Tema 2 la Învățare Automată: Predicții pe electrocardiograme

Alexandru-Mihai-Iulian Buzea, 341C3

May 26, 2024

Abstract

Acest document prezintă rezultatele analizei pe un dataset alcătuit din date secvențiale (electrocardiograme) pentru prezicerea anomaliilor cardiace. Sunt studiate 2 seturi de date: setul de date Patients din etapa anterioară, folosind de această dată modele din categoria rețelelor neurale (Multi-Layer Perceptron - MLP), precum și un nou set de date, PTB, diferit prin faptul că datele de intrare sunt serii de timp (și nu date tabelare, cum am avut în tema 1).

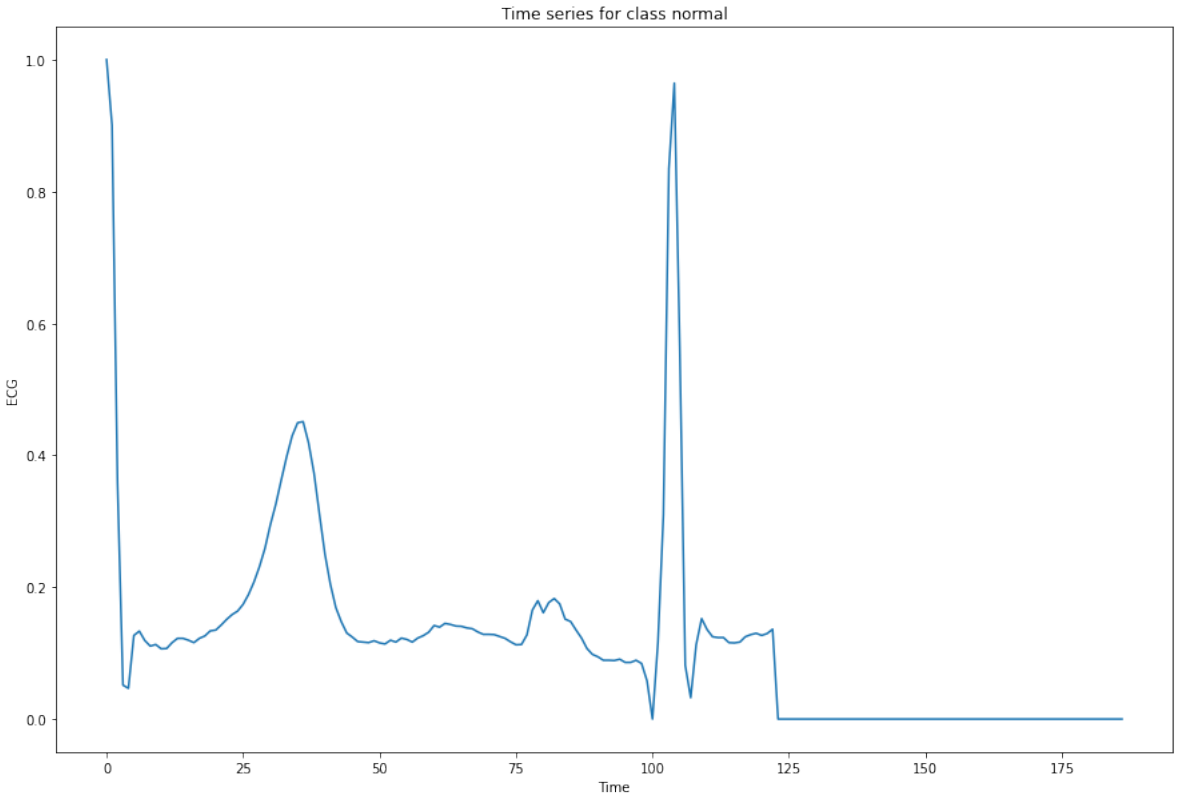
1 Introducere

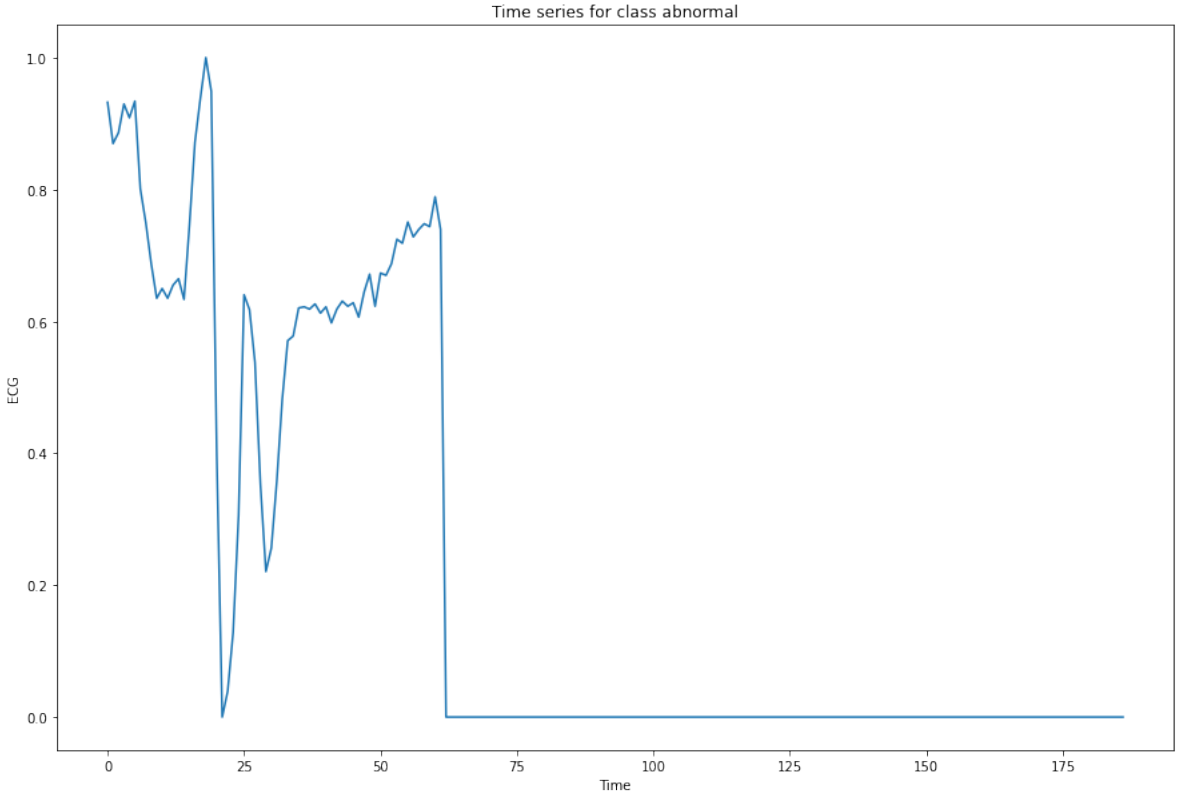
Am realizat analiza datelor pentru noul dataset (PTB - cu variantele sale normal și abnormal pe care le-am reunit în cadrul task-ului de clasificare) și apoi am realizat două tipare generale de arhitecturi: o rețea Multi-Layer Perceptron (MLP) pentru predicția clasei în cazul dataset-ului Patients și o arhitectură convoluțională custom pentru dataset-ul PTB alcătuit din electrocardiograme (serii de timp).

