele bazate pe rețea capsulă.

În tabela 4.2 se poate observa cum performanțele acestor rețele depășește pe cele ale metodelor clasice, ca și în cazul rețelei complet interconectate. Scopul acestor funcții de activare este de aduce părțile reale și imaginare într-o gamă cât mai restrânsă de valori (spre exemplu, cât mai aproape de cercul unitate de centru 0 din planul complex: $|z| < 1$, sau chiar în interiorul său). În acest fel, rețeaua învață mai ușor trăsăturile datelor respective, astfel performanțele ei cresc.

Model	Constrângere	Constanta Lipschitz (θ)	SNR	PSNR	CC
$s * h + n$	-	-	5	21.3	0.87
Deverb	-	-	3.13	19.39	0.72
Filtrul Wiener	-	-	9.23	25.49	0.94
Filtrul adaptiv	-	-	8.12	24.40	0.92
FCN	$\mathcal{D}$	2.74×10^{16}	12.94	29.19	0.975
	$\mathcal{D} \cap \mathcal{C}_i^1$	0.85	9.72	25.97	0.96
	$\mathcal{D} \cap \mathcal{C}_{i,n}$	1.01	6.31	22.56	0.89

Tabela 4.3: Comparație între metodele clasice și cele bazate pe modele constrânse.

Se propune o comparație între rețele antrenate folosind constrângerile propuse în Capitolul 3: $\mathcal{D}, \mathcal{C}_i^1, \mathcal{C}_i^3$. Rezultatele sunt ilustrate în tabelul 4.3. Pentru calculul acestor constrângeri s-a folosit setul propus în Capitolul 3: $\mathcal{C}_{i,n}$, împreună cu algoritmul *DFB*. Constrângerea rețelei pentru calculul precis al constantei Lipschitz se va face treptat, prima condiție fiind pozitivitatea

ponderilor. Performanțele rețelei scad pe măsură ce sunt aplicate diferite constrângeri în etapa de antrenare, întrucât rețeaua va avea din ce în ce mai puține grade de libertate pentru a-și învăța parametrii. Constanta Lipschitz poate fi calculată precis doar prin proiecția ponderilor pe spațiul dat de intersecția $\mathcal{D} \cap \mathcal{C}_{i,n}$. Scopul este de a obține un sistem nonexpansiv prin limitarea acestei constante la valoarea maximă $\theta = 1$.

Model	MSE		MAE	
	Antrenare	Validare	Antrenare	Validare
FCN	0.361	0.118	0.149	0.078
CapsNet 1	0.22	0.146	0.124	0.101
CapsNet 2	0.271	0.160	0.130	0.104
CapsNet 3	0.322	0.166	0.144	0.095

Tabela 4.4: FCN vs. CapsNet – metrici de performanță

Sesizăm în tabelul 4.4 diferența dintre valoarile funcției de cost la antrenare și validare pentru fiecare rețea. Funcția de cost folosită pentru optimizarea parametrilor este *Mean Squared Error*, iar metrica de evaluare este dată de *Mean Average Error*. Acestea au fost deja prezentate în Capitolul 1. Condiția de verificare că nu se ajunge la fenomenul de *overfit* este ca valoarea funcției de cost la validare să fie mai mică decât cea de la antrenare. Conform rezultatelor din tabel, în cadrul antrenării rețelelor nu s-a ajuns în nicio variantă la *overfit*. Modelele au fost salvate după cea mai mică valoare a funcției de cost în cadrul validării.

