Documentatie proiect - Inteligenta Artificiala

Borza Maria - Cristina

Grupa 244

1 Scopul proiectului

Scopul acestui proiect este antrenarea unui model pentru clasificarea unor imagini monocromatice, astfel incat acuratetea pe datele de testare sa fie cat mai mare.
In vederea atingerii scopului proiectului, am procedat la organizarea metodologica a etapelor de lucru si am utilizat mai multe metode.

2 Metode de lucru

Concret, in rezolvarea proiectului am testat urmatoarele modele, prezentate detaliat in continuare: masini cu vectori suport, retele neuronale, retele neuronale convolutionale.

2.1 Masini cu vectori suport (SVM)

Un prim model incercat a fost o masina cu vectori suport (SVM).
Pentru inceput am normalizat datele. Am incercat atat standardizarea, cat si normalizarea L1 si L2. Astfel am observat ca cea mai mare acuratete a fost obtinuta folosind normalizarea L2.
De asemenea, si pentru parametrii C, Kernel si Gamma am testat diverse valori. Valorile incercate au fost urmatoarele:

- Pentru **C = 1**, **Kernel = linear** si normalizarea standard a datelor am obtinut pe datele de validare acuratetea de 56.86% si matricea de confuzie (1).

$$\begin{pmatrix} 290. & 51. & 34. & 24. & 54. & 8. & 25. & 40. & 44. \\ 56. & 315. & 25. & 18. & 25. & 9. & 27. & 38. & 14. \\ 27. & 49. & 323. & 18. & 59. & 25. & 11. & 13. & 8. \\ 36. & 27. & 23. & 341. & 31. & 43. & 25. & 33. & 19. \\ 84. & 44. & 54. & 40. & 255. & 13. & 17. & 33. & 14. \\ 14. & 11. & 49. & 45. & 32. & 360. & 6. & 29. & 15. \\ 67. & 47. & 25. & 23. & 17. & 12. & 342. & 7. & 40. \\ 68. & 43. & 21. & 35. & 34. & 20. & 23. & 268. & 8. \\ 58. & 29. & 12. & 31. & 18. & 12. & 53. & 15. & 349. \end{pmatrix} \quad (1)$$

- Pentru **C = 1**, **Kernel = linear** si normalizarea L1 am obtinut pe datele de validare acuratetea de 13.4% si matricea de confuzie (2).

$$\begin{pmatrix} 0. & 0. & 0. & 5. & 498. & 0. & 60. & 0. & 7. \\ 0. & 0. & 0. & 6. & 507. & 0. & 14. & 0. & 0. \\ 0. & 0. & 0. & 1. & 528. & 0. & 4. & 0. & 0. \\ 0. & 0. & 0. & 62. & 500. & 0. & 16. & 0. & 0. \\ 0. & 0. & 0. & 4. & 548. & 0. & 2. & 0. & 0. \\ 0. & 0. & 0. & 2. & 559. & 0. & 0. & 0. & 0. \\ 0. & 1. & 0. & 4. & 525. & 0. & 50. & 0. & 0. \\ 0. & 0. & 0. & 5. & 512. & 0. & 3. & 0. & 0. \\ 0. & 0. & 0. & 11. & 460. & 0. & 96. & 0. & 10. \end{pmatrix} \tag{2}$$

- Pentru **C = 1**, **Kernel = linear** si normalizarea L2 am obtinut pe datele de validare acuratetea de 61.58% si matricea de confuzie (3).

$$\begin{pmatrix} 243. & 73. & 33. & 18. & 54. & 5. & 48. & 51. & 45. \\ 32. & 328. & 23. & 18. & 18. & 7. & 42. & 39. & 20. \\ 20. & 43. & 310. & 14. & 39. & 32. & 36. & 26. & 13. \\ 21. & 28. & 14. & 377. & 24. & 39. & 22. & 29. & 24. \\ 54. & 36. & 52. & 32. & 291. & 19. & 12. & 36. & 22. \\ 8. & 11. & 25. & 29. & 20. & 406. & 9. & 40. & 13. \\ 27. & 33. & 14. & 17. & 6. & 9. & 432. & 11. & 31. \\ 35. & 28. & 15. & 33. & 27. & 21. & 28. & 319. & 14. \\ 25. & 30. & 9. & 22. & 8. & 13. & 74. & 23. & 373. \end{pmatrix} \tag{3}$$

