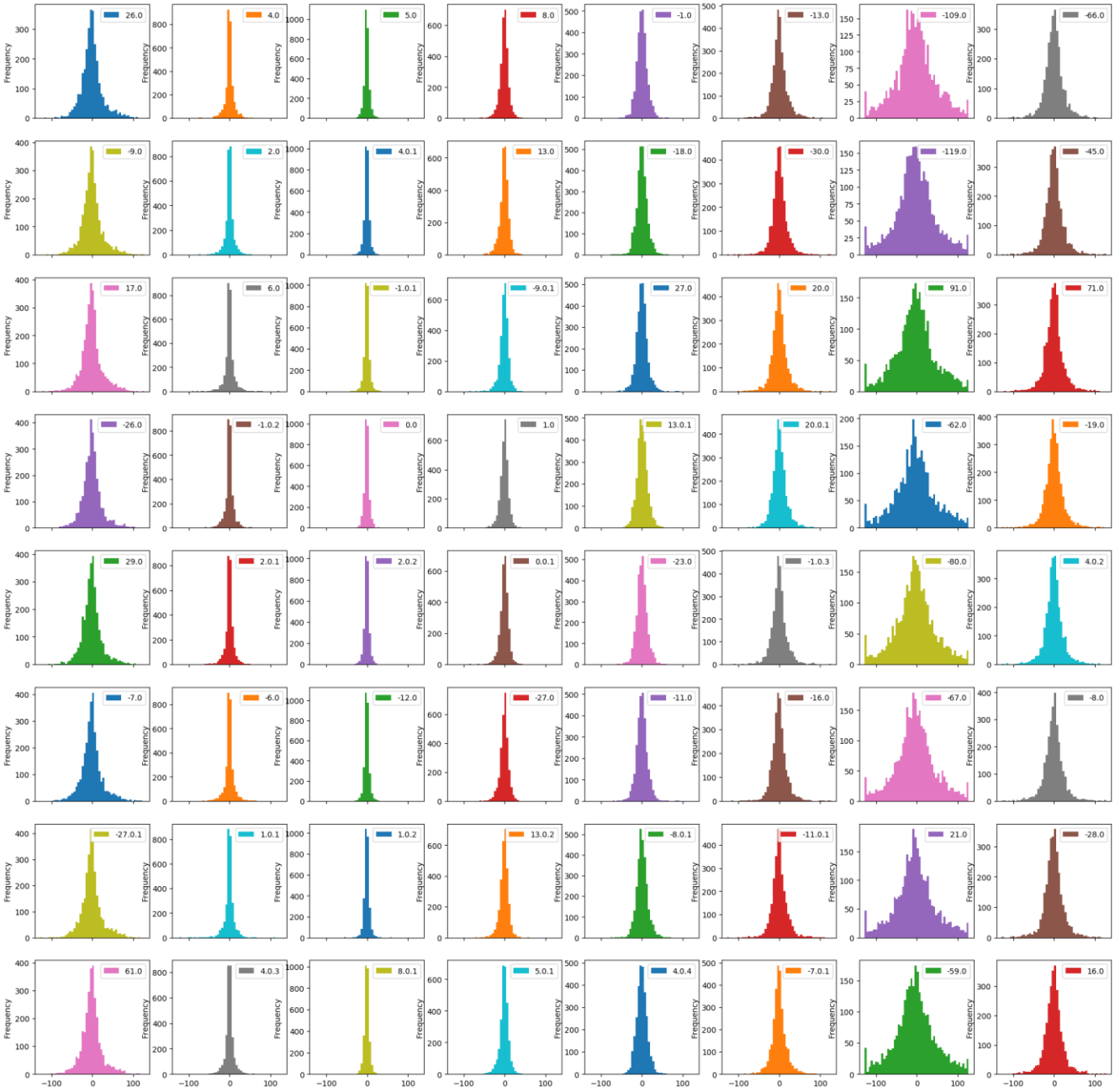
Classify gestures by reading muscle activity

