Primjene metoda dubokog učenja u kompleksnim sustavima

Mirković, Filip

SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

Filip Mirković

Primjene metoda dubokog učenja u kompleksnim sustavima

Diplomski rad

Zagreb, 2024.

SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

INTEGRIRANI PREDDIPLOMSKI I DIPLOMSKI SVEUČILIŠNI STUDIJ
FIZIKA; SMJER ISTRAŽIVAČKI

Filip Mirković

Diplomski rad

Primjene metoda dubokog učenja u kompleksnim sustavima

Voditelj diplomskog rada: prof. dr. sc. Davor Horatić

Ocjena diplomskog rada: _____

Povjerenstvo: 1. _____

2. _____

3. _____

Datum polaganja: _____
Zagreb, 2024.

Cijeli život sam imao privilegiju biti okružen prekrasnom obitelji i prekrasnim prijateljima te mislim da ne postoji niti jedno moje postignuće koje se dogodilo bez njihove podrške. Najveće zasluge sigurno imaju mama Silvana i tata Igor, koji su me strpljivo podržavali kroz cijeli studij, makar im velika većina toga čime sam se bavio nije bila jasna. Iskreno vjerujem kako mi je učenje fizike obogatilo život na razne načine, za to bi htio zahvaliti svim profesoricama i profesorima koji su mi prenijeli to znanje, davali mi razne savjete i pisali mi pisma preporuke. Za izradu ovog rada posebno sam zahvalan mentoru prof. dr. sc. Davoru Horvatiću, čiji su savjeti učinili ovaj rad puno kvalitetnijim. Njegova je pomoć bila od presudne važnosti u zadnjim trenutcima pisanja rada. Bojim se da bi popis ljudi za koje vjerujem da su mi direktno ili indirektno pomogli da prođem studij bio duži od samog rada tako da ih nažalost neću ovdje sve navoditi. Siguran sam, ako budu ovo čitali, da će se sami prepoznati. Hvala Vam.

Sažetak

U ovom radu bavimo se modeliranjem kompleksnih sustava korištenjem neuronskih mreža nad grafovima. Posebnu pažnju posvećujemo ugradnji simetrija fizikalnog sustava u model, koje su presudne za kvalitetno funkcioniranje modela. Koristimo O(3)-ekvivarijatnu inačicu transformera nad grafovima za predviđanje međuatomskih sila unutar malih organskih molekula: benzena, uracila, etanola i aspirina. Provjeravamo generalizacijska svojstva modela pri predviđanju sila na atome unutar molekule paracetamola. Na kraju rada, istražujemo robusnost modela u režimima malog broja podataka. Taj režim je ostvaren treniranjem modela na deset puta manjem skupu podataka od inicijalnog skupa podataka za trening.
Ključne riječi: Kompleksni sustavi, Molekularna dinamika, Neuronske mreže nad grafovima, Ekvivarijantnost

Deep learning methods in modeling coplex systems

Abstract

In this work, we examine the use of graph neural networks in modelling complex systems. We also emphasise the use of built-in symmetries in our model. We demonstrate these ideas on the problem of molecular dynamics. A version of an O(3)-equivariant graph attention network is used to predict interatomic forces in small organic molecules: benzene, ethanol, uracil, and aspirin. Furthermore, we test the model's generalization capabilities on various configurations of paracetamol. Lastly, we demonstrate the power of equivariance in low-data regimes. This regime is achieved by training the model with a ten times smaller dataset than the initial training set.

Keywords: Complex systems, Molecular dynamics, Graph neural networks, Equivariance

Sadržaj

1 Uvod

Područje istraživanja kompleksnih sustava nastalo je iz različitih pravaca istraživanja poput fizike, biologije i društvenih znanosti. Unatoč sveprisutnosti kompleksnih sustava, područje njihovog istraživanja je poprilično mlado do te mjere da ne postoji sveobuhvatna i rigorozna definicija kompleksnosti, pa tako ni kompleksnog sustava. S druge strane postoji niz karakteristika koje sustav mora imati da bismo ga smatrali kompleksnim, osnovne karakteristike su da sustav mora biti sastavljen od više dijelova te da ti dijelovi moraju moći nekako interagirati. Iz navedenih svojstava čini se da je svaki netrivijalan fizikalni sustav ujedno i kompleksan, stoga se osvrćemo na sustave koji pokazuju tzv. izranjajuća ponašanja (*eng.*emergent behaviour), odnosno ponašanja koja su rezultat interakcija sustava. Dakle možda najbitnija karakteristika jest da je ponašanje sustava kompliciranije od sume ponašanja njegovih dijelova, malo formalnije ovo znači da često pokazuju nelinearnu dinamiku. Zbog velikog naglaska na interakcije dijelova sustava, prirodno se promatraju putem kombinatornih grafova.

U ovom radu nastojimo modelirati svojstva kompleksnih sustava, korištenjem dubokih neuronskih mreža (DNN), odnosno neuronskih mreža nad grafovima (GNN). Bitna tema ovog rada bit će kako ugraditi spoznaje o simetrijama sustava direktno u model. Sustav na kojeg se fokusiramo su molekule, konkretno modelirat ćemo sile između atoma molekula. Primijenit ćemo model na molekule benzena, etanola, uracila, aspirina i paracetamola.

U poglavlju 2. detaljnije ćemo opisati razne kompleksne sustave u fizici, njihova svojstva, kako ih povezuje reprezentacija grafovima te izložiti ideje radova sličnih ovomu.

U poglavlju 3. opisat ćemo kako se molekule uklapaju u sliku kompleksnih sustava te standardne metode rješavanja problema molekularne dinamike. Ondje također motiviramo pristup modeliranja molekula putem DNN-ova te opisujemo podatke nad kojim vršimo analizu.

Poglavlje 4. posvećeno je opisu metoda strojnog i dubokog učenja. Konkretno bavit ćemo se tradicionalnijim arhitekturama poput DNN, objasnit ćemo postupak optimizacije DNN-a, koji je isti i za kompliciranije arhitekture te završavamo s opisom DNN-ova kao univerzalnih aproksimatora. U iduća dva poglavlja postupno,

objašnjavamo sve složenije arhitekture kojim modeliramo kompleksne sustave.

Nakon pregleda tradicionalnijih metoda, u poglavlju 5. detaljnije se bavimo neuronskim mrežama nad grafovima (GNN). Prvo iznosimo općenita svojstva grafova te objašnjavamo kako opisati kompleksne fizikalne sustave ili kompliciranе geometrijske domene grafom. Zatim objašnjavamo glavnu operaciju GNN-a: mehanizam slanja poruka. Na kraju opisujemo posebnu vrstu slanja poruka mehanizam pozornosti koji je temelj transformerskih modela.

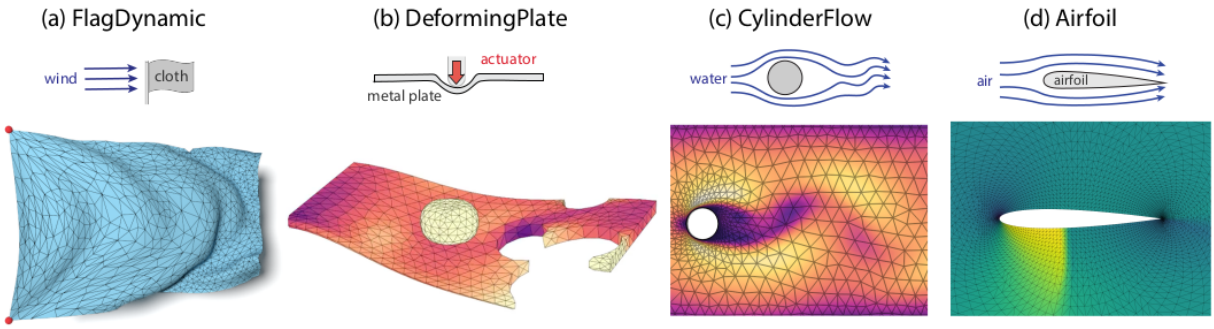
U poglavlju 6. detaljno opisujemo ugradnju simetrija u GNN-ove. Objašnjavamo kako zahtjev za simetrijama ograničava skup operacija koje su dozvoljene za izgradnju modela te opisujemo slanje poruka putem tenzorskog produkta i ulogu skalara u ugradnji nelinearnosti u model. Konkretne primjere dajemo za grupu O(3) i grupu translacija, koje povezujemo u euklidsku grupu simetrija E(3).

Zadnje poglavlje 7. povezuje sve od navedenog u O(3)-transformer, kojeg treniramo za zadatak predviđanja sila na atome unutar molekula benzena, etanola, aspirina te uracila. Pokazujemo kako modeli koji poštuju simetrije sustava dobro generaliziraju na podatcima koji su različiti od onih u skupu podataka za trening. Za provjeru generalizacije koristimo molekulu paracetamola. Osim generalizacijskih svojstava pokazujemo robusnost ovakvih modela u režimima malog broja podataka.

2 Kompleksni sustavi

U ovom poglavlju ćemo opisati neke popularne primjere kompleksnih sustava u fizici, neka njihova zanimljiva izranjajuća ponašanja te kako se modeliraju. Potom ćemo objasniti kako se svi oni reprezentiraju grafovima. Na kraju ćemo spomenuti niz metoda modeliranja kompleksnih sustava putem GNN-ova te ključnu ulogu simetrije za modeliranje.

