



UNIVERSITATEA DIN  
BUCUREŞTI



FACULTATEA DE  
MATEMATICA ŞI  
INFORMATICA

SPECIALIZAREA INFORMATICĂ

Lucrare de licență

# METODE DE CLASIFICARE PENTRU SEMNALE EEG

Absolvent  
Florete Fabian-Andrei

Coordonator științific  
Conf. Dr. Sergiu Nisioi

Bucureşti, Iunie 2025

## **Rezumat**

Creierul uman constituie una dintre cele mai complexe și mai promițătoare arii de cercetare. Își, contrar faptului că a fost studiat intens de-a lungul timpului, încă nu avem un model destul de clar al modului în care acesta funcționează.

În această lucrare de licență am urmărit să explorez, să evaluez și să compar cât mai mulți algoritmi legați de interpretarea semnalelor emise de creier. Scopul principal este de a vedea gradul de interpretabilitate generat de diverse stimuli de scurtă durată. Lucrarea mea se diferențiază de restul lucrărilor din domeniul ERP prin tipul datelor folosite și prin specificitatea subiecților. Astfel, datele mele provin de la oameni specializați, și anume arhitecți ce au privit imagini specifice domeniului, mai exact coloane generate artificial.

Pe parcursul lucrării am implementat și comparat sistematic algoritmi documentați în literatura de specialitate.

## **Abstract**

The human brain represents one of the most complex and promising areas of research. And, although it has been intensely studied throughout the years, we still do not have an accurate model of the way it functions.

In this thesis, I aimed to explore, evaluate and compare a wide range of algorithms related to the interpretation of brain-emitted signals. The main objective is to asses the degree of interpretability elicited by various short-duration stimuli. My work differs from the existing literature in the ERP domain via the type of data used and the specificity of the subjects. The data used comes from specialized subjects, more particularly architects that have been shown images specific to their domain, specifically, artificially generated columns.

Throughout the study, I systematically implemented and compared algorithms documented in the scientific literature.

# Cuprins

<b>1 Introducere</b>	<b>5</b>
1.1 Context și motivăție . . . . .	5
1.2 Obiectivele Lucrării . . . . .	5
1.3 Noțiuni preliminare despre paradigma ERP . . . . .	6
<b>2 Metodologia</b>	<b>7</b>
2.1 Setul de date . . . . .	7
2.1.1 Preluarea datelor și formatul lor . . . . .	7
2.2 Preprocesarea datelor . . . . .	8
2.2.1 Filtrare, interpolare, ICA . . . . .	8
2.2.2 Separare în epoci, AutoReject . . . . .	11
2.2.3 Balansarea claselor și modul de testare . . . . .	12
2.3 Extragerea caracteristicilor . . . . .	12
2.3.1 Datele drept semnale EEG . . . . .	13
2.3.2 Datele drept caracteristici statistice și descompuneri în domeniul frecvențelor . . . . .	15
2.3.3 Datele drept caracteristici geometrice Riemann . . . . .	15
2.4 Modele de clasificare . . . . .	16
2.4.1 Modele clasice din scikit-learn . . . . .	16
2.4.2 Braindecode . . . . .	17
2.4.3 Model personal . . . . .	19
<b>3 Arhitectura aplicației</b>	<b>21</b>
3.1 Împărțirea codului . . . . .	21
3.2 Alte librării folosite . . . . .	21
3.2.1 Optimizarea încărcării datelor folosind Ray . . . . .	21
3.2.2 Căutarea hiperparametrilor folosind Optuna . . . . .	22
<b>4 Concluzii</b>	<b>23</b>
4.1 Interpretarea rezultatelor . . . . .	23
4.2 Îmbunătățiri și perspective de viitor . . . . .	23

<b>Bibliografie</b>	<b>24</b>
<b>A Tabele cu rezultate</b>	<b>29</b>
<b>B Cod sursă</b>	<b>32</b>

# Capitolul 1

## Introducere

### 1.1 Context și motivație

EEG-ul, sau Electroencefalograma, reprezintă o modalitate sigură și non-invazivă de a măsura activitatea electromagnetică a creierului. Această măsurare este realizată printr-o cască ce conține mai mulți electrozi. Fiecare electrod în parte măsoara activitatea creierului în Volți. Aceste metode au aplicații într-o multitudine de domenii. Cele mai folosite cazuri în literatura de specialitate sunt: detectarea fazelor somnului, a simptomelor epilepsiei, a sindromului Alzheimer, a diferențelor emoției puternice, precum tristețe, frică, fericire și, în domeniul de Brain-Computer interface, care se concentrează pe decodarea semnalelor creierului asociate cu mișcarea membrelor. De-a lungul anilor, cercetătorii au obținut rezultate din ce în ce mai bune în domeniul clasificării emoțiilor induse de un stimул vizual. Dacă în anul 2016 cele mai bune modele se bazau pe SVM-uri și ajungeau la o acuratețe de 73% [6], pentru clasificări între două clase și 62% pentru clasificări între trei clase, în anul 2020, modele precum TSception, ce se bazează pe rețele neuronale ating acurateți de peste 86%[10].

Totuși, în majoritatea cazurilor, stimulii induși sunt cauzați fie de evenimente de lungă durată: video-uri, ore întregi de somn, fie de evenimente binare, având clase target/non-target în cazul, de exemplu, al visual speller-elor [36]. Mi-am propus să stabilesc dacă stimulii de scurtă durată au același potențial de a evoca răspunsuri capabile de a fi interpretate algoritmice precum cei de lungă durată.

### 1.2 Obiectivele Lucrării

Scopul lucrării este de a prezice reacția subconștientă pe care o are mintea unui arhitect, provocată de afișarea pe durată scurtă a unor imagini reprezentând coloane generate artificial. În acest context, datele reprezintă semnale EEG de tip ERP. În plus, prezentarea este făcută de-a lungul a trei dimensiuni afective: Valentă, excitare și dominantă.

Abordarea are la bază presupunerea că ERP-urile reflectă reacții spontane, și, faptul că, dacă un participant se gândește mai mult timp la o imagine începe să nu mai fie sigur dacă îi place sau nu. În îndeplinirea acestui scop, am folosit cât mai mulți algoritmi, comparându-i între ei pentru fiecare paradigmă de interpretare a semnalelor.

### 1.3 Notiuni preliminare despre paradigma ERP

ERP (Event-Related Potential), sau, în traducere, potențial legat de eveniment, reprezintă răspunsul creierului uman asociat unui eveniment stimulant [37]. Evenimentele pot fi cauzate de diversi astfel de stimuli, de exemplu auditivi sau vizuali. Aceștia sunt captați prin intermediul unei căști EEG ce înregistrează activitatea creierului. ERP-urile pot fi mai departe împărțite în mai multe categorii în funcție de semnificația pe care dorim să o extragem din semnal. ERP-urile reprezintă în sine o fereastră de timp din semnalul original unde se află răspunsul subconștient al subiectului în fața stimулului prezentat.

# Capitolul 2

## Metodologia

### 2.1 Setul de date

#### 2.1.1 Preluarea datelor și formatul lor

Datele constau în 26 de fișiere excel ce reprezintă citirile electrozilor subiecților experimentului. Experimentul a constat într-un grup de arhitecți care au privit imagini generate cu inteligență artificială reprezentând coloane. Imaginile au fost prezentate timp de 0.4 secunde pe ecran, în încercarea de a captura o reacție subconștientă a participanților. Ulterior, imaginile privite sunt notate pe 3 criterii: V(Valence/Valență), A(Arousal/Excitare) și D(Dominance/Dominanță). Notele stimulilor sunt discretizate în 3 categorii: stimul negativ (note de la 1 la 3), stimul neutru (note de la 4 la 6) și stimul pozitiv (note de la 7-10). Scopul este astfel prezicerea categoriei de stimul provocat de o imagine. În pofida faptului că aceleiași imagini au fost prezentate în mod repetat, în scopul evaluării pe cele trei dimensiuni, am considerat fiecare apariție ca fiind un eveniment independent, presupunând că răspunsul neuronal asociat fiecărei expunerii cât și punctajul acordat de participant poate varia. Motivul principal pentru care am ales această abordare a fost faptul că notele acordate nu au rămas constante de-a lungul prezentării aceleiași imagini.

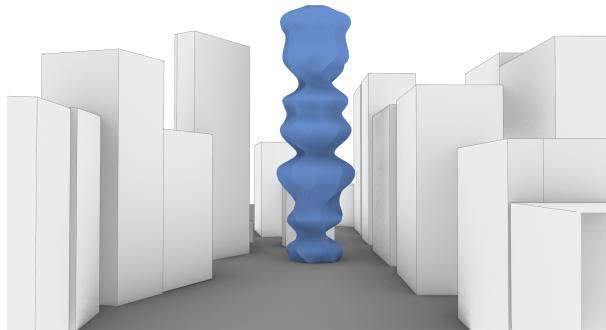


Figura 2.1: Imagine generată artificial.

Fiecare fișier conține înregistrările corespunzătoare a 21 de electrozi plasați de-a lungul capului. Ei au măsurat activitatea creierului cu o frecvență de 300 Hz. Există și un canal

de referință Pz care a fost folosit pentru a calibra măsurătorile. Astfel, el nu apare în setul de date. Corespunzător fiecărui sesiuni, am avut acces și la un fișier ce conține notele acordate de subiect fiecărei imagini văzute.

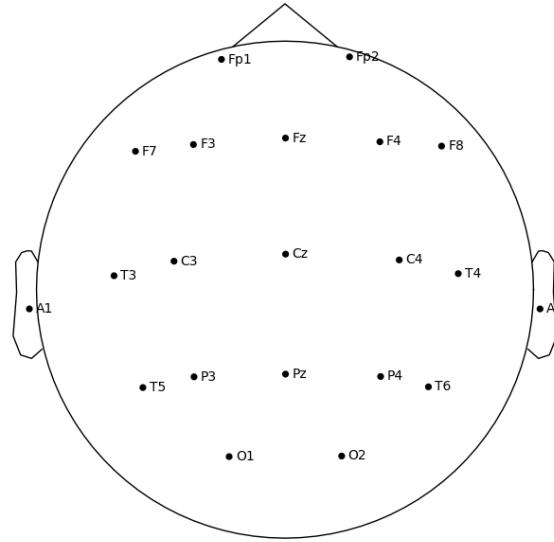


Figura 2.2: Pozițiile electrozilor la nivelul capului.

Datele din formatul brut sunt încărcate folosind librăria Pandas [31] din Python. Pe urmă, am mai adăugat coloane specifice etichetelor experimentului. Mai departe, am transformat datele în format .FIF și le-am încărcat în format de EEG-uri Raw din librăria MNE [13]. Aceasta este specializată în încărcarea și preprocesarea datelor EEG.

## 2.2 Preprocesarea datelor

Înainte de a introduce datele în modele, acestea trebuie curățate. În primul rând, nu toate frecvențele sunt relevante pentru problemele de clasificare. Multe pot proveni din zgomot de fundal, mișcarea fizică a electrozilor sau interferențe electrice, precum zgomotul generat de rețea electrică sau imperfecțiunea electrozilor la capturarea semnalelor. Toată partea de preprocesare este parametrizată pentru a permite căutarea valorilor optime să cum se poate vedea în figura B.6 din Appendix.

### 2.2.1 Filtrare, interpolare, ICA

Literatura de specialitate sugerează să filtrăm datele astfel încât să păstrăm doar frecvențele între 4-40 Hz[12]. Am folosit astfel un filtru trece-bandă care a păstrat doar frecvențele din acest interval. Cu ajutorul acestuia am eliminat frecvențele înalte cauzate de zgomotul electric al aparatului de măsurare, precum și drift-ul semnalului cauzat de modul în care aparatura a fost montată, neavând contact perfect cu scalpul.

