Documentație proiect ML

Stan Cătălin-Andrei
Grupa 252

Cuprins

1 K-nearest neighbors

1.1 Citirea, preprocesarea și augmentarea datelor

Am folosit biblioteca *pandas* pentru a citi fișierele de tip csv. Pentru a citi imaginile am folosit *PIL* din biblioteca *Image*. Am folosit *StandardScaler* pentru a scala imaginile de antrenare.

1.2 Model

Modelul ales este *KNeighborsClassifier* din bliblioteca *sklearn* care urmatorii hiperparametrii:

n_neighbors
: numărul de vecini pe care îi ia în considerare

weights
: pentru a acorda ponderi in funcție de distanță

p
: din distanța Chebyshev

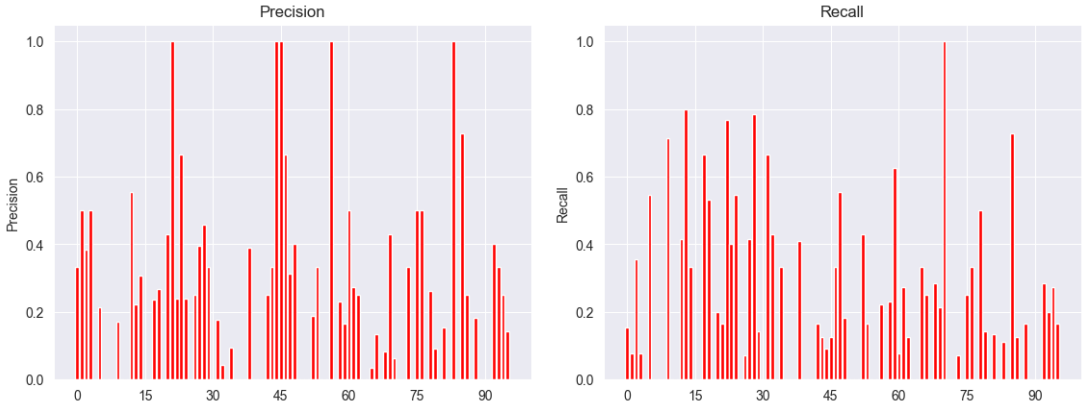
metric
: funcția de distanță

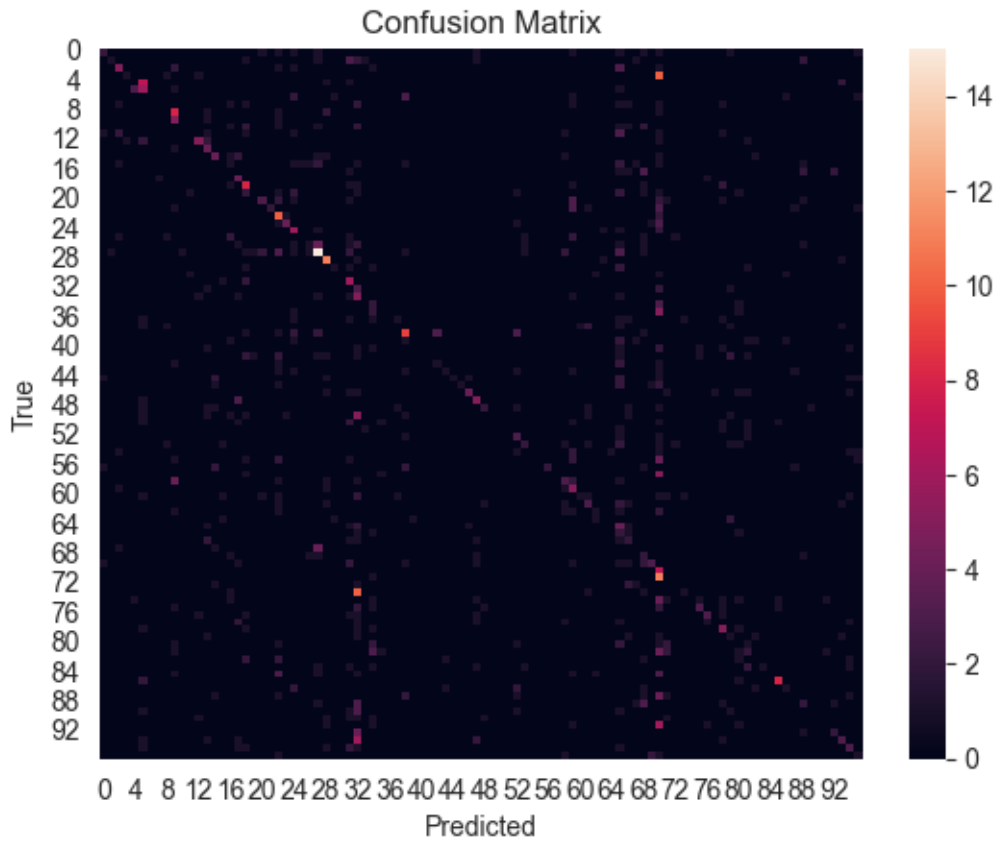
1.3 Tunarea hiperparametrilor

Am încercat toate combinațiile posibile dintre numărul de vecini din $\{1, 3, 5, 7, 11, 25, 55, 155, 255, 555\}$, ponderile $\{uniform, distance\}$, distanțele $\{cosine, l1, l2\}$, obținând următoarele rezultate Tabel kNN.

1.4 Cel mai bun model

Se observă că cel mai bun model s-a obținut pentru n = 25, w = distance și m = cosine.





2 Naïve Bayes

2.1 Citirea, preprocesarea și augmentarea datelor

Am folosit aceeași citire ca la modelul KNN. Pentru a folosi modelul MultinomialNB am folosit funcția *values_to_bins* (preluată din laborator) pentru a categorisi fiecare pixel.

