Documentație Proiect Machine Learning
Deep Halucination Classification

Iulia-Georgiana Talpalariu
Grupa 234

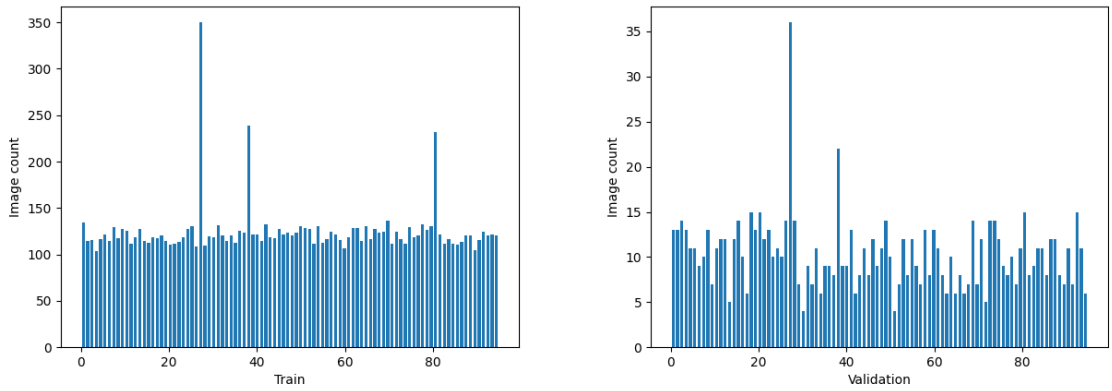
Iunie 2023

1 Introducere

Pentru a aduce o soluție la problema de clasificare a setului de date *Deep Halucination*, am abordat 2 strategii: SVM, CNN, cel mai mare punctaj obținându-l atât pe validare, cât și pe test cu rețeaua neurală convoluțională (CNN), inspirată de "trucurile" din rețele ca ResNet, dar și utilizând regularizarea de tip dropout, care nu se folosește de obicei în acest tip de rețele.

2 Prezentarea datelor

Înainte să încep încercarea de modele, m-am uitat la imagini - datele de intrare, care sunt 12.000 imagini color, la dimensiunea lor (64x64), dimensiune care va pune probleme în cazul unor clasificatoare, unde am încercat reducerea ei, am analizat distribuția imaginilor pe clase, pentru a înțelege, mai grafic, în ce mod învață unele clasificatoare. Din figura 1a, observăm o distribuție destul de uniformă pe train, cu unele clase (27, 38, 81) care au mai mulți reprezentanți, deci care ar putea fi mai ușor învățate de clasificatoarele pe care le voi antrena. Setul de validare conține 1.000 imagini cu aceleași caracteristici. Totuși pe validare (figura 1b), datele nu sunt atât de uniforme (fiind și mai reduse ca număr), dar distribuția seamănă în mare parte. Setul de test conține 5.000 imagini ale căror etichetă trebuie prezisă, utilizînd modele propuse.



(a) Datele de train.

(b) Datele de validare.

Figure 1: Distribuția datelor pe clase.

3 Metrici de evaluare a perfomanțelor modelelor propuse

În cursul rezolvării proiectului, am urmărit maximizarea valorii acurateței pe setul de validare, cu urmărirea (prin submisii) a valorilor pe setul de test. Pe lângă acuratețe, am urmărit și alte metrici importante: precizia și recall-ul.

3.1 Acuratețea

Acuratețea reprezintă procentul de imagini ale căror clasă este prezisă corect de modelul evaluat din numărul total de imagini. Această metrică este cea care trebuie maximizată în dezvoltarea unui model pentru acest proiect.

3.2 Precizia

Precizia pentru o clasă reprezintă câte dintre imaginile prezise cu acea clasă chiar au acea clasă și se calculează după formula: $precizia ClasaX = \frac{preziseCorectClasaX}{totalPreziseClasaX}$

3.3 Recall

Recall-ul pentru o clasă reprezintă pentru câte dintre imaginile cu o anumită clasă, modelul meu le pune clasa corectă și se calculează după formula: $recall ClasaX = \frac{preziseAdevaratClasaX}{totalAdevaratClasaX}$

3.4 F1-score

F1-score-ul combină cele două metrici cu formula : $F1 ScoreClasaX = 2 * \frac{precize*recall}{precizie+recall}$

4 Modele propuse

4.1 SVM

O încercare care a adus scoruri destul de bune, dar trainingul a durat destul de mult a fost SVM. Mașinile cu vectori suport (SVM) sunt utilizate pentru a găsi un hiperplan care separă 2 clase. Pentru a folosi SVM în cazul mai multor clase, avem 2 opțiuni: antrenăm pentru fiecare clasă un clasificator de tip One-vs-All sau antrenăm pentru fiecare pereche de clase distincte câte un clasificator One-vs-One. În primul caz ar trebui să antrenăm atâtea clafisicatoare câte clase avem, în cazul nostru 96, iar în al doilea caz, ar trebui să antrenăm atâtea clasificatoare câte perechi de clase distincte avem, adică $\frac{n*(n-1)}{2}$ clasificatoare distincte. În cazul nostru, având 96 de clase ambele valori sunt mari, totuși prima variantă pare o variantă care ar dura mai puțin timp. Astfel, am ales să antrenez 96 de clasificatoare de tip One-vs-Rest folosind clasificatorul care asamblează aceste 96 de modele cel deja implementat în *sklearn.multiclass.OneVsRestClassifier*. Acesta prezice eticheta unei imagini pe baza clasificatorului cu scorul cel mai mare. Pentru SVM, nu am folosit normalizarea datelor. În schimb imaginile sunt micșorate la 32x32 pixeli și nu utilizez valorile pixelilor ca atare, ci valorile aduse în intervalul 0 - 1 și transformate cu flatten astfel încât să am un singur vector de valori, nu o matrice (datorită formatului RBG - 32x32X3).