Školski primjer kompleksnog sustava jest sustav spinske rešetke. Iako je ponašanje svakog pojedinog spina jednostavno, sustav u cjelini stvara neke zanimljive pojave poput vrlo dugo dalekosežne interakcije spinova te pojava poput pamćenja prošlosti sustava koje se manifestira u obliku histereza u polarizaciji sustava [1]. Postoje i jednostavniji primjeri u kojima kompleksni sustav pokazuje pamćenje prošlosti. U slučaju dinamike deformabilnih tijela kada naprezanje sustava postaje dovoljno veliko deformacije postaju trajne, u oba se slučaja radi nelinearnom odzivu sustava na vanjsku pobudu. Dinamika deformabilnih tijela obuhvaća i dinamiku platna poput onog na slici 2.1. Osim što je dinamika platna kaotična, zbog elastične interakcije točaka na platnu javlja se kolektivno ponašanje u obliku valova.



Slika 2.1: Primjeri kompleksnih sustava koji se simuliraju na mreži: dinamika platna, kvazistacionarno deformiranje materijala, strujanje fluida oko prepreke. U situacijama gdje je mreža stacionarna svi su grafovi naslijeđeni od mreže, s druge strane kada se mreža deformira bitno je graditi grafove koristeći i udaljenost u prostoru u kojeg je mreža uronjena. Preuzeto iz [2].

Fluidi su kompleksan sustav u kojem interakcije ne moraju biti samo dalekog prostornog dosega već se kolektivno ponašanje može širiti kroz razne prostorne skale. Pojava koja ovo najbolje utjelovljuje je turbulencija, ona se često opisuje kao gubitak energije ili angularnog momenta s makroskopskih na mikroskopske skale. Ovisno o

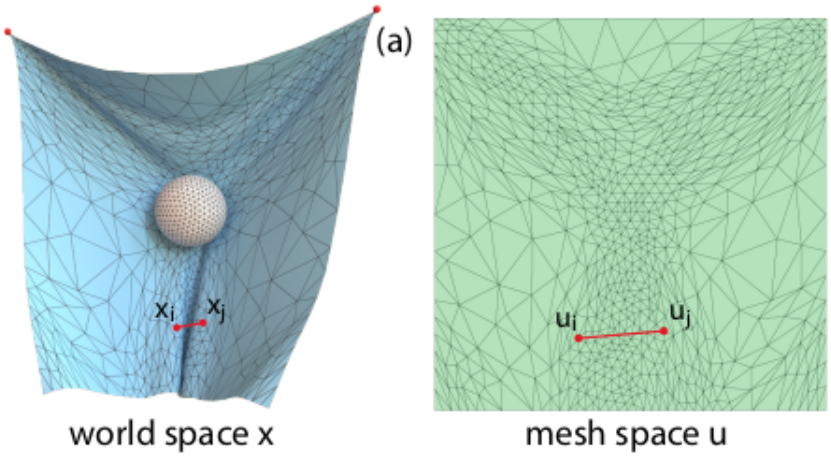
jakosti interakcija između točaka fluida te svojstvima tvari koja ga sačinjava, fluidi mogu pokazivati vrlo različito ponašanje od kompresibilnih i nekompresibilnih fluida do ne newtonskih fluida.

Osim klasičnih sustava, sličnim se pristupom mogu modelirati i kvantno mehanički sustavi poput molekula ili sustava elementarnih čestica. Molekule su primjer kompleksnog sustava koji jasno prokazuje koliko su interakcije dijelova sustava presudne u određivanju kolektivnog ponašanja sustava. Dvije molekule identičnog sastava, ali različitih struktura (izomeri) mogu imati vrlo različita svojstva primjerice gustoću, stabilnost, temperaturu vrelišta itd. Analogno vrijedi i za kristale, najznačajniji je primjer svih kristalnih rešetki ugljika.

Formalno graf je uređeni par skupa vrhova i skupa bridova koji označavaju interakciju između vrhova. Formalniji opis grafova dajemo u potpoglavlju 5.1, dok ovdje ukratko objašnjavamo zašto su toliko sveprisutni u modeliranju kompleksnih sustava. Iako su kvantni sustavi opisani valnim jednadžbama na nekom kontinuumu često posjeduju dijelove koji su dovoljno lokalizirani da se opisuju klasično ili semi-klasično, recimo atomske jezgre u molekulama i ionski ostatci u kristalima. Promatramo li samo te lokalizirane dijelove lako se vidi struktura grafa, dapače strukturne formule molekula upravo jesu vrsta grafa. U tom slučaju pitanje je kako reprezentirati sustave poput fluida ili deformabilnog dijela grafom? Jako često simulacije tih sustava odvijaju se na diskretnoj domeni, tj. mreži. Kada je mreža napravljena kvalitetno, odnosno dobro opisuje kontinuum tada možemo samu mrežu uzeti kao reprezentaciju grafa. Zaključak je da se u svim praktičnim primjenama dinamika kompleksnog sustava odvija na grafovima. Sama mreža ponekad nije dovoljna tako da imamo primjere u kojima se osim susjeda na mreži dodaju susjedi koji su bliski u prostoru [2], zoran primjer je dinamika platna u kojem se dvije točke koje su daleko u prostoru mreže mogu pronaći vrlo blizu jedna drugoj u euklidskom prostoru (vidi sliku 2.2).

Budući da su grafovi osnovan oblik reprezentacije kompleksnog sustava, modeliranje svojstava ili dinamike kompleksnih sustava strojnim učenjem najčešće se obavlja korištenjem GNN-ova [2, 3]. Međutim, pokazuje se da to često nije dovoljno tako da najrobusniji i najprecizniji modeli u sebi imaju ugrađenu simetriju sustava [4–7].

Pri modeliranju svojstava molekula može se direktno računati elektronska gustoća pa iz nje putem Hellman-Feynman teorema odrediti druga svojstva koja nas interesiraju. U radu [8] autori pokušavaju izgraditi GNN s ugrađenom euklidskom grupom

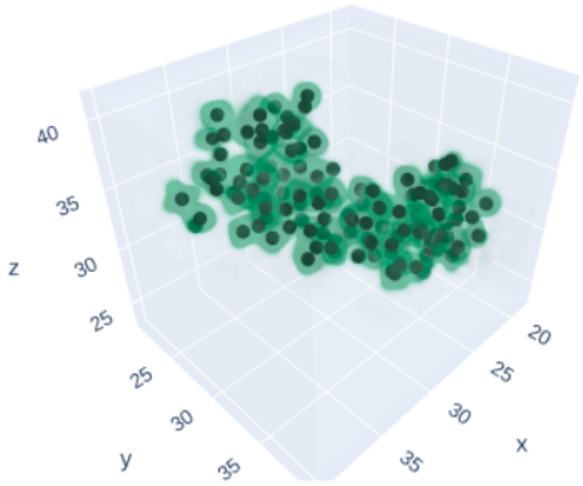


Slika 2.2: Primjer kompleksnog sustava opisanog mrežom. Graf se u ovom slučaj mora graditi s obzirom na udaljenost u prostoru mreže (desno) i euklidskom prostoru u koji je mreža uronjena (lijevo). Preuzeto iz [2].

simetrija kako bi odredili elektronsku gustoću manjih sustava molekula, a zatim to primjenjuju na sustave veće za nekoliko redova veličina. Ideja je slična kao u standardnoj teoriji funkcionala gustoće: gustoća elektrona raspisuje se po baznim funkcijama koje sadrže određene koeficijente. Koeficijenti se tradicionalno moraju prilagoditi svakoj konfiguraciji molekula te njihova prilagodba postaje znatno kompliciranija s većim brojem atoma i molekula u sustavu. U radu [8] zadatak određivanja koeficijenata prepuštaju GNN-u, tako da se za velike sustave ponovno gradi baza, a nekoć kompliciranо predviđanje koeficijenata u raspisu baze se obavlja vrlo brzo uz GNN-ove. Na slici 2.3 prikazujemo primjer elektronskih gustoća nakupine molekula vode.

S druge strane mogu se direktno predviđati svojstva poput međuatomskih sila, energija ionizacije, ukupne energije, dipolnog momenta itd. [4, 9, 10]. I u ovim se problemima nastoji ugraditi SO(3), O(3) ili pak E(3) grupa simetrija.

Sve od navedenog motiviralo nas je da pokušamo sličnom metodom predvidjeti sile među atomima organskih molekula te provjeriti generalizacijska svojstva ekvivarijantih modela na dosad neviđenim i većim molekulama.



Slika 2.3: Primjer kvantnog kompleksnog sustava: gustoće elektrona nakupina molekula vode. Pri modeliranju svojstava ovakvog sustava bitno je paziti na rotacijsku invarijantnost elektronske gustoće. Skala na x-y-z osima je u Å.

3 Molekule kao kompleksni sustavi

Molekule su kvantni sustavi više čestica, konkretno elektrona i atomskih jezgara, koje međusobno interagiraju Coulombski. Iz navedenog opisa lako se vidi kako su molekule kompleksni sustavi, radi se o sustavim sastavljenog od više dijelova (atoma) pri čemu raznolikost svojstava i dinamike sustava poizlazi iz interakcije pojedinih dijelova.

