Documentație proiect Machine Learning

Voica Ștefan-Alexandru

Grupa 251

Introducere

Acest proiect își propune clasificarea imaginilor în cinci categorii, folosind modele de Machine Learning. Am abordat două tipuri de modele de clasificare: o rețea de perceptroni multi-strat (MLP) și mai multe variante de rețele convoluționale (CNN), combinându-le pe cele din urmă într-un ansamblu.

Descrierea primei arhitecturi (MLP) – 59%

În această secțiune, voi descrie prima arhitectură testată, un model MLP, care a avut performanțe slabe comparativ cu CNN-urile ulterioare.

- **Descrierea setului de trăsături**

Imaginile au fost convertite în vectori de trăsături folosind tensorii asociați pixelilor din imagini. Nu s-au folosit trăsături extrase manual, deoarece modelele învățau automat reprezentările relevante.

- **Preprocesare și augmentare**

Augmentarea a fost aplicată doar pe setul de antrenament și a constat în oglindirea orizontală a imaginilor. Fiecare imagine originală a fost completată cu o versiune oglindită orizontal, dublând astfel volumul setului de antrenament.

- **Arhitectura modelului**

Modelul MLP este compus exclusiv din straturi complet conectate.

Nr. Crt.	**Tip strat**	**Dimensiuni intrare-ieșire**	**Activare / Dropout**
1	Flatten	3 x 100 x 100 -> 30000	-
2	Linear	30000 -> 512	ReLU + Dropout(0.3)
3	Linear	512 -> 128	ReLU + Dropout(0.3)
4	Linear (Output)	128 -> 5	Softmax

- **Antrenarea modelului și hiperparametri**

Modelul a fost antrenat pe 125 de epoci, folosind optimizatorul AdamW. Acesta include o penalizare weight_decay pentru a evita overfitting-ul. În plus, am utilizat un scheduler de tip Cosine Annealing, care scade treptat learning rate-ul de la valoarea inițială către un minim prestabilit (eta_min = 1e-5).

Hiperparametru	**Valoare**
Learning Rate	0.001
Batch Size	32
Epochs	125
Optimizator	AdamW
Scheduler	CosineAnnealingLR (T_max=125, n_min=1e-5)
Weight Decay	1e-4 (pentru regularizare L2)
Loss Function	CrossEntropyLoss
Augmentare	Flip orizontal (doar pe train set)
Normalizare	Media și deviația standard calculate pe train