2.2 Model

Modelul ales este *MultinomialNB* care are următorii hiperparametrii:

alpha
: parametru aditiv care este folosit în cazul probabilităților egale cu 0 care ar anula toată inmulțirea

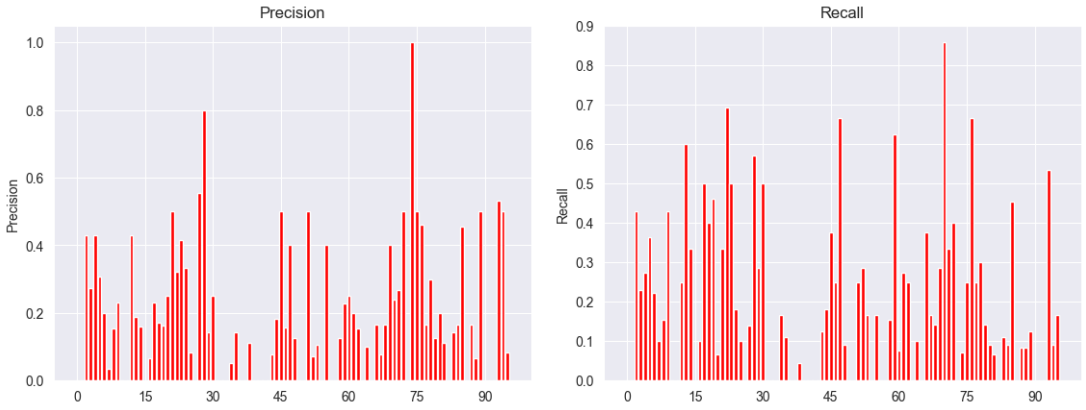
bins
: în câte categorii categorisesc fiecare pixel

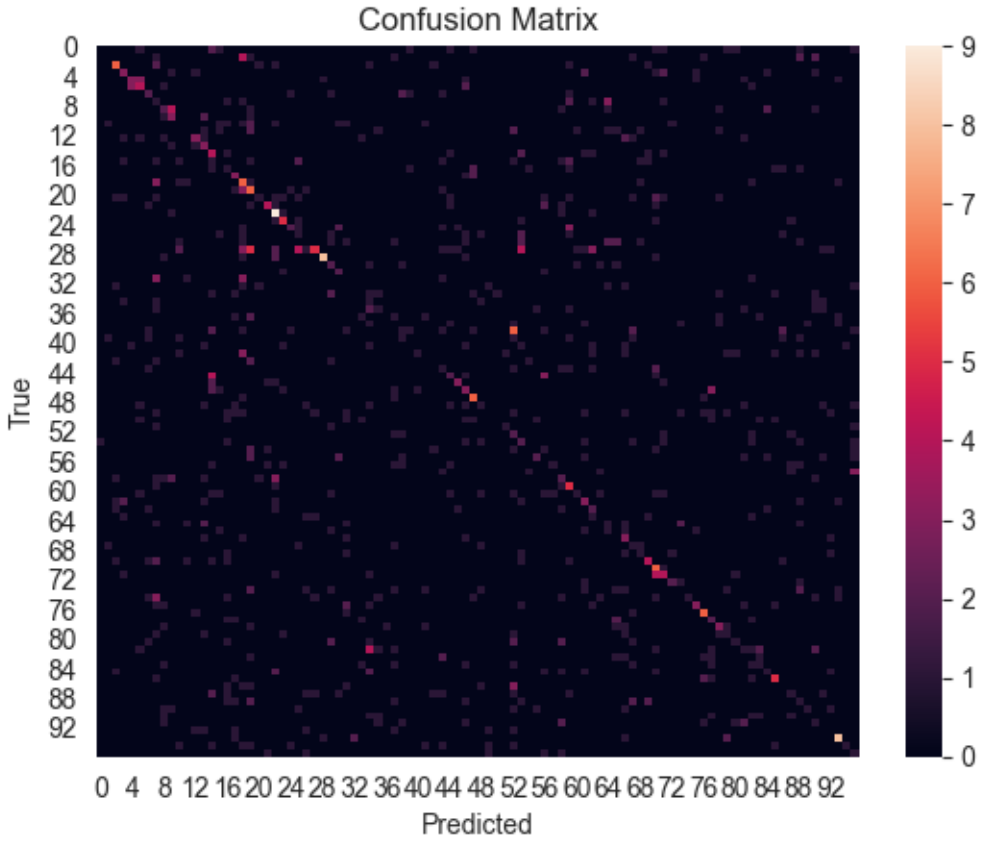
2.3 Tunarea hiperparametrilor

Am încercat toate valorile pentru bins din mulțimea $\{3, 5, 7, 15, 25, 35, 45, 85, 125, 175, 200, 225, 255\}$, iar pentru alpha am folosit valorile $\{0, 0.1, 0.01, 0.001, 0.5, 1\}$, obținând următoarele rezultate Tabel NB.

2.4 Cel mai bun model

Se observă că cel mai bun model s-a obținut pentru bins=25 și orice alpha din $\{0, 0.1, 0.01, 0.001, 0.5, 1\}$.





3 Neural Network

În continuare, pentru rețelele neuronale voi folosi biblioteca *pytorch* pentru a antrena modele pe GPU, iar procesul de citire, preprocesare și augmentarea datelor rămâne același.

3.1 Citirea, preprocesarea și augmentarea datelor

Am folosit clasa SetDate care moștenește clasa Dataset și implementează funcția *getitem* care ia o imagine, aplică transformările și o returnează. Folosesc *transforms.Compose* pentru a înlănțui mai multe transformări. Pentru augmentare am ales translații verticale și orizontale random cât și oglindiri orizontale random, iar la final am normalizat fiecare imagine, respectiv fiecare canal RGB aducând fiecare valoare în intervalul [-1, 1]. Deoarece funcția *ToTensor()* transformă implicit fiecare valoare în

intervalul [0, 1], pentru o valoare oarecare x, trebuie aplicată formula de normalizare:

$$\frac{x - \mu}{\sigma}$$

pentru

$$\mu = 0.5$$

și

$$\sigma = 0.5$$

Funcțiile *train_method* și *test_method* le-am preluat din laborator pe care le-am modificat pentru a vedea la fiecare epocă valoarea funcției de loss pe mulțimile de antrenare și validare cât și acuratețea pe cele două mulțimi.

Pentru fiecare rețea neuronală layer-ul de input o să aibă $64 \times 64 \times 3 = 12288$ neuroni, iar layer-ul de output o să aibă 96, deoarece sunt 96 de clase. Funcția pentru loss aleasă este *CrossEntropyLoss* deoarece măsoară eficient diferența dintre probabilitățile prezise ale claselor și etichetele reale ale claselor. Am folosit optimizatorul Adam cu learning rate 0.001.

