,

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

Contribuții la recunoașterea automată a vorbirii pentru limba română. Impactul utilizării seturilor de date nou apărute și a modelelor Fast Conformer

Lucrare de licență

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

Conducători științifici
Conf. dr. ing. Horia CUCU
Drd. Ing. Gabriel PÎRLOGEANU

Absolvent
Gabriel ION

Anul 2026

Universitatea Națională de Știință și Tehnologie POLITEHNICA București
Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei
Program de studiu **INF**

Anexa 1

TEMA PROIECTULUI DE DIPLOMĂ

a studentului **ION P. Gabriel, 444A**

1. Titlul temei: Contribuții la recunoașterea automată a vorbirii pentru limba română. Impactul utilizării seturilor de date nou apărute și a modelelor Fast Conformer

2. Descrierea temei:
Lucrarea de licență propusă are ca temă antrenarea unui sistem de recunoaștere automată a vorbirii (RAV) pentru limba română, utilizând un subset de aproximativ 10.000 de ore de vorbire adnotate automat din datasetul Granary, care reprezintă în prezent cea mai mare resursă de date adnotate disponibilă pentru limba română. Relevanța lucrării constă în contribuția adusă dezvoltării tehnologiilor RAV pentru o limbă cu resurse limitate, prin exploatarea unui volum de date fără precedent. Scopul principal este dezvoltarea și evaluarea unui model RAV modern și cuantificarea impactului utilizării acestui set de date asupra performanței sistemului, prin comparație cu sisteme RAV existente pentru limba română. În cadrul lucrării vor fi utilizate tehnici de învățare profundă și arhitecturi end-to-end specifice recunoașterii vorbirii, iar evaluarea se va realiza folosind metrici standard, precum Word Error Rate, pentru a evidenția avantajele și limitările abordării propuse.

3. Contribuția originală:
Contribuția studentului constă în proiectarea, implementarea și evaluarea experimentală a unui sistem de RAV pentru limba română prin adaptarea unui model preantrenat pe limba engleză, utilizând cantități variabile de date din subsetul Granary. Studentul va realiza experimente comparative pentru a cuantifica impactul volumului de date parțial adnotate asupra performanței sistemului, analizând eficiența acestei abordări în raport cu modelele RAV existente în literatura de specialitate. De asemenea, contribuția include construirea unui nou subset de date optimizat, care să poată fi utilizat eficient împreună cu resursele deja disponibile pentru limba română, precum și formularea de concluzii bazate pe măsurători și evaluări standard, în vederea evidențierii avantajelor și limitărilor soluției propuse.

4. Resurse și bibliografie:
1. Koluguri, Nithin Rao, Monica Sekoyan, George Zelenfroynd, Sasha Meister, Shuoyang Ding, Sofia Kostandian, He Huang et al. "Granary: Speech Recognition and Translation Dataset in 25 European Languages." arXiv preprint arXiv:2505.13404 (2025).

2. Rekesh, Dima, Nithin Rao Koluguri, Samuel Kriman, Somshubra Majumdar, Vahid Noroozi, He Huang, Oleksii Hrinchuk et al. "Fast conformer with linearly scalable attention for efficient speech recognition." In 2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), pp. 1-8. IEEE, 2023.

3.
Pirlogeanu, Gabriel, Alexandru-Lucian Georgescu, and Horia Cucu. "Open Source State-Of-the-Art Solution for Romanian Speech Recognition." arXiv preprint arXiv:2511.03361 (2025).

5. Data înregistrării temei: 2025-12-23 15:21:45

Conducător(i) lucrare,
Conf. dr. ing. Horia CUCU

Drd. Ing. Gabriel Pîrlogeanu

Student,
ION P. Gabriel

Director departament,
Ș.L. dr. ing Bogdan FLOREA

Decan,
Prof. dr. ing. Mihnea UDREA

Cod Validare: **63022dd6dc**

Declaraţie de onestitate academică

Prin prezenta declar că lucrarea cu titlul *Contribuții la recunoașterea automată a vorbirii pentru limba română. Impactul utilizării seturilor de date nou apărute și a modelelor Fast Conformer* , 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 *Ingineria informației* 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, Iulie 2026.
Absolvent: Gabriel ION

.....

Cuprins

Lista figurilor

Lista tabelelor

Lista acronimelor

ADAM - Adaptive Moment Estimation
AI - Artificial Intelligence (Inteligență Artificială)
ASR - Automatic Speech Recognition (Recunoaștere Automată a Vorbirii)
BCE - Binary Cross-Entropy (Entropie încrucișată binară)
BPE - Byte Pair Encoding (Codificare pe perechi de octeți)
CCE - Categorical Cross-Entropy (Entropie încrucișată categorică)
CDEP - Camera Deputaților (Setul de date al Camerei Deputaților)
CPS - Characters Per Second (Caractere pe secundă)
CTC - Connectionist Temporal Classification (Clasificare temporală conexionistă)
CUDA - Compute Unified Device Architecture (Arhitectură unificată de calcul pe dispozitiv)
CV - Common Voice
FLEURS - Few-shot Learning Evaluation of Universal Speech Representations
GPU - Graphics Processing Unit (Unitate de procesare grafică)
I/O - Input/Output (Intrare/Ieșire)
IBM - International Business Machines
JSONL - JSON Lines (Linii JSON)
LLM - Large Language Model (Model de limbaj de mari dimensiuni)
MAE - Mean Absolute Error (Eroare absolută medie)
MLP - Multi-Layer Perceptron (Perceptron multistrat)
MSE - Mean Squared Error (Eroare medie pătratică)
NeMo - Neural Modules
NLP - Natural Language Processing (Procesarea limbajului natural)
RAM - Random Access Memory (Memorie cu acces aleatoriu)
Regex - Regular Expression (Expresie regulată)
ReLU - Rectified Linear Unit
RNN - Recurrent Neural Network (Rețea neurală recurentă)
RNN-T - Recurrent Neural Network Transducer (Transductor cu rețea neurală recurentă)
RSC - Romanian Speech Corpus
SDP - Speech Data Processor
SGD - Stochastic Gradient Descent (Gradient descendent stocastic)
SSC - Spontaneous Speech Corpus
TDT - Token-and-Duration Transducer
TTS - Text-to-Speech (Sistem text-în-vorbire)
USPDATRO - Under-Represented Speech Dataset for Romanian
VAD - Voice Activity Detection (Detecția activității vocale)
VRAM - Video Random Access Memory (Memorie video cu acces aleatoriu)
WER - Word Error Rate (Rata de eroare a cuvintelor)
WPS - Words Per Second (Cuvinte pe secundă)
YAML - YAML Ain't Markup Language

Introducere

Motivație

În contextul evoluției rapide a Inteligenței Artificiale și a procesării limbajului natural (NLP), interacțiunea om-mașină a devenit un pilon central al tehnologiei moderne. Istoria recunoașterii vorbirii începe în anul 1952, când laboratoarele Bell Labs au construit primul „Sistem de Recunoaștere Automată a Cifrelor", ce putea recunoaște cifrele 0-9 și avea la bază analiza formatelor spectrale [4]. Din acel moment și până în zilele noastre, industria a cunoscut diverse schimbări majore în ceea ce privește metodele de recunoaștere a vorbirii, de la metodele deterministe (analize spectrale), până la cele probabilistice (modele Markov) sau conexioniste (rețele neurale).

Era modernă a recunoașterii vorbirii are la bază modele cu arhitecturi bazate pe rețele neuronale, care permit o robustețe mult mai mare în fața variabilității vorbirii umane. Astăzi, rețelele neuronale profunde nu doar transcriu sunete, ci integrează implicit cunoștințe gramaticale și semantice, fiind capabile să decodifice corect mesajul chiar și în condiții de zgomot de fundal sau accente diverse.

Atenția marilor companii de tehnologie s-a concentrat, în principal, asupra dezvoltării de modele performante pentru limbile de circulație internațională, precum engleza sau spaniola, care dispun de resurse lingvistice extinse. Acestea includ zeci de mii de ore de înregistrări audio diversificate, esențiale pentru antrenarea modelelor și pentru obținerea unei precizii ridicate. În comparație, limba română a beneficiat de resurse medii, ceea ce reprezintă un obstacol important în dezvoltarea unor modele performante.

Totuși, lansarea setului de date „Granary" de către Nvidia, în august 2025, a reprezentat un pas important pentru cercetarea în domeniul procesării vorbirii pentru 25 de limbi europene. Cu aproximativ 1 milion de ore de înregistrări audio, acest set de date oferă un volum semnificativ mai mare de resurse față de cele disponibile anterior pentru limbile cu resurse limitate, inclusiv pentru limba română, pentru care există aproximativ 17 mii de ore de înregistrări audio.

Accesul la acest corpus extins poate contribui semnificativ la îmbunătățirea performanței sistemelor de recunoaștere automată a vorbirii pentru română, în special în ceea ce privește reducerea ratei de eroare și adaptarea la diferite accente sau stiluri de vorbire. Acest lucru este important pentru aplicații practice, precum asistenții vocali, sistemele de transcriere automată sau soluțiile de dictare.

Obiective

Pentru acest proiect au fost propuse următoarele obiective principale:

1. Dezvoltarea unui sistem robust de recunoaștere automată a vorbirii (ASR) pentru limba română, utilizând arhitectura hibridă *FastConformer*.

2. Implementarea unui flux experimental de preprocesare pentru curățarea, filtrarea și normalizarea unui corpus masiv de date (Granary), în vederea obținerii unui set de antrenare de înaltă calitate.

3. Evaluarea impactului pe care scalarea volumului de date de antrenare (de la 500 la 10.000 de ore) îl are asupra performanței și convergenței rețelei acustice.

4. Analiza influenței dimensiunii arhitecturale asupra acurateței recunoașterii vocale, prin antrenarea și compararea variantelor de model *Large* (120 milioane de parametri) și *XLarge* (616 de milioane de parametri).

5. Testarea exhaustivă a modelelor dezvoltate pe seturi de date independente, acoperind diverse tipologii de discurs (citit, spontan, oratoric), și compararea performanțelor obținute cu modele de referință (*state-of-the-art*), precum *SpeD ParakeetRo* și *Whisper*.

Structura lucrării

Prezenta lucrare este structurată în trei capitole principale, după cum urmează:

Capitolul 1 stabilește fundamentul teoretic al lucrării. Acesta prezintă noțiunile de bază din domeniul învățării automate și introduce concepte specifice sistemelor de recunoaștere automată a vorbirii (ASR), detaliind arhitecturile și mecanismele matematice care stau la baza modelelor moderne de procesare a semnalelor acustice.

Capitolul 2 este dedicat metodologiei proiectului și descrie în detaliu etapele practice de dezvoltare. Aici sunt prezentate utilitarele software folosite, arhitectura fluxului de procesare a datelor, precum și configurațiile hiperparametrilor utilizate pentru antrenarea modelelor. De asemenea, capitolul definește metricile de evaluare și descrie seturile de date pe care s-au efectuat testele independente.

Capitolul 3 prezintă și analizează rezultatele obținute în urma antrenării și testării modelelor. Performanțele sistemelor dezvoltate sunt comparate direct cu soluțiile de referință actuale, evidențiind capacitatea de generalizare a arhitecturii în funcție de volumul datelor, dimensiunea modelului și condițiile variate de mediu acustic.

Capitolul 1
Noțiuni Teoretice

1.1 Învățarea Automată

Inteligența Artificială (AI) înglobează sistemele matematice prin care se simulează comportamentul inteligent și cuprinde multiple modalități de calcul, precum logica matematică (if-else) sau calculul probabilistic.

Învățarea Automată (Machine Learning) reprezintă un subdomeniu al **"Inteligenței Artificiale"**, în care un sistem matematic "învață" să ia decizii în urma modelării datelor observate. Aceste sisteme sunt folosite în multiple domenii, unde deciziile umane pot fi influențate de asistența modelelor matematice, precum medicina (cu aplicații în imagistica medicală), economie (în prelucrarea și interpretarea datelor numerice) sau în viața de zi-cu-zi, folosind modele de tip *Natural Processing Language* (NLP) și *Automatic Speech Recognition* (ASR), ce facilitează interacțiunea omului cu mașina.

În funcție domeniul de interes, există diverși algoritmi ce facilitează metodologia de antrenare a acestor modele matematice. Totuși, procesul general de pregătire și antrenare este asemănător pentru toate aplicațiile unde se dorește integrarea învățării automate și este reprezentat în Figura 1.1.

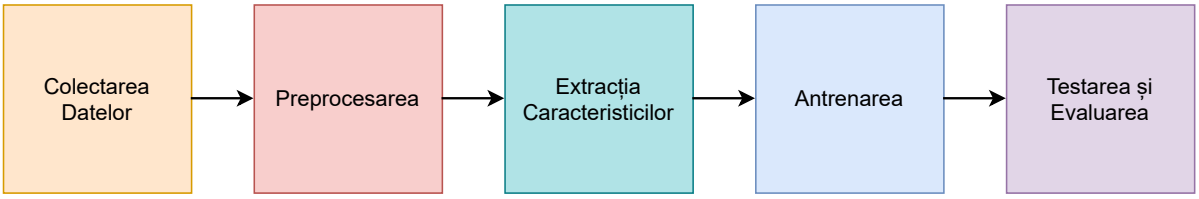

Figura 1.1: Procesul general de pregătire și antrenare.

Colectarea datelor reprezintă primul pas în procesul de pregătire al unui model și constă în adunarea informației din diferite surse cu scopul de a crea un set de antrenament reprezentativ, relevant și suficient de diversificat pentru problema pe care algoritmul urmează să o rezolve. Preprocesarea include pașii de filtrare, curățare sau ștergere, după caz, și reprezintă un proces important ce influențează direct rezultatul final al predicției, întrucât modelul nu poate distinge datele curate de cele poluate, iar antrenarea cu astfel de date va duce la decizii greșite.

Extragerea de caracteristici constă în transformarea datelor brute (imagini, text, sunet) în seturi de variabile sau trăsături numerice, care pot fi procesate de algoritmii de învățare automată. În funcție de domeniul și tipurile de date cercetate, algoritmii acestui proces variază semnificativ. În domeniul imaginilor (Computer Vision), se scot în evidență contururi, texturi, colțuri sau forme geometrice, transformând o matrice de pixeli într-un set de caracteristici abstracte. Pentru procesarea limbajului natural (NLP), cuvintele sunt convertite în vectori numerici pentru a surprinde semantica, frecvența sau contextul frazelor.

Pentru procesul de antrenare, algoritmii responsabili sunt divizați în patru categorii, în funcție

de modul în care modelul învață caracteristicile numerice ale setului de date: Învățare Asistată (*Supervised Learning*), Învățare Neasistată (*Unsupervised Learning*), Învățare Semi-Asistată (*Semi-supervised Learning*) și Învățare Bazată pe Recompense (*Reinforcement Learning*). În acest proiect, ne vom concentra atenția pe învățarea asistată, prezentă în algoritmii de recunoaștere automată a vorbirii.

1.1.1 Învățarea Asistată

Un model de învățare supervizată definește o corespondență de la una sau mai multe intrări către una sau mai multe ieșiri. Modelul reprezintă, în esență, o funcție matematică. Atunci când datele de intrare sunt procesate prin intermediul acestei funcții, ea calculează datele de ieșire, proces care poartă numele de inferență. De asemenea, ecuația modelului conține parametri. Valori diferite ale parametrilor modifică rezultatul calculului, astfel, ecuația modelului descrie o familie de posibile relații între intrări și ieșiri, în timp ce parametrii determină relația specifică [2].

În învățarea asistată, scopul este de a construi un model ce primește un tip de dată $\mathbf{x}$ la intrare și generează la ieșire predicția $\mathbf{y}$. Pentru aceasta, este necesară o funcție $\mathbf{f}[\bullet]$ care ia la intrare $\mathbf{x}$ și returnează $\mathbf{y}$:

$$\mathbf{y} = \mathbf{f}[\mathbf{x}]. \tag{1.1}$$

Modelul conține, de asemenea, parametrii ϕ. Alegerea parametrilor determină relația specifică dintre intrare și ieșire [2], așadar ar trebui, de fapt, să scriem:

$$\mathbf{y} = \mathbf{f}[\mathbf{x}, \phi] \tag{1.2}$$

În acest context, antrenarea reprezintă încercările de a găsi parametrii ϕ optimi care minimizează diferența (sau eroarea) dintre predicțiile generate de model și valorile reale (etichetele) din setul de date de antrenament. Pentru a cuantifica acest proces de minimizare a erorii, în loc să măsurăm similaritatea, de obicei discutăm despre conceptul opus acesteia: distanța dintre ieșirea sistemului și ieșirea de referință. Această distanță poartă numele de funcție de pierdere (*loss function*) sau funcție de cost (*cost function*) [5].

$$\hat{\phi} = \underset{\phi}{\operatorname{argmin}}[L[\phi]]. \tag{1.3}$$

Un alt lucru necesar în antrenare îl constituie algoritmul de optimizare, prin intermediul căruia actualizăm iterativ parametrii, cu scopul de a reduce funcția de pierdere menționată mai sus. Cel mai adesea este folosit Gradientul Descent (*Gradient Descent*), fiind unul dintre cei mai populari algoritmi cu care se realizează optimizări și de departe cea mai comună metodă de optimizare a rețelelor neuronale [6]. Gradientul Descent este o metodă de minimizare a unei funcții obiectiv $J(\theta)$, parametrizată de parametrii modelului $\theta \in \mathbb{R}^d$, prin actualizarea acestor parametri în direcția opusă gradientului funcției obiectiv $\nabla_\theta J(\theta)$ în raport cu parametrii [6]. Rata de învățare (*learning rate*) η determină dimensiunea pașilor pe care îi facem pentru a atinge un minim (local). Cu alte cuvinte, urmăm direcția pantei suprafeței create de funcția obiectiv, în jos (*downhill*), până când ajungem într-o vale [6].

Regula matematică de bază a algoritmului de gradient descent, pentru fiecare pas de antrenare (iterația t), este definită astfel:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta J(\theta_t) \tag{1.4}$$

unde:

- θ_{t+1} reprezintă noile valori ale parametrilor (după actualizare);

- θ_t reprezintă valorile curente ale parametrilor;
- η este rata de învățare (*learning rate*), care dictează mărimea pasului;
- $\nabla_\theta J(\theta_t)$ este gradientul funcției de pierdere calculat în punctul curent.

1.1.2 Rețele Neurale

Rețelele neurale reprezintă ansambluri parametrizate de algoritmi care au la bază, ca element structural fundamental, modelul perceptronului [7]. Conceptul teoretic din spatele acestuia a fost introdus inițial în anul 1943 de către cercetătorii Warren McCulloch și Walter Pitts în lucrarea „A Logical Calculus of Ideas Immanent in Nervous Activity" [7]. În acest studiu, autorii au demonstrat modul în care un neuron biologic poate fi abstractizat matematic sub forma unui simplu mecanism decizional bazat pe o valoare de prag, capabil să evalueze funcții logice [7]. Cu toate acestea, arhitectura operațională a perceptronului a fost dezvoltată și implementată abia în anul 1957, de către Frank Rosenblatt [8].

Utilizat preponderent în cadrul problemelor de clasificare binară, perceptronul reprezintă un algoritm de bază al învățării supervizate, a cărui structură conceptuală emulează funcționarea neuronului biologic. Din perspectivă matematică, acest model procesează informația calculând suma ponderată a variabilelor de intrare, înmulțite cu ponderile lor specifice, la care se adaugă un termen de deplasare (*bias*). În etapa finală, rezultatul agregat este evaluat prin intermediul unei funcții de activare, care dictează valoarea decizională de ieșire a rețelei. Acest model este reprezentat în binar de funcția următoarea, și poate fi vizualizat în Figura 1.2

$$f(x) = \begin{cases} 1 & \text{pentru } w \cdot x + b < 0 \\ 0 & \text{în caz contrar} \end{cases} \tag{1.5}$$

unde:

- w reprezintă vectorul ponderilor (alcătuit din numere reale);
- x constituie vectorul valorilor de intrare (datele procesate);
- b este termenul de deplasare (cunoscut și sub denumirea de *bias*).

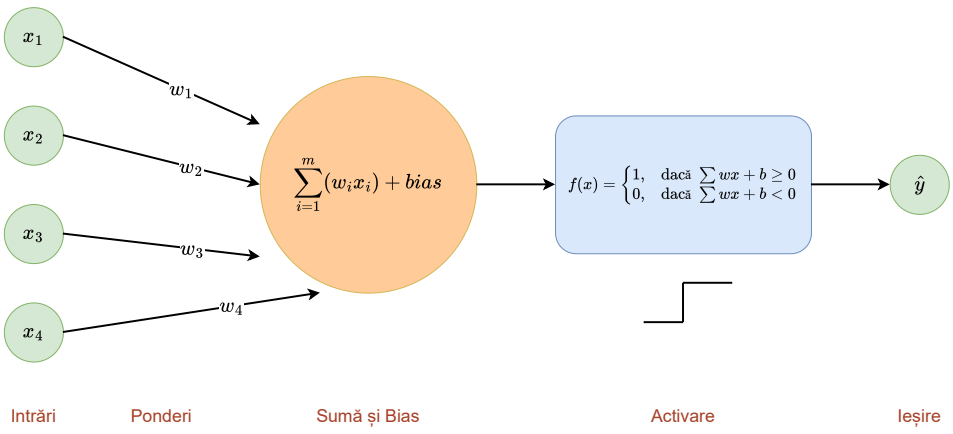

Figura 1.2: Structura Perceptronului: intrările x_i sunt înmulțite cu ponderile w_i, apoi împreună cu un bias sunt însumate, iar rezultatul este trecut printr-o funcție de activare, în cazul imaginii de tip treaptă (*Heaviside*), care produce ieșirea binară $\hat{y}$ [1].

Pe măsură ce numărul de unități ascunse (neuroni) crește, rețelele cu un singur strat își îmbunătățesc capacitatea de reprezentare. Cu toate acestea, s-a demonstrat că, pentru anumite funcții, numărul de unități ascunse necesar pentru o modelare precisă devine mult prea mare [2]. Pentru a depăși aceasta problema, au fost dezvoltate arhitecturile de tip *Multilayer Perceptron* (MLP), fiind concepute pentru a modela funcții cu un caracter pronunțat neliniar. Algoritmul de învățare al unui MLP se desfășoară în două etape. Prima fază, denumită propagarea înainte (*feedforward*), presupune transmiterea succesivă a semnalelor de la un strat la altul [2]. Pe parcursul acestui flux, fiecare neuron trece printr-un filtru decizional impus de funcția sa de activare, care îi dictează starea (activă sau de repaus). Acest proces iterativ continuă neîntrerupt până în momentul în care sunt generate valorile finale la nivelul stratului de ieșire [2].

Odată finalizată această primă etapă de propagare, predicțiile generate de model sunt evaluate în raport cu valorile reale (etichetele țintă). Discrepanța dintre cele două este cuantificată sub forma unei erori, calculată prin intermediul unei funcții de cost (*loss function*) alese în prealabil. Cea de-a doua fază fundamentală a procesului de antrenare constă în propagarea "înapoi" a acestei erori (*backpropagation*) prin întreaga arhitectură a rețelei [2]. Prin aplicarea gradientului descent, sistemul ajustează și optimizează iterativ ponderile fiecărui neuron în parte, corectând astfel modelul pentru a minimiza eroarea la următoarele iterații [2].

Funcția de Activare

Deși modelul clasic al perceptronului a fost conceput cu precădere pentru rezolvarea problemelor de clasificare binară, în funcție de specificul fiecărei aplicații, activarea neuronilor se poate realiza prin intermediul unei varietăți de funcții, depășind limitările simplei funcții treaptă. În practica antrenării modelelor de învățare automată, următoarele funcții de activare sunt implementate în mod frecvent:

- **ReLU** [9] (*Rectified Linear Unit*) reprezintă o transformare neliniară, deosebit de eficientă din punct de vedere computațional, care facilitează un proces de optimizare superior pentru rețelele neuronale:
$$ReLU(x) = \max(0, x) \tag{1.6}$$

- **Leaky ReLU** [10] reprezintă o adaptare a variantei ReLU standard. În această formulare este introdus un parametru α (frecvent inițializat cu valoarea 0.1), având rolul de a contracara fenomenul nedorit al neuronilor inactivi (cunoscut în literatură sub denumirea de *dying ReLU* [10]):
$$f(x) = \begin{cases} \alpha x & \text{pentru } x < 0 \\ x & \text{pentru } x \geq 0 \end{cases} \tag{1.7}$$

- **Sigmoid** este aplicată cu preponderență la nivelul stratului de ieșire al unei rețele pentru generarea predicțiilor, având proprietatea de a restrânge valorile rezultate în intervalul strict $[0, 1]$. Domeniile sale principale de aplicabilitate includ problemele de regresie și clasificarea binară:
$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{1.8}$$

- **Tangenta hiperbolică (Tanh)**, spre deosebire de funcția sigmoid, este centrată în origine (zero-centrică), generând o convergență mai lină a gradientului. Preia, însă, dezavantajul susceptibilității la problema dispariției gradientului și implică un cost computațional mai ridicat:
$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{1.9}$$

- **Softmax** este o funcție de activare destinată exclusiv straturilor de ieșire ale rețelelor proiectate pentru probleme de clasificare multiclasă:

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad \text{pentru } i = 1, 2, \ldots, K \tag{1.10}$$

unde z_i definește valoarea de intrare asociată celui de-al i-lea neuron, iar K reprezintă numărul total de neuroni din stratul de activare curent (care este, de obicei, stratul final). În mod uzual, acest număr discret de neuroni este determinat direct de numărul de clase specifice problemei modelate.