- Pentru **C = 1**, **Kernel = RBF**, **Gamma = 0.1** si normalizarea standard am obtinut pe datele de validare acuratetea de 11.64% si matricea de confuzie (4).

$$\begin{pmatrix} 0. & 0. & 0. & 0. & 570. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 527. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 533. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 578. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 554. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 561. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 569. & 0. & 11. & 0. & 0. \\ 0. & 0. & 0. & 0. & 518. & 0. & 1. & 0. & 1. \\ 0. & 0. & 0. & 0. & 560. & 0. & 0. & 0. & 17. \end{pmatrix} \tag{4}$$

Intrucat acuratetea obtinuta nu a fost satisfacatoare am mai incercat si testarea altor modele.

2.2 Retele neuronale

O a doua abordare a fost folosirea retelelor neuronale.
La fel ca la abordarea anterioara, am procedat prin a incepe cu normalizarea datelor. De data aceasta am normalizat datele folosind doar standardizarea.
Pentru construirea modelului am folosit urmatoarea secventa de cod:

```
model = Sequential([Flatten(input_shape = [32, 32]),
                    Dense(256, activation = "relu" ),
                    Dense(256, activation = "relu" ),
                    Dense(256, activation = "relu" ),
                    Dense(128, activation = "relu" ),
                    Dense(9, activation = "softmax")])
```

Primul strat pe care l-am folosit este un strat **Flatten**. Am folosit acest strat pentru a aplatiza array-urile de dimensiune 32×32 in array-uri de dimensiune 1×1024, astfel incat reteaua sa poata lucra cu ele.
Ulterior am adaugat inca 4 straturi **Dense**, primele 3 cu 256 de neuroni fiecare, iar urmatoarul cu cate 128 de neuroni, toate cu functia de activare **ReLU**.
In final am mai adaugat un strat **Dense**, cu 9 neuroni (deoarece sunt 9 clase care pot fi prezise) si functia de activare **softmax**.

La compilarea modelului am folosit functia de loss **Sparse Categorical Crossentropy**, optimizatorul **Stochastic Gradient Descent (SGD)** si ca metrica am folosit acuratetea.
Am antrenat modelul pe datele de antrenare, timp de 20 de epoci.

In urma acestei abordari am obtinut o acuratete de 74.14% si matricea de confuzie (5).

$$\begin{pmatrix} 342. & 36. & 17. & 16. & 54. & 5. & 45. & 32. & 23. \\ 12. & 453. & 11. & 5. & 8. & 2. & 9. & 23. & 4. \\ 13. & 35. & 388. & 16. & 37. & 16. & 8. & 19. & 1. \\ 24. & 18. & 20. & 410. & 22. & 19. & 20. & 23. & 22. \\ 37. & 33. & 26. & 24. & 383. & 8. & 6. & 26. & 11. \\ 4. & 10. & 26. & 32. & 23. & 423. & 8. & 29. & 6. \\ 27. & 9. & 9. & 9. & 7. & 3. & 494. & 3. & 19. \\ 30. & 19. & 15. & 32. & 28. & 26. & 15. & 347. & 8. \\ 27. & 4. & 2. & 8. & 4. & 7. & 56. & 2 & .467. \end{pmatrix} \tag{5}$$

Intrucat acuratetea obtinuta tot nu a fost suficient de buna, am mai incercat si o alta abordare.