3.2 Arhitecturi încercate

NN 1: un strat ascuns de dimensiune 128, funcția de activare ReLu

Epoch	Train Loss	Train Accuracy	Validation Accuracy
0	3.864	11.2%	12.6%
1	3.483	15.7%	16.3%
2	3.502	17.5%	15.1%
3	3.210	20.9%	17.8%
4	3.149	22.3%	18.9%
5	3.147	23.6%	21.4%
6	2.950	24.9%	20.7%
7	3.008	26.6%	23.9%
8	2.869	27.3%	20.4%
9	2.540	28.1%	23.8%
10	2.999	29.2%	24.1%
11	2.981	30.0%	24.7%
12	2.978	31.2%	26.4%
13	2.460	31.1%	26.8%
14	2.824	31.9%	26.0%
15	2.678	32.6%	27.2%
16	2.859	33.2%	27.3%
17	2.343	33.4%	24.0%
18	2.382	33.9%	27.9%
19	2.583	35.0%	27.3%

NN 2: un strat ascuns de dimensiune 128, funcția de activare LeakyReLu

Epoch	Train Loss	Train Accuracy	Validation Accuracy
0	4.051	12.1%	12.5%
1	3.388	15.8%	15.4%
2	3.137	18.0%	17.6%
3	3.263	20.3%	19.1%
4	3.160	21.0%	18.3%
5	2.942	24.0%	20.6%
6	3.162	25.1%	21.3%
7	2.855	26.5%	23.0%
8	2.917	27.2%	22.9%
9	2.739	27.8%	24.5%
10	2.946	28.8%	26.0%
11	2.849	29.1%	25.5%
12	2.805	30.4%	25.8%
13	2.830	29.9%	26.2%
14	2.518	30.8%	24.7%
15	2.629	31.5%	26.3%
16	2.625	31.5%	23.5%
17	2.911	31.6%	23.4%
18	3.113	32.2%	25.6%
19	2.278	31.9%	24.6%

NN 3: un start ascuns de dimensiune 256, funcția de activare ReLu

Epoch	Train Loss	Train Accuracy	Validation Accuracy
0	3.490	13.4%	15.0%
1	3.776	17.7%	17.1%
2	3.279	21.2%	20.9%
3	3.181	23.1%	19.1%
4	3.116	24.6%	22.1%
5	3.111	26.3%	21.4%
6	2.935	27.1%	20.5%
7	2.778	28.2%	23.0%
8	2.755	29.4%	23.8%
9	2.527	30.8%	23.1%
10	2.954	31.3%	23.1%
11	2.959	31.8%	21.8%
12	2.763	31.4%	24.7%
13	2.724	32.5%	24.4%
14	2.565	32.5%	23.0%
15	2.689	33.4%	24.7%
16	2.472	34.4%	23.3%
17	2.310	34.3%	26.0%
18	2.904	34.0%	22.6%
19	2.558	35.4%	24.6%

NN 4: un strat ascuns de dimensiune 256, funcția de activare LeakyReLu

Epoch	Train Loss	Train Accuracy	Validation Accuracy
0	3.600	13.6%	15.3%
1	3.379	17.1%	15.9%
2	3.244	20.0%	17.9%
3	3.077	22.2%	20.8%
4	3.020	23.8%	21.2%
5	2.680	25.8%	22.7%
6	3.026	27.8%	21.5%
7	2.907	28.8%	22.0%
8	2.812	29.6%	21.3%
9	2.624	30.5%	23.1%
10	2.878	31.3%	23.6%
11	2.599	32.8%	25.4%
12	2.955	32.5%	22.7%
13	2.697	32.7%	23.4%
14	2.684	32.3%	23.9%
15	2.685	32.9%	24.3%
16	2.658	33.8%	24.4%
17	2.668	34.9%	26.1%
18	2.186	35.1%	25.4%
19	2.577	35.5%	26.3%

NN 5: două straturi ascunse de dimensiune 128, funcția de activare ReLu

Epoch	Train Loss	Train Accuracy	Validation Accuracy
0	3.448	14.0%	14.5%
1	3.365	18.4%	19.6%
2	3.356	21.7%	21.0%
3	3.113	25.1%	24.4%
4	3.222	26.3%	23.2%
5	2.697	28.7%	24.3%
6	2.944	29.8%	25.5%
7	2.810	31.5%	25.7%
8	2.679	33.1%	26.6%
9	2.507	34.2%	26.8%
10	2.245	34.8%	24.5%
11	2.509	35.7%	25.8%
12	2.276	37.2%	25.7%
13	2.725	37.4%	26.6%
14	2.185	38.9%	27.8%
15	2.696	38.5%	27.0%
16	2.371	39.3%	27.8%
17	2.427	40.6%	29.6%
18	2.346	40.3%	28.2%
19	2.557	41.6%	29.9%

NN 6: două straturi ascunse de dimensiune 128, funcția de activare LeakyReLu

Epoch	Train Loss	Train Accuracy	Validation Accuracy
0	3.549	14.4%	14.7%
1	3.194	18.4%	19.2%
2	3.288	21.2%	20.8%
3	3.161	24.4%	21.9%
4	2.968	26.0%	22.7%
5	3.030	27.4%	23.0%
6	2.752	29.5%	21.8%
7	2.914	31.5%	26.0%
8	2.503	31.9%	26.0%
9	2.268	33.8%	24.9%
10	2.361	35.4%	26.6%
11	2.295	35.5%	26.5%
12	2.375	36.5%	27.9%
13	2.389	38.1%	28.1%
14	2.205	38.3%	28.8%
15	2.176	39.4%	27.5%
16	2.169	40.7%	29.7%
17	2.375	39.8%	27.6%
18	2.202	41.3%	28.9%
19	2.132	41.8%	28.2%

NN 7: două straturi ascunse de dimensiune 128, funcția de activare ReLu, droput de 0.25 pentru regularizare

