Universitatea Politehnica Timișoara
Facultatea de Automatică și Calculatoare
Departamentul Automatică și
Informatică Aplicată

CLASIFICAREA SEMNALELOR EEG FOLOSIND REȚELE CONVOLUȚIONALE PENTRU DETERMINAREA STĂRII MENTALE

Proiect de Diplomă

Năsui Alexandru-Andrei

Conducător științific
Ș.l.dr.ing. **Ana Maria DAN**

Timișoara
Iunie, 2020

Cuprins

Capitolul 1

Introducere

1.1 Tema și obiectivele lucrării

Această lucrare folosește rețelele neuronale convoluționale pentru clasificarea a trei clase/stării mentale diferite. Etapele parcurse în realizarea lucrării au fost următoarele:

1. Determinarea metodei de clasificare: clasificarea se bazează pe reprezentarea informațiilor statistice și spectrale a undelor cerebrale sub forma unor imagini alb-negru

2. Extragerea datelor EEG: datele EEG *(Electroencefalograma)* au fost extrase cu ajutorul căștii valabile comercial, *Muse 2016*. Pentru fiecare din cele trei clase/stări mentale a fost stabilită câte o activitate pentru a putea extrage date relevante

3. Prelucrarea datelor și extragerea atributelor: datele extrase au fost prelucrate și etichetate, rezultând 414 atribute și clasa de care aparțin, neutru, concentrat sau relaxat. După extragerea atributelor, au fost selectate și normalizate în intervalul $[0, 1]$ 400 de atribute pentru a putea reprezenta o imagine alb-negru de dimensiunea 20×20

4. Realizarea modelului: a fost implementată o rețea convoluțională

pentru clasificarea datelor rezultate. După antrenarea clasificatorului cu aceste imagini, acuratețea acestuia la prezicerea imaginilor, care nu se aflau in setul de date de antrenare, a fost de $\approx 92\%$

1.2 Încadrare temă

Tehnicile de învățare automată urmăresc crearea unor modele matematice bazate pe seturi de date inițiale, denumite *seturi de antrenare (training data)*, care pot generaliza informațiile din acestea, iar mai apoi să prezică răspunsul pentru seturi de date necunoscute. Învățarea automată este folosită intr-o largă gamă de aplicații, precum filtrarea mesajelor e-mail de tip spam de cele autentice, răspunsurile date de către motoarele de căutare, clasificarea celulelor tumorale in benigne sau maligne, recunoașterea facială, recunoașterea diverselor obiecte, recunoașterea limbajului vorbit și scris, și mai nou la conducerea automată a mașinilor.

Cele mai multe tehnici de învățare automată fac parte din una dintre cele trei categori:

- Învățare supervizată (Supervised Learning)
- Învățare nesupervizată (Unsupervised Learning)
- Învățare cu întărire (Reinforcement Learning)

Învățarea supervizată

Învățarea supervizată, în momentul de fața este cea mai răspandită metodă folosită în practică. Principiul din spatele acesteia constând în construirea unui model matematic, prin diferite tehnici, bazat pe un set de date etichetate. Acest set de date etichetate este alcătuit din înregistrări care reprezintă o corespondență intre atribute (intrări) si o clasă (ieșire). Astfel, se urmărește generalizarea acestor corespondețe si

posibilitatea prezicerii clasei unei înregistrări care nu aparține de datele folosite la învățare. Unii dintre cei mai folosiți algoritmi de învățare supervizată sunt:

- Arbori de decizie
- Metode de regresie
- Algoritmi genetici
- Rețele neuronale artificiale
- Mașini cu vector suport
- Rețele Bayesiene

Învățarea nesupervizată

Procesul de învățare nesupervizată diferă față de cel amintit anterior prin faptul că acesta folosește un set de date de antrenare neetichetat. Algoritmii primesc doar un set de atribute (date de intrare), ne știind ieșirea asociată acestora. Aceștia caută in aceste date asemănări și deosebiri, bazându-se pe proprietățile statistice a datelor. Printre cele mai răspândite tehnici se numără:

- Tehnici de grupare
  - Grupare ierarhizată
  - Tehnica k-means
- Hărți cu auto-organizare
  - Rețele Kohonen
- Modele Markov cu stări invizibile

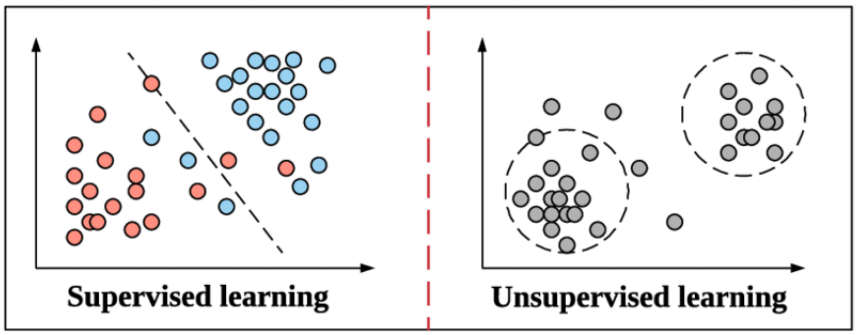


Figura 1.1: Diferența dintre modul de funcționare al învățării supervizare și învățării nesupervizate [1]

Învățarea cu întărire

Învățarea cu întărire este o metodă de învățare prin interacțiuni repetate a unui *agent (software agent)* cu mediul, cu urmărirea atingerii unui anumit scop (Figura 1.2). Interacțiunile se bazează pe acțiunile luate de agent la stimulii mediului, pentru care v-a primit o recompensă de la acesta în funcție de beneficiul adus îndeplinirii scopului. Recompensele primite au rolul de a îmbunătății capacitatea agentului de a lua cea mai bună decizie din starea în care acesta se afla la momentul acțiunii. Scopul pe termen lung al agentului este maximizarea numărului de recompense primite. După repetate interacțiuni cu mediul, capacitatea agentului de a lua decizii bune se v-a crește, iar drumul acestuia prin mediu va tinde spre optim.

1.3 Soluții existente

Apariția unor soluții comerciale *low-cost* non-invazive a facut posibilă înregistrarea și analiza undele cerebrale în afara domeniului medical [3]. În principal, aceste dispozitive sunt folosite în activități simple de interfațare a creierului cu calculatorul *(BCI - Brain-Computer Interface)*. Raportul zgomot-semnal util mare și eșantionarea

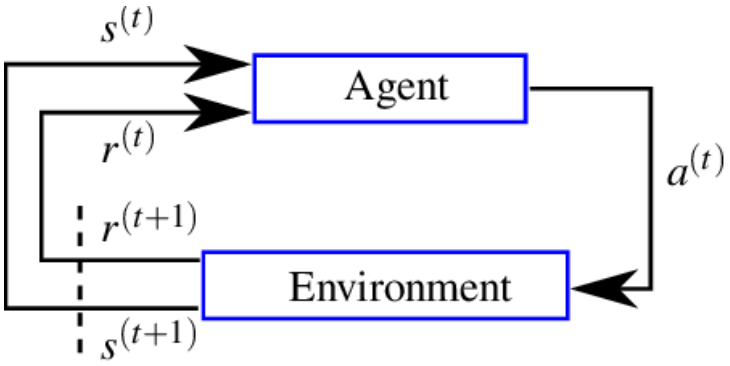


Figura 1.2: Modul de funcționare al învățării cu întărire [2]

imperfectă reprezentând dezavantajele acestor aparate comparativ cu cele de nivel medical. Acest lucru însă nu a împiedicat apariția a tot mai multor studii care folosesc aceste aparate [4].

EEG[5]

EEG-CNN[6]

TODO

1.4 Structurare pe capitole

Lucrarea este structurată după cum urmează. *Capitolul 2* prezintă bazele teoretice ale rețelelor neuronale de tip *feedforward*, iar mai apoi sunt prezentate rețelele neuronale convoluționale, care folosesc la baza lor concepte provenite de la rețelele neuronale simple. Tot în Capitolul 2 sunt prezentate și concepte fundamentale referitoare la undele cerebrale, ce sunt acestea, cum funcționează și felul în care pot fi detectate și înregistrate. În *Capitolul 3* este prezentată implementarea soluției propuse, împreună cu detaliile tehnice aferente. Explicarea tehnicilor de extragere și prelucrare a datelor, detalii privind arhitectura rețelei folosite și rezultatele produse de aceasta se regăsesc la finalul acestui capitol. Finalul lucrarii este alcătuit din *Capitolul 4*, care aduce concluzii legate de tema acestei lucrări.

Capitolul 2

Studiul teoretic

Acest capitol are rolul de a prezenta noțiuni și aspecte fundamentale legate atât de rețelele convoluționale cât și despre undele cerebrale. În prezentarea rețelelor convoluționale se va porni de la perceptronul simplu, care constituie fundamentul acestor algoritmi, apoi va fi prezentată o rețea alcătuită din perceptroni, iar la final va fi prezentată rețeaua convoluțională, construită cu ajutorul noțiunilor și elementelor provenite de la rețeaua neuronală. Spre finalul acestui capitol sunt prezentate conceptele fundamentale legate de undele cerebrale, ce sunt acestea, cum se formează și felul în care pot fi detectate și înregistrate.

În pregătirea și redactarea subcapitolelor §2.1 și §2.2 au fost folosite, în special, resursele teoretice din [7] și [8].

2.1 Rețele neuronale

Rețelele neuronale artificiale reprezintă un sistem de calcul inspirat după modelul rețelelor neuronale biologice, atât ca și structură, cât și ca mod de procesare a informației. Neuronul artificial reprezintă unitatea elementară a unei rețele neuronale artificiale. Precum neuronul biologic, care conține dendrite și axoni, neuronul artificial imită această structură prin noduri de intrare si noduri de ieșire.

2.1.1 Perceptronul

Perceptronul reprezintă un concept fundamental al rețelelor neuronale, fiind introdus de către *Frank Rosenblatt (1958)* [9] pe baza cercetărilor anterioare ale lui *Warren McCulloch* și *Walter Pitts* [10]. Pe baza perceptronului a fost dezvoltat ulterior modelul neuronului artificial.

Perceptronul primește o serie de valori la intrare, $x_1, x_2, x_3 \ldots$, și produce o singură ieșire. Pentru calculul ieșirii, sunt folosite așa numitele *ponderi (weights)*, notate cu $w_1, w_2, w_3, \ldots$, numere reale reprezentând importanța fiecărei intrări în determinarea ieșirii.