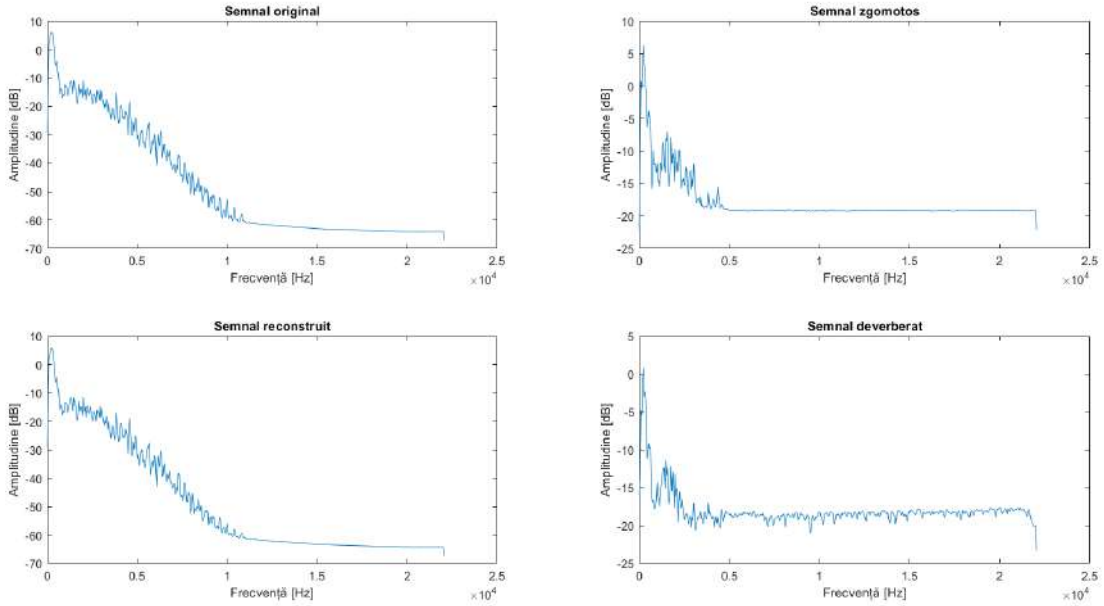


Figura 4.1: Comparație între densitățile spectrale de putere pentru baza de date de test.

Pentru a ilustra grafic performanța rețelelor, sunt prezentate în figura 4.1 densitățile spectrale de putere a diferite semnale de interes: semnalul inițial, semnalul corupt de zgomot, semnalul reverberat și cel reconstruit. Pentru calculul densității de putere s-a folosit metoda

Welch [32]. Astfel, semnalul este segmentat în cadre de analiză de lungime L, suprapuse parțial cu un anumit număr de eșantioane. Pentru al i-lea segment se va calcula densitatea spectrală de putere astfel:

$$S_{x_i}(e^{j\omega}) = \frac{1}{L}|X_i(e^{j\omega})|^2 \tag{4.5}$$

Densitatea spectrală de putere a întregului semnal va fi dată de suma acestor densități ale tuturor segmentelor:

$$S_x(e^{j\omega}) = \frac{1}{K}\sum_{i=0}^{K-1} S_{x_i}(e^{j\omega}) = \frac{1}{KL}\sum_{i=0}^{K-1} |X_i(e^{j\omega})|^2 \tag{4.6}$$

unde K este numărul de segmente în care a fost împărțit semnalul.

Concluzii

Concluzii generale

Prin acest proiect de diplomă s-a realizat eliminarea efectului reverberației și a zgomotului utilizând diferiți algoritmi bazați pe metode de învățare automată. În primă fază, au fost create semnale reverberate și zgomotoase pornind de la melodiile cuprinse în baza de date menționată în Capitolul 2. Semnalele au fost deverberate folosind un algoritm iterativ de deconvoluție, urmat de un algoritm de gradient proximal ce încearcă reducerea zgomotului. Această soluție a dus la eliminarea reverberației, dar nu și a zgomotului. Pentru a combate această problemă, au fost implementate mai multe soluții:

- În primă fază, au fost implementate două filtre clasice de eliminare a zgomotului: filtru Wiener și filtru adaptiv. Acestea analizează cadre din semnalul corupt de zgomot obținute folosind fereastra de analiză dreptunghiulară. Semnalul este reconstruit folosind aceste cadre din care a fost redus efectul zgomotului.
- Pentru a mări performanțele eliminării zgomotului, au fost implementate două tipuri de sisteme bazate pe rețele neurale. O rețea va prezice modulul, iar cealaltă părțile reale și imaginare coeficienților Fourier. De aceea, este necesară o preprocesare a semnalului zgomots. Astfel, semnalele sunt împărțite în segmente suprapuse, cărora le este aplicată transformata Fourier rapidă. Rețelele au fost antrenate atât neconstrâns, cât și folosind constrângerile propuse în Capitolul 3.

În concluzie, a fost implementat un sistem care să elimine complet efectul reverberației și al zgomotului folosind doar semnalul audio în cazul filtrării și o singră trăsătură a acestora, anume coeficienții Fourier, în cazul rețelelor neurale. Acest lucru constituie un mare avantaj, deoarece nu sunt necesare alte transformări pentru a evidenția anumite proprietăți specifice semnalului, fiind suficientă doar o postprocesare pentru reconstrucția lor. Implementarea filtrelor a fost făcută în programul `Matlab`, iar implementarea rețelelor a fost făcută folosind limbajul de programare `Python`.

Contribuții personale

- Obținerea semnalelor reverberate și zgomotoase folosind programul `Matlab`;
- Implementarea filtrelor propuse în Capitolul 2;
- Preprocesarea semnalelor pentru obținerea coeficienților Fourier necesari pentru antrenarea rețelelor;

- Implementarea rețelei interconectate folosind arhitectura propusă în Capitolul 3;
- Implementarea constrângerilor $\mathcal{D}, \mathcal{C}_i^1, \mathcal{C}_i^3$
- Implementarea stratului folosit în rețeaua capsulă și implementarea rețelei folosind arhitectura propusă în Capitolul 3
- Implementarea sistemului de postprocesare pentru reconstrucția semnalelor din coeficienții Fourier prezișI de către rețele.

Codul sursă al implementării metodelor prezentate în această lucrare este disponibil aici.