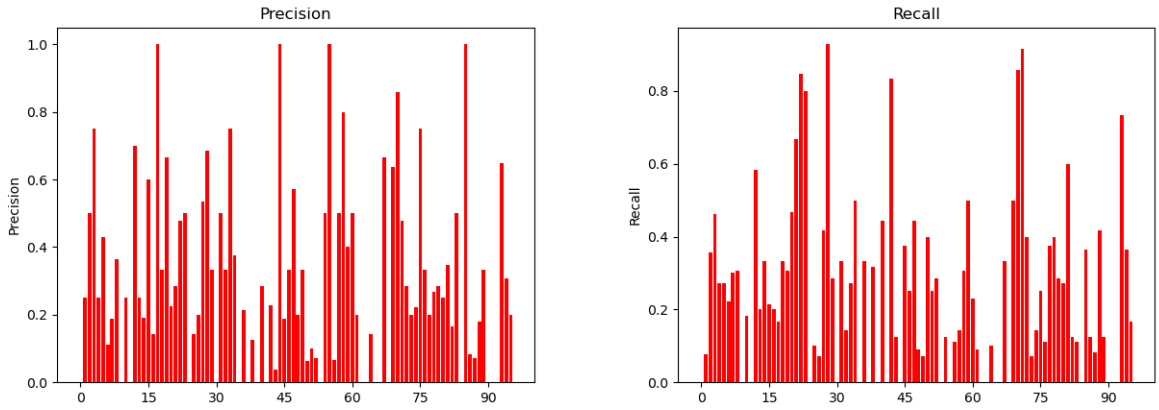
Epoch	Train Loss	Train Accuracy	Validation Accuracy
0	3.786	12.5%	12.0%
1	3.551	17.2%	17.3%
2	3.458	18.9%	20.1%
3	3.351	22.3%	20.1%
4	3.051	23.6%	20.0%
5	3.050	25.7%	21.6%
6	2.573	27.0%	21.2%
7	2.991	27.7%	22.8%
8	2.892	29.8%	24.6%
9	2.711	31.4%	23.7%
10	2.769	31.7%	24.3%
11	2.605	33.2%	23.3%
12	2.407	33.8%	24.6%
13	2.678	34.4%	24.8%
14	2.264	35.9%	27.2%
15	2.279	36.2%	27.4%
16	2.195	37.1%	26.5%
17	2.426	36.9%	28.0%
18	2.121	38.4%	27.2%
19	2.365	38.1%	25.7%

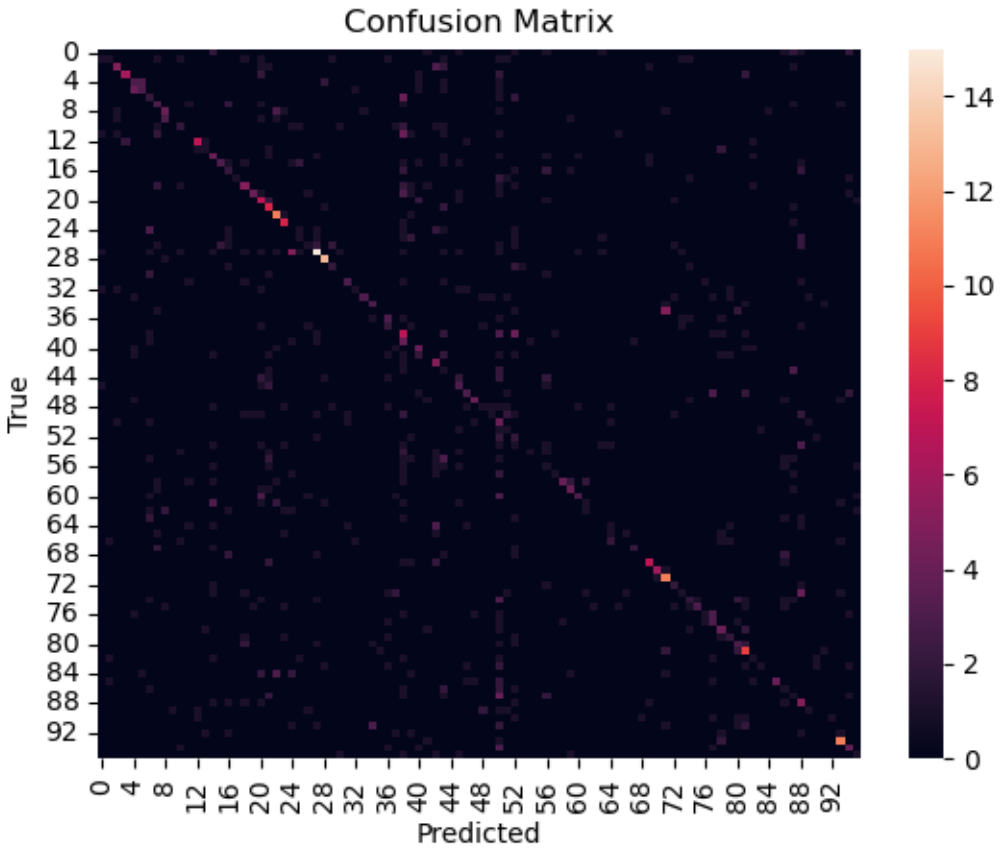
NN 8: două straturi ascunse de dimensiune 128, funcția activare LeakyReLu, droput de 0.25 pentru regularizare

Epoch	Train Loss	Train Accuracy	Validation Accuracy
0	3.706	13.3%	14.5%
1	3.468	17.7%	17.0%
2	3.241	22.4%	20.7%
3	2.870	24.7%	21.0%
4	3.069	25.3%	22.6%
5	3.192	27.2%	21.0%
6	2.673	30.2%	24.7%
7	2.649	29.8%	22.8%
8	2.592	31.7%	23.6%
9	2.515	33.6%	22.8%
10	2.399	34.3%	25.8%
11	2.460	34.9%	26.2%
12	2.225	35.0%	24.8%
13	2.440	36.5%	24.6%
14	2.117	35.7%	24.1%
15	2.238	37.4%	25.1%
16	2.142	37.5%	26.0%
17	2.178	38.4%	25.5%
18	2.372	37.8%	23.0%
19	1.981	39.2%	23.9%

3.3 Cel mai bun model

Se observă ca cel mai bun model este **NN5**.





4 Convolutional Neural Network

4.1 Arhitecturi încercate

CNN 1: am folosit un strat convoluțional cu kernel = 3, stride = 1 și 64 canale de ieșire, urmat de normalizarea ieșirilor și aplicarea unui strat de MaxPool de dimensiune 2. Urmează un strat fully connected de 128 neuroni și un strat de ieșire de 96 neuroni.