Funcția de Pierdere

Funcția de pierdere (cunoscută și sub denumirea de funcție de cost) deține un rol esențial în etapa de antrenare a unui model, având responsabilitatea de a ghida ajustarea parametrilor necesari pentru construirea unei funcții de mapare cât mai precise. În practică, următoarele funcții de cost sunt frecvent utilizate, alegerea lor depinzând de natura problemei abordate:

- **Eroarea Pătratică Medie (MSE)** este aplicată cu precădere în rezolvarea problemelor de regresie. O particularitate a acesteia constă în modul de penalizare a abaterilor: erorile mai mari de 1 sunt penalizate sever, în timp ce erorile subunitare au un impact considerabil diminuat asupra costului global:

$$MSE = \sum_{i=1}^{D} (x_i - y_i)^2 \tag{1.11}$$

- **Eroarea Absolută Medie (MAE)** reprezintă, de asemenea, o opțiune standard pentru modelele de regresie, având o dinamică asemănătoare cu MSE. Totuși, în contrast cu Eroarea Pătratică Medie, această funcție manifestă o sensibilitate mai accentuată față de erorile subunitare și evită penalizarea disproporționată a erorilor mari (care depășesc valoarea 1):

$$MAE = \sum_{i=1}^{D} |x_i - y_i| \tag{1.12}$$

- **Entropia Încrucișată Binară (BCE)** este concepută special pentru optimizarea problemelor de clasificare binară. În arhitecturile rețelelor neuronale, utilizarea acestei funcții este cel mai adesea cuplată cu o funcție de activare Sigmoidă la nivelul stratului de ieșire [2]:

$$BCE = -(y \log(p) + (1 - y) \log(1 - p)) \tag{1.13}$$

- **Entropia Încrucișată Categorială (CCE)** reprezintă o generalizare a funcției de cost binare, fiind aplicabilă în situațiile în care numărul de clase vizate, notat cu M, este strict mai mare decât 2. Aceasta este dedicată problemelor de clasificare multiclasă și operează în mod optim atunci când stratul final generează predicții folosind funcția de activare Softmax [2]:

$$CCE = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{1.14}$$

Propagarea înapoi - Backpropagation

Algoritmul de propagare înapoi a erorii (*backpropagation*) constituie mecanismul fundamental pentru antrenarea rețelelor neuronale multistrat.

Acesta aplică regula lanțului (*chain rule*) pentru a evalua gradientul funcției de pierdere în raport cu parametrii modelului [11]. Semnalul de eroare astfel calculat este propagat în sens invers, de la stratul de ieșire către straturile ascunse, permițând actualizarea sistematică a ponderilor și facilitând capacitatea rețelei de a aproxima funcții neliniare complexe [11].

Fie o rețea neuronală definită prin funcția $\mathbf{f}[\mathbf{x}, \phi]$, care primește un vector de intrare multivariabil $\mathbf{x}$, este controlată de setul de parametri ϕ și este structurată pe două straturi ascunse, notate cu $\mathbf{h}_1$ și $\mathbf{h}_2$, ilustrată în Figura 1.3:

$$\begin{aligned} \mathbf{h}_1 &= \mathbf{a}[\beta_0 + \mathbf{\Omega}_0 \mathbf{x}] \\ \mathbf{h}_2 &= \mathbf{a}[\beta_1 + \mathbf{\Omega}_1 \mathbf{h}_1] \\ \mathbf{f}[\mathbf{x}, \phi] &= \beta_2 + \mathbf{\Omega}_2 \mathbf{h}_2 \end{aligned} \tag{1.15}$$

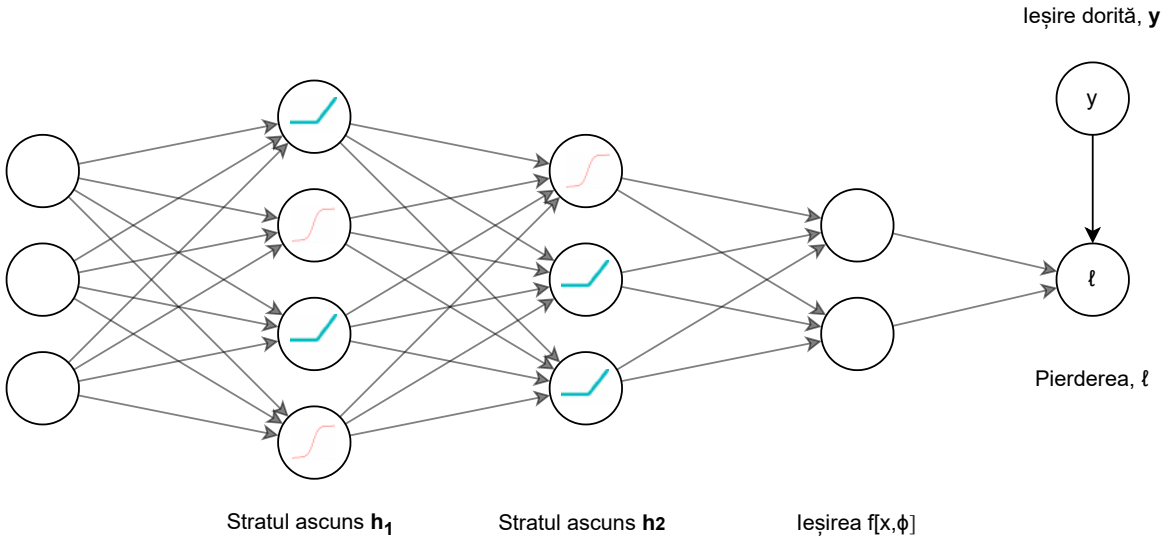

Figura 1.3: Rețea neurală cu 2 straturi ascunse: include funcțiile de activare ReLu și Sigmoid [2].

În această formalizare, operatorul $\mathbf{a}[\bullet]$ indică aplicarea funcției de activare în mod independent pe fiecare componentă a vectorului. Spațiul parametrilor optimizabili ai modelului, definit prin mulțimea:

$$\phi = \{\beta_0, \mathbf{\Omega}_0, \beta_1, \mathbf{\Omega}_1, \beta_2, \mathbf{\Omega}_2\}$$

cuprinde vectorii asociați termenilor liberi (*biases*) β_k, împreună cu matricile de greutăți (*weights*) $\mathbf{\Omega}_k$. Aceste elemente parametrizează transformările liniare aplicate datelor la trecerea prin fiecare strat computațional al arhitecturii.

Pentru antrenarea acestor arhitecturi neuronale, vom defini gradientul descendent stochastic (SGD). Acesta ajustează iterativ parametrii modelului aplicând următoarea regulă de actualizare [12]:

$$\phi_{t+1} \leftarrow \phi_t - \alpha \sum_{i \in \mathcal{B}_t} \frac{\partial \ell_i[\phi_t]}{\partial \phi}, \tag{1.16}$$

În această formulare, α reprezintă rata de învățare (*learning rate*), mulțimea $\mathcal{B}_t$ stochează indicii eșantioanelor de antrenare incluse în lotul curent la iterația t, iar ℓ_i definește funcția de pierdere (*loss*) evaluată strict pentru a i-a instanță de date. Mai exact, ℓ_i cuantifică eroarea individuală dintre predicția rețelei și valoarea reală dorită pentru un singur exemplu specific din setul de antrenare.

Pentru a putea procesa această actualizare a parametrilor globali, algoritmul impune calcularea derivatelor parțiale ale erorii în raport cu fiecare parametrii $\{\beta_k, \mathbf{\Omega}_k\}$ asociați fiecărui strat $k \in \{0, 1, \ldots, K\}$, calculul realizându-se individual pentru fiecare index i corespunzător instanțelor din lotul de antrenare:

$$\frac{\partial \ell_i}{\partial \beta_k} \quad \text{și} \quad \frac{\partial \ell_i}{\partial \mathbf{\Omega}_k} \tag{1.17}$$

Pentru a evalua eficient aceste derivate parțiale, se aplică regula lanțului (*chain rule*) din calculul diferențial, principiu matematic ce constituie nucleul algoritmului *backpropagation* [11]. Deoarece arhitectura rețelei presupune compunerea secvențială a mai multor funcții, gradientul erorii globale se obține prin calculul succesiv al derivatelor locale [11].

Introducând notația $\mathbf{z}_{k+1} = \beta_k + \mathbf{\Omega}_k \mathbf{h}_k$ pentru transformarea liniară intermediară (preactivarea stratului, cu convenția $\mathbf{h}_0 = \mathbf{x}$), regula lanțului permite descompunerea gradienților astfel:

$$\frac{\partial \ell_i}{\partial \mathbf{\Omega}_k} = \frac{\partial \ell_i}{\partial \mathbf{h}_{k+1}} \cdot \frac{\partial \mathbf{h}_{k+1}}{\partial \mathbf{z}_{k+1}} \cdot \frac{\partial \mathbf{z}_{k+1}}{\partial \mathbf{\Omega}_k} \tag{1.18}$$

$$\frac{\partial \ell_i}{\partial \beta_k} = \frac{\partial \ell_i}{\partial \mathbf{h}_{k+1}} \cdot \frac{\partial \mathbf{h}_{k+1}}{\partial \mathbf{z}_{k+1}} \cdot \frac{\partial \mathbf{z}_{k+1}}{\partial \beta_k} \tag{1.19}$$

În acest sistem, produsul primelor două derivate determină modul în care eroarea este propagată înapoi prin funcția de activare neliniară. Ultimul termen reprezintă derivata componentei liniare în raport cu parametrii optimizabili, reducându-se algebric la vectorul de intrare $\mathbf{h}_k$ în cazul matricelor de greutăți ($\mathbf{\Omega}_k$), respectiv la constanta 1 în cazul termenilor liberi (β_k).

Prin calcularea recursivă a acestor derivate locale, începând de la stratul de ieșire și progresând retrograd către stratul de intrare, algoritmul *backpropagation* previne calculele redundante și determină cu o eficiență computațională maximă gradientul global al funcției de pierdere [11]. Odată ce aceste derivate parțiale sunt extrase pentru întregul ansamblu de parametri, valorile rezultate sunt integrate direct în ecuația de actualizare a algoritmului de optimizare (conform ecuației 1.16). Acest pas marchează finalizarea unei iterații complete de antrenare, asigurând ajustarea sistematică a ponderilor și a termenilor liberi într-o direcție care conduce la minimizarea treptată a erorii de predicție a modelului.

Evaluarea Performanțelor

Odată ce algoritmul de *backpropagation* și mecanismele de optimizare (precum SGD) au finalizat ajustarea iterativă a parametrilor interni, faza de antrenare a modelului se încheie. Totuși, simpla minimizare a funcției de pierdere pe setul de antrenament nu garantează automat o capacitate de generalizare optimă a rețelei asupra unor instanțe de date complet noi. Prin urmare, pentru a valida eficacitatea reală a arhitecturii în afara mediului de antrenare, este

imperativă utilizarea unor metrici de performanță standardizate.

În ceea ce privește sistemele de recunoaștere automată a vorbirii, acestea generează secvențe de text de lungimi variabile, care nu sunt aliniate perfect cu cadrele semnalului audio de intrare. Din acest motiv, evaluarea se realizează prin metrici fundamentate pe distanța Levenshtein [13], standardul principal în domeniu fiind Rata de Eroare la Nivel de Cuvânt (WER) [5], reprezentată în formula 1.20:

$$WER = \frac{S + D + I}{N} \tag{1.20}$$

unde S reprezintă numărul de cuvinte substituite (înlocuite incorect), D numărul de cuvinte omise, I numărul de cuvinte inserate artificial de model, iar N definește numărul total de cuvinte din transcrierea de referință (textul corect).

Dincolo de simpla cuantificare a acestor penalități prin intermediul metricii WER, optimizarea riguroasă a unui model acustic necesită înțelegerea cauzelor profunde care generează aceste discrepanțe. Conform teoriei statistice a învățării automate, eroarea totală de predicție a unui algoritm pe date noi nu este un monolit, ci poate fi descompusă analitic în trei surse fundamentale:

$$\text{Eroare Totală} = \text{Bias}^2 + \text{Varianță} + \text{Zgomot Inerent} \tag{1.21}$$

În contextul specific al procesării semnalului vocal, aceste trei componente interacționează și limitează performanța sistemului ASR:

- **Zgomotul inerent** reprezintă eroarea ireductibilă a sistemului, generată de calitatea mediului fizic și a senzorilor. În aplicațiile audio, acesta se traduce prin zgomot de fundal puternic, calitatea slabă a microfoanelor, suprapunerea mai multor vorbitori sau pronunții extrem de ambigue pe care nici măcar un evaluator uman nu le-ar putea descifra cu certitudine absolută.

- **Eroarea sistematică (*Bias*)** măsoară deviația dintre predicția medie a modelului și distribuția reală a datelor, fiind principalul indicator al sub-învățării (*underfitting*). Un model ASR cu un *bias* ridicat posedă o arhitectură prea limitată pentru a asimila complexitatea fonetică a limbajului. Practic, modelul pornește de la presupuneri eronate, fiind incapabil să modeleze fenomene acustice complexe, cum ar fi coarticularea sunetelor sau accentele regionale atipice.

- **Varianța** cuantifică sensibilitatea arhitecturii la fluctuațiile minore din setul de date de antrenament, fiind direct responsabilă pentru fenomenul de supra-învățare (*overfitting*). Un sistem ASR cu o varianță ridicată va memora mecanic particularitățile acustice ale anumitor vorbitori sau „amprenta" microfoanelor din datele de antrenament. Deși va obține un WER excelent pe aceste date, modelul va eșua (generând erori majore de generalizare) atunci când este expus unor voci noi sau unor medii acustice neîntâlnite anterior.

Minimizarea erorii globale în dezvoltarea unui sistem ASR modern presupune gestionarea unui compromis fin între eroare sistematică și varianță, realizat de obicei prin creșterea capacității modelului, regularizare și aplicarea unor tehnici robuste de augmentare a datelor audio.

1.2 Recunoașterea Automată a Vorbirii

Recunoașterea Automată a Vorbirii (ASR) reprezintă o tehnologie de tip multidisciplinară, având la bază elemente din materii precum procesarea de semnal, lingvistica, fizica (acustică)

și informatica. Acest aspect face ca studiul recunoașterii vorbirii să fie catalogat drept dificil, constituind o provocare majoră pentru comunitatea științifică încă de la începuturile erei informatice. Eforturile în aceasta direcție au început în anul 1952, în *Bell Laboratories*, unde a fost construit primul „Sistem de Recunoaștere Automată a Cifrelor", ce putea recunoaște cifrele 0-9 și avea la bază analiza formatelor spectrale [4]. În anii 1960, au apărut și au fost publicate mai multe idei fundamentale în domeniul recunoașterii vorbirii , urmând ca, în anii '70, cercetarea în domeniul recunoașterii vorbirii să atingă o serie de repere semnificative, precum începutul unui efort de echipă de lungă durată în cadrul acestui domeniu folosind un vocabular extins în cadrul *IBM* [4].

De atunci și până acum, această tehnologie a beneficiat de dezvoltări majore, atât pe partea teoretica, cat și pe partea de resurse computaționale. În ceea ce privește partea teoretică, au fost proiectați și dezvoltați algoritmi specializați în recunoașterea vorbirii, pornind de la metodele deterministe, până la cele probabilistice (modele HMM [14]) sau conexioniste (rețele neurale [15]). Totodată, industria tehnologică a beneficiat de investiții majore și dezvoltării masive în puterea de calcul a mașinilor, astfel studiile teoretice din trecut au putut fi implementate în aplicații cu impact imediat. Reamintim apariția Kaldi ASR [16] și Apple Siri în 2011, Amazon Echo în 2014, pana la modelele neurale cu rata de eroare a cuvintelor sub 5% (precum Canary-Qwen 2.5B [17] sau OpenAI Whisper [18]).

Așadar, sistemele de recunoaștere automată a vorbirii, și implicit componentele acestora, au cunoscut o dezvoltare masivă de-a lungul ultimelor decenii. Această evoluție a vizat atât optimizarea tehnicilor de procesare a semnalului vocal, cât și tranziția către cele mai noi arhitecturi de codare și decodare. În secțiunile ce urmează, vor fi detaliate principalele arhitecturi și componente utilizate în studiul acestei lucrări.

Arhitectura Conformer

Recent, modelele bazate pe arhitecturi Transformer [19] și pe rețele neuronale convoluționale (CNN) [20] au arătat rezultate promițătoare în domeniul recunoașterii automate a vorbirii (ASR), depășind performanțele obținute de rețelele neuronale recurente (RNN) [21]. Modelele Transformer excelează în capturarea interacțiunilor globale bazate pe conținut, în timp ce rețelele CNN exploatează în mod eficient caracteristicile locale [21]. Totuși, aceste tipuri de modele își arată limitările când sunt utilizate singure, iar din acest motiv a fost studiată combinarea convoluției cu mecanismele de autoatenție, cercetare care arată rezultate mai bune în acest caz, conform autorilor [21].

Blocul Conformer conține două module Feed Forward ce încadrează într-o structură de tip sandwich modulul Multi-Headed Self-Attention și modulul de Convoluție [21], așa cum este arătat în Figura 1.4. Această structură de tip sandwich este inspirată de Macaron-Net [22], care propune înlocuirea stratului original feed-forward din blocul Transformer cu două straturi feed-forward de jumătate de pas, unul înainte de stratul de atenție și unul după. La fel ca în Macaron-Net, se utilizează ponderi reziduale de jumătate de pas în modulele feed-forward (FFN). Al doilea modul feed-forward este urmat de un strat final de normalizare (*layernorm*). Matematic, acest lucru înseamnă că, pentru intrarea x_i într-un bloc Conformer i, ieșirea y_i a blocului este [21]:

$$\begin{aligned}
\tilde{x}_i &= x_i + \frac{1}{2}\mathrm{FFN}(x_i) \\
x'_i &= \tilde{x}_i + \mathrm{MHSA}(\tilde{x}_i) \\
x''_i &= x'_i + \mathrm{Conv}(x'_i) \\
y_i &= \mathrm{Layernorm}\left(x''_i + \frac{1}{2}\mathrm{FFN}(x''_i)\right)
\end{aligned} \tag{1.22}$$

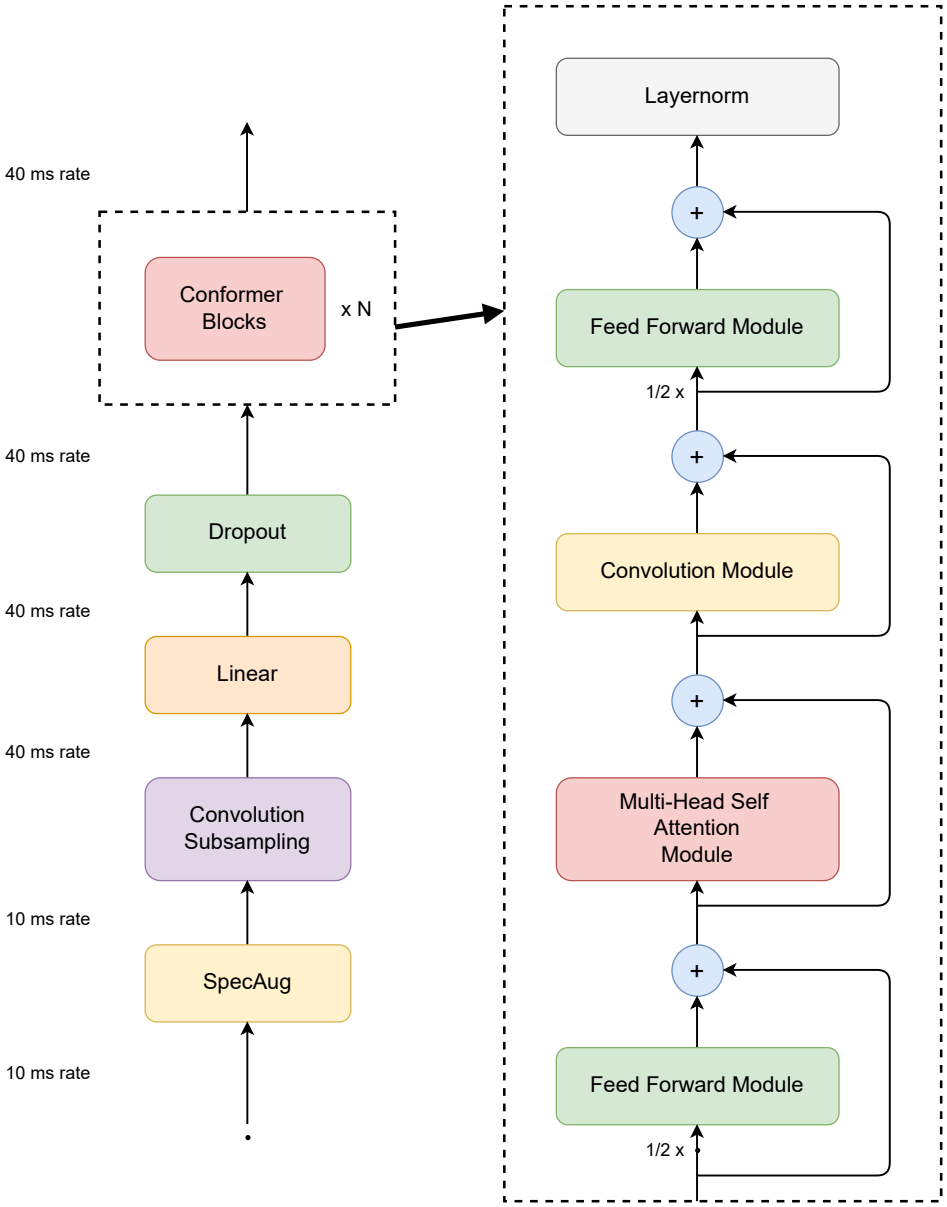

Figura 1.4: Arhitectura modelului Conformer pentru procesarea secvențelor audio [3].

Arhitectura FastConformer

Arhitectura FastConformer reprezintă, în esență, o îmbunătățire a arhitecturii Conformer [23], prezentată anterior. A fost introdusă în 2023, la 3 ani după apariția Conformer, și este concepută pentru a îmbunătăți performantele predecesoarei sale. Conform autorilor [23], schimbările aduse sunt:

1. *Downsampling* de 8x la începutul *encoder*-ului, reducând astfel costul computațional al straturilor de atenție ulterioare de 4 ori [23].
2. Înlocuirea straturilor originale de sub-eșantionare convoluțională cu convoluții separabile pe adâncime (*depthwise separable convolutions*) [23] [24].
3. Reducerea numărului de filtre convoluționale din blocul de *downsampling* la 256 [23].

4. Reducerea dimensiunii nucleului convoluțional (*kernel*) la 9 [23].

Aceste schimbari sunt ilustrate în Figura 1.5.