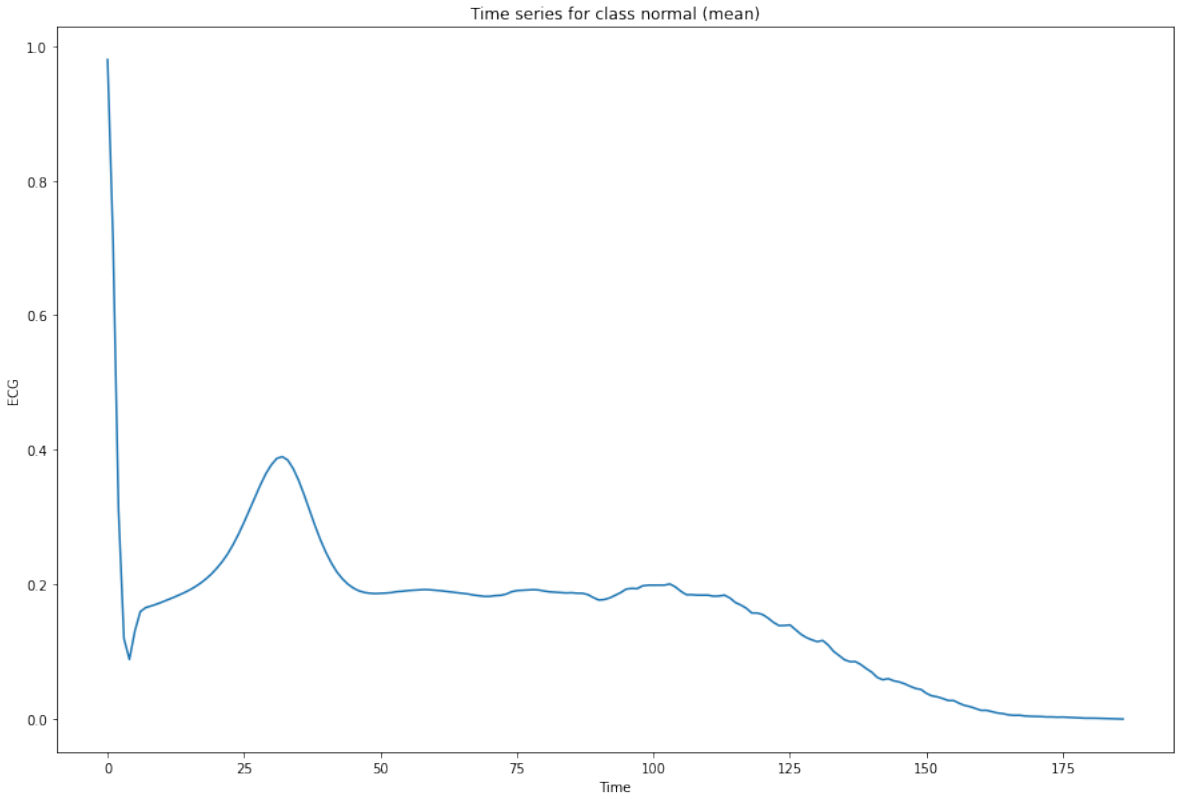
1.1 Explorarea datelor secvențiale

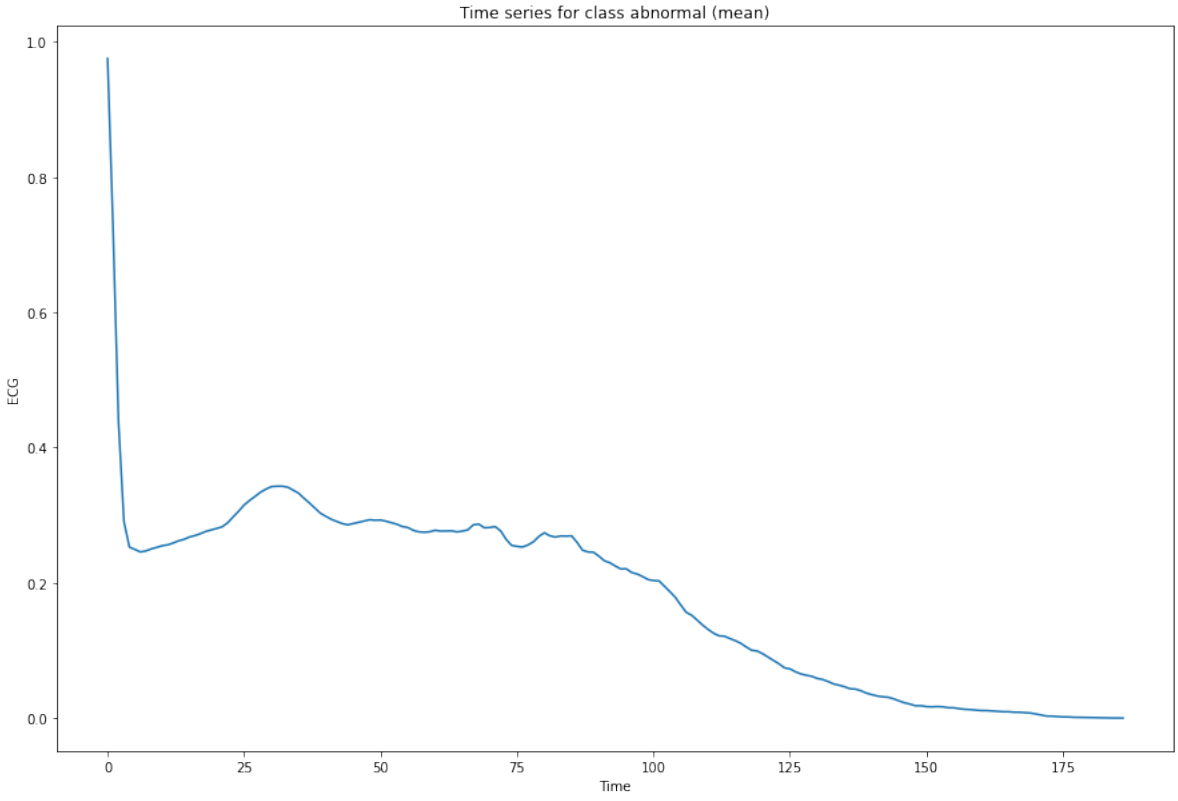
Am realizat explorarea datelor secvențiale din dataset-ul PTB prin analizarea unui exemplu de serie de timp (din fiecare clasă), analiza mediei seriei de timp per clasă și analiza deviației standard a seriei de timp per clasă. Întrucât la o primă intuiție seriile de timp ar fi trebuit să fie periodice, am realizat și o analiză în domeniul frecvență în care am trecut cu ajutorul transformatei Fourier rapide (FFT - transformare disponibilă în numpy).