Epoch	Train Loss	Train Accuracy	Validation Accuracy
0	3.257	24.5%	25.3%
1	2.264	40.6%	44.3%
2	2.016	46.7%	50.5%
3	1.809	53.1%	56.9%
4	1.595	55.8%	58.4%
5	1.353	60.0%	64.4%
6	1.385	61.2%	65.2%
7	1.147	63.4%	68.2%
8	1.215	66.8%	68.1%
9	1.004	66.5%	67.7%
10	1.160	68.6%	66.6%
11	1.002	70.0%	69.5%
12	1.009	69.9%	69.9%
13	1.173	73.0%	72.5%
14	0.893	72.5%	71.4%
15	0.953	72.2%	71.2%
16	0.873	74.6%	73.4%
17	0.824	75.0%	75.1%
18	0.813	74.9%	75.3%
19	0.727	75.6%	74.0%

CNN 2: o arhitectură similară cu cea precedentă doar ca setat dimensiunea kernelului 7 la startul convoluțional

Epoch	Train Loss	Train Accuracy	Validation Accuracy
0	3.073	25.1%	28.6%
1	2.423	40.0%	42.2%
2	2.302	45.7%	48.8%
3	1.683	47.7%	51.3%
4	1.521	54.3%	56.3%
5	1.543	57.0%	61.3%
6	1.410	60.8%	59.8%
7	1.420	60.7%	63.0%
8	1.139	65.0%	66.3%
9	1.227	64.3%	66.5%
10	1.271	66.4%	67.5%
11	1.153	67.8%	69.0%
12	0.976	69.3%	70.4%
13	1.172	69.4%	69.9%
14	1.042	68.3%	70.3%
15	0.909	72.2%	70.8%
16	1.074	72.8%	71.9%
17	0.803	71.6%	71.0%
18	0.748	72.3%	69.9%
19	0.970	73.1%	72.2%

CNN 3: 2 straturi convoluționale, fiecare cu 64 canale de ieșire si dimensiunea filtrului 3. Între fiecare am aplicat normalizare si la final un strat de MaxPool. Urmează un strat fully connected de 128 neuroni și un strat de ieșire de 96 neuroni.

Epoch	Train Loss	Train Accuracy	Validation Accuracy
0	3.086	28.3%	30.3%
1	1.925	43.9%	48.4%
2	1.685	51.1%	55.6%
3	1.443	58.5%	62.1%
4	1.266	60.1%	62.2%
5	1.259	65.4%	68.0%
6	1.028	68.0%	71.1%
7	0.999	68.1%	70.3%
8	1.104	73.1%	72.9%
9	0.896	72.9%	73.9%
10	0.798	74.0%	74.0%
11	0.849	74.2%	72.7%
12	0.759	76.0%	75.8%
13	0.921	76.1%	76.2%
14	0.846	78.1%	75.9%
15	0.688	75.6%	75.3%
16	0.552	77.3%	77.0%
17	0.688	79.8%	77.0%
18	0.822	78.4%	75.7%
19	0.716	78.3%	78.0%

CNN 4: o arhitectură similară cu cea precedentă doar că am schimbat kernelul primului start convoluțional in dimensiune 7, dimensiune 5 la al doilea și stride = 2

Epoch	Train Loss	Train Accuracy	Validation Accuracy
0	3.125	27.6%	28.1%
1	2.189	41.3%	45.8%
2	1.926	50.1%	51.7%
3	1.522	56.5%	59.3%
4	1.356	60.3%	64.3%
5	1.417	62.9%	66.1%
6	1.440	65.7%	68.4%
7	1.545	68.6%	70.3%
8	0.925	68.8%	67.7%
9	1.048	70.1%	70.5%
10	0.820	70.5%	69.9%
11	1.095	71.8%	72.6%
12	0.869	72.4%	70.8%
13	0.882	73.8%	75.1%
14	0.688	74.5%	73.6%
15	0.836	75.0%	72.3%
16	0.682	74.9%	74.3%
17	0.913	77.2%	74.1%
18	1.014	77.0%	73.6%
19	0.646	77.4%	74.7%

CNN 5: două straturi convoluționale , dar intre fiecare am pus un strat de MaxPool de dimensiune 2

Epoch	Train Loss	Train Accuracy	Validation Accuracy
0	2.492	38.6%	42.8%
1	1.989	53.3%	57.6%
2	1.299	61.4%	65.0%
3	0.923	66.2%	67.1%
4	1.083	66.9%	67.5%
5	0.997	70.7%	71.3%
6	0.770	73.4%	71.8%
7	1.019	73.7%	72.5%
8	1.082	75.7%	73.6%
9	0.705	76.4%	74.8%
10	0.699	78.8%	74.9%
11	0.822	78.5%	76.8%
12	0.686	79.2%	76.7%
13	0.847	79.8%	76.2%
14	0.433	78.5%	75.2%
15	0.489	81.8%	78.5%
16	0.442	83.5%	80.2%
17	0.459	83.0%	76.5%
18	0.571	85.0%	80.1%
19	0.520	84.0%	77.4%

CNN 6: trei straturi convoluționale (toate au 64 canale ieșire, iar dimensiunea filtrelor este 11, 5, respectiv 3) , dar intre fiecare am pus un strat de MaxPool de dimensiune 2

Epoch	Train Loss	Train Accuracy	Validation Accuracy
0	2.713	26.6%	30.6%
1	1.991	38.5%	43.3%
2	2.045	44.8%	49.0%
3	1.737	52.6%	52.8%
4	1.440	55.5%	54.8%
5	1.505	58.2%	57.1%
6	1.372	60.9%	58.7%
7	1.250	62.0%	58.2%
8	0.919	65.3%	61.3%
9	1.262	65.9%	61.1%
10	1.036	66.4%	63.7%
11	1.163	66.9%	61.1%
12	1.215	68.8%	64.8%
13	1.052	71.2%	65.7%
14	0.796	71.3%	65.8%
15	1.103	71.9%	66.0%
16	0.984	74.0%	66.8%
17	0.706	72.8%	66.3%
18	0.832	74.5%	68.2%
19	1.037	73.4%	63.0%

CNN 7: arhitectură similară cu cea precedentă doar ca dimensiunea filtrelor este 3, iar canalele de ieșire sunt 64, 128, 256.