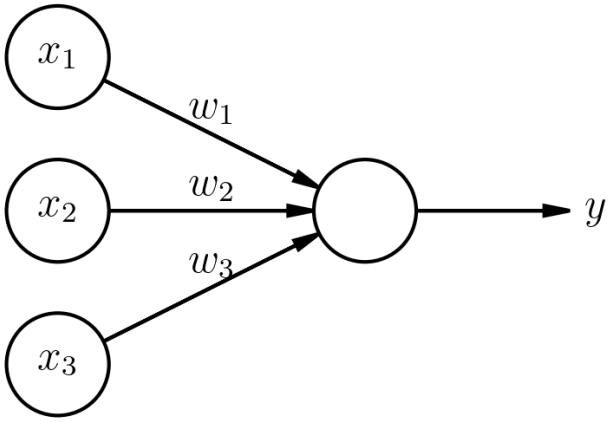


Figura 2.1: Reprezentarea grafică a unui perceptron

Astfel, ieșirea perceptronului, este determinată de suma ponderată $\sum_i x_i w_i$ comparată cu o valoare reală numită *prag*. Această comparație reprezintă o funcție pentru determinarea ieșirii y, denumită *funcție de activare*. Matematic, funcția de activare a perceptronului este reprezentată de o formă discretă a funcției *treaptă*

unitate, descrisă de ecuația (2.1).

$$y = \begin{cases} 0 & : \sum_i x_i w_i \leq prag \\ 1 & : \sum_i x_i w_i > prag \end{cases} \tag{2.1}$$

Modul prin care perceptronii sunt descriși poate fi simplificat, rescriind $\sum_i x_i w_i$ ca fiind produsul cartezian $x \cdot w$, unde x și w sunt vectori care conțin intrările x_i respectiv ponderile w_i. O a doua modificare posibilă este de a muta termenul $prag$ în partea stângă a inegalității, denumindu-l *bias/offset*, $b = -prag$. Astfel, ecuația (2.1) devine:

$$y = \begin{cases} 0 & : x \cdot w + b \leq 0 \\ 1 & : x \cdot w + b > 0 \end{cases} \tag{2.2}$$

Bias-ul poate fii considerat ca o intrare suplimentară de valoate $x_0 = 1$ și $w_0 = b$ care permite translatarea funcției de activare la stânga sau la dreapta.

2.1.2 Perceptronul multistrat

Prin conectarea mai multor perceptroni rezultă rețeaua numită „perceptronul multistrat” *(Multi-layer Perceptron - MLP)*. Aceasta este formată dintr-o succesiune de perceptroni complet conectați, așezați înntr-un stratz de intrare, unul sau mai multe straturi ascunse *(hidden layer)* și un strat de ieșire. Ieșirile unui strat reprezintă intrările pentru stratul următor. Rețeaua care conține mai multe straturi ascunse poartă denumirea de *„rețea neuronală adâncă” ("deep neural network")*, iar în cazul în care aceasta conține un singur strat ascuns poartă denumirea de *„rețea neuronală superficială” ("shallow neural network")*. Figura 2.2 prezintă o astfel de rețea.

Rețeaua prezentată mai sus este de tip *propagare înainte (feed-forward)*, adică, informația în rețea circulă într-o singură direcție, de la stânga la dreapta. Rezultatul procesării informațiilor de către

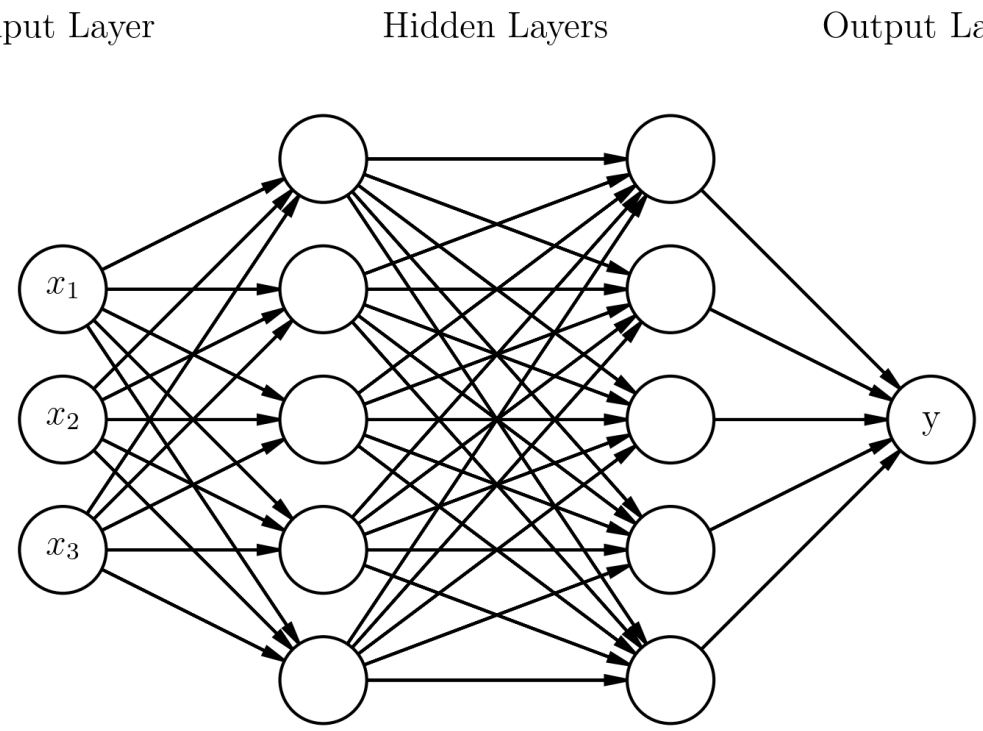


Figura 2.2: Rețea de tipul perceptron multistrat

primul strat de neuroni va reprezenta intrarea pentru stratul al doilea, astfel ieșirile acestuia având o semnificație mai abstractă și complexă comparativ cu primul strat. Cu fiecare strat ascuns adăugat rețelei, nivelul de abstractizare al informației va crește, astfel deciziile luate de rețea devenind tot mai sofisticate.

Limitarea acestui tip de rețea poate fii observată încercând să aplicăm schimbări mici ponderilor w conexiunilor (sau a $bias$-ului) unui strat pentru a obține o schimbare mică a ieșirii rețelei. Analitic, acest lucru se rezumă la următoarea ecuație:

$$\Delta y \approx \sum_i \frac{\partial y}{\partial w_i} \Delta w_i + \frac{\partial y}{\partial b} \Delta b \tag{2.3}$$

În realitate însă acest lucru nu se întâmplă întotdeauna. Aceste mici

modificări pot determina schimbarea complet a stării[1], spre exemplu de la 1 la 0. Acest comportament poate declanșa o schimbare foarte complicată și greu de controlat în întreaga rețea.

Limitarea dată de capacitatea perceptronului de clasificare binară, poate fii rezolvată însă folosind un alt tip de funcție de activare.

2.1.3 Neuronul

Neuronul artificial este foarte asemănător cu perceptronul prezentat anterior. Acesta este format dintr-un vector de intrări x, un vector al ponderilor w, un bias b și o ieșire y. Valorile vectorului de intrare x nu sunt însă limitate la valorile 1 sau 0, neuronul sigmoid fiind capabil să proceseze valori reale. Precum x, ieșirea y a neuronului poate lua valori reale. Figura 2.3 prezintă un astfel de neuron.

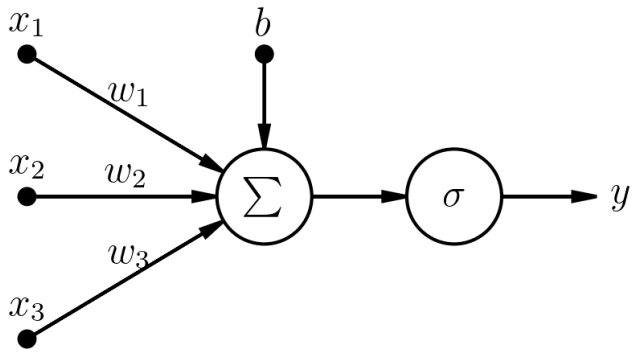


Figura 2.3: Reprezentarea grafică a unui neuron sigmoid

Neuronul care folosește *funcția sigmoid* ca și funcție de activare, dată de ecuația (2.4), se numește neuron sigmoid și este cel mai simplu tip de neuron artificial datorită proprietăților funcției logistice la derivare[2].

[1]Acest lucru poate fi observat în graficul funcției de activare al perceptronului din Figura 2.4

[2]Derivabilitatea funcțiilor de activare este o cerință în cadrul folosirii algoritmului de antrenare prin back-propagation prezentat în §2.1.4

$$\sigma(z) = \frac{1}{1+e^{-z}} \quad \text{, unde } z = w \cdot x + b \tag{2.4}$$

Analizând graficele din Figura 2.4 putem observa faptul că funcția sigmoid este de fapt o versiune netezită a funcției treaptă unitate. Acest lucru ne asigură că schimbările mici efectuate atât în vectorul ponderilor w cât și în b vor fi reflectate în ieșire și că nu vom avea salturi bruște de la 0 la 1 la ieșirea neuronului. Totuși, modelul perceptronului poate fii simulat folosind funcția sigmoid. Atunci când $z \to \infty$, $\sigma(z) \approx 1$, iar când $z \to -\infty$, $\sigma(z) \approx 0$. Folosind astfel de funcții de activare ne ajută să găsim ponderile potrivite mult mai ușor și putem afla felul în care modificările acestora afectează ieșirea [7].

În general, se folosesc funcții derivabile pe întreg domeniul de definiție, care nu au treceri bruște de la un capăt la altul, pentru a facilita antrenarea rețelei. În practică se mai folosesc funcții precum *tangenta hiperbolică*:

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{2.5}$$

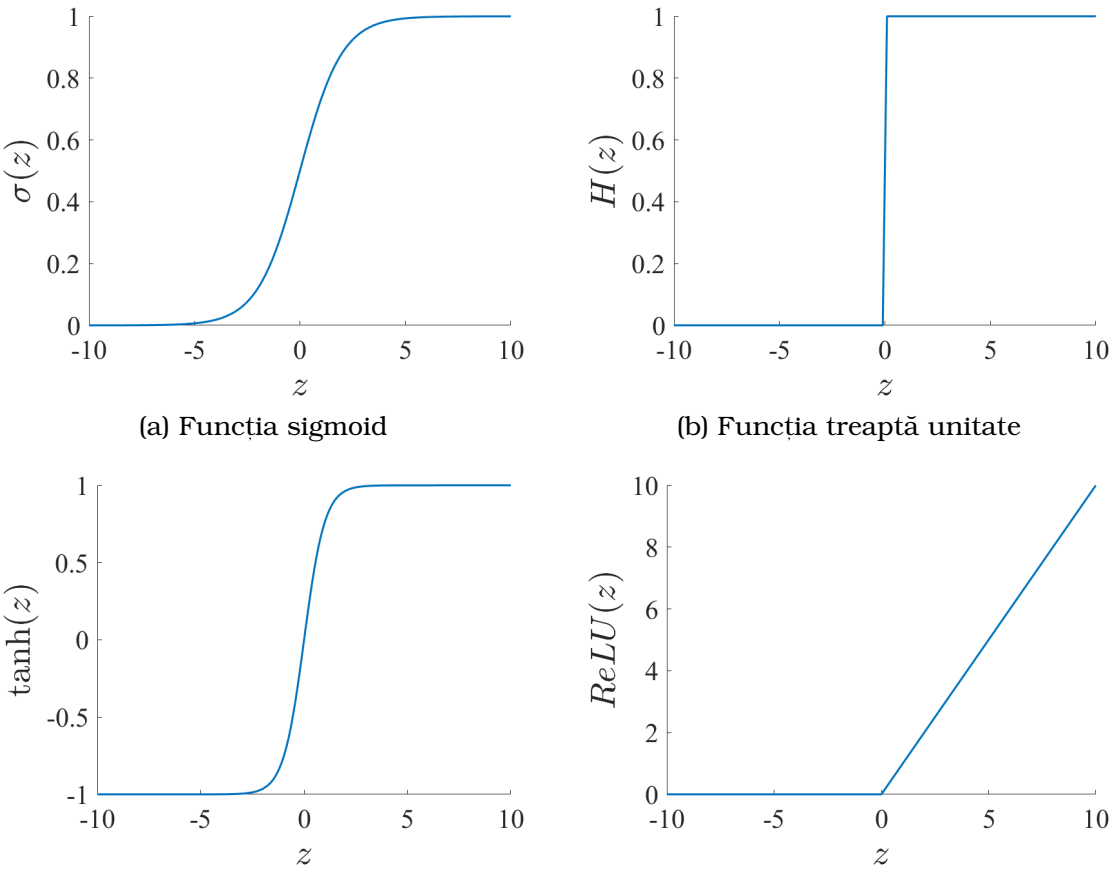
sau *funcția unitate liniară rectificată (ReLU)*:

$$R(z) = \max(0, z) \tag{2.6}$$

Figura 2.4 prezintă graficele diferitor funcții de activare. Este necesar de menționat faptul că aceste funcții se folosesc în special pentru neuronii aflați în straturile ascunse ale rețelei, stratul de ieșire folosind de obicei funcții logistice pentru clasificări binare și funcția *softmax* pentru clasificări multi-clasă. Rolul acesteia este de a furniza un vector de probabilități cu dimensiunea egală cu numărul claselor, fiecare poziție a vectorului reprezentând o probabilitate pentru clasa aferentă.

2.1.4 Antrenarea rețelelor

Algoritmul prin care se realizează antrenarea rețelelor de tipul feedforward, poartă denumirea de *algoritm de propagare înapoi*



(a) Funcția sigmoid

(b) Funcția treaptă unitate

(c) Funcția tangenta hiperbolică

(d) Funcția unitate liniară rectificată

Figura 2.4: Funcții de activare

(back-propagation). Acest algoritm a fost făcut faimos de către David Rumelhart, Geoffrey Hinton, și Ronald Williams în 1986 [11]. Algoritmul constă în modificarea repetată a ponderilor conexiunilor din rețea în încercarea de a minimiza o funcție care reprezintă eroarea dintre rezultatul așteptat și cel obținut. Funcțiile folosite cu scopul de a fii minimizate se numesc *funcții obiectiv* sau *funcții de cost* (§2.1.5) [7].

Ecuațiile algoritmului back-propagation

Algoritmul clasic de back-propagation a fost inițial conceput să rezolve probleme de regresie folosind funcții de activare sigmoide. Totuși, acesta poate fii aplicat și problemelor de clasificare cu sau fară astfel de funcții de activare. Pentru a putea face o descriere matematică asupra modului de funcționare al algoritmului, este necesară folosirea următoarelor notații [7]:

1. C pentru a reprezenta o funcție de cost
2. w^l_{jk} pentru specificarea ponderii conexiunii de la neuronul k în stratul $l-1$ către neuronul j în stratul l.
3. b^l_j pentru bias-ul neuronului j din stratul l
4. a^l_j pentru activarea neuronului j din stratul l

a^l_j depinde de activarea neuronului din stratul $l-1$ conform ecuației

$$a^l_j = \sigma(\sum_k w^l_{jk} a^{l-1}_k + b^l_j) \tag{2.7}$$

unde k reprezintă toți neuronii din stratul $l-1$. Folosind aceste notații pot fii descrise cele patru ecuații fundamentale ale acestui algoritm:

1. Ecuația pentru determinarea erorii din stratul de ieșire L

$$\delta^L_j = \frac{\partial C}{\partial a^L_j}\sigma'(z^L_j) \tag{2.8}$$

2. Ecuație pentru determinarea erorii din stratul l în raport cu eroarea din stratul $l+1$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \tag{2.9}$$

, unde $\odot$ reprezintă înmulțirea element cu element a matricelor, cunoscută și sub denumirea de *produs Hadamard*

3. Ecuație pentru determinarea ratei de schimbare a funcției de cost în funcție de orice bias din rețea

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \tag{2.10}$$

4. Ecuație pentru determinarea ratei de schimbare a funcției de cost în funcție de orice pondere din rețea

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1}\delta_j^l \tag{2.11}$$

Aplicarea algoritmului back-propagation

Inițial toți parametrii rețelei (ponderile w și bias-ul b) vor fi inițializați aleator[3]. Urmează apoi etapa de propagarea înainte a datelor din setul de date de antrenare și a comparării rezultatului prezis în comparație cu cel real. Această eroare se calculează conform ecuației (2.8), iar mai apoi va fii propagată înapoi în rețea, strat cu strat, începând cu stratul $l = L - 1$, conform ecuației (2.9). Parametrii vor fi ajustați în funcție de gradientul funcției de cost conform ecuațiilor (2.10) și (2.11). Gradientul funcției de cost se calculează folosind un algoritm de optimizare, cel mai folosit algoritm împreună cu back-propagation este *metoda gradientului (gradient descent)*. Algoritmul se repetă pentru un numar stabilit de pași (aleși experimentali) sau până când eroarea δ^L scade sub o valoare impusă. Odată cu finalizarea algoritmului, rețeaua va fii pregătită să proceseze date pe care nu le-a întâlnit în setul folosit la antrenare și să facă predicții asupra acestora [7],[8].

2.1.5 Funcții de cost

O funcție de cost este o funcție de forma [7]

$$C(w, b, S^T, E^T) \tag{2.12}$$

[3]Nu este neapărat ca parametrii rețelei să fie inițializați aleator, existând mai multe metode de inițializare [12]

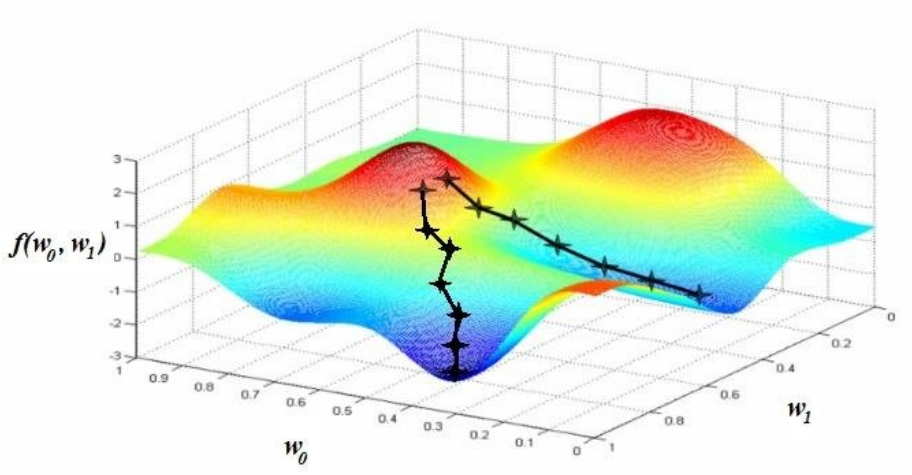


Figura 2.5: Ilustrarea algoritmului de optimizare gradient descent folosit în back-propagation [8]

unde w și b reprezintă parametrii rețelei S^T intrarea unui eșantion de date de antrenare și E^T ieșirea dorită din eșantionul respectiv.

Rezultatul dat de asemenea funcții este un număr care reprezintă performanța rețelei de a face preziceri pe baza modificărilor aduse parametrilor acesteia. Se urmărește prin diferite *tehnici de optimizare* minimizarea acestei funcții, valoarea returnată purtând denumirea de *eroare/cost/loss*. În anumite situații însă, spre exemplu în cazul învățării cu întărire, scopul este de a maximiza această funcție, rezultatul fiind denumit *recompensă*.

Atât funcțiile de cost, cât și tehnicile de optimizare ale acestora trebuie alese în funcție de problema în cauză, neexistând o soluție universal valabilă. Empiric, funcțiile de cost se pot împărți în două categorii:

- **pentru probleme de regresie**: eroarea medie absolută *(L1 loss)*, eroarea medie pătratică *(L2 loss)*, Huber Loss
- **pentru problemele de clasificare**: funcția de cost logaritmică *(cross-entropy)*, categorical cross-entropy, divergența Kullback–Leibler

2.2 Rețele convoluționale

Rețelele convoluționale sunt un tip specific de rețele neuronale adândci, inspirate din modul de reconoaștere al tiparelor în cortexul vizual uman. Arhitectura acestor rețele s-a dovedit a fi foarte eficientă atât în clasificarea imaginilor cât și în timpul necesar antrenării acesteia.

Una din primele rețele convoluționale de succes a fost *LeNet5 (1989)* [13], după repetate iterații începând cu anul 1988. Această rețea a fost folosită în principal pentru detectarea cifrelor și codurilor poștale.

În ziua de astăzi rețelele neuronale convoluționale sunt folosite în diverse aplicații, predominant fiind cele pe baza analizei imaginilor, precum detectarea și recunoașterea obiectelor din seturi de date cu mii de categorii, de exemplu *VGGNet* [14] pe setul de date *ImageNet*, determinarea profunzimii scenelor din imaginile video 2D [15], practic imitând principiul *LIDAR*[4], diferențierea obiectelor din scenă *(semantic segmentation)*. Rețelele convoluționale și-au găsit folosința și în aplicații care nu implică în mod direct imagini, cum ar fii procesarea limbajului natural sau clasificarea evenimentelor audio [16].

Arhitectura

Structura unei rețele convoluționale este formată din două părți. Partea convoluțională, compusă din stratul de convoluție cu funcția de activare aplicată ieșirii acestuia și stratul de reducere a dimensionalității. A doua parte a structurii este partea de clasificare alcătuită din straturi de neuroni complet conectați asemenea rețelei MLP prezentate în §2.1.2. Figura 2.6 prezintă o astfel de arhitectură.