A recording of human hand muscle activity producing four different hand gestures

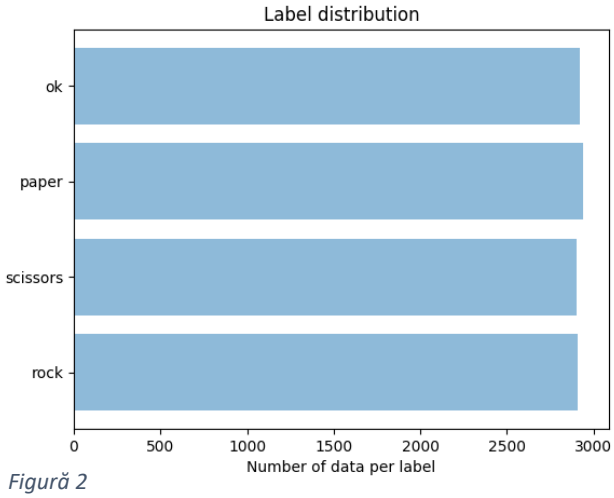
DESCRIEREA DATASETULUI

Datasetul este format din 11.674 date, fiecare cu cate 64 de feature-uri, distribuite in 4 clase (rock - 0, scissors - 1, paper - 2, ok - 3). O line din dataset reprezintă citirile repetitive a 8 mușchi a mâinii umane (citire1 mușchi 1, citire1 mușchi 2, ..., citire1 mușchi 8, citire2 mușchi 2, ... citire 8 mușchi 8) Datasetul a fost împărțit in 3 parți in train (80%), validare (10%) si test (10%). In *Figura 1* se pot observa distribuția datelor doar de la citirea mușchilor pentru gestul 0 (piatra). Mai exact pe direcție orizontala avem citirile pentru cei 8 mușchi distincți, iar pe direcție orizontala avem citirile repetitive pentru același mușchi.



Figură 1

Mai mult distribuția datelor pe clase este următoarea: 2909 (24.91%) date pentru clasa 0 (rock), 2902 (24.85%) date pentru clasa 1 (scissors), 2942 (25,2%) pentru clasa 2 (paper), 2921 (25,02%) pentru clasa 3 (ok). Acest lucru se poate observa si in *Figura 2*, si in plus putem spune ca datele sunt balansate.



Figură 2

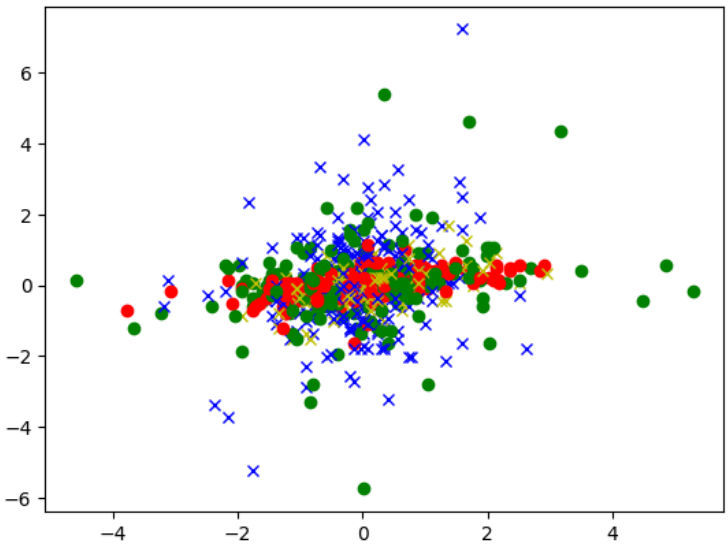
NORMALIZAREA DATELOR

Pentru datele de mai sunt se aplica o standardizare care scade media si împarte cu deviația standard. Acest lucru se realizează independent pe fiecare feature in parte însă independent de label, asumându-ne apriori ca datele vin din aceiași distribuție a claselor (ceea ce nu contrazici intuiția deoarece datele sunt balansate).