3.1 Rješavanje Schöedingerove jednadžbe

Čitava molekularna dinamika opisana je dobro poznatom nerelativističkom Schröedingerovom jednadžbom

$$i\hbar\partial_t\psi(t,\{\mathbf{r}_j\}_N) = \sum_k^N \Big(-\frac{\hbar^2}{2m_k}\nabla_j^2\psi(t,\{\mathbf{r}_j\}_N)\Big) + V\big(t,\{\mathbf{r}_j\}_N\big)\psi(t,\{\mathbf{r}_j\}_N) \tag{3.1}$$

Često nas zanimaju stacionarna svojstva molekula zbog čega je dovoljno rješavati, jednostavniju, stacionarnu Schröedingerovu jednadžbu

$$\sum_k^N \frac{\hbar^2}{2m_k}\nabla_j^2\psi(\{\mathbf{r_j}\}_N) + V\big(\{\mathbf{r}_j\}_N\big)\psi(\{\mathbf{r}_j\}_N) = E\psi(\{\mathbf{r}_j\}_N) \tag{3.2}$$

U gornjem izrazu oznaka $\{\mathbf{r}_j\}_N$ označava skup prostornih koordinata svih N čestica. Smatramo da su jezgre atoma zasebne kvantne čestice, čime zanemarujemo njihovu unutarnju strukturu, a time i bilo kakvu jaku ili slabu nuklearnu interakciju. Također, promatramo molekule pri energijama na kojim je svaka promjena elektromagnetskog polja trenutna i time opisana zakonima elektrostatike. Dakle preostaje jedino Coulombska interakcija tipa elektron-elektron (V_{ee}), jezgra-jezgra (V_{nn}) i elektron-jezgra (V_{en}). Odnosno

$$V\big(\{\mathbf{r}_j\}_N\big) = V_{nn} + V_{en} + V_{ee} \tag{3.3}$$

$$V_{nn} = \sum_{i<j}^{N_n} \frac{Z_i Z_j e^2}{|\mathbf{r}_{ni} - \mathbf{r}_{nj}|} \quad V_{en} = \sum_i^{N_n}\sum_j^{N_e} \frac{-Z_i e^2}{|\mathbf{r}_{ni} - \mathbf{r}_{ej}|} \quad V_{ee} = \sum_{i<j}^{N_e} \frac{e^2}{|\mathbf{r}_{ei} - \mathbf{r}_{ej}|} \tag{3.4}$$

Unatoč pojednostavljenjima fizikalne slike, jednadžba 3.1 je i dalje analitički i numerički ne rješiva za bilo koji veći broj elektrona i/ili jezgara, stoga uvodimo dodatne aproksimacije. Jedna od najbitnijih aproskimacija Schröedingerove jed-

nadnžbe za molekularne sustave jest Born-Oppenheimerova aproskimacija. Born-Oppenheimerovom aproksimacijom nastojimo razdvojiti stupnjeve slobode jezgara od stupnjeva slobode elektrona, pri čemu je glavni argument taj da je dinamika atomskih jezgara puno sporija od dinamike elektrona. Detaljan izvod se može pronaći u [11], no krajnji rezultat daje dvije pojednostavljene jednadžbe, jednu za elektrone 3.5 i jednu za atomske jezgre 3.6.

$$\Big[-\sum_{k=1}^{N_e} \frac{\hbar^2}{2m_e}\nabla_k^2 + V_{ee}(\{\mathbf{r}_e\}) + V_{ne}(\{\mathbf{r}_n\},\{\mathbf{r}_e\})\Big]\phi_e(\{\mathbf{r}_e\}) = E_e(\{\mathbf{r}_n\})\phi_e(\{\mathbf{r}_e\}) \tag{3.5}$$

$$\Big[-\sum_{k=1}^{N_n} \frac{\hbar^2}{2m_k}\nabla_k^2 + V_{nn}(\{\mathbf{r}_n\}) + E_e(\{\mathbf{r}_n\})\Big]\psi_n(\{\mathbf{r}_n\}) = E\psi_n(\{\mathbf{r}_n\}) \tag{3.6}$$

Bitno je za naglasiti da u elektronskoj jednadžbi 3.5 uzimamo koordinate svih jezgara $\{\mathbf{r}_n\}$ kao fiksne veličine te interakciju s elektrona s jezgrama tretiramo semi-klasično.

3.1.1 Hatree-Fockove jednadžbe

Born-Oppenheimerovom aprokismacijom zaista dobijemo jedonstavnije jednadžbe, no preostaje glavni problem Schröedingerove jednadžbe, a to je njezina dimenzionalnost. U slučaju N čestica, domena rješenja je 3N dimenzionalni prostor. Računalnim metodama ovo predstavlja problem pri aproksimaciji kontinuuma. Na primjer, rješavamo li jednadžbu na prostornoj domeni oblika kocke s bridom duljine 1 te uzmemo li za udaljenost između točaka rešetke od 10^{-1} ukupan broj točaka prostora koordinata svih čestica bio bi 10^{3N}. Analogno za proizvoljnu duljinu kocke L i duljine diskretizacije Δx broj potrebnih točaka raste eksponencijalno s brojem čestica. Osim rasta broja točaka u više dimenzija raste i njihova međusobna udaljenost zbog čega je aproksimacija kontinuuma sve lošija. Ova se pojava zove *prokletstvo dimenzionalnosti*.

Rješenja prokletstva dimenzionalnosti dolaze u obliku svođenja jednadžbe 3.5 na jednočestični oblik, putem Hatree-Fockove aproksimacije. U nastavku bavimo se isključivo rješavanjem elektronske jednadžbe 3.5 te ispuštamo eksplicitnu ovisnost o pozicijama jezgara $\{\mathbf{r}_n\}$. Ideja je separirati gibanje pojedinih elektrona, odnosno ukupnu elektronsku valnu funkciju zapisati kao produkt jednočestičnih valnih funkcija. Bitno je uzeti u obzir ovisnost elektronske valne funkcije o spinu u kontekstu Paulijevog principa isključenja, koji nam govori da elektronska valna funkcija mora biti antisimetrična s obzirom zamjenu para elektrona. U Hatree-Fockovoj aproksima-

ciji to se svojstvo ugrađuje uzimanjem rješenja u obliku Slaterove determinante, koja sadrži ortonormirane jednočestične valne funkcije $\psi(\mathbf{r})_i$

$$\phi(r_1, r_2, ...r_{N_e}) = \frac{1}{\sqrt{N_e!}} \begin{vmatrix} \psi_{i1}(\mathbf{r}_1) & ... & \psi_{i1}(\mathbf{r}_{N_e}) \\ \vdots & \vdots & \vdots \\ \psi_{iN_e}(\mathbf{r}_1) & ... & \psi_{iN_e}(\mathbf{r}_{N_e})_{i1} \end{vmatrix} \tag{3.7}$$

Uz ovaj ansatz i primjenu varijacijskog računa na Hamiltonijan sustava na dobivamo Hatree-Fockove jednadžbe za osnovno stanje molekule. Hatree-Fockove jednadžbe rješavaju se samosuglasno uz nekakvu inicijalnu pretpostavku o funkcijskom obliku rješenja $\psi_i(\mathbf{r})$.

$$\Big[-\frac{\hbar^2}{2m_e}\nabla^2 + V_{ne}(\mathbf{r}) + V_i^H(\mathbf{r}) + V_i^F(\mathbf{r}) \Big] \psi_i(\mathbf{r}) = \epsilon_i \psi_i(\mathbf{r}) \tag{3.8}$$

$$V_i^H(r) = \sum_{j \neq i} \int \mathrm{d}\mathbf{r}' \frac{e^2 |\psi_j(\mathbf{r}')|^2}{|\mathbf{r} - \mathbf{r}'|} \tag{3.9}$$

$$V_i^F(\mathbf{r}) = -\sum_{j \neq i} \int \mathrm{d}\mathbf{r}' \frac{e^2}{|\mathbf{r} - \mathbf{r}'|} \psi_j^*(\mathbf{r}') \psi_i(\mathbf{r}') \psi_j(\mathbf{r}) \delta_{\sigma\sigma'} \tag{3.10}$$

Hatreejev potencijal V^H odgovara usrednjenoj elektron-elektron interakciji, dok se Fockov član izmjene V^F dobije kada se uzme u obzir Paulijev princip isključenja. Kroneckerov simbol $\delta_{\sigma\sigma'}$ osigurava da član izmjene postoji jedino u slučaju kada dva jednočestična stanja imaju isti spin.

3.2 Teorija funkcionala gustoće

Uvođenjem Hatree-Fockove aproksimacije zaista rješavamo prokletstvo dimenzionalnosti Schöedingerove jednadžbe, no i dalje preostaje riješiti N_e samosuglasnih Hatree-Fockovih jednadžbi. Trenutno najuspješniji pristup modeliranja molekularnih sustava temelji se na teoriji funkcionala gustoće (DFT). Temeljito objašnjenje DFT-a bilo bi preopširno za ovaj rad stoga ovdje komentiramo temeljne ideje i principe, za detalje referiramo čitatelja na [12, 13]. Dva fundementalna rezultata na koje se oslanja DFT su Kohn-Hohenbergovi teoremi.

***Teorem* 3.1 (1. Kohn-Hohenberg)** *Energija osnovnog stanja Schöedingerove jednadžbe jedinstveni funkcional gustoće elektrona* $n(\mathbf{r})$.