2.3 Retele neuronale convolutionale

La fel ca la incercarile anterioare, am inceput aceasta abordare prin normalizarea datelor.
Initial, pentru construirea modelului am folosit urmatoarea secventa de cod:

```
model = Sequential([
    Conv2D(32, 8, activation="relu", padding= "same", input_shape=[32, 32, 1]),
    MaxPooling2D(2),
    Conv2D(64, 3, activation="relu", padding="same"),
    Conv2D(64, 3, activation="relu", padding="same"),
    MaxPooling2D(2),
    Flatten(),
    Dense(128, activation="relu"),
    Dense(64, activation="relu"),
```

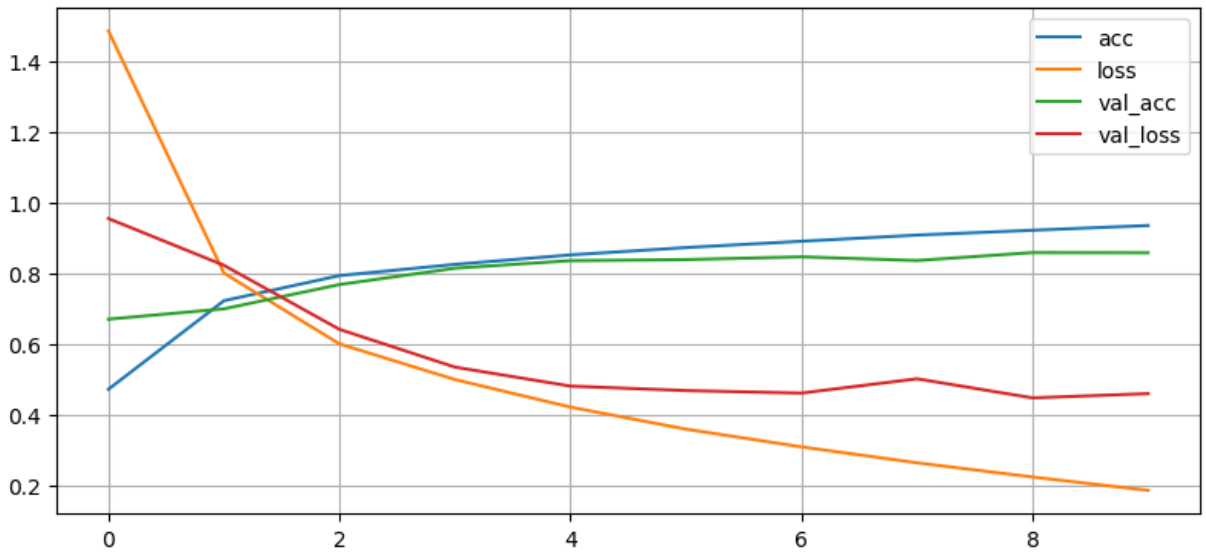
```
    Dense(64, activation="relu"),
    Dense(9, activation="softmax")
])
```

Primul strat pe care l-am adaugat este un strat convolutional, cu 32 de filtre, cu marimea kernel-ului de 8 × 8 si cu functia de activare **ReLU**. Deoarece acesta este primul strat din retea, am specificat si un input shape de 32 × 32 × 1.
Al doilea strat pe care l-am adaugat este un strat de tipul **MaxPooling**.
Urmatoarele straturi adaugate sunt 2 straturi convolutionale, dar cu un numar mai ridicat de filtre (64 de filtre) si inca un strat de tipul MaxPooling.
La final, dupa ce am mai adaugat un strat de tip Flatten, pentru a reformata datele la dimensiunea de 1 × 1024, am adaugat inca 4 straturi dense (similar cu ce am facut la abordarea anterioara).