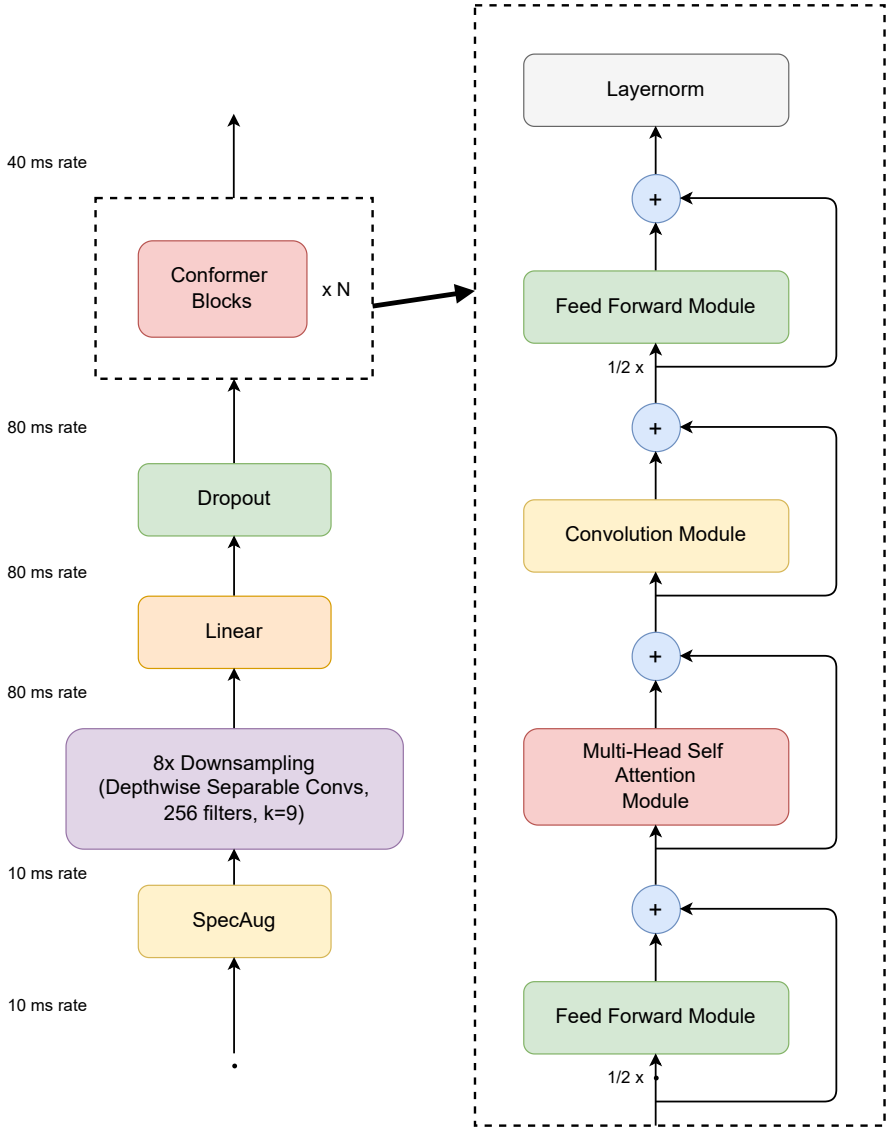

Figura 1.5: Arhitectura modelului FastConformer, cu schimbările aduse față de arhitectura Conformer.

Arhitectura FastConformer este clasificată pe două direcții fundamentale: numărul total de parametri ai modelului și algoritmul decodorului utilizat pentru generarea predicțiilor textuale. În tabelele 1.1 și 1.2 sunt detaliate principalele versiuni ale arhitecturii FastConformer, structurate în funcție de nivelul de scalabilitate și de tehnologia de decodare integrată.

In proiectul de fată, vom folosi această arhitectură întrucât aceasta reprezintă echilibrul perfect intre resursele computaționale necesare antrenării unui model ASR și performanțele obținute.

Versiune Model	Număr Parametri
FastConformer Small	14M
FastConformer Medium	32M
FastConformer Large	120M
FastConformer XLarge	616M
FastConformer XXLarge	1.2B

Tabela 1.1: Clasificarea modelelor FastConformer în funcție de numărul de parametri.

Model FastConformer	Tip Decodor	Principiu de Funcționare
FastConformer-CTC	Non-autoregresiv	Prezice tokenurile în mod independent pentru fiecare cadru de timp, eliminând ulterior duplicatele [25].
FastConformer-Transducer	Autoregresiv (RNN-T)	Emite textul cadru cu cadru, condiționând fiecare nouă predicție de tokenurile generate anterior [26].
FastConformer-Hybrid	Multi-task (RNN-T + CTC)	Modelul este antrenat simultan cu ambele decodoare, partajând același codificator (encoder) [27].
FastConformer-TDT	Autoregresiv cu durată	Prezice tokenul și determină în mod inteligent peste câte cadre audio se poate sări în siguranță [28].

Tabela 1.2: Clasificarea modelelor FastConformer în funcție de decodorul utilizat și principiul de funcționare.

Funcția de pierdere CTC

Clasificare temporală conexionistă (*Connectionist Temporal Classification*) este un algoritm și o funcție de pierdere utilizată la scară largă în recunoașterea automată a vorbirii [25]. A fost cercetată de către Alex Graves et al., care o definesc prin a fi "o metodă pentru etichetarea datelor secvențiale folosind Rețele Neuronale Recurente (RNN) care elimină necesitatea datelor de antrenament pre-segmentate și a post-procesării ieșirilor, și modelează toate aspectele secvenței în cadrul unei singure arhitecturi de rețea" [25]. Ieșirile rețelei devin probabilități pentru secvențele de etichete posibile. Astfel, avem funcția obiectiv diferențiabilă ce maximizează predicțiile corecte, permițând antrenarea prin propagare înapoi [25].

În mod tradițional, pentru a transcrie vorbirea, este necesară o aliniere directă între sunetul transmis în rețea și literele corespunzătoare. CTC rezolvă problema alinierii secvențiale necunoscute folosind 3 concepte-cheie, ilustrate în Figura 1.6:

1. Eticheta blank (spațiul gol), care este introdusă în cadrul algoritmului și care reprezintă lipsa unei litere valide într-un anumit moment;

2. Eliminarea dublurilor, care se realizează după ce algoritmul prezice o literă de mai multe ori la rând;

3. Ștergerea etichetei blank, care se efectuează la final, după eliminarea tuturor dublurilor din transcrierea sunetului.

Printre avantajele acestei funcții se numără antrenarea eficientă, care permite rețelelor neurale să învețe direct transcrierea unui flux audio, și lipsa necesității unor sisteme complexe bazate pe reguli fonetice. Toate acestea constituie motivul pentru care în acest proiect au fost studiate modele de recunoaștere automată a vorbirii bazate pe arhitectura FastConformer ce au implementată funcția CTC.

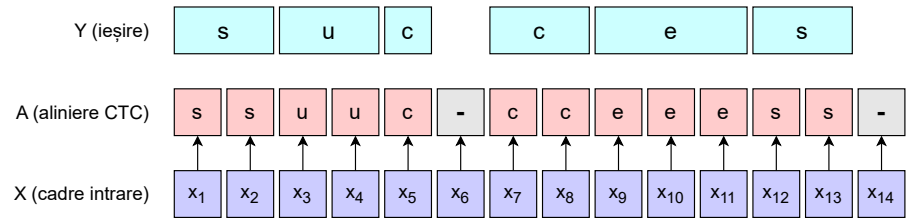

Figura 1.6: Ilustrarea alinierii CTC între cadrele de intrare $X_1 \dots X_{14}$ și secvența de ieșire Y.

Strategia de Decodare

Procesul de decodare este formulat strict ca o problemă de decizie statistică. Conform lucrării fundamentale documentate de Bahl et al., sarcina decodorului este de a găsi șirul de cuvinte W care maximizează probabilitatea *a posteriori* $P(W|A)$, unde A reprezintă semnalul acustic observat [29]. Totuși, calcularea acestei probabilități pentru toate combinațiile posibile de cuvinte dintr-un semnal vocal reprezintă o provocare computațională imensă, iar din acest motiv extragerea secvenței corecte de text necesită implementarea unor algoritmi de decodare eficienți, capabili să aproximeze soluția ideală prin restrângerea informației.

Modelul CTC face o presupunere de independență condițională. Mai exact, presupune că, dată fiind secvența acustică de intrare X, ieșirea la un anumit moment de timp t este independentă de predicțiile de la oricare alt moment temporal [5]. Astfel, probabilitatea unei alinieri complete A de lungime T se calculează ca produsul probabilităților individuale la fiecare pas, utilizând următoarea ecuație:

$$P_{\text{CTC}}(A|X) = \prod_{t=1}^{T} p(a_t|X) \tag{1.23}$$

Datorită acestei proprietăți de independență, cea mai uzuală strategie de decodare este Greedy (Lacomă), care constă în selectarea independentă a etichetei cu cea mai mare probabilitate la fiecare pas temporal. Conform lui Graves et al., "această strategie asignează transcrierea presupunând că cea mai probabilă secvență de etichete corespunde pur și simplu concatenării celor mai probabile ieșiri la fiecare moment de timp" [25], având ca fundament formula deciziei locale:

$$\hat{a}_t = \arg\max_{c \in \mathcal{C}} p_t(c \mid X) \tag{1.24}$$

unde $\hat{a}_t$ reprezintă caracterul ales la pasul temporal t din vocabularul $\mathcal{C}$. Ulterior, alinierea completă formată din aceste decizii locale este trecută prin funcția de colapsare CTC pentru a genera textul final.

Mecanisme de Tokenizare și Modelarea Sub-cuvintelor

În arhitecturile de recunoaștere automată a vorbirii, rețeaua acustică nu prezice direct cuvinte în format text, ci generează o distribuție de probabilități peste un set predefinit de unități

lingvistice discrete. Procesul de conversie a textului de antrenare în aceste unități de bază poartă denumirea de tokenizare.

În mod tradițional, sistemele ASR utilizau tokenizarea la nivel de caracter. Deși această abordare elimină teoretic problema cuvintelor necunoscute, ea atrage după sine expandarea excesivă a șirurilor decodificate. Acest lucru îngreunează capacitatea rețelei de a capta dependențe lingvistice pe termen lung și crește exponențial costul computațional în faza de decodare. La polul opus, tokenizarea strictă la nivel de cuvânt necesită un vocabular de dimensiuni colosale, fiind ineficientă pentru limbi cu morfologie bogată și complet incapabilă să proceseze termeni noi, neincluși în dicționarul de antrenare.

Pentru a depăși aceste limitări, sistemele ASR de moderne implementează tokenizarea la nivel de sub-cuvânt. Standardul actual este reprezentat de algoritmi precum Byte-Pair Encoding (BPE), implementat frecvent prin biblioteci ca SentencePiece [30]. Principiul de funcționare se bazează pe analiza distribuției de frecvențe din corpusul de antrenament. Inițial, vocabularul este alcătuit exclusiv din caractere individuale prezente în setul de date. Ulterior, algoritmul parcurge textul și identifică perechile de simboluri adiacente cu frecvență ridicată, pe care le fuzionează pentru a forma unități de tip token. Acest proces se repetă până când este atinsă o dimensiune prestabilită a vocabularului, tratată ca hiperparametru al modelului, de exemplu 128 pentru 1024 de tokenuri, în funcție de numărul de ore total din înregistrările disponibile în corpusul de date. Principalul avantaj al acestei strategii este echilibrul, întrucât cuvintele uzuale sunt compactate și reprezentate printr-un singur token (optimizând eficiența predicției), în timp ce cuvintele rare sunt descompuse automat în sub-cuvinte cunoscute sau caractere individuale. Astfel, vocabularul rămâne de dimensiuni rezonabile.

Capitolul 2
Metodologia Proiectului

2.1 Corpusul de date *Granary*

Performanta unui sistem modern de recunoaștere automată a vorbirii bazat pe arhitecturi de învățare profundă este fundamental dependentă de volumul, diversitatea și calitatea datelor de antrenament.

Importanța corpusului Granary [31] constă în abordarea concepută special pentru a rezolva problema deficitului de date pentru limbile mai puțin reprezentate. Acest proiect este rezultatul unui efort de cercetare colaborativ între echipele NVIDIA NeMo, Universitatea Carnegie Mellon și Fondazione Bruno Kessler [31].Prin consolidarea și standardizarea mai multor seturi de date sub un singur cadru arhitectural, inițiativa a generat un volum impresionant de aproximativ un milion de ore de vorbire adnotată automat la înaltă calitate. Această resursă masivă acoperă 25 de limbi, integrând atât limbi de largă circulație, precum engleza, franceza sau germana, cât și limbi istoric dezavantajate din punct de vedere al datelor open-source, printre care se numără româna, bulgara, ucraineana, malteza sau slovaca [31].

Până recent, majoritatea corpusurilor open-source disponibile pentru cercetare ofereau un volum de ordinul zecilor sau, în cel mai bun caz, al sutelor de ore de vorbire transcrisă în limba română [32]. Această cantitate este insuficientă pentru a optimiza eficient milioanele de parametri ai unei rețele neuronale (precum arhitectura FastConformer). Pentru a depăși acest obstacol, abordarea propusă în prezenta lucrare se bazează pe explorarea noului corpus de date Granary, lansat de NVIDIA în august 2025, care reprezintă cel mai mare volum de date disponibil pentru limba romana, cu aproximativ 17 mii de ore de înregistrări transcrise automat.

Dezvoltat ca parte a inițiativelor de a extinde suportul pentru limbile cu resurse limitate, corpusul Granary utilizează tehnici de adnotare automată pentru a procesa volume masive de conținut audio, folosind un flux de lucru integrat care asigură o înaltă calitate a datelor extrase. Această prelucrare debutează cu segmentarea fișierelor audio de lungă durată. Prin utilizarea algoritmilor de detectare a activității vocale (VAD) în combinație cu tehnici de aliniere forțată, sistemul decupează cu precizie segmente cu o lungime și o claritate optime pentru faza de inferență [31].

Transcrierea propriu-zisă a acestor fragmente se realizează printr-un proces de inferență în două etape, bazat pe modelul acustic Whisper-large-v3 [18]. Pentru a garanta relevanța datelor pentru limba vizată, acest pas este dublat de o verificare automată a limbii, asigurând excluderea promptă a segmentelor în care vorbitorii folosesc o limbă străină. Mai mult, pentru a contracara erorile inerente ale adnotării automate, se aplică mecanisme stricte de filtrare a calității [31]. Acestea au rolul de a elimina „halucinațiile" rețelei (situații în care modelul inventează text pe fond de zgomot), caracterele invalide și înregistrările foarte slabe din punct de vedere acustic.

În final, deoarece transcrierile brute generate de modelele acustice nu respectă întotdeauna regulile ortografice standard, ultimul pas al acestui flux constă în restaurarea punctuației și a majusculelor. Această operațiune de rafinare este delegată modelului de limbaj Qwen-2.5-7B [17] [31], care normalizează și corectează sintactic textul. Prin parcurgerea acestei metodologii riguroase, corpusul Granary reușește să transforme cantități uriașe de date brute în transcrieri finale îmbunătățite, facilitând antrenarea unor noi sisteme robuste.

2.2 Fluxul de procesare al datelor

Pentru a gestiona eficient volumul masiv de date aferent limbii române, însumând aproximativ 17.000 de ore de înregistrări, a fost necesară implementarea unui flux experimental de procesare a datelor (*data pipeline*). Deși arhitectura NeMo dezvoltată de NVIDIA oferă un utilitar standardizat pentru prelucrarea corpusurilor audio (descris in subsectiunea 2.3.1), explorarea aprofundată și analiza statistică a setului de date au evidențiat prezența unor segmente reziduale. Prin urmare, a fost necesară proiectarea și dezvoltarea unui flux suplimentar de procesare, conceput special pentru a identifica și corecta erorile nesoluționate de prima fază automată. Obiectivul principal al acestei etape avansate este garantarea unui grad maxim de fidelitate al transcrierilor în raport cu vorbirea efectivă din înregistrări, vizând normalizarea gramaticală și purificarea lexicală a textului. Această abordare asigură obținerea unui format optimizat, ușor de asimilat de către modelul de recunoaștere automată a vorbirii, eliminând ambiguitățile lingvistice și facilitând o convergență stabilă în timpul fazei de antrenament. Procesul iterativ de filtrare descris în subsecțiunile de mai jos este ilustrat în Figura 2.1.

Importul datelor

Fluxul de procesare debutează cu importul manifestului brut, denumit `ro_asr.jsonl`, care conține mapările inițiale dintre fișierele audio și transcrierile aferente. În această fază incipientă de inițializare și extragere a metadatelor, o primă operațiune necesară este rescrierea căilor către fișierele audio, o procedură ce asigură compatibilitatea infrastructurii de stocare cu mediul de antrenament. Imediat după restabilirea căilor de acces, sistemul calculează și adaugă durata exactă pentru fiecare segment audio, în secunde. Această valoare temporală influențează direct modul de funcționare și desfășurare a procesului de antrenare.

Normalizarea datelor

Ulterior, corpusul de date este supus unui proces riguros de normalizare textuală, etapă esențială pentru particularitățile lingvistice ale limbii române. Obiectivul central al acestei faze este simplificarea spațiului de ieșire al rețelei neuronale și minimizarea ambiguităților lexicale. În primă instanță, numeralele sunt convertite din reprezentare cifrică în formă textuală extinsă, o operațiune necesară deoarece modelul acustic trebuie să asimileze corespondența fonetică a cuvintelor, nu reprezentarea lor simbolic-matematică.

Concomitent, întregul set de transcrieri este convertit în litere minuscule, o strategie prin care reducem complexitatea și dimensiunea vocabularului. Un element important pentru prelucrarea limbii române îl constituie standardizarea diacriticelor întrucât această procedură previne fragmentarea vocabularului cauzată de codări inconsecvente și garantează uniformitatea deplină a textului. Etapa de normalizare se finalizează prin eliminarea caracterelor non-standard și a simbolurilor irelevante, purificând astfel transcrierile de elemente care ar putea genera zgomot sau perturbații în predicțiile modelului.

Filtrarea datelor

Odată finalizată normalizarea textuală, fluxul de procesare avansează către etapa de filtrare a datelor și reducere a zgomotului. În această fază, sunt aplicate filtre stricte pentru a garanta prezența exclusivă a discursului în limba română și pentru a elimina fragmentele dominate de zgomot de fond accentuat. Această validare lingvistică riguroasă a fost necesară deoarece modelul Whisper [18], utilizat pentru generarea transcrierilor automate preliminare, a produs ocazional erori de identificare a limbii, incluzând în corpusul țintă pentru limba română segmente transcrise în alte limbi.

Un alt criteriu de filtrare esențial aplicat în această etapă îl reprezintă analiza ratei de vorbire (cadența cuvintelor pe secundă). Pragurile de acceptabilitate au fost configurate între 0,3 și 5 cuvinte pe secundă, aceste valori fiind calibrate specific pentru ritmul oratoric caracteristic înregistrărilor provenite din Parlamentul European. Prin urmare, eșantioanele cu o cadență inferioară limitei de 0,3 sunt excluse, fiind asociate cu zgomotul ambiental. Reciproc, segmentele care depășesc pragul superior de 5 cuvinte pe secundă sunt de asemenea înlăturate deoarece indică, în majoritatea cazurilor, erori severe de transcriere automată. Includerea acestor anomalii în setul de antrenament ar diminua capacitatea modelului de a învață și generaliza alte înregistrări audio.

Partiționarea datelor

În cele din urmă, gestionarea unui volum de date de o asemenea magnitudine impune utilizarea unor tehnici avansate de partiționare pentru a face antrenamentul fezabil din punct de vedere computațional. Prin integrarea framework-ului Lhotse, datele sunt supuse unui proces de partiționare (*sharding*), fiind fragmentate în blocuri mai mici, ușor de încărcat secvențial sau paralel în memoria sistemului [33]. Tot în această ultimă fază de pregătire are loc estimarea și crearea grupelor de date (*buckets*), care reunesc fișierele audio cu durate similare. Această grupare strategică minimizează operațiunile de completare artificială cu zerouri în cadrul tensorilor, maximizând astfel eficiența memoriei hardware și viteza de procesare. Odată finalizat acest flux de transformare, filtrare și organizare, datele sunt pregătite pentru a fi introduse în arhitectura rețelei neuronale pentru faza finală de antrenament.

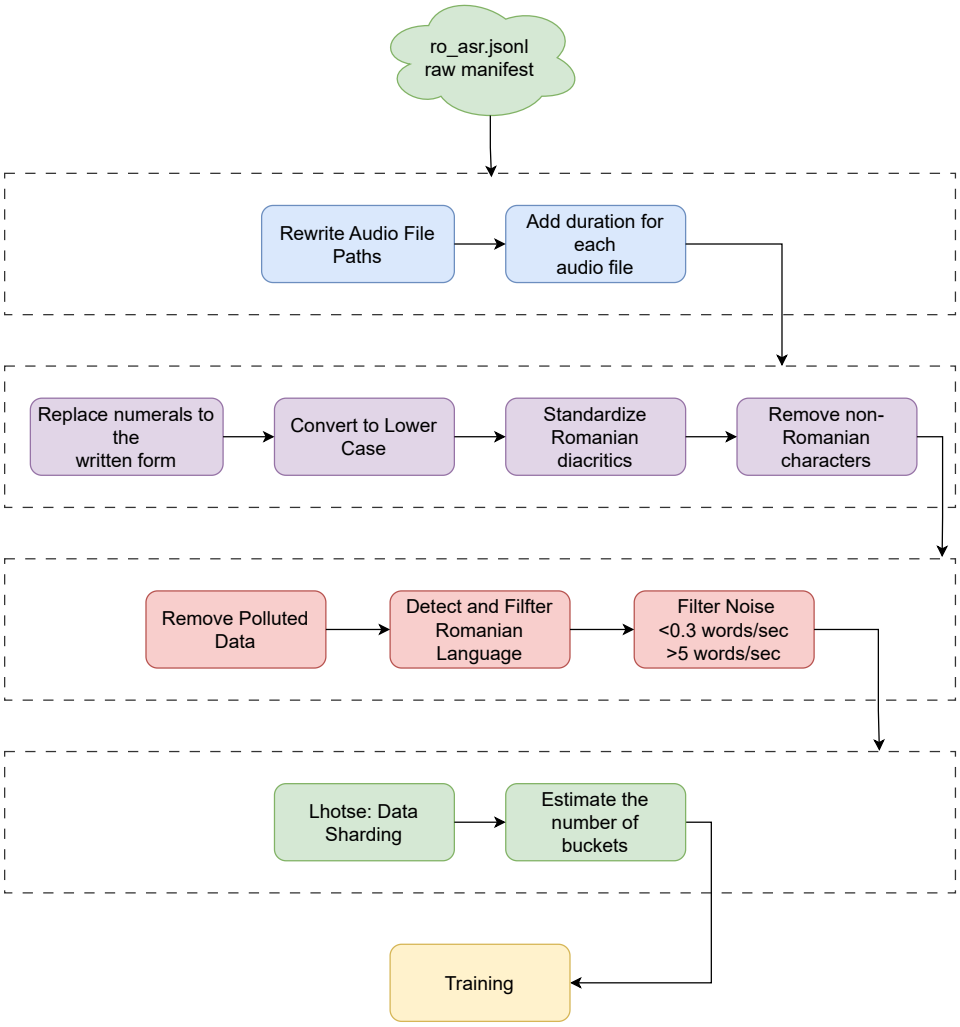

Figura 2.1: Procesul de pregătire, normalizare și filtrare al datelor în fișierul de manifest.

2.3 Configurarea Mediului de Lucru

Dezvoltarea sistemelor moderne de recunoaștere automată a vorbirii impune utilizarea unor versiuni specifice și interdependente de biblioteci software. Pentru a preveni conflictele dintre pachete și pentru a asigura reproductibilitatea experimentelor, s-a configurat un mediu virtual dedicat utilizând Anaconda. Această abordare metodologică izolează interpretorul Python și dependențele proiectului de restul sistemului de operare.

Necesitatea utilizării unui mediu Conda derivă din complexitatea pachetelor de învățare automată utilizate. Managerul de pachete Conda nu gestionează doar modulele Python, ci deține și capacitatea de a compila și instala dependențe la nivel de sistem (non-Python), cum sunt bibliotecile de procesare a semnalelor acustice. Mai mult, acesta permite instalarea directă a versiunilor compatibile de CUDA Toolkit în interiorul mediului virtual. Acest aspect asigură alinierea nativă între codul sursă și driverele hardware ale unității de procesare grafică (GPU), eliminând erorile de compatibilitate la nivelul nucleelor de calcul. Diagrama 2.2 ilustrează la nivel mare arhitectura mediului de execuție bazat pe Conda și modul în care acesta integrează dependințele în cadrul unui mediu virtual.

