

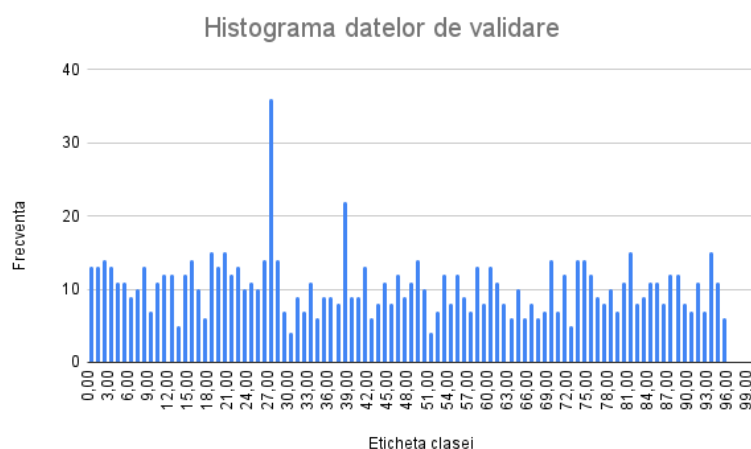
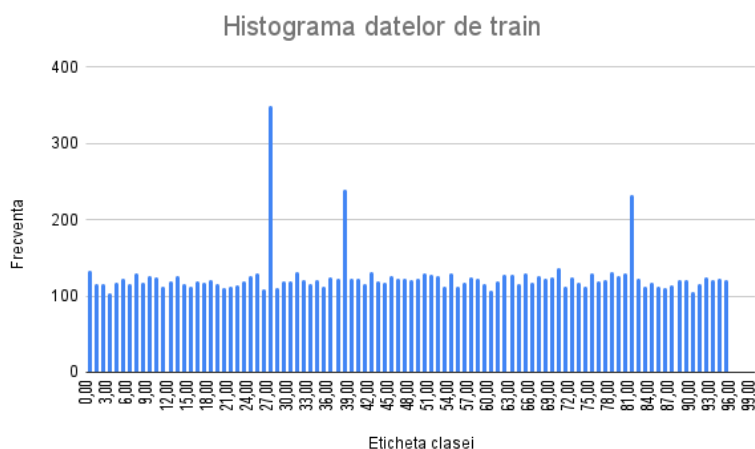
## Documentatie proiect - Deep Hallucination Classification

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. Fiindcă seturile de date deja au imaginile de aceeași 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 după rezultatele cele mai bune obținute din tabelul de mai jos și am antrenat modelul cu ajutorul acestora. În final am evaluat modelul pe datele de validare și am obținut metricile corespunzătoare.

#### b) Optimizarea hiperparametrilor

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

- kernel  $\in$  ["linear", "rbf", "poly"]
- C  $\in$  [0.1, 1, 10, 100]
- gamma  $\in$  [0.1, 1, "auto", "scale"]

În tabel se pot observa următoarele valori de acuratețe pentru versiunile de model încercate:

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.

```
{0: 0.987, 1: 0.987, 2: 0.989, 3: 0.988, 4: 0.987, 5: 0.99, 6: 0.988, 7: 0.986, 8: 0.99, 9: 0.994, 10: 0.985, 11: 0.987, 12: 0.986, 13: 0.985, 14: 0.988, 15: 0.984, 16: 0.99, 17: 0.989, 18: 0.986, 19: 0.987, 20: 0.985, 21: 0.985, 22: 0.993, 23: 0.987, 24: 0.996, 25: 0.98, 26: 0.992, 27: 0.989, 28: 0.994, 29: 0.991, 30: 0.991, 31: 0.992, 32: 0.89, 33: 0.991, 34: 0.989, 35: 0.991, 36: 0.992, 37: 0.992, 38: 0.992, 39: 0.989, 40: 0.992, 41: 0.988, 42: 0.991, 43: 0.989, 44: 0.986, 45: 0.992, 46: 0.99, 47: 0.996, 48: 0.993, 49: 0.988, 50: 0.992, 51: 0.988, 52: 0.991, 53: 0.993, 54: 0.987, 55: 0.991, 56: 0.987, 57: 0.987, 58: 0.989, 59: 0.992, 60: 0.994, 61: 0.99, 62: 0.989, 63: 0.992, 64: 0.994, 65: 0.992, 66: 0.986, 67: 0.99, 68: 0.992, 69: 0.988, 70: 0.995, 71: 0.986, 72: 0.986, 73: 0.992, 74: 0.983, 75: 0.992, 76: 0.983, 77: 0.993, 78: 0.987, 79: 0.989, 80: 0.938, 81: 0.992, 82: 0.991, 83: 0.988, 84: 0.981, 85: 0.992, 86: 0.988, 87: 0.988, 88: 0.988, 89: 0.978, 90: 0.993, 91: 0.993, 92: 0.979, 93: 0.989, 94: 0.989, 95: 0.994}
```