[4]Light Detection And Ranging - reprezintă un sistem similar de funcționare cu radar-ul, care utilizează laser-ul pentru a afla distanța până la țintă

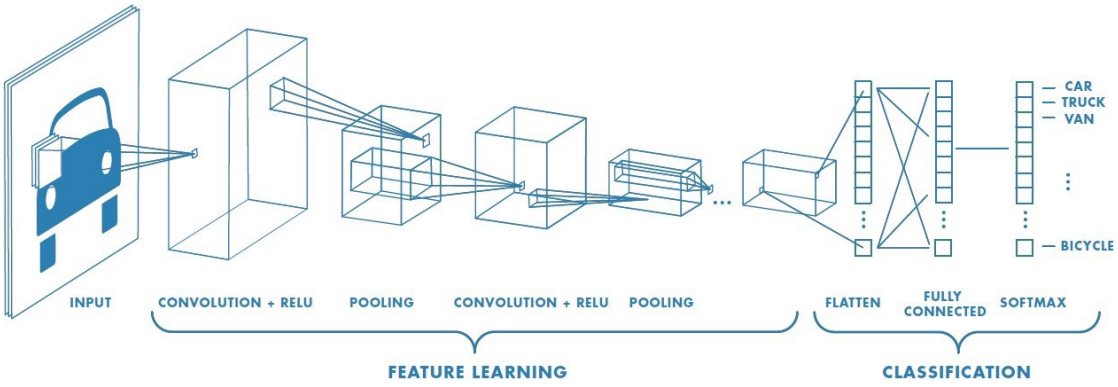


Figura 2.6: Arhitectura generală a unei rețele convoluționale [17]

Straturile de convoluție

Rețelele convoluționale sunt denumite după operația centrală acestora, operația de convoluție. Matematic, operația de convoluție reprezintă rezultatul combinării a două funcții pentru a forma o a treia. Această operație poate fi descrisă în varianta discretă, în spațiul 2D sub următoarea formă:

$$(f * g)[i, j] = \sum\sum f[m, n]g[i - m, j - n] \tag{2.13}$$

În straturile convoluționale, spre deosebire de cele complet conectate, neuronii ascunși din stratul l sunt conectați la un grup localizat de neuroni din stratul $l - 1$ denumit câmp receptor *(receptive field)*. Câmpul receptor reprezintă zona în care este aplicat filtrul/nucleul de convoluție asupra matricei de intrare. Crearea stratului l de neuroni ascunși se realizează aplicând un filtru/nucleu de convoluție de dimensiune $H \times W$, care va fi mutat peste întreaga imagine de intrare pornind din colțul stânga sus și deplasându-se pe orizontală și pe verticală cu un anumit „pas” *(stride)* notat s. În exemplul din Figura 2.7, pasul folosit este de $s = 1$. Se poate observa faptul că odată cu aplicarea operației de convoluție rezultă o reducere a dimensionalității a matricei rezultate. Acest lucru poate fi remediat brodând cu zero-uri *(zero padding)* marginile matricei de intrare.

Asupra rezultatului de convoluție dintre filtru și matricea de intrare

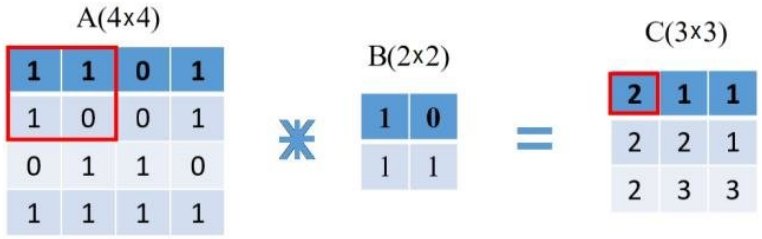


Figura 2.7: Exemplificarea convoluției a două matrici [8]

se aplică o funcție de activare, care este adesea de tip ReLU, amintită în §2.1.3. Matricea rezultată în urma operației de convoluție și aplicarea funcției de activare poartă denumirea de hartă de caracteristici *(feature map)* sau hartă de activare *(activation map)*.

Un strat de convoluție este alcătuit dintr-o multitudine de hărți de caracteristici create folosind diferite tipuri de filtre. Fiecare strat adăugat rețelei, va crea hărți care vor captura caracteristici tot mai complexe din imaginea inițială.

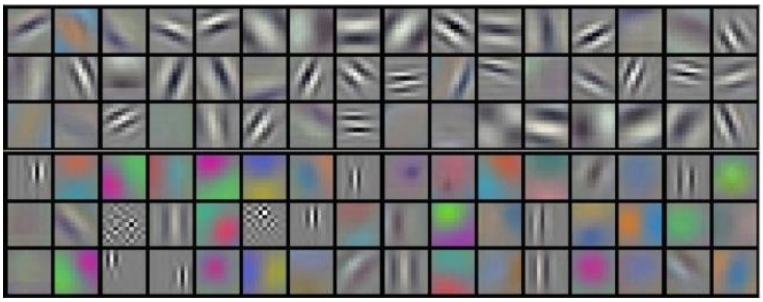


Figura 2.8: Filtrele învățate în primul strat de convoluție de către rețeaua AlexNet [18]

Reducerea dimensionalității

Deseori, după stratul de convoluție este aplicat un strat de reducere a dimensionalității *(pooling)*, cunoscut și sub numele de subeșantionare. Acesta are rolul de a micșora dimensiunile spațiale ale hărților de caracteristici și de a reduce numărul de parametrii

antrenabili, dar în același timp de a păstra informațiile esențiale. Tehnicile folosite uzual pentru operația de *pooling* sunt *Average Pooling, Max Pooling, Sum Poolling*. Aceste tehnici au la bază folosirea unei ferestre de dimensiuni relativ mici (de obicei 2×2), din care vor fi extrase valorile maxime în cazul Max Pooling, valorile medii pentru Average Pooling și suma tuturor valorilor pentru Sum Pooling. Empiric s-a observat faptul ca de cele mai multe ori tehnica Max Pooling are cele mai bune rezultate. Figura 2.9 exemplifică aplicarea acestei tehnici.

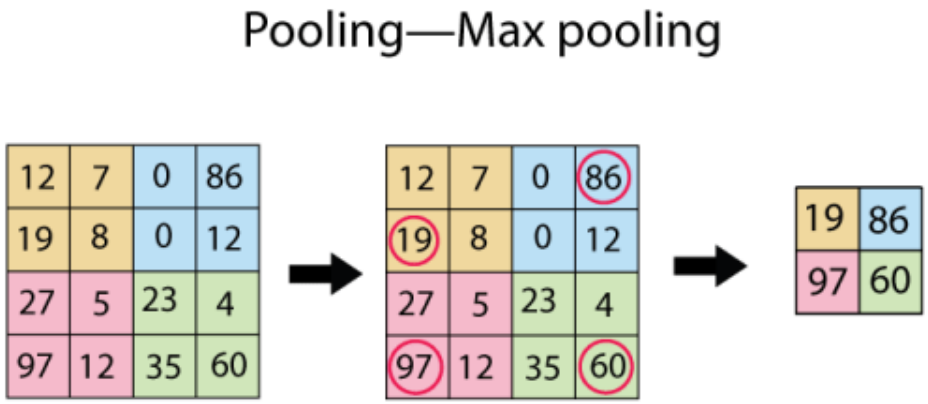


Figura 2.9: Exemplificarea aplicării tehnicii Max Pooling [19]

Operațiile de pooling descrise anterior pot fi înlocuite folosind în stratul convoluțional un pas de deplasare al filtrelor mai mare ca 1 sau folosind un strat convoluțional special cu dimensiunea filtrului 1×1 și cu pasul $s > 1$. Această operație poate fi benefică din punct de vedere computațional, deoarece atât operația de convoluție cât și operația de subeșantionare sunt aplicate în același timp însă poate crește dificultatea de antrenare a rețelei datorită creșterii numărului de parametrii antrenabili introduși în rețea comparativ cu tehnicile de pooling care nu conțin nici un parametru, sunt operații fixe.

Subeșantionarea folosind operațiile de pooling clasice poate cauza probleme rețelelor convoluționale prin pierderea informației poziționale

ale diferitelor obiecte prezente în imagine. Scopul inițial al introducerii acestui strat a fost de a reduce numărul de parametrii ai rețelei, deci reducerea timpului de antrenare a rețelei. Avansuri în dispozitive hardware tot mai puternice a redus nevoia unei astfel de tehnici, astăzi multe arhitecturi înlocuind această metodă de reducere a dimensionalității cu straturi speciale de convoluție precum *Separable Convolutions* și *Dilated Convolutions* [20].

Stratul complet conectat

Pentru a putea face legătura între straturile convoluționale și clasificatorul final, ieșirea din straturile convoluționale trebuie transformată din dimensiunea $H \times W \times D$, într-un vector coloană $H \times 1$. Această operație poartă denumirea de *aplatizare (flatten)* și este inclusă intr-un strat separat denumit *strat de aplatizare (flatten layer)*.

Odată transformată ieșirea sub formă de vector, aceasta este folosită ca intrare, de obicei, pentru rețele neuronale multistrat. Această rețea folosește pe stratul de ieșire funcția de activare *softmax*.

Se mai pot folosi în schimbul rețelelor neuronale și alte tehnici de clasificare, precum *mașini cu vectori suport (SVM)*.

2.3 Undele cerebrale

Creierul uman conține miliarde de celule specifice sistemului nervos, înalt specializate, numite *neuroni*, cu capacitatea de a genera, transmite și recepționa semnale electro-chimice. Un neuron este alcătuit din *corp celular*, *dendrite* și *axon* (Fig. 2.10). Legătura dintre mai mulți neuroni se numește *sinapsă* și este realizată între axonul neuronului presinaptic și dendritele sau corpul celular neuronului postsinaptic [21]. În creierul uman, un neuron formează mii de conexiuni sinaptice. Fiecare neuron deține o diferență de potențial în jurul membranei sale numită *potențial local*. În momentul în care

tensiunea electrică crește brusc, neuronul generează un puls electro-chimic denumit *potențial de acțiune*, care străbate rapid axonul neuronului activând conexiunile sinaptice ale acestuia [22].

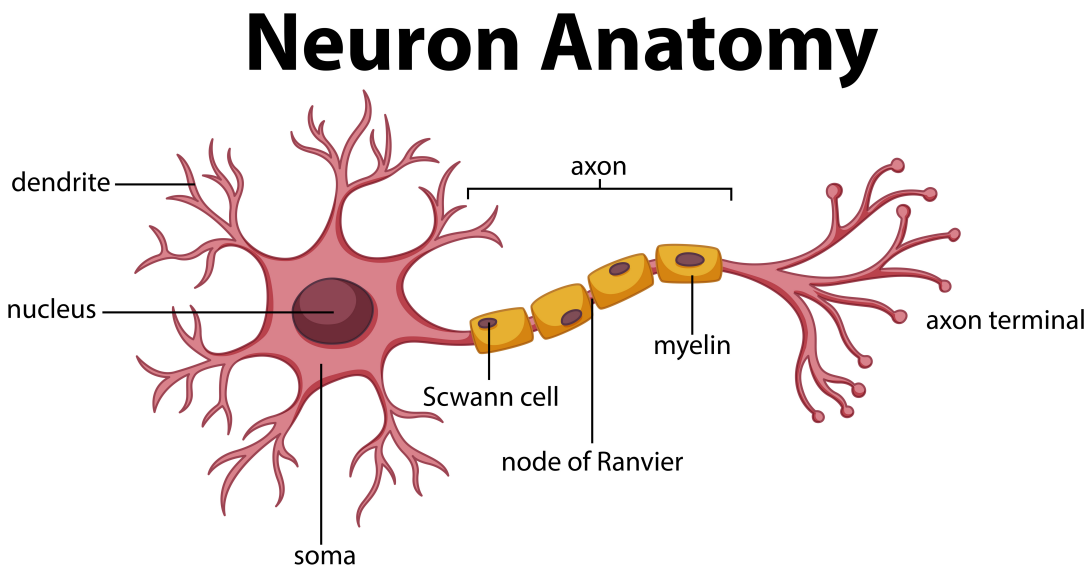


Figura 2.10: Anatomia unui neuron tipic [23]

Descărcările electrice repetate și sincronizate ale neuronilor rezultă în așa numitele *unde cerebrale*, cu benzile de frecvență cuprinse între 0.5 - 50 Hz. *Electroencefalografia (EEG)* este tehnica de detectare și înregistrate în timp a activității cerebrale. Aceasta poate fi folosită în două moduri, invaziv sau neinvaziv.