În interiorul acestui mediu izolat am instalat suita NVIDIA NeMo (*Neural Modules*), un

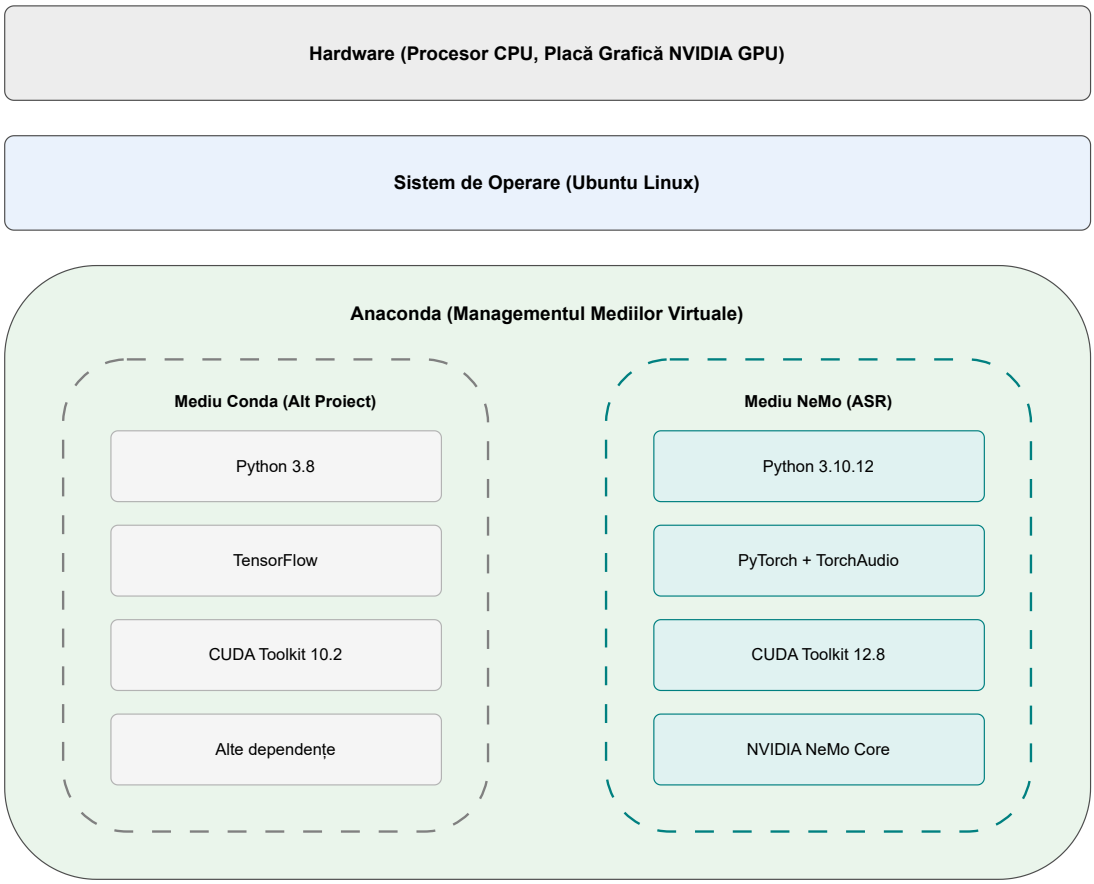

Figura 2.2: Medii Conda izolate, evidențiind separarea mediului și a dependințelor proiectului curent față de alte proiecte sau experimente pe același sistem de calcul.

framework specializat, construit peste PyTorch, utilizat pentru dezvoltarea aplicațiilor AI [3]. NeMo pune la dispoziție implementări optimizate pentru arhitecturile de ultimă generație din familia Conformer și FastConformer, reducând complexitatea structurală a codului prin utilizarea unor module preconfigurate. Toolkit-ul gestionează nativ fluxul de procesare specific ASR, de la încărcarea fișierelor audio și extragerea spectrogramelor Mel, până la aplicarea funcției de pierdere CTC și decodarea secvențelor de text.

Pe lângă antrenarea propriu-zisă a modelelor acustice oferită de suita NeMo, o etapă critică în obținerea unor performanțe ridicate o reprezintă calitatea și curățarea datelor textuale și audio utilizate, proces care necesită instrumente specializate de filtrare la scară largă, introduse în subsecțiunea de mai jos .

2.3.1 NVIDIA NeMo Curator

NVIDIA NeMo Curator este un set de instrumente modular, accelerat hardware (GPU) și extrem de scalabil, conceput special pentru extragerea, filtrarea și curățarea seturilor masive de date [3]. Utilizând o arhitectură distribuită bazată pe framework-ul Ray, acest modul a fost creat inițial pentru preprocesarea textelor destinate modelelor de limbaj de mari dimensiuni (LLM), fiind ulterior extins pentru a gestiona fluxuri de date multimodale, incluzând imagini, video și audio.

În contextul sistemelor de recunoaștere automată a vorbirii, NeMo Curator gestionează cap-coadă fluxul de pregătire a datelor acustice, automatizând transformarea unor corpusuri

brute în seturi de antrenare. Arhitectura sa de procesare a vorbirii execută o succesiune de etape esențiale: validarea și încărcarea eficientă a fișierelor audio, inferența ASR (pentru a genera transcrieri automate sau pseudo-etichete folosind modele pre-antrenate), evaluarea calității (prin estimarea ratei de eroare a cuvintelor - WER și filtrarea anomaliilor de durată sau format) și, în final, conversia metadatelor într-un format standardizat. Prin intermediul acestui flux, NeMo Curator garantează că rețeaua acustică va fi antrenată exclusiv pe date reprezentative, minimizând influența negativă a zgomotului inerent. Figura 2.3 ilustrează procesele îndeplinite de NeMo Curator, descrise anterior.

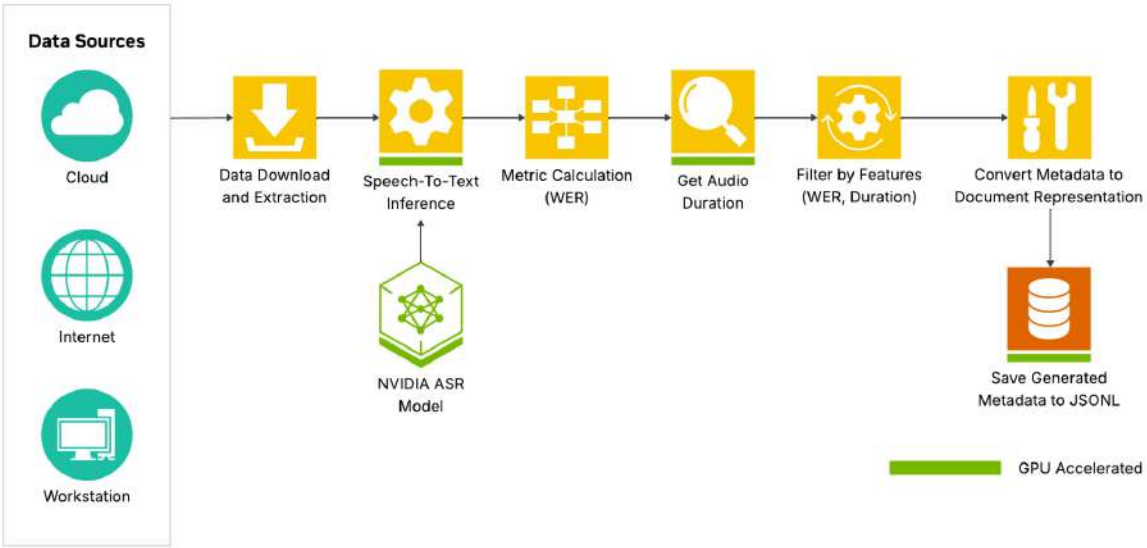

Figura 2.3: Fluxul de pregătire al datelor folosind NeMo Curator [3].

În cadrul acestui proiect, am optat pentru preluarea setului de date Granary procesat de NVIDIA cu NeMo Curator. Astfel, corpusul audio brut a fost supus unui flux de lucru standardizat, implementat la nivelul infrastructurii NVIDIA, care a inclus etapele de validare structurală a fișierelor, transcriere automată cu modelul Whisper v3-Large [18], filtrare acustică bazată pe durata și calitatea fiecărui eșantion, precum și eliminarea segmentelor considerate nesigure din punctul de vedere al raportului semnal-zgomot. Rezultatul acestui proces a constat într-o colecție de perechi audio-text, organizate sub forma unor fișiere manifest în format JSONL (JavaScript Object Notation Lines).

2.3.2 Nvidia NeMo Speech Data Processor

Înainte ca un model acustic să poată fi antrenat, datele brute provenite din diverse corpusuri publice necesită o standardizare riguroasă, deoarece fiecare set de date prezintă o structură proprie a directoarelor și convenții diferite de adnotare. Pentru a unifica această etapă, ecosistemul utilizat integrează NVIDIA NeMo Speech Data Processor (SDP), un instrument software conceput exclusiv pentru automatizarea pregătirii seturilor de date destinate recunoașterii automate a vorbirii. Obiectivul său central este de a minimiza efortul de programare repetitivă (*boilerplate code*), oferind un flux de lucru standardizat, portabil și ușor de reprodus pe diferite

arhitecturi [3].

Din punct de vedere operațional, SDP utilizează o abordare bazată pe modelarea transformărilor prin clase, denumite *procesoare*. Acestea primesc la intrare o cale către datele audio brute sau un fișier manifest inițial și aplică o serie de operațiuni secvențiale, definite clar într-un fișier de configurare (YAML).

Pentru integrarea instrumentului Speech Data Processor în fluxul de lucru, am descărcat modulul oficial de pe platforma GitHub, asigurând instalarea tuturor dependențelor necesare în mediul virtual Conda configurat anterior. Procesul de configurare propriu-zis s-a realizat prin crearea unui fișier de configurare structurat în format YAML. În interiorul acestui manifest, am declarat în mod explicit ordinea secvențială a procesoarelor utilizate pentru curățarea textului, incluzând modulele de conversie la caractere cu literă mică, substituție prin expresii regulate și filtrare alfabetică, analizate în subsecțiunile care urmează. Fiecare procesor a fost definit ca un obiect independent, având parametrii asociați stocați în mod centralizat.

Normalizarea datelor

Deși ecosistemul NVIDIA NeMo pune la dispoziție instrumentul Speech Data Processor pentru normalizarea fișierelor manifest, lipsa suportului oficial pentru limba română a necesitat dezvoltarea unei soluții personalizate. În cadrul acestui proiect, am realizat etapa de normalizare textuală în mod automatizat, prin implementarea unui script în limbajul Python, utilizând expresii regulate (*regex*) și biblioteca num2words. Obiectivul principal al acestei preprocesări a fost convertirea numeralelor (arabe și romane) direct în textul lor corespondent fonetic.

Analiza setului de date de antrenament indică faptul că numeralele romane sunt utilizate preponderent pentru desemnarea secolelor sau în contexte de tip ordinal generalizat (de tipul „Carol al II-lea” sau „clasa a X-a”). Această observație mi-a permis să evit o substituție globală simplă, care ar fi alterat în mod greșit acronime uzuale sau cuvinte care coincid cu numere romane (precum „cd”, „cv”, „mii”). În schimb, am implementat reguli specifice bazate pe *regex* care urmăresc exclusiv combinațiile lingvistice relevante. Astfel, algoritmul identifică și convertește numerele romane doar atunci când apar într-o structură ce conține cuvântul „secolul” (inclusiv variațiile declinate și acordul cu „al”/„a”) sau în structurile ordinale standardizate ce conțin sintagmele „al ... -lea” și „a ... -a”. Această abordare selectivă a asigurat convertirea numeralelor romane esențiale în text (ex: „secolul douăzeci”, „a zecea”), păstrând intacte abrevierile identice vizual.

În ceea ce privește numerele arabe, am implementat o transformare globală, înlocuind toate secvențele de cifre (ex: „123”) identificabile în text cu reprezentarea lor textuală completă în limba română (ex: „o sută douăzeci și trei”). Ca o etapă finală de rafinare, am introdus o regulă de siguranță pentru a corecta cratimele reziduale apărute în urma normalizării, ștergându-le doar dacă elementul din stânga lor este un număr validat, pentru a asigura o transcriere fluidă (ex: „zecea” în loc de „zece-a”).

Schimbarea Caracterelor cu Majuscula la Litere Mici

Ulterior etapei de normalizare numerică și a numeralelor romane, a fost necesară impunerea unei uniformități asupra întregului corpus de antrenament. Pentru a realiza acest lucru, am utilizat procesorul `SubMakeLowerCase` inclus în modulul `sdp.processors`, instrument pus la dispoziție nativ de suita NeMo Speech Data Processor. Funcționalitatea acestui modul constă în aplicarea unei transformări liniare asupra câmpului de text din fiecare linie a fișierelor manifest JSONL, convertind automat toate caracterele alfabetice majuscule în echivalentul lor cu literă mică.

Decizia de a aplica această procesare derivă dintr-un aspect fundamental al antrenării rețelelor acustice. În contextul recunoașterii automate a vorbirii, un cuvânt pronunțat deține exact

aceeași reprezentare acustică indiferent dacă este scris cu majusculă sau minusculă. Menținerea distincției dintre literele mari și cele mici ar fi dublat în mod artificial dimensiunea alfabetului de bază al modelului, introducând o complexitate computațională redundantă și crescând riscul de fragmentare a probabilităților în timpul deducției CTC.

Integrarea acestui procesor s-a realizat direct în fișierul de configurare YAML al suitei SDP, fiind declarat ca primul element din lista operațiunilor de transformare. Pentru a asigura o rulare eficientă și o procesare rapidă a volumului mare de date, am suprascris parametrii de execuție direct din linia de comandă în momentul lansării scriptului. Astfel, am setat argumentul `+processors.0.în_memory_chunksize=500000` pentru a permite încărcarea și stocarea temporară în memoria RAM a unui bloc extins de jumătate de milion de linii din manifest. Simultan, am configurat argumentul `+processors.0.chunksize=100000`, instruind algoritmul să fragmenteze acest volum și să proceseze în paralel loturi de câte o sută de mii de linii. Această alocare optimizată a resurselor a prevenit blocajele de memorie și a redus semnificativ timpul total de execuție a conversiei.

Normalizarea Diacriticelor și Filtrarea Alfabetului

După conversia textului la minuscule, ultimul pas al preprocesării a vizat standardizarea setului de caractere și eliminarea definitivă a oricărui zgomot rezidual din transcrieri. Pentru a realiza acest lucru, am utilizat procesorul `SubRegex` din suita SDP, configurându-l cu o listă secvențială de reguli de substituție. O problemă recurentă în corpusurile de limbă română este prezența diacriticelor învechite (cu sedilă: „ş”, „ţ”), astfel, prima acțiune a constat în înlocuirea automată a acestora cu variantele lor corecte și standardizate (cu virgulă: „ș”, „ț”). Ulterior, pentru a preveni extinderea inutilă a alfabetului din cauza numelor proprii străine sau a neologismelor, am mapat o serie de caractere speciale (cu accente sau treme, precum é, ö, ç, š) la literele lor de bază din alfabetul roman. În finalul acestui bloc de procesare, printr-o expresie regulată restrictivă, am eliminat absolut toate semnele de punctuație și simbolurile speciale, păstrând în text doar literele alfabetului românesc, spațiile și cratimele.

Pentru a garanta integritatea absolută a corpusului rezultat, am integrat ca ultim pas procesorul `DropNonAlphabet`. Acesta a funcționat ca o validare de siguranță, verificând fiecare linie de text împotriva unui alfabet strict și predefinit. Orice eșantion audio a cărui transcriere conținea încă simboluri atipice, care ar fi scăpat regulilor anterioare de curățare, a fost eliminat complet din fișierul manifest.

Eficiența și viteza de execuție pentru ambele module au fost maximizate prin suprascrierea parametrilor structurali direct din linia de comandă, alocând complet resursele hardware și optimizând gestionarea loturilor de date. Pentru procesorul de substituție complex (`processors.1`), am setat argumentul pentru utilizarea la maximum a nucleelor de calcul la valoarea minus unu, forțând astfel utilizarea tuturor nucleelor procesorului CPU în paralel. De asemenea, am alocat dimensiunea lotului de linii încărcate simultan în memoria volatilă la jumătate de milion și am configurat dimensiunea sub-lotului de procesare la o sută de mii de linii, manipulând textul fragmentat direct în memoria RAM pentru a preveni blocajele algoritmului *regex*. În mod similar, performanța filtrului alfabetic (`processors.2`) a fost asigurată prin maparea simetrică a setărilor de scalabilitate: activarea paralelismului pe toate nucleele disponibile, setarea limitei de linii stocate temporar în memorie la jumătate de milion de înregistrări și fragmentarea volumului de lucru în loturi de câte o sută de mii de linii pentru fiecare iterație. Prin aplicarea acestei strategii combinate de paralelizare pe ambele niveluri indexate, întregul flux de preprocesare s-a încheiat rapid, generând un manifest stabil și curățat integral de caractere invalide.

Figura 2.4: Exemplu cu interfața grafica a aplicației: sunt disponibile statistici precum numărul de ore, numărul de secvențe audio, dimensiunea vocabularului, numărul de caractere și afișarea lor, precum și anumite distribuții.

2.3.3 Nvidia NeMo Speech Data Explorer

Pentru a garanta calitatea și corectitudinea seturilor de date segmentate, am utilizat utilitarul NVIDIA NeMo Speech Data Explorer (SDE). Acesta este un instrument de vizualizare și analiză interactivă din ecosistemul NeMo, special conceput pentru a facilita explorarea corpusurilor destinate recunoașterii automate a vorbirii [3].

Din punct de vedere operațional, SDE primește ca date de intrare fișierul manifest în format `JSONL`. Odată lansat, utilitarul parcurge fiecare înregistrare din manifest și extrage valoarea din câmpul `duration` pentru fiecare eșantion. Prin însumarea acestora, instrumentul calculează automat durata totală a înregistrărilor din corpus, oferind astfel o imagine de ansamblu precisă asupra volumului de date disponibil pentru subsetul respectiv de antrenament.

Pe baza datelor încărcate, scriptul lansează nativ un server web local construit pe platforma Dash. Acest server randează dinamic o interfață grafică interactivă, accesibilă direct din browser. Prin intermediul acesteia, instrumentul oferă acces la o gamă largă de analize și distribuții statistice, ilustrate în Figura 2.4. Printre detaliile vizibile se numără dimensiunea și conținutul exact al alfabetului, vocabularul total de cuvinte unice din corpus, precum și frecvența de apariție a acestora. De asemenea, SDE generează histograme detaliate care ilustrează distribuția duratelor înregistrărilor, precum și distribuția numărului de cuvinte și de caractere per transcriere. Pe lângă aceste metrici globale agregate, interfața permite navigarea, filtrarea și inspecția individuală a eșantioanelor, oferind acces la redarea audio, vizualizarea formei de undă (*waveform*) și a spectrogramei aferente fiecărei înregistrări. Această validare vizuală a confirmat în mod practic faptul că procesul de segmentare și amestecare (*shuffling*) a decurs perfect, datele fiind astfel pregătite optim pentru faza de învățare automată.

2.3.4 Nvidia NeMo ASR

NVIDIA NeMo este un utlitar open-source, dezvoltat pentru construirea, antrenarea și ajustarea fină a modelelor de inteligență artificială [3]. NeMo facilitează dezvoltarea soluțiilor domenii

precum recunoașterea automată a vorbirii (ASR), procesarea limbajului natural (NLP) și sinteza vocală (TTS). Arhitectura sa este fundamentată pe PyTorch și integrează nativ modulul PyTorch Lightning. Această abstractizare deleagă aspectele de inginerie software de nivel scăzut, precum alocarea memoriei, distribuția antrenării pe multiple unități grafice și salvarea periodică a stărilor intermediare, permițând o concentrare exclusivă pe definirea arhitecturii și optimizarea hiperparametrilor.

Dintre aceste componente, am utilizat exclusiv suita NeMo ASR, dedicată transcrierii semnalului vocal. Acest modul funcționează ca o punte între seturile de date preprocesate și implementarea practică a rețelelor neuronale, acoperind întregul ciclu de viață al unui model acustic. Cadrul de lucru permite antrenarea modelelor de la zero prin inițializarea aleatoare a ponderilor, dar oferă suport și pentru aplicarea tehnicilor de învățare prin transfer. Această capabilitate mi-a facilitat ajustarea fină a unor arhitecturi pre-antrenate pe alte limbi, adaptându-le la specificul fonetic al limbii române, ceea ce reduce timpul de convergență și resursele computaționale necesare. Pe lângă faza de învățare, framework-ul include mecanisme dedicate pentru evaluarea performanțelor, calculând automat rata de eroare la nivel de cuvânt direct pe seturile de validare și testare, precum și suport pentru inferență, utilizând algoritmi de decodare a predicțiilor.

Pentru pregătirea infrastructurii software necesare etapei de modelare acustică, am procedat la instalarea și configurarea cadrului de lucru direct în interiorul mediului virtual Conda. Un pas preliminar și obligatoriu a fost instalarea framework-ului PyTorch. Pentru această componentă, am selectat o versiune strict compatibilă cu platforma CUDA disponibilă pe sistemul de calcul, garantând astfel suportul nativ pentru accelerarea hardware și interacțiunea optimă cu unitatea de procesare grafică (GPU). În paralel cu pachetele NeMo, am instalat dependențele de sistem necesare decodării și manipulării semnalelor audio la nivel de sistem de operare, integrarea bazându-se în principal pe biblioteca `libsndfile` și pe utilitarul `ffmpeg`. În final, instalarea propriu-zisă a ecosistemului s-a realizat prin intermediul managerului de pachete `pip`.

2.3.5 Lhotse

Inițial, procesul de antrenare a fost demarat utilizând eșantioanele audio în formatul lor original, cu extensia `.ogg`. Cu toate acestea, din cauza volumului masiv de date, constând în sute de mii de fișiere de mici dimensiuni, am observat o ineficiență severă la nivel hardware. Analiza performanței a relevat existența unor timpi morți semnificativi, cunoscuți drept blocaje de intrare-ieșire (*I/O bottlenecks*). Această problemă s-a reflectat printr-o fluctuație extremă a gradului de utilizare a plăcii video, care oscila constant între 0% și 100%. Instabilitatea era cauzată de faptul că acceleratorul grafic procesa rapid tensorii, dar intra ulterior în stare de așteptare până când procesorul și unitatea de stocare reușeau să localizeze, să citească și să decodeze următorul lot de fișiere audio independente.

Pentru a elimina această limitare și a asigura un flux continuu de date către unitatea de procesare grafică, am căutat o metodă de împachetare a datelor într-un format compact. Soluția integrată a constat în utilizarea Lhotse, o bibliotecă Python avansată, susținută nativ în ecosistemul NVIDIA NeMo, concepută special pentru manipularea eficientă a corpusurilor multimodale și de voce [33]. Această bibliotecă abstractizează modul în care rețelele neuronale interacționează cu seturile de date, oferind structuri optimizate care înlocuiesc citirea individuală a fișierelor cu o abordare secvențială, de înaltă performanță [33].

Din punct de vedere operațional, mecanismul bibliotecii Lhotse se fundamentează pe trei tipuri de manifeste de date: `RecordingSet` (care gestionează detaliile fizice ale semnalului audio), `SupervisionSet` (care stochează transcrierile) și `CutSet` (care îmbină informațiile și permite manipularea dinamică a eșantioanelor). Prin intermediul acestor structuri, am convertit manifestul NeMo într-un format Lhotse și am inițiat un proces de fragmentare (*sharding*).

Algoritmul preia secvențial fișierele `.ogg` și metadatele lor, le serializează și le grupează în arhive mari și compacte de tip POSIX TAR (*Tarred Datasets* sau formatul *Lhotse Shar*). Procesul descris poate fi vizualizat în Figura 2.5.

Rezultatul acestui proces de conversie a fost transformarea a sute de mii de fișiere mici într-o colecție restrânsă de fișiere `.tar` de dimensiuni fixe, de aproximativ 300 MB fiecare. În timpul antrenării, funcțiile de încărcare (*dataloaders*) ale framework-ului citesc aceste arhive în mod secvențial, extrăgând direct fragmentele audio și transcrierile în memoria RAM fără a mai interoga repetat sistemul de fișiere. Această standardizare structurală a eliminat complet timpii morți generați de operațiunile pe disc, permițând stabilizarea utilizării GPU-ului la o capacitate constantă de operare și reducând drastic timpul total necesar pentru convergența modelului acustic.