La compilarea modelului, la fel ca la incercarea anterioara, am folosit functia de loss **Sparse Categorical Crossentropy**, optimizatorul **Stochastic Gradient Descent (SGD)** si ca metrica - acuratetea.
De aceasta data am atrenat modelul timp de 10 epoci.
Acuratetea obtinuta pe datele de validare a crescut la 85.52%, observandu-se astfel o imbunatatire semnificativa fata de abordarile anterioare. Matricea de confuzie obtinuta a fost (6)

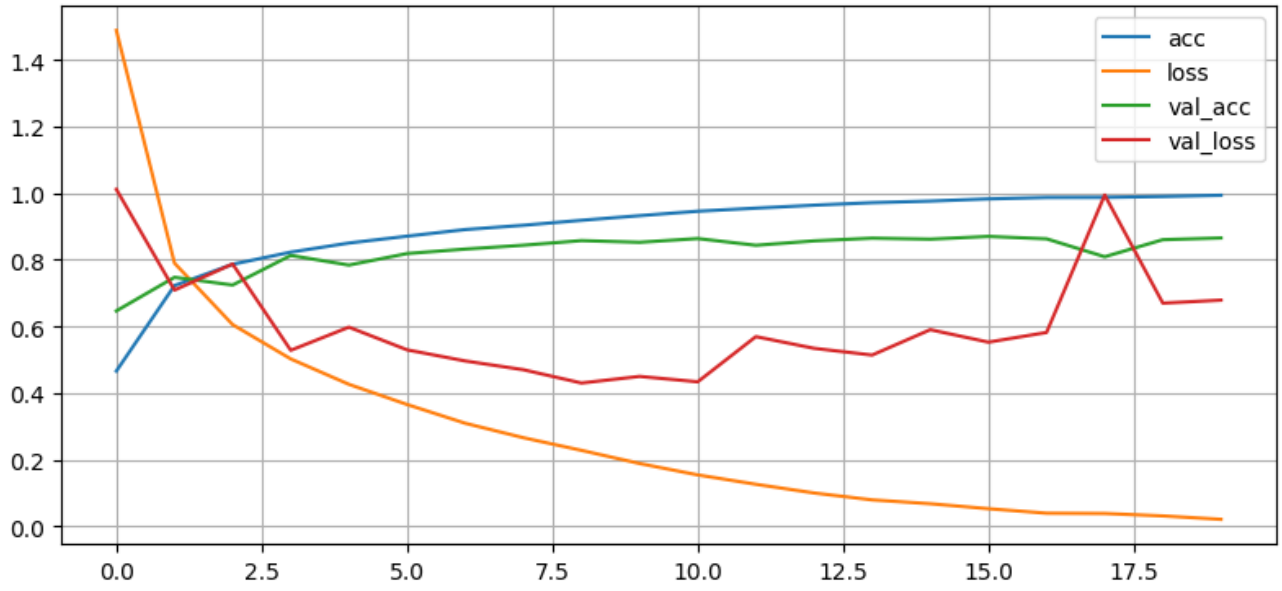
$$\begin{pmatrix} 444. & 16. & 6. & 7. & 25. & 3. & 24. & 18. & 27. \\ 6. & 490. & 8. & 6. & 2. & 3. & 2. & 7. & 3. \\ 6. & 14. & 463. & 11. & 11. & 14. & 5. & 9. & 0. \\ 7. & 1. & 12. & 517. & 12. & 6. & 7. & 6. & 10. \\ 22. & 8. & 24. & 37. & 424. & 8. & 6. & 12. & 13. \\ 5. & 5. & 18. & 23. & 7. & 463. & 12. & 22. & 6. \\ 7. & 7. & 5. & 1. & 2. & 1. & 543. & 4. & 10. \\ 10. & 12. & 17. & 22. & 3. & 9. & 11. & 431. & 5. \\ 9. & 2. & 0. & 11. & 0. & 3. & 36. & 3. & 513. \end{pmatrix} \tag{6}$$

De aceasta data, pentru a observa evolutia functiei de loss si a acuratetii pe datele de testare si de validarea, am decis plotarea acestora. Graficul rezultat este urmatorul:



De pe grafic se poate observa cum acuratetea creste treptat, iar functia de loss scade. De asemenea, se observa ca acurateatea pe datele de antrenare este mai mare, dar nu cu o diferenta semninficativa fata de datele de validare, ceea ce inseamna ca nu avem overfitting.

