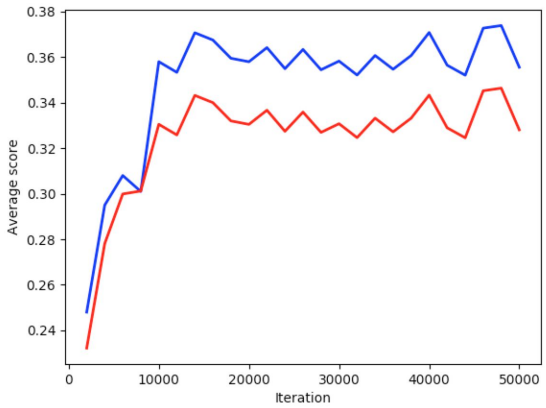
Grafice Retele Neurale

ROSU - test acc
ALBASTRU - train acc

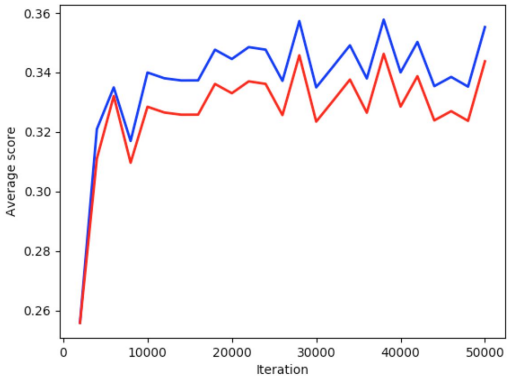
Arhitectura I
nn = FeedForward([LinearizeLayer(32, 32, 3), FullyConnected(32*32*3, 300, identity), Tanh(), FullyConnected(300, 10, identity), SoftMax()])

Train vs Test: FC 300 - 10

LR 0.001



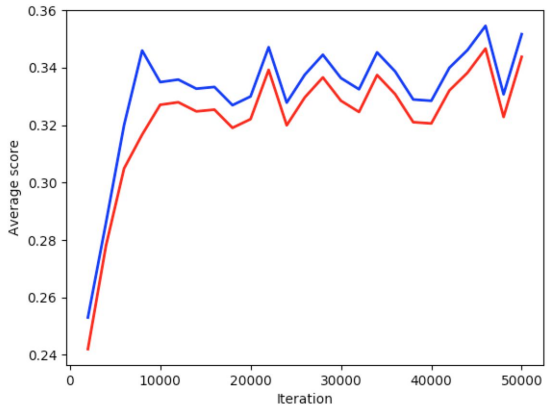
LR 0.005

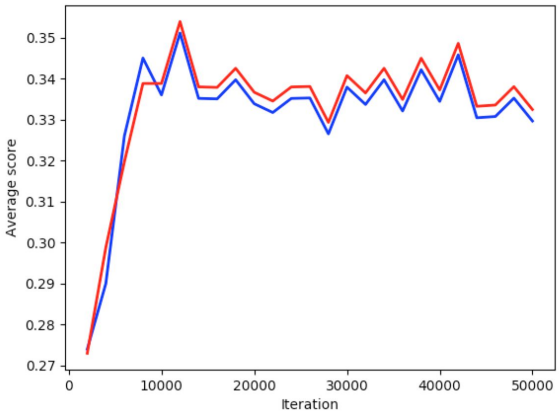


Train vs Test: FC 150 - 10

LR 0.001

LR 0.005

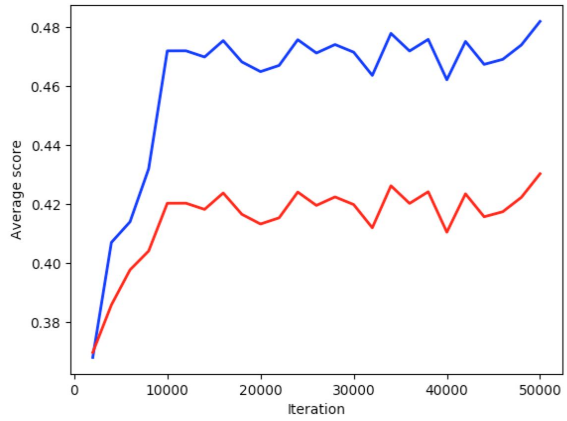


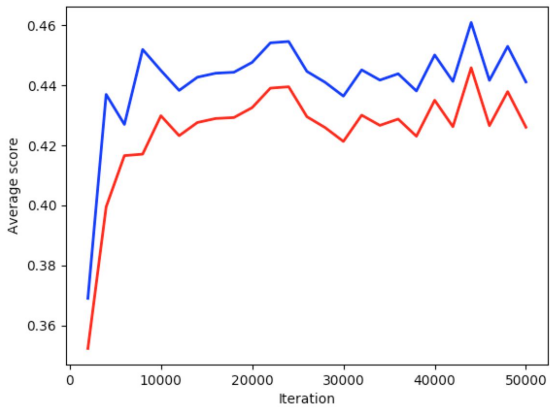


Best HighScores: FC 300

LR 0.001

LR 0.005





Concluzie:

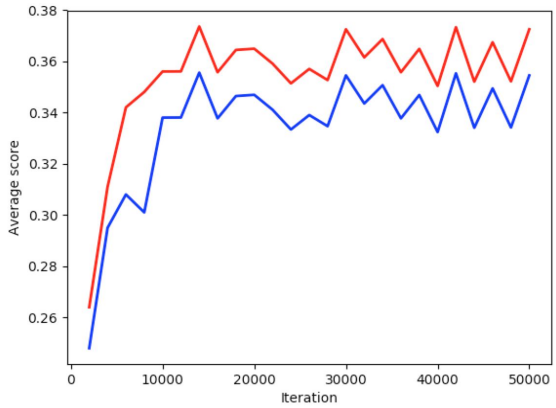
FC 150 fata de 300 este putin mai slab

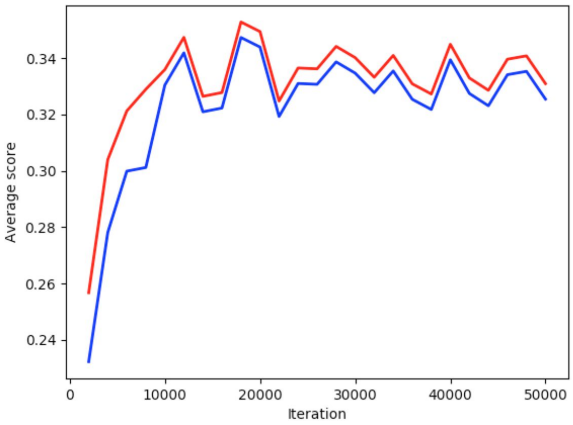
Influenta momentului

Train acc:

Test acc

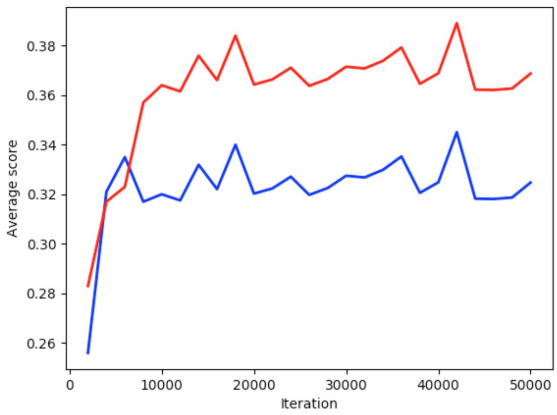
LR 0.001

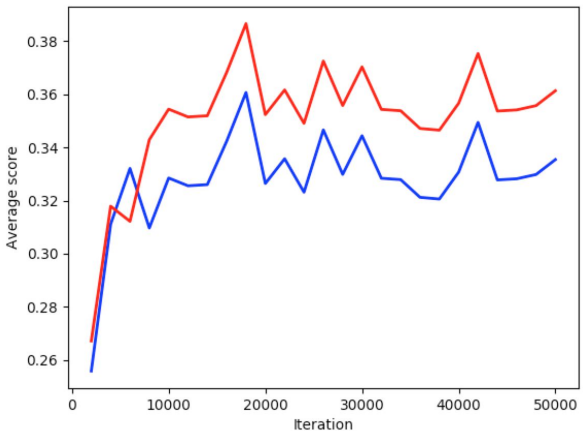




Train acc: Test acc

LR 0.005



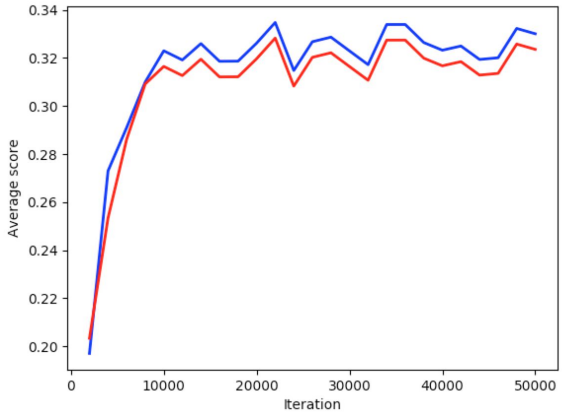


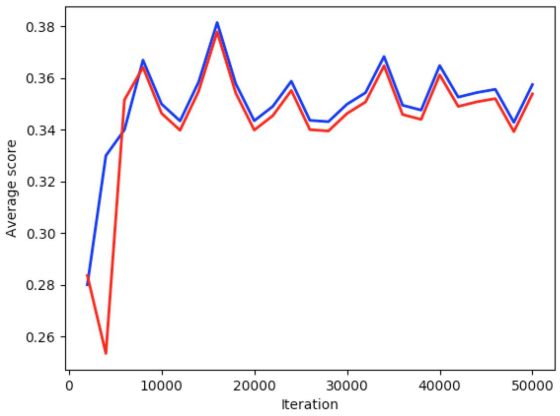
Arhitectura II

nn = FeedForward([LinearizeLayer(32, 32, 3), FullyConnected(32*32*3, 300, identity), Tanh(), FullyConnected(300, 200, identity), Tanh(), FullyConnected(200, 10, identity), SoftMax()])