4.1.1 Hiperparametrii

În cazul SVM, avem puțini hiperparametrii de ajustat. Primul este C, care reprezintă parametrul de penalitate pentru eroare sau o măsură invers proporțională cu marginea hiperplanului. Dacă avem un C mare, avem un hiperplan cu margine mică, iar pentru un C prea mare putem face ușor overfitting pe datele de antrenare, ceea ce va duce la rezultate mai slabe pe datele de test. Un C prea mic ar putea duce la underfitting pentru că marginea hiperplanului devine mai mare și voi avea mai multe date despre care nu știu. În cazul alegerii funcției de kernel ca fiind "rbf" sau Radial Basis Function (este kernelul default în cazul SVM), putem seta și ajusta și un al doilea hiperparametru - gamma. Funcția de kernel ajută la separarea datelor neliniar separabile prin traspunerea acestora într-un sistem cu mai multe dimensiuni unde SVM ar putea găsi un hiperplan de separare. Funcția kernel RBF se folosește de similitudinea între exemple și este definită prin $exp(-\gamma * \|x_1 - x_2\|^2)$. Gamma reglează cât de regulată sau curbată e granița de separare, un gamma mare face ca granița să fie neregulată, pe când un gamma mai mic face ca granița de separare să fie mai netedă. Cu un asftel de clasificator One-vs-Rest care utilizează 96 de clasificatoare SVM cu C = 10, kernelul default și gamma default, am obținut acuratețea de 42.8%, cu C=1 (default) am obținut 41%, iar cu C=0.1 am obținut 41.2%.