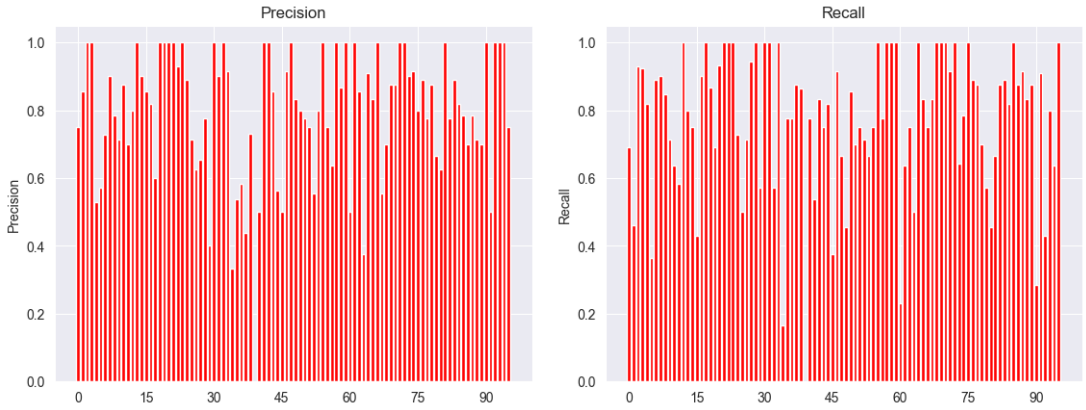
Epoch	Train Loss	Train Accuracy	Validation Accuracy
0	2.631	29.4%	28.4%
1	2.845	27.7%	29.4%
2	2.263	40.1%	39.3%
3	2.077	41.9%	42.7%
4	1.611	48.8%	47.5%
5	1.643	56.2%	57.1%
6	1.230	55.4%	55.0%
7	1.192	64.1%	61.7%
8	1.055	65.3%	62.2%
9	1.080	70.3%	67.1%
10	1.074	68.2%	65.1%
11	1.147	72.5%	69.3%
12	0.771	74.6%	70.9%
13	0.764	76.8%	71.8%
14	0.896	77.2%	71.3%
15	0.776	78.1%	71.0%
16	1.096	79.3%	73.4%
17	0.513	81.5%	74.5%
18	0.791	80.4%	76.6%
19	0.556	82.4%	76.9%

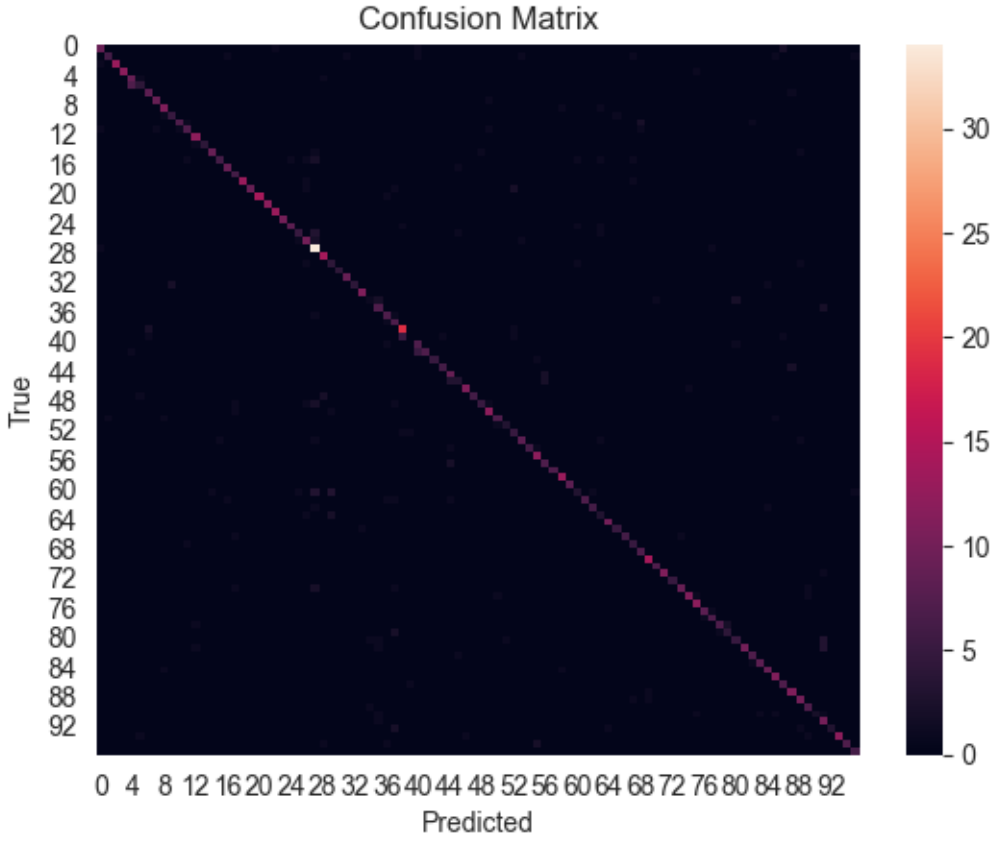
CNN 8: am păstrat arhitectura doar că înainte de ultimul start de ieșire cu 96 de neuroni am pus un dropout de 0.5 pentru regularizare

Epoch	Train Loss	Train Accuracy	Validation Accuracy
0	3.143	27.9%	28.2%
1	2.216	32.0%	34.6%
2	1.947	44.2%	44.3%
3	1.985	50.4%	51.3%
4	1.694	54.0%	54.5%
5	1.617	59.2%	58.4%
6	1.270	62.2%	62.7%
7	1.046	63.4%	64.6%
8	1.277	66.9%	66.5%
9	0.957	68.1%	66.6%
10	0.837	72.5%	70.7%
11	0.881	75.2%	72.1%
12	0.898	75.2%	71.1%
13	0.695	77.2%	73.0%
14	0.547	79.5%	75.6%
15	1.074	77.7%	70.8%
16	0.520	81.4%	77.4%
17	0.770	82.8%	78.5%
18	0.669	82.2%	78.2%
19	0.504	82.7%	75.5%

4.2 Cel mai bun model

Se observă că cel mai bun model este **CNN3**.