Train vs Test: FC 300 - 200 - 10

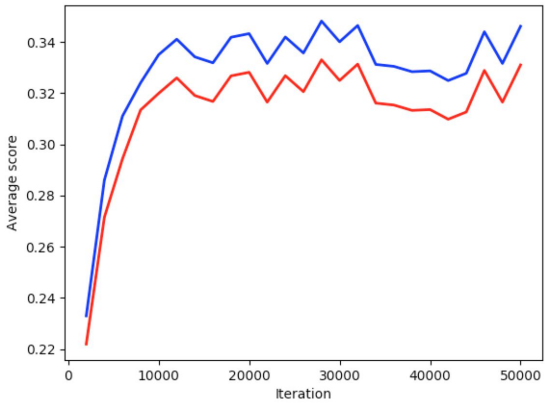
LR 0.001 LR 0.005

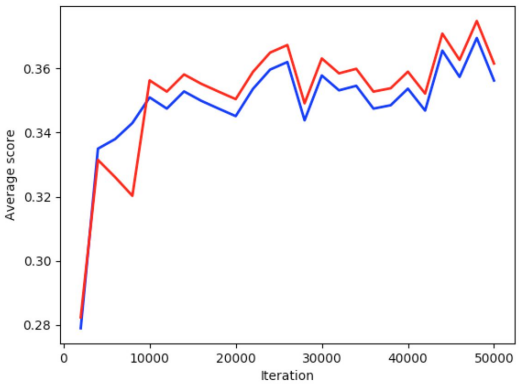




Train vs Test: FC 200 - 100 - 10

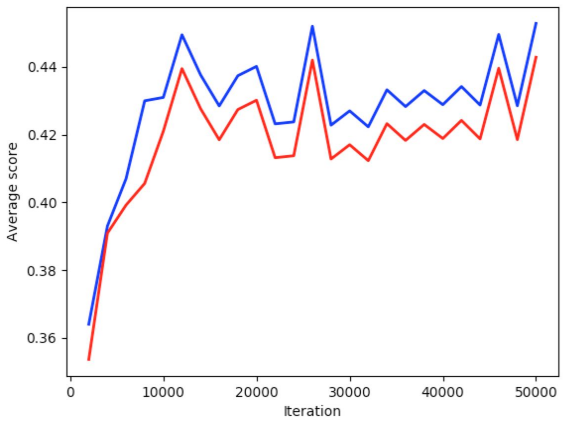
LR 0.001 **LR 0.005**

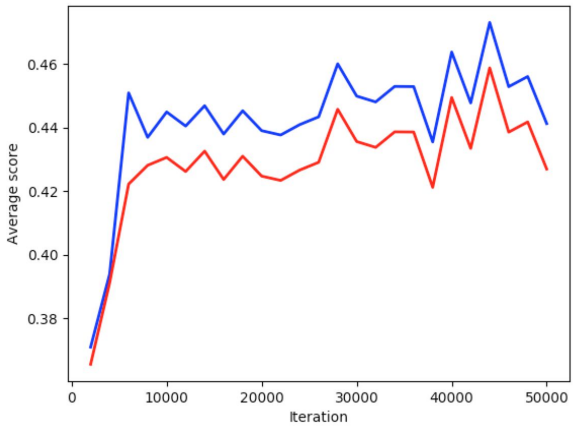




Best HighScores: FC 300 - 200 - 100

LR 0.001 **LR 0.005**



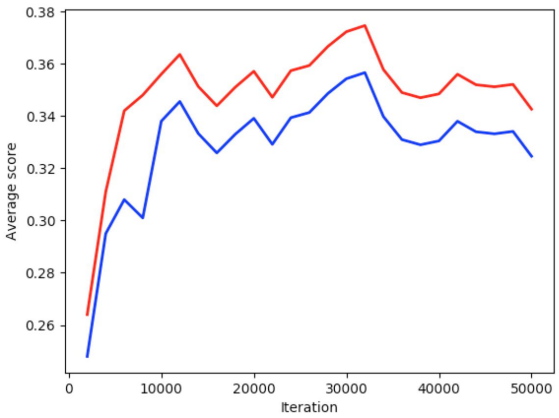


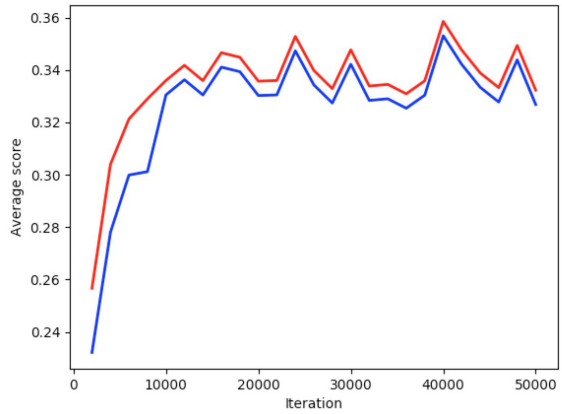
Concluzie:
FC 300 - 200 - 10 fata de 200 - 100 - 10 este putin mai bun