Table 1: Variatia acuratetei cu C

C	Acuratetea
0.1	40.6
1	41.2
10	42.8

Table 2: Variatia acuratetei cu γ pentru C setat la 10

γ	Acuratetea
default	42.8
0.01	42.5
0.001	36.5

4.1.2 Matricea de confuzie

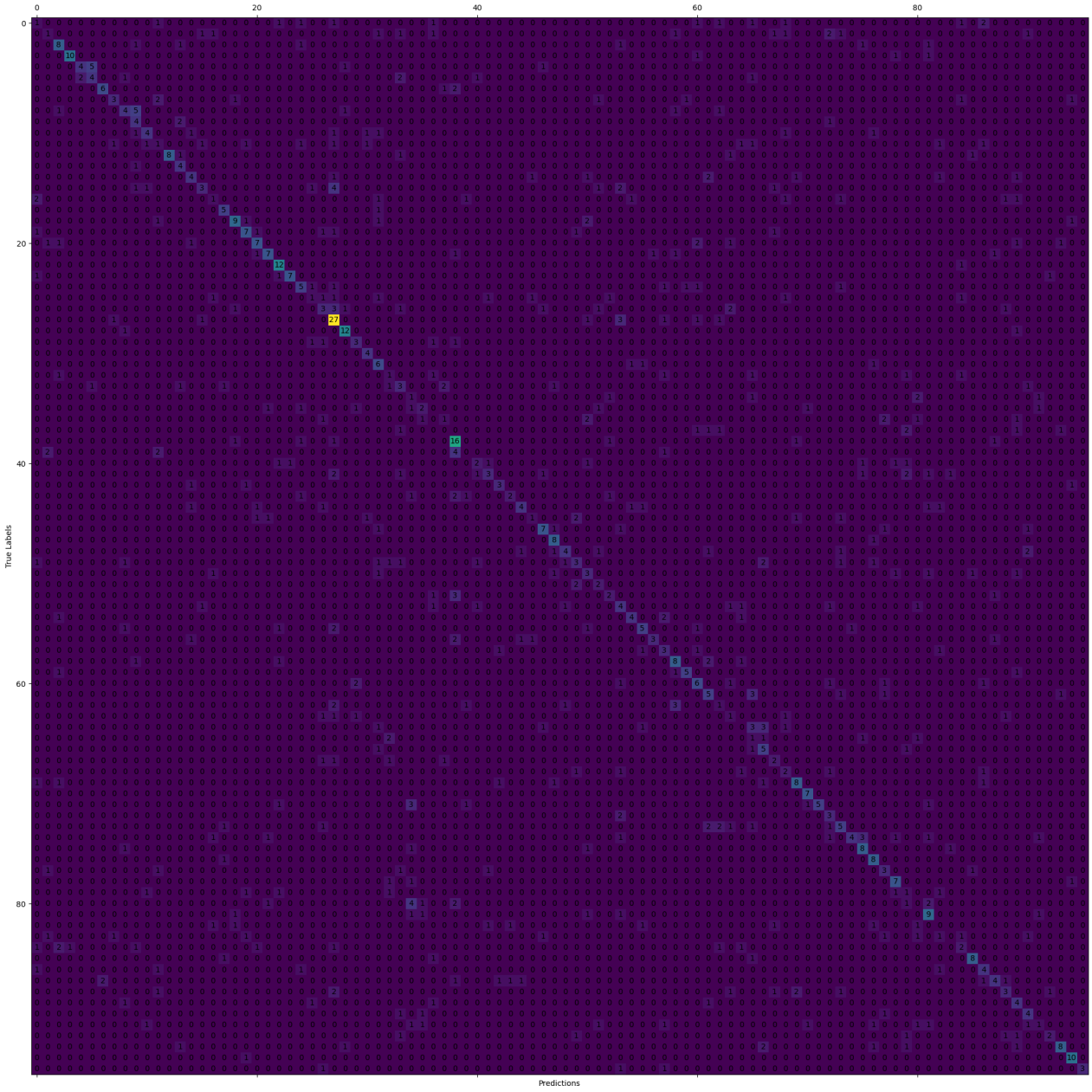


Figure 2: Matricea de confuzie pentru predicțiile date de clasificatorul One-Vs-Rest cu estimator SVM cu C=10 si kernel="rbf" și gamma=default antrenat și validat pe imagini micșorate la 32x32.

4.1.3 Evaluare

Cu modelul de One-vs-Rest cu clasificatoare de tip SVM cu C = 10 și kernel = "rbf", am obținut pe setul de validare acuratețea de 42.8%, iar pe setul de test privat 42.33%.

classes	precision	recall	f1-score	support
0	0.11	0.08	0.09	13
1	0.17	0.08	0.11	13
2	0.50	0.57	0.53	14
3	0.91	0.77	0.83	13
4	0.67	0.36	0.47	11
5	0.40	0.36	0.38	11
6	0.75	0.67	0.71	9
7	0.50	0.30	0.37	10
8	0.40	0.31	0.35	13
9	0.27	0.57	0.36	7
10	0.50	0.36	0.42	11
11	0.10	0.08	0.09	12
12	1.00	0.67	0.80	12
13	0.40	0.80	0.53	5
14	0.44	0.33	0.38	12
15	0.43	0.21	0.29	14
16	0.17	0.10	0.12	10
17	0.56	0.83	0.67	6
18	0.64	0.60	0.62	15
19	0.54	0.54	0.54	13
20	0.58	0.47	0.52	15
21	0.64	0.58	0.61	12
22	0.63	0.92	0.75	13
23	0.88	0.70	0.78	10
24	0.42	0.45	0.43	11
25	0.20	0.10	0.13	10
26	0.25	0.21	0.23	14
27	0.51	0.75	0.61	36
28	0.75	0.86	0.80	14
29	0.43	0.43	0.43	7
30	0.57	1.00	0.73	4
31	0.35	0.67	0.46	9
32	0.12	0.14	0.13	7
33	0.21	0.27	0.24	11
34	0.07	0.17	0.10	6
35	0.29	0.22	0.25	9
36	0.00	0.00	0.00	9
37	0.00	0.00	0.00	8
38	0.47	0.73	0.57	22
39	0.00	0.00	0.00	9
40	0.33	0.22	0.27	9
41	0.43	0.23	0.30	13
42	0.50	0.50	0.50	6
43	0.50	0.25	0.33	8
44	0.57	0.36	0.44	11
45	0.25	0.12	0.17	8
46	0.58	0.58	0.58	12
47	0.62	0.89	0.73	9
48	0.57	0.36	0.44	11
49	0.30	0.21	0.25	14
50	0.23	0.30	0.26	10

51	0.25	0.50	0.33	4
52	0.33	0.29	0.31	7
53	0.22	0.33	0.27	12
54	0.57	0.50	0.53	8
55	0.50	0.42	0.45	12
56	0.75	0.33	0.46	9
57	0.27	0.43	0.33	7
58	0.53	0.62	0.57	13
59	0.71	0.62	0.67	8
60	0.40	0.46	0.43	13
61	0.36	0.45	0.40	11
62	0.12	0.12	0.12	8
63	0.11	0.17	0.13	6
64	0.00	0.00	0.00	10
65	0.07	0.17	0.10	6
66	0.36	0.62	0.45	8
67	0.29	0.33	0.31	6
68	0.22	0.29	0.25	7
69	0.62	0.57	0.59	14
70	0.70	1.00	0.82	7
71	1.00	0.42	0.59	12
72	0.27	0.60	0.37	5
73	0.38	0.36	0.37	14
74	0.80	0.29	0.42	14
75	0.53	0.67	0.59	12
76	0.53	0.89	0.67	9
77	0.30	0.38	0.33	8
78	0.58	0.70	0.64	10
79	0.09	0.14	0.11	7
80	0.00	0.00	0.00	11
81	0.47	0.60	0.53	15
82	0.00	0.00	0.00	8
83	0.00	0.00	0.00	9
84	0.25	0.18	0.21	11
85	0.73	0.73	0.73	11
86	0.36	0.50	0.42	8
87	0.50	0.33	0.40	12
88	0.38	0.25	0.30	12
89	0.29	0.50	0.36	8
90	0.36	0.57	0.44	7
91	0.20	0.09	0.13	11
92	0.50	0.29	0.36	7
93	0.73	0.53	0.62	15
94	0.71	0.91	0.80	11
95	1.00	0.50	0.67	6
accuracy			0.43	1000
macro avg	0.42	0.42	0.40	1000
weighted avg	0.44	0.43	0.42	1000

4.2 Rețea neurală convoluțională

Pentru a construi o rețea neurală convoluțională, am încercat să analizez diversele arhitecturi și să înțeleg cum funcționează și cum se pot implementa cu ajutorul *pytorch*. Am analizat arhitectura rețelelor ResNet și, după un lung proces de try-and-error, am realizat o arhitectură mixtă (nu pur inspirată din ResNet).