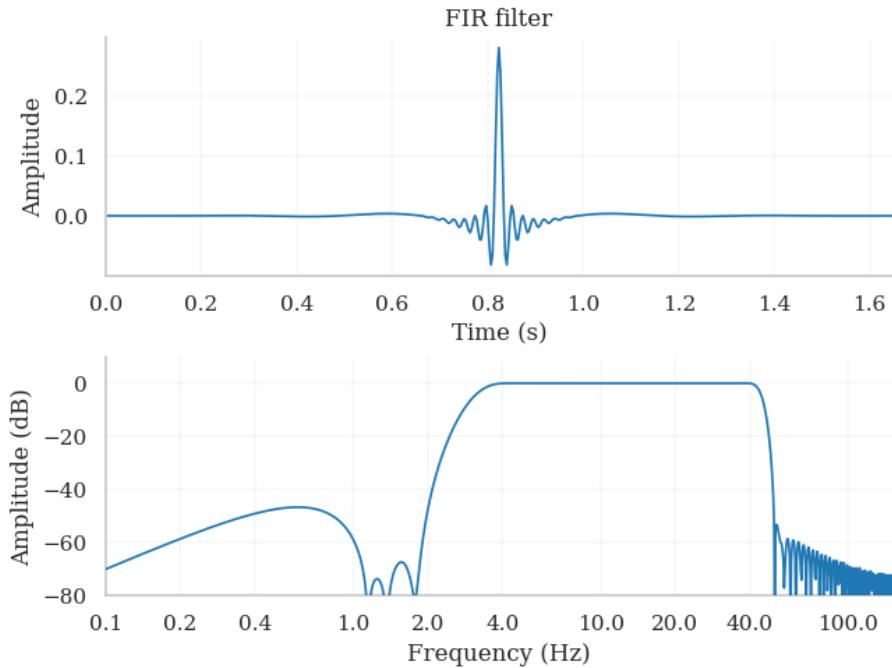


Figura 2.3: Filtrul bandpass folosit cu frecvențe între 4 și 40 Hz.

Urmatoarea preprocesare aplicată este interpolarea canalelor corupte. Semnalul canalelor marcate ca fiind corupte este înlocuit prin interpolarea canalelor vecine acestuia. Mai departe, am aplicat tehnica de ICA(Independent Component Analysis) pentru a elimina clipirile din canalele Fp1 și Fp2 ce aveau reverberări și asupra celorlalte canale. Tehnica se bazează pe descompunerea semnalelor într-un număr de componente specificate. În contextul modului prin care am eliminat componente legate de ochi, am implementat două abordări. Prima este de a selecta manual componentele ce vrem să le eliminăm. Am făcut distincția dintre componentele ce trebuie păstrate și cele ce trebuie eliminate uitându-mă la rezultatul descompunerii și eliminând componentele ce au activări mai mari în zona ochilor. În exemplul din figura 2.4 acestea sunt: ICA002, ICA014 și ICA015. A doua abordare o reprezintă setarea unui prag de 2.5 deviații standard după care componentele să fie eliminate. În principal, componentele cele mai active sunt chiar cele din zona ochilor, acestea ajungând astfel să fie eliminate.

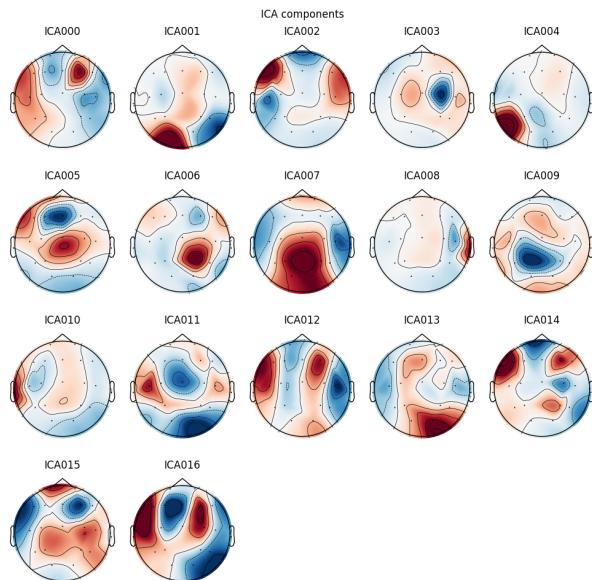


Figura 2.4: Descompunerea ICA a semnalului.

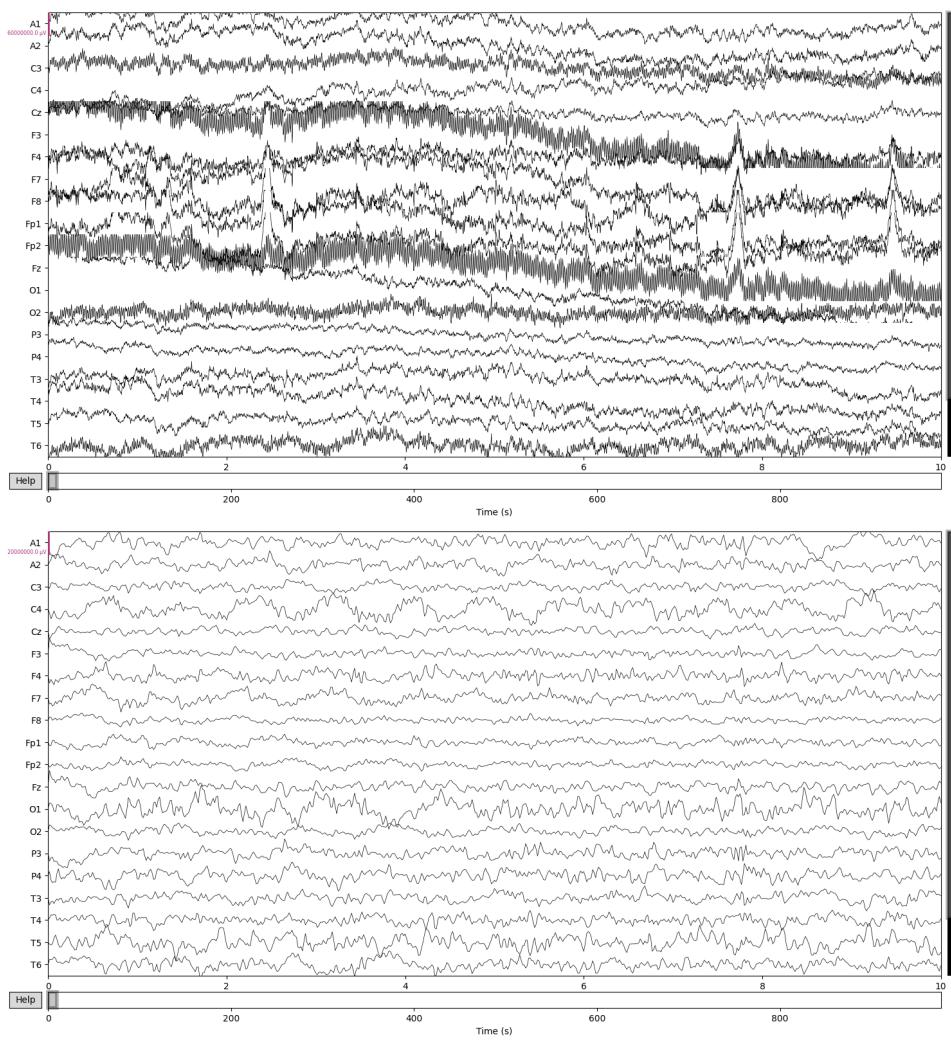


Figura 2.5: Semnalul înainte și după filtrare.

Calitatea datelor a fost semnificativ îmbunătățită. Am eliminat zgomotul cauzat de clipiri, drift-urile și frecvențele înalte.

## 2.2.2 Separare în epoci, AutoReject

Mai departe, pentru a avea date pentru clasificare, am împărțit semnalul continuu în epoci. O epocă reprezintă o fereastră mai mică din semnalul original. Aceasta este extrasă din semnalul continuu cu ajutorul unui canal auxiliar ce conține indexul stimулului prezentat sau 0 dacă în acel moment nu este prezentat niciun stimул. Pentru a elimina eventualele probleme cauzate de mărimi diferite ale datelor, pentru fiecare epocă am luat și 0.2 secunde de semnal de dinaintea evenimentului de declansare (afisarea imaginii). Această porțiune a fost folosită pe post de valoare de referință, valoarea acesteia fiind scăzută din restul semnalului. În acest fel, începutul și sfârșitul fiecărei epoci au fost centrate în jurul valorii zero. Figura 2.6 ne confirmă atât faptul că începutul și sfârșitul epocilor sunt centrate corespunzător, cât și faptul că ICA a reușit să eliminate activările din zona ochilor Fp1 și Fp2. Principalele activări se situează în zona O1, O2.

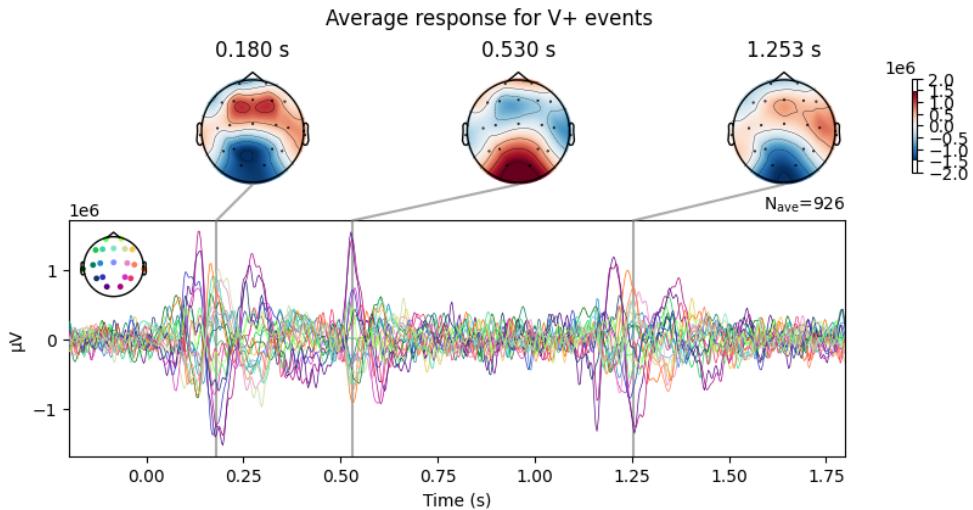


Figura 2.6: Raspunsul mediu al tuturor subiecților pentru o imagine cu valență pozitivă.

Totuși, nu toate epocile sunt bune. Unele dintre ele încă au imperfecțiuni și pot contamina datele de antrenare, așa că ele trebuie aruncate. Pentru a automatiza procesul de găsire și eliminare a epocilor corupte am folosit AutoReject[15]. Această librerie găsește atât epoci corupte cât și canale corupte în epocile valide. AutoReject funcționează prin calcularea amplitudinii între vârfuri și văi (peak to peak amplitude) din cadrul unui semnal și selectarea anumitor praguri candidate. Algoritmul face acest lucru prin împărțirea epocilor în epoci de antrenare și testare, iar apoi folosește epocile de antrenare pentru a calcula pragul de eliminare, iar pe cele de testare pentru a calcula eroarea cauzată de acest prag. Eroarea este calculată între valoarea medie a setului de antrenare și mediana setului de testare folosind rădăcina medie pătrată a erorilor. Pragurile sunt alese dintr-un sir de praguri candidat.

Epocile selectate sunt eliminate complet, iar canalele considerate corupte sunt înlocuite prin interpolare. Impactul aplicării acestui proces a fost scăderea acurateșii cu 10%. Motivul pentru care performanța a scăzut este deoarece și datele de antrenare au scă-

zut semnificativ. Din figura 2.7 reiese faptul că AutoReject găsește multe probleme în semnalul original. Există posibilitatea ca AutoReject să depisteze probleme exact acolo unde se află răspunsul creierului la stimul și prin urmare să scoată informația folositoare. Această teorie este susținută și de faptul că răspunsurile creierului nu sunt în conformitate cu restul semnalului. Figura 2.6 ne indică faptul că amplitudinea semnalului în urma afișării stimулului este în afara mediei celorlalte canale. Prin urmare, AutoReject poate interpreta aceste vârfuri ca fiind date corupte.

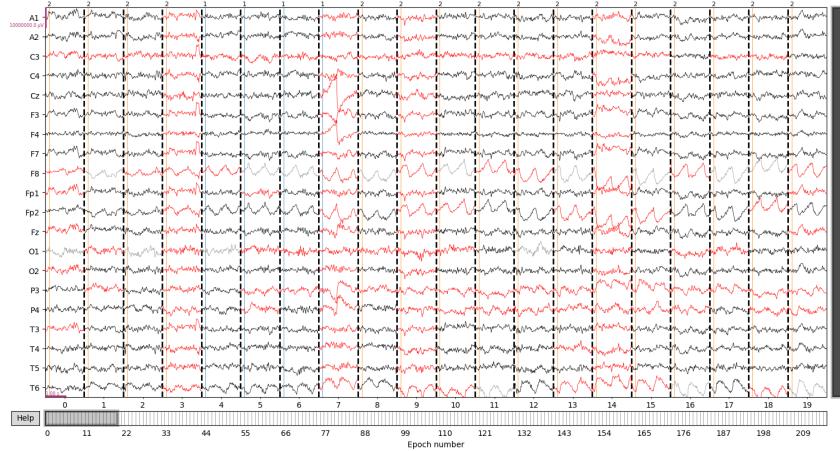


Figura 2.7: Epocile și canalele marcate de AutoReject ca fiind corupte.