Influenta momentului

Train acc: Test acc

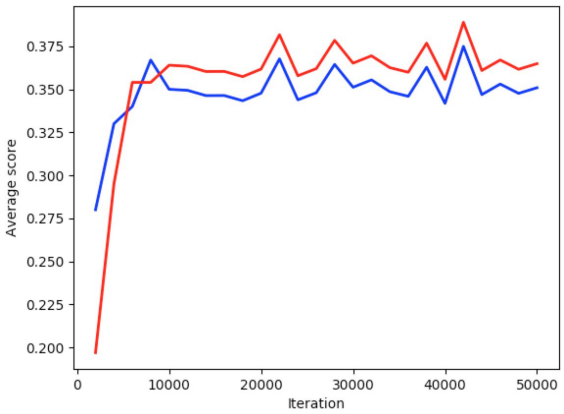
LR 0.001

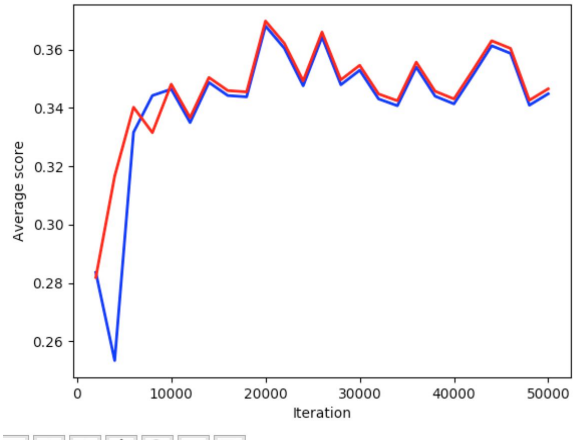




Train acc: Test acc

LR 0.005





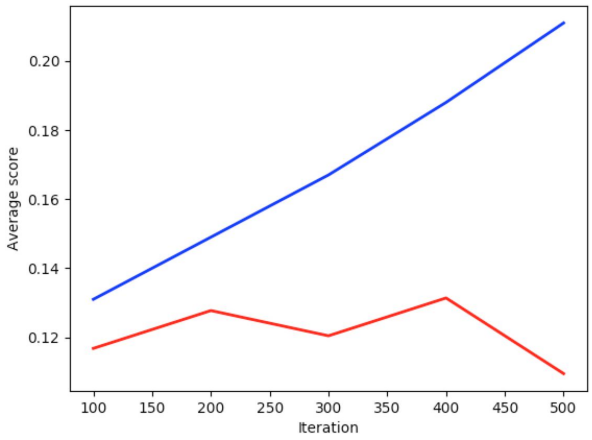
Dataset 2

Am folosit 70% poze pentru train, 30% poze pentru test. Totusi numarul de poze este destul de mic, 600 poze de train, 200 pentru test. Am folosit tot 10 clase.

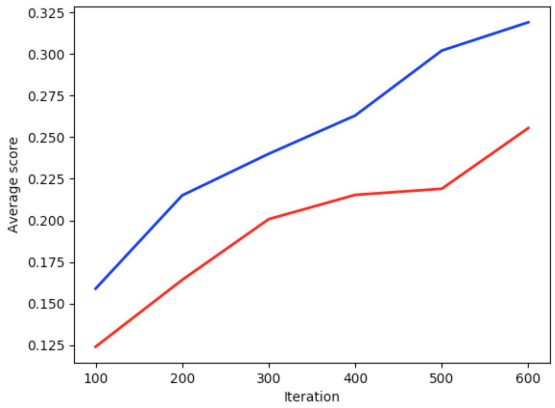
Rezultatele nu sunt prea relevante, in raport cu cifar.