4.2.1 Arhitectura

Blocuri

1. Bloc 1
   - Strat convolutional cu 64 filtre de 5x5, stride de 2 si padding de 2
   - Normalizare pe batch
   - Activare relu
   - Strat de pooling (max-pooling) cu kernelul de 3x3, stride 2 si padding 1
2. Bloc 2-4
   - Subbloc de bază (Bloc "rezidual")
     - strat convolutional cu f filtre 3x3, stride x, padding 1
     - strat convolutional cu f filtre 1x1, stride 1
     - normalizare dupa fiecare convoluție, adăugarea inputului modificat (modificat pentru primul subbloc din bloc) printr-un strat convolutional cu f filtre 3x3 cu stride x și padding 1 + normalizare input
     - activare relu după normalizare și adăugarea inputului
   - Bloc 2: 2 subblocuri de bază cu f = 64 filtre pe fiecare strat convoluțional (4 straturi convoluționale) și stride x = 1
   - Bloc 3 : 4 subblocuri de bază cu câte f = 256 de filtre pe fiecare strat convoluțional (8 straturi convoluționale) și stride x = 2
   - Bloc 4 : 1 subbloc de bază cu câte f = 512 de filtre pe fiecare strat convoluțional (2 straturi convoluționale) și stride x = 2
3. Strat final - complet conectat compus din :
   - strat de average pooling adaptiv
   - strat care face flatten (transformarea din 4 dimensiuni în 2)
   - strat de dropout (20%)
   - strat liniar cu 96 neuroni

4.2.2 Preprocesarea imaginilor și augumentarea

Folosesc imaginile ca atare, încărcate sub formă de tensori în dimensiunea 3x64x64 cu ajutorul clasei extinse de mine din clasa existentă *Dataset* . Privind imaginile, am decis că pentru cele mai multe din încercări să existe doar o transformare random cu probabilitate de 0.5 de tip flip orizontal. Am făcut un test și am adăugat și transformarea ColorJitter, schimbând puțin luminozitatea și hue.

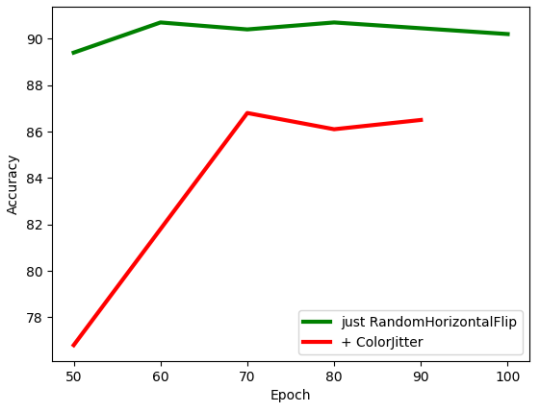


Figure 3: Comparație între modelul antrenat cu datele transformate random doar cu flip orizontal și modelul antrenat cu datele transformate cu flip orizontal (random) și schimbarea luminozității și/sau a saturației.

4.2.3 Hiperparametrii

Hiperparametrii sunt extrem de mulți în cazul rețelelor convoluționale neuronale. Eu am încercat să aduc modelul la o performanță bună prin ajustarea mai întâi (destul de aleatorie) a numărului de subblocuri și chiar de blocuri. Apoi am încercat mai multe variante pentru parametru momentum al algoritmului de updatare al weight-urilor (SGD) și am optat pentru una care convergea mai rapid (0.94), datorită avantajului că avea un timp mai scurt de antrenare. Pentru rata de învățare, am observat că depind destul de mult momentele (epocile) la care ea trebuie redusă pentru ca învățarea să poată continua (să nu diveargă modelul). Inițial (când încă nu ajustasem numărul de blocuri și subblocuri în varianta finală) am folosit MultiStepLr - care permitea reducerea ratei de învățare la epocile cu indicii înscriși într-un vector dat ca parametru acestui scheduler (MultiStepLr). Ulterior, am descoperit ReduceOnPlateau careia îi puteam da media pierderii calculată pe un batch și putea face o analiză, dacă pierderea a crescut, să aștepte 3 epoci (patience = 3) până când să reducă rata de învățare de 10 ori (factor = 0.1). Pentru algoritmul de optimizare, l-am folosit de la început pe cel de Stochastic Gradient Descent, inițial pe cel cu momentum, dar fără regularizare L2 (weight_decay), ulterior aplicând și regularizare. Mai jos atașez o comparație atât pe acuratețe, cât și pe pierdere modelului propus de mine în antrenarea cu SGD cu 2 momentumuri diferite. Numărul de epoci reprezintă un alt hiperparametru. Inițial, înainte de a folosi un număr mai scăzut de straturi, antrenam modelul meu pe 30-35 epoci și obțineam acuratețe cuprinsă între 70 și 80 %. Ulterior, am redus numărul de straturi și am introdus și reducerea ratei de învățare, astfel că antrenarea în mai multe epoci a fost posibilă și chiar necesară pentru a modela cât mai bine weight-urile. Pentru dimensiunea mini-batch-ului am ales 128.

