ETAPA 2 - Marcu Genţiana Adeane, 343C2

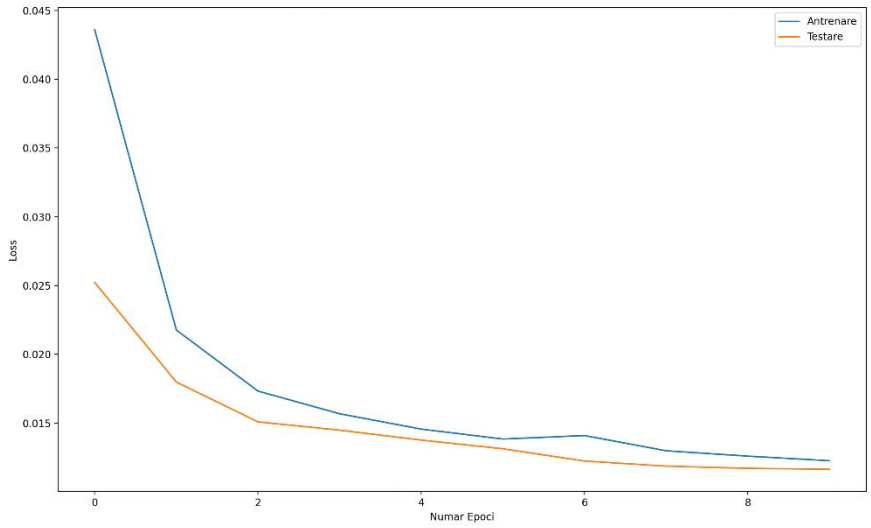
Dupa cum se specifica in enunt, avand in vedere ca datele pe care le avem la antrenare sunt relativ putine, este nevoie de o metode de rugularizare. Pentru fiecare model in parte, pentru fiecare dintre cele 2 seturi de date cu aritmii(MITBIH si PTBDB) am ales adaugarea de straturi de Droupout de 20%.

In afara de straturile de Dropout pentru fiecare model am adaugat 2 straturi specifice modelului Avand in vedere dimensiunile celor 2 fisiere, am ales dimensiunea primului strat de 64, iar dimensiunea celui de al doilea 128. Optimizatorul ales in antrenarea modelelor este Adam cu parametrii default.

Pentru evaluarea curbelor de los am folosit metrica mean_squared_error. Modelele au fost evaluate pe 10 epoci. Pentru setul de date MITBIH am folosit batch_size = 128, iar pentru setul de date PTBDB, avand in vedere ca acesta contine un numar de date considerabil mai mic am folosit batch_size = 32.

Acestea sunt reazultatele obtinute in urma rularii:

- MITBIH arhitectura de tip MLP

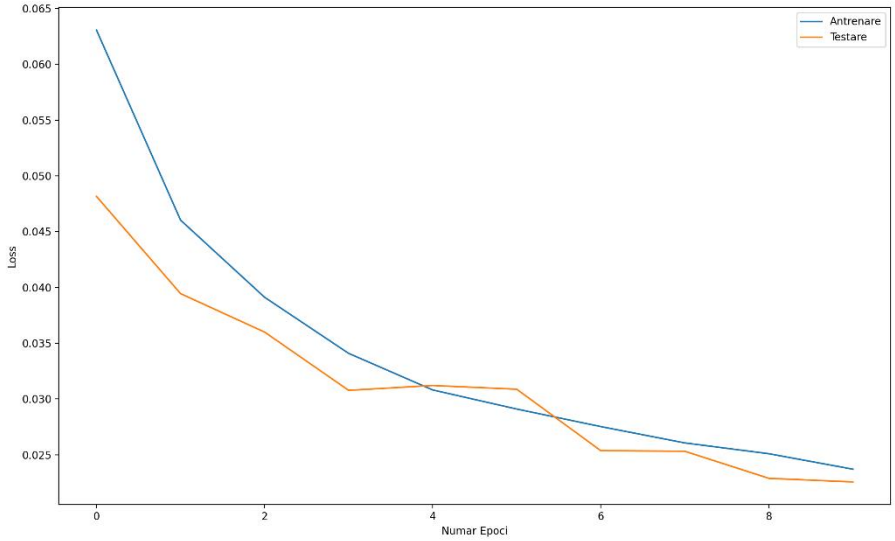


Acuratete, tabel si matrice de confuzie:

```
0.9666088068700895
[[18076     6    23     5     8]
 [  240   306     9     0     1]
 [  191     1  1221    29     6]
 [   52     0    10   100     0]
 [  141     0     9     0  1458]]
              precision    recall  f1-score   support

         0.0       0.97      1.00      0.98     18118
         1.0       0.98      0.55      0.70       556
         2.0       0.96      0.84      0.90      1448
         3.0       0.75      0.62      0.68       162
         4.0       0.99      0.91      0.95      1608
```