Arhitectura I

LR 0.001

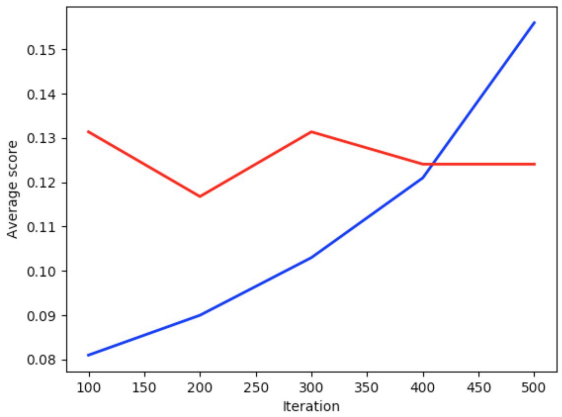


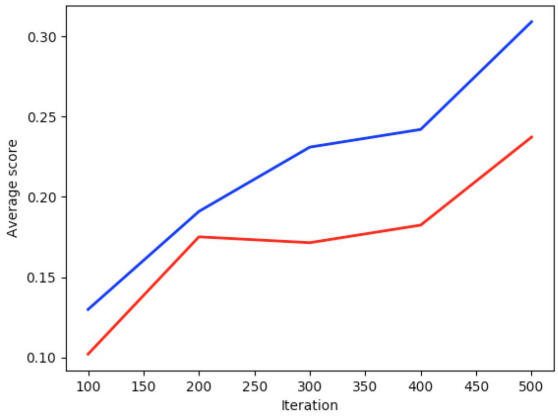
LR 0.005



Arhitectura II

LR 0.001 **LR 0.005**





Pentru LR mai mare se descurca mai bine fata de LR mic (0.001).

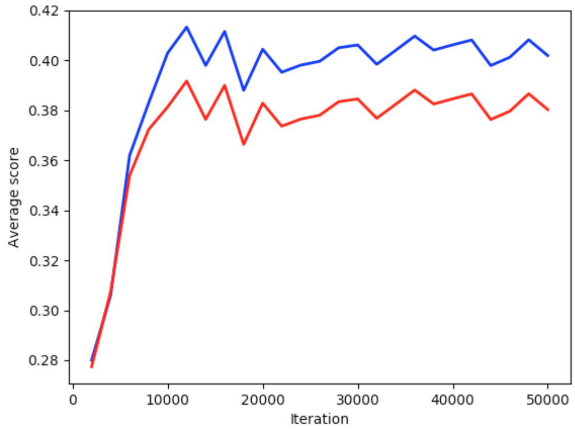
CONVOLUTIE

Arhitectura I:

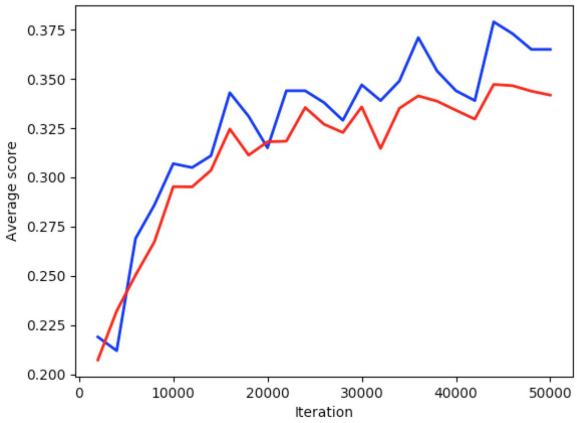
```
FeedForward([
        ConvolutionalLayer(3, 32, 32, 6, 5, 1),
        MaxPoolingLayer(2),
        ReluLayer(),
        ConvolutionalLayer(6, 14, 14, 16, 5, 1),
        MaxPoolingLayer(2),
        ReluLayer(),
        LinearizeLayer(16, 5, 5),
        FullyConnected(400, 300, relu),
        FullyConnected(300, 10, identity),
        SoftMax()])
```