5 Residual Neural Network

5.1 Blocul rezidual

Structura de baza pentru rețeaua reziduală aste blocul rezidual parametrizat prin numărul de canale de intrare, numărul de canale de ieșire și stride-ul primului strat convoluțional. Am folosit dimensiunea kernelului 3. În cazul în care canalele de intrare diferă de canalele de ieșire sau stride-ul este diferit de 1 înseamna ca la final outputul o sa fie de dimensiuni diferite față de input. Cum acestea sunt adunate la final (definiția unui bloc rezidual) trebuie adăugat un strat convoluțional de kernel egal cu 1 și stride egal cu cel dat pentru a aduce inputul la aceleași dimensiuni. După fiecare strat convolutional am adăugat și normalizare.

5.2 Arhitecturi încercate

ResNet 1: un singur bloc rezidual cu 64 canale de iesire, un MaxPool de dimensiune 2 si un strat ascuns cu 128 neuroni. Dupa fiecare iesire aplic normalizare

Epoch	Train Loss	Train Accuracy	Validation Accuracy
0	2.885	30.0%	36.0
1	1.637	50.8%	54.1
2	1.348	54.3%	53.9
3	1.382	63.6%	63.5
4	0.976	66.8%	67.1
5	1.088	69.5%	71.8
6	0.910	72.0%	70.1
7	0.910	73.1%	71.4
8	0.928	75.3%	76.4
9	0.724	77.3%	77.5
10	0.650	78.4%	76.5

ResNet 2: un strat convolutional urmat de 2 blocuri reziduale cu 64 canale ieșire, un MaxPool de dimensiune , inca 2 blocuri reziduale cu 128 canale ieșire și un MaxPool de dimensiune 2 și stride 2, iar intre stratul de neuroni de input și output un strat ascuns de dimensiune 256.

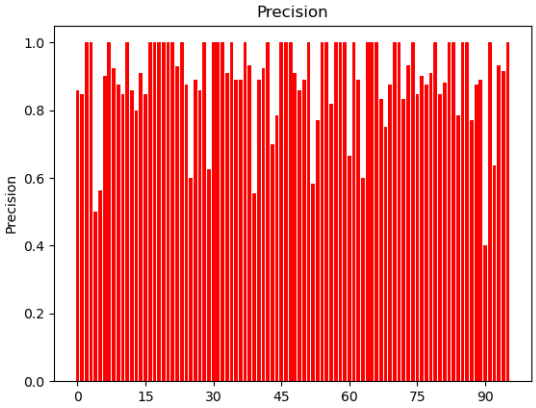
Epoch	Train Loss	Train Accuracy	Validation Accuracy
0	2.016	44.5%	48.3
1	2.443	40.5%	41.6
2	1.513	52.6%	56.8
3	1.307	55.1%	58.8
4	0.721	65.3%	69.3
5	1.248	68.2%	69.5
6	1.060	71.9%	71.6
7	0.798	74.4%	74.6
8	0.841	74.8%	72.9
9	0.787	76.6%	73.9
10	0.664	82.1%	80.2
11	0.585	83.6%	81.0
12	0.502	83.0%	79.1
13	0.646	83.2%	81.0
14	0.488	84.7%	81.2
15	0.499	85.3%	81.3
16	0.314	87.6%	83.9
17	0.412	87.3%	84.0
18	0.323	86.2%	82.2
19	0.340	89.2%	84.8

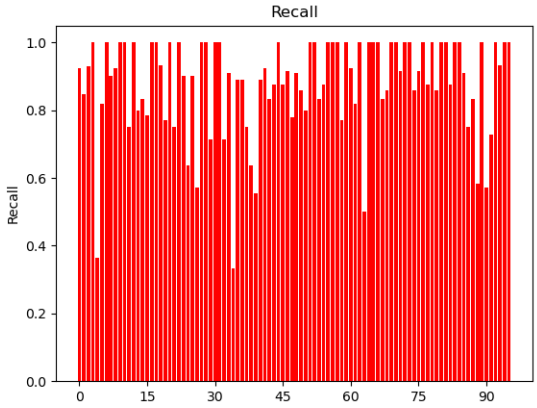
ResNet 3: un strat convolutional, urmat de normalizare, doua blocuri reziduale si un MaxPool, inca doua blocuri reziduale cu 128 canale iesire si un MaxPool, inca 2 blocuri reziduale cu 256 canale iesire si un MaxPool, iar intre stratul de neuroni de intrare si iesire un strat ascuns de 256 neuroni si un dropout pentru regularizare.

