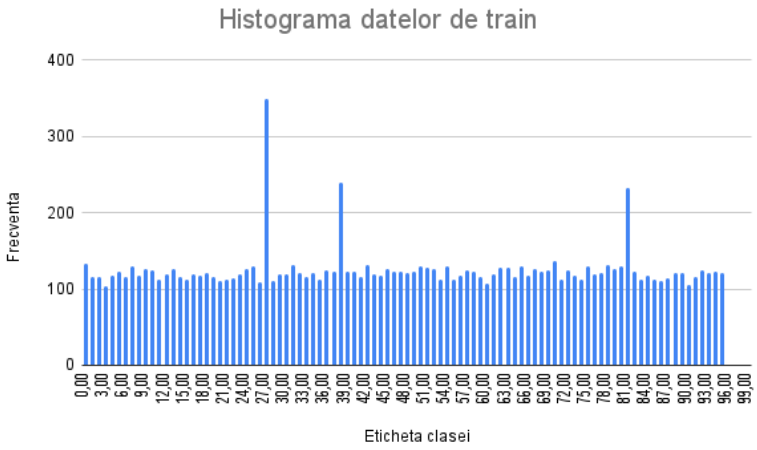
Documentatie proiect - Deep Hallucination Classification

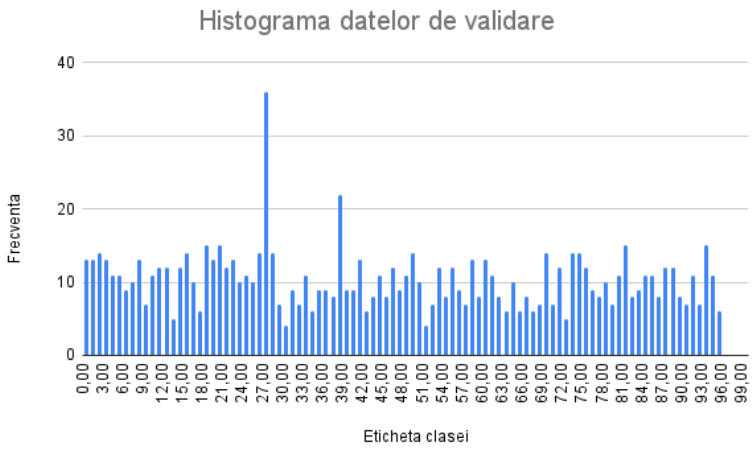
Acest proiect consta in clasificarea unor imagini de 64x64 in una din cele 96 de clase (cu etichetele de la 0 la 95). Setul de date de training contine 12,000 imagini, setul de validare 1,000, iar setul de testare 5,000.

Pentru rezolvarea cerintei am implementat urmatoarele doua modele:

1. SVM (Support Vector Machine) cu acuratetea maxima de 0.145.
2. CNN (Convolutional Neuronal Network) cu acuratetea maxima de **0.887.**

Dupa o analiza a histogramelor seturilor de date de train si de validare se poate observa ca nu au o distributie uniforma a claselor, unele clase sunt predominante, iar altele in minoritate. Pentru a avea o predictie cat mai buna a fost luat in considerare acest aspect prin calcularea unor ponderi a importantei fiecarei clase si folosirea acestora la antrenare.





1. Modelul SVM

a) Descrierea implementarii

Primul model implementat este un SVM (Support Vector Machine), care este un algoritm de învățare supervizată utilizat pentru probleme de clasificare, care poate fi aplicat cu succes în domeniul prelucrării imaginilor. Fiindca seturile de date deja au imaginile de aceeasi dimensiune (64x64) nu a mai fost necesar pasul de redimensionare al acestora.

Înainte de a antrena modelul SVM, am efectuat câteva operații de preprocesare asupra datelor de antrenare, validare și testare. Am citit imaginile din seturile de date și le-am transformat din matrice în vectori, apoi le-am salvat în liste corespunzătoare. Am păstrat și etichetele imaginilor ca șiruri de caractere în liste separate.