- 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. 0.09090909 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. ],
```

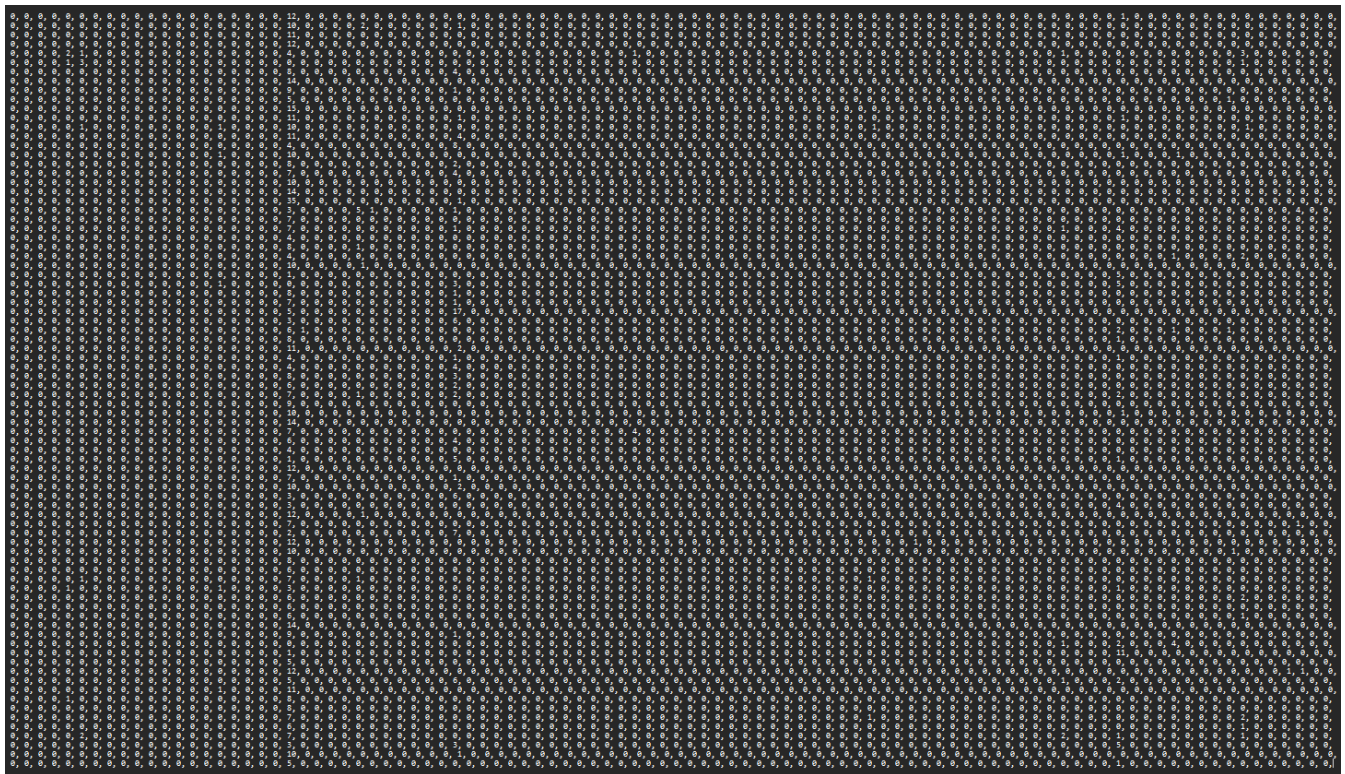
- Recall score

```
[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. Fiindcă seturile de date deja au imaginile de aceeași dimensiune (64x64) nu a mai fost necesar pasul de redimensionare al acestora. Pentru normalizarea datelor am translatat valorile pixelilor imaginilor din intervalul  $[0, 255]$  în  $[0, 1]$ . Am încercat și translatarea în intervalul  $[-1, 1]$ , dar acesta nu a dat rezultate prea bune așa că am renunțat la această abordare. Normalizarea este utilă pentru ca asigură convergența și stabilitatea antrenării.