Dezvoltări ulterioare

Se va încerca pe viitor antrenarea rețelelor capsulă folosind constrângerile propuse în această lucrare astfel încât performanțele acestora să fie cât mai apropiate de cele neconstrânse. Pornind de la aceste modele, se va încerca implementarea unui algoritm iterativ ce va elimina efectul reverberației și al zgomotului simultan.

Bibliografie

[1] Guoshen Yu, Stéphane Mallat, and Emmanuel Bacry. Audio denoising by time-frequency block thresholding. *IEEE Transactions on Signal Processing*, 56(5):1830–1839, 2008.

[2] Pascal Scalart and Jozue Filho. Speech enhancement based on a priori signal to noise estimation. volume 2, pages 629 – 632 vol. 2, 06 1996.

[3] N. Alamdari, A. Azarang, and N. Kehtarnavaz. Improving deep speech denoising by noisy2noisy signal mapping. *Applied Acoustics*, 172:107631, 2021.

[4] Bilal Dendani, Halima Bahi, and Toufik Sari. Speech enhancement based on deep autoencoder for remote arabic speech recognition. In Abderrahim El Moataz, Driss Mammass, Alamin Mansouri, and Fathallah Nouboud, editors, *Image and Signal Processing*, pages 221–229, Cham, 2020. Springer International Publishing.

[5] D. Burileanu. Prelucrarea digitală a semnalelor. *Note de curs, UPB ETTI,* 2019.

[6] `http://rf-opto.etc.tuiasi.ro/docs/files/RRCS_cap%202.pdf`.

[7] Constanin Strîmbu. Semnale și circuite electronice - analiza si prelucrarea semnalelor. 2007.

[8] `https://sp.utcluj.ro/ROXAC2016/Prezentari_postere/PreprocesareaSemnalelorAudio.pdf`.

[9] D. Burileanu. Tehnici avansate de prelucrare digitală a semnalelor. *Note de curs, UPB ETTI,* 2019.

[10] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.

[11] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pages 65–386, 1958.

[12] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In Yves Lechevallier and Gilbert Saporta, editors, *Proceedings of COMPSTAT'2010*, pages 177–186, Heidelberg, 2010. Physica-Verlag HD.

[13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[14] Marco Jeub, Magnus Schäfer, and Peter Vary. A binaural room impulse response database for the evaluation of dereverberation algorithms. In *Proceedings of International Conference on Digital Signal Processing (DSP)*, pages 1–4, Santorini, Greece, July 2009. IEEE, IET, EURASIP, IEEE.

[15] D.L. Donoho. De-noising by soft-thresholding. *IEEE Transactions on Information Theory*, 41(3):613–627, 1995.

[16] Santiago Pascual, Antonio Bonafonte, and Joan Serrà. SEGAN: speech enhancement generative adversarial network. *CoRR*, abs/1703.09452, 2017.

[17] Francois G. Germain, Qifeng Chen, and Vladlen Koltun. Speech denoising with deep feature losses, 2018.

[18] Yunpeng Li, Beat Gfeller, Marco Tagliasacchi, and Dominik Roblek. Learning to denoise historical music, 2020.

[19] Parnian Afshar, Arash Mohammadi, and Konstantinos N. Plataniotis. Brain tumor type classification via capsule networks. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 3129–3133, 2018.

[20] Wei Zhao, Jianbo Ye, Min Yang, Zeyang Lei, Suofei Zhang, and Zhou Zhao. Investigating capsule networks with dynamic routing for text classification, 2018.

[21] G.M. Georgiou and C. Koutsougeras. Complex domain backpropagation. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 39(5):330–334, 1992.

[22] Chiheb Trabelsi, Olexa Bilaniuk, Ying Zhang, Dmitriy Serdyuk, Sandeep Subramanian, João Felipe Santos, Soroush Mehri, Negar Rostamzadeh, Yoshua Bengio, and Christopher J Pal. Deep complex networks, 2018.

[23] Matthieu Terris, Audrey Repetti, Jean-Christophe Pesquet, and Yves Wiaux. Building firmly nonexpansive convolutional neural networks. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8658–8662, 2020.

[24] Patrick L. Combettes and Jean-Christophe Pesquet. Lipschitz certificates for layered network structures driven by averaged activation operators, 2020.

[25] Aladin Virmaux and Kevin Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[26] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2014.

[27] Jan Chorowski and Jacek Zurada. Learning understandable neural networks with nonnegative weight constraints. *IEEE transactions on neural networks and learning systems*, 26:62–9, 01 2015.

[28] Patrick L. Combettes and Jean-Christophe Pesquet. Proximal splitting methods in signal processing, 2010.

[29] http://math.etc.tuiasi.ro/apletea/cursuri/algcap5.pdf.

[30] Ana Neacsu, Jean-Christophe Pesquet, and Corneliu Burileanu. Accuracy-robustness trade-off for positively weighted neural networks. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8389–8393. IEEE, 2020.

[31] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[32] P. Welch. The use of fast fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Transactions on Audio and Electroacoustics*, 15(2):70–73, 1967.