Intrucat acuratetea pe datele de antrenare este doar de 90.23% , am decis ca in urmatoarea incercare sa pastrez acelasi model, dar de data aceasta sa antrenez modelul 20 de epoci.
Acuratetea obtinuta in acest caz pe datele de validare a fost de 86.54%. Totusi, dupa ce am plotat acelasi grafic, acum am observat fenomenul de overfit - acuratetea pe datele de antrenare ajunge aproape de 1 (0.9934), in timp de acuratetea pe datele de validare nu se imbunatateste.



Urmatoarea incercare a fost sa pastrez modelul, dar de aceasta data l-am antrenat 12 epoci, iar ca optimizator am folosit **Adaptive Moment Estimation (ADAM)**. De aceasta data insa, acuratetea obtinuta pe datele de validare a scazut la 82.84%.

Urmatoarea modificare pe care am incercat sa o fac cu acest model a fost adaugarea unori straturi **Dropout**. Noul model pe care l-am construit arata in felul urmator:

```
model = Sequential([
    Conv2D(32, 8, activation="relu", padding= "same", input_shape=[32, 32, 1]),
    MaxPooling2D(2),
    Dropout(0.2),
    Conv2D(64, 3, activation="relu", padding="same"),
    Conv2D(64, 3, activation="relu", padding="same"),
    Dropout(0.2),
    MaxPooling2D(2),
    Flatten(),
    Dense(128, activation="relu"),
    Dense(64, activation="relu"),
    Dense(64, activation="relu"),
    Dropout(0.5),
```

```
    Dense(9, activation="softmax")
])
```

Intrucat in incercarile anterioare am observat ca folosind optimizatorul ADAM am obtinut rezultate mai slabe, de aceasta data am incercat sa folosesc din nou optimizatorul SGD.
Dupa ce am antrenat reteaua, timp de 20 de epoci, acuratetea obtinuta pe datele de validare a fost de 86.54% si matricea de confuzie (7)

$$\begin{pmatrix} 486. & 1. & 7. & 3. & 44. & 5. & 4. & 8. & 12. \\ 18. & 461. & 11. & 7. & 13. & 7. & 1. & 8. & 1. \\ 6. & 5. & 451. & 15. & 25. & 19. & 3. & 9. & 0. \\ 12. & 2. & 10. & 480. & 26. & 25. & 4. & 11. & 8. \\ 17. & 1. & 17. & 9. & 487. & 14. & 1. & 3. & 5. \\ 5. & 0. & 12. & 9. & 7. & 516. & 0. & 10. & 2. \\ 26. & 3. & 6. & 5. & 7. & 13. & 504. & 4. & 12. \\ 21. & 11. & 17. & 12. & 23. & 17. & 1. & 415. & 3. \\ 17. & 1. & 0. & 9. & 1. & 7. & 14. & 1. & 527. \end{pmatrix} \tag{7}$$

O alta abordare pe care am incercat-o a fost sa schimb marimea kernelului din primul strat convolutional tot la 3. Dupa ce am antrenat reteaua cu aceasta modificare, timp de 20 de epoci, am obtinut acuratetea de 86.96% pe datele de validare si matricea de confuzie (8)

$$\begin{pmatrix} 487. & 6. & 7. & 3. & 29. & 1. & 8. & 11. & 18. \\ 10. & 478. & 6. & 4. & 6. & 4. & 0. & 16. & 3. \\ 12. & 19. & 428. & 16. & 22. & 19. & 2. & 15. & 0. \\ 8. & 4. & 8. & 492. & 16. & 12. & 8. & 11. & 19. \\ 28. & 0. & 12. & 18. & 466. & 5. & 2. & 16. & 7. \\ 5. & 3. & 10. & 14. & 10. & 496. & 7. & 12. & 4. \\ 19. & 3. & 0. & 3. & 3. & 4. & 543. & 0. & 5. \\ 12. & 8. & 5. & 16. & 11. & 16.712. & 432. & 8. & \\ 24. & 0. & 0. & 4. & 1. & 5. & 15. & 2. & 526. \end{pmatrix} \tag{8}$$