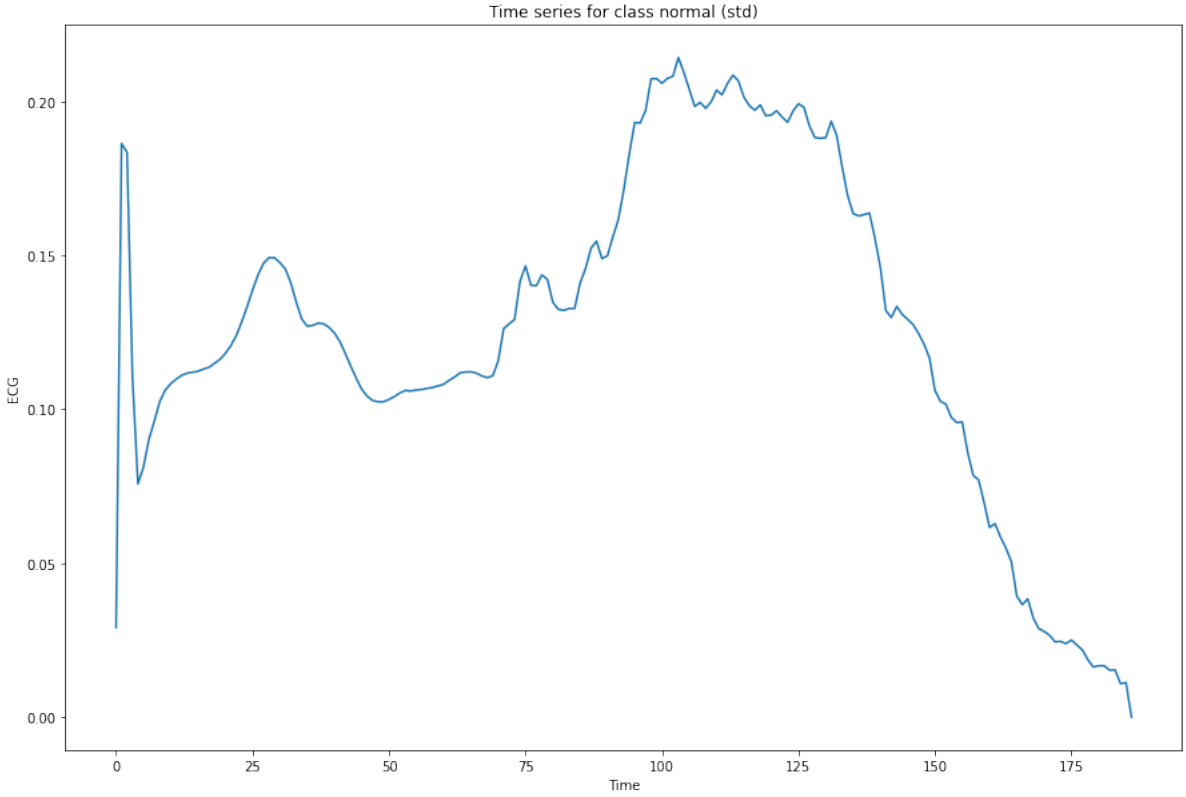
Se observă că analiza mediei are avantajul de a ne scăpa de zgomot (întrucât mediem seriile de timp pe un număr mare de exemple), dar totuși este importantă și analiza datapoint-urilor individuale (a se observa deviația standard având valori chiar și de 0.25 în anumite puncte, o valoare semnificativă). Analiza în domeniul frecvență ne relevă de asemenea câteva intuiții: componenta constantă (frecvență 0) este cea care ne apare cel mai des/în proporție cea mai mare în semnalul analizat, iar componentele de frecvență mare sunt practic absente. O diferență importantă însă între analiza unei electrocardiograme normale și a uneia anomalice este că în cadrul anomaliilor avem și componente de frecvență scăzută care nu există în mod normal.

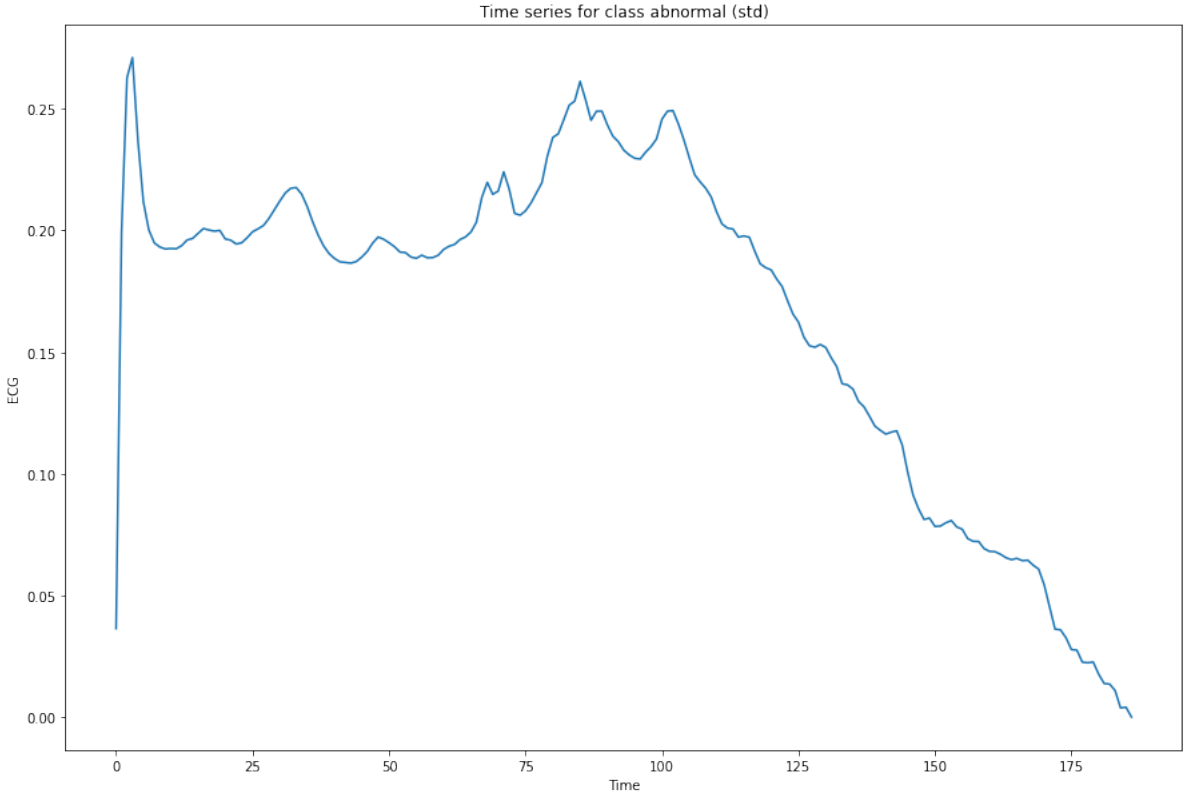

Time series for class normal
ECG
Time
1.0
0.8
0.6
0.4
0.2
0.0
0
25
50
75
100
125
150
175