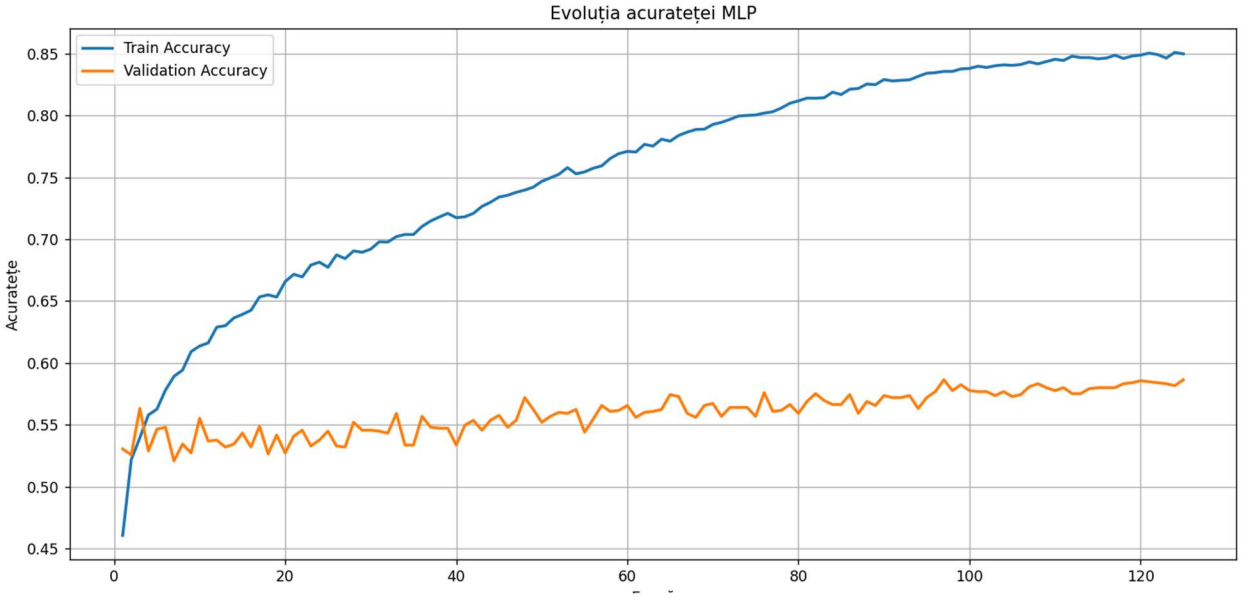
- **Încercări eșuate**

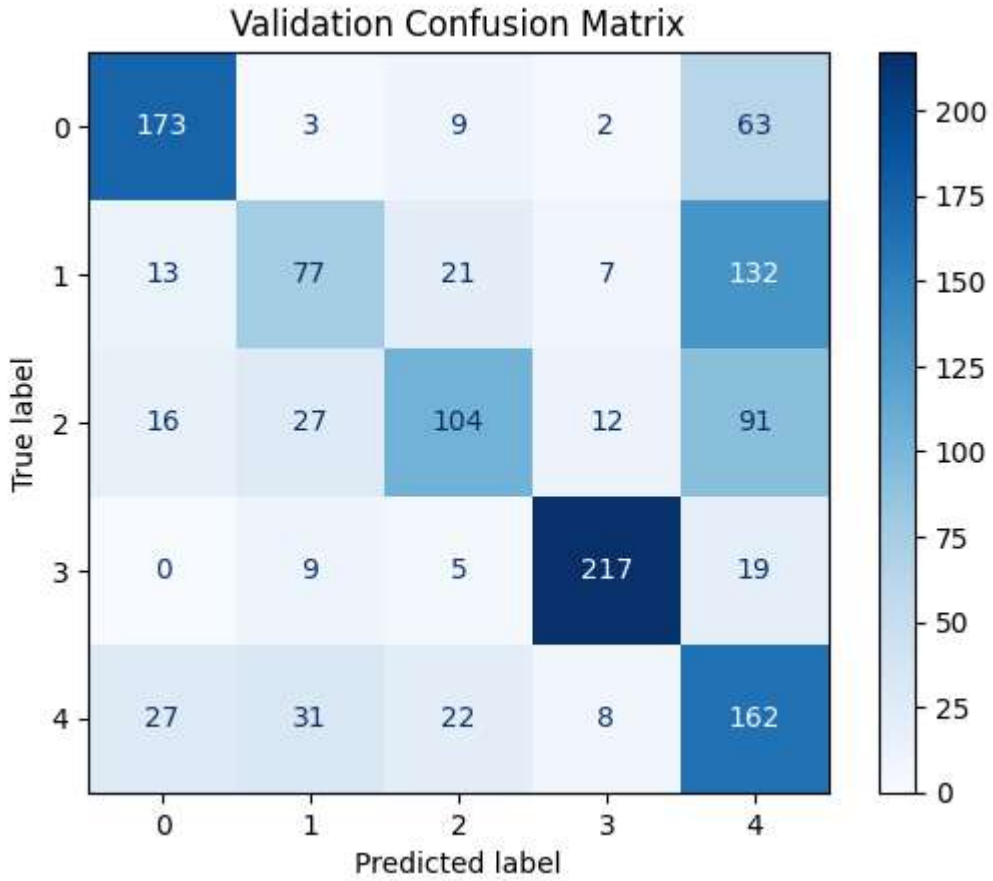
Metodă	**Rezultat**
Fără augmentare	Overfitting după 30 epoci
Learning Rate = 0.01	Modelul diverge, loss crescut rapid
SGD ca optimizator	Convergență lentă, acuratețe redusă (40%)
Dimensiune imagine 64x64	Informație pierdută, performanță slabă
Fără Dropout	Modelul face overfitting mult mai repede

- **Perfomanță obținută**

Modelul MLP a obținut o acuratețe slabă pe setul de validare(59%), demonstrând că nu este suficient pentru sarcina de clasificare a imaginilor deepfake. Acuratețea pe setul de

antrenament crește constant, semn că modelul învață bine datele de train. Pe setul de validare, acuratețea rămâne aproape constantă și mult mai mică decât pe train. După aproximativ 30 de epoci, validarea nu mai aduce îmbunătățiri semnificative.





Descriere detaliată a ansamblului convoluțional – 92,68%

Ansamblul este compus din mai multe rețele neuronale convoluționale (CNN), fiecare având ca scop clasificarea imaginilor în cinci clase distincte. După antrenare, greutățile modelelor au fost salvate în fișierele *model1.pt*, *model2.pt*, *model3.pt* și *model4.pt*. Toate rețelele au fost implementate în PyTorch, folosind o arhitectură de bază comună: patru blocuri convoluționale urmate de un clasificator *fully-connected*.

În continuare, voi descrie în detaliu modelul *model1.pt*, inclusiv procesul de ajustare a hiperparametrilor. Celelalte modele păstrează structura arhitecturală de bază, dar introduc variații precum modificarea numărului de neuroni în stratul final, ajustarea ratei de regularizare (*weight_decay*) sau aplicarea unor metode suplimentare de augmentare (de exemplu, rotația aleatorie a imaginilor în timpul antrenării). Antrenarea a fost realizată pe același set de date, folosind aceleași transformări, iar performanțele au fost evaluate și comparate pe setul de validare.

Modelul CNN256 (model1.pt)

Modelul *model1.pt* reprezintă prima instanță a rețelei convoluționale utilizate în proiect. Acesta servește drept punct de referință pentru evaluarea celorlalte modele testate. Arhitectura sa include patru blocuri convoluționale, fiecare urmat de operații de normalizare, activare ReLU și pooling, iar la final, un clasificator *fully-connected* care produce predicția finală. Modelul a fost implementat cu ajutorul bibliotecii PyTorch și antrenat pe setul de date inițial, fără augmentări suplimentare în afara oglinzirii imaginilor în setul de antrenare.

După antrenare, greutățile acestui model au fost salvate în fișierul *model1.pt* și au fost ulterior utilizate atât pentru validare, cât și ca parte a ansamblului de modele.

Preprocesarea și augmentarea datelor

Pentru acest proiect am avut la dispoziție imagini RGB, etichetate în 5 clase distincte. Preprocesarea și augmentarea au avut un rol esențial în performanțele rețelelor convoluționale.

Pentru antrenarea modelului am implementat o tehnică simplă dar eficientă de augmentare: flip orizontal (oglindire stânga-dreapta). Astfel, în momentul încărcării setului de date am aplicat această augmentare care a dublat numarul de imagini pe care s-a antrenat modelul.

Această augmentare a fost aplicată în funcția de încărcare *load_dataset*. La fiecare imagine originală, se genera și versiunea oglindită, iar ambele erau adăugate în setul de antrenament împreună cu eticheta corectă.

Reprezentarea caracteristicilor

La intrare, modelul primește imagini color de dimensiunea 100x100x3. Acestea sunt normaliate conform mediei și deviației standard calculate din setul de antrenare pe fiecare canal.