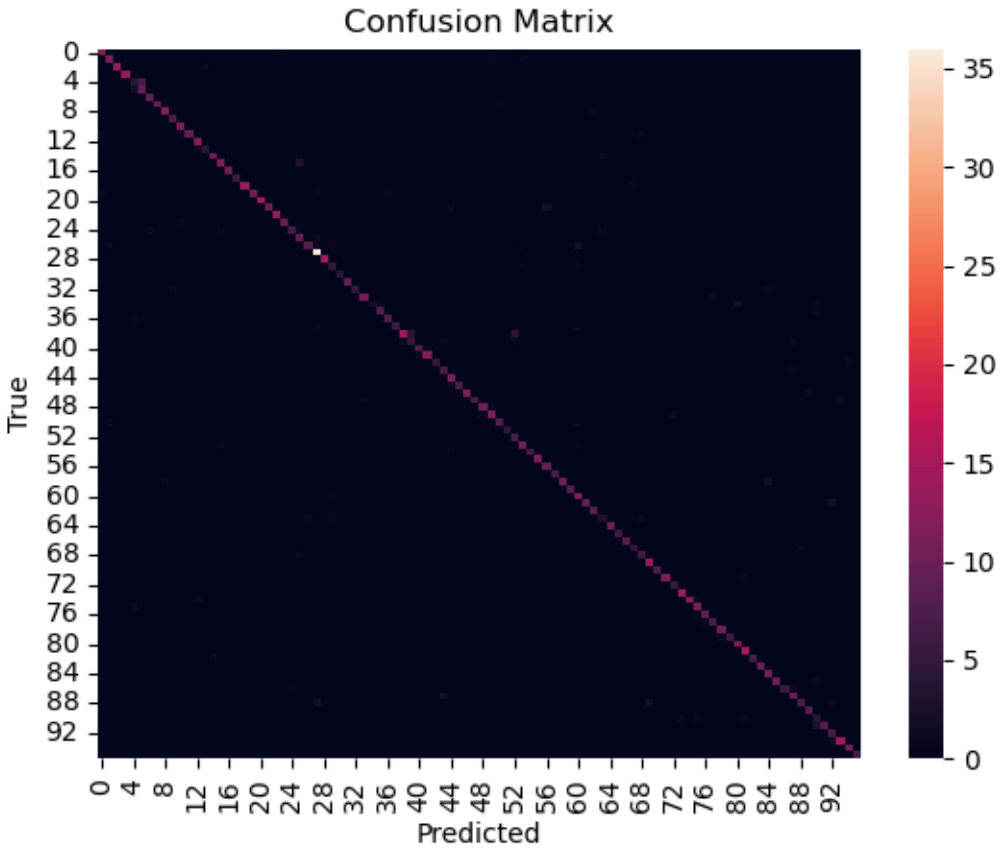
Epoch	Train Loss	Train Accuracy	Validation Accuracy
0	2.457	37.6%	41.7%
1	4.140	6.0%	7.1%
2	3.747	10.6%	11.6%
3	2.777	20.6%	20.6%
4	2.398	28.0%	30.7%
5	2.479	31.2%	32.8%
6	2.058	41.5%	42.8%
7	1.963	48.4%	48.2%
8	1.469	54.0%	55.8%
9	1.635	55.4%	57.4%
...	...	...	...
97	0.093	95.7%	89.6%
98	0.104	95.4%	86.2%
99	0.047	97.1%	88.4%
100	0.033	96.9%	88.1%

5.3 Cel mai bun model

Se observa ca cel mai bun model este **ResNet 3**, cu care am reusit sa iau submisia maxima.






Confusion Matrix
True
Predicted
0
4
8
12
16
20
24
28
32
36
40
44
48
52
56
60
64
68
72
76
80
84
88
92
35
30
25
20
15
10
5
0

6 Tabele

Tabela 1: kNN tunarea hiperparametrilor

n	w	p	m	Acuratetea
1	uniform	1	-	0.15
1	uniform	2	-	0.113
1	uniform	-	cosine	0.16
1	distance	1	-	0.15
1	distance	2	-	0.113
1	distance	-	cosine	0.16
3	uniform	1	-	0.141
3	uniform	2	-	0.112
3	uniform	-	cosine	0.183
3	distance	1	-	0.16
3	distance	2	-	0.116
3	distance	-	cosine	0.181
5	uniform	1	-	0.155
5	uniform	2	-	0.113
5	uniform	-	cosine	0.189
5	distance	1	-	0.161
5	distance	2	-	0.118
5	distance	-	cosine	0.197
7	uniform	1	-	0.165
7	uniform	2	-	0.118
7	uniform	-	cosine	0.203
7	distance	1	-	0.167
7	distance	2	-	0.116
7	distance	-	cosine	0.198
11	uniform	1	-	0.157
11	uniform	2	-	0.117
11	uniform	-	cosine	0.196
11	distance	1	-	0.164
11	distance	2	-	0.119
11	distance	-	cosine	0.21

n	w	p	m	Acuratetea
25	uniform	1	-	0.151
25	uniform	2	-	0.115
25	uniform	-	cosine	0.197
25	distance	1	-	0.159
25	distance	2	-	0.116
25	distance	-	cosine	0.205
55	uniform	1	-	0.142
55	uniform	2	-	0.102
55	uniform	-	cosine	0.189
55	distance	1	-	0.143
55	distance	2	-	0.102
55	distance	-	cosine	0.189
155	uniform	1	-	0.118
155	uniform	2	-	0.088
155	uniform	-	cosine	0.164
155	distance	1	-	0.118
155	distance	2	-	0.088
155	distance	-	cosine	0.173
355	uniform	1	-	0.097
355	uniform	2	-	0.076
355	uniform	-	cosine	0.156
355	distance	1	-	0.102
355	distance	2	-	0.079
355	distance	-	cosine	0.157
555	uniform	1	-	0.083
555	uniform	2	-	0.064
555	uniform	-	cosine	0.132
555	distance	1	-	0.092
555	distance	2	-	0.07
555	distance	-	cosine	0.145

Tabela 2: NB tunarea hiperparametrilor

bins	*alpha*	Acuratetea
3	0	0.016
3	0.1	0.149
3	0.01	0.151
3	0.001	0.151
3	0.5	0.149
3	1	0.15
5	0	0.179
5	0.1	0.179
5	0.01	0.179
5	0.001	0.179
5	0.5	0.179
5	1	0.177
7	0	0.184
7	0.1	0.184
7	0.01	0.184
7	0.001	0.184
7	0.5	0.184
7	1	0.184
15	0	0.186
15	0.1	0.186
15	0.01	0.186
15	0.001	0.186
15	0.5	0.186
15	1	0.186
25	0	0.188
25	0.1	0.188
25	0.01	0.188
25	0.001	0.188
25	0.5	0.188
25	1	0.188
35	0	0.188
35	0.1	0.188
35	0.01	0.188
35	0.001	0.188
35	0.5	0.188
35	1	0.188
45	0	0.186
45	0.1	0.186
45	0.01	0.186
45	0.001	0.186
45	0.5	0.186
45	1	0.186
85	0	0.184
85	0.1	0.184
85	0.01	0.184
85	0.001	0.184
85	0.5	0.184
85	1	0.184
125	0	0.184
125	0.1	0.184
125	0.01	0.184
125	0.001	0.184
125	0.5	0.184
125	1	0.184
175	0	0.184
175	0.1	0.184
175	0.01	0.184
175	0.001	0.184
175	0.5	0.184
175	1	0.184
200	0	0.184
200	0.1	0.184
200	0.01	0.184
200	0.001	0.184
200	0.5	0.184
200	1	0.184
225	0	0.184
225	0.1	0.184
225	0.01	0.184
225	0.001	0.184
225	0.5	0.184
225	1	0.184
255	0	0.184
255	0.1	0.184
255	0.01	0.184
255	0.001	0.184
255	0.5	0.184
255	1	0.184