Pentru a asigura o mai bună convergență a modelului SVM, am normalizat datele utilizând metoda de normalizare standard. Apoi am ales hiperparametrii modelului dupa rezultatele cele mai bune obtinute din tabelul de mai jos si am antrenat modelul cu ajutorul acestora. In final am evaluat modelul pe datele de validare si am obtinut metricile corespunzatoare.

b) Optimizarea hiperparametrilor

Pentru optimizarea hiperparametrilor am incercat diferite valori pentru hiperparametrii modelului (kernel, C, gamma) din multimile urmatoare:

- kernel ∈ ["linear", "rbf", "poly"]
- C ∈ [0.1, 1, 10, 100]
- gamma ∈ [0.1, 1, ”auto”, ”scale”]

In tabel se pot observa urmatoarele valori de acuratete pentru versiunile de model incercate:

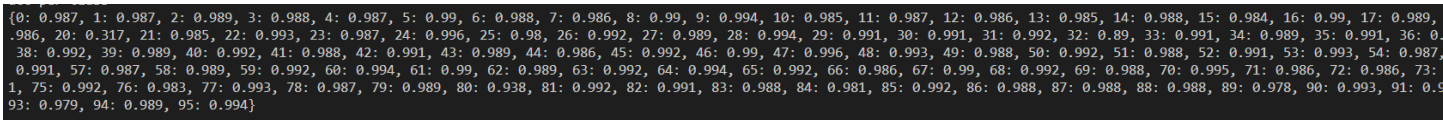
Hiperparametrii	**Acuratete pe setul de validare**
Kernel=”linear”, C=0.1, gamma=1, normalizare standard	0.085
Kernel=”linear”, C=1, gamma=”auto”, normalizare standard	0.085
Kernel=”linear”, C=10, gamma=”scale”, normalizare standard	0.085
Kernel=”linear”, C=0.1, gamma=”auto”	0.076
Kernel=”poly”, C=10, gamma=0.1, normalizare standard	0.089
Kernel=”poly”, C=100, gamma=”auto”, normalizare standard	0.084
Kernel=”poly”, C=0.1, gamma=”scale”	0.104
Kernel=”poly”, C=0.1, gamma=”auto”, normalizare standard	0.048
Kernel=”poly”, C=1, gamma=”scale”, normalizare standard	0.072
Kernel=”rbf”, C=0.1, gamma=”scale”, normalizare standard	0.079
Kernel=”rbf”, C=10, gamma=”auto”	0.036

Kernel=”rbf”, C=100, gamma=0.01, normalizare standard	0.036
Kernel=”rbf”, C=10, gamma=”auto” , normalizare standard	0.036
Kernel=”rbf”, C=1, gamma=”scale”	0.145
Kernel=”rbf”, C=10, gamma=”scale”	0.141
Kernel=”rbf”, C=0.1, gamma=”scale”	0.075

Acuratetea cea mai buna a fost obtinuta pentru kernelul=”rbf”, C=1 si gamma=”scale”. Pentru acest model am obtinut urmatoarele metrici:

- Accuracy score

  Acuratetea este afisata intr-un dictionar care are ca cheie eticheta clasei si ca valoare acuratetea clasei.



- Precision score

```
[0.07692308 0.07692308 0.09090909 0.         0.25       0.2
 0.33333333 0.         0.         0.         0.         0.07692308
 0.35714286 0.06666667 0.25       0.07692308 0.         0.45454545
 0.         0.         0.77777778 0.07142857 0.14285714 0.
 0.5        0.         0.         0.27272727 0.16666667 0.
 0.11111111 0.125      0.68181818 0.         0.09090909 0.
 0.         0.         0.         0.09090909 0.125      0.25
 0.         0.18181818 0.         0.18181818 0.         0.
 0.28571429 0.         0.         0.         0.44444444 0.28571429
 0.15384615 0.375      0.22222222 0.         0.09090909 0.
 0.         0.         0.16666667 0.         0.         0.
 0.         0.         0.         0.         0.         0.07142857
 0.21428571 0.08333333 0.         0.25       0.1        0.
 0.15384615 0.         0.66666667 0.         0.11111111 0.
 0.         0.         0.         0.         0.5        0.28571429
 0.         0.         0.         0.         0.18181818 0.        ],
```