### 2.2.3 Balansarea claselor și modul de testare

În total am dispus de 5598 epoci provenite din cei 26 de subiecți. Din rândul tuturor epocilor, 942 au valență negativă, 3642 neutră și 1014 pozitivă. Numărul poate varia, deoarece AutoReject conține și componente non-deterministe. Pentru a testa performanța modelelor, am utilizat validarea încrucișată asupra celor 26 participanți în felul următor: 21 de participanți au fost extrași pentru antrenare, 2 pentru validare și încă 3 pentru testare. În total am trecut prin câte 8 subseturi pentru fiecare model în parte.

Constatăm faptul că epocile nu sunt distribuite după etichetă în mod egal. Dacă modelul ar prezice numai valență neutră, ar avea acuratețe mai mare decât dacă ar prezice aleator. Pentru a aborda problema imbalansului claselor am folosit două abordări. Prima este de a calcula ponderile claselor, transmitându-le mai departe funcției de pierdere a clasificatorului. A doua este de a utiliza metoda numită Synthetic Minority Oversampling Technique (SMOTE)[18] pentru a genera date sintetice.

## 2.3 Extragerea caracteristicilor

În domeniul clasificării semnalelor EEG există 2 paradigme pentru extragerea caracteristicilor. Prima este de a trata semnalul EEG ca fiind o serie de timp. Datele de esantionare ale semnalului sunt trimise mai departe spre model ca input. În cele mai

multe cazuri, modelele ce însotesc această abordare sunt bazate pe rețele neuronale adânci, reprezentate de LSTM-uri, rețele convoluționale și transformere. A doua este de a extrage caracteristici statistice și geometrice din semnal. Acestea sunt sub forma de medii, proprietăți spectrale și caracteristici geometrice Riemann, care mai apoi sunt transmise unui model. Corespunzător acestei abordări sunt modelele mai simpliste, precum SVM-uri sau arbori de decizie, capabile de a separa liniar, folosind un număr mic de parametrii caracteristicile extrase. Contraștării observații, am încercat prima abordare inclusiv cu modele liniare, plecând de la ipoteza că reacția participanților se asemănă între experimente de-a lungul celor trei categorii și produc răspunsuri neuronale asemănătoare ce ar avea poziții apropriate în spațiul de separare.

### 2.3.1 Datele drept semnale EEG

După cum am spus, semnalul EEG are 20 de canale. Totuși, dintre acestea, nu toate sunt relevante. Pentru a găsi cele mai relevante canale, în primul rând am transpus semnalul în domeniul timp-frecvență utilizând implementarea din MNE[13] a undelor Morlet. Am aplicat această operație pentru toate epocile și, astfel, am obținut coerentă inter-experiment(ITC, sau inter-trial coherence). ITC-ul îmi arată asemănarea în descompunerea din domeniul timp-frecvență a epocilor.

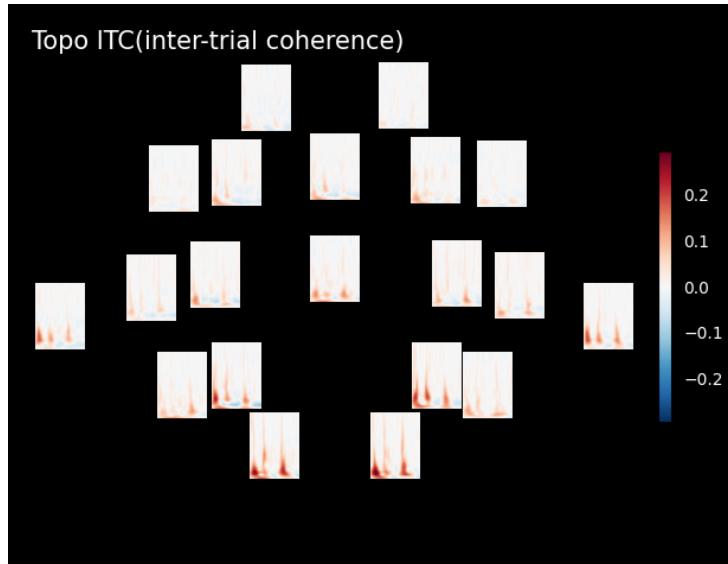


Figura 2.8: ITC-ul de-a lungul tuturor epocilor înregistrate pentru toți participanții.

Putem remarca faptul că diagrama 2.8 are o disperzie asemănătoare cu cea a plotării electrozilor pe creier, figura 2.2. Din această diagramă observăm importanța fiecărui canal în timpul unei epoci. Astfel, de exemplu, canalele Fp1 și Fp2 nu oferă la fel de multă informație comparativ cu O1 sau O2. Canalele alese de mine ca fiind relevante au fost: O1, O2, P3, P4, A1, A2, T5, T6. Alegând un subset de canale din cele initiale, am mărit viteza cu care am procesat datele cu 36%, precum și acuratețea modelului cu 3-4%.

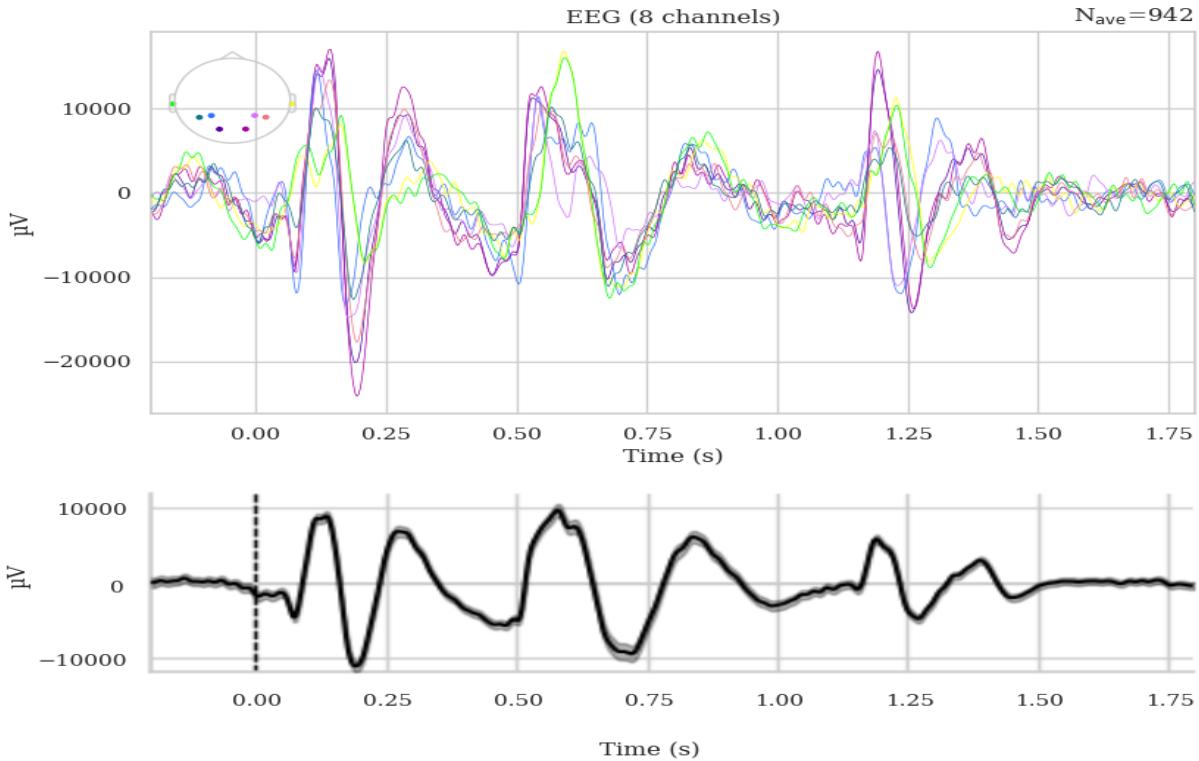


Figura 2.9: ERP-ul canalelor selectate O1, O2, P3, P4, A1, A2, T5, T6.

Un alt rezultat al alegerii unui subset de canale a fost faptul că, am reușit să apropie evenimentul de unul de tip P300[30], fiind bine cunoscut și studiat în neuroștiință. Teoria spune că, odată prezentat un stimul unui participant, răspunsul cognitiv apare la aproximativ 300 de milisecunde după evenimentul declanșator. În cazul meu, observăm că aproximativ după 0.3 s de la afișarea stimulului (momentul 0) apare un vârf, fiind precedat și antecedat de 'vai'. Prima vale are denumirea de P200, iar a doua de P300. Această disperare a semnalului confirmă faptul că preprocesările făcute nu au alterat semnificația semnalului.

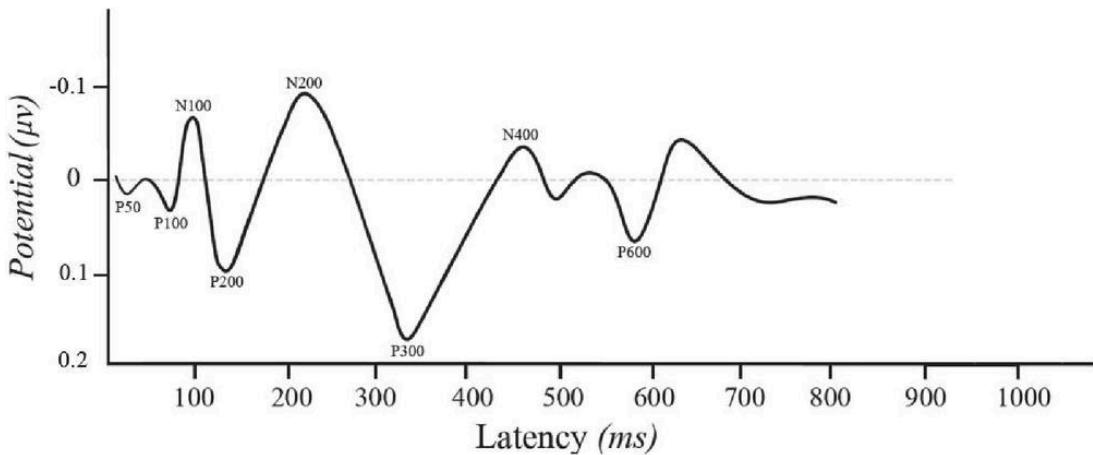


Figura 2.10: Evenimentele asociate unui ERP[29].

Am confirmat că într-adevăr există diferențe între fiecare tip de stimuli unind toți participanții și realizând în Appendix un grafic al mediei răspunsurilor celor trei tipuri de stimuli. Graficul poate fi văzut în figura A.1 din Appendix.

### **2.3.2 Datele drept caracteristici statistice și descompuneri în domeniul frecvențelor**

O altă abordare a extragerii caracteristicilor o reprezintă datele statistice și speciale din semnalul EEG. Pentru a extrage caracteristicile semnalului am utilizat mne-features[26]. Librăria reușește să extragă în total 30 de tipuri de caracteristici. Am restrâns aceste caracteristici la o submulțime mai mică, implementând metoda prezentată în [35]. și anume folosirea algoritmilor genetici împreună cu un clasificator liniar SVM pentru a alege o submulțime de caracteristici din cea inițială.

### **2.3.3 Datele drept caracteristici geometrice Riemann**

Alte modalități, care au recăștigat popularitate recent, constau în folosirea geometriei Riemann. În domeniul semnalelor EEG, o caracteristică importantă o reprezintă matricile de covarianță. Acestea indică impactul unui canal asupra celorlalte canale. O matrice de covarianță este o matrice pătratică, având canalele dispuse pe verticală și orizontală. Pe diagonala principală valorile indică varianța unui canal față de media sa, iar în afara diagonalei matricea indică influența unui canal asupra altui canal. Dacă un astfel de coeficient de covarianță este pozitiv, înseamnă că semnalele se mișcă în același sens, iar dacă este negativ se mișcă în sens opus. Este dovedit faptul că aceste matrici sunt simetrice și pozitiv definite (SPD). Astfel, spațiul din care provin ele nu este unul euclidian ci unul cu neliniar, curbat, și anume spațiul Riemann. Prin urmare, transpunerea din spațiul euclidian în spațiul Riemann are efectul de a reintroduce caracteristicile pierdute din 'aplatizarea' spațiului din care provin matricile de covarianță [32].