Stride 1,1 k = 5, 5

LR = 0.001

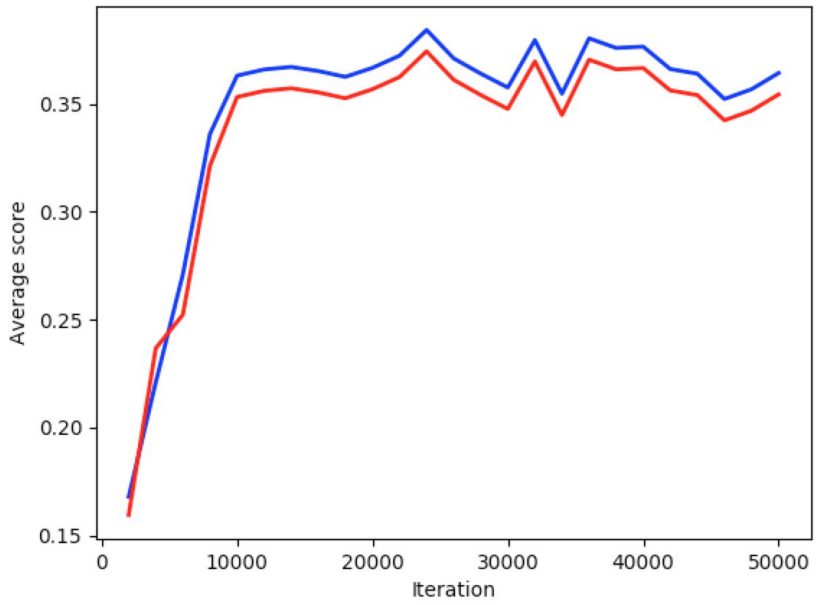


LR 0.002

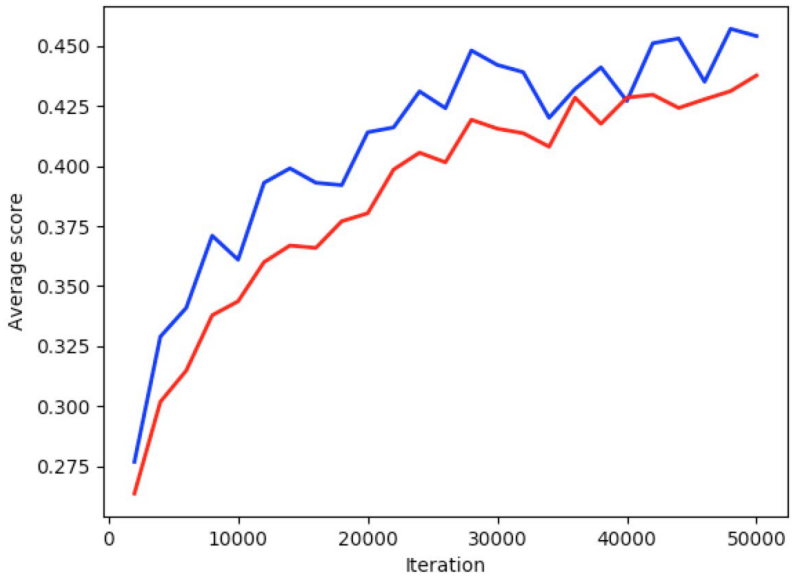


Stride 1,1 k = 9, 7

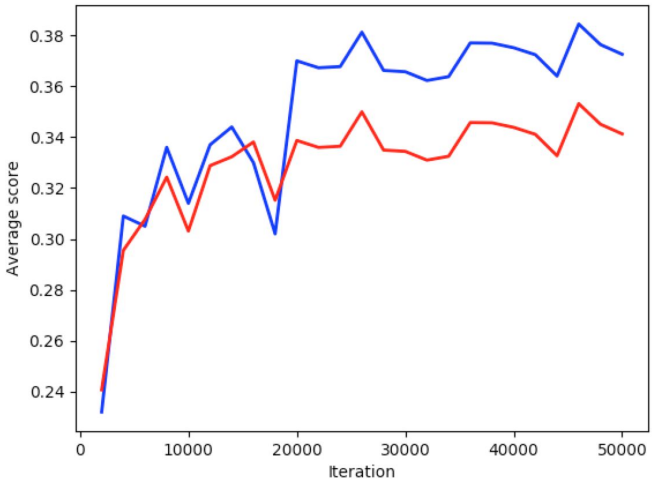
LR 0.002



Stride = 2, 2 k = 6, 4
LR = 0.001 (Highscore)



Filters number 8, 20
LR 0.001

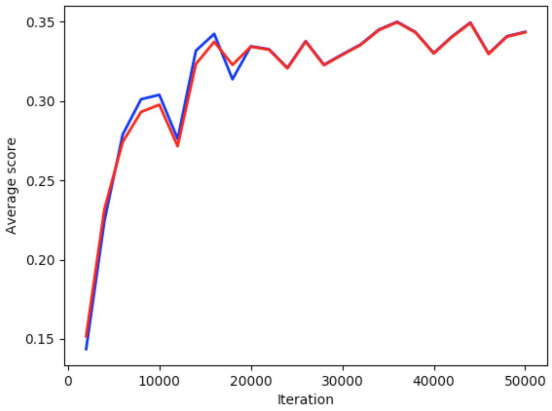


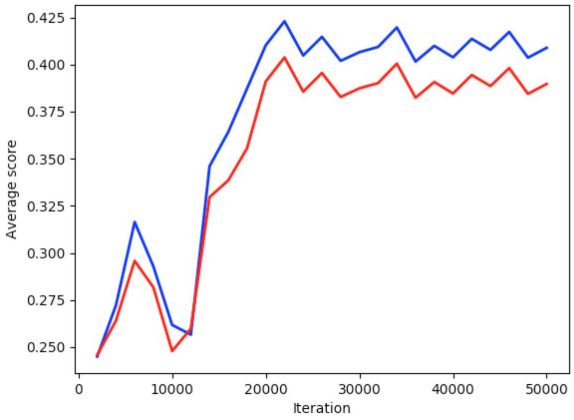
Arhitectura II:

FeedForward([

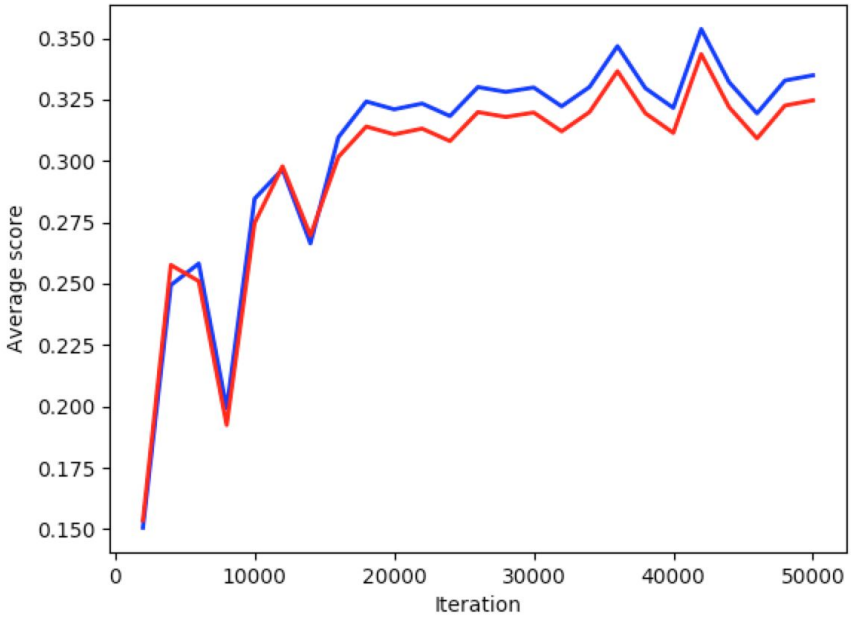
ConvolutionalLayer(3, 32, 32, 6, 5, 1),
ReluLayer(),
ConvolutionalLayer(6, 28, 28, 16, 5, 1),
ReluLayer(),
ConvolutionalLayer(16, 24, 24, 20, 5, 1),
MaxPoolingLayer(2),
ReluLayer(),
LinearizeLayer(20, 10, 10),
FullyConnected(2000, 100, logistic),
FullyConnected(100, 10, identity),
SoftMax()])

LR 0.001 **LR 0.002**

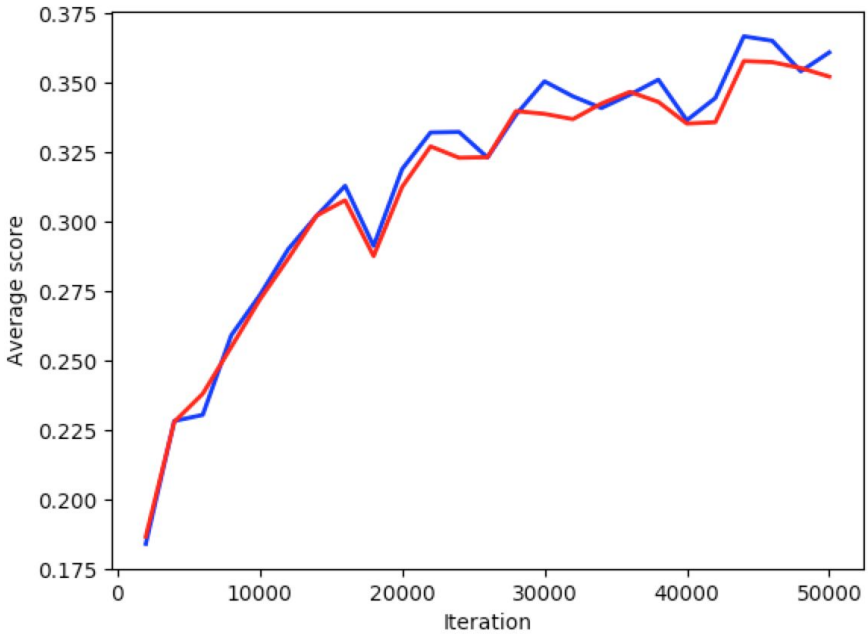




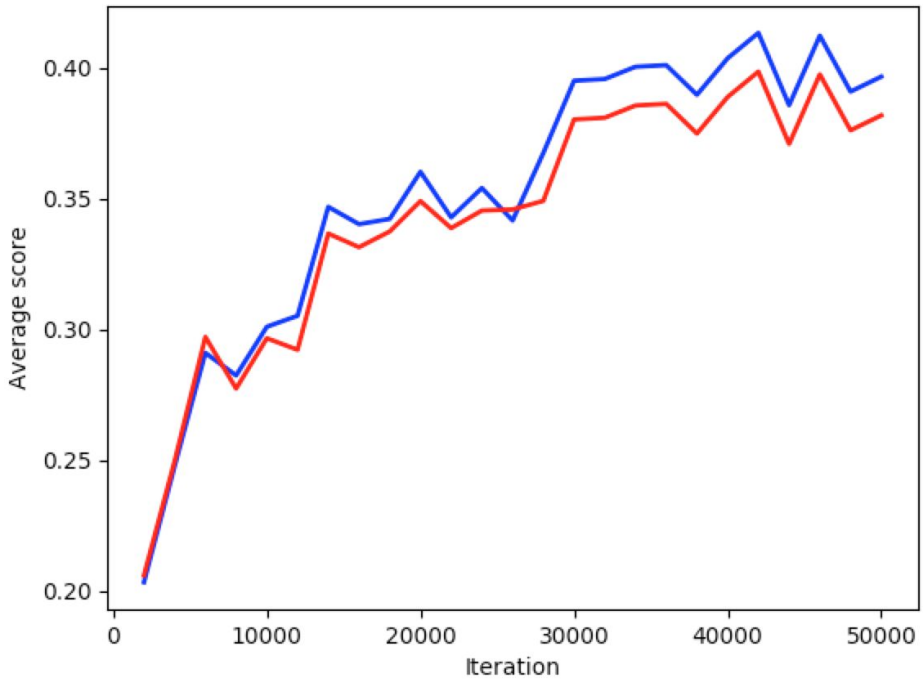
Stride = 1, 1, 1 **k = 7, 7, 7**
LR 0.001