DESCRIEREA METODELOR FOLOSITE

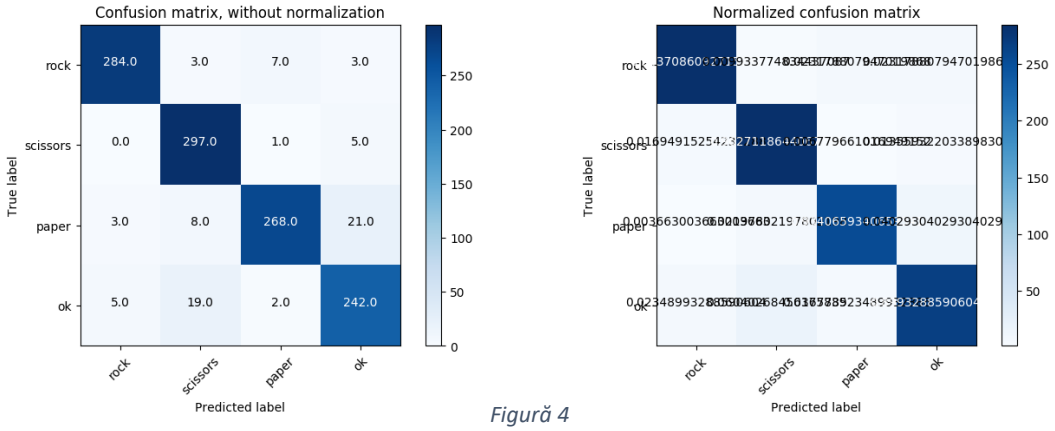
1. SVM

In *Figura 3* sunt plotate 1000 de datele random in 2-D anume, se i-a in considerare doar primele 2 feature-uri (mușchiul 1 si mușchiul 2 la prima citire). Se poate observa cu ușurința ca datele nu sunt separabile deloc in 2-D. O soluție ar fi maparea datelor intr-un spațiu de o dimensionalitate mai mare cu speranța unei separări in acel spațiu. Ei bine metoda SVM (Support Vector Machines) asta face, încearcă sa mapeze datele intr-un spațiu de o dimensionalitate mai mare apoi se încerca in noul spațiu o separare a datelor. In plus metoda SVM este una de clasificare binara, noi cum avem 4 clase avem nevoie de un truc pentru a folosi SVM. Acesta truc este antrenarea a 6 clasificatori diferiți (combinări de 2 luate cate clase avem, la noi 4, deci 6 clasificatori). Acesta este deja implementat si se numește “one-against-one”.



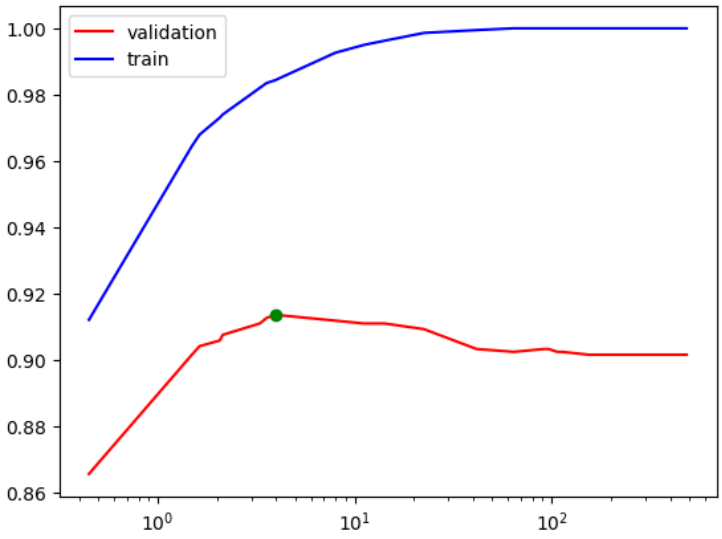
Figură 3

Apoi după un grid search după parametru C pe spațiul logaritmic [0.99, 999] din care se aleg 25 de puncte echidistante se obține o acuratețe maxima de 93.15% pentru C= 4.633413251903491.



Figură 4

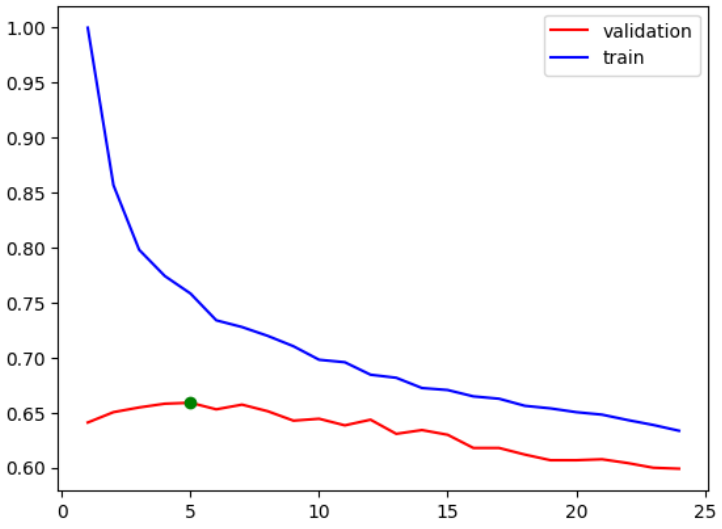
Apoi pentru un random search pe același spațiu logaritmic [0.99, 999], din care se aleg random 25 de puncte se obține o acuratețe maxima pe mulțimea de 93.40% pentru C = 3.9732891501042005. Acest lucru se poate observa si in *Figura 5*. In Figura 4 avem matricele de confuzie normala si normalizata pentru SVM-ul optim obținut după random search.