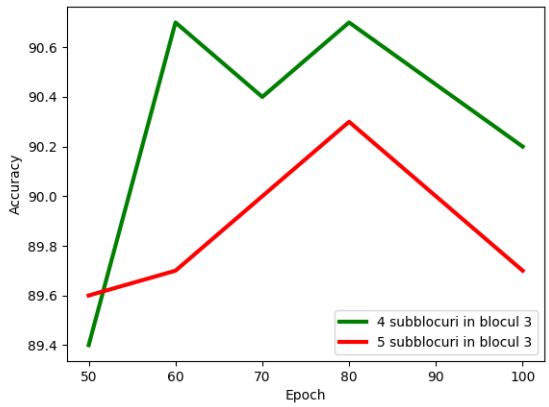


Figure 4: Comparație între modelul antrenat cu 4 subblocuri în blocul 3 și model antrenat cu 5 subblocuri în blocul 3.

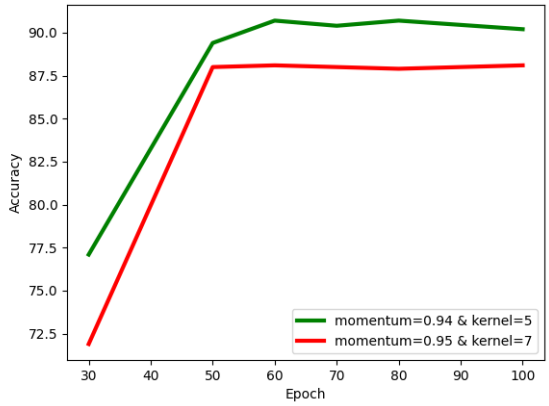
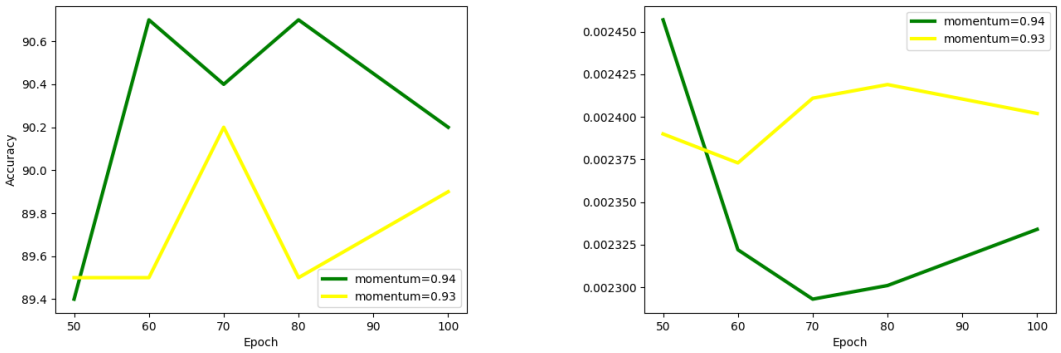


Figure 5: Comparație între modelul antrenat cu momentum 0.94 la SGD și kernel size 5 pentru primul bloc și modelul antrenat cu momentum 0.95 la SGD și kernel size 7 pentru primul bloc.



(a) Acuratețe

(b) Valoarea funcției de pierdere

Figure 6: Comparație între două antrenări ale modelului propus de mine cu SGD cu momentum diferit. Acuratețea și pierderea (loss-ul) sunt calculate pe setul de validare.

4.2.4 Regularizare

Din cauza numărului mare de parametrii (milioane) dat de numărul destul de mare de straturi și mai ales de filtre/strat și din cauza faptului că modelul meu converge destul de repede datorită și momentumului de 0.94, am decis să folosesc atât regularizarea L2, folosind parametrul weight_decay al SGD-ului, cât și regularizarea de tip dropout (nefolosită de obicei în rețele de tip ResNet).