***Teorem* 3.2 (2. Kohn-Hohenberg)** *Gustoća elektrona koja minimizra energiju energiju ukupnog funkcionala jest prava gustoća elektrona koja korespondira rješenju pune Schöedingerove jednadžbe.*

Teoremi 3.1, 3.2 su od iznimne važnosti jer s njima možemo ne samo smanjiti dimenzionalnost problema s $3N$ na 3, već trebamo rješavati jednu varijacijku jednadžbu umjesto više Hatree-Fockovih. Za početak definiramo elektronsku gustoću izrazom

$$n(\mathbf{r}) = \sum_{\sigma} \sum_{i} |\psi_i(\mathbf{r})|^2 \tag{3.11}$$

Pri čemu σ označava spinska stanja elektrona, a $\psi_i(\mathbf{r})$ jednočestične elektronske valne funkcije. Energetski funkcional gustoće danoj funkciji gustoće elektrona pridružuje energiju, a po teoremu 3.1 to je upravo $E_e(\{\mathbf{r}\}_n, \{\mathbf{r}\}_e)$ iz izraza 3.6. Energetski funkcional gustoće rastavljamo na četiri jednostavnijih funkcionala

$$\hat{E}[n(\mathbf{r})] = \hat{T}[n] + \hat{E}^H[n] + \hat{E}^{ext}[n] + \hat{E}^{xc}[n] \tag{3.12}$$

$$\hat{T}[n] = \sum_{\sigma} \sum_{i} - \int \mathrm{d}\mathbf{r} \psi_i^*(\mathbf{r}) \frac{\hbar^2 \nabla^2}{2m_e} \psi_i(\mathbf{r}) \tag{3.13}$$

$$\hat{E}^H[n] = \frac{1}{2} \int \mathrm{d}\mathbf{r} \int \mathrm{d}\mathbf{r}' \frac{e^2 n(\mathbf{r}') n(\mathbf{r})}{|\mathbf{r} - \mathbf{r}'|} \tag{3.14}$$

$$\hat{E}^{ext}[n] = \int \mathrm{d}\mathbf{r} V^{ext}(\mathbf{r}) n(\mathbf{r}) \tag{3.15}$$

Kinetički $\hat{T}[n]$ funkcional, Hatreejev funkcional $\hat{E}^H[n]$ te funkcional interakcije s vanjskim poljem $\hat{E}^{ext}[n]$ su općenito poznati dok funkcional izmjene $\hat{E}^{ex}[n]$ općenito nije poznat i izvor je najvećeg broja problema. Postupak pronalaska minimalne energije odvija se samosuglasno, tako da se prvo pretpostavi inicijalna gustoća elektrona i pripadni funkcional izmjene čime se mogu dobiti jednočestične valne funkcije, zatim se se ponovno računa gustoća iz jednočestičnih valnih funkcija. Cijelo vrijeme prati se iznos dobivene energije. Ovaj se proces ponavlja sve dok vrijednost energije ne konvergira. Funkcionalni oblik gustoće elektrona se zapsiuje putem baznih funkcija, koje su često ravni valovi ili pak sferni harmonici modulirani gausijanima. Koliko baznih funkcija uzeti za dani problem i koje su bazne funkcije prikladnije, najčešće se bira iskustveno ili heuristički.

Svaki numerički postupak rješavanja Schröedingerove jednadžbe posjeduje nekakvu kompleksnost s obzirom na broj čestica. To je teorijska vrijednost koja daje procjenu broja operacija potrebnih za izvršenje računalnog postupka i to u asimptotskom režimu broja čestica. Tipično skaliranje DFT procedura je $O(N^3)$ što znači da broj operacija raste s kubom broja čestica sustava. U usporedbi s ostalim metodama, DFT ima poprilično dobro skaliranje, međutim kubno skaliranje je i dalje računalno zahtjevno pa se ne može koristiti za modeliranje velikih kvantnih sustava. Ovaj se problem skaliranja pokušava riješiti uporabom DNN-ova.

3.3 Motivacija i formulacija problema

Postavlja se očito pitanje: ako već postoje razni simulatori, zašto bismo trenirali DNN na postojećim podatcima simulacija? Radovi poput [4, 8, 14] upućuju na na gotovo linearno skaliranje vremenske kompleksnosti dubokih modela s brojem čestica i to u raznim primjerima bilo predviđanje međuatomskih sila, elektronskih gustoća ili energetskih svojstava molekule. Dakle jedan razlog je svojevrsno recikliranje podataka simulacija kako bi se mogli modelirati sustavi puno većeg broja čestica, koji su računalno prezahtjevni za klasične simulatore. Osim toga nadamo se da će dobro istreniran model *naučiti dovoljno fizike* te moći precizno modelirati ne samo veće sustave iste vrste, nego i dosad *ne viđene*[1] sustave. Ovu hipotezu provjeravamo na skupu podataka Md17.

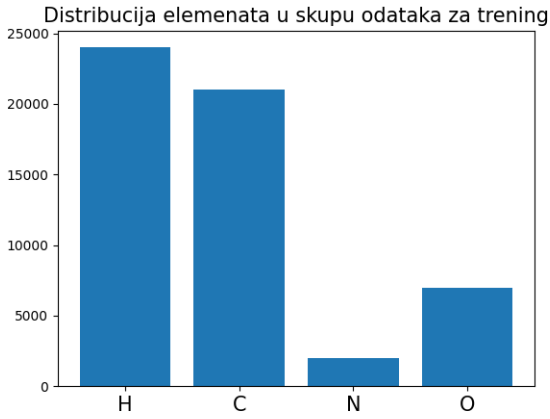
3.3.1 Skup podataka Md17

Skup podataka Md17 sastoji se o raznih ab-initio simulacija molekularne dinamike te je javno dostupan na stranici [15], dok su detalji o izgradnji samog skupa izloženi u radu [16]. Skup podataka Md17 nudi primjere molekularnih konfiguracija koje su opisane slijedećim veličinama: pozicijama pojedinih atoma u nekom koordinatnom sustavu, tipovima atoma, silom na svaki atom te ukupnom energijom molekule. Potonje dvije veličine predviđene su kao regresijske mete za modele strojnog učenja. U ovom radu fokusiramo se samo na predviđanje sila. Sve veličine udaljenosti u skupu podataka mjere se u angstremima Å, energije u $\frac{\text{kcal}}{\text{mol}}$, a sile u $\frac{\text{kcal}}{\text{molÅ}}$. Cilj je iz dane konfiguracije pozicija i vrsta atoma predvidjeti silu na svaki atom. Izgradnja grafova

[1] Pod pojmom *ne viđeni sustav* smatramo sustave čiji podatci nisu prisutni u podatcima za trening.

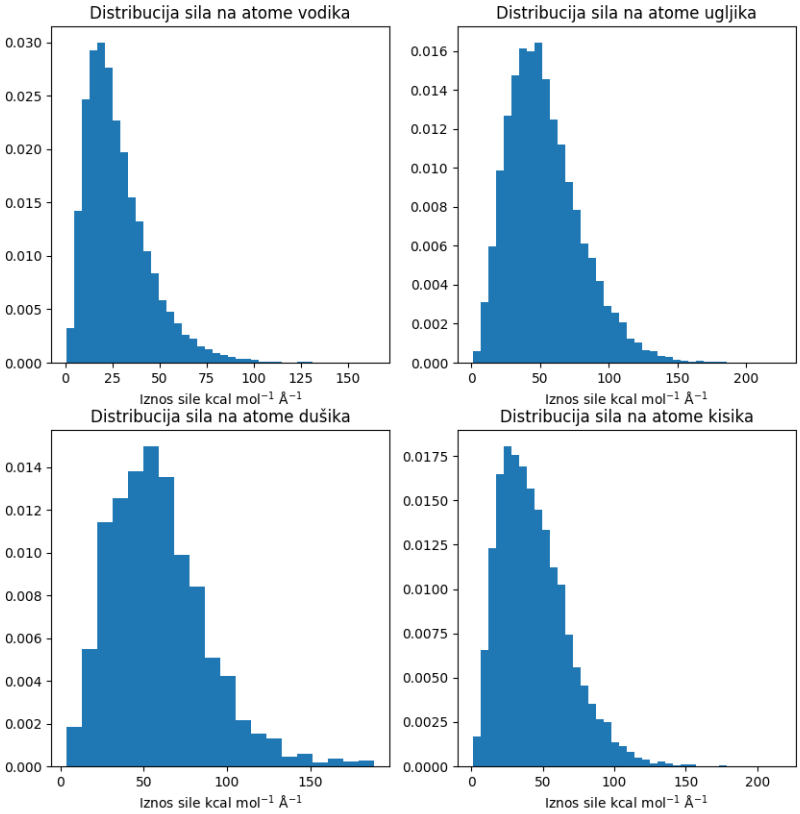
i detalji o modeliranju objašnjeni su u poglavlju 7.

Ne preporuča se koristiti čitav Md17 za trening, stoga uzimamo po tisuću konfiguracija molekula benzna, uracila, etanola i aspirina. Za validaciju i testiranje uzimamo po još tisuću konfiguracija za svaku molekulu. Za kraj uzimamo tisuću konfiguracija molekule paracetamola za provjeru generalizacije modela. Hipoteza jest da će model predviđati gotovo jednako dobro sile paracetamola, ako dobro predviđa sile na uracilu, etanolu, aspirinu i benzenu. Bitno je naglasiti da model ne može biti primijenjen na molekule koje sadrže atome koji nisu prisutni u skupu podataka za trening. Konkretno molekulske formule benzena, etanola, uracila i aspirina su redom C_6H_6, C_2H_6O, $C_4H_4N_2O_2$, $C_9H_8O_4$, dok je formula paracetamola $C_8H_9NO_2$. Vidi se kako se svi atomi paracetamola mogu pronaći u nekoj od molekula skupa podataka za trening. Međutim, nisu svi atomi jednako zastupljeni u skupu podataka za trening, distribucija elemenata u skupu podataka za trening prikazana je na slici 3.1.



Slika 3.1: Zastupljenost elemenata u skupu podataka za trening, vidi se kako, je vodik daleko najzastupljeniji, dok je dušik skoro deset puta manje prisutan u podatcima.

Osim zastupljenosti atoma provjeravamo i distribuciju iznosa sila na pojedini atom nju prikazujemo na slici 3.2. Svi atomi izuzev vodika imaju sličnu očekivanu silu, varijancu i oblik distribucije. Sam vodik prati sličan trend kao i ostali atomi uz blagi pomak srednje vrijednosti prema manjim iznosima sile.



Slika 3.2: Distribucija iznosa sila po atomu u skupu podataka za učenje. Čini se da sile svih atoma osim vodika prate istu distribuciju, distribucija sila na vodik je pomaknuta prema nižim vrijednostima. Bitno je naglasiti da se vrijednosti sila prožimaju kroz nekoliko redova veličina. Ovo postaje bitna informacija pri odabiru metrike za evaluaciju modela.