Figură 5

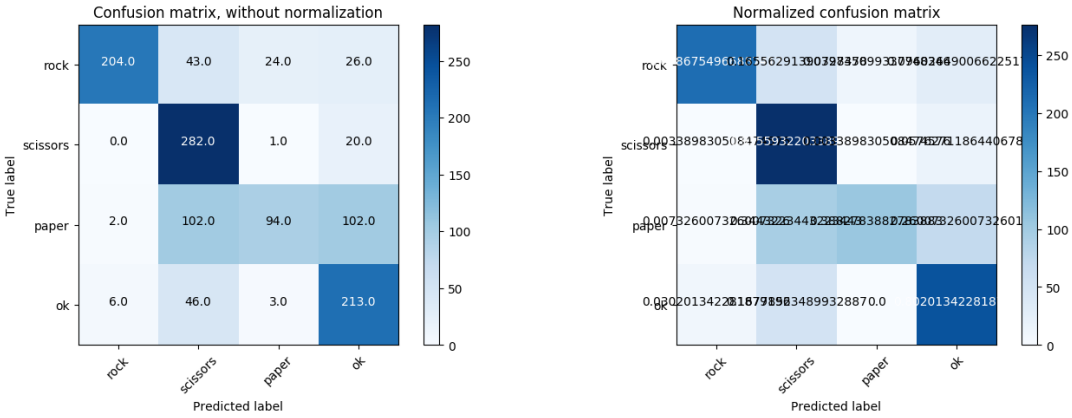
2. K-NN

Metoda k-NN consta in identificarea celor mai apropiați k vecini din mulțimea de train pentru un exemplu din test. La noi se calculează acești k vecini intr-un spațiu 64-dimensional mai exact $\mathbb{R}^{64}$. Metrica folosita este Minkowski. In continuare trebuie sa găsim k optim pentru care acuratețea pe mulțimea de validare este maxima.



Figură 6

După un grid search (random search nu are sens deoarece dorim numere naturale intre 1 si un maxim predefinit, am luat 25) se identifica k optim dintre toate posibile pana la un rang (la noi 25). Se identifica k optim ca fiind 5, cu o acuratețe pe mulțimea de test 67.83%.



Figură 7

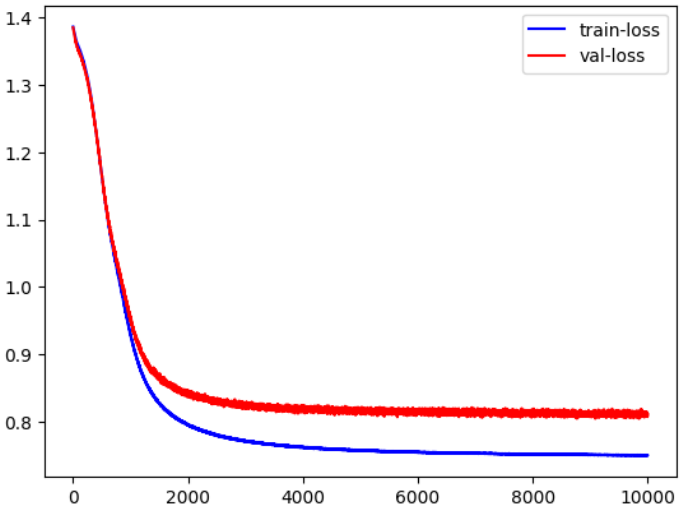
Se poate observa in *Figura 6* ca cu cat creștem numărul de vecini cu atât acuratețea scade pe validare dar si pe train de la 5 in colo. In *figura 7* avem matricele de confuzie pentru cel mai bun model k-NN adică 5-NN.

3. NN

Arhitectura primei rețele este formata din mai multe layere liniare: stratul de intrare de dimensiune 64 (numărul de feature-uri), doua hidden layere primul de dimensiune 512, al doilea de dimensiune 128 si stratul de ieșire de 4 perceptori. Fiecare strat ascuns are funcția de activare ReLU iar stratul de ieșire are activare softmax (întrucât ne dorim o delimitare clara a claselor la ieșirea din clasificator). In plus pentru a evita overfittingul după fiecare strat ascuns exista si procedeul de Dropout (cu probabilitatea custom p[1] = 0.05) care ignora datele cu probabilitatea setata. O descriere mai detaliata se afla mai jos in imagine:

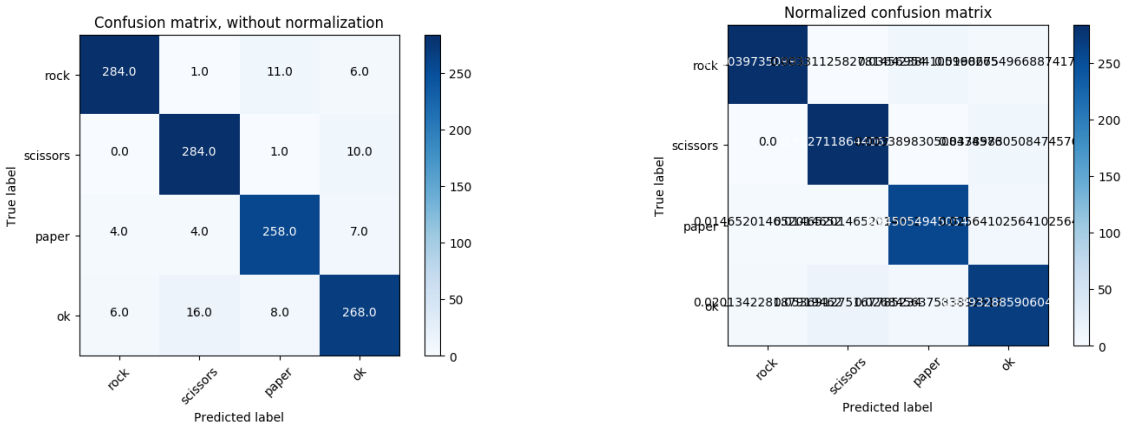
```
Net(
  (_layer1): Linear(in_features=64, out_features=512, bias=True)
  (_layer2): Linear(in_features=512, out_features=128, bias=True)
  (_layer3): Linear(in_features=128, out_features=4, bias=True)
  (_dropout1): Dropout(p=0.05, inplace=False)
  (_dropout2): Dropout(p=0.05, inplace=False)
)
```

Spre exemplu pentru o serie de hiperparametrii (learning rate = 0.01, si probabilitățile de dropout = 0.05) si optimizor SGD (coborârea pe gradient) se opțiune o performanta de 94.09% după 10000 de epoci. In *Figura 8* avem evoluția funcție de loss cu roșu pe validare si albastru pe mulțimea de train pe parcursul celor 10000 de epoci. Menționez ca pe post de funcție de loss s-a folosit crossentropia.



Figură 8

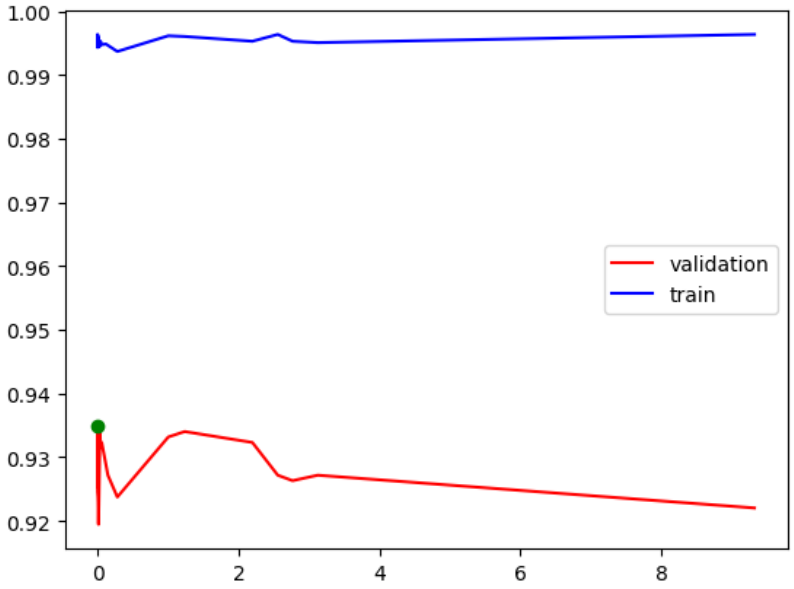
[1] Un p mare ingreuneaza invatarea iar un p care tinde la 0 Favorizeaza overfittingul.



Figură 9

In plus după un random search după parametrul learning rate pentru optimizorul SGD obținem ca cel mai bun learning rate este 4.434704499077834e-05 si o acuratețe de 95.29%. Menționez ca aceasta căutare se face pe un spațiu logaritmic discret de la 10^{-5} la 10 din care se aleg random 25 de potențiale learning rateuri.