Nr.	Componentă	Tipul stratului	Parametri	Dimensiune ieșire
1	Conv2d(3, 32, 3, padding=1)	Convoluțional	3 canale intrare, 32 filtre, kernel 3×3, padding 1	32×100×100
2	MaxPool2d(2)	Pooling	Kernel 2x2	32×50×50
3	Conv2d(32, 64, 3, padding=1)	Convoluțional	64 filtre noi, kernel 3×3	64×50×50
4	MaxPool2d(2)	Pooling	Kernel 2x2	64x25x25
5	Conv2D(64, 128, 3, padding=1)	Convoluțional	64 canale, 128 filtre	128x25x25
6	MaxPool2D(2)	Pooling	Kernel 2x2	128x12x12
7	Conv2D(128, 256, 3, padding=1)	Convoluțional	128 canale, 256 filtre	256x12x12
8	MaxPool2D(2)	Pooling	Kernel 2x2	256x6x6

La finalul acestor blocuri, imaginea este comprimată într-un tensor cu dimensiunea 256×6×6 = 9216. Acest vector apalizat este trimis către clasificatorul fully-connected, nu înainte de a preveni supraînvățarea prin dezactivarea aleatoare a unor neuroni prin Dropout(0.5).

Clasificatorul

Clasificatorul este responsabil pentru conversia vectorului de caracteristici într-o predicție de clasă.

Layer 1	nn.Linear(256 * 6 * 6, 256)	Redimensionează și reduce dimensiunea caracteristicilor de la 9216 la 256
Activare	nn.ReLU()	Activare
Dropout	nn.Dropout(0.5)	Previne suprainvățarea
Layer 2	nn.Linear(256, num_classes)	Ultimul strat(256->5) produce logit-urile pentru cele 5 clase

Analiza hiperparametrilor

1. Numărul de epoci
   Am antrenament toate modelele pe 125 de epoci. În primele 40 de epoci, atât acuratețea pe setul de antrenare, cât și pe cel de validare creștea rapid, după care

progresul devenea mai lent. Am antrenat până la limitele supraînvățării, salvând modelul care performa cel mai bine pe setul de validare. Am încercat și variante cu early stopping, dar unele modele creșteau acuratețea pe validare cu 0.5% chiar și când se făcea overfitting clar.

Modificări testate

Număr de epoci	Acuratețe validare
5	83%
10	86%
50	90%
58-125	91-92%

2. Rata de învățare

   Am folosit un learning rate de 1e-3 pentru a obține cea mai stabilă convergență. Folosind scheduler-ul, rata a fost redusă treptat până la 1e-5 la finalul antrenării.

LR	Observații
1e-2	Modelul oscila și nu convergea
1e-4	Învățarea era prea lentă, ajungeam la doar 84% acuratețe pe validare.

3. Batch size

   Dimensiunea aleasă pentru batch a fost de 32 pentru a eficientiza rularea pe GPU. Am încercat să rulez imaginile individual, dar costul computational a fost prea mare. Am testat și o dimensiune de 64 în cadrul modelului 4, aspect prezentat în cadrul modelului 3.

4. Weight Decay

Weight decay-ul este un parametru de regularizare L2 aplicat în optimizator. În model1.pt, am folosit valorea 1e-4. Aceasta penalizează coeficienții prea mari, reducând riscul de overfitting.

Weight decay	Observații
1e-4	Standardul pe care l-am folosit, modelul stabilizându-se între 91-92%
5e-4	Modelul generaliza aproape la fel de bine, dar învățarea era mai lentă; obține rezultate asemănătoare (91-92%)
0	Fără regularizare, modelul supraînvăța ajungând la o acuratețe de 99% pe training după 10-15 epoci

5. Optimizator

Optimizatorul utilizat în antrenarea modelului CNN256 (model1.pt) a fost AdamW, datorită performanței sale superioare în sarcini de clasificare de imagini, în special în combinație cu weight_decay. AdamW este o variantă îmbunătățită a Adam care decuplează penalizarea L2 de actualizarea efectivă a parametrilor, ceea ce îl face mai robust pentru rețele adânci.

Alți optimizatori folosiți

Denumire	Hiperparametrii	Observații
SGD	Momentum=0.9 Learning_rate=0.01	Convergență mai lentă Acuratețea plafonată la 82% Fără scheduler, SGD a stagnat dupa 30 de epoci.

6. Scheduler – CosineAnnealingLR

Am ales scheduler-ul CosineAnnealingLR, care reduce progresiv rata de învățare de la o valoare inițială (1e-3) până la o valoare minimă (1e-5), urmând o funcție cosinusoidă. Această scădere lentă este eficientă în faza inițială când se permit pași mari și faza finală care conține pași mici pentru rafinare.

Scheduler	Descriere	Comportament
CosineAnnealingLR	Scade treptat LR cu o funcție cosinus	Stabil, eficient
StepLR	Scade LR în trepte la epoci fixe	Reducerea învățării mult prea devreme, underfitting, acuratețe 88,78% pe validare
ReduceLROnPlateau	Scade LR dacă validarea nu se îmbunătățește	Neregulat, prea lent, sensibil la fluctuații

7. Dropout

Pentru clasificatorul complet conectat, am folosit două straturi de Dropout, cu valori de **0.5**. Astfel, în fiecare pas de antrenare, 50% din neuroni sunt dezactivați aleator. Dezactivarea a avut ca scop reducerea overfitting-ului, forțând rețeaua să învețe reprezentări distribuite.