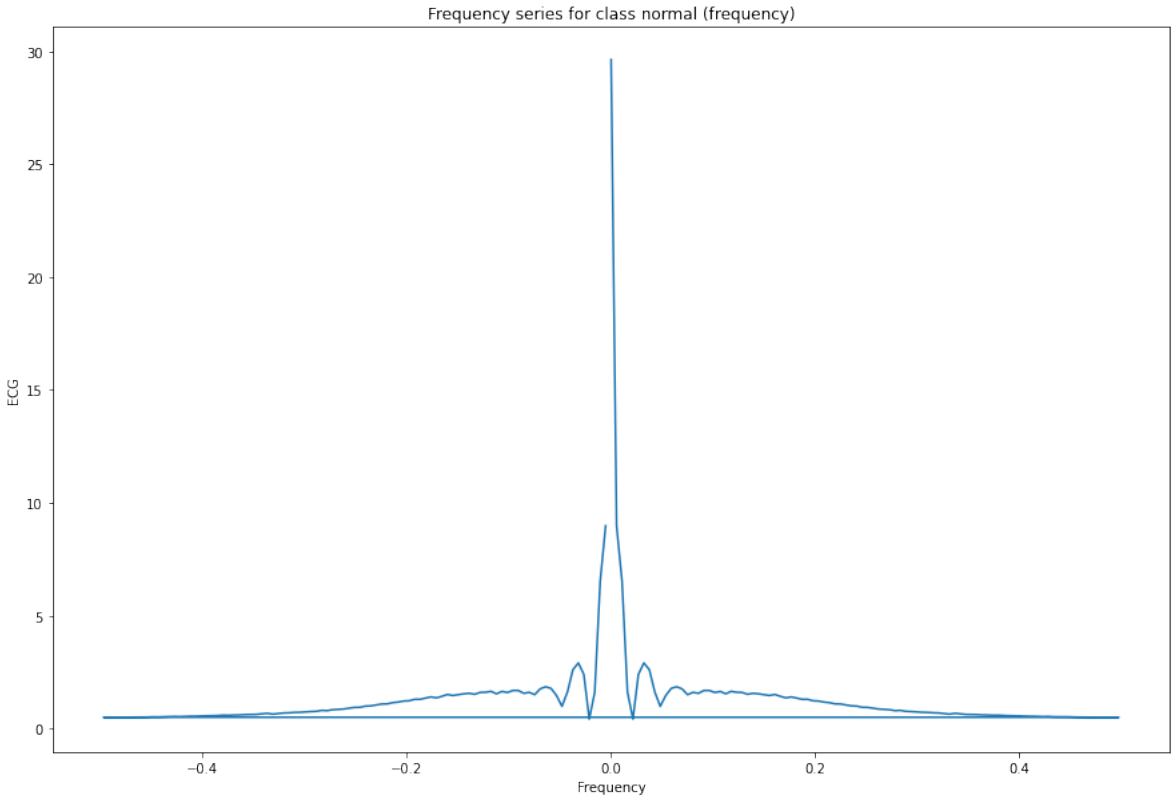

Time series for class abnormal
ECG
Time
1.0
0.8
0.6
0.4
0.2
0.0
0
25
50
75
100
125
150
175

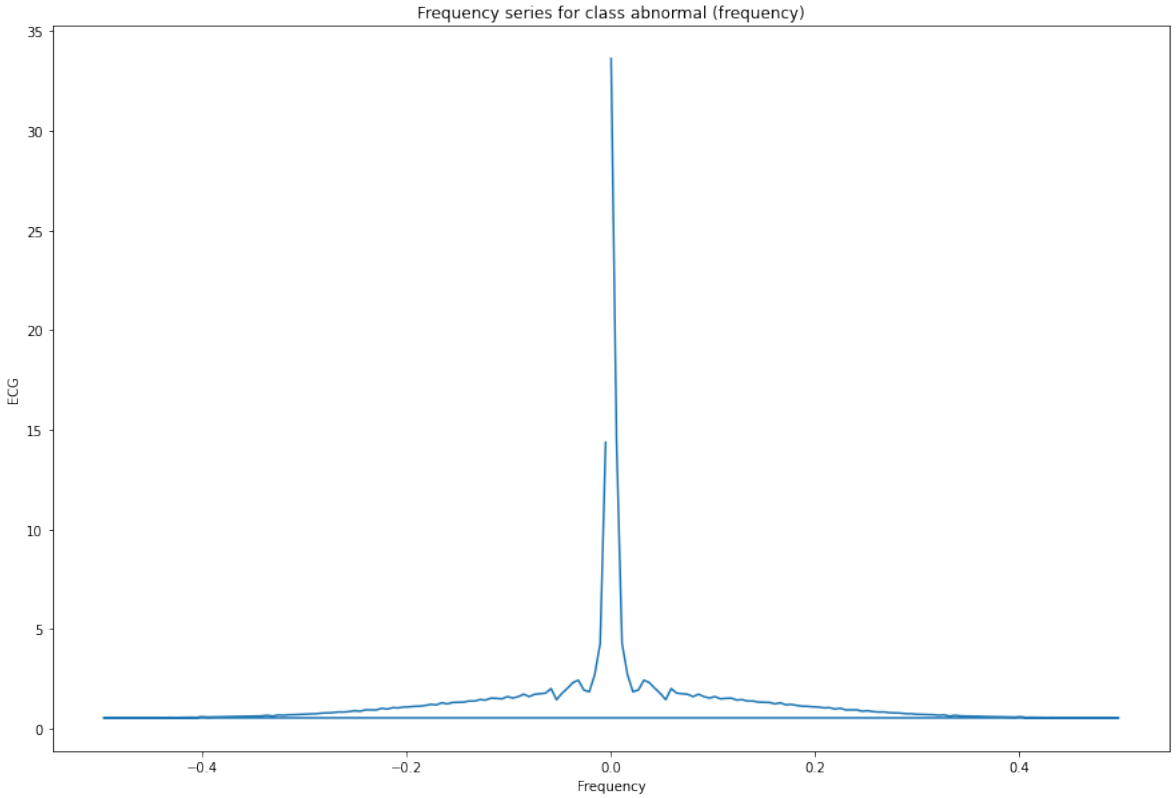

Time series for class normal (mean)
ECG
Time
1.0
0.8
0.6
0.4
0.2
0.0
0
25
50
75
100
125
150
175


Time series for class abnormal (mean)
ECG
Time
1.0
0.8
0.6
0.4
0.2
0.0
0
25
50
75
100
125
150
175


Time series for class normal (std)
ECG
Time
0.00
0.05
0.10
0.15
0.20
0
25
50
75
100
125
150
175