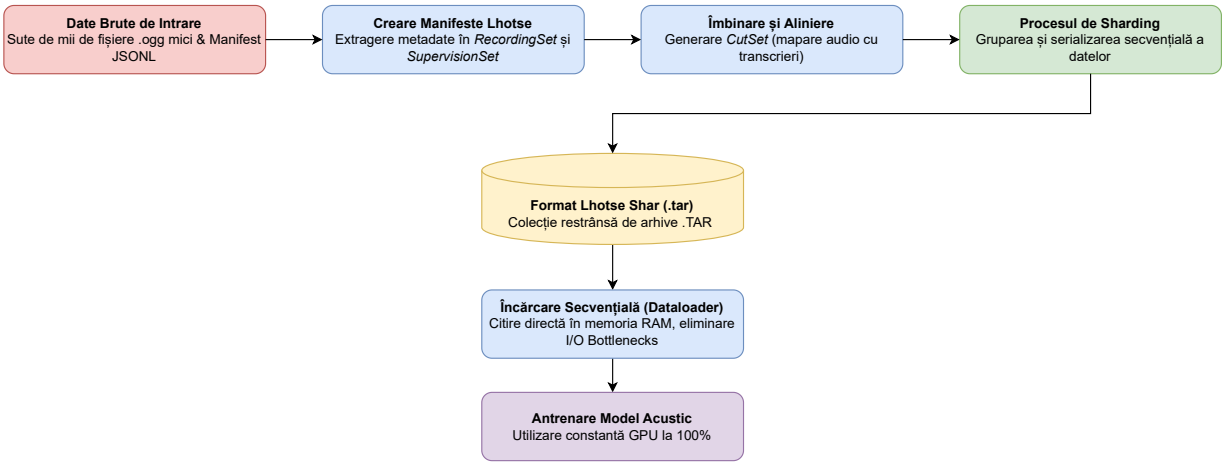

Figura 2.5: Pipeline de pregătire a datelor folosind biblioteca Lhotse.

Pentru a transpune acest concept în practică, am utilizat scriptul dedicat pus la dispoziție în suita NeMo. Acest proces de conversie și împachetare a fost aplicat în mod sistematic pentru patru subseturi de date pregătite din corpusul studiat, ce vor fi documentate ulterior, corespunzătoare volumelor de 500, 2.500, 5.000 și 10.000 de ore de înregistrări.

Pe lângă generarea arhivelor propriu-zise, o funcționalitate critică a acestui utilitar a fost calcularea metricilor pentru tehnica de grupare dinamică. La finalul procesării fiecărui subset, scriptul a analizat distribuția globală a duratelor și a estimat în mod automat numărul optim de categorii (*buckets*).

2.4 Procesul de antrenare și designul experimental

Pentru a asigura o evaluare reproductibilă a arhitecturii acustice, procesul de antrenare a fost structurat într-o serie de experimente succesive. Obiectivul acestui design a fost izolarea și cuantificarea impactului pe care calitatea datelor, volumul de antrenare, dimensiunea modelului și hiperparametrii îl au asupra performanței sistemului de recunoaștere automată a vorbirii (ASR).

Experimentele au fost organizate în următoarele direcții de evaluare:

1. Testarea de control (500 de ore format brut)

Acest experiment inițial ilustrează performanța arhitecturii *FastConformer Large* utilizând date brute, reprezentând starea corpusului Granary înainte de aplicarea algoritmilor de filtrare statistică. Experimentul utilizeaza o metodă standard de încărcare a datelor cu loturi de dimensiune fixă.

Această rulare stabilește nivelul de referință al proiectului. Fără un model antrenat pe date neprelucrate (conținând zgomot ambiental și erori de aliniere) este dificil de cuantificat obiectiv aportul etapelor de preprocesare. Configurarea antrenamentului pe o durată fixă de 50 de epoci și 5.000 de pași de *warmup* reprezintă o traiectorie de învățare standard în literatură, utilă pentru acest nivel de referință.

2. Evaluarea scalabilității datelor (Subseturile Granary: 500h, 2.500h, 5.000h și 10.000h)

Această suită de experimente este concepută pentru a analiza efectul de scalare în recunoașterea vocală pentru limba română. Obiectivul este observarea evoluției ratei de eroare (WER) odată cu creșterea volumului de date filtrate prin metricile WPS și CPS, menținând constantă arhitectura de 120 milioane de parametri.

Configurarea se bazează pe principiul conform căruia rețelele neuronale profunde necesită volume extinse de date pentru a-și atinge capacitatea de generalizare. Prin activarea bibliotecii *Lhotse*, loturile audio au fost grupate dinamic pe baza duratelor (între 1.000 și 2.000 de secunde per lot). Această metodă, împreună cu ajustarea numărului de pași de antrenare și a acumulatorului de gradienți, a asigurat o utilizare eficientă a memoriei GPU, permițând identificarea punctului în care extinderea setului de date generează o plafonare a performanței.

3. Comparația de referință calitativă (Corpusul CDEP - 2.048 de ore)

Obiectivul acestui experiment este ilustrarea influenței pe care corectitudinea etichetelor textuale o are asupra modelului, comparativ cu efectul creșterii volumului de date. S-a utilizat un set de date cu înregistrări din Camera Deputaților, cu o dimensiune de 2.048 de ore.

Rularea evaluează dacă antrenarea pe un set extins prelucrat automat (Granary) oferă rezultate superioare față de antrenarea pe un set mai restrâns, dar verificat (CDEP). Experimentul a fost configurat pentru 120.000 de pași, utilizând o acumulare a gradienților de 8 pe un lot de 1.000 de secunde. Aceasta asigură o actualizare constantă a ponderilor și oferă o bază de comparație echitabilă cu subsetul de 2.500 de ore din corpusul Granary.

4. Evaluarea dimensiunii modelului (Arhitectura XLarge - 10.000 de ore)

Experimentul testează relația dintre capacitatea parametrică a rețelei și volumul setului de date. Prin tranziția de la varianta *Large* (120M parametri) la versiunea *XLarge* (616M parametri), păstrând setul de date la 10.000 de ore, se evaluează reducerea ratei de eroare determinată exclusiv de scalarea arhitecturală.

Creșterea numărului de parametri permite modelarea unor reprezentări acustice mai complexe, reducând riscul de sub-învățare (*underfitting*) pe seturi mari de date. Datorită limitărilor de memorie video (VRAM) impuse de modelul extins, lotul *Lhotse* a fost redus la 500 de secunde, iar parametrul de acumulare a gradientului a fost crescut la 8 pentru a menține stabilitatea matematică a optimizatorului.

5. Optimizarea ratei de învățare (Studiu comparativ pe arhitectura XLarge)

Acest experiment analizează sensibilitatea procesului de optimizare la ajustarea ratei de învățare (*Learning Rate*). Utilizând aceeași arhitectură *XLarge* și volumul de 10.000 de ore, a fost modificat comportamentul planificatorului.

Configurația testează dublarea ratei maxime de învățare la 2×10^{-4} și reducerea perioadei de *Cosine Annealing* la 160.000 de pași. Scopul este de a observa dacă un pas de actualizare mai mare în faza inițială facilitează depășirea minimelor locale. Scăderea mai rapidă a ratei spre finalul antrenării testează capacitatea modelului de a stabiliza ponderile într-un număr mai redus de iterații.

Pentru trasabilitatea deciziilor experimentale, Tabelul 2.1 centralizează parametrii configurați pentru fiecare etapă descrisă.

Experiment	**(Parametri)**	**Lhotse**	**Grad. Acc.**	**Max Steps**	**Learning Rate**	**Warmup**
1. Default Brut (500h)	Large (120M)	Inactiv	4	50 Epoci	10^{-4}	5.000
2. Granary (500h)	Large (120M)	1000s	4	330.000	10^{-4}	3.000
3. Granary (2.500h)	Large (120M)	2000s	4	330.000	10^{-4}	3.000
4. Granary (5.000h)	Large (120M)	1700s	4	450.000	10^{-4}	3.000
5. Granary (10.000h)	Large (120M)	1700s	4	240.000	10^{-4}	3.000
6. Referință CDEP (2.048h)	Large (120M)	1000s	8	120.000	10^{-4}	3.000
7. Dimensiune (XL 10.000h)	XLarge (616M)	500s	8	240.000	10^{-4}	3.000
8. Optimizare LR (XL)	XLarge (616M)	500s	8	160.000	2×10^{-4}	3.000

Tabela 2.1: Configurațiile hiperparametrilor utilizați pentru fiecare experiment de antrenare.

2.4.1 Configurarea teoretică a hiperparametrilor

Pentru a asigura convergența procesului de antrenare, deciziile de parametrizare s-au bazat pe principii stabilite în domeniu, menținând un echilibru între utilizarea hardware-ului și minimizarea erorii.

Optimizatorul și Regularizarea

Actualizarea ponderilor rețelei a fost realizată cu algoritmul AdamW [34]. Acesta decuplează penalizarea ponderilor (*weight decay*) de calculul gradientului adaptiv, reducând interferențele matematice dintre cele două operații [34]. Acest mecanism asigură o convergență mai stabilă pentru rețelele extinse. În toate experimentele, parametrul de *weight decay* a fost setat la 10^{-3}, funcționând ca o metodă de prevenire a supraînvățării (*overfitting*).

Planificatorul Rețelei de Învățare (Scheduler)

Rata de învățare a fost ajustată dinamic, existând posibilitatea de a selecta între strategiile de tip Noam Annealing și Cosine Annealing, ambele funcții fiind reprezentate comparativ în Figura 2.6. Pentru aceste experimente s-a optat pentru implementarea funcției Cosine Annealing, deoarece această abordare asigură o descreștere stabilă a pasului de gradient până la o valoare minimă definită explicit la finalul procesului, ce permite modelului să se stabilizeze precis într-un minim global.

Metoda implică o etapă inițială de creștere treptată a ratei (*warmup*) până la valoarea maximă (de exemplu, 10^{-4}), prevenind variațiile ample ale gradientului la începutul antrenării. Ulterior, rata scade progresiv pe un profil cosinusoidal până la pragul de 10^{-6}. Această reducere facilitează optimizarea fină (*fine-tuning*) în etapele avansate ale procesului, permițând calibrarea precisă a ponderilor rețelei.

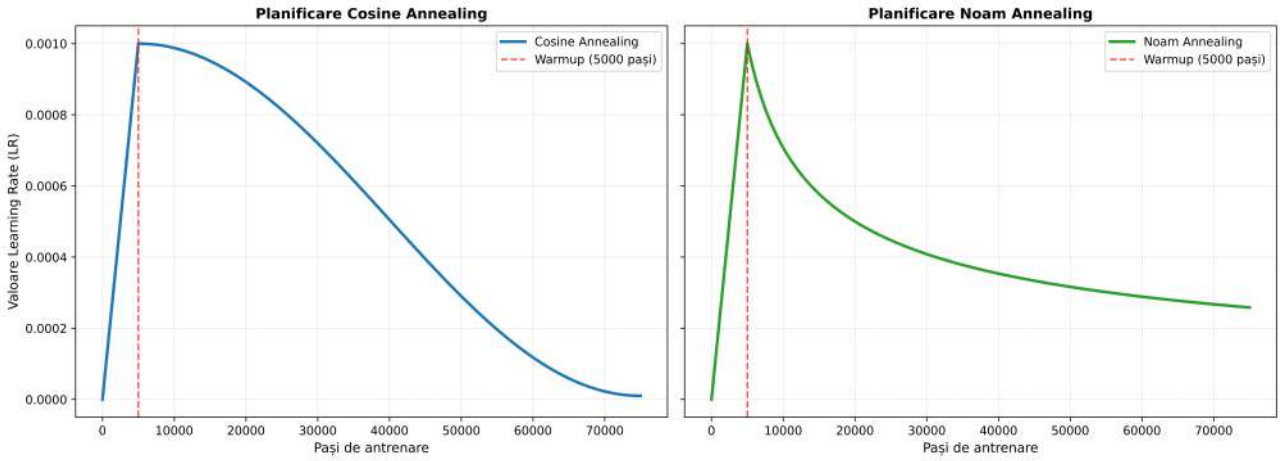

Figura 2.6: Comparație între strategiile de planificare a ratei de învățare (*learning rate*): Cosine Annealing (stânga) și Noam Annealing (dreapta).

Managementul Memoriei: Acumularea Gradienților și Precizia BF16

Restricțiile de memorie video impuse de antrenarea pe un singur accelerator grafic, în special pentru varianta *XLarge*, au necesitat utilizarea unor metode de conservare a resurselor.

Acumularea gradienților este utilizată pentru a depăși limitările stricte de memorie video (VRAM) la antrenarea modelelor neuronale de mari dimensiuni, permițând utilizarea unui lot de date efectiv superior capacității fizice a hardware-ului [35]. În mod standard, actualizarea ponderilor se realizează după propagarea înainte și înapoi a unui singur lot de date. Prin acumularea gradienților, procesul este decuplat: lotul țintă este divizat în micro-loturi procesate secvențial [35]. Algoritmul calculează gradienții pentru fiecare micro-lot și îi adună iterativ într-un acumulator intern, fără a modifica ponderile rețelei. Optimizatorul efectuează o singură actualizare a parametrilor modelului abia după ce un număr predefinit de pași de acumulare a fost atins [35]. Din punct de vedere matematic, adunarea succesivă a gradienților obținuți din micro-loturi este echivalentă cu calculul direct al gradientului pe un lot unificat masiv, menținând stabilitatea matematică a optimizatorului [35]. De exemplu, pentru antrenarea variantei XLarge, limitările memoriei au impus reducerea lotului procesat fizic la doar 500 de secunde audio. Totuși, prin setarea parametrului de acumulare la 8, modelul a fost optimizat folosind un lot efectiv echivalent cu 4000 de secunde audio, asigurând o estimare stabilă a gradientului global.

Calculele au fost realizate utilizând formatul *Brain Floating Point* pe 16 biți [36]. Pentru context, formatul standard de precizie simplă pe 32 de biți (fp32) alocă 1 bit pentru semn, 8 biți pentru exponent (determinând gama dinamică a valorilor) și 23 de biți pentru mantisă (determinând precizia zecimală). Formatul tradițional de jumătate de precizie (fp16) reduce consumul de memorie alocând doar 5 biți pentru exponent și 10 biți pentru mantisă. Deși eficient spațial, reducerea exponentului restrânge sever gama dinamică, expunând modelele profunde la instabilități numerice, precum erorile de tip *underflow* sau *overflow*, în timpul calculului

gradienților. Formatul bf16 oferă un compromis arhitectural optim: trunchiază mantisa la 7 biți, dar păstrează integral exponentul de 8 biți specific formatului fp32 [36]. Această abordare reduce amprenta de memorie GPU cu aproximativ 50% comparativ cu fp32, menținând concomitent gama dinamică necesară pentru a asigura stabilitatea propagării înapoi (*backpropagation*).

2.5 Procesul de evaluare

Pentru a garanta o testare obiectivă a modelelor acustice antrenate, performanța a fost evaluată pe un ansamblu divers de seturi de date independente și verificate manual. Această metodologie previne supraestimarea acurateței pe un domeniu singular și permite analizarea comportamentului rețelei în condiții acustice variate: de la discursuri oratorice prelungite, la vorbire citită în medii controlate și dialoguri spontane caracterizate prin zgomot de fond.

Statisticile descriptive ale fiecărui subset de evaluare au fost extrase și centralizate în Tabelul 2.2. Analiza acestor metrici—în special a duratei medii a eșantioanelor și a ritmului de vorbire (cuvinte pe secundă - WPS)—evidențiază complexitatea specifică a fiecărui domeniu testat.

Set de evaluare	**Eșantioane (N)**	**Durată medie (s)**	**Cuvinte medii**	**Caractere medii**	**WPS mediu**	**CPS mediu**
CDEP	296	60,00	116,0	733,8	1,93	12,23
Fleurs-RO	883	10,31	23,6	146,3	2,30	14,20
RSC	2.504	7,62	16,8	100,3	2,23	12,97
CV-21	3.929	4,20	7,2	42,9	1,77	10,43
USPDATRO	2.637	5,89	14,9	80,6	2,65	13,94
SSC-eval1 (complet)	3.035	4,13	12,0	70,1	2,99	16,87
– *subset clean*	1.935	4,12	12,2	71,5	3,08	17,28
– *subset other*	1.100	4,14	11,6	67,8	2,85	16,16

Tabela 2.2: Statistica descriptivă a seturilor de date utilizate în faza de evaluare

2.5.1 Tipologia seturilor de date

Seturile de testare acoperă trei categorii fonetice principale, fiecare impunând dificultăți tehnice distincte sistemului de recunoaștere:

1. Discursul oratoric

- **CDEP (Camera Deputaților):** Acest corpus cuprinde înregistrări oficiale din ședințele parlamentare. Datele statistice arată că acesta este de departe cel mai extins format temporal testat, cu o durată medie a eșantionului de exact 60,00 secunde. Valoarea scăzută a indicatorului WPS (1,93 cuvinte/secundă) reflectă cadența formală, pauzele retorice și dicția controlată specifice discursului politic. Astfel, CDEP testează în principal capacitatea modelului de a menține un context lingvistic pe o perioadă îndelungată, fără a devia sau halucina.

2. Vorbirea prin citire

- **RSC (Romanian Speech Corpus):** Un set de date curat, dezvoltat în mediu controlat. Având o durată medie de 7,62 secunde și un ritm de 2,23 WPS, acesta reprezintă standardul acustic pentru limba română citită fluent, testând capacitatea de recunoaștere în absența zgomotului de fond.

- **Fleurs-RO:** Parte a corpusului multilingv FLEURS dezvoltat de Google, acesta conține propoziții citite cu o lungime medie de 10,31 secunde. Ritmul de 2,30 WPS confirmă o dicție naturală și stabilă, comparabilă cu cea din RSC.

- **CV-21 (Common Voice):** Ediția de testare din corpusul *crowdsourced* susținut de Mozilla. Indicatorii statistici reliefează un profil atipic: segmentele sunt foarte scurte (4,20 secunde medie), iar ritmul este cel mai lent dintre toate seturile (1,77 WPS). Această viteză redusă indică o citire ezitantă, specifică utilizatorilor non-profesioniști care interacționează cu interfețe de colectare vocale.

3. Vorbirea spontană

- **SSC (Spontaneous Speech Corpus):** Dedicat discursului liber, non-formal. Metrica WPS ilustrează clar natura acestui corpus: pe subsetul curat (*clean*), ritmul atinge 3,08 cuvinte pe secundă, indicând o vorbire extrem de rapidă, cu suprapuneri fonetice frecvente. Subsetul cu zgomot (*other*) prezintă o ușoară scădere a ritmului (2,85 WPS), asociată cu ezitările și degradarea acustică. Evaluarile pe aceste seturi testează robustețea rețelei la coarticulare (pronunțarea contopită a cuvintelor).

- **USPDATRO (Under-Represented Speech Dataset):** Un corpus axat pe dialecte, accente și moduri de exprimare insuficient reprezentate. Cu un ritm de 2,65 WPS și durate medii de aproximativ 6 secunde, acest set acționează ca o punte între vorbirea citită și cea puternic spontană, testând flexibilitatea modelului în fața variațiilor lexicale și a condițiilor acustice suboptime.

Capitolul 3
Rezultate Experimentale

3.1 Analiza Statistică a Setului de Date

3.1.1 Analiza normalizării datelor

Această secțiune este dedicată analizei statistice a setului de date utilizat pentru antrenarea modelului acustic.

Pentru început, am calculat metricile generale, ce se regăsesc în Tabelul 3.1. După cum se poate observa, setul de date este unul de dimensiuni considerabile, însumând aproximativ 17.776 de ore de material audio, împărțite în peste 2,3 milioane de segmente. Durata medie a înregistrărilor de 27.77 de secunde, combinata cu ritmul vorbirii de 1.63 de secunde, dovedește caracterul oratoric al înregistrărilor preluate din Parlamentul European. Dimensiunea vocabularului de peste 900,000 de cuvinte oferă o capacitate mare de generalizare a modelelor, însă este important de menționat faptul ca datele nu sunt normalizate și prelucrate, astfel, pentru aceleași cuvinte de baza, în calcul intră atât cele ce includ majuscula, cât și cele fara. Mai mult, trebuie luat în considerare un eventual zgomot prezent atât în înregistrări, cât și în eventualele transcrieri automate, fapt ilustrat de dimensiunea foarte mare a alfabetului, de 242 de caractere unice.

Metrică Generală	**Valoare**
Segmente totale	2.304.276
Total Ore	17.776 ore
Medie Durată (s)	27,77
Medie Cuvinte (w)	45,2
Medie Word Rate (wps)	1,63
Medie Caractere (c)	279,1
Medie Char Rate (cps)	10,05
Dimensiune Vocabular	902.886
Dimensiune Alfabet	242

Tabela 3.1: Statisticile generale ale setului de date.

Calculul distribuției duratelor înregistrărilor, ilustrat în Tabelul 3.2, reprezintă un pas important pentru a înțelege modul în care vor fi grupate înregistrările. Datele disponibile redau o distribuție asimetrică, dar concentrată a înregistrărilor, în intervalul de 25-30 de secunde, reprezentând 80% din totalul de segmente disponibile în corpus. Acest lucru poate ajuta procesul, fiind nevoie doar pentru aproximativ 20% să se realizeze procesul de egalare artificial al lungimilor pentru un lot.

Totuși, aceleași date ne indica posibilitatea grupării eficiente folosind loturi construite dinamic cu ajutorul `Lhotse` [33]. Baza este reprezentata de cele 80.07% de înregistrări pentru fiecare lot, urmând ca algoritmul să completeze cu restul înregistrărilor de durate mai mici din cadrul corpusului.

Interval Durată	**Număr Segmente (Count)**	**Procent**
10-15 s	2,797	0.12%
15-20 s	250,128	10.85%
20-25 s	201,070	8.73%
25-30 s	1,845,016	80.07%
30+ s	5,265	0.23%

Tabela 3.2: Distribuția duratei segmentelor audio.

Distribuțiile ratei de cuvinte și a numărului de caractere pe secunda sunt direct corelate și redau informații atât despre caracterul de vorbire al înregistrărilor, cat și despre zgomotul sau transcrierile eronate. Astfel, identificarea valorilor extreme în aceste metrici ne permite să izolăm și să eliminăm segmentele cu anomalii, precum transcrierea incompletă sau halucinativă a textului sau prezența unor pauze neobișnuit de lungi.

Valorile din Tabelul 3.5 ilustrează o distribuție simetrică a valorilor în jurul mediei de 1.63 de cuvinte pe secunda, care confirmă caracterul oratoric al înregistrărilor. Așa cum reiese detaliat din date, ritmul predominant de vorbire se situează în intervalul 1.5 - 2.0 wps, care grupează cea mai mare pondere a segmentelor (38,38%). Această concentrare masivă în jurul mediei demonstrează un debit verbal constant pe parcursul majorității înregistrărilor.

Extinzând analiza, se remarcă faptul că o proporție covârșitoare a datelor, 80,01%, este cumulată în plaja de ritmuri moderate, cuprinsă între 1.0 și 3.0 wps (16,59% + 38,38% + 25,04%). Această consistență denotă un discurs controlat, cursiv și clar, specific cadenței din prezentările publice, unde scopul principal este inteligibilitatea mesajului și menținerea atenției auditoriului.

În contrast, extremele distribuției au o pondere marginală, dar care trebuie analizată. Astfel, în urma filtrării secvențelor în funcție de rata de vorbire, am descoperit prezența datelor poluate. Pentru secvențele cu sub 0.3 cuvinte pe secundă, toate transcrierile sunt fie incomplete (modelul a început transcrierea corectă, dar s-a oprit brusc), fie au fost transcrise la început în altă limbă (limbi romanice, asemănătoare cu româna, sau engleză), urmând ca procesul să se încheie brusc.

O excepție notabilă pe segmentul inferior de viteză apare în intervalul 0.6 - 0.7 cuvinte pe secunda (13,79%). Această concentrare secundară atipică poate fi explicată prin natura însăși a stilului oratoric, fiind cel mai probabil reprezentativă pentru segmentele audio ce conțin pauze lungi intre discursuri.

Interval Word Rate	Număr Segmente (Count)	Procent
0.0-0.1 wps	15,939	0.69%
0.1-0.2 wps	11,621	0.50%
0.2-0.3 wps	6,653	0.29%
0.3-0.4 wps	5,240	0.23%
0.4-0.5 wps	4,776	0.21%
0.5-0.6 wps	6,066	0.26%
0.6-0.7 wps	317,766	13.79%
0.7-0.8 wps	21,423	0.93%
0.8-0.9 wps	24,721	1.07%
0.9-1.0 wps	32,283	1.40%
1.0-1.5 wps	382,315	16.59%
1.5-2.0 wps	884,353	38.38%
2.0-3.0 wps	576,956	25.04%
3.0-4.0 wps	4,198	0.18%
4.0-5.0 wps	276	0.01%
5.0+ wps	9,690	0.42%

Tabela 3.3: Distribuția ratei de cuvinte (Word Rate).