4 Osnove dubokog učenja

Budući da u ovom radu za modeliranje kompleksnih sustava koristimo modele dubokog učenja, potrebno je objasniti način na koji DNN-ovi funkcioniraju te motivirati njihovo korištenje u obliku univerzalnog aproksimacijskog teorema. Kroz ovo poglavlje objasnit ćemo tri komponente svakog postupka strojnog učenja, arhitekturu i optimizaciju modela dubokog učenja, te konačno iskazat ćemo univerzalni aproksimacijski teorem u jednoj od svojih formi.

4.1 Tri komponente strojnog učenja

Područje dubokog učenja je grana strojnog učenja koja se temelji na korištenju DNN-ova. Iz tog razloga smatramo je korisno objasniti temeljne principe strojnog učenja, koji su također primijenijvi u dubokom učenju. Tipični problemi u nadziranom strojnom učenju svode se na predviđanje svojstava nekih objekata. Neka je $\mathcal{X}$ skup značajki kojim identificiramo svaki objekt, na način da ako dva objekta posjeduju svojstva $x_1, x_2 \in \mathcal{X}$ tada su oni neraspoznatljivi. Svakom je objektu pridružena vrijednost y iz skupa $\mathcal{Y}$. Cilj postupka strojnog učenja jest što točnije aproksimirati funkciju

$$\widehat{f} : \mathcal{X} \longrightarrow \mathcal{Y}$$

iz konačnog skupa opažanja $\mathcal{D} = \big\{(x_i, y_i) \,|\, x_i \in \mathcal{X},\, y_i \in \mathcal{Y},\, i = 1, ..., N\big\}$. Tri komponente postupka strojnog učenja su model, funkcija pogreške i optimizacijski postupak. Model je skup svih funkcija kojima pokušavamo aproksimirati $\widehat{f}$. Najčešće model definiramo putem skupa parametriziranih funkcija $f : \Theta \times \mathcal{X} \longrightarrow \mathcal{Y}$, gdje je Θ prostor parametara aproksimacijskih funkcija. Za pojedini izbor parametara $\theta \in \Theta$, aproksimacijsku funkciju označavamo s f_θ, a model $\mathcal{M} = \big\{f_\theta \,|\, \theta \in \Theta\big\}$.

Jednom kada su definirani model i skup opažanja $\mathcal{D}$ potrebno je pronaći konkretne parametre $\theta \in \Theta$ kojim najbolje aproksimiramo $\widehat{f}$. Budući da skup parametara može biti ogroman ili pak kontinuum, direktno pretraživanje skupa Θ nije opcija, stoga se uvodi heuristika koju zovemo funkcija pogreške. Neformalno, funkcija pogreške je funkcija

$$\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \longrightarrow \mathbb{R}$$

kojom opisujemo bliskost između izlaza funkcije $\widehat{f}$ i aproksimacije f_θ. Funkcije po-

greške razlikuju se od problema do problema te uvelike ovise o svojstvima skupova $\mathcal{X}$ i $\mathcal{Y}$. U ovom se radu bavimo problemima regresije, stoga je tipičan izbor funkcije gubitka srednje kvadratno odstupanje. Srednje kvadratno odstupanje pogodan je izbor funkcije gubitka jer minimiziranje njezine vrijednosti odgovara maksimiziranju izglednosti podataka za dani model [17].

$$\mathcal{L}(\mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} \big(y_i - f_\theta(x_i)\big)^2 \tag{4.1}$$

Funkcija pogreške služi kao kriterij kojim pretražujemo skup aproksimacijskih funkcija te razlikujemo bolje aproksimacije od lošijih. Najbolje aproksimacija bit će ona koja minimizira funkciju pogreške[2]. Minimum funkcije pogreške tražimo optimizacijskim postupkom.

Optimizacijski postupak ovisi o prirodi problema, izboru modela i dakako o izboru funkcije gubitka. U slučaju DNN-ova optimizacija se provodi procjenom gradijenta funkcije pogreške u ovisnosti s parametrima modela, više o tome u idućem potpoglavlju.

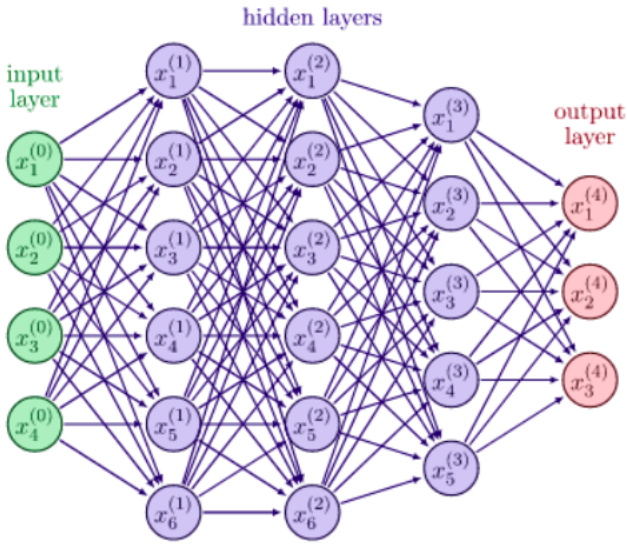
4.2 Modeli dubokog učenja

U 2022. godini, na jednoj od najvećih i najprestižnijih konferencija o strojnom učenju i dubokom učenju, NeurIPS bilo je čak 2905 prihvaćenih članaka [18]. Ta informacija ilustrira brzinu kojom se područje dubokog učenja razvija, zbog toga panoramski pregled postojećih arhitektura bio bi preopširan za ovaj rad. Ovdje ćemo objasniti arhitekturu koja se u raznim oblicima provlači kroz gotovo svaki model dubokog učenja - potpuno povezanu neuronsku mrežu. U poglavlju 5. fokusirat ćemo se na trenutno, najopćenitiju arhitekturu: neuronske mreže nad grafovima (*eng.*graph neural network).

4.2.1 Potpuno povezana neuronska mreža

Potpuno povezana neuronska mreža nastala je pokušajem modeliranja interakcije bioloških neurona u mozgu. Opisno, ona se sastoji od slojeva neurona i veza između njih. Veze modeliraju jakost interakcije neurona u susjednim slojevima. Slojeve dije-

[2]Analogno se može i definirati funkcija uspjeha kao $-\mathcal{L}$, kojoj tražimo maksimum.



Slika 4.1: Potpuno povezana neuronska mreža: Gornji indeksi označavaju redni broj sloja, dok donji označavaju komponentu tog sloja. Strelice označavaju linearnu transformaciju koja povezuje dva susjedna sloja, nakon linearne transformacije primjenjuje se proizvoljna linearna funkcija. Zeleni neuroni predstavljaju ulaz modela, plavi skrivene slojeve modela, a crveni izlaz modela. Preuzeto iz [19].

limo na ulazni sloj, izlazni sloj te skrivene slojeve. Primjer mreže s tri skrivena sloja dan je na slici 4.1.

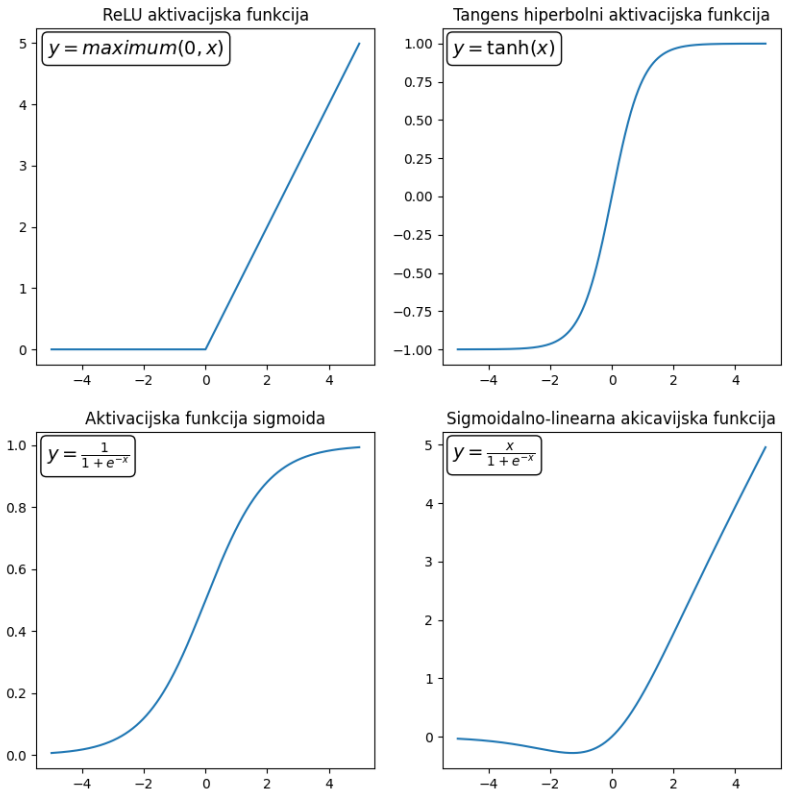
Ideja u originalnom radu [20] iz 1958. bila je da prilikom učenja nekog zadatka, biološki neuroni ostvaruju optimalnu povezanost za taj zadatak, ako je *podražaj* susjednih neurona dovoljno velik on će aktivirati neuron na koji susjedi djeluju. Ovaj dio procesa modeliran je aktivacijskom funkcijom. Originalno ona je bila step funkcija, međutim s vremenom su se češće koristile druge aktivacijske funkcije prikazane na slici 4.2.

Formalnije rečeno, potpuno povezana neuronska mreža je funkcija sastavljena od niza afinih transformacija i nelinearnih aktivacija na slijedeći način.