Asupra datelor este aplicat algoritmul de denoising XDAWN[24], urmând, pe urmă, ca semnalul să fie transpus într-un spațiu geometric Riemann. Pentru a implementa acești algoritmi m-am folosit de librăria pyRiemann[7]. Algoritmii ce se folosesc de acest principiu și pe care i-am implementat și testat la rândul meu sunt: XDAWNCov + TS + SVM[8], pe care ulterior l-am modificat și am utilizat Linear Discriminant Analysis(LDA) din scikit-learn[22], XDAWN + LDA[23], și, derivat din aceste concepte XDAWN + Cov + TS + LDA. Benchmark-urile MOABB[5] arată că XDAWNCov + TS + SVM și XDAWN + MDM obțin cele mai bune performanțe pentru stimulii de tip P300. Rezultatele obținute de mine sunt vizibile în figura 2.11.

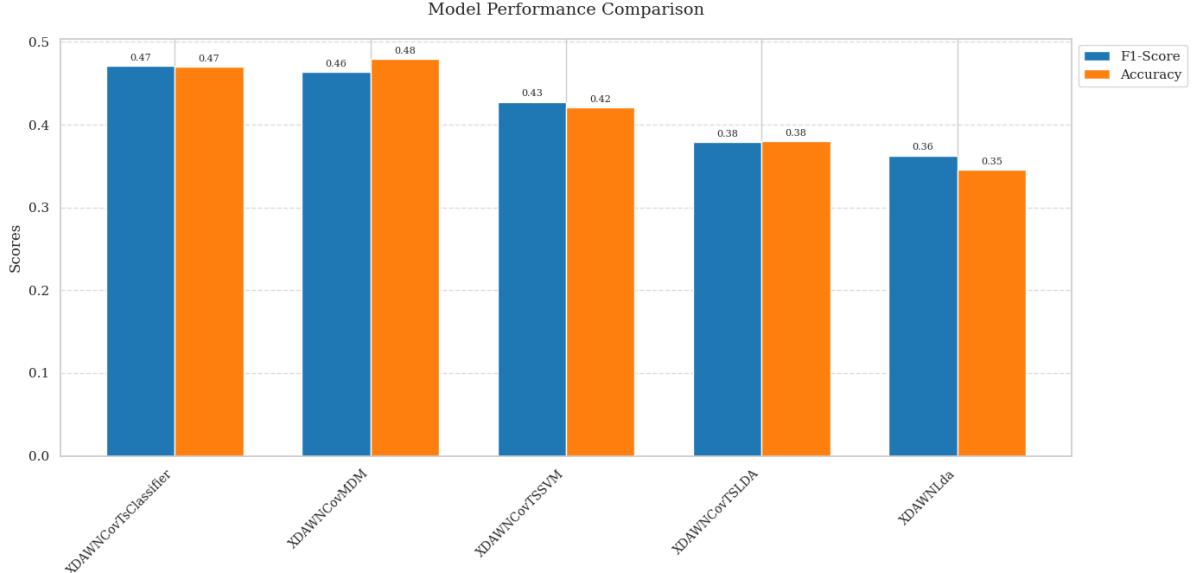


Figura 2.11: Comparație a performanțelor algoritmilor XDAWN.

## 2.4 Modele de clasificare

Privind abordarea modelelor am ales să încerc cât mai multe modele, pornind de la simple clasificatoare liniare, precum SVM-uri din scikit-learn[22] și până la rețelele conoluționale din braindecode[27]. De asemenea, am creat, utilizând keras[9], și modele proprii pentru un mai bun control asupra complexității și funcționalității modelului.

### 2.4.1 Modele clasice din scikit-learn

Pentru a evalua modelele clasice, prima dată am trecut datele de dimensiune(8, 601) adică 8 canale, fiecare canal având 601 puncte de discretizare, aceste 601 puncte provin de la frecvența înregistrărilor de 300 x 2 secunde ce reprezintă durata unei epoci, prin operația de flatten, astfel încât input-ul meu să fie 1-dimensional. Pe urmă, am normalizat datele și le-am încadrat în intervalul (0, 1) utilizând StandardScaler din scikit-learn[22]. În final, fiecare input reprezintă un vector cu 4808 caracteristici.

Utilizând acest pipeline, ce poate fi observat și în figura B.9 din Appendix, am putut să compar sistematic mai multe tipuri de modele. Pentru a evalua performanța modelelor, am utilizat f1-score și accuracy. Pentru a lua în calcul imbalansul claselor, am creat date sintetice utilizând metoda de oversampling SMOTE[18].

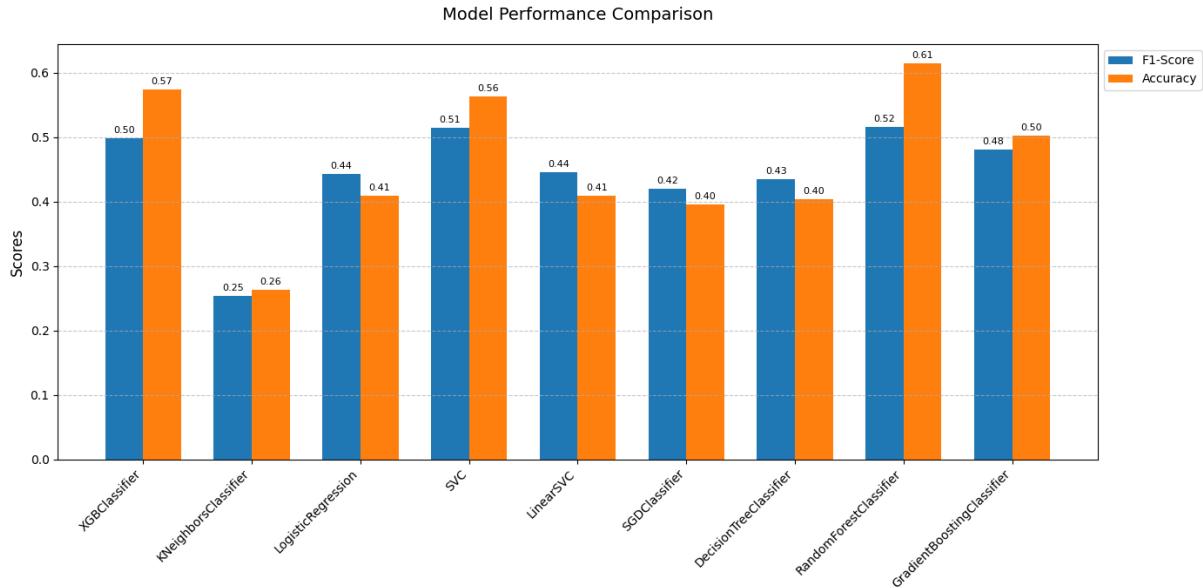


Figura 2.12: Comparație a performanțelor modelelor clasice din scikit-learn[22].

Mai departe, în scopul de a implementa abordarea bazată pe caracteristici statistice și spectrale, am utilizat mne-features[26]. Pentru a extrage un subset din feature-urile totale, am utilizat o librărie specializată în rularea algoritmilor genetici, și anume pygad [11]. Funcția de fitness a fost calculată folosind un tuplu format din f1-score și balanced-accuracy. Configurația specifică poate fi văzută în figura B.8 din Appendix.

Un rezultat intermediar a indicat o acuratețe de 35% și un scor F1 de 38%. Având în vedere performanțele modeste și dimensiunea ridicată a spațiului de căutare, am decis să încrerup rularea experimentului în această configurație. Continuarea optimizării în aceste condiții nu era justificată, întrucât rezultatele obținute nu indicau un potențial de îmbunătățire semnificativă. Rezultatele lucrării care implementează această idee [6], indică faptul că algoritmul genetic a făcut ca acuratețea să varieze cu un maxim de 5%, rezultat care în continuare l-ar plasa sub performanța celorlalte modele încercate.

## 2.4.2 Braindecode

Pentru a avea acces la arhitecturi neuronale avansate, create special pentru domeniul analizei EEG-urilor, am utilizat librăria Braindecode[27]. Ele primesc drept intrare direct semnalul de dimensiune (8, 601). Fiecare model din braindecode trebuie trimis mai departe unui wrapper, fie de regresie fie de clasificare. Apelul wrapper-ului poate fi văzut în figura B.1 din Appendix. În cazul meu l-am folosit pe cel de clasificare. Pentru a mări viteza de procesare, disponând de o placă video, am rulat antrenarea epocilor pe placa video folosind cuda.

La nivelul acestuia am setat funcția de pierdere ca fiind CrossEntropyLoss din librăria pytorch [21], precum și EarlyStopping și LRScheduler drept callback-uri din librăria skorch [33]. Astfel, modelul evită overfitting-ul prin oprirea antrenării în momentul în care nu a mai progresat din punctul de vedere al funcției de pierdere pe setul de validare timp de

20 de epoci. De asemenea, acesta dispune și de un learning rate variabil ce îl ajută să conveargă rapid la o soluție.

$$\text{loss}(x, y) = -\frac{1}{N} \sum_{n=1}^N w_{y_n} \cdot \log \left( \frac{\exp(x_{n,y_n})}{\sum_{c=1}^C \exp(x_{n,c})} \right) \times \mathbb{1}\{y_n \neq \text{ignore\_index}\} \quad (2.1)$$

Putem remarca faptul că formula pentru CrossEntropyLoss conține un  $w_{y_n}$ . Acesta reprezintă un parametru legat de imbalansul claselor. Astfel, modelul îmi permite să abordez ambele modalități de rezolvare a problemei de balansare a claselor. Evaluarea modelelor am făcut-o asemănător cu cea a modelelor clasice, în funcție de f1-score și accuracy.

Cu scopul de a avea o analiză amplă a acestei tehnici, am utilizat toate modelele disponibile din librărie ce au fost dezvoltate în direcția interpretării evenimentelor de tip P300. Astfel, am comparat următoarele modele: TSceptionV1[10], EEGNetv4[17], EEGInceptionERP[25], ATCNet [3][2][4], EEGTCNet[14], TIDNet[16], ShallowFBCSPNet[28].

De asemenea, am creeat noi date folosind augmentări,. Observând că studii din trecut au obținut rezultate mai bune după ce le-au aplicat [34]. Augmentările folosite au fost: deplasarea întregului semnal cu până la 20 de secunde înainte și înapoi, zgomot gaussian, și setarea semnalului la valoarea 0 pentru diverse perioade. Codul cu care am realizat augmentările poate fi văzut în Appendix în figura B.7.

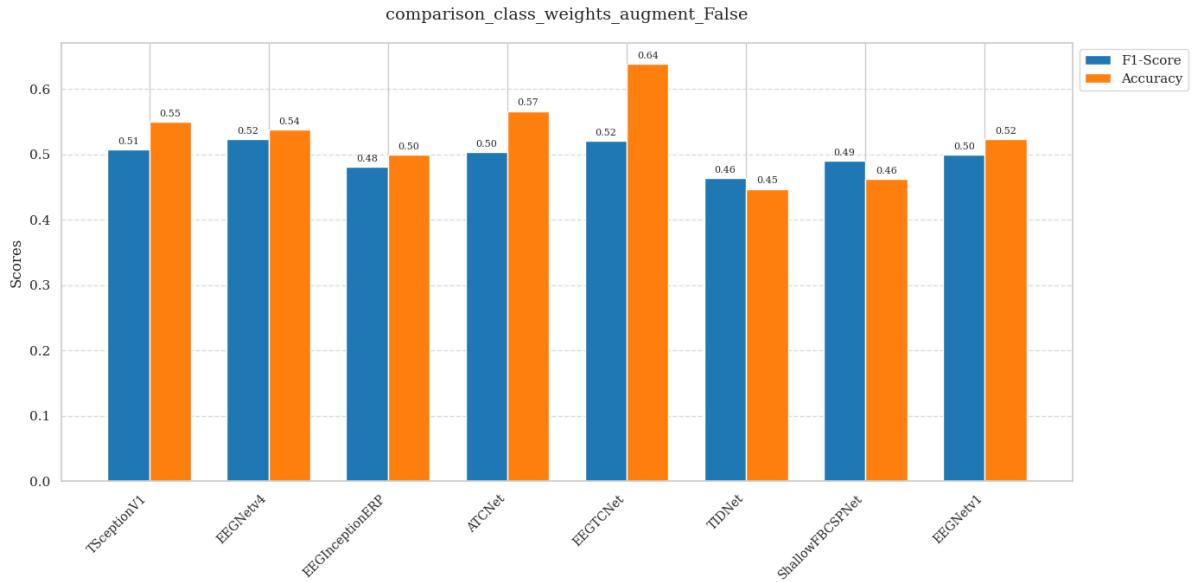


Figura 2.13: Comparație a performanțelor modelelor din braindecode[27].

