3. naloga: Določanje elastičnih konstant v nematskem tekočem kristalu

Matic Debeljak

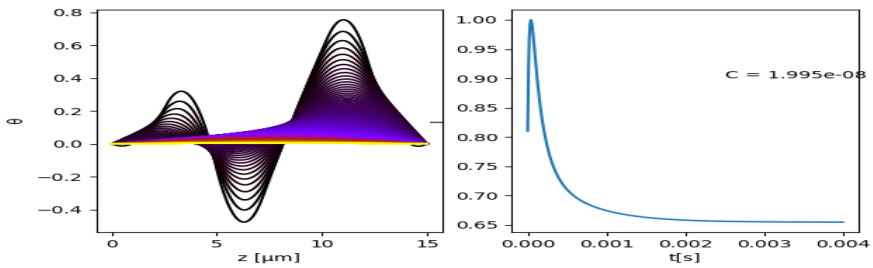
9. februar 2021

1 Uvod

Uporaba tekočih kristalov pogosto temelji na njihovih elastičnih lastnostih. V tej nalogi bomo spoznali potencialen nov način za določitev elastičnih konstant v nematskem tekočem kristalu s pomočjo nevronske mreže iz meritev intenzitete prepuščene svetlobe

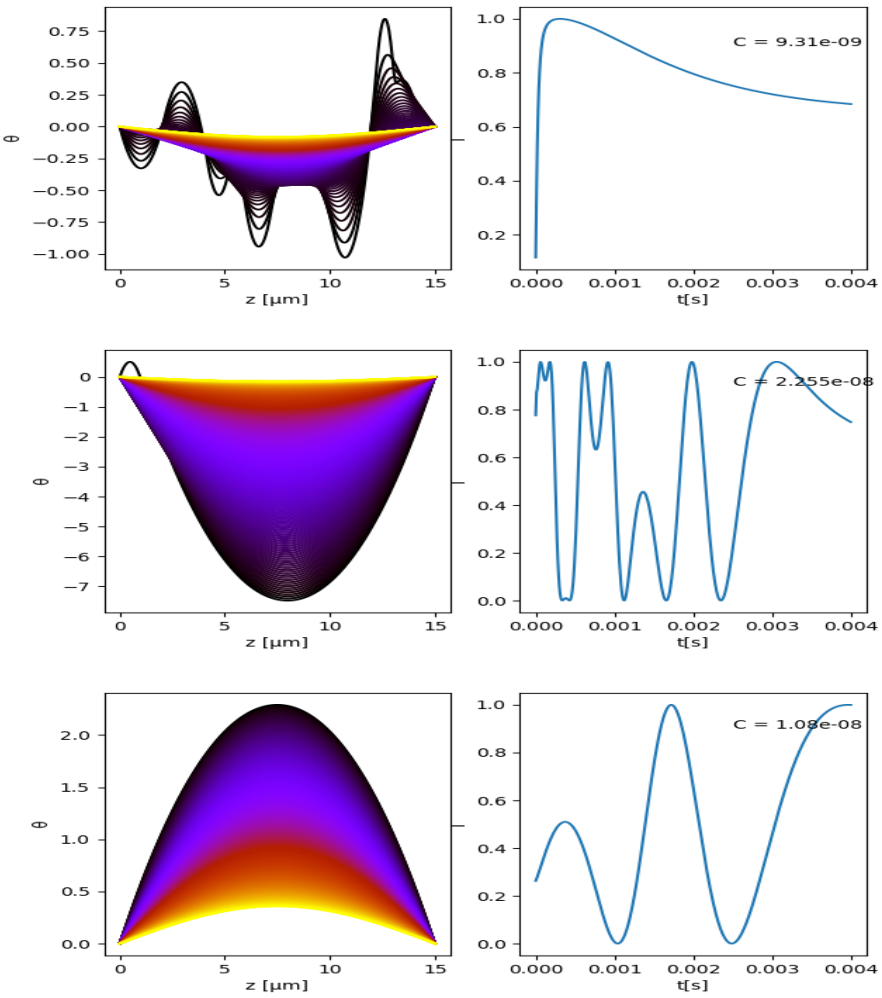
2 Podatkovni set

Podatke za uporabo pri tej nalogi sem generiral s pomočjo podanega programa, ki je določil začetni profil orientacije, izžrebal naključno konstano C, ki pa je določila kako se profil časovno razvije. Iz časovnega razvoja profila smo lahko nato določili itenziteto, ki pa je bila vhodni podatek za nevronsko mrežo. Za lažjo predstavo je na sliki 1 narisan en potek razvoja profila in krivulja itenzitete s pripadajočo konstanto.