***Definicija* 4.1 (Neurnoska mreža)** *Neka je $\mathcal{S} = \{s_i | i = 1...N\}$ skup slojeva neuronske mreže, i-ti sloj s_i je uređena trojka $s_i = (W_i, b_i, a_i)$ matrice težina $W_i \in \mathbb{R}^{n_i \times m_i}$, vektora pomaka $b_i \in \mathbb{R}^{n_i}$ i aktivacije $a_i : \mathbb{R}^{n_i} \longrightarrow \mathbb{R}^{n_i}$. Definiramo preslikavanje među susjednim slojevima kao*

$$f_l = a_l(W_l x_{l-1} + b_l)$$

Potpuno povezana neuronska mreža $\mathcal{N} : \mathbb{R}^{N_o} \longrightarrow \mathbb{R}^N$ je kompozicija svih preslikavanja



Slika 4.2: Često korištene aktivacijske funkcije. Većina ih nastoji modelirati odziv neurona na određenu jakost podražaja. Bitno je svojstvo aktivacijskih funkcija nelinearnost, bez toga DNN-ovi nebi bili univerzalni akproksimatori.

slojeva

$$\mathcal{N}(x) = f_N \circ f_{N-1} \circ ... \circ f_1(x)$$

Prisutnost nelinearnih aktivacija nužna je kako bi se neuronskom mrežom mogle aproksimirati bilo koje funkcije osim afinih i linearnih. Njihova važnost iskazana je u teoremu 4.1. Prije rasprave o univerzalnom aproksimacijskom teoremu, ukratko ćemo opisati optimizaciju neuronskih mreža.

4.2.2 Optimizacija i algoritam širenja pogreške unazad

Slojevita struktura neuronske mreže omogućuje jednostavan proces optimizacije. U ovom potpoglavlju opisat ćemo postupak optimizacije dubokih modela.

Neka $\mathcal{N}_\Theta$ je neuronska mreža, $\mathcal{X} = \{x_i | i = 1...N\}$ skup primjera te $\mathcal{Y} = \{y_i | i = 1...N\}$ skup oznaka primjera. Ovdje Θ označava skup parametara neuronske mreže

$\Theta = \{W_l, b_l \,|\, l = 1...M\}$, gdje je M broj slojeva mreže. Cilj je da za svaki $x \in \mathcal{X}$, vrijednost $||\mathcal{N}(x_i) - y_i||_2^2$ bude što manja, odnosno da funkcija pogreške $\mathcal{L}(\Theta) = \frac{1}{N}||\mathcal{N}(x_i) - y_i||_2^2$ postigne globalni minimum. Optimizacija se provodi gradijentnim metodama kojima je potrebna informacija o $\partial_{\theta_i}\mathcal{L}\ \forall\theta_i \in \Theta$. Pri konstrukciji neuronske mreže koriste se nelinearnosti čiji se gradijenti poznaju analitički te se jednostavno računaju.

Neka je izlaz modela $\mathcal{N}(x) = \widehat{y}$, za dani $\widehat{y}$ možemo izračunati funkciju pogreške $\mathcal{L}$ i gradijent funkcije pogreške po izlazu modela $\frac{\partial\mathcal{L}}{\partial\widehat{y}}$. Informacija o $\frac{\partial\mathcal{L}}{\partial\widehat{y}}$ pruža se posljednjem sloju mreže te se s njom računaju dvije vrste gradijenata: $\frac{\partial\mathcal{L}}{\partial\theta_i^{M-1}}$ i $\frac{\partial\mathcal{L}}{\partial f_{M-1}}$, gdje su θ_i^{M-1} parametri posljednjeg sloja, a f_{M-1} izlaz pretposljednjeg sloja, odnosno ulaz posljednjeg sloja. Gradijent $\frac{\partial\mathcal{L}}{\partial\theta_i^{M-1}}$ koristi se za optimizaciju parametara θ_i^{M-1}, dok se $\frac{\partial\mathcal{L}}{\partial f_{M-1}}$ pruža pretposljednjem sloju. Radi se samo o uzastopnoj primjeni lančanog pravila te se na taj način čini da greška *putuje unazad* kroz mrežu.

Opisani postupak služi kao *baza* induktivnog postupka za računanje gradijenta funkcije pogreške po parametrima dubljih slojeva. Neki l-ti sloj od svog sljedbenika će primiti informaciju o $\frac{\partial\mathcal{L}}{\partial f_l}$. Koristeći lančano pravilo i definiciju 4.1 može gradijente po parametrima 4.2 4.3 i ulazima sloja 4.4

$$\frac{\partial\mathcal{L}}{\partial W_l} = \frac{\partial\mathcal{L}}{\partial f_l}\frac{\partial f_l}{\partial W_l} = \frac{\partial\mathcal{L}}{\partial f_l}a_l' f_{l-1} \tag{4.2}$$

$$\frac{\partial\mathcal{L}}{\partial b_l} = \frac{\partial\mathcal{L}}{\partial f_l}\frac{\partial f_l}{\partial b_l} = \frac{\partial\mathcal{L}}{\partial f_l}a_l' \tag{4.3}$$

$$\frac{\partial\mathcal{L}}{\partial f_{l-1}} = \frac{\partial\mathcal{L}}{\partial f_l}\frac{\partial f_l}{\partial f_{l-1}} = \frac{\partial\mathcal{L}}{\partial f_l}a_l' W_l \tag{4.4}$$

Kada su gradijenti po svim parametrima modela poznati većina optimizacijskih algoritama radi malene promjene parametara u suprotnom smjeru gradijenta pogreške, odnosno u n-tom koraku optimizacije neki parametar θ poprima vrijednost

$$\theta_n = \theta_{n-1} - \eta\frac{\partial\mathcal{L}}{\partial\theta} \tag{4.5}$$

gdje je η stopa učenja neki malen broj tipično u rasponu od $5 \cdot 10^{-2}$ do 10^{-5}. Optimizacija ovog tipa zove se gradijentni spust. U pokusima provedenim u ovom radu koristi se unaprijeđena inačica gradijentnog spusta Adam [21].

4.2.3 Aproksimacijska svojstva modela dubokog učenja

Kao što smo najavili na početku ovog poglavlja motivacija iza korištenja DNN-ova na širokom spektru problema jest što su oni univerzalni aproksimatori. Samo područje dubokog učenja plodno je novim modelima te se ne rijetko dogodi da je arhitektura korištena bez potpunog poznavanja njenih svojstava, međutim za arhitekture poput DNN-a s jednim ili više skrivenih slojeva poznato je pod kojim uvjetima postaju univerzalni aproksimatori. U nastavku ćemo skicirati iskaz univerzalnog aproksimacijskog teorema, a za detalje ostavljamo [22, 23]. Kompliciranije arhitekture često su svojom matematičkom formom prilagođene problemu na koji se primjenjuju [24–26], a u sebi sadrže DNN-ove na čija se aproksimacijska svojstva oslanjaju.

***Teorem* 4.1 (Skica: Unverzalni aproksimacijski teorem)** *Neka je $\Omega \subseteq \mathbb{R}^n$ kompaktan skup, te neka je $C(\Omega)$ skup svih neprekidnih funkcija $f : \Omega \longrightarrow \mathbb{R}$. Neka je na skupu Ω zadana mjera μ te neka je na skupu $C(\Omega)$ zadana norma $|| \cdot || : C \longrightarrow \mathbb{R}^+$, oblika $||f|| = \int_\Omega \mathrm{d}\mu(x)|f(x)|^2$. U [19] i [22] autori pokazuju kako je skup funkcija*

$$\mathcal{G} = \big\{ g \,|\, g(x) = \sum_j^m w_j^{(2)} \mathbf{a}\big(\sum_i^n w_{i,j}^{(1)} x_i + b_j \big); m, n \in \mathbb{N} \big\}$$

gust u skupu $C(\Omega)$ pod uvjetom da je aktivacijska funkcija $\mathbf{a}$ nelinearnost nepolinomijalnog tipa. Ako je sve od navedenog zadovoljeno tada $\forall \epsilon > 0 \; \exists g \in \mathcal{G}$ takav da $||f - g|| < \epsilon$.

Dakle za proizvoljnu funkciju f i traženu preciznost ϵ možemo pronaći funkciju unutar $\mathcal{G}$ koja će ju aproksimirati s pogreškom manjom od ϵ. Preostaje problem s brojevima m i n, naime potreban broj parametara mreže $w_j^{(2)}$ i $w_i^{(1)}$ raste eksponencijalno s traženom preciznosti. Za prilagodbu većeg broja parametara potrebna je i veća količina podataka. Iz tog razloga pogodnije je u arhitekture ugraditi neko prethodno znanje o problemu čime se znatno smanjuje broj potrebnih parametara. U ovom radu istražit ćemo ugradnju poznatih simetrija fizikalnih sustava kako bi se poboljšala efikasnost modela s obzirom na broj podataka te generalizacijska sposobnost modela.

5 Neuronske mreže nad grafovima

U poglavlju 4. opisali smo princip rada neuronske mreže na najjednostavniji geometrijski objekt: vektor (točku). Česti problemi u praksi zahtijevaju obrađivanje podataka koji su na neki način povezani i/ili uređeni, primjerice slike možemo shvatiti kako 2-D vektorska polja, gdje je svakoj točki pridružen 3-D vektor intenziteta piksela, slični se problemi javljaju u obradi teksta, riječi se reprezentiraju vektorima, i ti su vektori vremenski uređeni. U fizici važnost vremenskog uređaja (točnije kauzalnog uređaja) i lokalne povezanosti postaje još izraženija, stoga ako želimo modelirati netrivijalne fizikalne sustave putem DNN-ova, trebat će nam nešto složenije od potpuno povezane neuronske mreže. Iz tog razloga su izmišljene arhitekture poput konvolucijskih neuronskih mreža, transformera te povratnih neuronskih mreža [24, 25, 27]. Autori [26] pokazuju kako su kombinatorni grafovi poopćenje svih navedenih domena te da su spomenute arhitekture samo posebni slučajevi neuronskih mreža nad grafovima. Ostatak poglavlja posvećujemo se grafovima, kako se njima opisuju nepravilne i kompliciranе fizikalne domene te kako primijeniti DNN-ove na njih što rezultira GNN-ovima.