Este evident din figura 2.13, faptul că EEGTCNet[14] a obținut cele mai bune rezultate. Acestea au fost obținute atunci când modul de tratare a imbalansului claselor a constat în calcularea weight-urilor claselor, trimis mai departe la funcția de pierdere și

atunci când nu au fost aplicate augmentări pe date. Rezultatul celorlalte abordări privind imbalansul claselor și augmentări se află în figurile A.2, A.3 și A.4 din Appendix.

### 2.4.3 Model personal

Am creeat modele personale utilizând librăria Keras[9]. Am ales să folosesc această bibliotecă în defavoarea PyTorch, deoarece Keras oferă un nivel mai ridicat de abstractizare, ceea ce simplifică considerabil procesul de dezvoltare a modelelor. Un alt motiv pentru care am utilizat Keras este faptul că pașii de antrenare, optimizare și validare sunt deja integrați și gestionăți bine de bibliotecă. În contextul acestei lucrări, nu a fost necesară personalizarea acestor pași sau implementarea unor mecanisme avansate de calculare a funcției de pierdere sau de control de date.

Paradigma pe care am abordat-o a fost cea în care trimit modelului semnalul în sine, urmând ca în cadrul modelului să extragem proprietăți ale semnalului folosind rețele convoluționale. Arhitectura modelului meu constă în 3 straturi de conoluție, între ele aflându-se straturi de MaxPooling. La final am aplatizat rezultatul conoluțiilor și am terminat modelul cu 3 straturi dense. Pentru a preveni problem overfitting-ului am aplicat regularizarea L1 și L2, am introdus dropout în straturile dense și am augmentat datele la intrarea în model cu zgomot gausian. Drept optimizator am folosit Adam. Neuronilor de output le-am aplicat funcția de activare softmax pentru a fi compatibil cu funcția de pierdere CategoricalCrossEntropyLoss. Pentru restul neuronilor am utilizat funcțiile de activare ReLU și ELU. Arhitectura modelului detaliată este reprezentată în figurile B.3 și B.4 din Appendix.

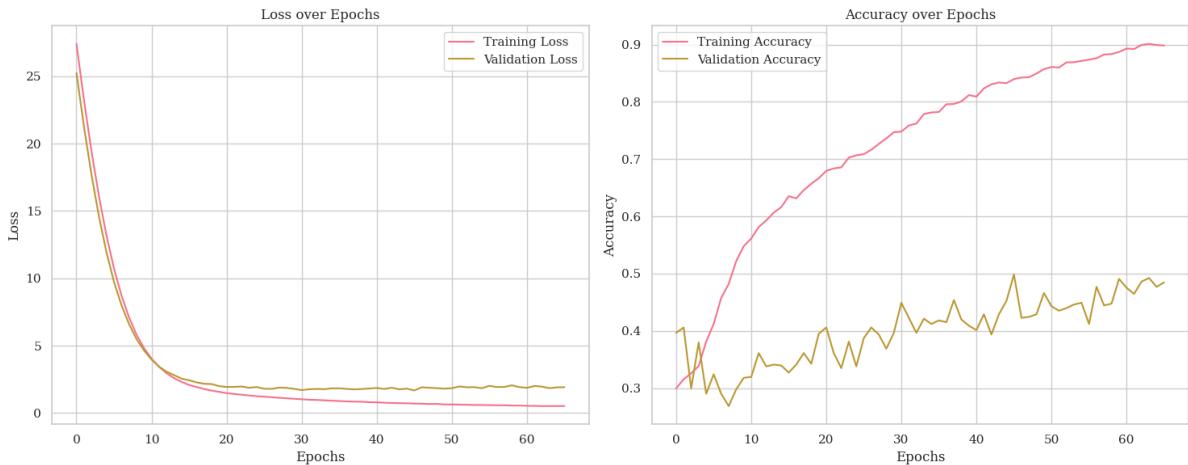


Figura 2.14: Evoluția funcției de pierdere și a acurateții de-a lungul epocilor.

Figura 2.14 ne arată evoluția funcției de pierdere, precum și a acurateții modelului pe parcursul epocilor. Putem observa că modelul reușește să învețe până ajunge la o acuratețe de sub 50%. În evaluarea finală în care am combinat toate rezultatele din urma validării în cruce a modelul am obținut o acuratețe de 47%. Din matricea de confuzie prezentată în figura 2.15 observăm că modelul în continuare are o predispoziție în a prezice

valența neutra (2).

		Precision			
		1	2	3	
		167 3.0%	719 12.8%	192 3.4%	15.49%
Predictions	1	577 10.3%	<b>2247 40.1%</b>	585 10.5%	65.91%
	2	198 3.5%	676 12.1%	<b>237 4.2%</b>	21.33%
	3	17.73%	61.70%	23.37%	<b>47.36%</b>
Recall		1	2	3	

Figura 2.15: Matricea de confuzie rezultată în urma validărilor.

Lipsa capacitatei modelului de a învăța diferențe clare între clase poate fi atribuită metodelor de balansare a acestora. În cazul balansării prin calcularea weight-urilor, modelul acordă o importanță mai mare instantelor în care subiecții au spus că le place/ nu le place imaginea (acestea fiind cele mai rare). Astfel, deși modelul ajunge să învețe bine exemplele din timpul antrenării, performanța sa pe date noi scade, întrucât răspunsul creierului variază semnificativ atât în intensitate, cât și în timp. Punând un accent mai mare pe exemplele rare dintr-un set de date limitat ca dimensiune, modelul nu reușește să surprindă variabilitatea naturală a activității cerebrale nici la nivel intra-subiect (același participant), nici inter-subiect (între participanți diferiți).

Fenomenul acesta este observat și în cadrul balansării folosind SMOTE[18]. Algoritmul de generare al datelor sintetice încearcă să umple golul dintre sample-uri prin interpolare. Totuși, aceste instanțe nu introduc informație nouă, ci doar reflectă o medie a celorlalte două puncte din vecinătatea exemplului sintetic. Mecanismul de generare de date sintetice nu reproduce modul natural de funcționare al creierului uman. Chiar și în condiții de expunere repetată la același stimул, variația răspunsului cerebral nu este rezultatul unei medii liniare între două stări, ci al unui proces complex de interpretare și adaptare.

# Capitolul 3

## Arhitectura aplicației

### 3.1 Împărțirea codului

Codul este împărțit în 4 clase diferite. Ele au rolul de a forma un pipeline ce primește la intrare fișiere în format csv, împreună cu montura electrozilor și ca rezultat oferă obiecte în formatul librăriei MNE[13]. În total, pipeline-ul meu este format din 5 clase: EEGPipelineOptions, EEGSubjectPipeline, EEGDataLoader, EEGPreprocessor și EEGEpochedData.

Prima dată, opțiunile de preprocesare și de formatare în epoci sunt setate utilizând EEGPipelineOptions. Acest obiect este trimis mai departe către EEGSubjectPipeline. Aici, este orchestrat restul pipeline-ului în felul următor: datele din format .csv sunt transformate în format .fif(compatibil cu MNE[13]) utilizând EEGDataLoader. Datele în format .fif sunt transformate în date de forma mne.Raw(semnalul întreg, nesegmentat). Aici sunt totodată și preprocesate utilizând filtrul trece-bandă, ICA, interpolarea, și/sau, optional, normalizare și scalare în intervalul [0, 1]. Ultimul pas îl reprezintă EEGEpochedData, unde semnalul Raw este segmentat în epoci. În funcție de opțiunile alese, aici poate fi aplicat și algoritmul de AutoReject[15] pentru a elimina de epocile corupte. Totodată, în această clasă este situată și funcția de extragere a setului de date, împreună cu labelul acestuia.

### 3.2 Alte librării folosite

#### 3.2.1 Optimizarea încărcării datelor folosind Ray

Pentru a încărca eficient cei 26 de participanți am utilizat librăria Ray[20]. Ray este o librărie specializată în paralelizarea programelor Python, având ca scop principal programele din domeniul de Machine Learning. Modul de paralelizare a constat prin decorarea functiilor de încarcare a datelor cu @ray.remote și așteptarea rezultatului cu ray.get. Timpul de încărcare a datelor participanților a fost redus de la 37 secunde la

18 secunde. Paralelizarea folosind Ray poate fi observată în figura B.2 din Appendix. Această scădere a avut un impact mai accentuat în momentul căutării hiperparametrilor, am putut astfel să dublez totalul parametrilor căutați în același interval de timp.

### 3.2.2 Căutarea hiperparametrilor folosind Optuna

Optuna[1] este o librărie ce se ocupă de căutarea hiperparametrilor. Parametrii căutați au fost cei legați de EEGPipelineOptions. Scopul a fost să înțeleg ce preprocesări trebuie execuțiate pentru performanță maximă. Pentru a rula pipeline-ul meu folosind optuna, am definit o funcție obiectiv, și anume f1 score-ul obținut în urma validării încrucișate. Mai departe, am setat hiperparametrii pe care ar trebui să-i caute librăria. Am salvat cele mai bune parametrii în funcție de scorul F1. Astfel, am putut să cauț sătematic cele mai bune hiperparametrii de preprocesare. Întreg procesul poate fi văzut în detaliu în figura B.5 din Appendix.

# Capitolul 4

## Concluzii

### 4.1 Interpretarea rezultatelor

Observând rezultatele obținute, am ajuns la concluzia că stimulii nu au fost destul de puternici astfel încât să existe o distincție clară între reacția subiecților la aceștia. Iar, dacă privim literatura de specialitate în domeniul EEG-urilor, și în special pe cel al clasificării emoțiilor, putem observa rezultate cel mult medicore[19], chiar și pe seturi de date bine cunoscute. Încercând cât mai multe lucruri și citind lucrări de specialitate am observat că există destule limitări în acest domeniu. Multe dintre limitări sunt legate de metodele de recoltare a datelor și de zgomotul rezultat de acestea, altele sunt legate de diferențele fiziologice dintre oameni, neputând astfel generaliza un stimул unifrom de-a lungul unor subiecți. O analiză mai amplă și precă poate fi facută, momentan, folosind tehnologie non-invazivă, doar pe stimuli de durată mai lungă, unde cel mai important factor este frecvența, sau stimuli mai puternici și mai diversi.

### 4.2 Îmbunătățiri și perspective de viitor

Pentru a îmbunătăți rezultalte, experimentul ar putea fi realuat folosind imagini mai stimulante, afișate pe o perioadă mai lungă de timp. De asemenea, încă o îmbunătățire ar putea fi adusă și prin folosirea unui eyetracker, astfel combinând reacția, cu zona în care s-a uitat participantul când a avut-o. În plus, un alt aspect care ar putea îmbunătăți rezultalte, ar fi creșterea gradului de imersiune folosind, de exemplu, căști de realitate virtuală.