Slika 1: Prikaza tipičnega časovnega razovja profila - začetek temnejše barve in konec svetlejše berve (levo) ter intenzitetne krivulje (levo).

Na sliki 2 pa bom prikazal nekaj primerov, ki odstopajo od najbolj tipičnih, saj lahko pri takih primerih pričakujemo več težav pri določanju konstante.

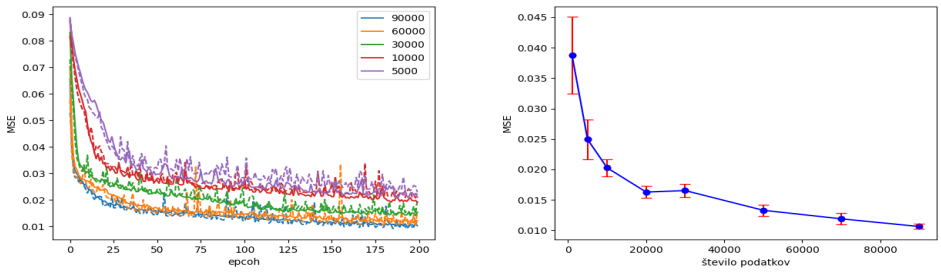


Slika 2: Nekaj primerov časovnega razovja profila - začetek temnejše barve in konec svetlejše berve (levo) ter intenzitetne krivulje (levo), ki odstopajo od najbolj tipičnih.

Na sliki 2 vidmo, da se nobena od izbranih intenzitetnih krivulj ni ustalila, kar verjetno pomeni, da smo naredili premalo korakov.

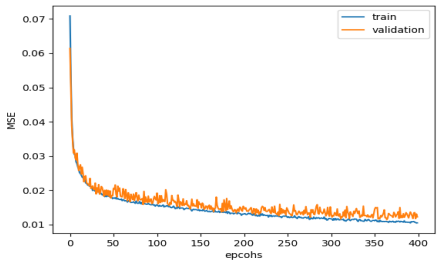
3 Nevronska mreža

Najprej sem sestavil gosto povezano nevronsko mrežo z enakmimi hiperparametri kot v naovdilih (2 skriti plasti, optimizer = adam in activation = sigmoid). Odločil sem se previriti kako na uspešnost (oziroma na napako MSE) vpliva število učnih podatkov, ki jih prejme nevronska mreža. Rezulatati so prikazani na sliki 3.



Slika 3: Prikaz napake MSE v odvinosti od števila epoch za različno število učnih podatkov (levo)- s polno črto je prikazana napaka na učnih podatkih, s črtkano pa napaka na tesnti množici. Na desni pa je prikazana nedoločenost napake v odvisnosti od števila podatkov pridobljena s k-fold validacijo.

Kot pričakovano lahko opazimo, da se mreža bolje uči na večjem številu podatkov, a večje število podatkov pomeni tudi večjo časovno zahtevnost. Odločil sem se, da bom časovno zahtevnejše postopke kot so grid search izvajal na 30000, saj je na takem številu podatkov nedoločenost napake še spremenljivo majhna. Na sliki 3 (levo) opazimo, da so rezultati precej "zašumljeni", tega se lahko znebimo s k-fold navzkrižno validacijo.



Slika 4: Prikaz napake MSE v odvinosti od števila epoch za 30000 učnih podatkov s k-fold navzkrižno validacijo ($k = 5$)

Na sliki 4 vidimo, da število epoch večje od 200 ni preveč smiselno, saj napaka pada samo na učnih podatkih, ne pa tudi na validacijskih, kar je cilj nevrnoske mreže.

4 Mrežno iskanje

Mrežno iskanje (grid search) nam omogoča iskanje optimalnih hiperperametrov modela. Do te točke v poročilu sem v modelu uporabljal hiperparametre, ki so bili podani v navodilih. Sedaj pa bom s pomočjo mrežnega iskanja poskusil določiti optimalne. Težava mrežnega iskanja je predvsem časovna zatevnost, saj moramo za vsako kombinacijo hiperparametrov izvesti k-fold navzkrižno validacijo (da dobimo natančnejši rezultat). Ob velikem številu hiperparametrov (število skritih plasti, število nevronov v vsaki skriti plasti, aktivacijske funkcije, optimizatorji itd.) ter različnih izbir za vsak hiperparameter, kmalu dobimo časovno zelo zahteven problem. Ena izmed rešitev je da hiperparametre razdelimo v manjše skupine, ki so med seboj neodvisne. Najprej sem testiral število epoch in batch size (tabela 1).