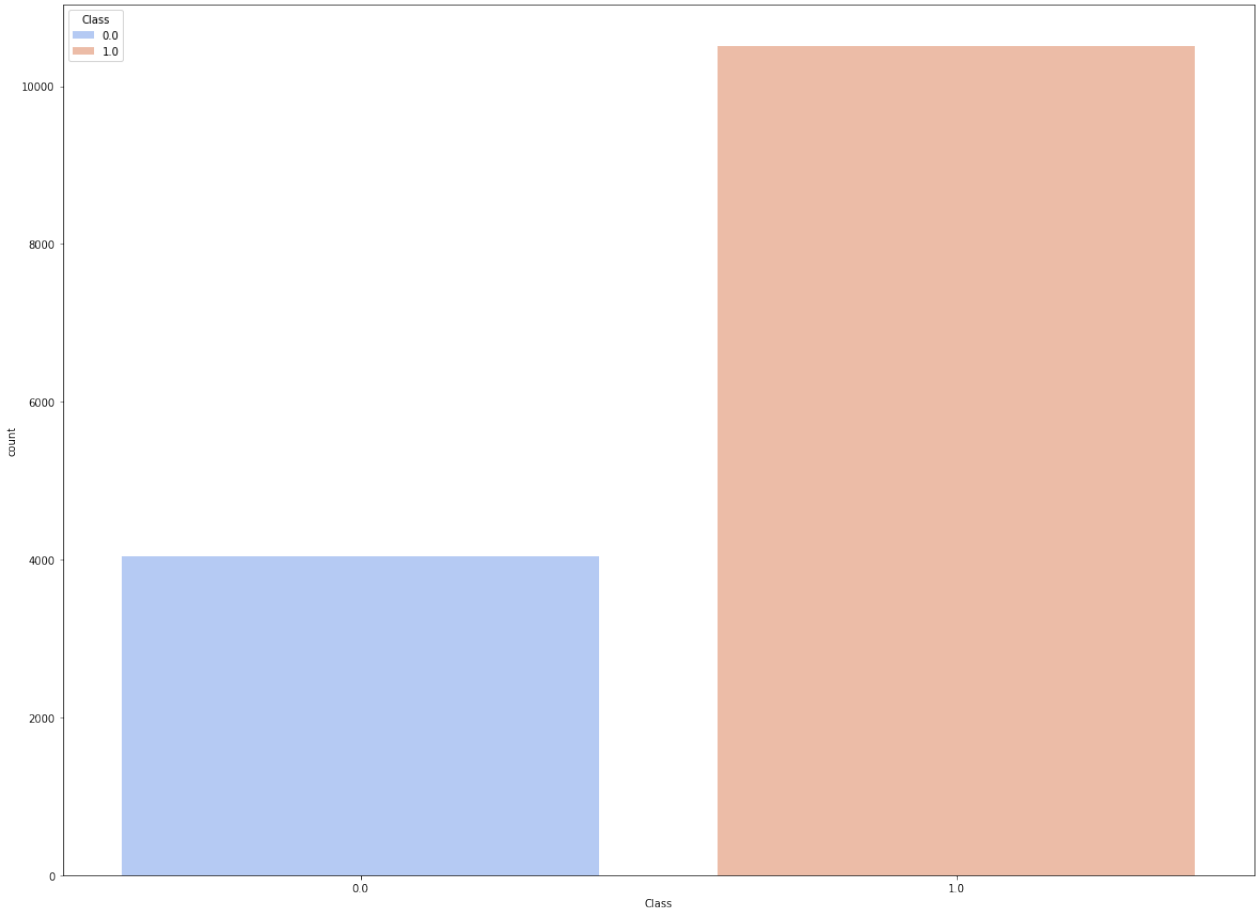

Time series for class abnormal (std)
ECG
Time
0.00
0.05
0.10
0.15
0.20
0.25
0
25
50
75
100
125
150
175


Frequency series for class normal (frequency)
ECG
Frequency
30
25
20
15
10
5
0
−0.4
−0.2
0.0
0.2
0.4


Frequency series for class abnormal (frequency)
ECG
Frequency
35
30
25
20
15
10
5
0
−0.4
−0.2
0.0
0.2
0.4

1.2 Echilibrul claselor

Din punctul de vedere al claselor, avem de-a face cu un task de clasificare binară, existând numai 2 clase: normal și anormal.



Observăm că clasele sunt destul de disproporționate (skewed classes) deci realizarea clasificării va fi o provocare pentru modelul ales (vom tinde să clasificăm seriile de timp mai degrabă ca anormale decât ca serii normale).

2 Explorarea modelelor bazate pe rețele neurale

2.1 Dataset-ul Patients

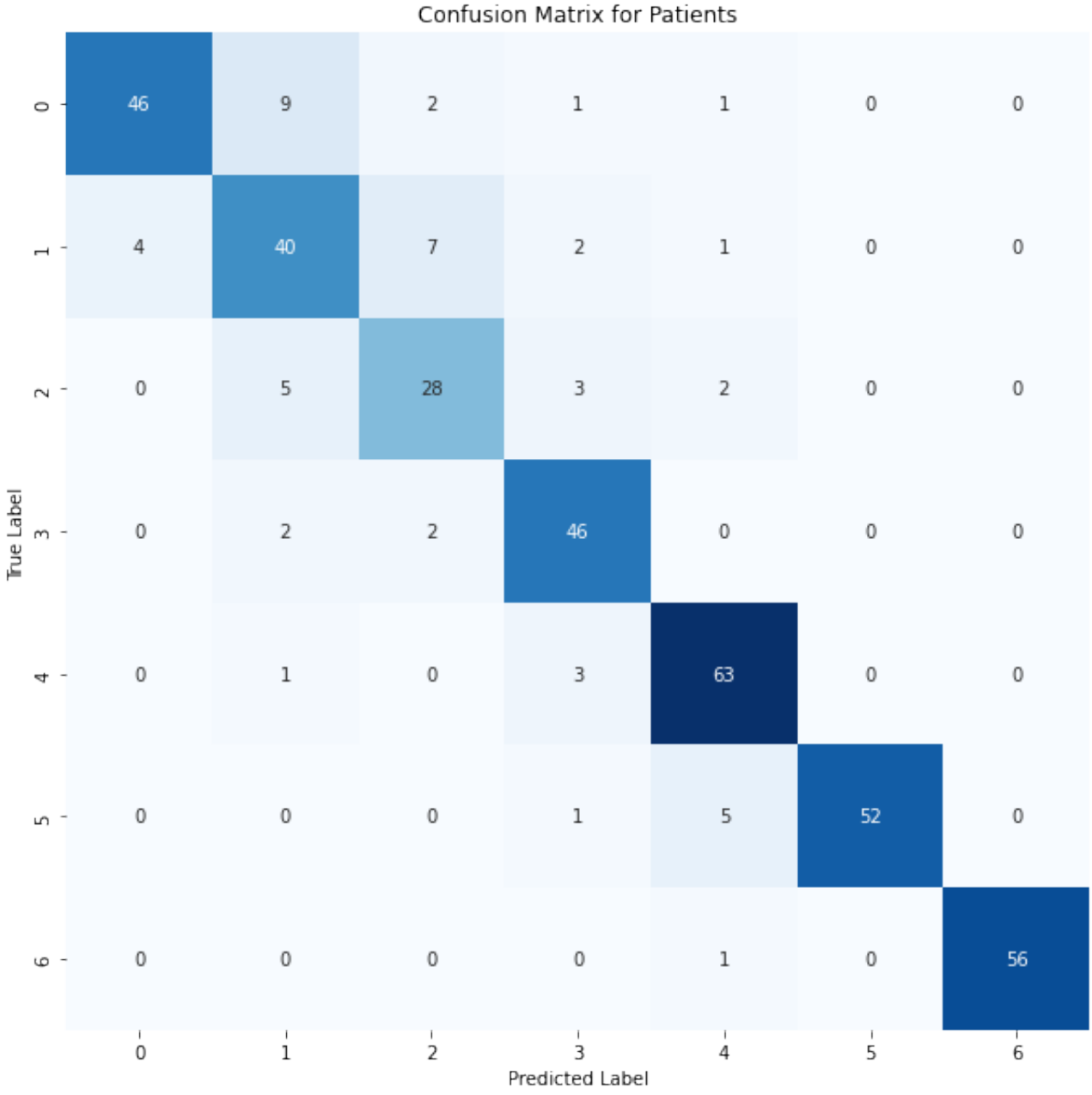
Pentru analiza dataset-ului Patients am folosit un model de rețea neurală de tip Multi-Layered Perceptron. Acest model de rețea implică existența unor straturi liniare între care se aplică o funcție de activare. Am folosit PyTorch drept framework de construcție al rețelei.