Pentru antrenarea modelului am selectat funcția de loss `CrossEntropyLoss` și funcția de optimizare `Adam`. Am împărțit datele de training în batchuri și am monitorizat funcția de loss și acuratetea pe setul de validare ca să detectăm `overfittingul` și să ajustez hiperparametrii. Regularizarea este o modalitate de adăugare a unor constrângeri sau penalități la model, astfel încât să nu se producă `overfit` pe datele de antrenament. Ca metode de regularizare am folosit următoarele:

- Am folosit `dropoutul`, adică am făcut `drop` la niște neuroni random din rețea ca să adăugăm diversitate și să prevenim `overfittingul`.
- Am folosit regularizarea `l2`. Am adăugat o penalitate la funcția de loss care încurajează ponderile să fie mai mici sau împrastiate.
- Augmentarea datelor
- `Early stopping`. Adică am oprit antrenarea când pierderea pe datele de validare nu se îmbunătățește ca să prevenim `overfittingul`.
- Normalizarea batchurilor. Am normalizat inputul și outputul fiecărui layer ca să reduc deplasarea covariabilă internă și pentru a îmbunătăți stabilitatea și performanța modelului.

### b) Prelucrarea datelor și 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 încercat următoarele transformări cu diverse valori ale parametrilor: RandomHorizontalFlip, RandomVerticalFlip, RandomGrayscale, GaussianBlur, RandomErasing și RandomAffine. În final cea mai bună acuratețe am obținut-o folosind: RandomAffine(degrees=(-15, 15)), RandomHorizontalFlip(), RandomGrayscale(0.05), ToTensor(), RandomErasing(0.15, scale=(0.02, 0.20)).

### c) Optimizarea hiperparametrilor

Pentru tunarea hiperparametrilor modelului, am modificat arhitectura rețelei, rata de învățare, cu cât se degradează rata de învățare și aplicarea de regularizare (L2/Dropout). Am pornit de la o arhitectură cu 5 straturi convoluționale și un strat complet conectat. După fiecare strat convoluțional, am făcut max pooling, normalizare a batchului, dropout și ReLU (în ordinea asta). În următorul tabel se poate observa evoluția modelului:

Hiperparametrii	Acuratete pe setul de validare
5 straturi convoluționale (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, regularizare l2 pe funcția de loss, dropout (0.015, 0.03, 0.05, 0.15, 0.4)	0.36133
5 straturi convoluționale (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, regularizare l2 pe funcția de loss, dropout (0.4, 0.4, 0.4, 0.4, 0.4). Graficul cu funcția de loss pentru această configurație se poate vedea în Figura 1.	0.80866
5 straturi convoluționale (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, regularizare l2 pe funcția de loss, dropout (0.015, 0.03, 0.05, 0.15, 0.5). Graficul cu funcția de loss pentru această configurație se poate vedea în Figura 2.	0.83266
5 straturi convoluționale (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, regularizare l2 pe funcția de loss, dropout (0.015, 0.03, 0.05, 0.15, 0.4). Graficul cu funcția de loss pentru această configurație se poate vedea în 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)	<b>0.842</b>
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	<b>0.88228</b>
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.	<b>0.887685</b>

Early stopping with loss: 2.11732666939497 and val\_loss for this epoch: 2.11732666939497 ...