Modificări testate:

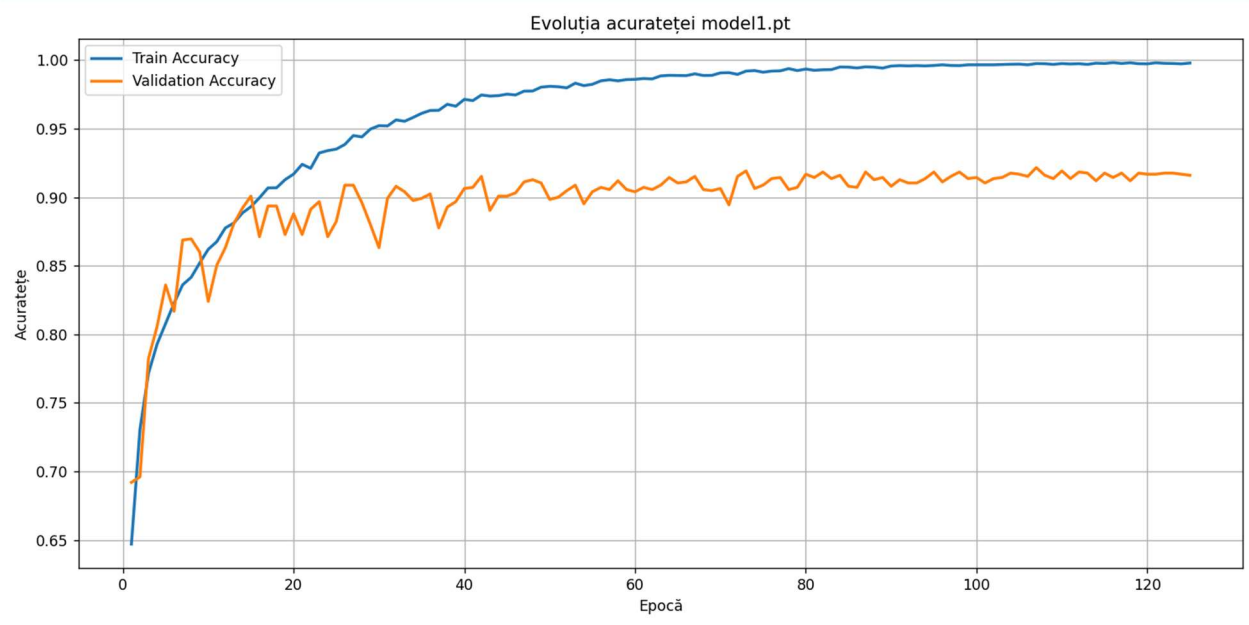
Valoare Dropout	Observații

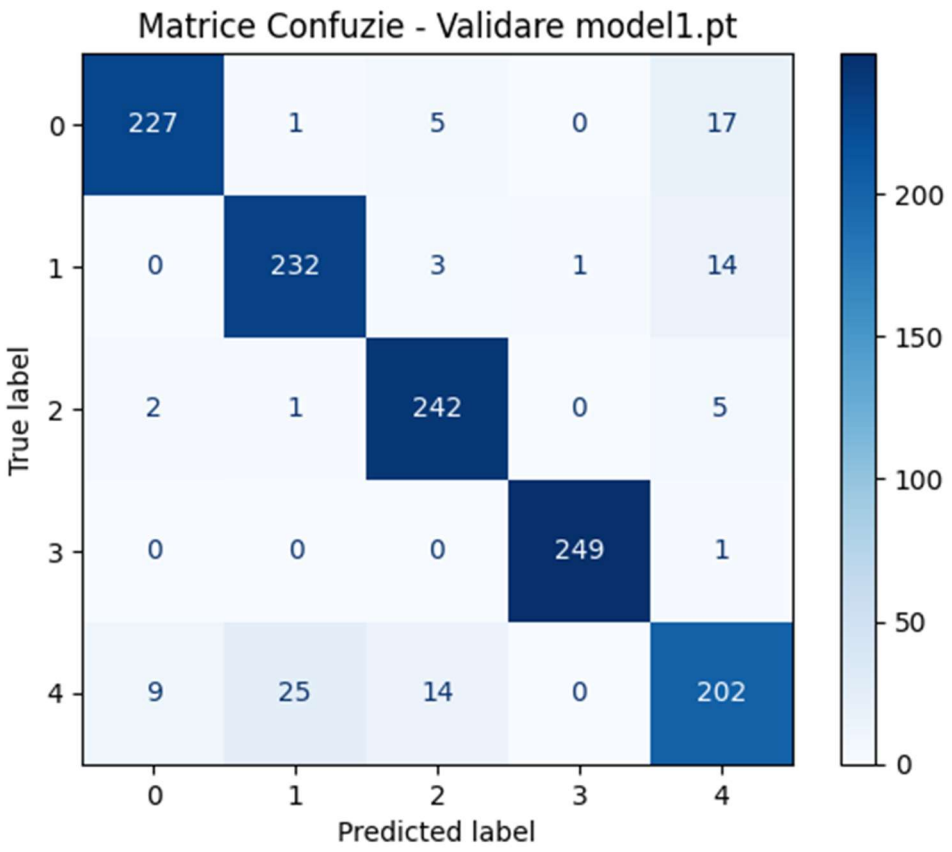
0	Modelul memora perfect datele, ajungând chiar la 100% acurateţe pe traing, performând prost pe validare
0.3	Uşor mai bine, dar în continuare se ajungea la 98-99% pe training
0.5	Am observant un overfitting mai lent, iar validarea a putut converge la 92%

8. **Funcția de pierdere – CrossEntropyLoss**

În cadrul tuturor experimentelor din acest proiect, inclusiv pentru antrenarea modelului 1, am utilizat funcția de cost CrossEntropyLoss, implementată nativ în PyTorch.