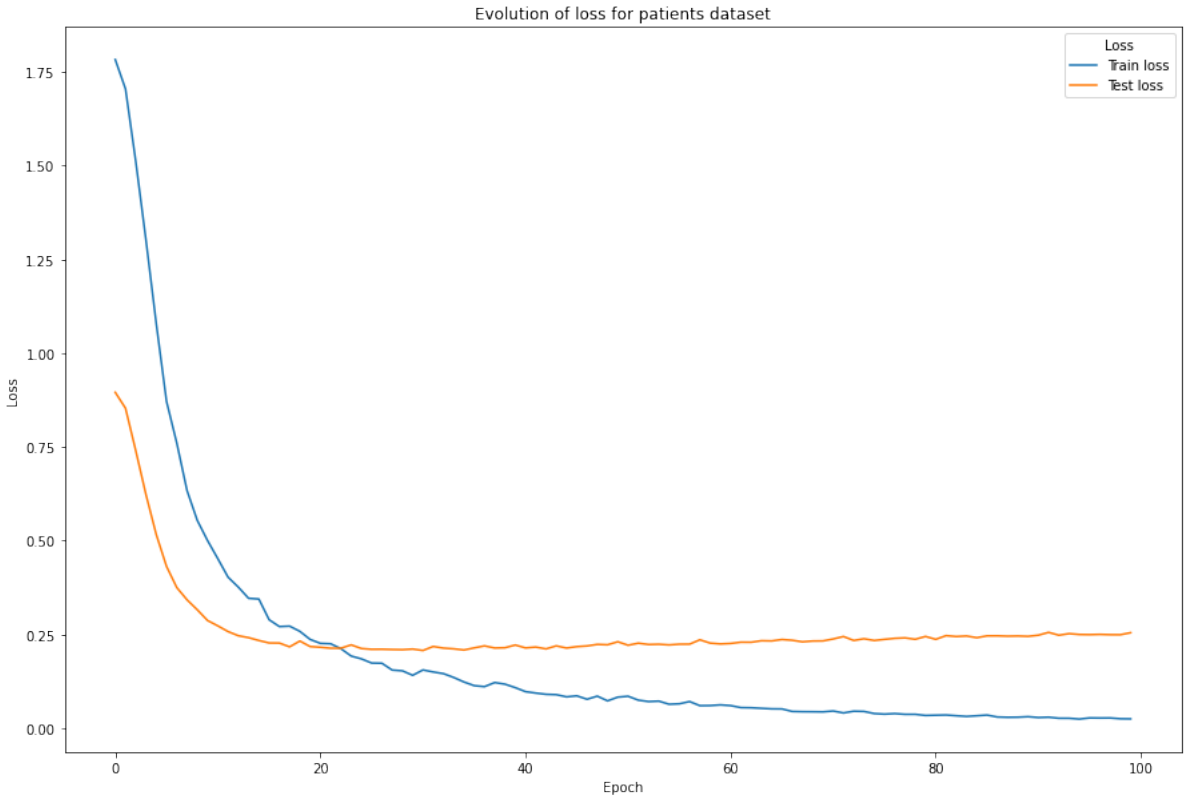
Am studiat mai multe arhitecturi particulare prin implementarea unei unice arhitecturi, foarte generale, caracterizată de următorii hiperparametri: numărul de unități din straturile ascunse (am presupus că toate straturile ascunse au același număr de unități), rata de învățare, constanta/parametrul de regularizare, optimizatorul (am ales între Adam, și respectiv SGD - Stochastic Gradient Descent cu learning rate scheduler), variația relativă a ratei de învățare pentru scheduler (am ales o scădere exponențială, acest hiperparametru subunitar fiind baza termenului ce produce scăderea), dimensiunea unui batch (luată deseori, din motive de optimizare, ca o putere de 2) și nu în ultimul rând numărul de straturi ascunse (am variat între 2 și 4 straturi ascunse).

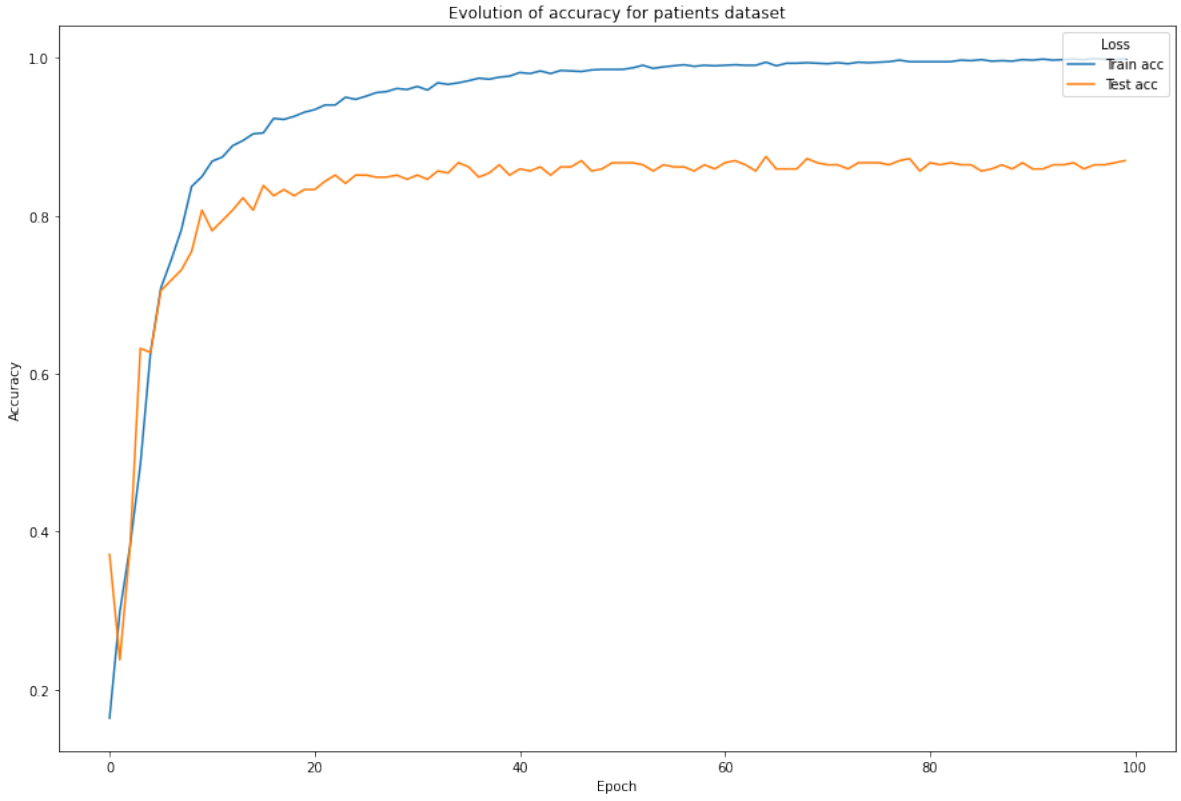
Rezultatele obținute prin varierea hiperparametrilor și cel mai bun set de arhitectură pot fi obținute din documentul de aici.