Pentru a obtine o acuratete mai buna, am incercat sa maresc numarul de filtre de pe fiecare strat convolutional, iar la final, sa inlocuiesc cele 3 straturi dense cu un singur strat dens cu 1024 de neuroni. Astfel, am modificat modelul in felul urmator:

```
model = Sequential([
    Conv2D(64, 3, activation="relu", padding= "same", input_shape=[32, 32, 1]),
    MaxPooling2D(2),
    Dropout(0.2),
    Conv2D(128, 3, activation="relu", padding="same"),
    Conv2D(128, 3, activation="relu", padding="same"),
    Dropout(0.2),
    MaxPooling2D(2),
    Flatten(),
    Dense(1024, activation="relu"),
    Dropout(0.5),
```

```
    Dense(9, activation="softmax")
])
```

Dupa antrenarea modelului 20 de epoci, am obtinut acuratetea de 88.32% si urmatoarea matrice de confuzie:

$$\begin{pmatrix} 464. & 5. & 4.74. & 33. & 1. & 16. & 20. & 23. & \\ 5. & 477. & 13. & 4. & 4. & 3. & 4. & 13. & 4. \\ 4. & 6. & 471. & 11. & 15. & 12. & 4. & 10. & 0. \\ 7. & 1. & 10. & 510. & 11. & 10. & 7. & 9. & 13. \\ 20. & 2. & 17. & 15. & 471. & 10. & 4. & 8. & 7. \\ 3. & 0. & 19. & 15. & 7. & 497. & 9. & 9. & 2. \\ 4. & 0. & 4. & 3. & 4.77. & 548. & 6. & 4. & \\ 9. & 7. & 14. & 13. & 8. & 10. & 6. & 450. & 3. \\ 9. & 1. & 0. & 7. & 2.74. & 23. & 3. & 528. & \end{pmatrix} \quad (9)$$

3 Solutia finala

In cele din urma, in solutia finala am parcurs urmatorii pasi:

3.1 Citirea datelor

Am citit atat imaginile de train si de validare, impreuna cu etichetele corespunzatoare, cat si imaginile de test.

3.2 Normalizarea datelor

Am scris o functie care normalizeaza imaginile de train, de validate si de test. Am decis sa aleg normalizarea standard, adica pentru fiecare x_i sa scad media si sa impart la abaterea standard. Pentru a face acest lucru am folosit obiectul **StandardScaler** din **Scikit-learn**.

3.3 Constructia modelului

In solutia finala am decis sa folosesc o retea neuronala convolutionala. Am construit reteaua in felul urmator:

```
model = Sequential([
    Conv2D(64, 3, activation="relu", padding= "same", input_shape=[32, 32, 1]),
    Conv2D(64, 3, activation="relu"),
    MaxPooling2D(2),
    Dropout(0.25),
    Conv2D(128, 3, activation="relu", padding="same"),
    Conv2D(128, 3, activation="relu"),
    MaxPooling2D(2),
    Dropout(0.25),
```

```
    Flatten(),
    Dense(1024, activation="relu"),
    Dropout(0.6),
    Dense(9, activation="softmax")
])
```