4.2.5 Matricea de confuzie

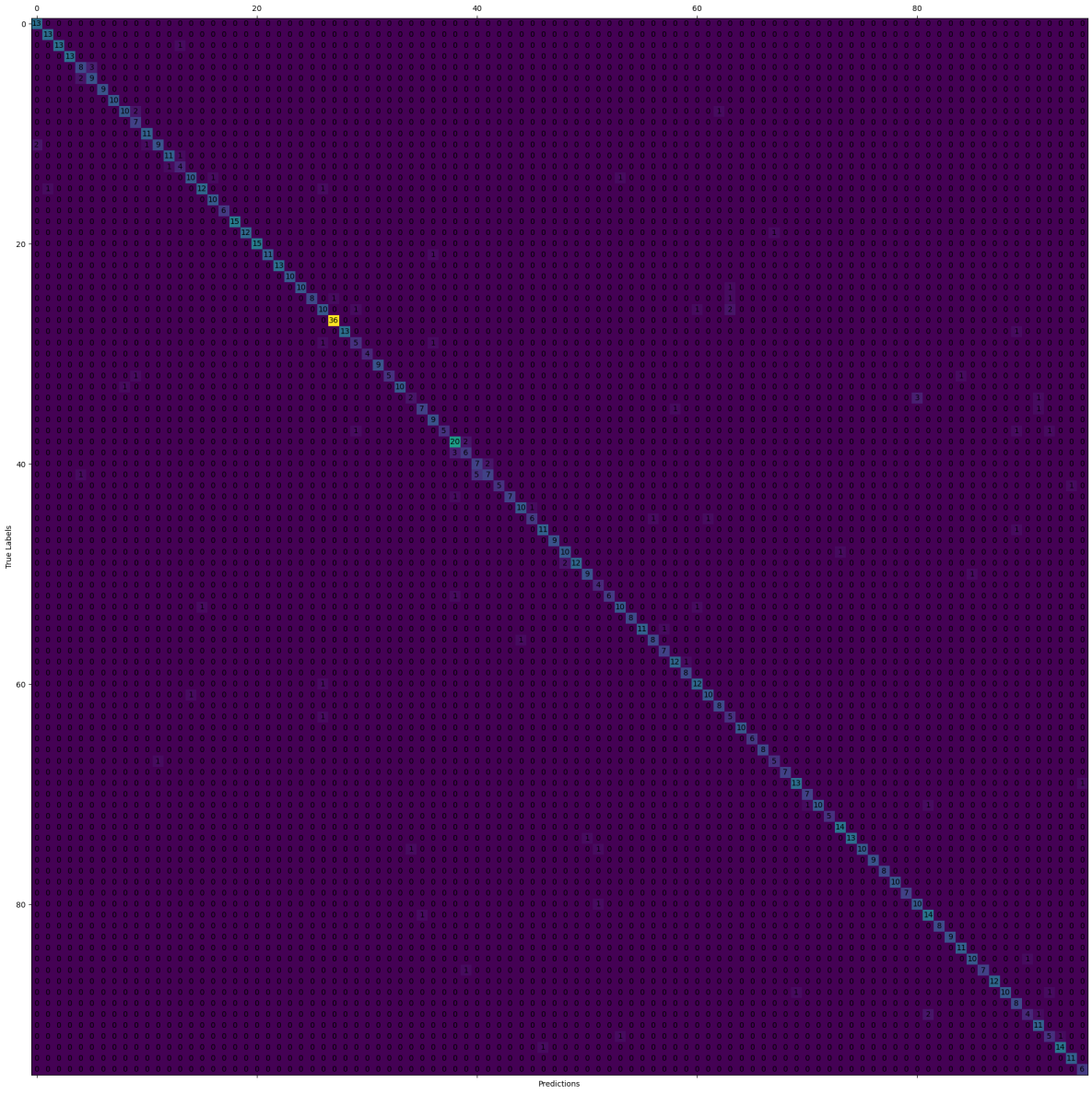


Figure 7: Matricea de confuzie pentru predicțiile date de rețeaua neurală convoluțională construită de mine, după antrenare timp de 60 de epoci.