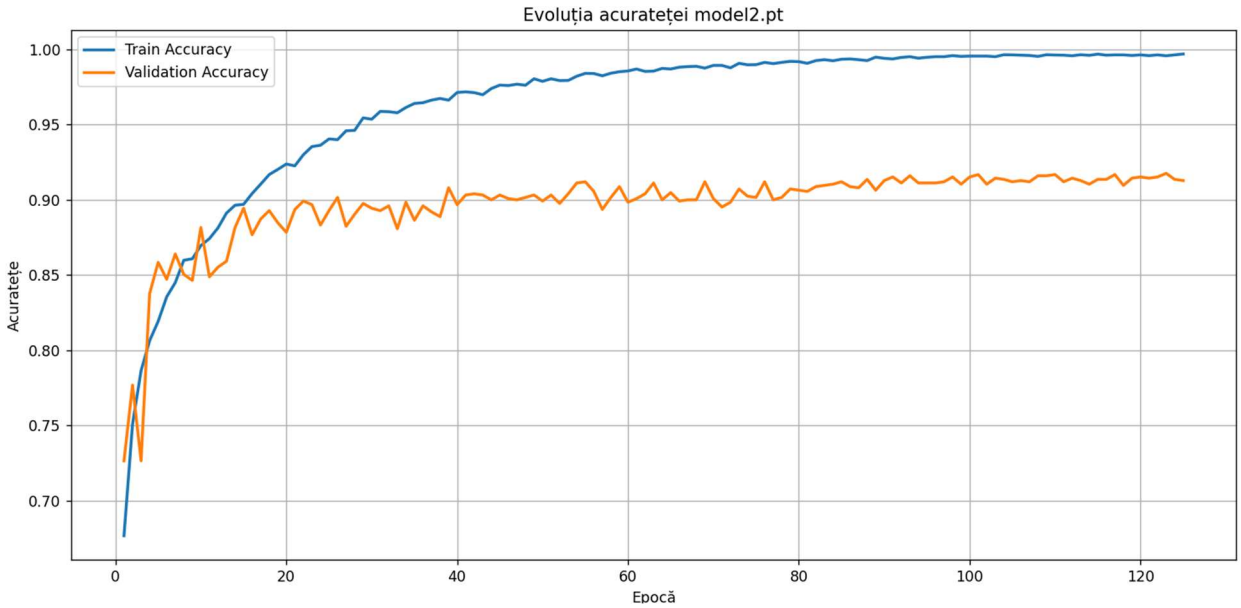
Aceasta este funcția standard în probleme de clasificare multi-clasă și măsoară diferența dintre distribuția prezisă și distribuția reală.



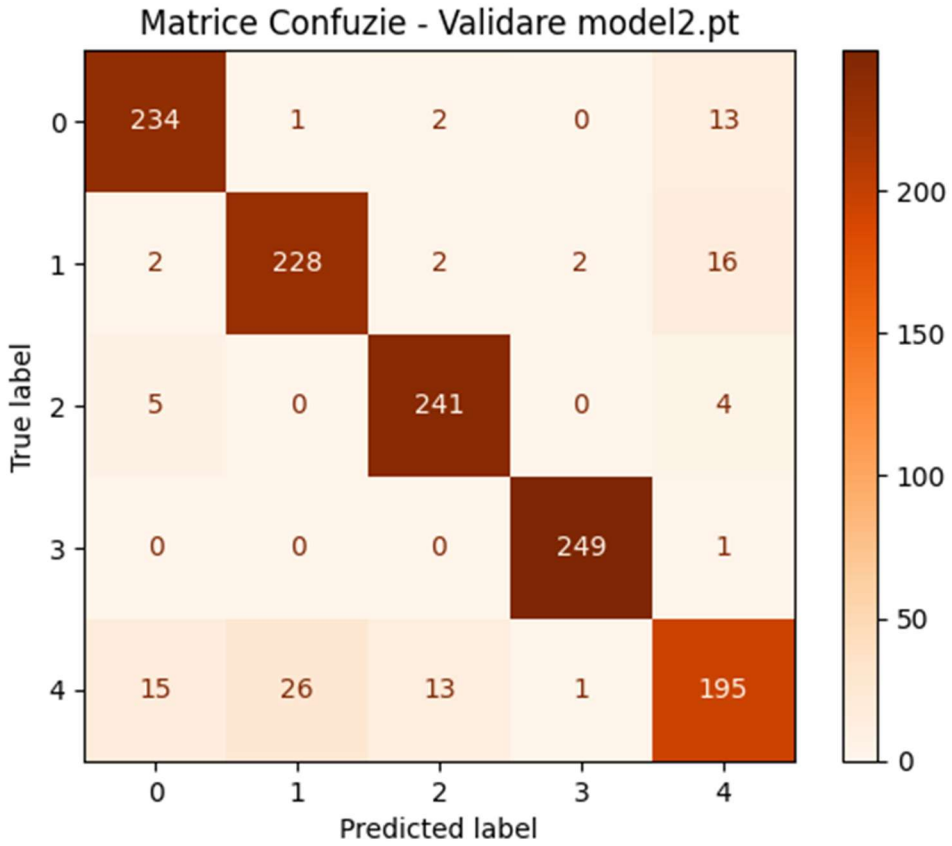


Modelul CNN128 (*model2.pt*)

Modelul *model2.pt* utilizează o arhitectură convoluțională similară cu cea a modelului *model1.pt*, păstrând aceleași blocuri convoluționale și aceeași structură generală. Diferența principală constă în dimensiunea stratului *fully-connected* intermediar din clasificator, care a fost redus la 128 de neuroni (față de 256 în modelul anterior). Această modificare a fost introdusă pentru a observa efectul reducerii capacității rețelei asupra performanței.