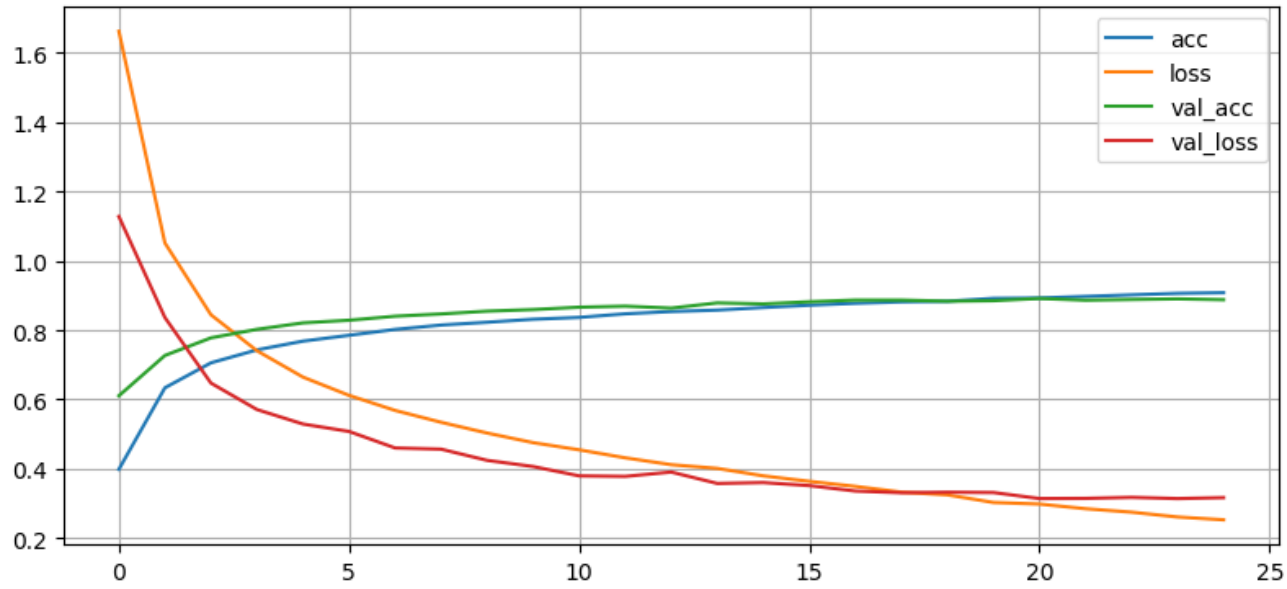
La compilarea modelului am decis sa folosesc functia de loss **Sparse Categorical Crossentropy**, optimizatorul **Stochastic Gradient Descent** si ca metrica - acuratetea.
Am antrenat modelul pe datele de antrenare timp de 35 de epoci.
In urma antrenarii modelului am obtinut urmatoarele rezultate:

- Acuratetea pe datele de validare: 88.78%
- Matricea de confuzie:

$$\begin{pmatrix} 486. & 4. & 4. & 6. & 23. & 3. & 11. & 13. & 20. \\ 6. & 493. & 6. & 3. & 3. & 3. & 0. & 10. & 3. \\ 7. & 13. & 457. & 14. & 15. & 18. & 3. & 6. & 0. \\ 9. & 6. & 12. & 502. & 13. & 7. & 7. & 8. & 14. \\ 28. & 9. & 11. & 16. & 458. & 7. & 4. & 11. & 10. \\ 3. & 0. & 15. & 13. & 7. & 500. & 9. & 10. & 4. \\ 8. & 1. & 1. & 2. & 1. & 3. & 557. & 0. & 7. \\ 16. & 12. & 8. & 15. & 6. & 9. & 9. & 443. & 2. \\ 9. & 0. & 0. & 3. & 1. & 3. & 16. & 2. & 543. \end{pmatrix}$$

3.4 Reprezentare grafica a rezultatelor

In varianta finala am decis plotarea acuratetii si a functiei de loss pe datele de antrenare si de validare, pentru a putea observa daca se intampla fenomenul de overfit sau de underfit.
Graficul rezultat este urmatorul:



3.5 Afisarea rezultatelor in fisier

Ultimul lucru pe care l-am facut in solutia finala a fost sa generez fisierul cu predictiile. Pentru fiecare imagine am ales ca predictie label-ul cel mai probabil si am scris in fisierul de output.

4 Concluzii

In concluzie, in urma incercarii mai multor variante de modele, cum ar fi: Masini cu vectori suport, Retele neuronale si retele neuronale convolutionale am reusit sa gasesc un model care, in urma antrenarii, a reusit sa clasifice imaginile din datele de validare cu o acuratete de !!%