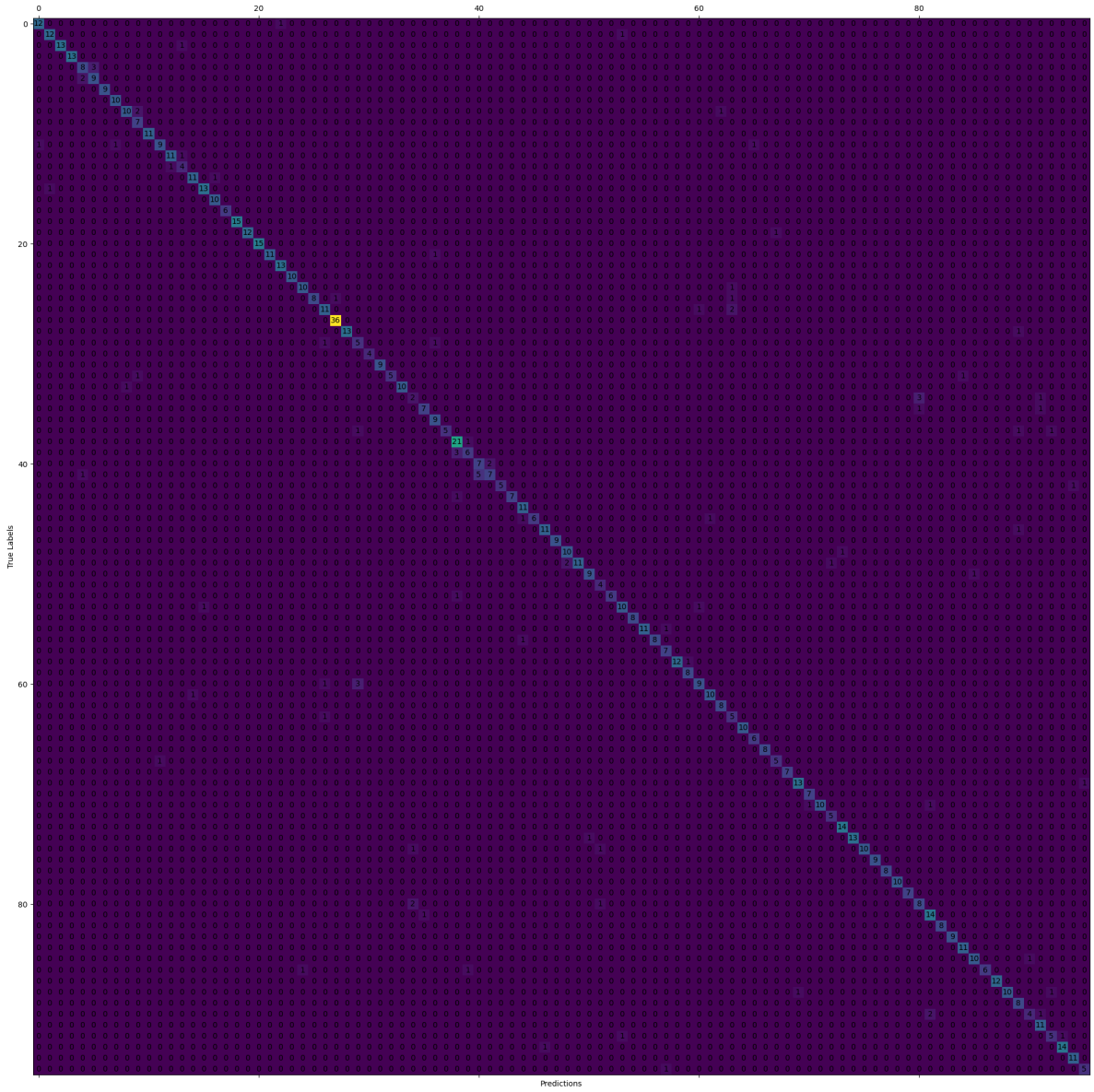


Figure 8: Matricea de confuzie pentru predicțiile date de rețeaua neurală convoluțională construită de mine, după antrenare timp de 100 de epoci.

4.2.6 Evaluare

Model antrenat 60 epoci- Acuratețe pe setul de validare 90.7, acuratețe pe setul de test privat 90.257

classes	precision	recall	f1-score	support
0	0.87	1.00	0.93	13
1	0.93	1.00	0.96	13
2	1.00	0.93	0.96	14
3	1.00	1.00	1.00	13
4	0.73	0.73	0.73	11
5	0.75	0.82	0.78	11
6	1.00	1.00	1.00	9
7	1.00	1.00	1.00	10
8	0.91	0.77	0.83	13
9	0.70	1.00	0.82	7
10	0.92	1.00	0.96	11
11	0.90	0.75	0.82	12
12	0.92	0.92	0.92	12
13	0.67	0.80	0.73	5
14	0.91	0.83	0.87	12
15	0.92	0.86	0.89	14
16	0.91	1.00	0.95	10
17	1.00	1.00	1.00	6
18	1.00	1.00	1.00	15
19	1.00	0.92	0.96	13
20	1.00	1.00	1.00	15
21	1.00	0.92	0.96	12
22	1.00	1.00	1.00	13
23	1.00	1.00	1.00	10
24	1.00	0.91	0.95	11
25	1.00	0.80	0.89	10
26	0.71	0.71	0.71	14
27	0.97	1.00	0.99	36
28	1.00	0.93	0.96	14
29	0.71	0.71	0.71	7
30	1.00	1.00	1.00	4
31	1.00	1.00	1.00	9
32	1.00	0.71	0.83	7
33	1.00	0.91	0.95	11
34	0.67	0.33	0.44	6
35	0.88	0.78	0.82	9
36	0.82	1.00	0.90	9
37	1.00	0.62	0.77	8
38	0.80	0.91	0.85	22
39	0.67	0.67	0.67	9
40	0.58	0.78	0.67	9
41	0.78	0.54	0.64	13
42	1.00	0.83	0.91	6
43	1.00	0.88	0.93	8
44	0.91	0.91	0.91	11
45	0.86	0.75	0.80	8
46	0.92	0.92	0.92	12
47	1.00	1.00	1.00	9
48	0.83	0.91	0.87	11
49	1.00	0.86	0.92	14
50	0.90	0.90	0.90	10

51	0.67	1.00	0.80	4
52	1.00	0.86	0.92	7
53	0.83	0.83	0.83	12
54	1.00	1.00	1.00	8
55	1.00	0.92	0.96	12
56	0.89	0.89	0.89	9
57	0.88	1.00	0.93	7
58	0.92	0.92	0.92	13
59	0.89	1.00	0.94	8
60	0.86	0.92	0.89	13
61	0.91	0.91	0.91	11
62	0.89	1.00	0.94	8
63	0.56	0.83	0.67	6
64	1.00	1.00	1.00	10
65	1.00	1.00	1.00	6
66	1.00	1.00	1.00	8
67	0.83	0.83	0.83	6
68	1.00	1.00	1.00	7
69	0.93	0.93	0.93	14
70	0.88	1.00	0.93	7
71	1.00	0.83	0.91	12
72	1.00	1.00	1.00	5
73	0.93	1.00	0.97	14
74	1.00	0.93	0.96	14
75	1.00	0.83	0.91	12
76	1.00	1.00	1.00	9
77	1.00	1.00	1.00	8
78	1.00	1.00	1.00	10
79	1.00	1.00	1.00	7
80	0.77	0.91	0.83	11
81	0.82	0.93	0.87	15
82	1.00	1.00	1.00	8
83	1.00	1.00	1.00	9
84	0.92	1.00	0.96	11
85	0.91	0.91	0.91	11
86	1.00	0.88	0.93	8
87	1.00	1.00	1.00	12
88	1.00	0.83	0.91	12
89	0.73	1.00	0.84	8
90	0.80	0.57	0.67	7
91	0.79	1.00	0.88	11
92	0.71	0.71	0.71	7
93	0.93	0.93	0.93	15
94	0.92	1.00	0.96	11
95	0.86	1.00	0.92	6
accuracy			0.91	1000
macro avg	0.91	0.90	0.90	1000
weighted avg	0.91	0.91	0.91	1000