# Bibliografie

- [1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta și Masanori Koyama, „Optuna: A Next-generation Hyperparameter Optimization Framework”, în *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.
- [2] Hamdi Altaheri, Ghulam Muhammad și Mansour Alsulaiman, „Dynamic convolution with multilevel attention for EEG-based motor imagery decoding”, în *IEEE Internet of Things Journal* 10.21 (2023), pp. 18579–18588, doi: [10.1109/JIOT.2023.3281911](https://doi.org/10.1109/JIOT.2023.3281911).
- [3] Hamdi Altaheri, Ghulam Muhammad și Mansour Alsulaiman, „Physics-Informed Attention Temporal Convolutional Network for EEG-Based Motor Imagery Classification”, în *IEEE Transactions on Industrial Informatics* 19.2 (2023), pp. 2249–2258, doi: [10.1109/TII.2022.3197419](https://doi.org/10.1109/TII.2022.3197419).
- [4] Hamdi Altaheri, Ghulam Muhammad, Mansour Alsulaiman, Syed Umar Amin, Ghadir Ali Altuwaijri, Wadood Abdul, Mohamed A Bencherif și Mohammed Faisal, „Deep learning techniques for classification of electroencephalogram (EEG) motor imagery (MI) signals: A review”, în *Neural Computing and Applications* 35.20 (2023), pp. 14681–14722, doi: [10.1007/s00521-021-06352-5](https://doi.org/10.1007/s00521-021-06352-5).
- [5] Bruno Aristimunha, Igor Carrara, Pierre Guetschel, Sara Sedlar, Pedro Rodrigues, Jan Sosulski, Divyesh Narayanan, Erik Bjareholt, Quentin Barthelemy, Robin Tibor Schirrmeister, Reinmar Kobler, Emmanuel Kalunga, Ludovic Darmet, Cattan Gregoire, Ali Abdul Hussain, Ramiro Gatti, Vladislav Goncharenko, Jordy Thielen, Thomas Moreau, Yannick Roy, Vinay Jayaram, Alexandre Barachant și Sylvain Chevallier, *Mother of all BCI Benchmarks*, versiunea v1.2.0, 2025, doi: [10.5281/zenodo.10034223](https://doi.org/10.5281/zenodo.10034223), URL: <https://github.com/NeuroTechX/moabb>.
- [6] John Atkinson și Daniel Campos, „Improving BCI-based emotion recognition by combining EEG feature selection and kernel classifiers”, în *Expert Systems with Applications* 47 (2016), pp. 35–41, ISSN: 0957-4174, doi: <https://doi.org/10.1016/j.eswa.2015.10.049>, URL: <https://www.sciencedirect.com/science/article/pii/S0957417415007538>.

- [7] Alexandre Barachant, Quentin Barthélemy, Jean-Rémi King, Alexandre Gramfort, Sylvain Chevallier, Pedro L. C. Rodrigues, Emanuele Olivetti, Vladislav Goncharenko, Gabriel Wagner vom Berg, Ghiles Reguig, Arthur Lebeurrier, Erik Bjäreholt, Maria Sayu Yamamoto, Pierre Clisson, Marie-Constance Corsi, Igor Carrara, Apolline Mellot, Bruna Junqueira Lopes, Brent Gaisford, Ammar Mian, Anton Andreev, Gregoire Cattan și Arthur Lebeurrier, *pyRiemann*, versiunea v0.8, Feb. 2025, DOI: [10.5281/zenodo.593816](https://doi.org/10.5281/zenodo.593816), URL: <https://doi.org/10.5281/zenodo.593816>.
- [8] Sylvain Chevallier, Guillaume Bao, Mayssa Hammami, Fabienne Marlats, Louis Mayaud, Djillali Annane, Frédéric Lofaso și Eric Azabou, „Brain-Machine Interface for Mechanical Ventilation Using Respiratory-Related Evoked Potential”, în *International Conference on Artificial Neural Networks*, Oct. 2018, pp. 662–671, ISBN: 978-3-030-01423-0, DOI: [10.1007/978-3-030-01424-7\\_65](https://doi.org/10.1007/978-3-030-01424-7_65).
- [9] François Chollet et al., *Keras*, <https://keras.io>, 2015.
- [10] Yi Ding, Neethu Robinson, Qiuhan Zeng și Duo Chen, „TSception: A Deep Learning Framework for Emotion Detection Using EEG”, în *Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN)*, Singapore: IEEE, Iul. 2020, p. 8, DOI: [10.1109/IJCNN48605.2020.9206750](https://doi.org/10.1109/IJCNN48605.2020.9206750), URL: [https://www.researchgate.net/publication/347156394\\_TSceptionA\\_Deep\\_Learning\\_Framework\\_for\\_Emotion\\_Detection\\_Using\\_EEG](https://www.researchgate.net/publication/347156394_TSceptionA_Deep_Learning_Framework_for_Emotion_Detection_Using_EEG).
- [11] Ahmed Fawzy Gad, „Pygad: An intuitive genetic algorithm python library”, în *Multimedia Tools and Applications* (2023), pp. 1–14.
- [12] Alexandre Gramfort, Martin Luessi, Eric Larson, Denis A. Engemann, Daniel Strohmeier, Christian Brodbeck, Roman Goj, Mainak Jas, Teon Brooks, Lauri Parkkonen și Matti Hämäläinen, „MEG and EEG data analysis with MNE-Python”, în *Frontiers in Neuroscience* Volume 7 - 2013 (2013), ISSN: 1662-453X, DOI: [10.3389/fnins.2013.00267](https://doi.org/10.3389/fnins.2013.00267), URL: [https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2013.00267](https://doi.org/10.3389/fnins.2013.00267).
- [13] Alexandre Gramfort, Martin Luessi, Eric Larson, Denis A. Engemann, Daniel Strohmeier, Christian Brodbeck, Roman Goj, Mainak Jas, Teon Brooks, Lauri Parkkonen și Matti S. Hämäläinen, „MEG and EEG Data Analysis with MNE-Python”, în *Frontiers in Neuroscience* 7.267 (2013), pp. 1–13, DOI: [10.3389/fnins.2013.00267](https://doi.org/10.3389/fnins.2013.00267).
- [14] Thorir Mar Ingolfsson, Michael Hersche, Xiaying Wang, Nobuaki Kobayashi, Lukas Cavigelli și Luca Benini, *EEG-TCNet: An Accurate Temporal Convolutional Network for Embedded Motor-Imagery Brain-Machine Interfaces*, 2020, arXiv: [2006.00622 \[eess.SP\]](https://arxiv.org/abs/2006.00622), URL: <https://arxiv.org/abs/2006.00622>.

- [15] Mainak Jas, Denis Engemann, Federico Raimondo, Yousra Bekhti și Alexandre Gramfort, „Automated rejection and repair of bad trials in MEG/EEG”, în *6th International Workshop on Pattern Recognition in Neuroimaging (PRNI)*, Trento, Italy, Iun. 2016, URL: <https://hal.science/hal-01313458>.
- [16] Demetres Kostas și Frank Rudzicz, „Thinker invariance: enabling deep neural networks for BCI across more people”, în *Journal of Neural Engineering* 17.5 (Oct. 2020), p. 056008, DOI: [10.1088/1741-2552/abb7a7](https://dx.doi.org/10.1088/1741-2552/abb7a7), URL: <https://dx.doi.org/10.1088/1741-2552/abb7a7>.
- [17] Vernon J Lawhern, Amelia J Solon, Nicholas R Waytowich, Stephen M Gordon, Chou P Hung și Brent J Lance, „EEGNet: a compact convolutional neural network for EEG-based brain-computer interfaces”, în *Journal of Neural Engineering* 15.5 (2018), p. 056013, URL: <http://stacks.iop.org/1741-2552/15/i=5/a=056013>.
- [18] Guillaume Lemaître, Fernando Nogueira și Christos K. Aridas, „Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning”, în *Journal of Machine Learning Research* 18.17 (2017), pp. 1–5, URL: <http://jmlr.org/papers/v18/16-365.html>.
- [19] Jesus Arturo Mendivil Sauceda, Bogart Yail Marquez și José Jaime Esqueda Elizondo, „Emotion Classification from Electroencephalographic Signals Using Machine Learning”, în *Brain Sciences* 14.12 (2024), ISSN: 2076-3425, DOI: [10.3390/brainsci14121211](https://www.mdpi.com/2076-3425/14/12/1211), URL: <https://www.mdpi.com/2076-3425/14/12/1211>.
- [20] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I. Jordan și Ion Stoica, „Ray: a distributed framework for emerging AI applications”, în *Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation*, OSDI’18, Carlsbad, CA, USA: USENIX Association, 2018, pp. 561–577, ISBN: 9781931971478.
- [21] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai și Soumith Chintala, „PyTorch: An Imperative Style, High-Performance Deep Learning Library”, în *Advances in Neural Information Processing Systems* 32, Curran Associates, Inc., 2019, pp. 8024–8035, URL: <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.
- [22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Co-

- urnapeau, M. Brucher, M. Perrot și E. Duchesnay, „Scikit-learn: Machine Learning in Python”, în *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.
- [23] Bertrand Rivet, Antoine Souloumiac, Virginie Attina și Guillaume Gibert, „xDAWN Algorithm to Enhance Evoked Potentials: Application to Brain–Computer Interface”, în *Biomedical Engineering, IEEE Transactions on* 56 (Sept. 2009), pp. 2035–2043, DOI: [10.1109/TBME.2009.2012869](https://doi.org/10.1109/TBME.2009.2012869).
- [24] Bertrand Rivet\*, Antoine Souloumiac, Virginie Attina și Guillaume Gibert, „xDAWN Algorithm to Enhance Evoked Potentials: Application to Brain–Computer Interface”, în *IEEE Transactions on Biomedical Engineering* 56.8 (2009), pp. 2035–2043, DOI: [10.1109/TBME.2009.2012869](https://doi.org/10.1109/TBME.2009.2012869).
- [25] Eduardo Santamaría-Vázquez, Víctor Martínez-Cagigal, Fernando Vaquerizo-Villar și Roberto Hornero, „EEG-Inception: A Novel Deep Convolutional Neural Network for Assistive ERP-Based Brain-Computer Interfaces”, în *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 28.12 (2020), pp. 2773–2782, DOI: [10.1109/TNSRE.2020.3048106](https://doi.org/10.1109/TNSRE.2020.3048106).
- [26] Jean-Baptiste Schiratti, Jean-Eudes Le Douget, Michel Le van Quyen, Slim Essid și Alexandre Gramfort, „An ensemble learning approach to detect epileptic seizures from long intracranial EEG recordings”, în *ICASSP 2018 - 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, Canada: IEEE, Apr. 2018, URL: <https://hal.science/hal-01724272>.
- [27] Robin Tibor Schirrmeyer, Jost Tobias Springenberg, Lukas Dominique Josef Fiederer, Martin Glasstetter, Katharina Eggensperger, Michael Tangermann, Frank Hutter, Wolfram Burgard și Tonio Ball, „Deep learning with convolutional neural networks for EEG decoding and visualization”, în *Human Brain Mapping* (Aug. 2017), ISSN: 1097-0193, DOI: [10.1002/hbm.23730](https://doi.org/10.1002/hbm.23730), URL: <http://dx.doi.org/10.1002/hbm.23730>.
- [28] Robin Tibor Schirrmeyer, Jost Tobias Springenberg, Lukas Dominique Josef Fiederer, Martin Glasstetter, Katharina Eggensperger, Michael Tangermann, Frank Hutter, Wolfram Burgard și Tonio Ball, „Deep learning with convolutional neural networks for EEG decoding and visualization”, în *Human Brain Mapping* 38.11 (2017), pp. 5391–5420, DOI: <https://doi.org/10.1002/hbm.23730>, eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/hbm.23730>, URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/hbm.23730>.
- [29] Wanus Srimaharaj și Roungsan Chaisricharoen, „A Novel Processing Model for P300 Brainwaves Detection”, în *Journal of Web Engineering* 20.8 (Nov. 2021), pp. 2545–2570, DOI: [10.13052/jwe1540-9589.20815](https://doi.org/10.13052/jwe1540-9589.20815), URL: <https://journals.riverpublishers.com/index.php/JWE/article/view/4873>.