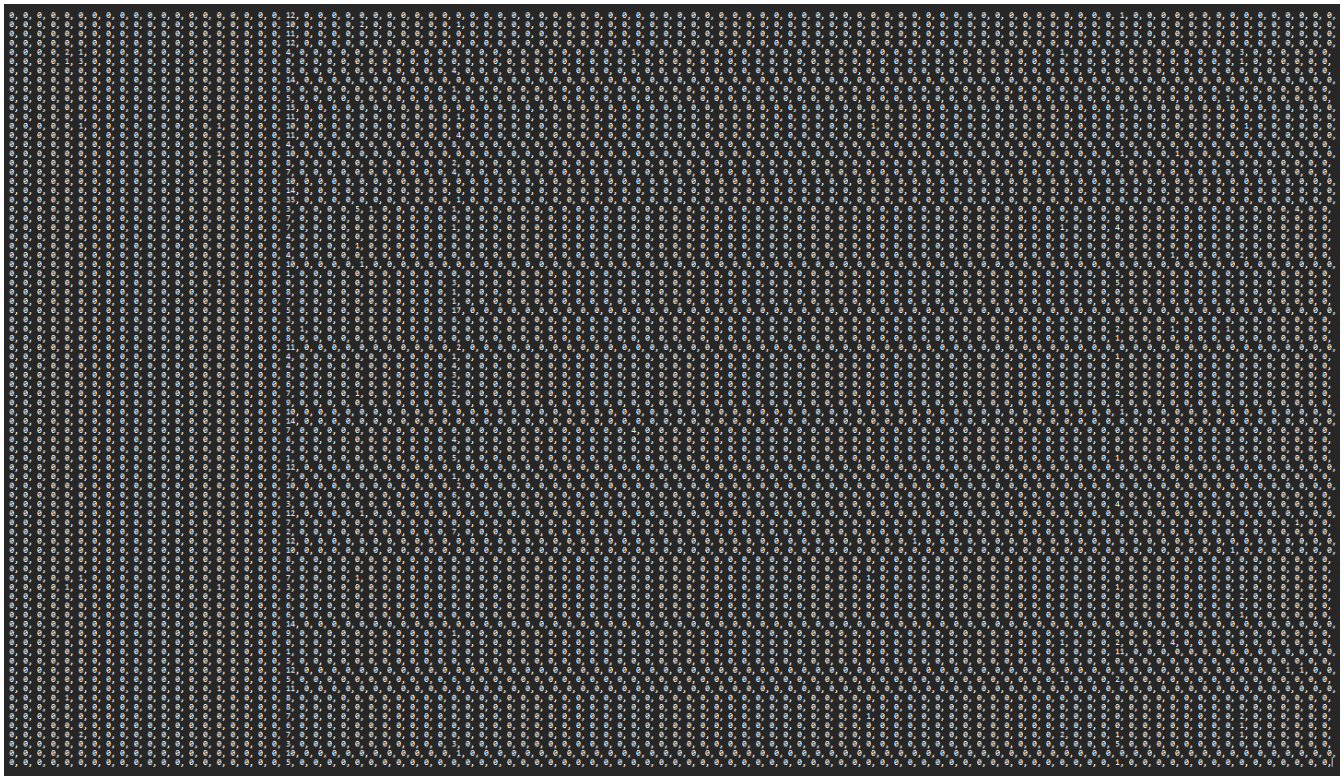
- Recall score

```
[0.08333333 0.11764706 0.13333333 0.         0.35294118 0.25
 0.20512821 0.         0.         0.         0.         0.08695652
 0.43478261 0.04347826 0.23076923 0.11764706 0.         0.5
 0.         0.         0.33333333 0.07142857 0.18181818 0.
 0.30769231 0.         0.         0.33333333 0.15384615 0.
 0.1        0.11764706 0.28846154 0.         0.15384615 0.
 0.         0.         0.         0.05128205 0.0625     0.28571429
 0.         0.14814815 0.         0.28571429 0.         0.
 0.22222222 0.         0.         0.         0.32       0.36363636
 0.09302326 0.24       0.22222222 0.         0.04166667 0.
 0.         0.         0.16666667 0.         0.         0.
 0.         0.         0.         0.         0.         0.13333333
 0.33333333 0.14285714 0.         0.18181818 0.14285714 0.
 0.21052632 0.         0.33333333 0.         0.2        0.
 0.         0.         0.         0.         0.14545455 0.21052632
 0.         0.         0.         0.         0.30769231 0.        ]
```