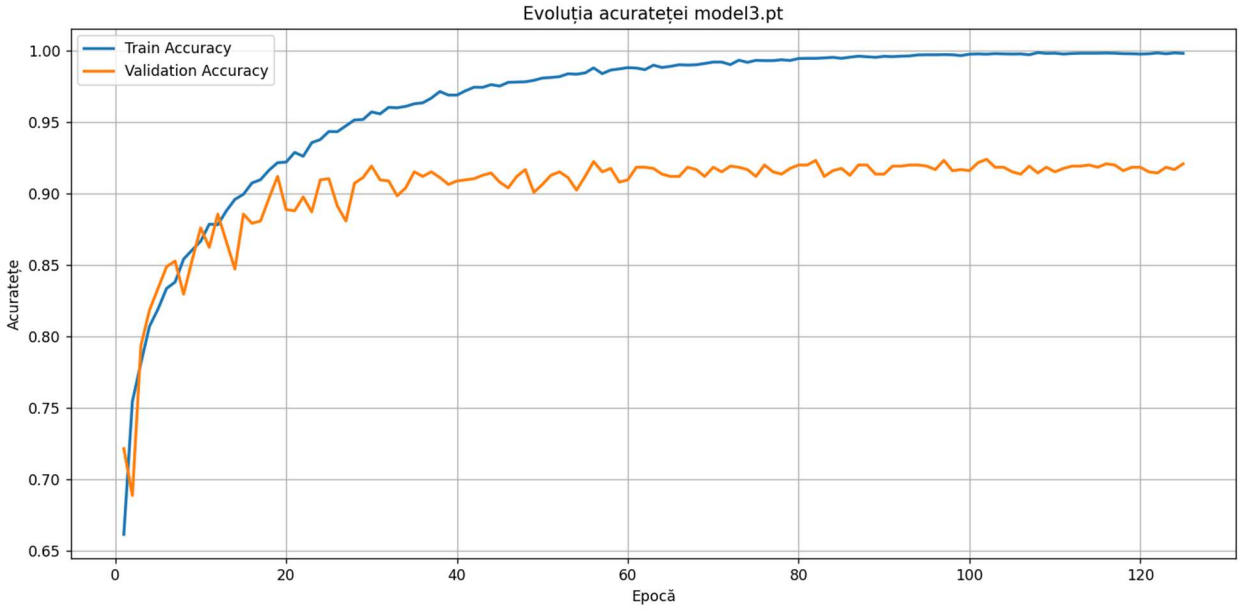
Scopul acestui experiment a fost compararea performanțelor obținute de o rețea cu o capacitate mai redusă. Astfel, am observat posibile îmbunătățiri în generalizare și timp de antrenare.

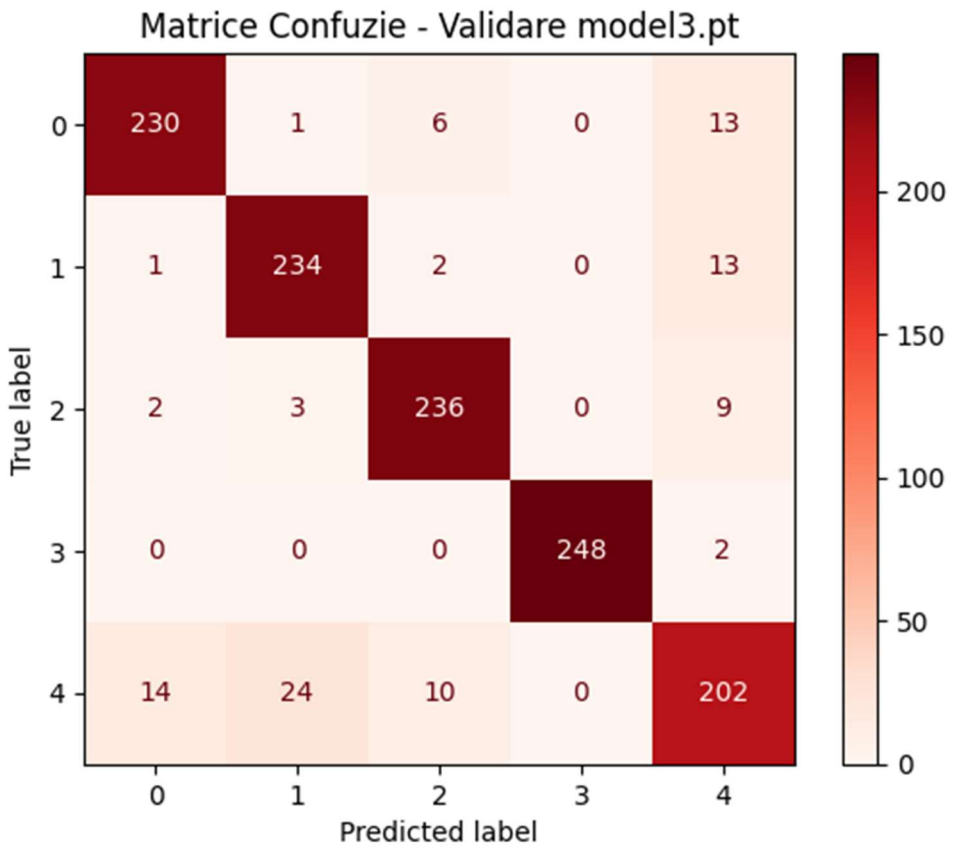


Modelul CNN256 (*model3.pt*)

Modelul *model3.pt* folosește aceeași arhitectură ca și *model1.pt* (CNN256), menținând structura cu patru blocuri convoluționale și un clasificator *fully-connected* cu 256 de neuroni în stratul intermediar. Diferența față de *model1.pt* constă în modificarea hiperparametrului de regularizare: valoarea *weight_decay* a fost crescută de la 1e-4 la 5e-4.

Această ajustare are ca scop testarea efectului regularizării mai puternice asupra capacității modelului de a evita supraînvățarea. Alte aspect au rămas neschimbate.