batch size	epochs	MSE
50	50	0.01789
50	100	0.01431
50	200	0.01571
100	50	0.01791
100	100	0.01688
100	**200**	**0.01384**
200	50	0.01877
200	100	0.01627
200	200	0.01537

Tabela 1: Najboljša kombinacija: batchsize = 100 in epochs = 200

Nato sem tesiral še število nevronov v prvi in drugi skriti plasti (tabela 2)

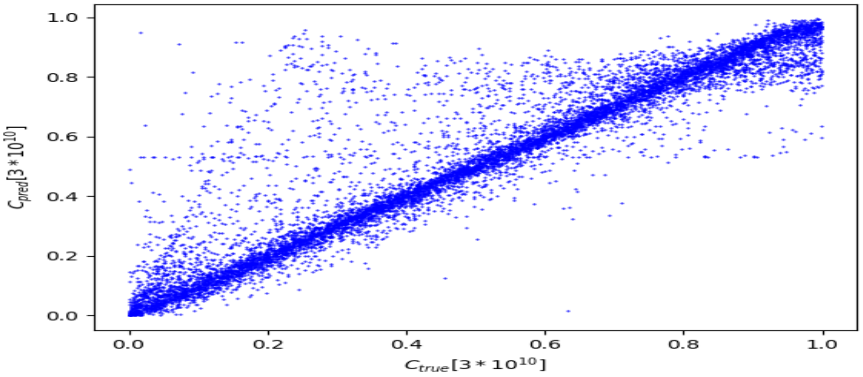
n1	n2	MSE
50	50	0.01533
50	100	0.01572
50	200	0.01571
50	300	0.01769
100	50	0.01791
100	100	0.01749
100	200	0.01505
100	300	0.1633
200	**50**	**0.01471**
200	100	0.01558
200	200	0.01576
200	200	0.01576
200	300	0.01051
300	50	0.01812
300	100	0.01786
300	200	0.01558
300	300	0.01480

Tabela 2: Najboljša kombinacija: n1 = 200 in n2 = 50

Nato sem tesiral še aktivacijske funkcije, optimizator in aktivacijsko funkcijo na izhodu mreže. Optimalna izbira za te je bila: aktivacijska funkcija relu, optimizator adam in izodna aktivacijska funkcija sigmoid.

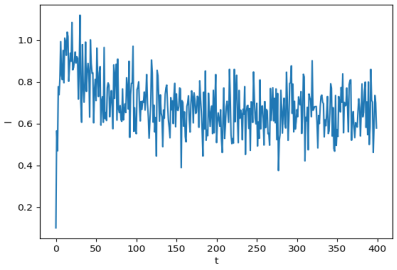
5 Napoved za neznane podatke

Cilj te naloge je bil da bi čimbolje določili neznano konstanto iz krivulje itenzitete. Poglejmo si kako uspešna je pri napovedi mreža, z hiperparmateri določenimi v prejšnem poglavju.

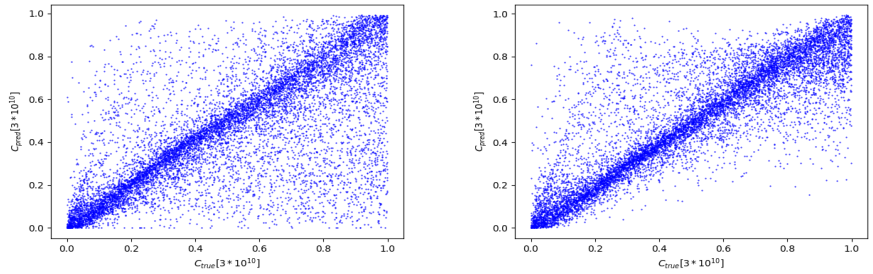


Slika 5: Prikaz napovedi za konstanto C in njeno dejansko vrednost na podatkih, ki jih mreža še ni videla

Na sliki 5 vidimo precej dobro ujemanje med napovedjo in dejansko vrednostjo konstante. Prihaja pa vseno do odstopanja v določenih krivuljah, med katerimi pa nisem opazil večje podobnosti. Lahko si pogledamo še kako na napoved vpliva prisotnost šuma v podatkih. Ločimo dva primera in sicer mrežo treniramo na nezašumljenih podatkih in jo ocenimo na zašumljenih podatkih, ter mrežo učimo in ocenjujemo na zašumljenih podatkih.



Slika 6: Prikaz zašumljene intenzitetne krivulje.