- F1 score

```
[0.09090909 0.25       0.25       0.         0.6        0.33333333
 0.14814815 0.         0.         0.         0.         0.1
 0.55555556 0.03225806 0.21428571 0.25       0.         0.55555556
 0.         0.         0.21212121 0.07142857 0.25       0.
 0.22222222 0.         0.         0.42857143 0.14285714 0.
 0.09090909 0.11111111 0.18292683 0.         0.5        0.
 0.         0.         0.         0.03571429 0.04166667 0.33333333
 0.         0.125      0.         0.66666667 0.         0.
 0.18181818 0.         0.         0.         0.25       0.5
 0.06666667 0.17647059 0.22222222 0.         0.02702703 0.
 0.         0.         0.16666667 0.         0.         0.
 0.         0.         0.         0.         0.         1.
 0.75       0.5        0.         0.14285714 0.25       0.
 0.33333333 0.         0.22222222 0.         1.         0.
 0.         0.         0.         0.         0.08510638 0.16666667
 0.         0.         0.         0.         1.         0.        ],
```

- Matricea de confuzie:



c) Preprocesarea si augmentarea datelor

Am augmentat la date imaginile rotite la dreapta cu 90 de grade. Ca preprocesare am folosit blurarea imaginilor, folosind GaussianBlur cu radius=5. Astfel, am obținut mai multe exemple pentru fiecare imagine de antrenare, crescând astfel diversitatea setului de antrenare.

2. Modelul CNN

a) Descrierea implementarii

Pentru al doilea model am implementat un CNN (Convolutional Neuronal Network). CNN-urile sunt o abordare puternică în domeniul prelucrării imaginilor, deoarece pot captura caracteristici complexe și ierarhice din imagini, permițând astfel clasificarea precisă. Fiindca seturile de date deja au imaginile de aceeasi dimensiune (64x64) nu a mai fost necesar pasul de redimensionare al acestora. Pentru normalizarea datelor am translatat valorile pixelilor imaginilor din intervalul [0, 255] in [0, 1]. Am incercat si translatarea in intervalul [-1, 1], dar acesta nu a dat rezultate prea bune asa ca am renuntat la aceasta abordare. Normalizarea este utila pentru ca asigură convergența și stabilitatea antrenării.

Pentru antrenarea modelului am selectat functia de loss CrossEntropyLoss si functia de optimizare Adam. Am impartit datele de traning in batchuri si am monitorizat functia de loss si acuratetea pe setul de validare ca sa detectam overfittingul si sa ajustez hiperparametrii. Regularizarea este o modalitate de adăugare a unor constrângeri sau penalități la model, astfel încât să nu se produca overfit pe datele de antrenament. Ca metode de regularizare am folosit urmatoarele:

- Am folosit dropoutul, adica am facut drop la niste neuroni random din retea ca sa adaugam diversitate si sa prevenim overfittingul.
- Am folosit regularizarea l2. Am adaugat o penalitate la functia de loss care incurajeaza ponderile sa fie mai mici sau imprastiate.
- Augmentarea datelor
- Early stopping. Adica am oprit antrenarea cand pierderea pe datele de validare nu se imbunatateste ca sa previn overfittingul.
- Normalizarea batchurilor. Am normalizat inputul si outputul fiecarui layer ca sa reduc deplasarea covariabilă internă și pentru a îmbunătăți stabilitatea și performanța modelului.

b) Prelucrarea datelor si augmentarea

Pentru a prelucra imaginile, am utilizat biblioteca PIL pentru încărcarea imaginilor și am aplicat o serie de transformări asupra acestora în funcție de setul de date (antrenament, validare sau testare). Pentru setul de antrenament, am aplicat o serie de transformări, cum ar fi color jittering, flip-uri orizontale și blur gaussian, pentru a genera mai multe variante ale imaginilor și a îmbunătăți performanța

modelului. Pentru seturile de validare și testare, am aplicat doar transformările necesare pentru a redimensiona imaginea și a o converti în tensor.

Mai exact, ca tehnici de augmentare a datelor am incercat urmatoarele transformari cu diverse valori ale parametrilor: RandomHorizontalFlip, RandomVerticalFlip, RandomGrayscale, GaussianBlur, RandomErasing si RandomAffine. In final cea mai buna acuratete am obtinut-o folosind: RandomAffine(degrees=(-15, 15)), RandomHorizontalFlip(), RandomGrayscale(0.05), ToTensor(), RandomErasing(0.15, scale=(0.02, 0.20)).