5.1 Grafovi

***Definicija* 5.1 (Graf)** *Graf G je uređeni par (V, E). Neprazan skup $V = \{v_i\}$ zovemo skup vrhova, dok $E \subseteq V \times V$ zovemo skup bridova.*

Skup vrhova predstavlja elemente grafa, npr. atome u kristalnoj rešetci ili molekuli, dok bridovi predstavljaju egzistenciju interakcije među elementima grafa, npr. vrh (v_i, v_j) označava da postoji interakcija između atoma v_i i v_j. Zbog toga što je definicija grafa toliko apstraktna njome je jednostavno modelirati vrlo kompliciranе diskretne geometrijske domene, a time i razne fizikalne sustave.

Najčešće je vrhovima potrebno pridružiti određena svojstva, recimo u modeliranju molekula grafom, moguće je svakom vrhu pridružiti vrstu atoma, masu atoma te poziciju atoma u nekom koordinatnom sustavu. Time je svakom vrhu pridružena petorka realnih brojeva. Ove vrijednosti zovemo značajke vrhova, one su najčešće vektori. Skup svih mogućih vrijednosti koje značajke x mogu poprimiti zovemo skup značajki $\mathcal{X}$.

U principu skup vrhova V može sadržavati potpuno apstraktne matematičke objekte kojima označavamo vrhove, recimo vektori, funkcije ili pak skupovi. Jednom kada smo uveli skup značajki $\mathcal{X}$ u njega pospremamo sva kompliciranа ili apstraktna svojstva vrhova, dok skup V sadrži samo indekse kojima označavamo vrhove.

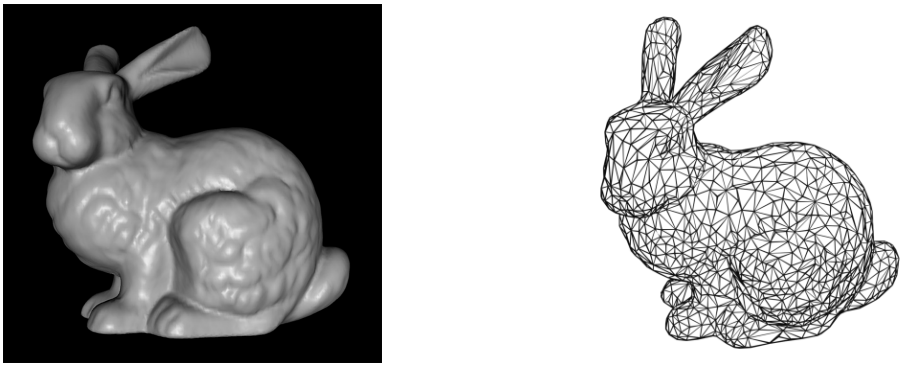
Bridovima grafa također je moguće pridružiti značajke. Naime u odsustvu skupa značajki bridova, pojedini brid označava egzistenciju nekakve interakcije ili odnosa među dvama vrha te ne donosi nikakvu dodatnu informaciju o prirodi interakcije ili vrsti odnosa. Dodatne informacije o vrsti interakcije modeliramo značajkama bridova $\mathcal{E}$.

5.1.1 Prikazivanje mnogostrukosti na računalu korištenjem grafova

Dvodimenzionalne rešetke nad kojima rade konvolucije [24] i vremenski sljedovi nad kojima rade transformeri [25] su diskretna reprezentacija signala (funkcija) na mnogostrukostima. Rešetke u dvije dimenzije su diskretne reprezentacije ravnina, dok su sljedovi diskretna reprezentacija signala na krivuljama. Osim ova dva primjera fizikalne se simulacije često odvijaju ma mrežama (*eng.mesh*), koje nemaju pravilu strukturu poput rešetki zbog toga osim nekog euklidskog prostora, mogu biti diskretna reprezentacija mnogostrukosti. Budući da se dinamika raznih fizikalnih sustava odvija često na takvim kompleksnim geometrijama, potrebno je poznavati kako ispravno baratati s grafovima i kako učiti GNN-ove nad njima. Također, moramo pripaziti da model poštuje inherentnu kauzalnost, lokalnost i simetrije grafa. Kauzalnost i lokalnost ćemo obraditi u potpoglavlju 5.2 dok ugradnju simetrija obrađujemo u zasebnom poglavlju 6.

Ponekad želimo naglasiti da postoje više različitih vrsta veza između vrhova grafa stoga skup bridova dijelimo na više različitih skupova od kojih svaki odgovara pojedinoj vrsti interakcije. Ovakva struktura naziva se multigaf. Muligrafovi se često koriste u modeliranju signala na mnogostrukostima koje su uronjene u neki ambijenti prostor. Multigraf je uređen par (V, E_M), gdje je V i dalje skup vrhova, a $E_M = \{E_i \subseteq V \times V \,|\, i = 1...n\}$ skup različitih skupova bridova, dakle neki vrhovi mogu imati višestruke veze. Na primjer, modeliramo li evoluciju nekakvog fizikalnog polja na površini oblika zeca poput one na slici 5.1, prvo diskretiziramo mnogostrukost konstrukcijom mreže, a zatim izvrjednjujemo polje na vrhovima mreže. Dinamika polja u danom vrhu ovisit će o vrijednosti tog polja u susjednim vrhovima.

Susjedstvo vrha v_i je skup $\mathcal{N}_i = \{v_j \mid (v_i, v_j) \in E \text{ ili } (v_j, v_i) \in E\}$. Povezanost grafa, a tako i susjedstvo gradimo s obzirom na nekakvu mjeru udaljenosti. U slučaju mreže imat ćemo dvije vrste susjeda s obzirom na dvije metrike udaljenosti. Susjedi prve vrste su susjedi bliski po metrici induciranoj na mnogostrukosti koju aproksimira mreža. Jednostavnije rečeno vrhovi su povezani, ako su bliski u prostoru mreže. Druga vrsta susjeda su oni koji su bliski s obzirom na udaljenost u ambijentnom prostoru u kojem je uronjena mreža. Na slici 5.1 zec dodiruje prednje šape u prostoru mreže točke kontakta na svakoj od šapa su daleko, a interakcija između njih očito postoji. S druge strane vrhovi mreže nisu uniformno distribuirani unutar mreže, stoga ako koristimo samo euklidsku udaljenost za konstrukciju vrhova, postoji rizik da ćemo propustiti interakciju vrhova mreže u područjima gdje je mreža rjeđa.

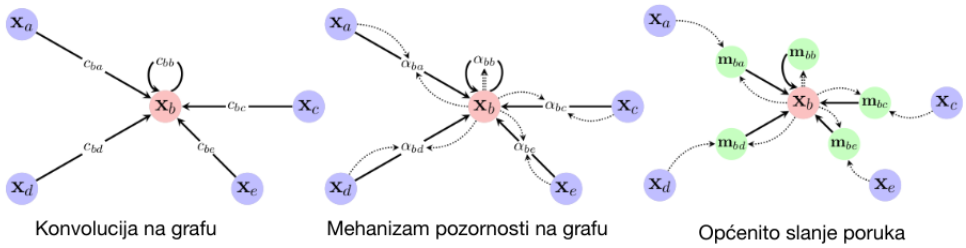


Slika 5.1: Na lijevoj slici prikazana je površina *stanfordskog* zeca kao primjer mnogostrukosti uronjene u $\mathbb{R}^3$. Na desnoj slici je prikazana mreža istog zeca koja je diskretna aproksimacija površine zeca. Iz mreže konstruiramo multigraf generirajući dvije vrste vrhova. Prva vrsta vrhova opisuje lokalne interakcije na mreži, dok druga lokalne interakcije u prostoru u kojem je uronjena mreža. Lijeva slika preuzeta iz [28], a desna iz [29].

Zaključak je da tretman mreža kao grafova zahtjeva korištenje multigrafova tipa (V, E_M, E_e), gdje su V vrhovi, E_M bridovi inducirani bliskošću na mreži, a E_e bridovi inducirani bliskošću u $\mathbb{R}^3$ u kojem je mreža uronjena.

5.2 Neuronske mreže nad grafovima i mehanizam slanja poruka

Neuronske mreže nad grafovima (GNN) su vrsta DNN-ova koja je arhitekturom prilagođena podatcima na grafovima. GNN-ovi uzimaju u obzir strukturu grafa (V, E), prostor značajki vrhova $\mathcal{X}$ i po potrebi prostor značajki bridova $\mathcal{E}$. GNN-ove možemo koristiti u problemima u kojima moramo modelirati neka svojstva vrhova $\mathcal{Y}$, svojstva bridova $\mathcal{Y}_E$ ili pak globalna svojstva grafova $\mathcal{G}$.