Această caracteristică a discursului este reconfirmată și la un nivel de granularitate mai fină, prin analiza ratei de caractere pe secundă (CPS). Întrucât lungimea medie a cuvintelor tinde să se mențină relativ constantă în cadrul aceluiași tip de vocabular instituțional, distribuția CPS este direct corelată cu cea a ratei de cuvinte (WPS). Astfel, metricile referitoare la caractere, prezentate în Tabelul 3.6, validează observațiile anterioare, oglindind aceeași simetrie, observata prin totalul secvențelor din intervalele 5.0–10.0 cps și 10.0–15.0 cps, reprezentând peste 79% din totalul disponibil de înregistrări.

Interval CPS	Număr Segmente (Count)	Procent
0.0-5.0 cps	386.589	16,78%
5.0-10.0 cps	559.647	24,29%
10.0-15.0 cps	1.263.650	54,84%
15.0-20.0 cps	84.220	3,65%
20.0-25.0 cps	214	0,01%
25.0+ cps	9.956	0,43%

Tabela 3.4: Distribuția ratei de caractere.

Datele centralizate în tabelele 3.5 și 3.6 oferă o perspectivă complementară asupra densității informaționale și a complexității structurale a corpusului. După cum este firesc pentru un set de date coerent, se observă o convergență clară între cele două metrici, ambele indicând o concentrare masivă a segmentelor în intervalele mediane.

Mai exact, o proporție de aproximativ 74,48% din totalul înregistrărilor conțin între 25 și 75 de cuvinte, valori care se aliniază proporțional cu distribuția caracterelor, unde majoritatea datelor (aproape 59%) se situează în plaja de 200–400 de caractere per segment.

În același timp, distribuțiile scot în evidență extremele setului de date, vitale pentru etapa de curățare a corpusului. Fragmentele atipice, cum ar fi cele cu peste 400 de cuvinte (o anomalie pentru un segment de 30 de secunde) sau cele cu sub 50 de caractere, deși sunt marginale din punct de vedere procentual, semnalează erori la nivelul datelor brute, așa cum am menționat și mai sus, în cadrul distribuțiilor ratei și a caracterelor pe secundă.

Număr Cuvinte	**Număr Segmente (Count)**	**Procent**
0-5 w	25.830	1,12%
5-10 w	12.976	0,56%
10-15 w	18.724	0,81%
15-20 w	376.870	16,36%
20-25 w	77.688	3,37%
25-50 w	798.882	34,67%
50-75 w	917.233	39,81%
75-100 w	65.736	2,85%
100-400 w	10.335	0,45%
400+ w	2	0,00%

Tabela 3.5: Distribuția numărului de cuvinte per segment.

Număr Caractere	**Număr Segmente (Count)**	**Procent**
0-50 c	32.180	1,40%
50-100 c	24.402	1,06%
100-150 c	420.841	18,26%
150-200 c	175.799	7,63%
200-300 c	536.951	23,30%
300-400 c	820.143	35,59%
400-500 c	270.649	11,75%
500-1000 c	15.110	0,66%
1000+ c	8,201	0,36%

Tabela 3.6: Distribuția numărului de caractere per segment.

Analiza metricilor confirmă faptul ca setul de date de peste 17.700 de ore reprezintă o bază robustă pentru antrenarea modelului acustic. Concentrarea a 80% din înregistrări în intervalul 25–30 de secunde justifică utilizarea utilitarului `Lhotse` pentru gruparea dinamică a loturilor (*dynamic batching*), optimizând astfel memoria GPU și eliminând ineficiența tehnicii clasice de *padding*. Mai mult, corelarea distribuțiilor ratei de cuvinte (WPS) și a numărului de caractere

(CPS) a facilitat auditarea datelor și proiectarea unei filtrari precise a anomaliilor.

Astfel, în următoarea secțiune, voi prezenta etapele de filtrare și preprocesare dezvoltate pe baza acestor analize, urmând să comparam pentru fiecare pas evoluția corpusului de date.

3.1.2 Analiza filtrarii datelor

În vederea optimizării performanțelor modelului acustic, am demarat o serie de analize și antrenări preliminare, urmărind îmbunătățirea treptată a rezultatelor prin aplicarea unor filtre experimentale asupra setului de date. În urma unei investigații mai profunde a corpusului, am constatat prezența unor transcrieri poluate, al căror text nu avea nicio legătură semantică cu materialul audio înregistrat.

Acest fenomen a apărut din motive independente de metodologia noastră de procesare. Modelul *Whisper V3-Large* [18], utilizat pentru transcrierea automată a înregistrărilor brute, a manifestat un comportament de tip „halucinație". Concret, pentru un număr de 352.249 de segmente audio, modelul a transcris eronat secvențe standardizate din mediul online, precum *„NU UITAȚI SĂ DAȚI LIKE, SHARE ȘI SUBSCRIBE"*, ignorând discursul real.

Identificarea și eliminarea acestor segmente corupte a reprezentat un prim pas esențial în rafinarea setului de date. Tabelul 3.7 ilustrează efectul cantitativ al acestei prime etape de curățare asupra corpusului, evidențiind modificările absolute și procentuale ale metricilor generale.

Metrică	**Manifest Brut**	**Eliminare Halucinații**	**Diferență**	**% Schimbare**
Segmente totale	2.304.276	1.952.027	-352.249	-15,29%
Total Ore	17.775,94	14.923,95	-2.851,99	-16,04%
Medie Durată (s)	27,77	27,52	-0,25	-0,90%
Medie Cuvinte (w)	45,2	48,9	3,7	8,19%
Medie Word Rate (wps)	1,63	1,77	0,14	8,59%
Medie Caractere (c)	279,1	303,6	24,5	8,78%
Medie Char Rate (cps)	10,05	10,96	0,91	9,05%
Dimensiune Vocabular	902.886	902.859	-27	0,00%

Tabela 3.7: Impactul eliminării transcrierilor halucinate asupra corpusului.

Analiza datelor din Tabelul 3.7 confirmă eficiența etapei de eliminare a transcrierilor halucinate și a anomaliilor de aliniere. Deși această filtrare a redus volumul total al corpusului cu aproximativ 16% (peste 2.800 de ore eliminate), procesul a condus la o îmbunătățire considerabilă a densității informaționale a setului de date rămas.

Cel mai important aspect reiese din dinamica metricilor per segment: deși durata medie a înregistrărilor a înregistrat o ușoară scădere (de la 27,77 la 27,52 secunde), numărul mediu de cuvinte și caractere, precum și ratele de distribuție ale acestora (WPS și CPS), au crescut semnificativ (cu aproximativ 8,5% – 9%). Acest aspect demonstrează existenta înregistrărilor lungi, dar cu transcrieri scurte și eroante, folosind expresia *„NU UITAȚI SĂ DAȚI LIKE, SHARE ȘI SUBSCRIBE"*.

Alte etape de filtrare care s-au dovedit eficiente sunt reprezentate de `detectarea limbii`

`înregistrărilor` și de `eliminarea segmentelor cu o viteză de vorbire neconformă`. În cadrul corpusului de date au fost identificate manual (vizual) înregistrări transcrise în alte limbi, cele mai frecvente fiind limbile romanice (datorită similitudinilor structurale și fonetice cu limba română) sau limba engleză (datorita prezentei anumitor termeni în engleza în cadrul discursurilor). În ceea ce privește viteza de vorbire, au fost excluse segmentele cu o rată sub 0,3 cuvinte pe secundă (*wps*) sau peste 5 cuvinte pe secundă. Justificarea pragului minim este evidentă în contextul dat: pentru o înregistrare cu o durată medie de 30 de secunde, o rată sub 0,3 cuvinte pe secunda corespunde unui număr de maximum 9 cuvinte, o densitate lexicală nerealist de scăzută pentru ritmul și dinamica unui discurs oratoric din Parlamentul European. Analiza manuală a eșantioanelor cu aceste praguri a confirmat o calitate precară a transcrierii automate realizate de modelul de bază, acesta generând fie un număr excesiv de cuvinte (fenomen de repetiție sau inserție), fie mult prea puține în raport cu semnalul audio real. Păstrarea acestor transcrieri poluate ar fi introdus instabilitate în procesul de optimizare, forțând modelul să învețe corelații false care degradează direct capacitatea de generalizare și cresc rata de eroare.

În urma tuturor procesărilor aplicate asupra corpusului, setul de date a atins o formă optimizată pentru rezultate bune în procesul de antrenare. Tabelul 3.8 prezintă comparația generală între stadiul inițial (manifestul brut) și cel final, ilustrând impactul cumulat al acestor metode asupra metricilor de bază.

Metrică	**Manifest Brut**	**Corpus Final**	**Diferență**	**% Schimbare**
Segmente totale	2.304.276	1.902.536	-401.740	-17,43%
Total Ore	17.775,94	14.523,78	-3.252,16	-18,30%
Medie Durată (s)	27,77	27,48	-0,29	-1,04%
Medie Cuvinte (w)	45,2	50,2	5,0	11,06%
Medie Word Rate (wps)	1,63	1,81	0,18	11,04%
Medie Caractere (c)	279,1	305,6	26,5	9,49%
Medie Char Rate (cps)	10,05	11,04	0,99	9,85%
Dimensiune Vocabular	902.886	509.347	-393.539	-43,59%

Tabela 3.8: Comparația metricilor între corpusul inițial și cel final, după filtrare și normalizare.

Deși aplicarea filtrelor a redus numărul total de segmente cu 17,43% și volumul de date cu 18,30%, acest pas a fost necesar pentru a asigura calitatea datelor de antrenare prin eliminarea eșantioanelor corupte. În ciuda scăderii volumului total, mediile pentru numărul de cuvinte, rata de cuvinte și rata de caractere au crescut cu valori cuprinse între 9% și 11%. Acest lucru se explică prin eliminarea înregistrărilor cu transcrieri incomplete sau eronate, cu prea multe cuvinte transcrise, rezultând un corpus format din discursuri mai dense și cu un flux continuu.

De asemenea, dimensiunea vocabularului a scăzut cu 43,59%. Această diferență majoră reflectă eficiența etapei de normalizare a textului, nu o pierdere a informației. Reducerea a fost realizată prin conversia literelor cu majuscula în litere mici, schimbarea numerelor cu forma lor literara în text, precum și prin eliminarea caracterelor străine sau inexistente în alfabetul limbii române. Astfel, a fost eliminată redundanța care ne asigura o antrenare mai stabilă a modelului.

3.2 Antrenarea folosind corpusul initial

Această secțiune detaliază experimentul preliminar realizat pe un subset de 500 de ore. Pentru a respecta metodologia propusă, la acest nivel s-a folosit corpusul inițial: datele audio au fost păstrate în formatul brut („.ogg”), neprocesate, aplicându-se exclusiv normalizarea la nivelul textului. Este esențial de subliniat că antrenarea a fost validată folosind un set de date extras din același corpus necurățat. Evoluția procesului a fost monitorizată pe parcursul a aproximativ 51.000 de pași.

Evaluările pe acest set de validare brut (ilustrate în Figura 3.1 indică o scădere inițială rapidă a ratei de eroare (WER), care a coborât de la 73,45% (la pasul 1.024) până la aproximativ 20% în jurul pasului 10.000. După această etapă, curba de învățare a intrat într-un platou pronunțat. Performanța maximă a modelului pe datele de validare interne a fost atinsă abia la pasul 50.224, indicând un WER aparent de 14,30%.

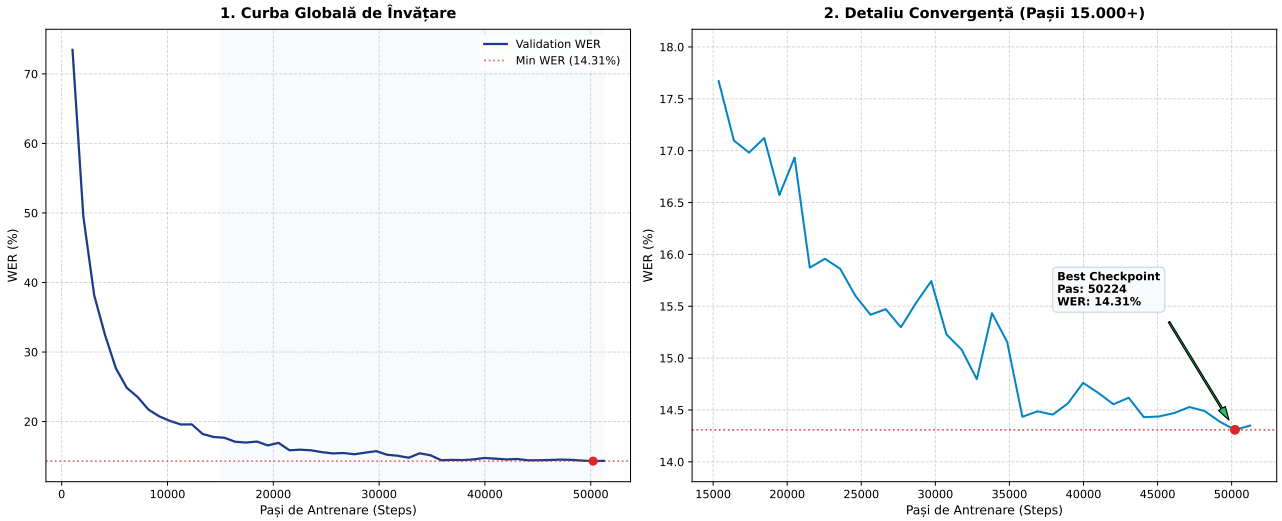

Figura 3.1: Evoluția Word Error Rate (WER) pe setul de validare brut, neprocesat.

Totuși, pentru a evalua performanța reală a acestui model preliminar, el a fost testat pe seturile de evaluare standard externe. Rezultatele, detaliate în Tabelul 3.9, indică o acuratețe extrem de slabă pe toate tipurile de discurs. Ratele de eroare (WER) variază de la 84,56% până la aproape 100%, demonstrând incapacitatea sistemului de a recunoaște corect vorbirea.

Această discrepanță majoră între performanța la validare și evaluarea pe date independente confirmă o vulnerabilitate critică: validarea pe date poluate oferă o falsă impresie de convergență. În realitate, rețeaua neuronală a asimilat zgomotul de fond, erorile de aliniere și eșantioanele corupte specifice corpusului brut, dezvoltând asocieri greșite între semnalul audio și text (supraspecializare pe zgomot). Prin urmare, rezultatele impun aplicarea unui flux riguros de curățare și filtrare a datelor înainte de reluarea experimentelor, pentru a garanta că modelul învață exclusiv trăsături acustice valide.

3.3 Antrenarea cu datele procesate prin fluxul proiectat

3.3.1 Antrenarea folosind subseturi diferite

Pentru a evalua impactul cantității datelor asupra capacității de generalizare a modelului acustic, a fost realizat un studiu comparativ utilizând patru subseturi distincte: 500, 2.500, 5.000 și

Training Dataset	Training Hours	Read		Spontaneous				Oratory		
		RSC	CV-21	SSC-eval1 clean	SSC-eval1	SSC-eval1 other	USPDATRO	Granary	CDEP	Fleurs-RO
Default	500	96,54	84,56	94,25	95,47	97,55	99,61	-	92,69	94,66

Tabela 3.9: Rezultatele evaluării pentru modelul preliminar FastConformer CTC Large antrenat și validat pe date brute (nefiltrate).

10.000 de ore. Toate experimentele au folosit arhitectura FastConformer Large și strategia de decodare Greedy. Pentru aceste evaluări, numărul de pași de antrenare a fost ajustat strategic: experimentele pe subseturile de 500 și 5.000 de ore au fost rulate pentru un număr mai mic de pași cu scopul de a demonstra convergența rapidă inițială, în timp ce antrenările pe 2.500 și 10.000 de ore au fost extinse considerabil pentru a găsi punctul absolut de stabilizare.

O primă analiză a curbelor de învățare indică un fenomen important la nivelul pragului de 30.000 de pași. În această etapă de antrenare, comparând modelele, se observă că cele mai bune rezultate ale ratei de eroare (WER) aparțin subseturilor cu un volum mai mic de date. Fenomenul prezentat este ilustrat în Figurile 3.2 și 3.3.

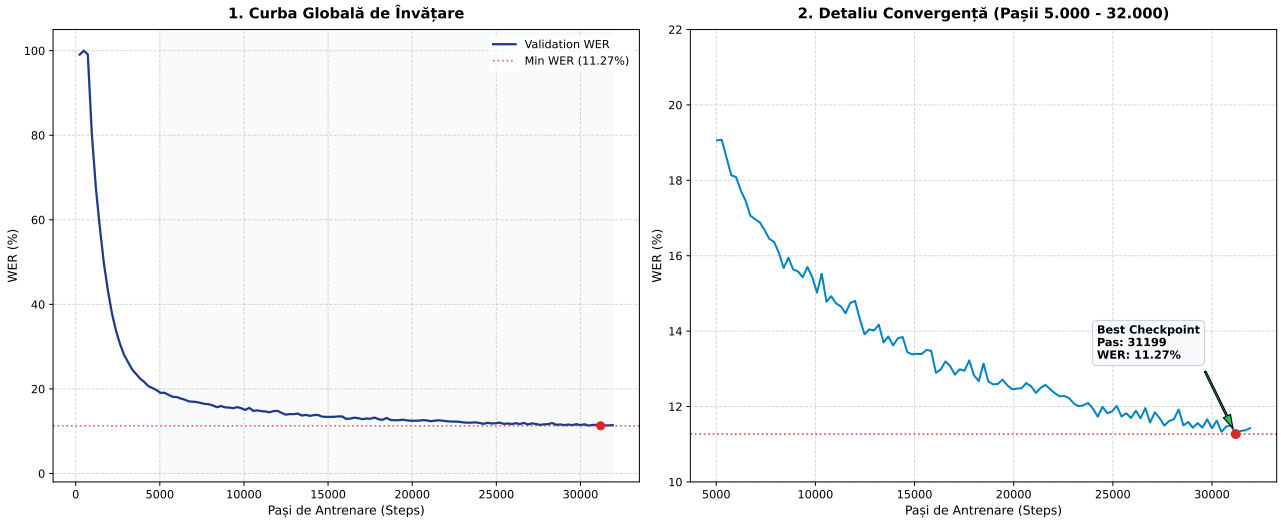

Figura 3.2: Evoluția Word Error Rate (WER) pentru antrenarea cu 500 de ore

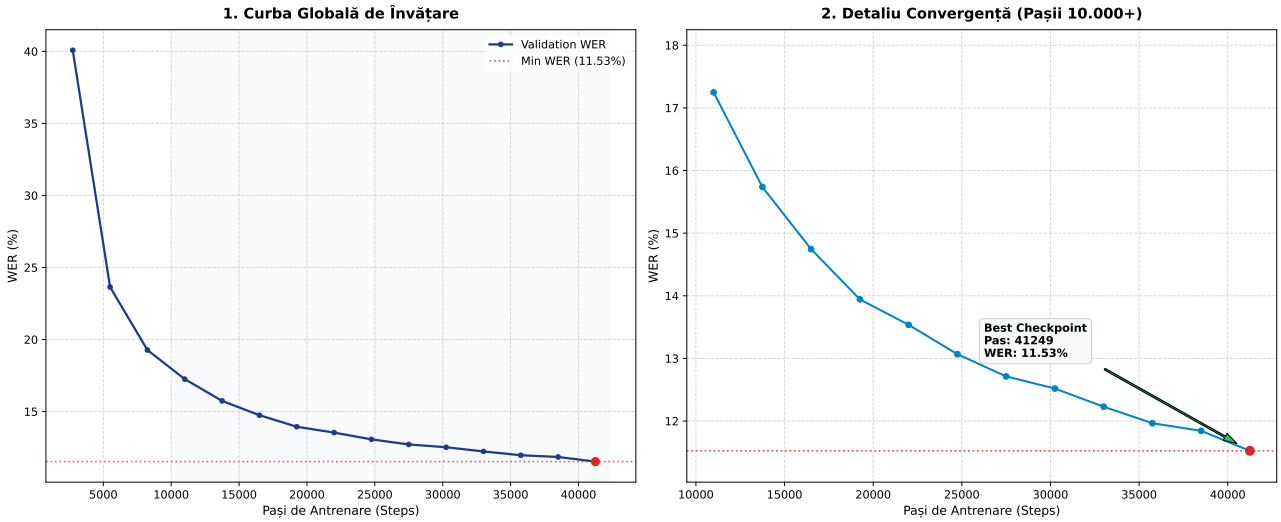

Figura 3.3: Evoluția Word Error Rate (WER) pentru antrenarea cu 5.000 de ore

Totuși, odată cu creșterea substanțială a numărului de pași, avantajul volumului mare de date devine evident, aspect demonstrat de experimentele extinse. Deoarece rulările pentru 500 și 5.000 de ore au fost oprite timpuriu pentru a analiza doar convergența inițială, o comparație a acurateței maxime pe termen lung este relevantă și obiectivă doar între subseturile de 2.500 și 10.000 de ore. Analizând aceste două experimente, se observă că un set de date mai vast permite scăderea erorii la un nivel inferior în etapele avansate de antrenare. Astfel, în timp ce modelul antrenat pe 2.500 de ore a coborât eroarea la 10,03% (la pasul 123.749), cea mai bună acuratețe a fost obținută prin expunerea la diversitatea maximă a subsetului de 10.000 de ore, care a înregistrat un WER minim absolut de 9,39% la pasul 131.944. Această dinamică subliniază importanța scalării datelor pentru atingerea unei generalizări superioare în faza de optimizare fină. Acest detaliu poate fi vizualizat în Figurile 3.4 și 3.5.

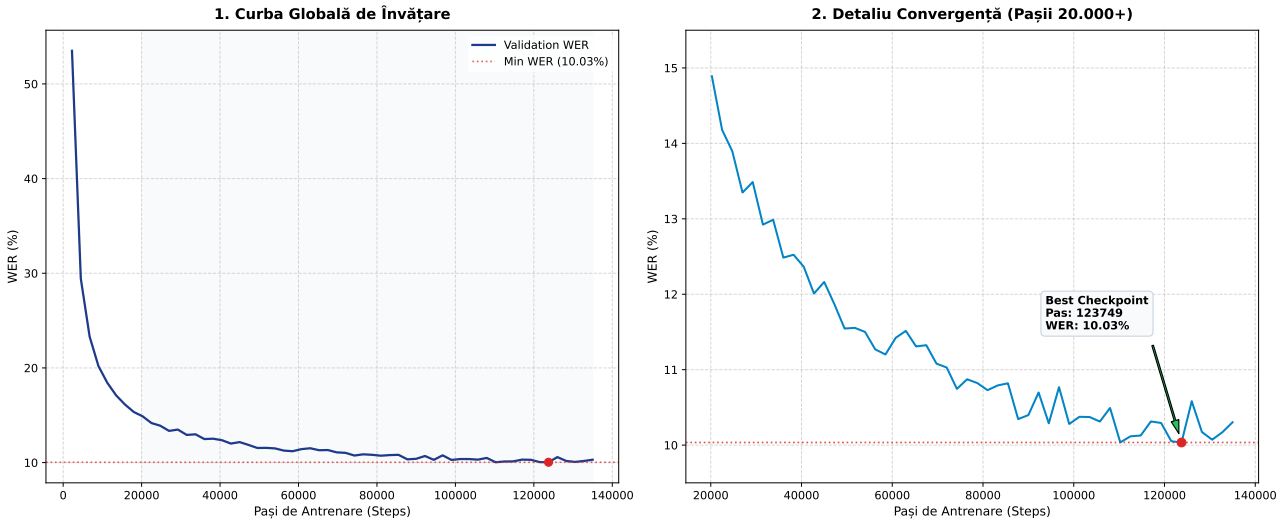

Figura 3.4: Evoluția Word Error Rate (WER) pentru antrenarea cu 2.500 de ore

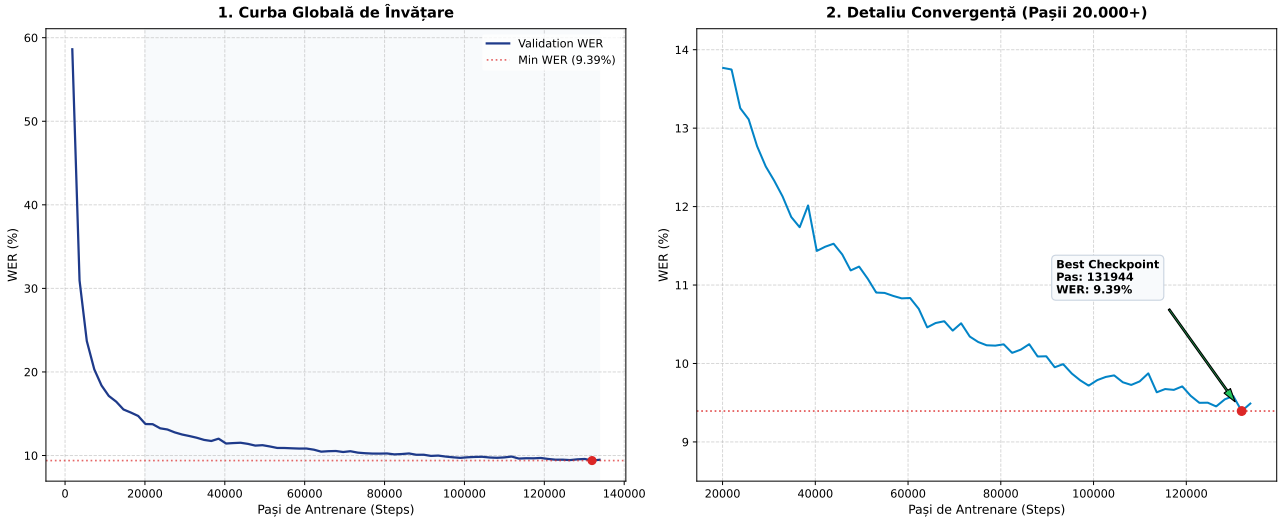

Figura 3.5: Evoluția Word Error Rate (WER) pentru antrenarea cu 10.000 de ore

Pentru a sintetiza această dinamică într-o perspectivă de ansamblu, Figura 3.6 reunește evoluția tuturor celor patru experimente, oferind atât o imagine completă a procesului de antrenare (Panoul A), cât și o vizualizare detaliată a etapei inițiale de convergență (Panoul B).