c) Optimizarea hiperparametriilor

Pentru tunarea hiperparametrilor modelului, am modificat arhitectura retelei, rata de invatare, cu cat se degradeaza rata de invatare si aplicarea de regularizare (L2/Dropout). Am pornit de la o arhitectura cu 5 straturi convolutionale si un strat complet conectat. Dupa fiecare strat convolutional, am facut max pooling, normalizare a batchului, dropout si ReLU (in ordinea asta). In urmatorul tabel se poate observa evolutia modelului:

Hiperparametrii	**Acuratete pe setul de validare**
5 straturi convolutionale (n1=32, n2=64, n3=64 n4=128, n5=256), un strat complet conectat (n=16 *n4), lr=0.003, degradare=0.0001, epoci=50, batch=32, early stopping, reglarizare l2 pe functia de loss, dropout (0.015, 0.03, 0.05, 0.15, 0.4)	0.36133
5 straturi convolutionale (n1=32, n2=64, n3=64 n4=128, n5=256), un strat complet conectat (n=16 *n4), lr=0.0003, degradare=0.00001, epoci=30, batch=32, early stopping, reglarizare l2 pe functia de loss, dropout (0.4, 0.4, 0.4, 0.4, 0.4). Graficul cu functia de loss pentru aceasta configuratie se poate vedea in Figura 1.	0.80866
5 straturi convolutionale (n1=32, n2=64, n3=64 n4=128, n5=256), un strat complet conectat (n=16 *n4), lr=0.0003, degradare=0.00001, epoci=24, batch=32, early stopping, reglarizare l2 pe functia de loss, dropout (0.015, 0.03, 0.05, 0.15, 0.5). Graficul cu functia de loss pentru aceasta configuratie se poate vedea in Figura 2.	0.83266
5 straturi convolutionale (n1=32, n2=64, n3=64 n4=128, n5=256), un strat complet conectat (n=16 *n4), lr=0.0003, degradare=0.00001, epoci=37, batch=32, early stopping, reglarizare l2 pe functia de loss, dropout (0.015, 0.03, 0.05, 0.15, 0.4). Graficul cu functia de loss pentru aceasta configuratie se poate vedea in Figura 3.	0.82833

4 straturi convolutionale (n1=32, n2=32, n3=64, n4=64), doua straturi complet conectate (16 *n2, n5; ReLU, Softmax), lr=0.0003, degradare=0.00001, epoci=37, batch=32, early stopping, reglarizare l2 pe functia de loss, dropout (0.25, 0.25, 0.25, 0.25, 0.5).	0.24349
5 straturi convolutionale (n1=32, n2=32, n3=64 n4=128, n5=256), un strat complet conectate (4 *n4), lr=0.0003, degradare=0.00001, epoci=39, batch=32, early stopping, reglarizare l2 pe functia de loss, dropout (0.015, 0.03, 0.05, 0.15, 0.5)	**0.842**
4 straturi convolutionale (n1=32, n2=64, n3=128 n4=128), doua straturi complet conectate (16 *n3, 4*n3), lr=0.0003, degradare=0.00001, epoci=20, batch=32, early stopping, reglarizare l2 pe functia de loss, fara dropout	**0.88228**
4 straturi convolutionale (n1=32, n2=64, n3=128 n4=128), un strat complet conectate (16 *n4), lr=0.0003, degradare=0.00001, epoci=36, batch=32, early stopping, reglarizare l2 pe functia de loss, dropout 0.25 doar ultimul ultimul strat convolutional. Graficul cu functia de loss pentru aceasta configuratie se poate vedea in Figura 4.	**0.887685**