- [30] Samuel Sutton, Margery Braren, Joseph Zubin și E. R. John, „Evoked-Potential Correlates of Stimulus Uncertainty”, în *Science* 150.3700 (1965), pp. 1187–1188, DOI: [10.1126/science.150.3700.1187](https://doi.org/10.1126/science.150.3700.1187), eprint: <https://www.science.org/doi/pdf/10.1126/science.150.3700.1187>, URL: <https://www.science.org/doi/abs/10.1126/science.150.3700.1187>.
- [31] The pandas development team, *pandas-dev/pandas: Pandas*, versiunea latest, Feb. 2020, DOI: [10.5281/zenodo.3509134](https://doi.org/10.5281/zenodo.3509134), URL: <https://doi.org/10.5281/zenodo.3509134>.
- [32] Imad Eddine Tibermacine, Samuele Russo, Ahmed Tibermacine, Abdelaziz Rabehi, Bachir Nail, Kamel Kadri și Christian Napoli, *Riemannian Geometry-Based EEG Approaches: A Literature Review*, 2024, arXiv: [2407.20250 \[eess.SP\]](https://arxiv.org/abs/2407.20250), URL: <https://arxiv.org/abs/2407.20250>.
- [33] Marian Tietz, Thomas J. Fan, Daniel Nouri, Benjamin Bossan și skorch Developers, *skorch: A scikit-learn compatible neural network library that wraps PyTorch*, Iul. 2017, URL: <https://skorch.readthedocs.io/en/stable/>.
- [34] Fang Wang, Sheng-hua Zhong, Jianfeng Peng, Jianmin Jiang și Yan Liu, „Data Augmentation for EEG-Based Emotion Recognition with Deep Convolutional Neural Networks”, în *MultiMedia Modeling*, ed. de Klaus Schoeffmann, Thanarat H. Chalidabhongse, Chong Wah Ngo, Supavadee Aramvith, Noel E. O’Connor, Yo-Sung Ho, Moncef Gabbouj și Ahmed Elgammal, Cham: Springer International Publishing, 2018, pp. 82–93, ISBN: 978-3-319-73600-6, DOI: [10.1007/978-3-319-73600-6\\_8](https://doi.org/10.1007/978-3-319-73600-6_8).
- [35] Lei Wang, Guizhi Xu, Jiang Wang, Shuo Yang, Lei Guo și Weili Yan, „GA-SVM based Feature Selection and Parameters Optimization for BCI Research”, în *2011 Seventh International Conference on Natural Computation*, IEEE, 2011, pp. 580–584, DOI: [10.1109/ICNC.2011.6022564](https://doi.org/10.1109/ICNC.2011.6022564).
- [36] Kyungho Won, Moonyoung Kwon, Minkyu Ahn și Sung Chan Jun, „EEG Dataset for RSVP and P300 Speller Brain-Computer Interfaces”, în *Scientific Data* 9.1 (2022), p. 388, ISSN: 2052-4463, DOI: [10.1038/s41597-022-01509-w](https://doi.org/10.1038/s41597-022-01509-w), URL: <https://doi.org/10.1038/s41597-022-01509-w>.
- [37] Geoffrey F. Woodman, „A brief introduction to the use of event-related potentials in studies of perception and attention”, în *Attention, Perception, & Psychophysics* 72.8 (2010), pp. 2031–2046, DOI: [10.3758/APP.72.8.2031](https://doi.org/10.3758/APP.72.8.2031).

## **Anexa A**

### **Tabele cu rezultate**

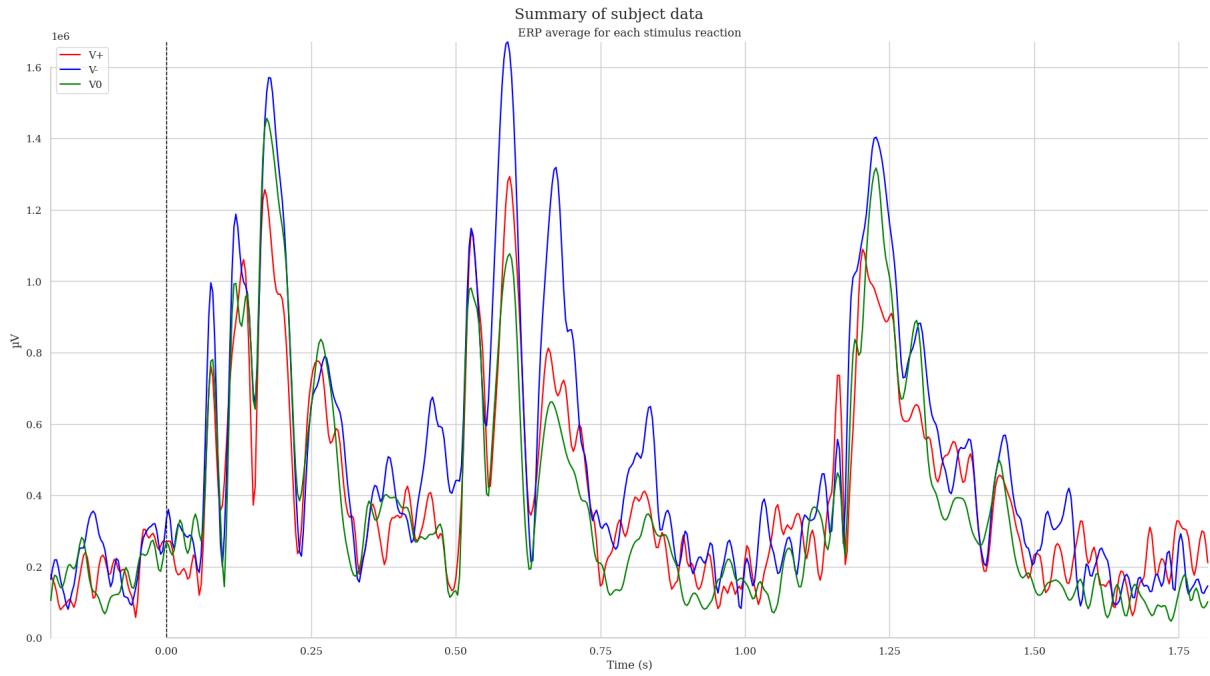


Figura A.1: Raspunsul mediu pentru fiecare tip de eveniment.

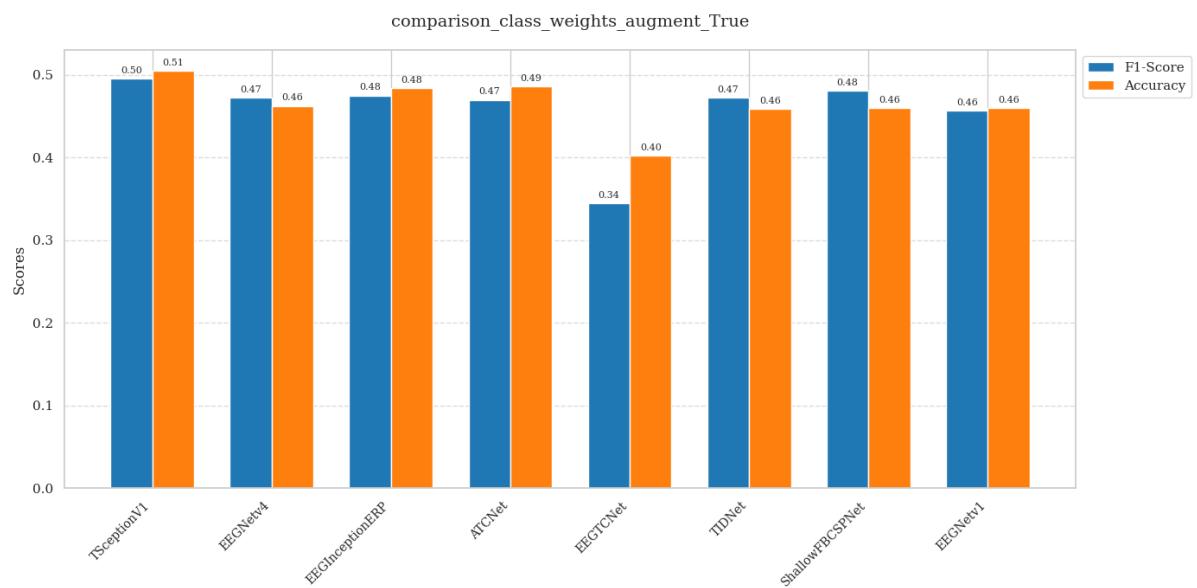


Figura A.2: Performanta modelelor cu weight-uri de clase si augmentari.

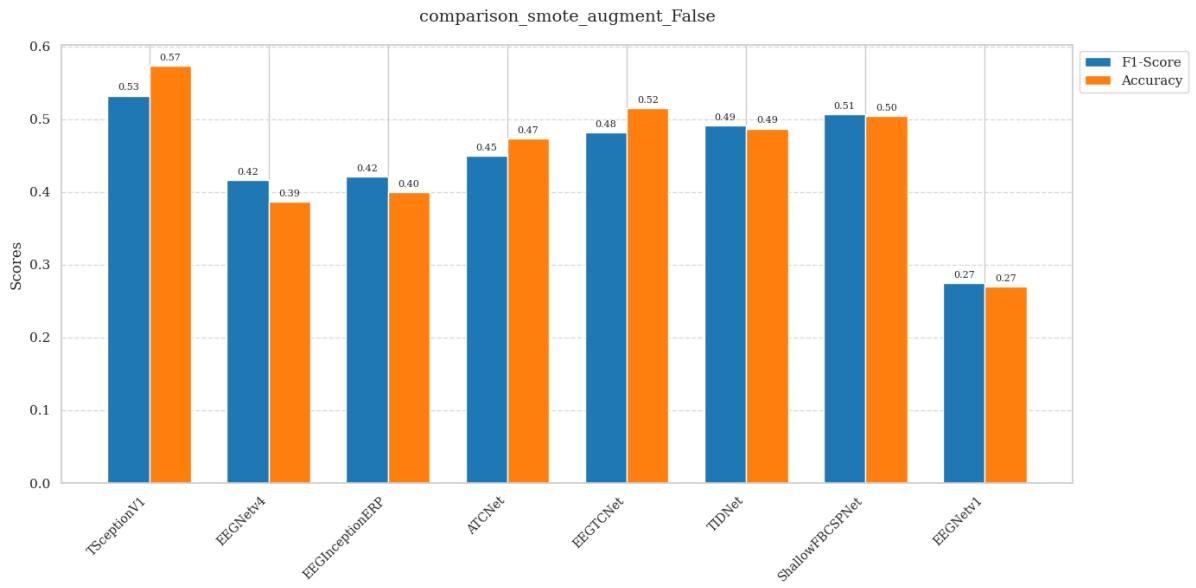


Figura A.3: Performanța modelelor folosind date sintetice, fără augmentări.

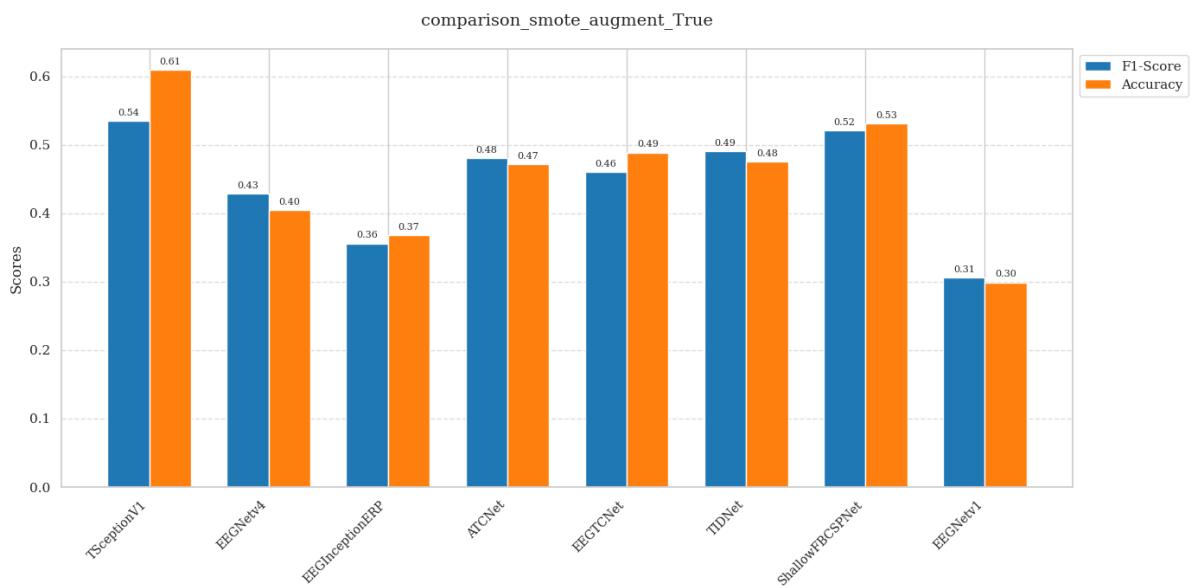


Figura A.4: Performanța modelelor, folosind date sintetice și augmentări.

# Anexa B

## Cod sursă

```
1 classifier = braindecode.classifier.EEGClassifier(  
2     model,  
3     criterion=torch.nn.CrossEntropyLoss,  
4     criterion_weight=torch.tensor(class_weights, dtype=torch.float32) if  
        imbalance_mode=='class_weights' else None,  
5     train_split=predefined_split(torch.utils.data.TensorDataset(torch.  
        tensor(X_valid).float(), torch.tensor(y_valid).long())),  
6     device='cuda',  
7     max_epochs=1000,  
8     callbacks=[skorch.callbacks.EarlyStopping(patience=20, load_best=True),  
                 skorch.callbacks.Checkpoint(), skorch.callbacks.LRScheduler()],  
9 )
```

Figura B.1: Apelarea wrapper-ului din braindecode.

```
1 @ray.remote  
2 def create_worker(participant, options):  
3     return EEGSubjectPipeline(participant, participant.replace("raw.csv", "  
        quizz.xlsx"), 'dsi-24.elc', options)  
4  
5 workers = []  
6 for participant in participants:  
7     workers.append(create_worker.remote(participant, options))  
8  
9 all_pipelines = ray.get(workers)  
10  
11 ray.shutdown()
```

Figura B.2: Paralelizare utilizând Ray.