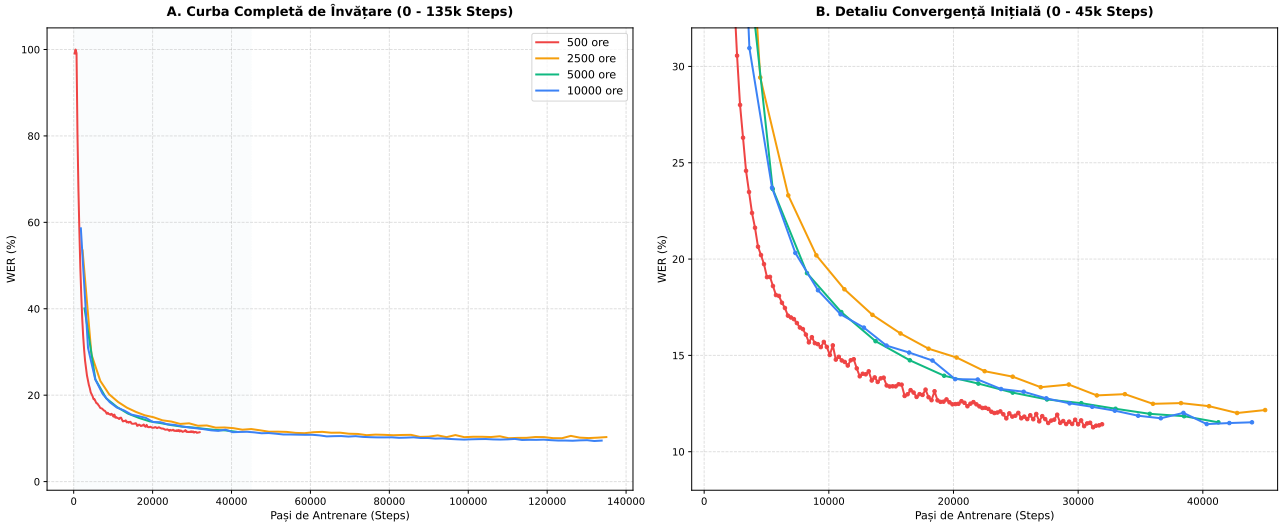

Figura 3.6: Comparatia evolutiilor celor 4 antrenari cu acelasi set de validare.

Analiza comparativă a acestor curbe explică aparentul paradox din primele etape ale antrenării, unde subseturile reduse înregistrează o scădere mult mai rapidă a erorii. Totuși, această convergență rapidă este limitată de diversitatea redusă a datelor, forțând modelul să se plafoneze prematur.

În contrast, curbele modelelor expuse la volume mai mari de date (10.000 de ore) ilustrează o paradigmă de învățare diferită. La același prag de 30.000 de pași, modelul antrenat pe 10.000 de ore abia a parcurs o fracțiune din totalul setului de date, procesând o variație continuă de voci, accente și zgomote de fond. Deși acest nivel ridicat de diversitate generează o traiectorie de convergență inițială mai lentă, el previne supraînvățarea (overfitting) și permite optimizarea continuă a parametrilor pe termen lung.

Așa cum se observă în Panoul A, pe măsură ce antrenarea avansează dincolo de pragul de 100.000 de pași, avantajul diversității devine dominant. Pentru a atinge potențialul maxim de generalizare și a coborî rata de eroare la minimul absolut, modelul necesită în mod imperativ o scalare proporțională a datelor și un număr extins de iterații.

3.3.2 Comparația cu alt corpus de date oratoric

Pentru a stabili un punct de referință obiectiv, performanțele obținute au fost comparate cu un experiment suplimentar. Acesta a utilizat corpusul „CDEP" de 2048 de ore, format din înregistrări ale ședințelor din Palatul Parlamentului. Din punct de vedere al tiparului de discurs (oratoric, instituțional), acest set de date este similar cu cel experimentat anterior.

Monitorizarea procesului de antrenare pe corpusul CDEP a indicat o convergență rapidă, modelul atingând o rată minimă de eroare de 6.65% la pasul 48374. Rezultatul este ilustrat în Figura 3.7.

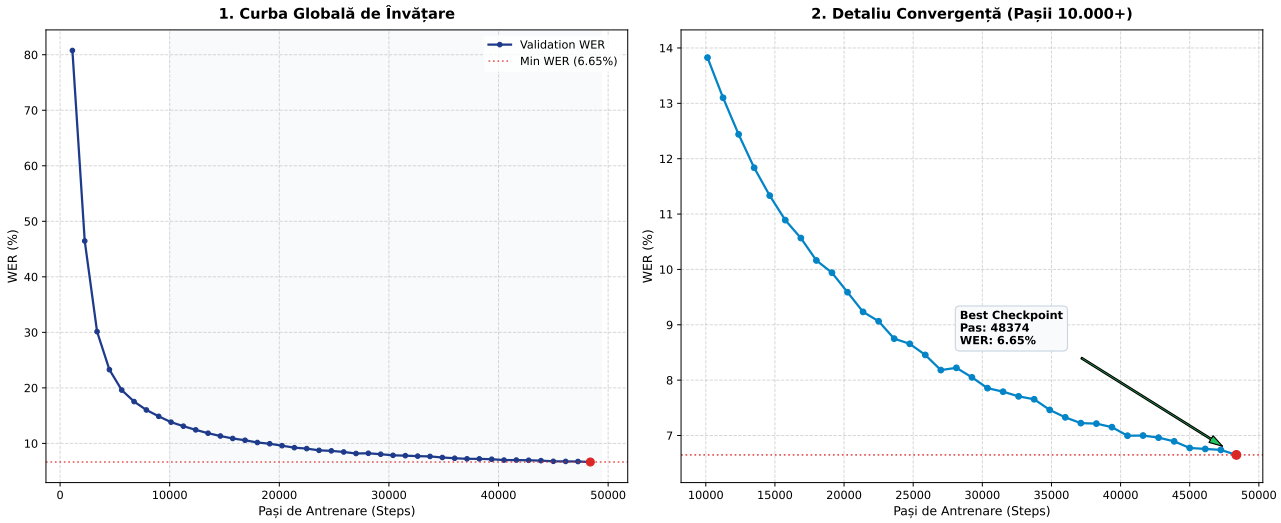

Figura 3.7: Evoluția Word Error Rate (WER) pentru antrenarea folosind corpusul "CDEP"

Comparând acest rezultat cu experimentul similar de 2.500 de ore detaliat anterior, se observă diferențe clare de performanță. Acuratețea modelului antrenat pe CDEP este superioară, obținând o eroare cu 3,38 procente mai mică (6.65% față de 10.03%), optim atins într-un număr mai mic de pași (48.374 comparativ cu 123,749). Mai mult, performanța obținută pe cele 2.500 de ore din CDEP depășește cel mai bun rezultat înregistrat pe setul maxim de 10.000 de ore (WER de 9,39%).

Figura 3.8 arata evolutia celor 3 antrenari pentru primii 50.000 de pasi. Am ales comparatia setului CDEP cu subsetul de 2.500 de ore intrucat au aceeasi dimensiune a corpusului în ceea ce priveste numarul total de ore, precum și subsetul de 10.000 de ore, care a inregistrat cea mai buna performanta pentru arhitectura `FastConformer Large`.

Aceste rezultate demonstrează că, deși domeniul și vocabularul sunt asemănătoare, calitatea datelor joacă un rol determinant. Obținerea unei rate de eroare mai mici cu un volum de date redus și în mai puțini pași indică o calitate superioară a corpusului CDEP.

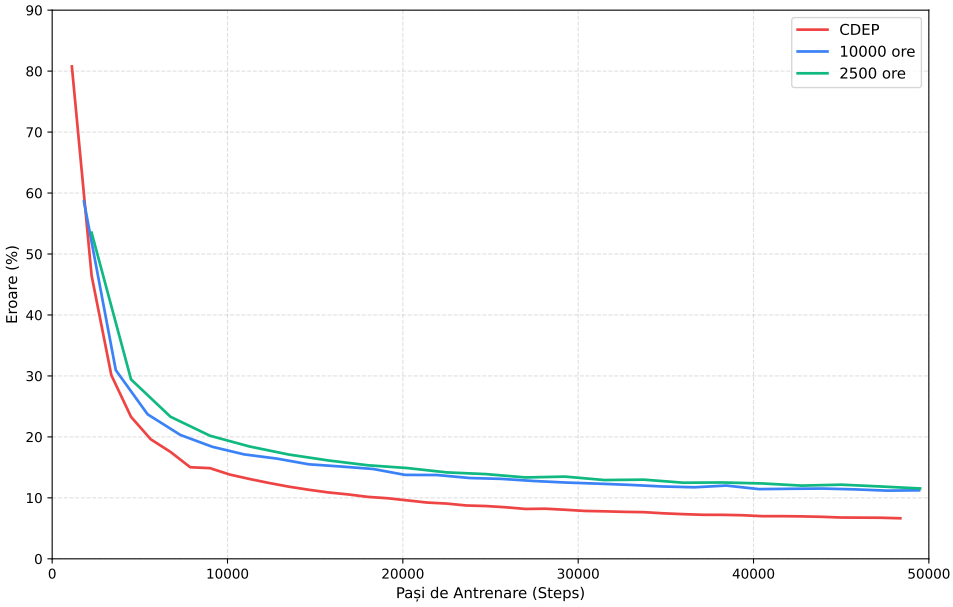

Figura 3.8: Comparatia evolutiilor celor 3 antrenari: cu rosu reprezentam evolutia antrenarii folosind corpusul CDEP, iar pentru subsetul `Granary` de 10.000 de ore albastru, respectiv verde pentru cel de 2.500 de ore.

3.3.3 Evaluarea modelelor experimentale

Pentru a evalua capacitatea de generalizare a modelelor *FastConformer Large* antrenate în etapele anterioare, acestea au fost testate pe o varietate de corpusuri de evaluare independente, clasificate în funcție de tipul de discurs: citit, spontan și oratoric. Tabelul 3.10 prezintă rezultatele obținute (măsurate în WER) pentru fiecare subset de date antrenat, inclusiv comparația cu modelul de referință antrenat pe corpusul CDEP.

Training Dataset	**Training Hours**	**Read**		**Spontaneous**				**Oratory**		
		RSC	**CV-21**	**SSC-eval1 clean**	**SSC-eval1**	**SSC-eval1 other**	**USPDATRO**	**Granary**	**CDEP**	**Fleurs-RO**
Granary	500	13,31	10,29	25,99	29,87	37,10	37,62	11,42	15,76	24,92
Granary	2.500	10,35	8,37	23,11	26,51	32,70	34,59	10,30	13,84	21,00
Granary	5.000	14,03	11,02	24,36	27,75	33,86	38,52	11,52	15,69	26,29
Granary	10.000	9,57	**7,71**	20,44	23,74	29,76	**32,78**	**9,48**	12,79	**20,01**
CDEP-train	2.048	**8,81**	9,43	**14,43**	**19,03**	**27,46**	35,55	15,83	**6,67**	22,15

Tabela 3.10: Rezultate experimentale pentru modelul FastConformer CTC Large 120M cu date filtrate

Analiza comparativă a rezultatelor de evaluare, centralizate în Tabelul 3.10, evidențiază o dinamică tehnică complexă între volumul datelor de antrenare, calitatea acestora și capacitatea de generalizare a arhitecturii acustice. Deoarece toate experimentele prezentate au utilizat același model de bază (FastConformer CTC Large, echipat cu 120 milioane de parametri) folosind aceeasi parametrii de configurare, variațiile de performanță reflectă exclusiv impactul corpusurilor utilizate.

La o primă vedere, experimentele rulate pe subseturile Granary validează regula scalării datelor în Deep Learning: creșterea volumului de la 500 la 10.000 de ore conduce la o minimizare

progresivă a ratei de eroare. Modelul antrenat pe 10.000 de ore atinge cele mai bune performanțe din seria sa pe absolut toate coloanele. Cu toate acestea, introducerea în analiză a experimentului de referință antrenat pe corpusul CDEP (2.048 de ore) schimbă paradigma evaluării.

Pentru o înțelegere obiectivă a raportului calitate-cantitate, cea mai relevantă este comparația directă între modelul CDEP (2048 de ore) și modelul Granary antrenat pe 2.500 de ore. Având volume de date comparabile și aceeași capacitate parametrică (120M), diferențele de performanță demonstrează cum calitatea superioară a etichetelor influențează decodarea, după cum urmează:

- **Vorbirea prin citire:** Comportamentul modelelor este mixt, reflectând natura specifică a seturilor. Pe corpusul RSC, modelul CDEP demonstrează o finețe acustică superioară, obținând un WER de 8,81% față de 10,35% obținut de Granary 2.500. În schimb, pe setul CV-21, care este alcătuit din înregistrări cu variații mari de microfoane și zgomot de fond, modelul Granary preia conducerea (8,37% comparativ cu 9,43%). Acest lucru indică faptul că, deși CDEP este un corpus extrem de curat, Granary oferă o diversitate acustică superioară, utilă în medii necontrolate.

- **Vorbirea spontană:** Aici se observă cel mai frapant avantaj al corpusului CDEP. Pe subseturile SSC, diferențele sunt majore: CDEP obține un excelent 14,43% pe *SSC-eval1 clean*, în timp ce Granary 2.500 se plafonează la 23,11%. Diferențele masive se păstrează pe toate diviziunile SSC, demonstrând că un corpus curățat meticulos (CDEP) ajută modelul să rezolve mai eficient ezitările și dicția imperfectă. Excepția apare pe setul USPDATRO, cel mai dificil și zgomotos set de test, unde Granary 2.500 obține un mic avantaj (34,59% vs. 35,55%), reconfirmând reziliența sa la zgomotul ambiental.

- **Discursul oratoric (Oratory):** Evaluarea pe această categorie scoate în evidență fenomene clare de adaptare la domeniu. Prevăzut, modelul CDEP excelează pe propriul set de test, atingând o eroare remarcabil de mică (6,67%), distanțându-se categoric de Granary 2.500 (13,84%). Totuși, generalizarea modelului CDEP scade pe celelalte seturi oratorice: modelul Granary câștigă atât pe propriul set de test (10,30% față de 15,83%), cât și pe multilingvul *Fleurs-RO* (21,00% vs. 22,15%).

Aceste rezultate subliniază o concluzie esențială pentru antrenarea modelelor ASR: într-un scenariu în care capacitatea parametrică este constantă (120M parametri), un corpus de 2048 de ore curățat și transcris impecabil (CDEP) este capabil să obțină performanțe net superioare pe multe aspecte comparativ cu un set de date similar ca dimensiune (2.500h), dar poluat rezidual.

Mai mult, deși modelul Granary antrenat pe 10.000 de ore reușește, prin forța brută a volumului de date, să recupereze o parte din acest decalaj și să ofere cea mai echilibrată generalizare, el rămâne în urma modelului CDEP de 2048 de ore pe subseturi majore (RSC, întregul spectru SSC și CDEP test). Astfel, evaluarea reconfirmă faptul că scalarea cantitativă a datelor nu poate masca în totalitate imperfecțiunile de etichetare, calitatea rămânând principalul motor al performanței pentru o arhitectură de 120M parametri.

3.4 Analiza performanțelor în funcție de dimensiunea modelului

Pentru a evalua impactul capacității arhitecturale asupra acurateței recunoașterii vocale, am extins experimentele prin antrenarea unui model de dimensiuni superioare, *FastConformer XLarge*, utilizând același subset maxim de 10.000 de ore. Analiza evoluției ratei de eroare la nivel de cuvânt pe setul de validare, pe parcursul a aproximativ 138.600 de pași, indică o convergență foarte stabilă. Modelul a asimilat rapid trăsăturile de bază, coborând sub pragul de 10% eroare încă din jurul pasului 32.000. Ulterior, procesul a intrat într-o etapă lungă de optimizare fină,

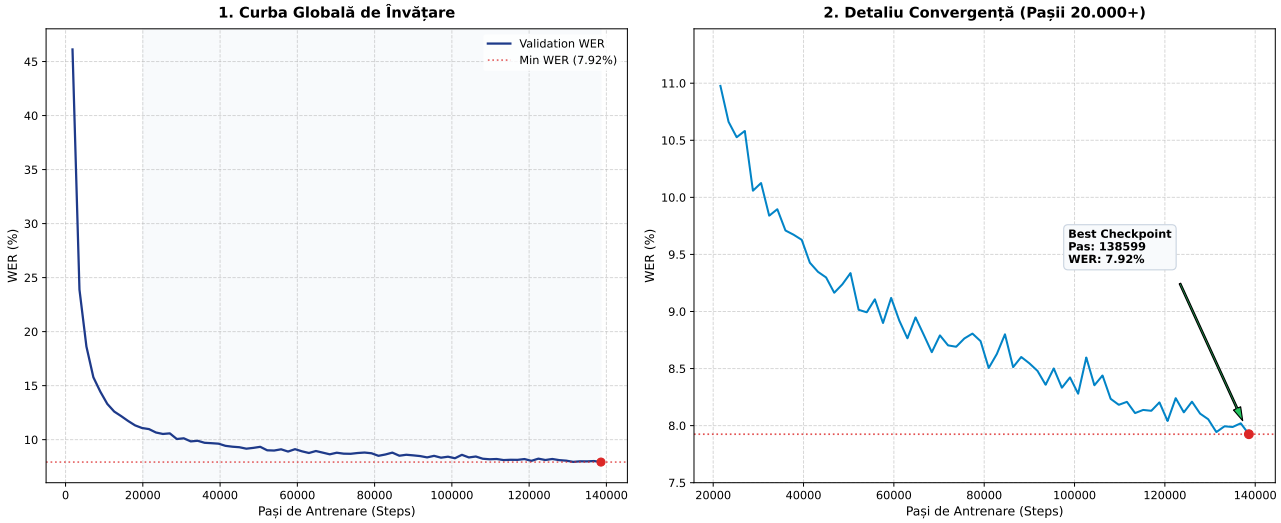

Figura 3.9: Evoluția Word Error Rate (WER) pentru antrenarea modelului `FastConformer XLarge`

atingând o rată minimă de eroare de 7,92% la pasul 138.599. Rezultatele descrise sunt vizibile în Figura 3.9.

Comparând aceste rezultate cu cele obținute de arhitectura *FastConformer Large* pe același volum de date, avantajul extinderii numărului de parametri devine evident. În timp ce varianta *Large* a atins un prag de performanță cu un WER de 9,39% (la pasul 131.944), varianta *XLarge* a reușit să reducă rata de eroare la 7,92%. Această îmbunătățire absolută de 1,47 procente confirmă faptul că un set de date masiv, de 10.000 de ore, este suficient de vast pentru a susține și a valorifica o arhitectură de o asemenea anvergură.

Model	Hours	Read		Spontaneous				Oratory		
		RSC	CV-21	SSC-clean	SSC-eval1	SSC-other	USPDATRO	Granary	CDEP	Fleurs-RO
Large (120M)	10.000	9.57	7.71	20.44	23.74	29.76	32.78	9.48	12.79	20.01
Large (120M)	10.000	9,57	7,71	20,44	23,74	29,76	32,78	9,48	12,79	20,01
XLarge (600M)	10.000	**6,13**	**5,77**	**15,03**	**18,30**	**24,37**	**28,35**	**7,69**	**11,26**	**14,53**

Tabela 3.11: Comparație între arhitecturile Large vs XLarge ale modelului FastConformer CTC (antrenate pe 10.000h)

În Tabelul 3.11 regăsim rezultatele evaluării celor două modele asupra seturilor de testare independente. Analiza comparativă detaliată a acestor date relevă modul în care surplusul de capacitate parametrică al variantei *XLarge* (660 de milioane de parametri) deblochează o acuratețe superioară față de versiunea *Large* (120 milioane de parametri). Această extindere arhitecturală nu a generat o îmbunătățire uniformă, ci a influențat selectiv performanța în funcție de complexitatea acustică a fiecărui domeniu:

- **Vorbirea prin citire:** Pe acest segment, trecerea la arhitectura superioară a determinat o scădere masivă și consistentă a ratei de eroare. Pe setul RSC, valoarea WER s-a redus substanțial, coborând de la un nivel de 9,57% (specific modelului *Large*) până la un excelent 6,13% pentru varianta *XLarge*. O dinamică similară de optimizare se observă și pe corpusul CV-21, unde eroarea a fost comprimată de la 7,71% la 5,77%. Aceste salturi

calitative demonstrează că o rețea cu o capacitate de reprezentare mai mare poate modela cu o precizie mult mai fină corelațiile fonetice formale atunci când semnalul acustic de bază este clar.

- **Vorbirea spontană:** În cazul discursului liber și nestructurat, impactul extinderii modelului este puternic vizibil. Pe subsetul general *SSC-eval1*, modelul *XLarge* reușește o corecție importantă a erorilor, coborând spectaculos de la un WER de 23,74% la 18,30%. Chiar și pe eșantioanele cele mai complexe și degradate acustic (*SSC-other*), eroarea scade de la 29,76% la 24,37%. Pe setul zgomotos *USPDATRO*, eroarea este redusă de la 32,78% la 28,35%. Deși reducerile sunt mari (între 4 și 5 procente absolute), menținerea unor rate de eroare generale mai ridicate în această categorie confirmă că, pe date ezitante, performanța rămâne totuși ancorată în limitările calitative ale corpusului de antrenare.

- **Discursul oratoric:** Această categorie evidențiază adaptabilitatea superioară a noii arhitecturi la medii acustice reverberante. Pe setul *Fleurs-RO*, modelul *XLarge* comprimă rata de eroare la 14,53% (o reducere masivă de la 20,01% înregistrată de versiunea *Large*). Mai mult, pe corpusul parlamentar CDEP, se observă o adaptare robustă, eroarea fiind diminuată de la 12,79% la 11,26%. Arhitectura extinsă demonstrează așadar o capacitate net superioară de a procesa cadențele, intonațiile și reverberațiile caracteristice discursurilor publice.

Rezultatele confirmă validitatea empirică a trecerii la arhitectura *XLarge*. Cele 10.000 de ore de antrenare extrase din corpusul Granary au oferit diversitatea necesară pentru a satura un model de 660 de milioane de parametri, generând scăderi mai mari ale ratei de eroare pe toate axele majore de evaluare comparativ cu versiunea *Large*.

3.5 Evoluțiile antrenărilor în funcție de Rata de Învățare

Rata de învățare (*learning rate*), notată teoretic cu η, reprezintă unul dintre cei mai influenți hiperparametri în procesul de optimizare a rețelelor neuronale profunde. În contextul antrenării unui model acustic de înaltă complexitate, precum arhitectura *FastConformer*, pe un volum masiv de date, calibrarea acestui parametru devine o etapă critică. Alegerea valorii optime nu dictează doar acuratețea finală a modelului, ci și fezabilitatea computațională a întregului experiment.

Prin urmare, experimentele detaliate în cele ce urmează ilustrează modul în care ajustarea acestui hiperparametru influențează dinamica reducerii ratei de eroare, scopul fiind identificarea acelui echilibru optim care garantează o învățare rapidă, stabilă și capabilă să generalizeze eficient complexitatea limbajului vorbit.