Electroencefalografia invazivă presupune așezarea electrozilor direct pe suprafața creierului. Folosirea acestei metode de înregistrare a undelor cerebrale rezultă în date achiziționate fără zgomot și precise referitor la zona creierului studiată. Dezavantajul este însă marcat de complexitatea metodei, necesitând intervenție chirurgicală cu riscuri mari.

Achiziția undelor cerebrale într-un mod neinvaziv constă în plasarea electrozilor pe scalp, astfel se elimină necesitatea unei operați chirurgicale. Prin folosirea acestei metode datele achiziționate vor fi alterate de zgomot produs de contracțiile mușchilor din zona capului,

datele având nevoie de preprocesare înainte de a putea fi folosite.

Există cinci tipuri de bază de unde cerebrale, fiecăreia fiindui atribuită o literă grecească, după cum urmează *delta, teta, alfa, beta, gama*. Undele cerebrale se modifică în funcție de activitatea desfășurată și de dispoziție. Însemnătatea acestora fiind corelată cu locația detectării în creier. Acestea pot fi descrise, conform [24] și [25] astfel:

- **Delta**: undele delta au cea mai scăzută bandă de frecvența dintre toate undele cerebrale, încadrându-se în intervalul 0.5-4 Hz. Se întâlnesc cel mai des în timpul unui somn adânc sau în timpul meditației profunde. Această stare mentală stimulează regenerarea și vindecarea corpului.

- **Teta**: banda de frecvență caracteristică undelor teta e între 5 și 8 Hz. Adesea, acestea sunt foarte ușor detectabile în momentele în care visăm. Starea mentală teta mai poate fi observată și în momentul unor activități foarte comune, în care procesul de realizare devine automat. Această stare dă naștere unui șir de gândire și idei liber în care creativitatea este sporită.

- **Alfa**: undele alfa, cu banda de frecvență între 9 și 14 Hz, reprezintă starea de relaxare conștientă. Plimbările prin parc, momentele de relaxare după îndeplinirea unei sarcini, reflecția sau meditația ușoară reprezintă momente în care undele alfa sunt proeminente.

- **Beta**: undele beta, având banda de frecvență între 15 și 40 Hz, sunt asociate cu activitatea mentală normală. Acestea sunt prezente în momente precum concentrarea asupra unei probleme, învățarea de noi concepte, stare de alertă sau luarea deciziilor. Concret, această stare este caracteristică gândirii active.

- **Gama**: undele gama au banda de frecvență cea mai înaltă, >40 Hz, fiind asociate cu procesarea simultană a informațiilor din

zone diferite ale creierului. Acestea au fost inițial încadrate ca fiind „zgomot cerebral" până când cercetătorii au observat o activitate accentuată a acestora în stări cognitive și concentrație intensă. Modul de generare al acestor unde cerebrale este încă necunoscut, frecvența acestora fiind peste capacitatea de descărcare a neuronilor.

Capitolul 3

Prezentarea aplicației

Scopul lucrării îl constituie folosirea paradigmei învățării automate supervizate, mai precis, utilizarea rețelelor neuronale convoluționale pentru clasificarea a trei stări mentale diferite, *neutru, relaxat* și *concentrat*. Datele aferente fiecărei stări provin de la un dispozitiv comercial, *Muse 2016*, capabil de a înregistra activitatea cerebrală folosind tehnica de imagistică *EEG (Electroencefalografia)*, neinvazivă. Aceste înregistrări reprezintă activitatea creierului din jurul electrozilor dispusă în timp. Deoarece dispozitivul folosit folosește o tehnică neinvazivă cu electrozi uscați datele finale ale înregistrării conțin zgomot produs de mișcările persoanei, de contracțiile mușchilor sau chiar de clipit. Din aceste motive datele trebuie prelucrate. Este de menționat faptul că nu există o anumită formulă de prelucrare pentru ca datele sa fie perfecte pentru etapa de clasificare, metodele folosite în această etapă fiind stabilite adesea empiric. După prelucrarea datelor, urmează un pas de extragere a anumitor atribute/caracteristici statistice și spectrale pe baza cărora vor fi construite imaginile alb-negru. Fiecare imagine va avea o etichetă aferentă clasei din care face parte, neutru, relaxat sau concentrat.

După etapa de prelucrare și etichetare a datelor, urmează pregătirea datelor pentru transformarea acestora sub forma unor imagini alb-negru. Această etapă include selectarea celor mai relevante

400 de atribute, dintr-un total de 414. Pentru a putea fi folosite ca și componente ale unei imagini alb-negru, este necesară normalizarea datelor în intervalul $[0, 1]$, 0 reprezentând negru, 1 reprezentând alb, orice altă valoare aparținând intervalului reprezentând o nuanță de gri. Imaginile sunt împărțite apoi în diferite seturi, fiecare cu scopul său specific; setul de antrenare, setul de validare și setul de testare. Antrenarea rețelei convoluționale se realizează folosind setul de antrenare și setul de validare pentru evaluarea performanțelor pe parcursul antrenării. În final, setul de testare este folosit pentru a determina performanța rețelei pe date complet noi, rezultând metricile finale, precum acuratețea sau valoarea funcției de cost *(loss)*. În cazul rețelei dezvoltate în această lucrare a fost atinsă o acuratețe de $\approx 92\%$ cu valoarea funcției de loss de ≈ 0.27.

În continuarea capitolului, etapele prezentate sumar anterior vor fi detaliate și explicate, urmând ca la finalul capitolului să fie prezentate rezultatele implementării.

3.1 Etape implementare

3.1.1 Achiziție date

Pentru achiziția datelor a fost folosită casca comercial valabilă Muse 2016. Interfațarea acesteia cu calculatorul a fost realizată folosind biblioteca *MuseLSL* [26], pentru limbajul de programare *Python*. Această bibliotecă oferă posibilitatea conectării căștii Muse 2016, prin Bluetooth, cu calculatorul pentru transmisia de date, salvarea acestora în fișiere sau afișarea sub forma unui grafic.

Casca Muse 2016 conține cinci electrozi uscați, unul fiind folosit ca punct de referință, iar ceilalți patru pentru a înregistra semnalele EEG. Electrozii sunt poziționați în locațiile *TP9, AF7, Fpz, AF8, TP10*, conform unei versiuni modificate a sistemului internațional 10-20 de poziționare al electrozilor EEG. Electrodul Fpz este folosit ca electrod de referință.

În Figura 3.1 este prezentat aranjamentul electrozilor.

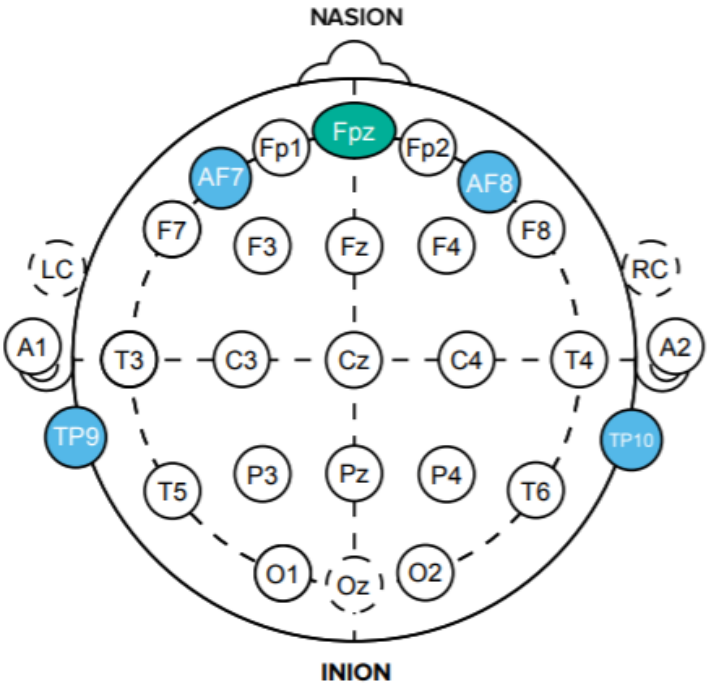


Figura 3.1: Versiunea modificată a sistemului internațional 10-20 împreună cu marcajul poziției electrozilor căștii Muse [27]

Pentru a diminua zgomotul înregistrat de senzori, pentru fiecare stare au fost alese activități care necesitau o mișcare minimă a persoanei examinate. Citirile datelor de la electrozii AF7 și AF8, fiind situați pe frunte, sunt distorsionate de mișcările generate de clipit. Astfel, pentru a păstra o uniformitate asupra celor trei stări, clipitul nu a fost nici încurajat nici descurajat. Chiar dacă acesta este considerat zgomot, rata de clipire este influențată de starea de concentrare a persoanei, acest lucru fiind util în final pentru algoritmul de clasificare. Totuși, persoanelor examinate le-a fost specificat să nu inchidă ochii pe durata unui test.

Pentru determinarea celor trei stări mentale, au fost stabilite trei activități, conform [5]. Pentru înregistrarea stării neutre, participanții

au fost îndrumați să păstreze o poziție confortabilă a corpului și o stare mentală echilibrată, nici prea relaxată dar nici prea activă. Înregistrarea stării mentale de relaxare a presupus ca subiecții să asculte o piesă muzicală cu o tonalitate și un tempo scăzut. Acestora le-a fost indicat faptul de a se relaxa complet, atât fizic cât și psihic. Pentru determinarea stării mentale active și concentrate participanții au fost instruiți să joace o variantă a jocului „alba-neagra", în care un obiect este ascuns sub unul din mai multe pahare, care mai apoi sunt amestecate între ele. Scopul jocului fiind de a identifica paharul care conține obiectul. Pe parcursul jocului, numărul paharelor și viteza cu care acestea se mișcă crește.

Activitățile au fost desfășurate începând cu starea neutră, urmată de starea de relaxare, iar la final starea de concentrare. Pentru fiecare activitate în parte au fost efectuate două înregistrări ale datelor EEG pe o durată de 65 de secunde. Primele 3 și ultimele 2 secunde fiind șterse ulterior, scopul acestora fiind de a crea o zonă de tranziție. Datele EEG au fost colectate de la 12 persoane, 5 de sex feminin, 7 de sex masculin. În total au fost înregistrate 26 de minute per stare, din care 24 de minute utilizabile. În Figura 3.2 pot fi observate semnalele provenite de la fiecare electrod în parte. Casca oferă posibilitatea de a adăuga un senzor în plus, semnalul acestuia fiind transmis prin canalul RightAUX. Semnalul provenit de la canalul în cauză a fost eliminat deoarece nu avea conectat un senzor, transmițând astfel doar zgomot.