Pentru modelul cel mai bun, am determinat metricile de Precision, Recall și F1 score, matricea de confuzie și am realizat două reprezentări grafice pentru evoluția acurateței și a loss-ului (în acest caz, CrossEntropyLoss).

```
Label 0: P = 0.92, R = 0.7796610169491526, F1 = 0.8440366972477064
Label 1: P = 0.7017543859649122, R = 0.7407407407407407, F1 = 0.7207207207207207
Label 2: P = 0.717948717948718, R = 0.7368421052631579, F1 = 0.7272727272727273
Label 3: P = 0.8214285714285714, R = 0.92, F1 = 0.8679245283018867
Label 4: P = 0.863013698630137, R = 0.9402985074626866, F1 = 0.9
Label 5: P = 1.0, R = 0.896551724137931, F1 = 0.9454545454545454
Label 6: P = 1.0, R = 0.9824561403508771, F1 = 0.9911504424778761
```







2.2 Dataset-ul PTB

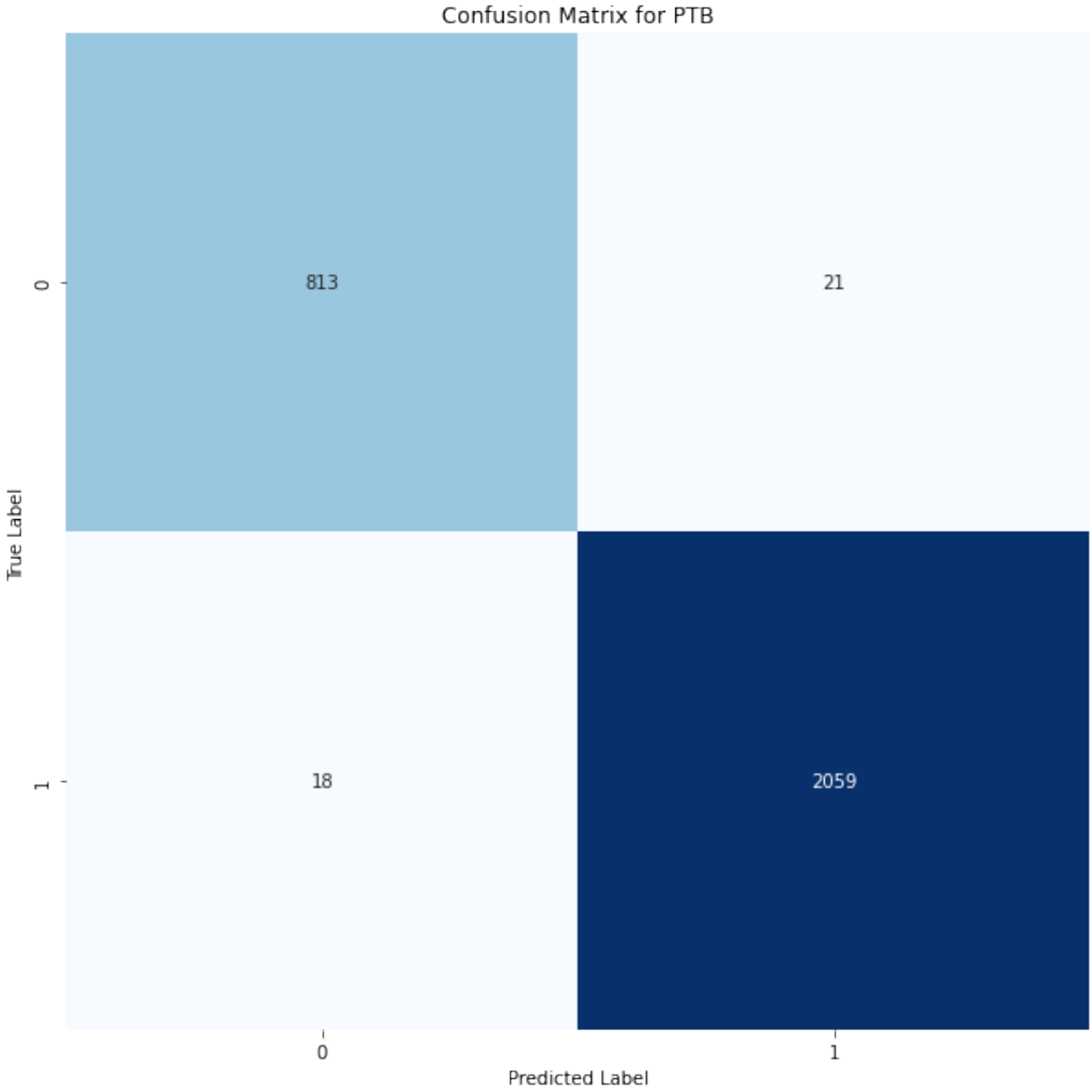
Pentru analiza acestui dataset, întrucât vorbim despre un model de clasificare, am ales o arhitectură convoluțională cu 2 straturi de convoluție, urmată de un număr variabil de straturi liniare caracterizate de aceiași parametri enunțați la secțiunea 2.1. În plus, am adăugat ca hiperparametri de variat

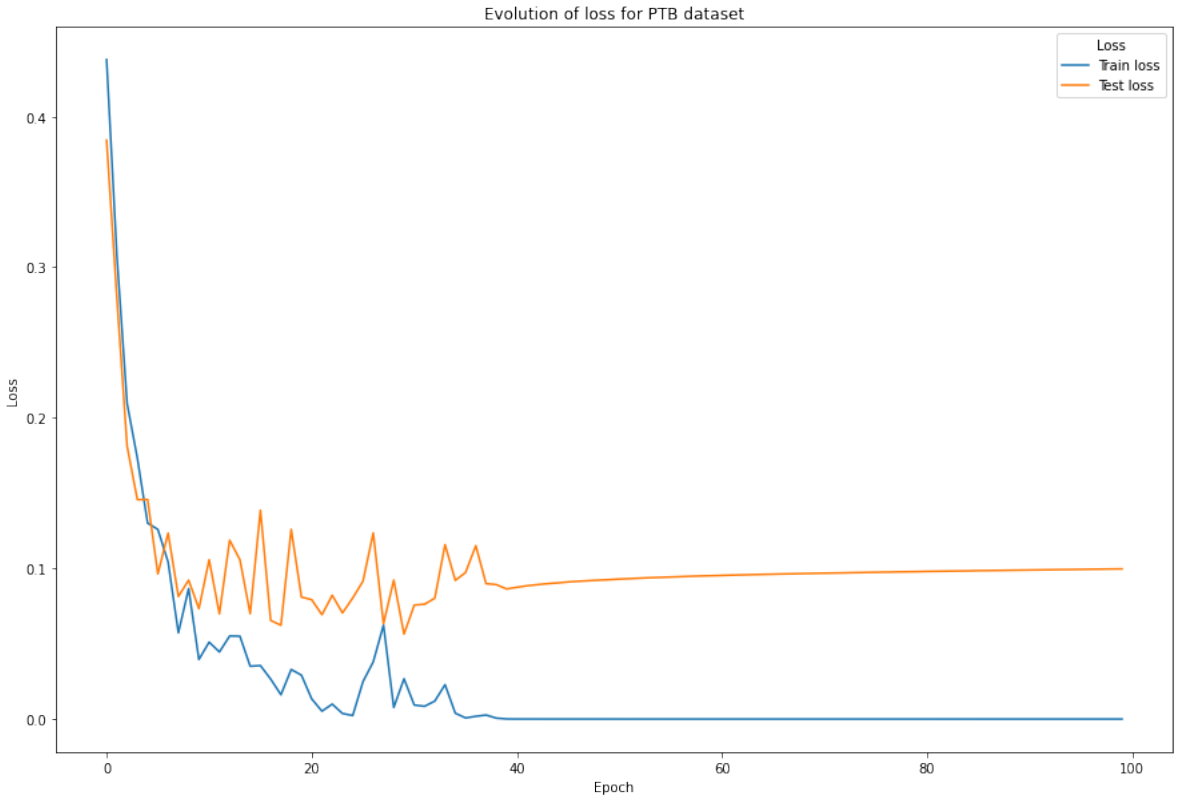
dimensiunile celor 2 kernele unidimensionale (numere impare), precum și numărul de map-uri obținute în fiecare feature map (6, respectiv 16 - am ales aceste valori cu rezultate foarte bune și de aceea nu le-am variat efectiv în explorarea noastră).