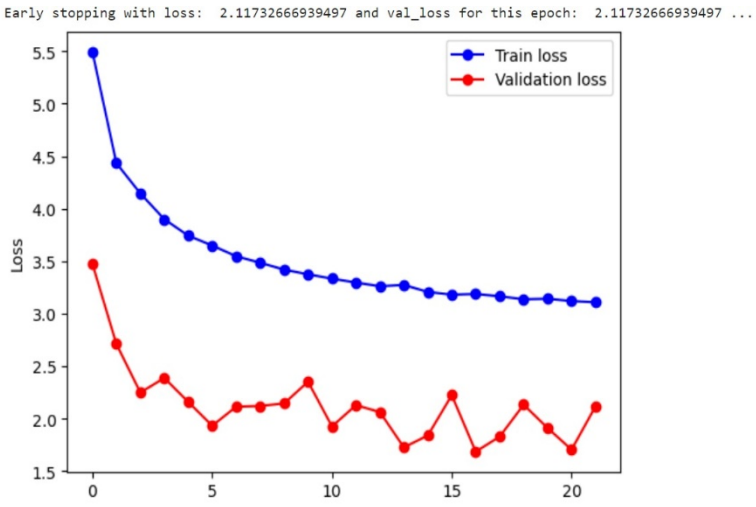


Figura 1

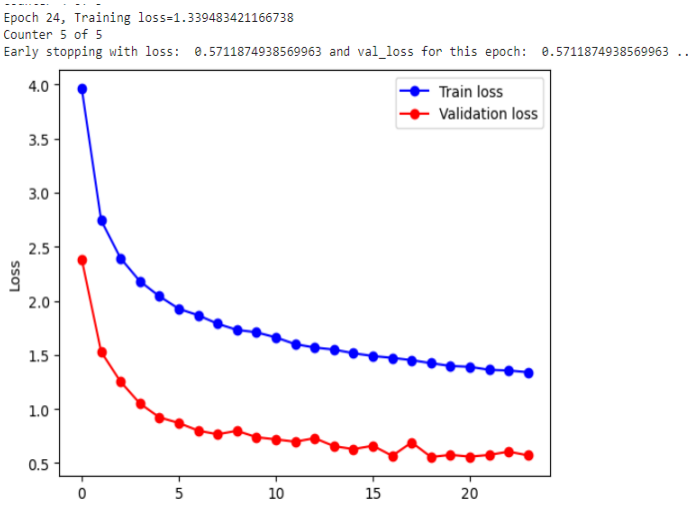


Figura 2

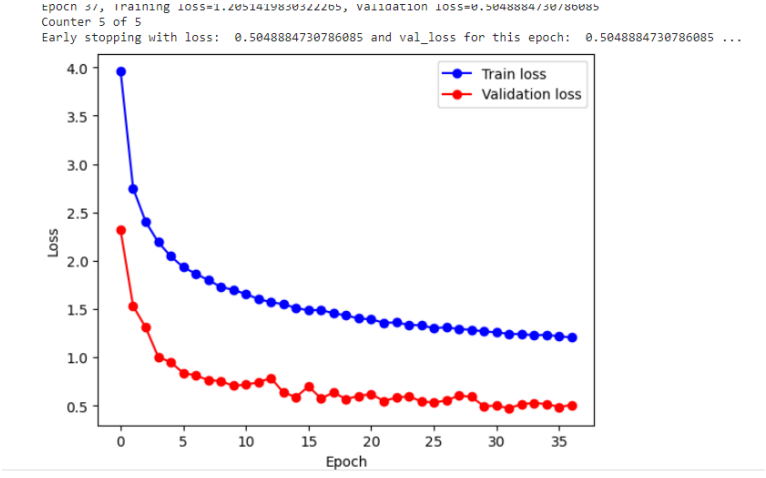


Figura 3

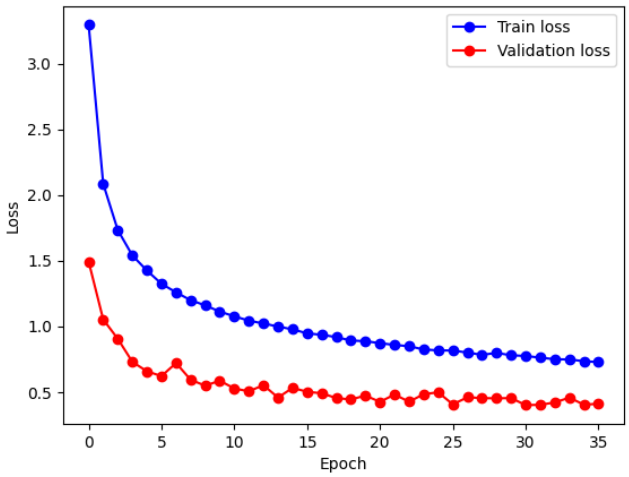
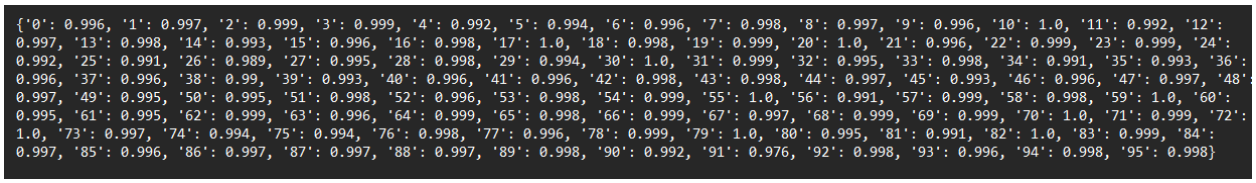