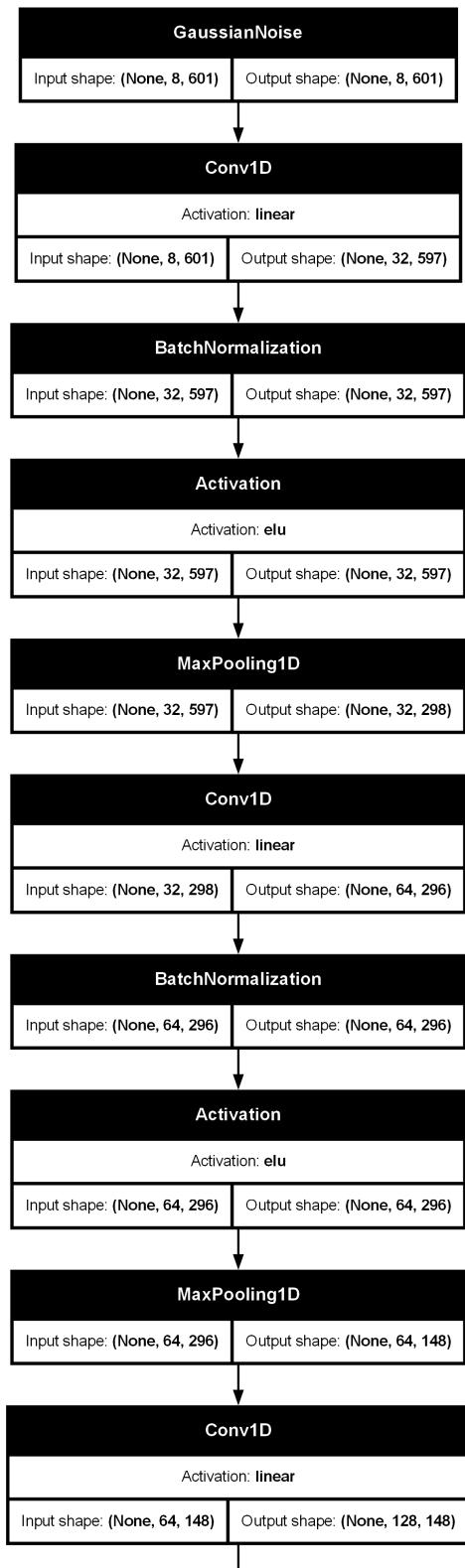


Figura B.3: Model personal, partea 1.

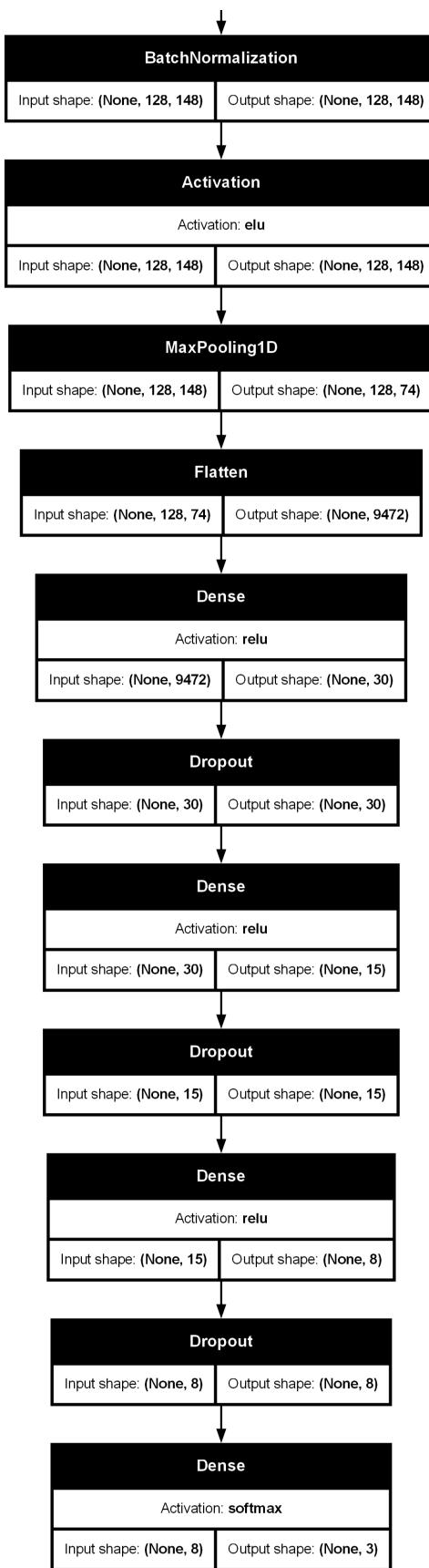


Figura B.4: Model personal, partea 2.

```

1 def objective(trial):
2     options = EEGpipelineOptions()
3
4     options.interpolate_bad = trial.suggest_categorical('interpolate_bad',
5         [False, True])
6     options.apply_minmax_scaling_raw = trial.suggest_categorical(
7         'apply_minmax_scaling_raw', [False, True])
8     options.apply_eog_regression = trial.suggest_categorical(
9         'apply_eog_regression', [False, True])
10    options.apply_normalisation_raw = trial.suggest_categorical(
11        'apply_normalisation_raw', [False, True])
12    options.apply_eeg_reference = trial.suggest_categorical(
13        'apply_eeg_reference', [False, True])
14
15    options.apply_filters_raw = trial.suggest_categorical(
16        'apply_filters_raw', [False, True])
17    options.apply_auto_ica_raw = False
18
19    options.apply_manual_ica = False
20    options.ica_components_raw = 20
21    options.ica_threshold_raw = 2.5
22    options.apply_autoreject = trial.suggest_categorical('apply_autoreject'
23        , [False, True])
24    options.l_freq_raw = 4
25    options.h_freq_raw = 40
26    options.t_min = -0.2
27    options.t_max = 1.8
28    options.apply_xdawn_denoising = False
29    options.apply_epoch_baseline = True
30
31 @ray.remote
32 def create_worker(participant, options):
33     return EEGsubjectPipeline(participant, participant.replace("raw.csv", "quizz.xlsx"), "dsi-24.elc", options)
34
35 workers = []
36 for participant in participants:
37     workers.append(create_worker_remote(participant, options))
38 all_pipelines = ray.get(workers)
39
40 ray.shutdown()
41
42 return get_model_metrics(braindecode.models.EEGNetv4(n_channels=n_chans,
43     n_outputs=3, sfreq=300, n_times=601), all_pipelines)[1]
44
45 def callback(study, trial):
46     if study.best_trial == trial:
47         filehandler = open("output/braindecode_models/best_study.obj", "wb"
48         )
49         pkl.dump(study, filehandler)
50         filehandler.close()
51
52 study = optuna.create_study(direction="maximize")
53 study.optimize(objective, n_trials = 64, callbacks=[callback])

```

Figura B.5: Cautarea hiperparametrilor folosind optuna.

```

1 def preprocess(self, apply_normalization = True, apply_minmax_scaling=True,
2                 apply_filters = True, apply_ica = True, apply_eog_regression
3                 = True,
4                 apply_manual_ica=True, apply_eeg_reference = True):
5     if self.__isPreprocessed == True:
6         return self
7     self.__isPreprocessed = True
8     if apply_eeg_reference:
9         self.__apply_eeg_reference()
10    if apply_filters:
11        self.__filter_raw()
12    if apply_eog_regression:
13        self.__eog_regression()
14    if apply_ica:
15        self.__apply_auto_ica()
16    if apply_manual_ica:
17        self.__apply_ica()
18    if apply_normalization:
19        self.__normalize_raw()
20    if apply_minmax_scaling:
21        self.__minmax_scaling_raw()
22    return self

```

Figura B.6: Parametrizarea preprocesării semnalului.

```

1 if augment:
2     X_train_jittered = add_temporal_jitter(X_train, jitter_range_ms=20,
3                                             sampling_rate=300)
4     X_train = np.concatenate([X_train, X_train_jittered], axis=0)
5     y_train = np.concatenate([y_train, y_train], axis=0)
6
7     X_train = torch.Tensor(X_train)
8     y_train = torch.Tensor(y_train)
9     X_train_noised, y_noised = braindecode.augmentation.GaussianNoise(
10        probability=0.5) \
11        .operation(X_train, y_train, std=0.1)
12     X_train = X_train.detach().cpu().numpy()
13     y_train = y_train.detach().cpu().numpy()
14
15     X_train_noised = X_train_noised.detach().cpu().numpy()
16     y_noised = y_noised.detach().cpu().numpy()
17     X_train = np.concatenate([X_train, X_train_noised], axis=0)
18     y_train = np.concatenate([y_train, y_noised], axis=0)
19
20     X_train = torch.Tensor(X_train)
21     y_train = torch.Tensor(y_train)
22     X_train_noised, y_noised = braindecode.augmentation.SmoothTimeMask(
23        probability=0.5) \
24        .operation(X_train, y_train, mask_start_per_sample=torch.randint(
25            low=0, high=400, size=(X_train.shape[0],)),
26            mask_len_samples=40)
27     X_train = X_train.detach().cpu().numpy()
28     y_train = y_train.detach().cpu().numpy()
29
30     X_train_noised = X_train_noised.detach().cpu().numpy()
31     y_noised = y_noised.detach().cpu().numpy()
32     X_train = np.concatenate([X_train, X_train_noised], axis=0)
33     y_train = np.concatenate([y_train, y_noised], axis=0)

```

Figura B.7: Codul folosit pentru argumentarea datelor.

```

1 import pygad
2
3 all_features = ["mean", "variance", "std", "ptp_amp", "skewness", "kurtosis",
4                 "rms", "quantile", "hurst_exp",
5                 "app_entropy", "samp_entropy", "decorr_time", "pow_freq_bands", "hjorth_mobility_spect", "hjorth_complexity_spect",
6                 "hjorth_mobility", "hjorth_complexity", "higuchi_fd", "katz_fd", "zero_crossings", "line_length", "spect_slope",
7                 "spect_entropy", "svd_entropy", "svd_fisher_info", "energy_freq_bands", "spect_edge_freq", "wavelet_coef_energy",
8                 "teager_kaiser_energy"]
9
10 desired_output1 = 1
11 desired_output2 = 1
12
13 def fitness_func(ga_instance, solution, solution_idx):
14     model = create_feature_pipeline(sklearn.neighbors.KNeighborsClassifier()
15                                     (), [x[0] for x in zip(all_features, solution) if x[1] > 0])
16     output1, output2, _ = get_model_metrics(model, all_pipelines)
17     return [output1, output2]
18
19 num_generations = 10
20 num_parents_mating = 10
21
22 sol_per_pop = 20
23 num_genes = len(all_features)
24
25 ga_instance = pygad.GA(num_generations=num_generations,
26                         num_parents_mating = num_parents_mating,
27                         sol_per_pop = sol_per_pop,
28                         num_genes = num_genes,
29                         fitness_func=fitness_func,
30                         parent_selection_type='nsga2',
31                         parallel_processing=('process', 8),
32                         save_best_solutions=True,
33                         save_solutions=True
34 )
35
36 ga_instance.run()

```

Figura B.8: Configurația codului folosit pentru algoritmul genetic.

```

1 def create_pipeline(model):
2     class Normalizer(sklearn.base.BaseEstimator, sklearn.base.
3         TransformerMixin):
4         def fit(self, X, y=None):
5             self.mean = X.mean()
6             self.std = X.std()
7             return self
8
9         def transform(self, X):
10            return (X - self.mean) / self.std
11
12    class FlattenTransformer(sklearn.base.BaseEstimator, sklearn.base.
13        TransformerMixin):
14        def fit(self, X, y=None):
15            return self
16
17        def transform(self, X):
18            return X.reshape(X.shape[0], -1)
19
20    steps = [("flatten", FlattenTransformer()), ("normalize", Normalizer())
21          , ("scaler", sklearn.preprocessing.StandardScaler()), ("model",
22              model)]
23    pipe = Pipeline(steps)
24    return pipe

```

Figura B.9: Pipeline folosit pentru a normaliza și aplatiza datele trimise modelelor clasice.