3.1.2 Extragerea atributelor

Etapa de extragere a atributelor are rolul de a crea un set de date bazat pe semnalele EEG înregistrate, cu scopul de a fi cât mai informative pentru obiectul lucrării. Astfel, se elimină informația redundantă și nesemnificativă obiectivului, aducând avantaje precum scăderea timpului de învățare a rețelei și o mai bună generalizare a datelor de către aceasta.

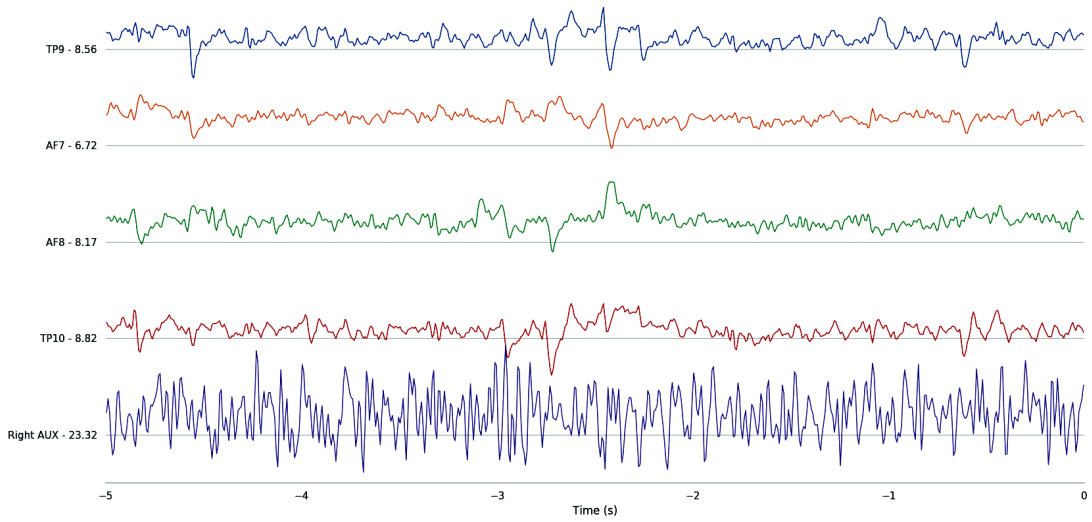


Figura 3.2: Exemplu semnale EEG Muse 2016, extrase folosind librăria MuseLSL

Prin set de date se înțelege o matrice de N linii și M coloane. Cele M coloane reprezintă atributele semnalelor, 414 la număr în această lucrare, iar cele N linii reprezintă valorile calculate pentru fiecare atribut în parte, 8568 în total.

În această lucrare, pe baza semnalelor EEG au fost determinate două seturi de date folosind atributele statistice ale semnalelor, respectiv atribute spectrale ale acestora. Prin atribute statistice se înțelege un set de caracteristici de natură statistică, precum media, deviația standard, varianța etc. Prin atribute spectrale se înțelege un set de caracteristici legate de spectrul semnalului EEG. Această maniera de abordare a informației conținute de semnalele EEG, precum și modul de obținere a acestor seturi de date se bazează pe cele prezentate în lucrările [6] și [5].

Pentru calculul atributelor, semnalul a fost împărțit în „ferestre”, notate w_i, conținând un set de N valori ale semnalului pentru durata de 1 secundă, transmise cu frecvența 256Hz. Ferestrele conțin o suprapunere de 0.5 secunde, astfel $w_1 = [0, 1]$, $w_2 = [0.5, 1.5]$, $w_3 = [1, 2]$, etc. Din fiecare fereastră rezultată au mai fost create, prin

împărțirea fiecărei ferestre w_i, jumătăți și sferturi de fereastră. Rezultă, astfel, w_{h1} și w_{h2}, reprezentând prima respectiv a doua jumătate a ferestrei, fiecare conținând $N/2$ valori, iar w_{q1}, w_{q2}, w_{q3}, w_{q4} reprezentând sferturile ferestrei, fiecare conținând $N/4$ valori.

Atribute statistice

Având aceste ferestre create, pentru fiecare fereastră w_i au fost extrase atribute statistice considerând fie întreaga fereastră, fie jumătățile, fie sferturile. Astfel, pentru fiecare semnal EEG, au fost extrase următoarele atribute:

- Considerând o întreagă fereastra, w:

  1. Media valorilor ferestrei

  $$\mu = \frac{1}{N}\sum_{n}^{N} x_n \tag{3.1}$$

  unde x reprezintă setul de valori dintr-o fereastră

  2. Deviația standard a valorilor

  $$\sigma = \sqrt{\frac{1}{N}\sum_{n}^{N}(x_n - \mu)^2} \tag{3.2}$$

  3. Varianța (dispersia) valorilor

  $$\sigma^2 = \frac{1}{N}\sum_{n}^{N}(x_n - \mu)^2 \tag{3.3}$$

  4. Momentul centrat de ordin 3 (asimetria valorilor)

  $$\mu_3 = \frac{1}{N}\sum_{n}^{N}(x_n - \mu)^3 \tag{3.4}$$

5. Momentul centrat de ordin 4 (aplatizarea valorilor)

$$\mu_4 = \frac{1}{N}\sum_{n}^{N}(x_n - \mu)^4 \tag{3.5}$$

6. Covarianța perechilor de semnale EEG (TP9, AF7, AF8, TP10)

$$cov(X, Y) = \frac{1}{N}\sum_{n}^{N}(x_n - \mu_x)(y_n - \mu_y) \tag{3.6}$$

   unde x și y reprezintă setul de valori dintr-o fereastră a unui semnal EEG, iar μ_x și μ_y reprezintă media acestora calculată conform (3.1)

7. Valoarea maximă și valoarea minimă a ferestrei

- Considerând jumătățile de fereastră, w_{h1} și w_{h2}:

  1. Diferența dintre media valorilor jumătăților
  2. Diferența dintre deviația standard a valorilor jumătăților
  3. Diferența dintre maximul valorilor jumătăților
  4. Diferența dintre minimul valorilor jumătăților

- Considerând sferturile de fereastră, w_{q1}, w_{q2}, w_{q3}, w_{q4}:

  1. Media valorilor sferturilor de fereastră
  2. Diferența mediilor perechilor de ferestre
  3. Valoarea maximă și valoarea minimă a ferestrelor
  4. Diferența maximelor perechilor de ferestre
  5. Diferența minimelor perechilor de ferestre

La final, combinând atributele pentru fiecare semnal în parte, rezultă un total de 174 de atribute statistice.

Atribute spectrale

Comparativ cu atributele statistice, generarea atributelor spectrale a fost făcută doar pe ferestre întregi, w_i. În cadrul extragerii atributelor spectrale a fost efectuată o prelucrare a semnalului înainte de a calcula atributele. A fost necesară o procesare inițială a acestora datorită zgomotului introdus semnalului util prin înregistrare. Inițial, a fost realizată o centrare a întregului semnalului în jurul valorii 0, iar mai apoi a fost aplicat un filtru trece-jos de tip Chebyshev I (Fig. 3.3), cu frecvența de tăiere de 50Hz. A fost folosit acest filtru datorită proprietăților sale de a oferi o bandă de tranziție mai îngustă dar cu introducerea unui riplu acceptabil în banda de trecere. Riplul maxim admis folosit în implementarea filtrului a fost de 0.5%. Prin această filtrare se păstrează doar banda de frecvență relevantă undelor cerebrale, anume $1 - 50\text{Hz}$.

După aplicarea filtrului asupra întregului semnal EEG, a fost calculată Transformata Fourier Discretă *(DFT)*, folosind algoritmul *Fast Fourier Transform (FFT)*. Apoi, au fost extrase amplitudinile fiecărei componente spectrale ale semnalului filtrat, rezultând 50 de atribute per semnal. Ultimele atribute extrase au fost amplitudinile celor mai puternice 10 frecvențe din componența semnalului. Acestea au fost ordonate crescător în funcție de frecvența semnalului. În total, au rezultat 240 de atribute.

Combinând cele 174 de atribute statistice cu aceste 240 de atribute spectrale, rezultă un total de 414 atribute. La setul de date format de acestea, se adaugă și o etichetă reprezentând clasa fiecărei înregistrări (0 - neutru, 1 - concentrat, 2 - relaxat). Prin procesarea celor 24 de minute pentru fiecare stare, rezultă un set de date final cuprins din 8568 de linii, reprezentând ferestrele procesate și 415 coloane reprezentând atributele și clasa fiecărei ferestre. Ultimul pas a fost de a amesteca intrările setului de date creat pentru a-l omogeniza.

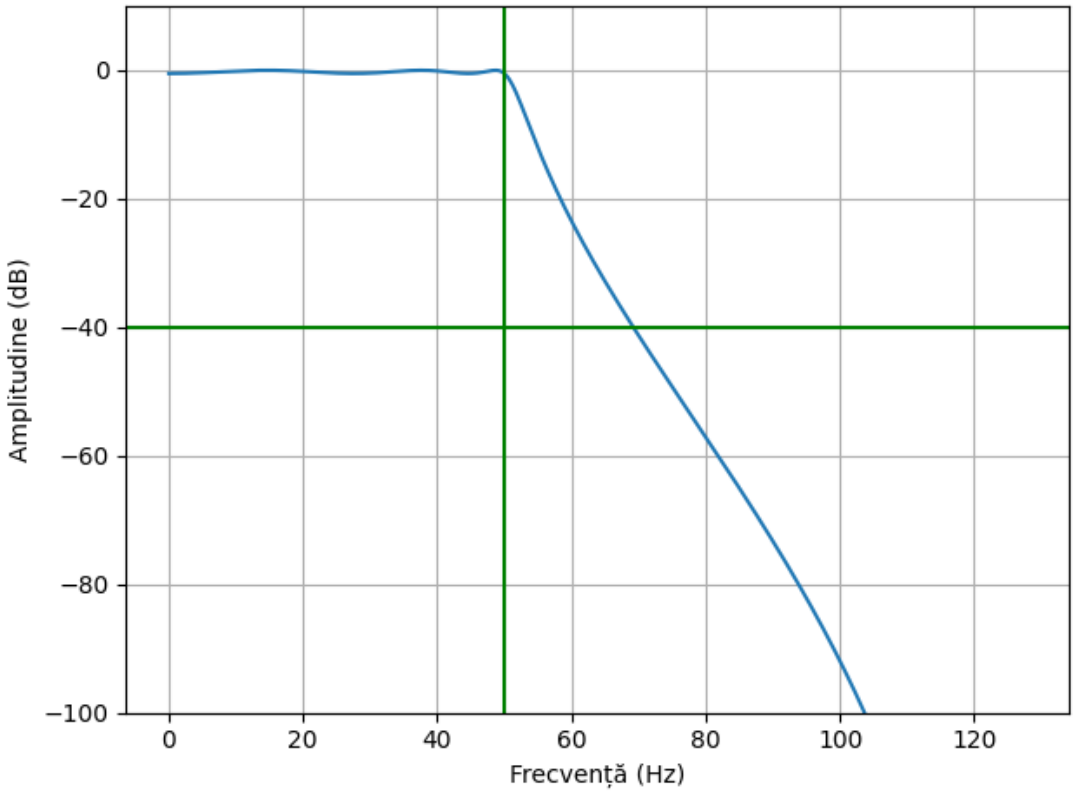


Figura 3.3: Răspunsul în frecvența a filtrului trece-jos de tip Chebyshev I de ordin 6

Procesarea setului de date

După crearea setului de date, acesta trece printr-o serie de modificări înainte de a fi folosit ca și intrare pentru rețeaua convoluțională. Pentru a putea trimite datele rețelei, este necesar ca acestea să fie sub formă matriceală. Cea mai mare matrice pe care o putem forma este o matrice de dimensiunea 20×20, folosind doar 400 din cele 414 atribute ale acestui set de date. Această cerință impune folosirea unui algoritm pentru selecția celor mai semnificative atribute.