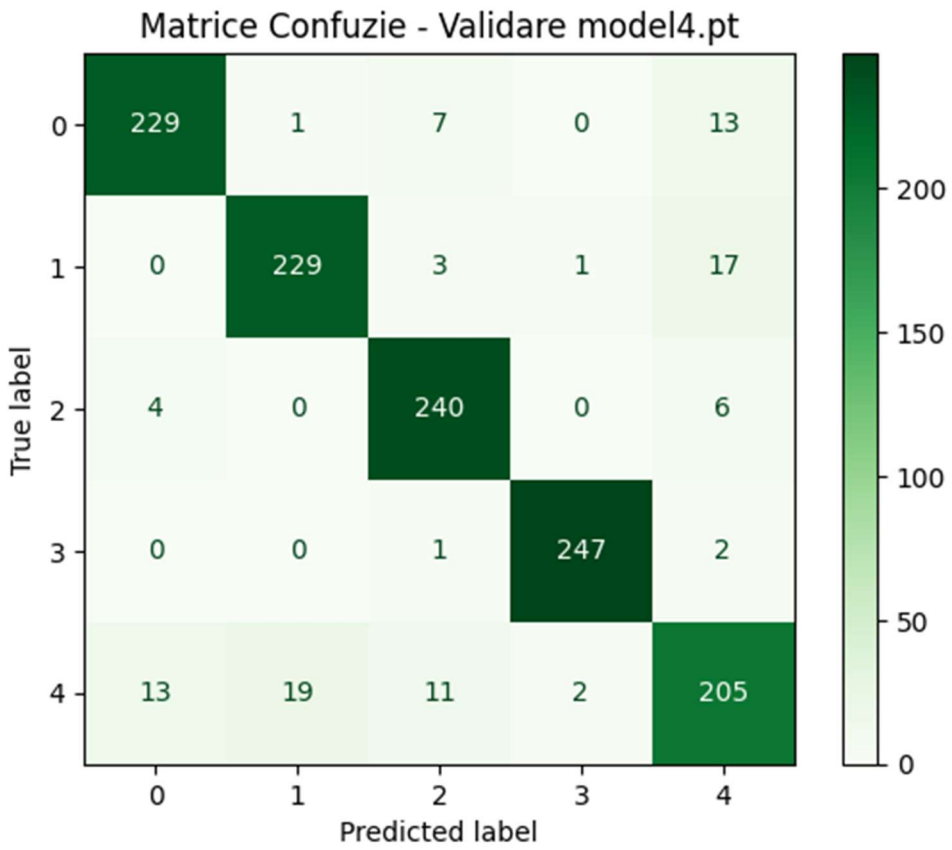


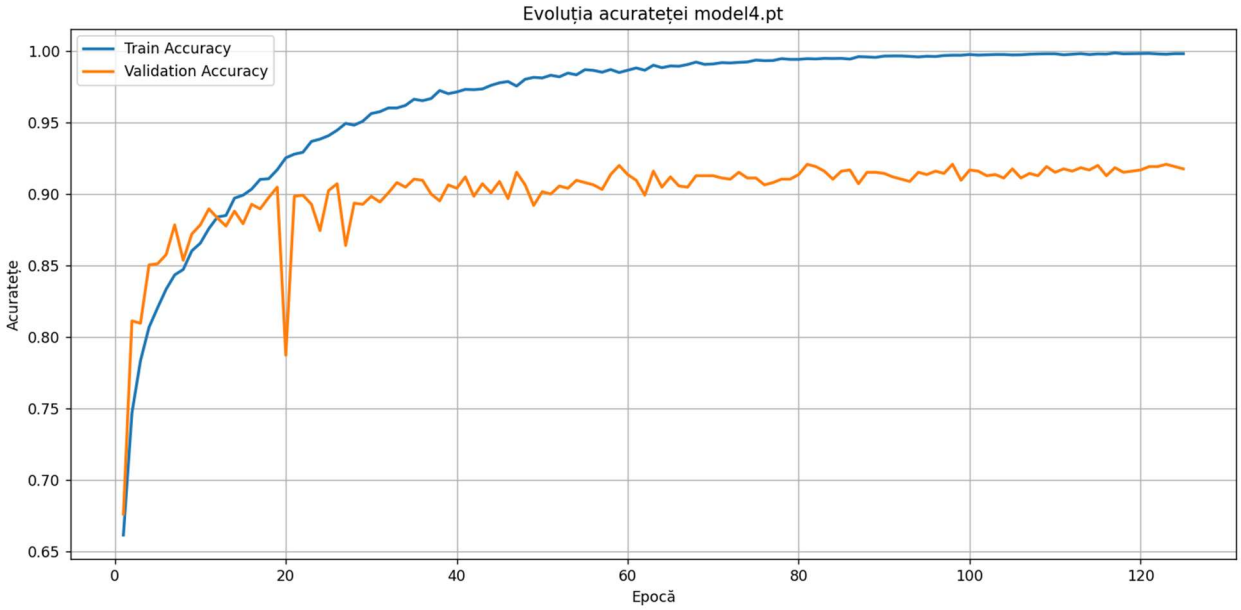


Modelul CNN256 (*model4.pt*)

Modelul *model4.pt* păstrează arhitectura de bază CNN256 utilizată și în *model1.pt* și *model3.pt*, cu același număr de blocuri convoluționale și clasificator *fully-connected* cu 256 de neuroni. Diferența principală adusă de acest model constă în aplicarea unei augmentări suplimentare a datelor: în timpul antrenării, imaginile sunt rotite aleator cu un unghi cuprins între -5° și +5° (*RandomRotation*).

Această tehnică are scopul de a crește diversitatea datelor de intrare și de a îmbunătăți capacitatea modelului de a generaliza. Față de *model3.pt*, hiperparametrul *weight_decay* a fost readus la valoarea inițială de 1e-4.