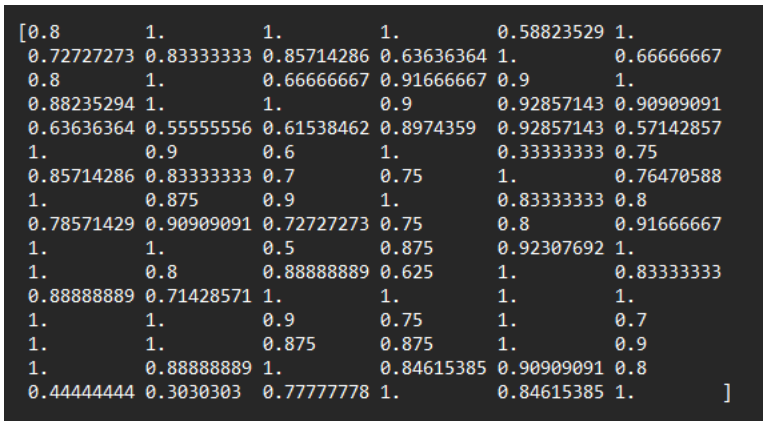
Figura 4

Cea mai buna acuratete am obtinut-o pe versiunea cu patru straturi convolutionale (n1=32, n2=64, n3=128 n4=128) cu functia de activare ReLU, urmate de max pooling si normalizare inainte de ultimul strat complet conectat. Pentru aceasta versiune am obtinut urmatoarele metrici:

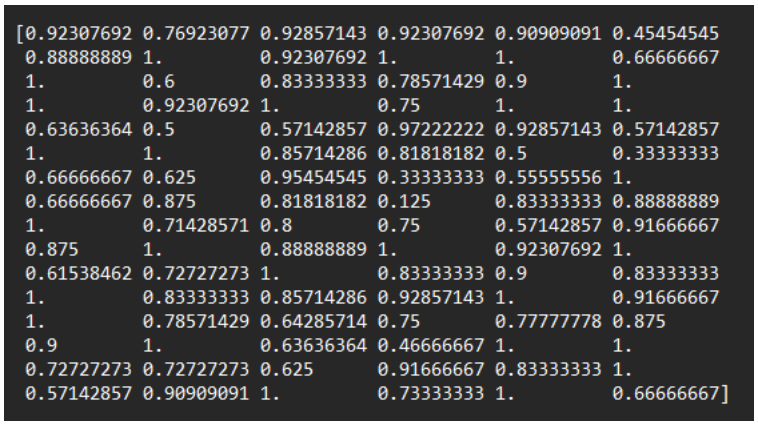
- Accuracy score



- Precision score



- Recall score



- F1 score

```
[0.85714286 0.86956522 0.96296296 0.96       0.71428571 0.625
 0.8        0.90909091 0.88888889 0.77777778 1.         0.66666667
 0.88888889 0.75       0.74074074 0.84615385 0.9        1.
 0.9375     0.96       1.         0.81818182 0.96296296 0.95238095
 0.63636364 0.52631579 0.59259259 0.93333333 0.92857143 0.57142857
 1.         0.94736842 0.70588235 0.9        0.4        0.46153846
 0.75       0.71428571 0.80769231 0.46153846 0.71428571 0.86666667
 0.8        0.875      0.85714286 0.22222222 0.83333333 0.84210526
 0.88       0.8        0.76190476 0.75       0.66666667 0.91666667
 0.93333333 1.         0.64       0.93333333 0.92307692 1.
 0.76190476 0.76190476 0.94117647 0.71428571 0.94736842 0.83333333
 0.94117647 0.76923077 0.92307692 0.96296296 1.         0.95652174
 1.         0.88       0.75       0.75       0.875      0.77777778
 0.94736842 1.         0.73684211 0.60869565 1.         0.94736842
 0.84210526 0.8        0.76923077 0.88       0.86956522 0.88888889
 0.5        0.45454545 0.875      0.84615385 0.91666667 0.8       ]
```

- Matricea de confuzie