Selecția atributelor a fost făcută folosind *testul chi-pătrat (χ^2) al asocierii* [28], care măsoară asocierea a două variabile cu scopul de a determina o relație între acestea. Pentru comparație, a mai fost folosită

metoda *Informației Mutuale* [29]. Aceasta măsoară dependența dintre două variabile, determinând cantitatea de informație furnizată de o variabilă cu privire la alta.

Pe setul de date rezultat după aplicarea algoritmului de selecție este aplicată o scalare a fiecărei înregistrare, prin procedura de *scalare min-max* care comprimă valorile în intervalul $[0, 1]$. Imaginile rezultate se pot observa în Figura 3.4. După aceste procesări, datele au fost transformate din dimensiunea 1×400 într-o matrice pătratică de dimensiunea $20 \times 20 \times 1$. Ultima dimensiune reprezentând numărul de canale al imaginii. Imaginea fiind alb-negru conține un singur canal. Ultima etapă înainte de a trimite datele rețelei este de a împărți setul de date în trei seturi distincte, pentru antrenare, validare și testare. Din totalul de date valabile, $\approx 72\%$ sunt folosite ca și date de antrenare, $\approx 18\%$ ca și date de validare, iar restul de $\approx 10\%$ pentru testarea finală.

3.1.3 Arhitectura rețelei convoluționale

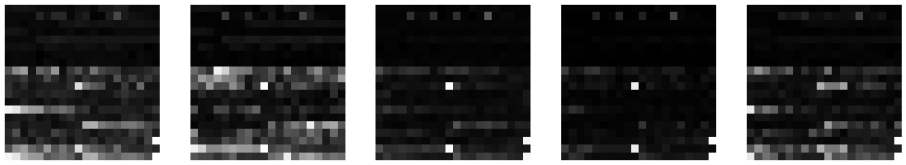
Arhitectura rețelei este compusă dintr-un total de 15 straturi secvențiale, 6 straturi de convoluție, 3 straturi de normalizare, 3 straturi de *dropout*, 1 strat de aplatizare și 2 straturi complet conectate, având următoarele funcții:

- **Conv2D:** Reprezintă stratul de convoluție, cu rolul de a învăța harta de caracteristici. Dimensiunea tuturor filtrelor folosite în această rețea este de 2×2. Pentru reducerea dimensionalității a fost folosit, în stratul 3 și stratul 5 de convoluție, pasului (stride) cu valoarea 2. Tuturor straturilor le-a fost aplicată funcția de activare ReLU (2.6)

- **BatchNormalization** [30]**:** Acest strat are rolul de a aplica o metodă de normalizare a datelor de intrare, reducând media acestora la ≈ 0, iar varianța la ≈ 1. Printre efectele acestei operații se numără accelerarea învățării rețelei neuronale și

(a) Imagini reprezentând starea neutră

(b) Imagini reprezentând starea concentrată



(c) Imagini reprezentând starea relaxată

Figura 3.4: Reprezentarea intrărilor din setul de date sub forma unor imagini alb-negru

prevenirea efectului de *overfitting*, care reprezintă o învățare prea bună a datelor de antrenare de către rețea, eșuând atunci când vine vorba de o clasificare a unor date noi

- **Dropout** [31]**:** Stratul dropout elimină conexiunile dintre straturi, într-o anumită proporție, pentru a „forța" rețeaua de a se adapta și de a generaliza datele de intrare. Prin acest procedeu se reduce fenomenul de *overfitting*
- **Flatten:** Acest strat are rolul de a „aplatiza" matricea provenită de la ultimul strat convoluțional, transformând-o într-un vector coloană de dimensiunea $(M * N * C) \times 1$, unde M este numărul de linii, N, numărul de coloane, iar C numărul de filtre
- **Dense:** Reprezintă straturile complet conectate, specifice unei rețele neuronale, cu rolul în clasificarea datelor procesate de către

partea convoluțională a rețelei. Ultimul strat conține funcția de activare *softmax* (§2.2), specifică clasificării multi-clasă

Tabelul 3.1 prezintă structurat arhitectura rețelei convoluționale implementate. Este util de menționat faptul că primul element din componența dimensiunilor datelor de intrare și ieșire reprezintă dimensiunea grupului de date trimis (*batch size*, explicat în §3.1.4). Aceasta se notează cu 0 sau *None*, fiind o variabilă necunoscută în momentul implementării arhitecturii. Rețeaua convoluțională a fost implementată folosind biblioteca *Keras*, conținută în framework-ul *TensorFlow* [32].

Tabelul 3.1: Arhitectura rețelei convoluționale implementate

#	Strat	Intrare	Ieșire	Parametri
1	Conv2D (ReLU)	(0, 20, 20, 1)	(0, 20, 20, 32)	160
2	Conv2D (ReLU)	(0, 20, 20, 32)	(0, 19, 19, 32)	4128
3	BatchNormalization	(0, 19, 19, 32)	(0, 19, 19, 32)	128
4	Dropout (0.3)	(0, 19, 19, 32)	(0, 19, 19, 32)	0
5	Conv2D (ReLU)	(0, 19, 19, 32)	(0, 10, 10, 64)	8256
6	Conv2D (ReLU)	(0, 10, 10, 64)	(0, 9, 9, 64)	16448
7	BatchNormalization	(0, 9, 9, 64)	(0, 9, 9, 64)	256
8	Dropout (0.25)	(0, 9, 9, 64)	(0, 9, 9, 64)	0
9	Conv2D (ReLU)	(0, 9, 9, 64)	(0, 4, 4, 128)	32896
10	Conv2D (ReLU)	(0, 4, 4, 128)	(0, 3, 3, 256)	131328
11	BatchNormalization	(0, 3, 3, 256)	(0, 3, 3, 256)	1024
12	Flatten	(0, 3, 3, 256)	(0, 2304)	0
13	Dense (ReLU)	(0, 2304)	(0, 256)	590080
14	Dropout (0.5)	(0, 256)	(0, 256)	0
15	Dense (softmax)	(0, 256)	(0, 3)	771

3.1.4 Antrenarea rețelei

Antrenarea rețelei a fost realizată folosind mediul online *Google Colaboratory*, care pune la dispoziție atât resurse hardware cât și software specializate pentru dezvoltarea algoritmilor de învățare automată. Resursele hardware folosite pentru antrenare au fost: placa grafică NVidia Tesla K80, procesorul Intel Xeon @ 2.00 GHz și 13GB RAM. Folosind aceste resurse, timpul total de antrenare al rețelei a fost de aproximativ 3 secunde per epocă.

Antrenarea a fost realizată folosind:

- Funcția de cost: *Categorical Cross-Entropy*, folosită în problemele de clasificare multi-clasă, se folosește în conjuncție cu funcția de activare *softmax* de la ieșirea rețelei pentru a calcula un vector de probabilități de dimensiunea numărului de clase [33]. Clasa fiecărei înregistrări este codată folosind tehnica *one-hot encode* care presupune scrierea clasei sub forma unui vector care conține valoarea 1 la poziția specificată de clasa reprezentată, restul fiind 0, de exemplu $[0, 1, 0]$ pentru clasa 1. Astfel, doar scorul obținut pentru clasa respectivă va fi folosit în calculul valorii de *loss/cost*

- Funcția de optimizare: Pentru optimizarea funcției de cost a fost folosită funcția *Adam* [34], folosind o rată de învățare inițială $\eta = 7 * 10^{-4}$ și reducerea acesteia cu factorul $7 * 10^{-2}$ dacă valoarea funcției de cost nu se îmbunatățește pentru o perioadă de 5 epoci, până la valoarea minimă 10^{-5}

Modelul realizat a fost antrenat pentru o perioadă de 100 de epoci, folosind un *grup (batch size)* de 32 de imagini. Epoca este perioada de timp în care toate înregistrările conținute în setul de antrenare sunt trecute prin rețea. *Batch size* reprezintă o serie/grup de înregistrări pe care rețeaua le ia în considerare înainte de a-și modifica parametrii. Prin folosirea unor serii în detrimentul unei singure înregistrări, capacitatea de generalizare a informației de către rețeaua antrenată crește, rezultând o mai bună adaptare către noi date. Acest procedeu

permite folosirea calculului paralel, prin distribuirea înregistrărilor pe mai multe fire de execuție rezultând un timp de antrenare al rețelei mai scăzut.

3.2 Rezultate

În această secțiune sunt prezentate rezultatele obținute de modelul rețelei convoluționale implementate. În Tabelul 3.2 sunt prezentate comparativ rezultatele obținute în urma antrenării modelului rețelei convoluționale implementate pe setul de date creat, folosind ca algoritm de selecție al atributelor pe rând două metode: metoda de testare *chi-pătrat*, iar mai apoi metoda *informației mutuale*. Testele comparative au fost realizate de 3 ori pentru fiecare metodă, folosind setul de test.