- PTBDB arhitectura de tip MLP

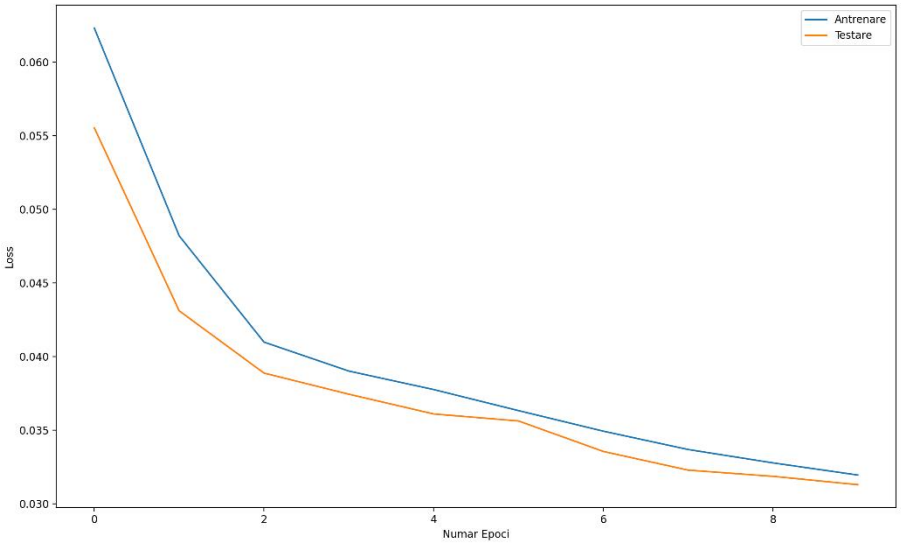


Acuratete, tabel si matricea de confuzie:

```
0.9271727928546891
[[ 749   85]
 [ 127 1950]]
              precision    recall  f1-score   support

         0.0       0.86      0.90      0.88       834
         1.0       0.96      0.94      0.95      2077
```

- MITBIH arhitectura de tip convolutionala

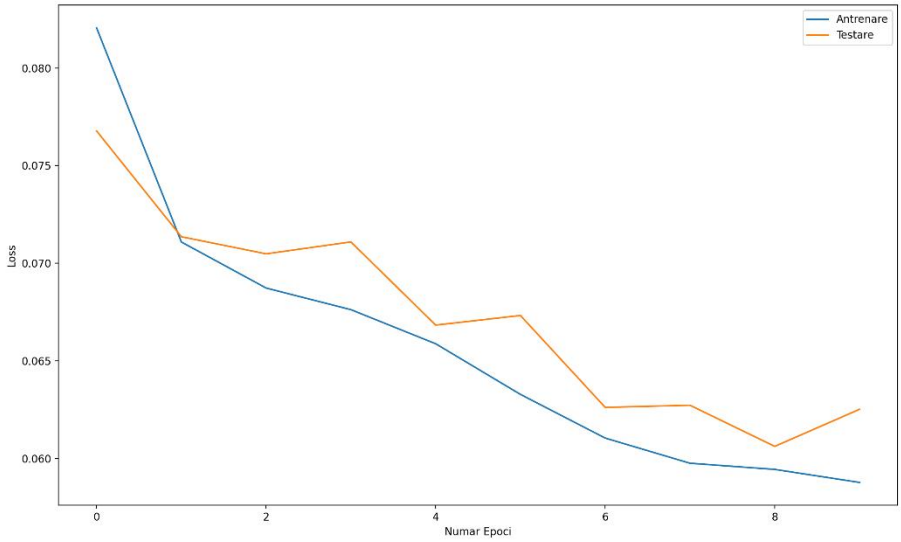


Acuratete, tabel si matricea de confuzie:

```
0.9073634204275535
[[17927     0   153     0    38]
 [  505     0    50     0     1]
 [  848     0   579     0    21]
 [  160     0     2     0     0]
 [  196     0    54     0  1358]]
              precision    recall  f1-score   support

         0.0       0.91      0.99      0.95     18118
         1.0       0.00      0.00      0.00       556
         2.0       0.69      0.40      0.51      1448
         3.0       0.00      0.00      0.00       162
         4.0       0.96      0.84      0.90      1608
```