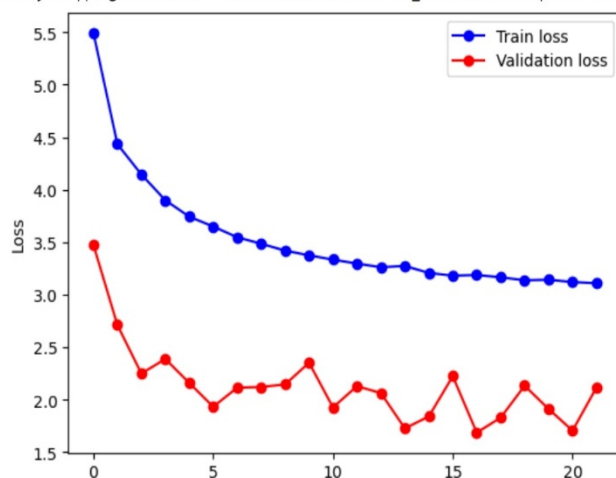


Figura 1

Epoch 24, Training loss=1.339483421166738

Counter 5 of 5

Early stopping with loss: 0.5711874938569963 and val\_loss for this epoch: 0.5711874938569963 ...

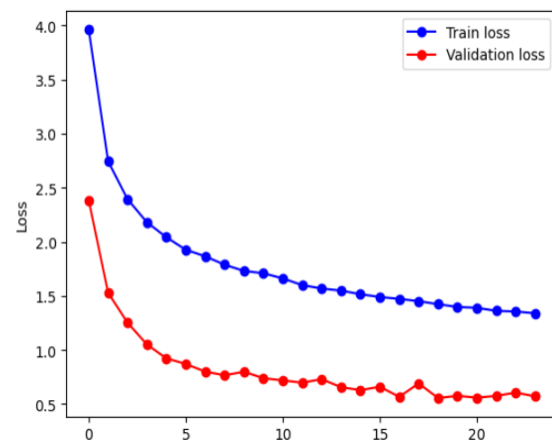


Figura 2

Epoch 37, Training loss=1.2051419830322205, Validation loss=0.5048884730786085  
Counter 5 of 5  
Early stopping with loss: 0.5048884730786085 and val\_loss for this epoch: 0.5048884730786085 ...

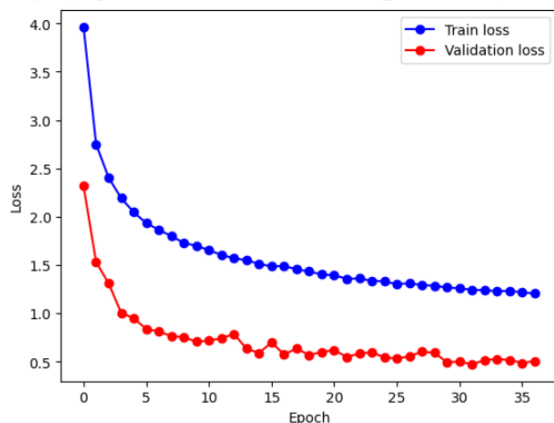


Figura 3

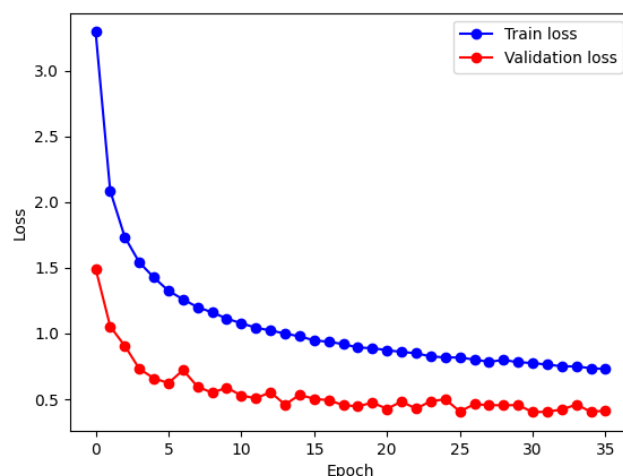


Figura 4

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 metrice:

- Accuracy score

```
{'0': 0.996, '1': 0.997, '2': 0.999, '3': 0.999, '4': 0.992, '5': 0.994, '6': 0.996, '7': 0.998, '8': 0.997, '9': 0.996, '10': 1.0, '11': 0.992, '12': 0.997, '13': 0.998, '14': 0.993, '15': 0.996, '16': 0.998, '17': 1.0, '18': 0.998, '19': 0.999, '20': 1.0, '21': 0.996, '22': 0.999, '23': 0.999, '24': 0.992, '25': 0.991, '26': 0.989, '27': 0.995, '28': 0.998, '29': 0.994, '30': 1.0, '31': 0.999, '32': 0.995, '33': 0.998, '34': 0.991, '35': 0.993, '36': 0.996, '37': 0.996, '38': 0.99, '39': 0.993, '40': 0.996, '41': 0.996, '42': 0.998, '43': 0.998, '44': 0.997, '45': 0.993, '46': 0.996, '47': 0.997, '48': 0.997, '49': 0.995, '50': 0.995, '51': 0.998, '52': 0.996, '53': 0.998, '54': 0.999, '55': 1.0, '56': 0.991, '57': 0.999, '58': 0.998, '59': 1.0, '60': 0.995, '61': 0.995, '62': 0.999, '63': 0.996, '64': 0.999, '65': 0.998, '66': 0.999, '67': 0.997, '68': 0.999, '69': 0.999, '70': 1.0, '71': 0.999, '72': 1.0, '73': 0.997, '74': 0.994, '75': 0.994, '76': 0.998, '77': 0.996, '78': 0.999, '79': 1.0, '80': 0.995, '81': 0.991, '82': 1.0, '83': 0.999, '84': 0.997, '85': 0.996, '86': 0.997, '87': 0.997, '88': 0.997, '89': 0.998, '90': 0.992, '91': 0.976, '92': 0.998, '93': 0.996, '94': 0.998, '95': 0.998}
```

- Precision score

```
[0.8      1.      1.      1.      0.58823529 1.
 0.72727273 0.83333333 0.85714286 0.63636364 1.      0.66666667
 0.8      1.      0.66666667 0.91666667 0.9      1.
 0.88235294 1.      1.      0.9      0.92857143 0.90909091
 0.63636364 0.55555556 0.61538462 0.8974359 0.92857143 0.57142857
 1.      0.9      0.6      1.      0.33333333 0.75
 0.85714286 0.83333333 0.7      0.75      1.      0.76470588
 1.      0.875      0.9      1.      0.83333333 0.8
 0.78571429 0.90909091 0.72727273 0.75      0.8      0.91666667
 1.      1.      0.5      0.875      0.92307692 1.
 1.      0.8      0.88888889 0.625      1.      0.83333333
 0.88888889 0.71428571 1.      1.      1.      1.
 1.      1.      0.9      0.75      1.      0.7
 1.      1.      0.875      0.875      1.      0.9
 1.      0.88888889 1.      0.84615385 0.90909091 0.8
 0.44444444 0.3030303 0.77777778 1.      0.84615385 1.]
```

- Recall score

```
[0.92307692 0.76923077 0.92857143 0.92307692 0.90909091 0.45454545
 0.88888889 1.      0.92307692 1.      1.      0.66666667
 1.      0.6      0.83333333 0.78571429 0.9      1.
 1.      0.92307692 1.      0.75      1.      1.
 0.63636364 0.5      0.57142857 0.97222222 0.92857143 0.57142857
 1.      1.      0.85714286 0.81818182 0.5      0.33333333
 0.66666667 0.625      0.95454545 0.33333333 0.55555556 1.
 0.66666667 0.875      0.81818182 0.125      0.83333333 0.88888889
 1.      0.71428571 0.8      0.75      0.57142857 0.91666667
 0.875      1.      0.88888889 1.      0.92307692 1.
 0.61538462 0.72727273 1.      0.83333333 0.9      0.83333333
 1.      0.83333333 0.85714286 0.92857143 1.      0.91666667
 1.      0.78571429 0.64285714 0.75      0.77777778 0.875
 0.9      1.      0.63636364 0.46666667 1.      1.
 0.72727273 0.72727273 0.625      0.91666667 0.83333333 1.
 0.57142857 0.90909091 1.      0.73333333 1.      0.66666667]
```



- |             |            |            |            |            |            |
|-------------|------------|------------|------------|------------|------------|
| [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        |
| ]           |            |            |            |            |            |

- 
- This block contains a large, dark, textured rectangular area that appears to be a placeholder or a very dark image. It occupies the majority of the page below the header and above the footer.