Tabelul 3.2: Prezentarea performanțelor modelului folosind metodele *chi-pătrat* și *informație mutuală* (§3.1.2) pentru selecția atributelor

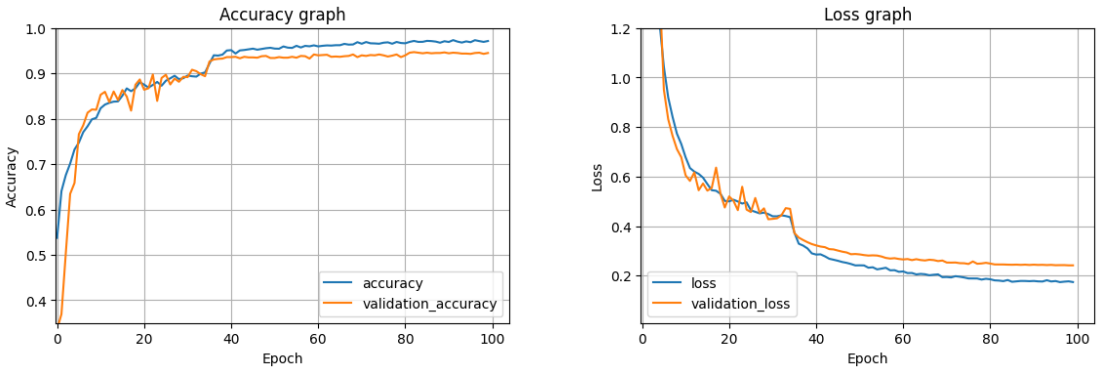
Metodă	#	Valoare loss	Valoare acuratețe
	1	0.268	0.931
Chi pătrat	2	0.278	0.919
	3	0.269	0.924
Media		0.272	0.925
	1	0.283	0.928
Informație	2	0.282	0.920
mutuală	3	0.282	0.927
Media		0.282	0.925

Rezultatele inițiale obținute fiind foarte asemanatoare, am decis folosirea metodei chi-pătrat pentru selecția atributelor. Toate rezultatele prezentate în continuare reies din setul de date format prin această metodă.

Evoluția pe perioada antrenării a acurateței și a valorii funcției de cost poate fi observată în Figura 3.5. Colorat cu albastru este reprezentată acuratețea/valoarea funcție de cost a modelului provenite din analizarea performanțelor acestuia pe setul de antrenare. Cu portocaliu este reprezentată performanța acestuia în clasificarea datelor din setul de validare. Deoarece modelul își ajustează parametrii, în cea mai mare măsură, în funcție de setul de antrenare, metricile rezultate pentru acest set vor fi întotdeauna mai bune decât metricile calculate pe restul seturilor de date. Pe parcursul antrenării, la finalul fiecărei epoci, setul de validare este folosit pentru evaluarea modelului și ajustarea parametrilor acestuia.

Urmărind cele două grafice putem observa faptul că evoluția metricilor modelului calculate pe setul de validare este foarte apropiată de cele calculate pe setul de antrenare. Acest lucru sugerează o performanța bună a clasificatorului, acesta fiind capabil să generalizeze datele primite la învățare și să aplice aceste „cunoștiințe" pentru a clasifica cât mai corect posibil date pentru care nu este antrenat în mod direct. Din aceste grafice mai poate fi observată utilitatea modificării ratei de învățare. La epoca 36, rata de învățare se diminuează cu 0.07 ajungând de la $7 * 10^{-4}$ la valoarea $4.9 * 10^{-5}$, crescând acuratețea modelului și diminuând valoarea funcției de cost. Este posibil ca acest lucru să se fi întâmplat și fară a face această operațiune, însă pe o perioadă de timp mult mai lungă, crescând timpul total de învățare.

După finalizarea antrenării modelului, capacitățile acestuia au fost testate asupra setului de date de test, date pe care acesta nu le „văzuse" niciodată. Evaluarea a fost efectuată de trei ori, rezultând o acuratețe medie de $\approx 92\%$ și o valoarea medie a funcție de cost de ≈ 0.2.



(a) (b)

Figura 3.5: Reprezentările grafice ale evoluției acurateței modelului (a) și evoluției valoarii funcției de cost (b)

Pentru determinarea performanțelor, pe lângă acuratețea modelului și valoarea funcției de cost, au mai fost calculate următoarele metrici, valorile rezultate sunt prezentate în Tabelul 3.3:

- Precision: reprezintă precizia clasificatorului, fiind determinată de proporția cazurilor *true positive* din totalitatea prezicerilor clasificate ca aparținând de clasa adevărată (suma clasificărilor *true positive* și *false positive*)
- Recall: reprezintă „rata de regăsire„ a clasificatorului, fiind determinată de proporția cazurilor *true positive* din totalitatea prezicerilor care aparțin cu adevărat de clasa respectivă (suma clasificărilor *true positive* și *false negative*)
- F_β Score: reprezintă o media armonică ponderată între precision și recall. Dacă ponderea β are valoarea 1, cele două metrice sunt luate în considerare în aceeași proporție, fiind considerate la fel de importante

Figura 3.6 afișează performanțele clasificatorului prin așezarea prezicerilor acesteia într-o matrice denumită Matricea Confuziilor

Tabelul 3.3: Reprezentarea valorilor metricilor *precision*, *recall* și F_1 *score* pentru fiecare clasă

	Precision	Recall	F_1 Score
neutru	0.91	0.95	0.93
concentrat	0.97	0.94	0.93
relaxat	0.92	0.90	0.91

(Confusion Matrix). Matricea conține pe diagonala principală toate clasificările corecte.

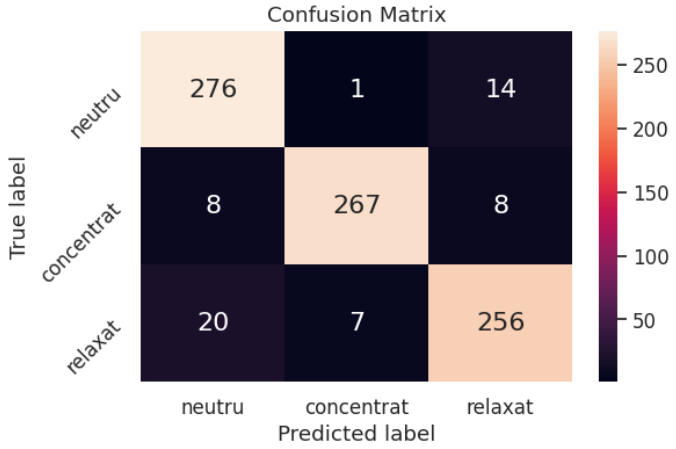


Figura 3.6: Reprezentarea performanței rețelei prin matricea confuziilor

Din datele afișate atât în Tabelul 3.3, cât și în Figura 3.6, unde este reprezentată matricea confuziilor, poate fi observată dificultatea clasificatorului de a distinge clasele *neutru* și *relaxat*. Această dificultate în distingere apare datorită metodelor asemănatoare de culegere a datelor EEG pentru ambele clase, descrise în §3.1.1.

Capitolul 4

Concluzii

Bibliografie

[1] Bin Qian, Jie Su, Zhenyu Wen, Renyu Yang, Albert Zomaya, and Omer Rana. Orchestrating development lifecycle of machine learning based iot applications: A survey. 10 2019.

[2] Christopher Gatti and M. Embrechts. Reinforcement learning with neural networks: Tricks of the trade. *Studies in Computational Intelligence*, 410:275–310, 01 2013.

[3] EMOTIV. What is an eeg headset? definition & faqs. `https://www.emotiv.com/glossary/eeg-headset/`. [Online; Accesat 29.04.2020].

[4] Alejandro Morán and Miguel C. Soriano. Improving the quality of a collective signal in a consumer eeg headset. *PLOS ONE*, 13(5):1–21, 05 2018.

[5] Jordan Bird, Luis Manso, Eduardo Ribeiro, Aniko Ekart, and Diego Faria. A study on mental state classification using eeg-based brain-machine interface. 09 2018.

[6] Jodie Ashford, Jordan Bird, Felipe Campelo, and Diego Faria. *Classification of EEG Signals Based on Image Representation of Statistical Features*, pages 449–460. 01 2020.

[7] Michael A. Nielsen. Neural networks and deep learning. `http://neuralnetworksanddeeplearning.com/`, 2015. [Online; Accesat 29.04.2020].

[8] Mihnea VREJOIU. Reţele neuronale convoluţionale, Big Data şi Deep Learning în analiza automată de imagini. *Revista Română de Informatică şi Automatică*, 29:91–114, 04 2019.

[9] F. Rosenblatt. *Principles of neurodynamics: perceptrons and the theory of brain mechanisms*. Report (Cornell Aeronautical Laboratory). Spartan Books, 1962.

[10] W S McCulloch and W Pitts. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.*, 5:115–133, 1943.

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

[12] Saurabh Yadav. Weight Initialization Techniques in Neural Networks. `https://towardsdatascience.com/weight-initialization-techniques-in-neural-networks-26c649eb3b78`, 11 2018. [Online; Accesat 02.05.2020].

[13] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[14] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition, 2014.

[15] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G. Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017.

[16] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron Weiss,

and Kevin Wilson. CNN Architectures for Large-Scale Audio Classification. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017.

[17] Raghav Prabhu. Understanding of Convolutional Neural Network (CNN) — Deep Learning. https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99 03 2018. [Online; Accesat 02.05.2020].

[18] Krizhevsky, Alex and Sutskever, Ilya and Hinton, Geoffrey E. ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

[19] Craig Will. Does pooling in convolutional networks actually work? https://principlesofdeeplearning.com/index.php/2018/08/27/is-pooling-dead-in-convolutional-networks/, 27 2018. [Online; Accesat 27.04.2020].

[20] Paul-Louis Pröve. An Introduction to different Types of Convolutions in Deep Learning. https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d, 07 2017. [Online; Accesat 28.09.2020].

[21] Facultatea de Educație Fizică și Sport, Universitatea din Craiova. Introducere in sistemul nervos. https://efs.ucv.ro/pdf/studenti/cursuri_master/note_curs_nervos.pdf, 11 2007. [Online; Accesat 29.04.2020].

[22] Marius Ignătescu. Cum functionează neuronul biologic. https://www.descopera.org/cum-functioneaza-neuronul-biologic/, 11 2009. [Online; Accesat 29.04.2020].

[23] Vecteezy. Diagram of neuron anatomy. https://www.vecteezy.com/vector-art/358962-diagram-of-neuron-anatomy, 03 2019. [Online; Accesat 01.05.2020].

[24] Brainworks. What Are Brainwaves ? Types Of Brain Waves | EEG Sensor And Brain Wave – UK. https://brainworksneurotherapy.com/what-are-brainwaves. [Online; Accesat 29.04.2020].

[25] Scientific American. What is the function of the various brainwaves? https://www.scientificamerican.com/article/what-is-the-function-of-t-1997-12-22/, 2018. [Online; Accesat 29.04.2020].

[26] Alexandre Barachant. Muse LSL. https://github.com/alexandrebarachant/muse-lsl. [Online; Accesat 20.04.2020].

[27] Muse. Muse electrode placement. http://developer.choosemuse.com. [Online; Accesat 15.05.2020].

[28] M. Popa. Distribuţia multinomială. Testul chi-pătrat. https://psygnificant.files.wordpress.com/2013/01/st1_13_multinomial.pdf. [Online; Accesat 26.05.2020].

[29] Facultatea Transporturi, Universitatea Politehnica București. Entropie Şi informaŢie. http://tet.pub.ro/pages/Tti/tic_cap_2.pdf. [Online; Accesat 26.05.2020].

[30] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[31] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors, 2012.

[32] Google Brain Team. TensorFlow. https://www.tensorflow.org/guide/keras. [Online].

[33] Raúl Gómez Bruballa. Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names. https://gombru.github.io/2018/05/23/cross_entropy_loss/, 5 2018. [Online; Accesat 28.05.2020].

[34] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.