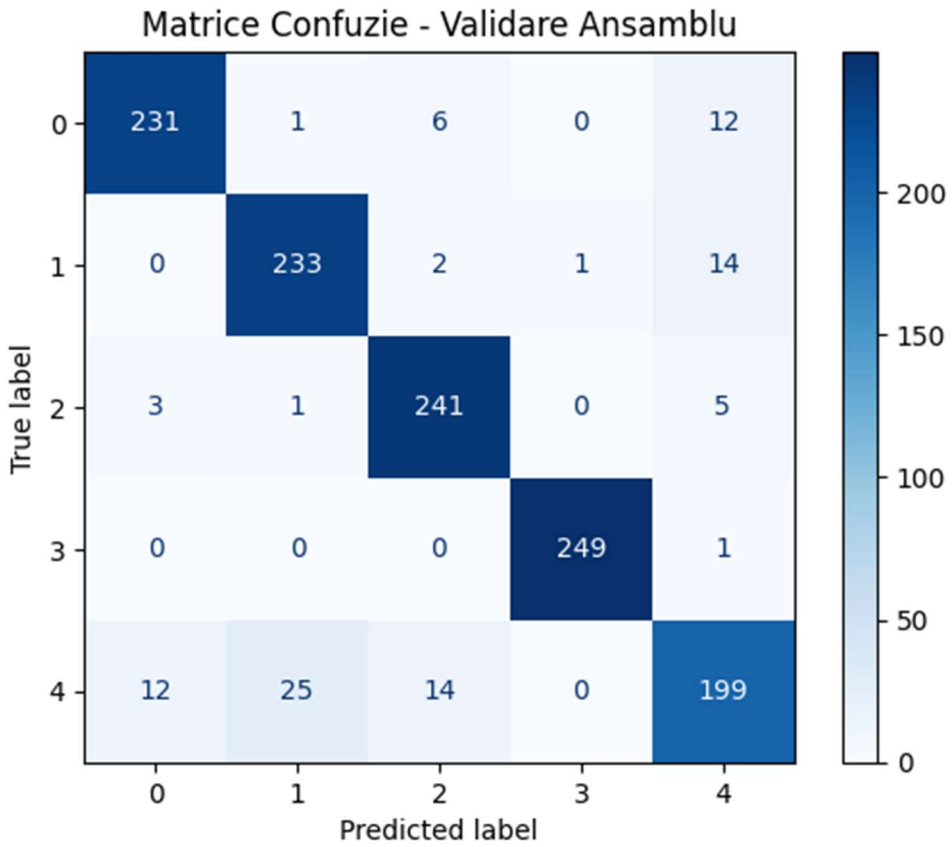


Ansamblul de modele

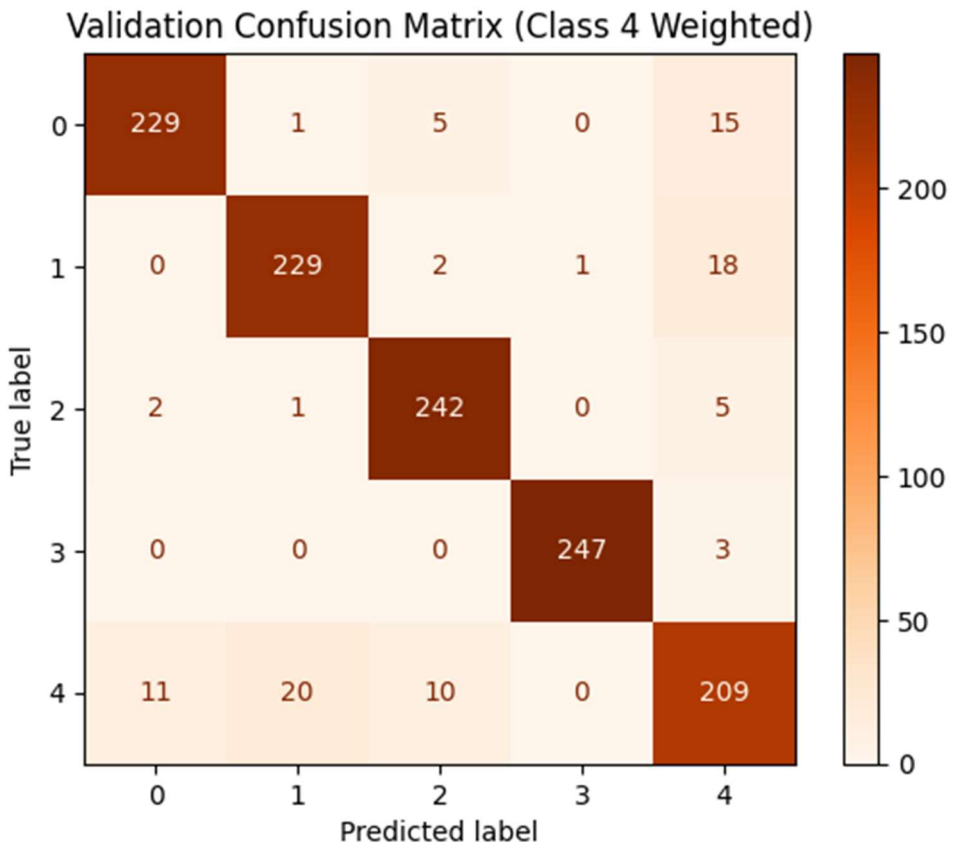
Pentru a încerca creșterea acurateței, am inclus cele 4 modele CNN menționate într-un ansamblu. Cum toate au avut la bază același set de imagini (imaginile normale plus cele

inversate), nu a fost o problemă în normalizarea ansamblului pentru datele de validare, respective de testare.

În varianta de bază a ansamblului, fiecare model face o predicție sub formă de probabilități (cu ajutorul funcției softmax). Apoi, se face media acestor probabilități între toate modelele, și se alege clasa cu cea mai mare valoare. Practic, fiecare model „votează”, iar decizia finală se ia în funcție de scorurile medii. Un avantaj al acestei metode este următoarea: dacă un model greșește, celelalte îl pot „trage în sus”, și în general se ajunge la o decizie mai corectă. Ansamblul ajută mult la consistența predicțiilor, mai ales când modelele sunt antrenate diferit.



Am încercat și o variantă în care nu toate modelele aveau același cuvânt de spus. Mai exact, modelele 3 și 4, care s-au descurcat mai bine pe clasa 4, au primit o pondere mai mare atunci când decideam eticheta finală. Deși părea promițător, rezultatele finale nu s-au schimbat prea mult – am obținut tot o acuratețe de **92,48%**, aproape la fel ca în varianta simplă(**92,64%**). Așadar, am rămas cu varianta cu ponderi egale, care era mai simplă și mai stabilă.


Validation Confusion Matrix (Class 4 Weighted)
True label
Predicted label
0
1
2
3
4
229 1 5 0 15
0 229 2 1 18
2 1 242 0 5
0 0 0 247 3
11 20 10 0 209
200
150
100
50
0