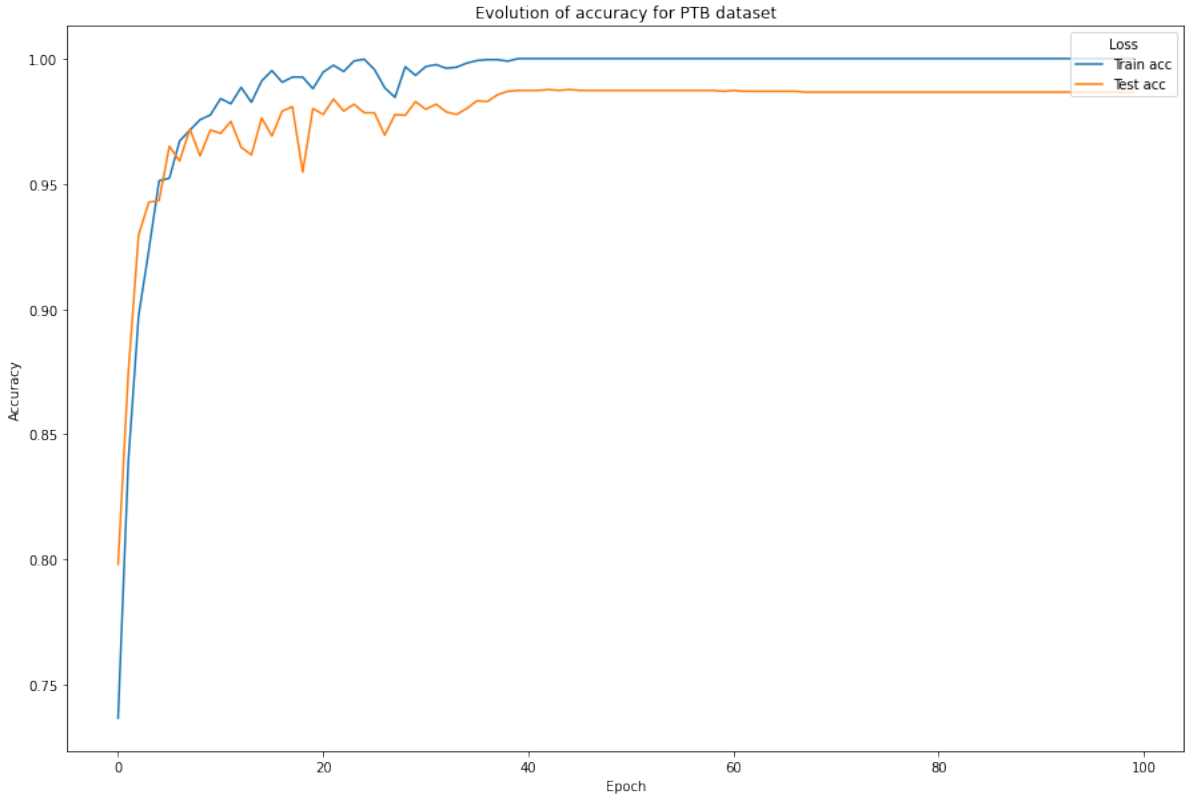
Rezultatele obținute prin varierea hiperparametrilor și cel mai bun set de arhitectură pot fi obținute din documentul de aici.

Pentru modelul cel mai bun, am determinat metricile de Precision, Recall și F1 score, matricea de confuzie și am realizat două reprezentări grafice pentru evoluția acurateței și a loss-ului (în acest caz, CrossEntropyLoss).

```
P = 0.9899038461538462
F1 = 0.9906182343035843
R = 0.9913336543090997
```




Evolution of loss for PTB dataset
Loss
Train loss
Test loss
0.4
0.3
0.2
0.1
0.0
Loss
0
20
40
60
80
100
Epoch


Evolution of accuracy for PTB dataset
Loss
Train acc
Test acc
1.00
0.95
0.90
0.85
0.80
0.75
Accuracy
0
20
40
60
80
100
Epoch

3 Observatii finale

- Optimizatorul ales depinde de modelul analizat. Dacă este un model simplu, pentru care ne permitem rulări succesive într-un timp bun, un optimizator clasic cu un learning rate scheduler bine ales va depăși performanța de antrenare a unui optimizator adaptiv (Adam).
- Dimensiunea batch-ului contează în procesul de învățare: pentru batch-uri mici, învățarea este mai rapidă, deși apare un compromis în ceea ce privește acuratețea globală. Pentru batch-uri mai mari observăm o creștere, în general, a acurateței.
- Rata de învățare depinde foarte mult de tipul optimizatorului. Unele optimizatoare precum Adam lucrează bine cu o rată de învățare mai mică (de ordinul 10^{-3}), în timp ce prezența unui learning rate scheduler ne permite să începem cu o rată de învățare mai mare (chiar de ordinul 10^{-1}) pentru că în timp aceasta va scădea și vom ajunge la convergență.
- Este important totuși să alegem ca rata de învățare să scadă lent în timp. O rată de învățare care scade rapid nu va conduce la un optim și va duce la performanțe slabe.
- Dimensiunea modelului (numărul de straturi ascunse) și numărul de unități neuronale determină complexitatea modelului. Un model mai complex va putea atinge performanțe mai bune, având însă riscul de overfitting și un timp de antrenare mai ridicat.
- Numărul de epoci mai mare va aduce în general performanțe mai bune, însă de la un anumit pas acuratețea modelului nu va mai varia mult după un anumit număr de epoci, caz în care putem opri antrenarea modelului.

4 Feedback

A fost o temă interesantă, cu ajutorul căreia am reușit să îmi exersez programarea Python și cu ajutorul căreia m-am familiarizat cu framework-ul PyTorch (cu care lucrasem anterior, dar nu aveam multă experiență).