Model antrenat 100 epoci- Acuratețe pe setul de validare 90.2, acuratețe pe setul de test privat 90.342

classes	precision	recall	f1-score	support
0	0.92	0.92	0.92	13
1	0.92	0.92	0.92	13
2	1.00	0.93	0.96	14
3	1.00	1.00	1.00	13
4	0.73	0.73	0.73	11
5	0.75	0.82	0.78	11
6	1.00	1.00	1.00	9
7	0.91	1.00	0.95	10
8	0.91	0.77	0.83	13
9	0.70	1.00	0.82	7
10	1.00	1.00	1.00	11
11	0.90	0.75	0.82	12
12	0.92	0.92	0.92	12
13	0.67	0.80	0.73	5
14	0.92	0.92	0.92	12
15	0.93	0.93	0.93	14
16	0.91	1.00	0.95	10
17	1.00	1.00	1.00	6
18	1.00	1.00	1.00	15
19	1.00	0.92	0.96	13
20	1.00	1.00	1.00	15
21	1.00	0.92	0.96	12
22	0.93	1.00	0.96	13
23	1.00	1.00	1.00	10
24	0.91	0.91	0.91	11
25	1.00	0.80	0.89	10
26	0.79	0.79	0.79	14
27	0.97	1.00	0.99	36
28	1.00	0.93	0.96	14
29	0.56	0.71	0.63	7
30	1.00	1.00	1.00	4
31	1.00	1.00	1.00	9
32	1.00	0.71	0.83	7
33	1.00	0.91	0.95	11
34	0.40	0.33	0.36	6
35	0.88	0.78	0.82	9
36	0.82	1.00	0.90	9
37	1.00	0.62	0.77	8
38	0.81	0.95	0.88	22
39	0.75	0.67	0.71	9
40	0.58	0.78	0.67	9
41	0.78	0.54	0.64	13
42	1.00	0.83	0.91	6
43	1.00	0.88	0.93	8
44	0.85	1.00	0.92	11
45	1.00	0.75	0.86	8
46	0.92	0.92	0.92	12
47	1.00	1.00	1.00	9
48	0.83	0.91	0.87	11
49	1.00	0.79	0.88	14
50	0.90	0.90	0.90	10
51	0.67	1.00	0.80	4
52	1.00	0.86	0.92	7

53	0.83	0.83	0.83	12
54	1.00	1.00	1.00	8
55	1.00	0.92	0.96	12
56	1.00	0.89	0.94	9
57	0.78	1.00	0.88	7
58	1.00	0.92	0.96	13
59	0.89	1.00	0.94	8
60	0.82	0.69	0.75	13
61	0.91	0.91	0.91	11
62	0.89	1.00	0.94	8
63	0.56	0.83	0.67	6
64	1.00	1.00	1.00	10
65	0.86	1.00	0.92	6
66	1.00	1.00	1.00	8
67	0.83	0.83	0.83	6
68	1.00	1.00	1.00	7
69	0.93	0.93	0.93	14
70	0.88	1.00	0.93	7
71	1.00	0.83	0.91	12
72	0.83	1.00	0.91	5
73	0.93	1.00	0.97	14
74	1.00	0.93	0.96	14
75	1.00	0.83	0.91	12
76	1.00	1.00	1.00	9
77	1.00	1.00	1.00	8
78	1.00	1.00	1.00	10
79	1.00	1.00	1.00	7
80	0.67	0.73	0.70	11
81	0.82	0.93	0.87	15
82	1.00	1.00	1.00	8
83	1.00	1.00	1.00	9
84	0.92	1.00	0.96	11
85	0.91	0.91	0.91	11
86	1.00	0.75	0.86	8
87	1.00	1.00	1.00	12
88	1.00	0.83	0.91	12
89	0.73	1.00	0.84	8
90	0.80	0.57	0.67	7
91	0.79	1.00	0.88	11
92	0.71	0.71	0.71	7
93	0.93	0.93	0.93	15
94	0.92	1.00	0.96	11
95	0.83	0.83	0.83	6
accuracy			0.90	1000
macro avg	0.90	0.90	0.89	1000
weighted avg	0.91	0.90	0.90	1000

5 Concluzii

Modelele propuse de mine rezolvă probloma de clasificare propusă, ambele obținând un scor mai mare de *baseline*. La realizarea rețelei neuronale, contribuind mulți hiperparametrii, procesul a fost de încărcări de combinații aleatorii de hiperparametrii, cu stabilizarea unora dintre ei pe măsură ce mă acomodam cu modul lor de funcționare.