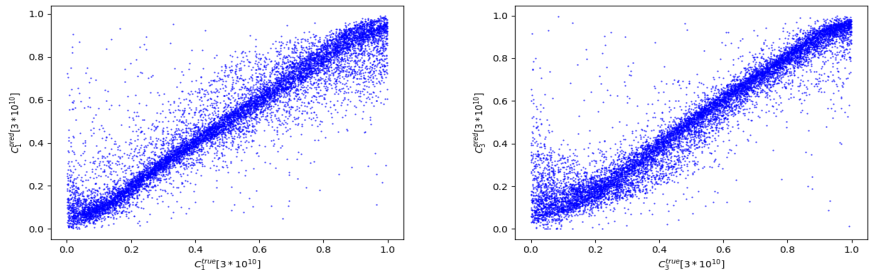


Slika 7: Prikaz napovedi za konstanto C in njeno dejansko vrednost na zasušumljenih podatkih, ki jih mreža še ni videla. Učenje na nezašumljenih podatkih (levo) in na zašumljenih podatkih (desno).

Kljub dokaj veliki zašumljenosti podatkov (slika 6) dobimo dokaj dobro napoved. Na sliki 7 opazimo, da če želimo napovedati konstanto za zašumljene podatke je mrežo bolje učiti na zašumljenih podatkih kar je tudi za pričakovati.

6 Določitev dveh konstant

Če želimo določiti dve konstanti moramo spremeniti sidranje, saj drugače nimamo dovolj podatkov. Postopek za izboljšave mreže pa je enak kot pri eni konstanit.

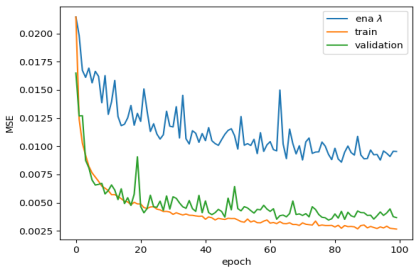


Slika 8: Prikaz napovedi za konstanti C_1 (levo) in C_3 (desno) ter njuni dejanski vrednosti na podatkih, ki jih mreža še ni videla.

Na sliki 8 vidmo, da mreža dokaj dobro napove obe konstanti, a vseno se napake zdijo večje kot pri samo eni konstanti, zato želimo napoved še izboljšat. To lahko storimo tako, da mrežo učimo z intenzitenimi krivuljami

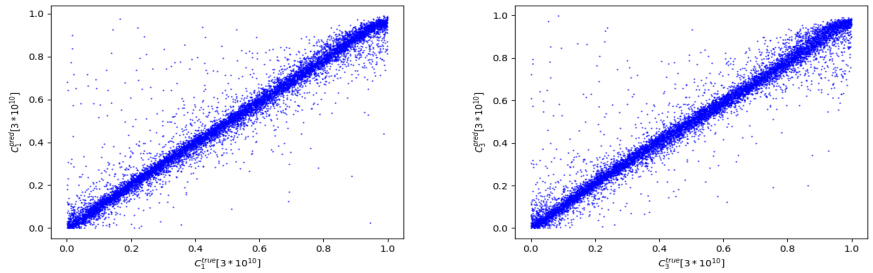
različnih valovnih dolžin (do sedaj sem uporabljal samo eno valovno doložino $\lambda = 1000$ nm).

Uporabil bom štiri različne valovne dolžine, to pomeni štirikrat več podatkov in s tem tudi večjo časovno zahtevnost. Zato za več različnih valovnih dolžin nebom naredil k-fold navzkrižne validacije in mrežnega iskanja optimalnih hiperparametrov, privzel bom, da so te kar enaki kot v primeru z eno valovno dolžino. Kljub neoptimalnemu postopku dobimo boljše rezultate (slika 9).



Slika 9: Primerjava MSE za eno valovno dolžino in več valovnih dolžin.

Na sliki 9 vidimo, da se v primeru uporabe štirih valovnih dolžin napaka zmanjša približno za faktor 2, kar je velik napredek. Še vedno pa je najpomebnejša napoved na še neveidenih podatkih prikazana na sliki 10.



Slika 10: Prikaz napovedi mreže učene na štirih valovnih dolžinah za konstanti C_1 (levo) in C_3 (desno) ter njuni dejanski vrednosti na podatkih, ki jih mreža še ni videla. M

Napoved na sliki 10 je bistveno boljša od napovedi na sliki 8. Podobno bi lahko izboljšal tudi napoved za eno konstanto.

7 Zaključek

Pri tej nalogi smo se spoznali z nevronskimi mrežami, ter nekaj orodij, ki se pogosto uporabljajo za izboljšavo napovedi nevronskih mrež.