Pentru acest experiment, am ales doua cazuri, ambele folosind arhitectura `FastConformer XLarge` datorita performantelor cele mai bune din cadrul acestui proiect, cu doua rate de invatare diferite. Este important de mentionat ca, pentru toate acestea, s-a folosit un planificator de tip `CosineAnnealing`, detaliat în Figura 2.6, cu acelasi numar de pasi de incalzire (4000) și valori diferite pentru numarul total de pasi, ce determina modul în care se va schimba rata de invatare.

Pentru primul experiment, am folosit rata minima de 10^{-6}, urmand ca, dupa cei 4000 de pasi de incalzire, sa ajunga la valoarea de 10^{-4}. Mai departe, rata de învățare urmează o traiectorie descendentă dictată de funcția de planificare *Cosine Annealing*. După cum se observă în Figura 3.10, în prima fază post-încălzire (între pașii 4.000 și aproximativ 50.000), rata de învățare se menține la un nivel relativ ridicat, scăzând foarte lent.

Pe măsură ce antrenarea avansează, panta curbei devine vizibil mai abruptă, rata de învățare coborând treptat până la o valoare de aproximativ 4×10^{-5} la pasul 140.000. Această descreștere

controlată este importanta în etapele finale ale antrenării intrucat scăderea dimensiunii pașilor de actualizare acționează ca un mecanism de optimizare fină.

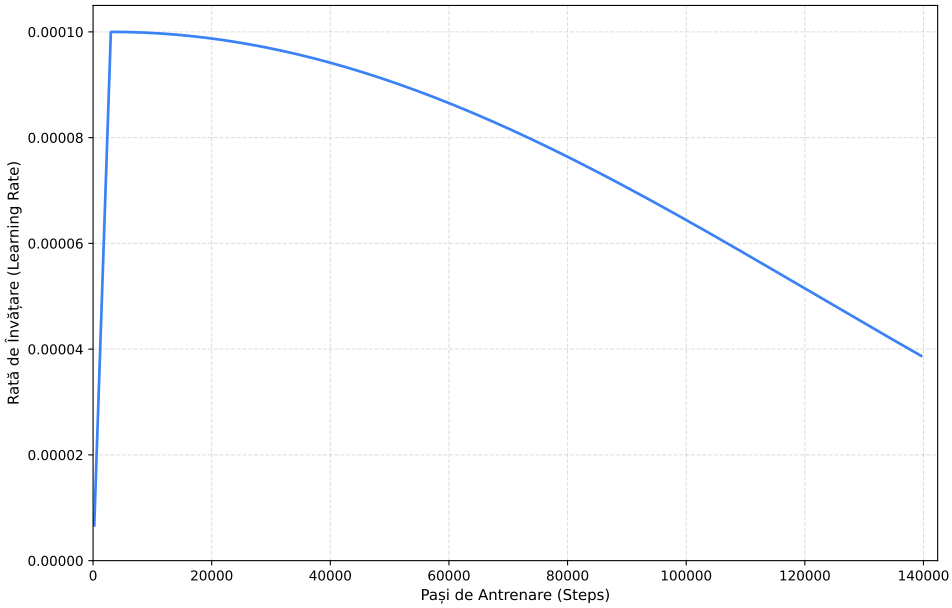

Figura 3.10: Aspectul funcției `CosineAnnealing` pentru primul experiment

Pentru cel de-al doilea experiment, am intensificat faza de explorare a algoritmului, dublând valoarea maximă a ratei de învățare până la 2×10^{-4}, vizibil în Figura 3.11. Similar abordării anterioare, procesul debutează cu o fază de încălzire, în care parametrul η crește accelerat de la o valoare minimă de bază până la acest nou prag maxim.

Odată atins acest vârf, curba intră sub incidența funcției de planificare *Cosine Annealing*, însă prezintă o dinamică mult mai amplă. Datorită pragului superior mai ridicat, modelul beneficiază de o perioadă extinsă în care pașii de actualizare sunt foarte mari (între pașii 4.000 și 40.000). Această agresivitate inițială permite rețelei să traverseze rapid spațiul soluțiilor și să sară peste minimele locale care ar putea bloca optimizarea.

Diferența majoră față de primul experiment se observă în panta de coborâre și în valoarea finală. În acest caz, funcția cosinusoidală dictează o descreștere completă, rata de învățare prăbușindu-se treptat până la valori foarte mici începând cu pragul de 120.000 de pași. Această curba este ideală pentru maximizarea performanței finale în teorie, ce determina modelul sa intre într-o fază de micro-ajustare.

Performanțele detaliate în Tabelul 3.12 relevă o dinamică echilibrată între cele două rate de învățare experimentate. Privind în ansamblu, rezultatele sunt extrem de apropiate, indicând că ambele valori reprezintă o plajă de optimizare validă pentru arhitectura *FastConformer XLarge*.

Analizând defalcat, se observă variații contextuale minore:

- **Vorbirea citită:** Diferențele sunt neglijabile (de exemplu, paritate perfectă de 6,13% pe RSC), ambele strategii convergând optim pe date controlate.
- **Vorbirea spontană și oratorică:** Rata mai conservatoare (10^{-4}) demonstrează o stabilitate marginal superioară pe seturi imprevizibile sau zgomotoase (USPDATRO, Fleurs-RO, Granary). În schimb, rata mai agresivă (2×10^{-4}) ajută modelul să se adapteze ușor mai bine la acustici specifice, precum cele din sălile de plen (CDEP) sau din inregistrarile fara zgomot (SSC-clean).

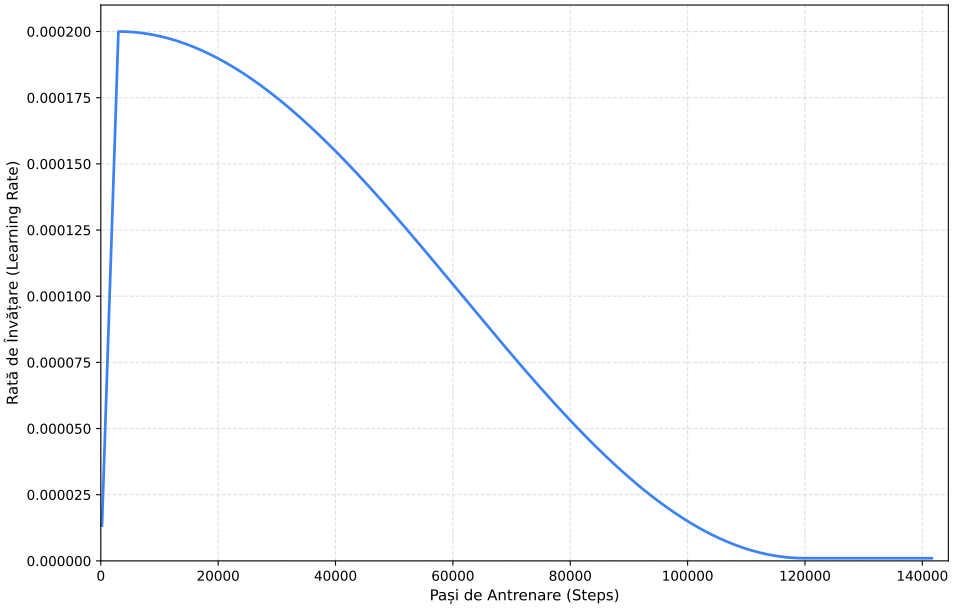

Figura 3.11: Aspectul funcției `CosineAnnealing` pentru cel de-al doilea experiment.

Astfel, fluctuațiile minore generate de dublarea ratei maxime de învățare demonstrează că ajustarea fină a hiperparametrilor oferă doar beneficii limitate și strict punctuale. Faptul că performanța rămâne robustă indiferent de aceste variații validează o ipoteză centrală a procesului de antrenare: calitatea informației primează în fața optimizării algoritmice. Aceste experimente confirmă că imbunatatirea filtrarii și a curățarii setului de date `Granary` este mult mai eficientă pentru maximizarea acurateței și capacității de generalizare a modelului decât calibrarea exhaustivă a parametrilor de antrenare.

Model	**LR**	**Read**		**Spontaneous**				**Oratory**		
		RSC	**CV-21**	**SSC-clean**	**SSC-eval1**	**SSC-other**	**USPDATRO**	**Granary**	**CDEP**	**Fleurs-RO**
XLarge	10^{-4}	**6,13**	5,77	15,03	**18,30**	**24,37**	**28,35**	**7,69**	11,26	**14,53**
XLarge	2×10^{-4}	**6,13**	**5,72**	**14,88**	18,33	24,51	29,84	7,97	**11,08**	15,34

Tabela 3.12: Comparație între diferite valori ale ratei de învățare

3.6 Comparația între cele mai bune modele

Această secțiune prezintă o analiză comparativă între cel mai bun model dezvoltat pe baza corpusului Granary (*FastConformer CTC XLarge*, antrenat pe 10.000 de ore) și trei modele de referință pentru limba română: HMM-DNN (TDNN) [37], RoWhisper-large-v2 [38], Whisper-large-v3 [18] și SpeD ParakeetRo [32].

Model	Read		Spontaneous		Oratory	
	RSC	**CV-21**	**SSC-eval1**	**USPDATRO**	**CDEP**	**Fleurs-RO**
HMM-DNN (TDNN) [37]	**1,90**	-	**9,40**	-	5,40	-
RoWhisper-large-v2 [38]	3,09	9,31	25,05	28,00	62,83	-
Whisper-large-v3 [18]	4,30	6,06	13,99	22,74	26,90	11,08
SpeD ParakeetRo CTC Greedy [32]	2,57	**3,29**	10,10	**27,80**	**4,80**	**8,85**
FastConformer CTC XLarge	6,13	5,77	18,30	28,35	11,26	14,53

Tabela 3.13: Comparație între cel mai bun model dezvoltat pe baza dataset-ului Granary cu decodare CTC Greedy vs baseline-urile: HMM-TDNN, modelele Whisper V2 si V3 și SpeD Parakeet Ro CTC greedy

Pentru a înțelege în profunzime avantajele tehnice și limitările modelului propus, evaluarea a fost defalcată pe cele trei mari categorii de discurs care descriu complexitatea mediului acustic și lexical al limbii române:

- **Vorbirea prin citire:** Această categorie reprezintă benchmark-ul ideal pentru evaluarea acurateței fonetice pure, datele fiind înregistrate în condiții controlate. Pe setul RSC (*Romanian Speech Corpus*), arhitectura hibridă tradițională *HMM-DNN* dezvoltată de Georgescu și colab. [37] menține cel mai scăzut WER din literatură (1,90%), urmată de *SpeD ParakeetRo* [32] cu 2,57%. Modelul nostru, *FastConformer CTC XLarge*, obține 6,13%. Această diferență sugerează că modelele optimizate strict pe corpusuri native mici și atent etichetate rețin mai bine tiparele acustice formale. Totuși, modelul propus se evidențiază pe setul *Common Voice 21* (CV-21), unde atinge un WER de 5,77%, surclasând categoric arhitectura multilingvă *RoWhisper-large-v2* [38] (9,31%) și demonstrând o reziliență notabilă la variațiile canalelor de înregistrare.

- **Vorbirea spontană:** Discursul liber, caracterizat prin ezitări și repetiții, este cel mai exigent test de generalizare. Pe setul USPDATRO, modelul nostru obține un WER de 28,35%, la paritate cu stadiul actual al artei reprezentat de *SpeD ParakeetRo* [32] (27,80%) și devansând modelul *RoWhisper* [38] (28,00%). Această consistență validează diversitatea acustică adusă de corpusul Granary. Pe subsetul SSC-eval1, deși HMM-TDNN [37] (9,40%) și *SpeD ParakeetRo* [32] (10,10%) conservă un avantaj clar, *FastConformer* limitează degradarea performanței la 18,30%, o valoare net superioară celei obținute de *RoWhisper-large-v2* (25,05%).

- **Discursul oratoric:** În această categorie, rezultatele scot în evidență o dinamică particulară a modelelor bazate pe arhitecturi Whisper. Pe setul CDEP, modelul *RoWhisper-large-v2* [38] înregistrează o scădere severă a performanței (62,83% WER). Mai important pentru contextul lucrării de față, modelul de bază *Whisper-large-v3* [18] înregistrează un WER ridicat, de 26,90%, pe acest corpus parlamentar. În contrast, modelul *FastConformer CTC XLarge* obține un WER stabil de 11,26%. Flexibilitatea este confirmată și pe setul *Fleurs-RO*, unde modelul livrează un scor de 14,53%.

O atenție deosebită în cadrul acestei evaluări trebuie acordată modelului *SpeD ParakeetRo CTC Greedy* [32]. Acest sistem reprezintă în prezent cel mai puternic model de tip *state-of-the-art* (SOTA) deschis pentru recunoașterea vorbirii în limba română și este construit pe piloni tehnologici identici cu cei utilizați în proiectul de față: aliniere temporală de tip CTC și o strategie de decodare de tip *Greedy* [32].

Analiza comparativă structurală scoate în evidență un contrast tehnologic: modelul *SpeD ParakeetRo* este o arhitectură compactă, ce integrează doar 110 milioane de parametri [32], în timp ce modelul dezvoltat de noi, varianta *XLarge*, este de șase ori mai mare, având 616 de milioane de parametri. Cu toate acestea, deși a fost antrenat pe un volum substanțial mai mic de date (2.600 de ore, comparativ cu cele 10.000 de ore ale corpusului Granary), modelul *SpeD* reușește să obțină rezultate superioare pe toate subseturile testate, stabilind diferențe clare de acuratețe (cum ar fi 3,29% față de 5,77% pe CV-21 sau 4,80% față de 11,26% pe CDEP). Din acest motiv, *SpeD ParakeetRo* servește drept nivel de referință pentru evaluarea eficienței arhitecturii noastre.

Totodată, trebuie luat în considerare modul în care a fost generat corpusul de antrenare. Spre deosebire de corpusurile de referință care au beneficiat de o verificare manuală, volumul masiv al setului Granary a impus o transcriere complet automatizată. Transcrierea inițială a corpusului Granary a fost realizată utilizând exclusiv modelul *Whisper-large-v3* [18]. Faptul că acest model obține o eroare substanțială de 26,90% pe setul CDEP, un domeniu acustic (discurs parlamentar) asemănător cu sursa principală a înregistrărilor din Granary, sugerează o cauză fundamentală a limitărilor modelului nostru. Rata ridicată de eroare a lui *Whisper* pe aceste date indică faptul că etichetele textuale generate pentru Granary sunt inerent poluate cu erori de transcriere sau halucinații. În ciuda etapelor noastre iterative de filtrare, aceste anomalii persistă în setul de antrenare.

Astfel, concluzia acestor experimente confirmă o axiomă fundamentală în antrenarea modelelor acustice: o creștere exponențială a volumului de date nu poate compensa integral o calitate inferioară a etichetelor de bază, preluând implicit limitările modelului utilizat pentru transcriere. Pentru ca modelele de capacitate mare să depășească pragurile actuale, accentul trebuie mutat spre dezvoltarea unor algoritmi mai avansați de filtrare și aliniere, capabili să purifice datele generate automat.

Concluzii

În cadrul acestui proiect, a fost dezvoltată și evaluată o metodologie complexă pentru antrenarea sistemelor automate de recunoaștere a vorbirii (ASR) în limba română, utilizând arhitectura FastConformer CTC. Experimentele realizate au demonstrat importanța critică a etapelor de preprocesare, evidențiind corelația directă dintre volumul, calitatea datelor și performanța finală a modelelor acustice.

Studiul comparativ a validat faptul că simpla scalare cantitativă a unui corpus nu poate compensa lipsa filtrării acustice. În timp ce modelul preliminar antrenat pe date brute a eșuat la evaluarea externă, aplicarea fluxului proiectat de curățare și normalizare a permis obținerea unor rezultate competitive. Extinderea subseturilor de la 500 la 10.000 de ore a asigurat o scădere progresivă a ratei de eroare (WER) care demonstreaza capacitatea arhitecturii de a absorbi volume masive de informație.

Mai mult, analiza structurală a confirmat utilitatea scalării parametrice a rețelei neuronale. Trecerea de la varianta *Large* (120M parametri) la cea *XLarge* (616M parametri) pe setul de 10.000 de ore a adus o îmbunătățire absolută a WER de 1,47 procente, obținându-se performanțe robuste pe diverse tipare de discurs (citit, spontan și oratoric).

Contribuții personale

Această lucrare reprezintă rezultatul unui efort susținut de cercetare și implementare, materializat prin următoarele contribuții semnificative în domeniul tehnologiilor limbajului vorbit pentru limba română:

- **Explorarea și preprocesarea avansată a corpusului NVIDIA Granary:** Am realizat analiza statistică detaliată și preprocesarea celui mai extins set de date disponibil pentru limba română (aproximativ 17.000 de ore). Prin implementarea unui flux experimental de procesare utilizând NVIDIA NeMo Speech Data Processor și dezvoltarea unor scripturi Python personalizate bazate pe expresii regulate, am normalizat textul, am eliminat transcrierile halucinate și am filtrat segmentele poluate acustic, reducând semnificativ zgomotul din datele de antrenament.

- **Optimizarea mediului de antrenament la nivel hardware:** Am implementat un mecanism eficient de gestionare a volumului masiv de date prin integrarea bibliotecii Lhotse. Fragmentarea a sute de mii de fișiere audio individuale în arhive compacte (format *Lhotse Shar*) și gruparea lor dinamică (*dynamic batching*) au eliminat blocajele de intrare-ieșire (*I/O bottlenecks*), stabilizând gradul de utilizare a unității de procesare grafică (GPU) și reducând considerabil timpul de convergență.

- **Antrenarea și evaluarea modelelor FastConformer CTC:** Am proiectat și executat o suită complexă de antrenări experimentale, evaluând impactul scalării volumului de date (de la 500 la 10.000 de ore) și al dimensiunii parametrice (variantele *Large* de 110M și *XLarge* de 600M). Modelele obținute au fost testate riguros pe o diversitate de corpusuri externe (oratoric, spontan, citit), rezultatele validând capacitatea arhitecturii de a obține performanțe competitive, în mod special pe discursul de tip oratoric.

- **Analiza comparativă a impactului calității vs. cantității datelor:** Am demonstrat empiric că un corpus masiv (10.000h) poate absorbi o cantitate semnificativă de zgomot rezidual, însă nu poate compensa integral imperfecțiunile de etichetare. Comparația tehnică între performanțele obținute pe subseturile Granary și cele antrenate pe un set mult mai redus, dar verificat manual (CDEP de 2.048h), a confirmat că acuratețea și calitatea adnotărilor primează în fața scalării pur cantitative.

Remarci finale

Proiectul aduce o contribuție importantă în domeniul tehnologiilor lingvistice pentru limba română prin testarea intensivă a corpusului Granary. Din perspectivă tehnică, deși modelul dezvoltat nu depășește în totalitate soluțiile dedicate de top (precum SpeD ParakeetRo [32]), rezultatele obținute sunt competitive și demonstrează un potențial solid, în special pe discursul de tip oratoric.

În același timp, rezultatele globale sugerează că, deși Granary reprezintă o resursă cantitativă, prezența eșantioanelor poluate acustic acționează ca un plafon pentru rețelele neuronale de mari dimensiuni. Acest comportament reconfirmă o axiomă actuală în învățarea automată: calitatea adnotărilor și claritatea semnalului sunt la fel de importante precum dimensiunea brută a setului de date.

Dezvoltări ulterioare

Direcțiile viitoare de cercetare se vor concentra pe depășirea limitărilor calitative identificate în actualul corpus. În primă fază, ne propunem optimizarea setului de date Granary prin imbunatatirea fluxului de pregatire al datelor NeMo Curator, precum si cel de preprocesare.

Odată obținut acest corpus regenerat și curățat riguros, prioritățile vor viza dezvoltarea și antrenarea unor noi modele de ASR pentru limba română. Ne propunem să testăm strategii avansate de decodare si să explorăm alte arhitecturi performante, precum *Parakeet* [39].

Bibliografie

[1] Perceptron: A basic neural network model for deep learning. https://towardsai.net/p/l/perceptron-a-basic-neural-network-model-for-deep-learning.

[2] Simon J.D. Prince. *Understanding Deep Learning*. The MIT Press, 2023.

[3] NVIDIA Corporation. Nvidia nemo framework user guide, 2026.

[4] Lawrence R. Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition*. Prentice Hall, Englewood Cliffs, NJ, 1993.

[5] Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. Online Draft, 2026.

[6] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[7] Walter Pitts Warren S. McCulloch. A logical calculus of the ideas immanent in nervous activity. https://doi.org/10.1007/BF02478259.

[8] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. http://dx.doi.org/10.1037/h0042519.

[9] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[10] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30, page 3, 2013.

[11] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[13] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

[14] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[15] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, and Brian Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[16] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding (ASRU)*, number EPFL-CONF-192584. IEEE Signal Processing Society, 2011.

[17] Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[18] Daniel Galvez Zhang, Wei Kang, Fangjun Yang, Zengwei Yao, Pengfei Guo, Jinyu Kuang, Long Xie, and Daniel Povey. Zipformer: A faster and better encoder for automatic speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[20] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on audio, speech, and language processing*, 22(10):1533–1545, 2014.

[21] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. Conformer: Convolution-augmented transformer for speech recognition. In *Interspeech*, 2020.

[22] Yiping Lu, Zhuohan Li, Di He, Zhiqing Sun, Bin Dong, Tao Qin, Liwei Wang, and Tie-Yan Liu. Understanding and improving transformer from a multi-particle dynamic system point of view. *arXiv preprint arXiv:1906.02762*, 2019.

[23] Dima Rekesh, Samuel Kriman, Vahid Reda Noroozi, He Huang, Oleksii Hrinchuk, Ankur Kumar, Saurabh Bhati, and Boris Ginsburg. Fast conformer with linearly scalable attention for efficient speech recognition. *arXiv preprint arXiv:2305.05084*, 2023.

[24] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1251–1258, 2017.

[25] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*. ACM, 2006.

[26] Alex Graves. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*, 2012.

[27] Shinji Watanabe, Takaaki Hori, Suyoun Kim, John R Hershey, and Tomoki Hayashi. Hybrid ctc/attention architecture for end-to-end speech recognition. In *2017 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 468–475. IEEE, 2017.

[28] Hainan Xu, Fei Jia, Somshubra Majumdar, He Huang, Shinji Watanabe, and Boris Ginsburg. Efficient sequence transduction by jointly predicting tokens and durations. In *International Conference on Machine Learning (ICML)*, 2023.

[29] Lalit R Bahl, Frederick Jelinek, and Robert L Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2), 1983.

[30] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations (EMNLP)*, pages 66–71, Brussels, Belgium, 2018. Association for Computational Linguistics.

[31] NVIDIA Research. Granary: A large-scale multilingual speech dataset. *arXiv preprint arXiv:2505.13404*, 2025.

[32] Gabriel Pîrlogeanu, Alexandru-Lucian Georgescu, and Horia Cucu. Open source state-of-the-art solution for romanian speech recognition. In *2025 International Conference on Speech Technology and Human-Computer Dialogue (SpeD)*, pages 102–107. IEEE, 2025.

[33] Piotr Żelasko, Jan Trmal, Daniel Povey, and Sanjeev Khudanpur. Lhotse: a speech data representation library for the modern deep learning ecosystem. *arXiv preprint arXiv:2110.12561*, 2021.

[34] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

[35] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, and Zhifeng Chen. GPipe: Easy scaling with micro-batch pipeline parallelism. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

[36] Dhiraj Kalamkar, Dheevatsa Mudigere, Naveen Mellempudi, Dipankar Das, Kunal Banerjee, Sasikanth Avancha, Dharma Teja Vooturi, Nataraj Jammalamadaka, Jianyu Huang, Hector Youssef, et al. A study of bfloat16 for deep learning training. *arXiv preprint arXiv:1905.12322*, 2019.

[37] A.L. Georgescu, H. Cucu, and C. Burileanu. Improvements of speed's romanian asr system during reterom project. In *2021 International Conference on Speech Technology and Human-Computer Dialogue (SpeD)*, 2021.

[38] V. Pais, V. B. Mititelu, R. Ion, and E. Irimia. Evaluating a fine-tuned whisper model on underrepresented romanian speech. In *2023 International Conference on Speech Technology and Human-Computer Dialogue (SpeD)*, pages 141–145, 2023.

[39] Monica Sekoyan, Nithin Rao Koluguri, Nune Tadevosyan, Piotr Zelasko, Travis Bartley, Nikolay Karpov, Jagadeesh Balam, and Boris Ginsburg. Canary-1b-v2 & parakeet-tdt-0.6b-v3: Efficient and high-performance models for multilingual asr and ast. *arXiv preprint arXiv:2509.14128*, 2025.

Anexa A
Codul Sursă și fișierele de configurație

Github repository: `https://github.com/igaby27/Licenta_Granary_2026`