stride = 2, 2, 1
LR 0.001



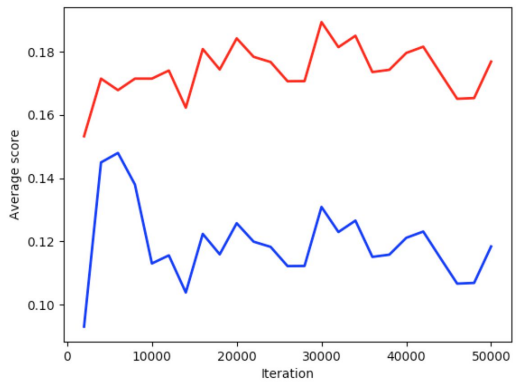
Stride 1, 1, 1 Filters_number 8, 20, 25
LR 0.001



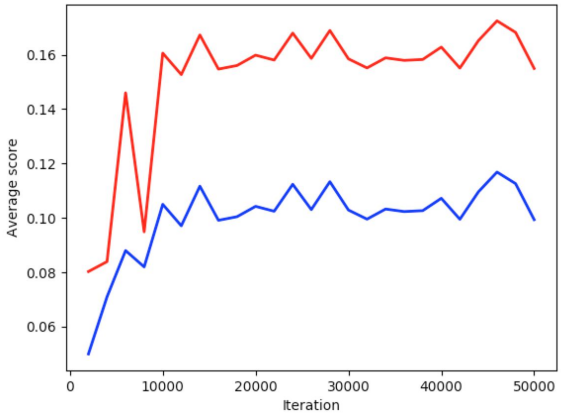
DATASET 2

Arhitectura I

LR 0.001

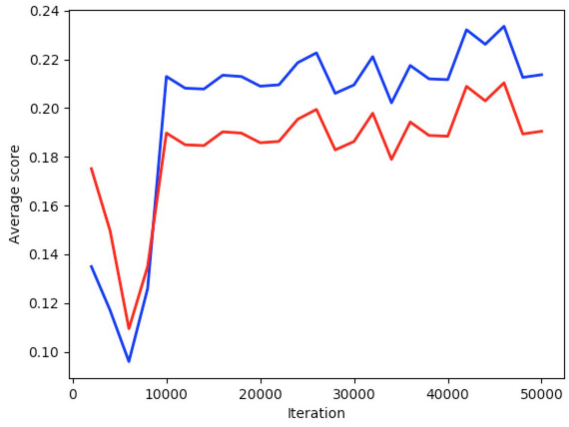


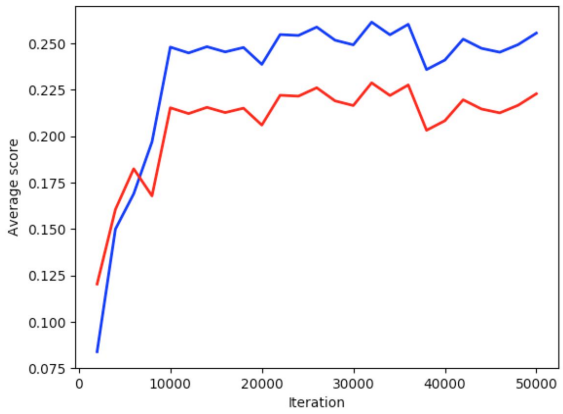
LR 0.005



Arhitectura II

LR 0.001 **LR 0.005**





Concluzie finala:

Convolutia a avut rezultate aseamanatoare cu reteaua liniarizata, dar o panta de invatare putin mai mare. Rezultatele cele mai bune au fost atinse de ambele retele.
In multe cazuri cu cat LR este mai mare cu atat reteaua invata mai repede, dar poate sa fie si instabila.
LR 0.001 si 0.002 au dat rezultate bune.
In combinatie cu momentum, daca prinde o directie de invatare gresita, cu LR mare se poate indrepta foarte mult in acea directie, lucru care nu ne favorizeaza deloc. Dar daca prinde o directie buna de invatare, procentul de puritate poate sa creasca si cu 2 - 3 %.

Pe un set de date mic LR 0.005 a dat rezultate mai bune, deoarece numarul de poze este mai mic, iar pozele sunt cu cifre, care sunt mai usor de distins.

Link-uri ajutatoare:
http://sebastianruder.com/optimizing-gradient-descent/index.html#momentum
http://wiseodd.github.io/techblog/2016/07/16/convnet-conv-layer/