Mai mult in *Figura 10* avem progresul acurateței pe mulțimea de validare in funcție de learning rate (punctul cu verde indica learning rate-ul optim ales) pentru care acuratețea pe mulțimea de validare este maxima. In *Figura 9* avem matricele de confuzie normala si normalizata a modelului optim (cel cu learning rate-ul ales).



Figură 10

COMENTAREA REZULTATELOR

După cum se poate observa anterior cea mai buna performanta este atinsa de rețeaua neuronalala cu arhitectura 64-512-128-4 si cu algoritmul de coborâre pe gradient lu learning_rate = 4.434704499077834e-05. Menționez ca rețeaua curenta a fost antrenata si cu optimizării Adagrad, RMSprop si Adam însă performantele sunt mai mici decât SGD. In plus este clar ca rețeaua are putere de generalizare fapt dovedit de performanta. Mai mult este capabila sa învețe fapt dovedit de acuratețea de aproape 100% pe train, deci are capacitate de învățare.

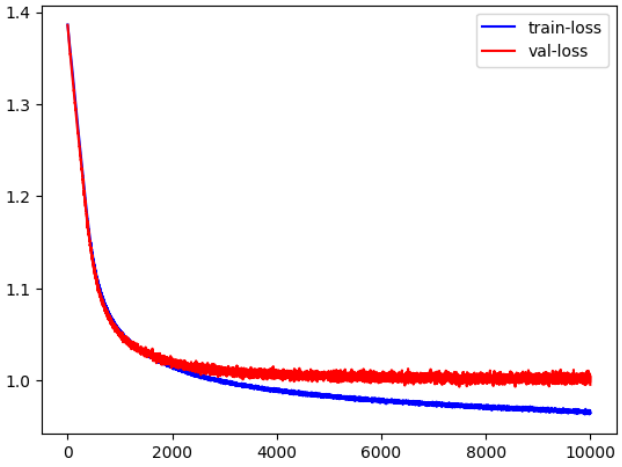
In plus aceasta performanta este comparabila cu cea obținuta de SVM. Menționez ca SVM-ul merge bine pe cazul de fata deoarece se știe ca aceasta metoda este una care nu face overfitting. Apoi la noi numărul de feature-uri este 64 mult mai mic decât numărul de exemple care este de 11674 si 64 este o dimensionalitate mare lucru la care metoda SVM este una buna. Un alt motiv pentru care SVM-ul funcționează bine la noi este dat de faptul ca am scalat datele (medie 0 si deviație 1). Mai mult, cum la noi un parametru optim pentru C este 3.97 > 1 rezulta, printre altele, ca datele noastre nu sunt zgomotoase (senzori care citesc mușchii sunt precișI).

Comparativ cu metoda k-NN, metodele anterioare sunt net superioare pe acest data-set deoarece datele nu sunt separabile in spațiul 64-dimensional (am văzut asta la secțiunea SVM ca nu erau separabile nici in 2-D pe primele doua feature-uri) pe când SVM si NN lucrează in dimensionalitate mai mari decât numărul feature-urilor. In plus se dovedește ca o vecinătate de 5 vecini este cea mai semnificativă, însă nu este relevanta pentru problema noastră.

SVM	k-NN	NN cu optimizator SGD si arhitectura (64,512,128,4)
C = 3.9732	K = 5	Lr = 4.434704499077834e-05
93.40%	67.83%	**95.29%**

Tabel 1

Mai mult decât atât, daca luam in considerare doar prima citire a celor 8 mușchi, si toate 8 obținem o acuratețe cu aceiași arhitectura a rețelei neuronale (8-512-128-4) de 73,11%. (*Figura 11*). Pentru un SVM (C=1) se obține acuratețea de 67.12%.



Figură 11

CONCLUZII

Prezentul proiect a reușit sa prezică cu o precizie de 95.29% semnele mâini umane (doar 4, piatra, foarfeca, hârtie si ok) dându-se 8 citiri a 8 mușchi ale mâinii. Se folosesc mai multe tehnici de ML cum ar fi SVM cu o acuratețe cu 2% mai mica decât cea mai buna din lucrare, k-NN cu o acuratețe de 67% si rețele neuronale care dau precizez cea mai mare dintre cele enumerate si anume 95.29%. In plus de demonstrează ca daca reducem numărul de feature-uri cu 87.5% (adică împărțim numărul de feature-uri la 8) obținem o performanta de cel puțin 73,11%.