- PTBDB arhitectura de tip convolutionala

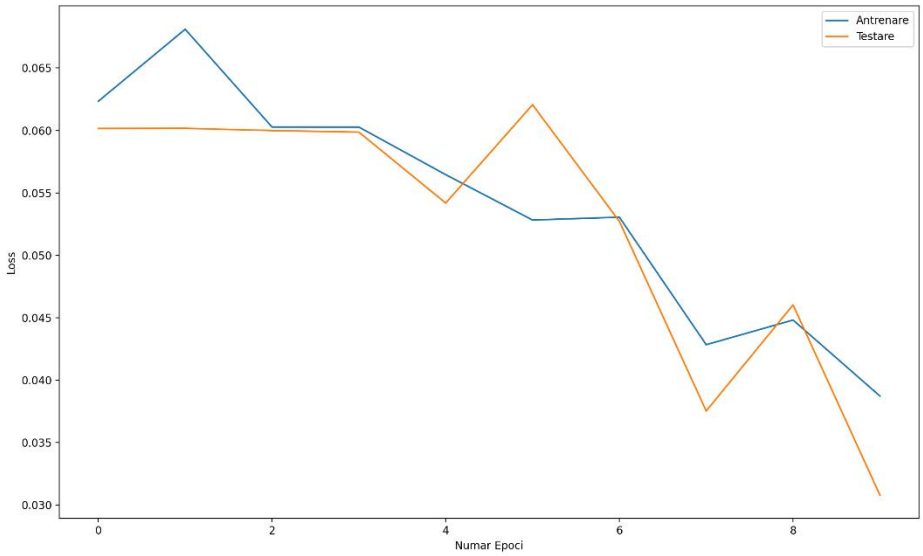


Acuratete, tabel si matricea de confuzie:

```
0.7746478873239436
[[ 530  304]
 [ 352 1725]]
              precision    recall  f1-score   support

         0.0       0.60      0.64      0.62       834
         1.0       0.85      0.83      0.84      2077
```

- MITBIH arhitectura de tip recurent(LSTM)

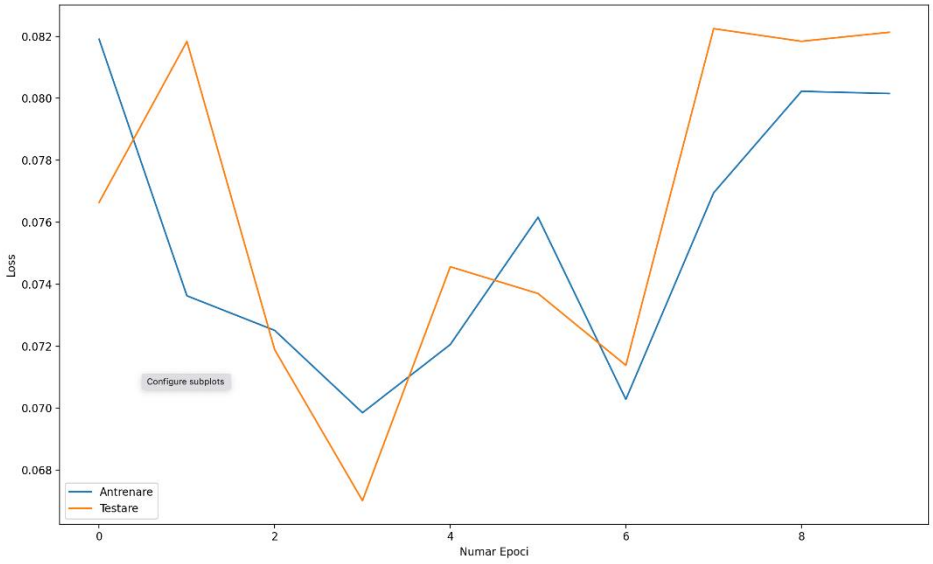


Acuratete, tabel si matricea de confuzie:

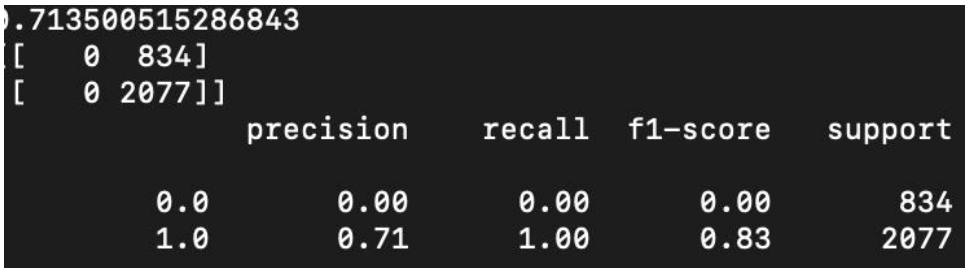
```
0.9110177233692673
[[18001     0    72     0    45]
 [  486     0    69     0     1]
 [  607     0   683     0   158]
 [  150     0    12     0     0]
 [  316     0    32     0  1260]]
              precision    recall  f1-score   support

         0.0       0.92      0.99      0.96     18118
         1.0       0.00      0.00      0.00       556
         2.0       0.79      0.47      0.59      1448
         3.0       0.00      0.00      0.00       162
         4.0       0.86      0.78      0.82      1608
```

- PTBDB arhitectura de tip recurent(LSTM)



Acuratete, tabel si matricea de confuzie:



```
.713500515286843
[   0  834]
[   0 2077]]
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00       834
         1.0       0.71      1.00      0.83      2077
```

Se observa ca, pentru ambele seturi de date, pentru configuratia aleasa, arhitectura de tip MLP este cea mai eficienta, si din punctul de vedere al performantei(96% pe MITBIH si 92% pe PTBDB) dar si din punct de vedere al timpului de rulare. Acest lucru se poate observa si pe curbele de loss. Pentru ultimele 2 modele, pentru setul de date PTBDB se poate obtine o acuratete de peste 90% scazand betch_sizeul la 16.