Slika 5.2: Primjer najčešće korištenih funkcija slanja poruka, s lijeva na desno: konvolucija, mehanizam pozornosti i općenito slanje poruka. Operacije postaju sve apstraktnije i većeg aproksimacijskog kapaciteta, no tako i veće računalne složenosti. Preuzeto iz [26]

Kao i svaki duboki model i GNN-ovi se sastoje od nekoliko slojeva koje u ovom slučaju zovemo *slanje poruka*. Slanje poruka se odvija u tri koraka: konstrukcija poruka, agregacija poruka i ažuriranje značajki vrhova ili bridova. Za i-ti vrh sa značajkama $\mathbf{x}_i$ i susjedstvom $\mathcal{N}_i$ koraci slanja poruka su

1. Konstrukcija poruka: $m_{ij} = \psi(\mathbf{x}_i, \mathbf{x}_j, e_{ij})$, $e_{ij} \in \mathcal{E}$, $\forall j \in \mathcal{N}_i$
2. Agregacija: $M_i = \bigoplus_{j \in \mathcal{N}_i} m_{ij}$
3. Ažuriranje značajki i-tog vrha: $\mathbf{h}_i = \phi\big(\mathbf{x}_i, M_i\big)$

Sve zajedno

$$\mathbf{h}_i = \phi\Big(\mathbf{x}_i, \bigoplus_{i \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j, e_{ij})\Big) \tag{5.1}$$

U izrazu 5.1 funkcije ϕ i ψ su često varijante potpuno povezanih neuronskih mreža te posjeduju skup parametara koji se mogu prilagoditi podatcima. Simbol $\bigoplus$ označava bilo kakvu funkciju agregacije recimo: sumu, srednju vrijednost, maksimum, minimum itd. Bitno svojstvo agregacije $\bigoplus$ jest invarijantnost na permutaciju indeksa vrhova. Kao što smo spomenuli u odjeljku 5.1, kada imamo skup značajki $\mathcal{X}$, skup vrhova postaje skup indeksa vrhova V. Jasno je da bilo kakvo fizikalno svojstvo grafa ne smije ovisiti o proizvoljnom indeksiranju vrhova grafa, stoga i funkcija slanja poruka u sebi mora imati ugrađenu invarijantnost na permutaciju indeksa. U izrazu 5.1 ta invarijantnost se ugrađuje u model izborom agregacije poruka $\bigoplus$. Ako $\bigoplus$ eksplicitno ne ovisi o indeksima, onda će zadovoljiti permutacijsku invarijantnost. Svi spomenuti primjeri agregacijskih funkcija zadovoljavaju permutacijsku invarijantnost.

Osim permutacijske invarijantnosti, operacija slanja poruka po definiciji poštuje lokalnost jer je se obavlja na svakom vrhu i njegovom susjedstvu. Naime struktura

veza unutar grafa definira lokalnu povezanost vrhova grafa, tako da dani vrh grafa može primati poruke samo od svojih neposrednih susjeda. Ovo može biti problem u situacijama u kojima je cilj modelirati globalna svojstva grafova, međutim nizanjem više slojeva slanja poruka vrh posredno prima informaciju i od daljih vrhova. U slučaju jednog sloja vrh prima poruke od prvih susjeda, u slučaju dva sloja prima poruke od susjeda svojih susjeda itd. Ovisno o pretpostavkama o svojstvima koje modeliramo prilagodit ćemo broj slanja poruka. Za sustave sa izrazito velikim korelacijskim duljinama broj slanja poruka može predstavljati usko grlo u numeričkim računima.

5.3 Mehanizam pozornosti na grafu

Arhitekturu koju primjenjujemo na problem predviđanja molekulskih sila jest O(3)-evkivarijantni transformer nad grafovima. Pitanjem ugradnje simetrija grupe O(3) bavimo se u poglavlju 6., dok ćemo ovdje ukratko objasniti mehanizam pozornosti koji je ključan za izgradnju transformera na grafovima.

Mehanizam pozornosti vrsta je slanja poruka oblika

$$m_{ij} = \alpha(x_i, x_j)x_j \tag{5.2}$$

Pri čemu je α funkcija s prilagodljivim parametrima. U originalnom članku u kojem se spominje ovakvo slanje poruka [30] na vektore x_i i x_j primjenjuju se linearne transformacije W_Q i W_K te se između dobivenih vektora traži skalarni produkt, koji mjeri neku vrstu sličnosti između vrhova i i j.

$$s_{ij} = \langle W_Q x_i, W_K x_j \rangle \tag{5.3}$$

Osim običnih linearnih transformacija W_Q i W_K, mogu se koristiti i kompliciranije operacije čime općenito dobivamo vektore upita (*eng.*query) i vektore ključa (*eng.*key). U ovoj najjednostavnijoj inačici vektori ključa i upita su redom $W_K x_j$, $W_Q x_i$, u ekvivarijantnoj inačici transformatora nad grafovima ti vektori se dobivaju na kompliciraniji način.

Koeficijenti sličnosti se normiraju unutar svakog susjedstva, zatim ih se koristi kao

težine u agregaciji značajki susjeda

$$\alpha(x_i, x_j) = \frac{e^{s_{ij}}}{\sum_{j \in \mathcal{N}_i} e^{s_{ij}}} \tag{5.4}$$

Ažurirane značajke vrhova se dobivaju putem

$$h_i = x_i + \sum_{j \in \mathcal{N}_j} \alpha_{ij} x_j \tag{5.5}$$

Ovakvo slanje poruka se zove mehanizam pozornost jer izgleda kao da model uči na koje susjede treba obratiti više *pozornosti*.

6 G-ekvivarijantni duboki modeli

Više puta smo kroz rad naglasili kako ugradnja spoznaja o simetrijama problema čini model boljim u generalizaciji te efikasnijim s obzirom na potrebnu količinu podataka za njegov trening. U ovom odjeljku fokusiramo se na matematički formalizam ugradnji simetrija. Model gradimo koristeći isključivo evivarijantne funkcije.

***Definicija* 6.1 (Ekvivarijantnost)** *Neka je G grupa simetrija te neka su V i W apstraktni vektorski prostori sa reprezentacijama ρ_V i ρ_W. Za funkciju $f : V \longrightarrow W$ kažemo da je G-ekvivarijantna ako zadovoljava*

$$\rho_W(g)f(x) = f(\rho_V(g)x) \quad \forall\, x \in V\,, \forall g \in G$$

Iz definicije 6.1 vidi se da je invarijantnost samo poseban slučaj ekvivarijantnosti u kojem $\rho_W(g) = I_W \;\forall\, g \in G$. Invarijantne veličine s obzirom na neku grupu zovemo skalarima.

Dobra svojstva ekvivarijantnih modela dolaze pod cijenu da je da je funkcionalni oblik svake komponente modela sada uvelike ograničen. Iz tog razloga obične potpuno povezane neuronske mreže iz poglavlja 4. ne možemo koristiti želimo li ekvivarijantne modele. Iako postoje mnogi načini za postizanje ekvivaijantnosti, u ovome se radu fokusiramo na korištenje ireducibilnih reprezentacija i tenzorskih produkata.

6.1 Slanje poruka putem tenzorskog produkta

Recimo da na elemente prostora značajki $\mathcal{X}$ mogu djelovati elementi neke grupe G te pretpostavimo da su $x \in \mathcal{X}$ dani u bazi ireducibilnih reprezentacija. Cilj je konstruirati nelinearnu funkciju s nekim brojem parametara koji se mogu prilagoditi na podatke, koja će čuvati grupnu strukturu. Jedan način da to obavimo jest da definiramo skalarni produkt $\langle \cdot, \cdot \rangle$ na prostoru značajki te slanje poruka definiramo kao i u [31] putem

$$h_i = \sum_n \sum_{j \in \mathcal{N}_i} f_n(\langle x_i, x_j \rangle) x_j \tag{6.1}$$

U ovom slučaju zaista dobivamo ekvivarijantno slanje poruka, međutim izlaz ovakvog sloja ostaje tenzor istog ranga kao i tenzori na ulazu. Ponekad su tenzori višeg ranga potrebni za opis kompliciranih interakcija sustava. Za dobivanje tenzora višeg ranga,

poruku možemo definirati putem tenzorskog produkta značajki vrha sa značajkama njegovih susjeda. Primjerice neka su značajke vrhovi tenzori prvog reda $x_{i,\mu}$[3]. Tada je tenzorski produkt značajku dvaju susjeda

$$(x_i \otimes x_j)_{\mu\nu} = x_{i,\mu} x_{j,\nu} \tag{6.2}$$

Dobiveni tenzor jest zaista višeg ranga, konkretno nalazi se u prostoru direktnog produkta ulaznog prostora $\mathcal{X} \otimes \mathcal{X}$. Analogno vrijedi za tenzore proizvoljnog reda

$$(A \otimes B)_{\mu_1\mu_2\mu_3\ldots\nu_1\nu_2\nu_3\ldots} = A_{\mu_1\mu_2\mu_3\ldots} B_{\nu_1\nu_2\nu_3\ldots} \tag{6.3}$$

Iako korištenje tenzora višeg reda poboljšava ekspresivnost njime ne gradimo univerzalne aproksimatore, naime tenzorski produkt stvara samo polinome značajki stoga preostaje primijeniti nelinearnost koja nije polinomijalna i čuva grupnu strukturu. Problem nelinearnosti objasnit ćemo kasnije u potpoglavlju 6.1.2, no prvi problem na kojeg se moramo osvrnuti jest da tenzorski produkt definiran jednadžbom 6.3 nije dekomponiran na ireducibilne reprezentacije. Odnosno operaciju koju trebamo je tenzorski produkt te Clebsch-Gordanova dekompozicija. Osim što Clebsch-Gordanovom dekompozicijom održavamo ireducibilnost značajki, njome direktno biramo koju vrstu tenzora na izlazu želimo. Budući da je grupa O(3) ključna za problem modeliranja svojstava molekula, ostatak poglavlja fokusira se na O(3) i translacijsku ekvivarijantnonst.

6.1.1 Tenzorski produkt: grupe SO(3) i O(3)

Za grupu SO(3) vektorski prostori ireducibilnih reprezentacija su identificirani s indeksom l te su dimenzije $2l + 1$. Konkretno za $l = 0$ dobivamo skalare, za $l = 1$ vektore, a $l \geq 2$ općenito tenzore višeg reda. Rezultat tenzorskog produkta dviju ireducibilnih reprezentacija h_{l_1} i h_{l_2} dekomponira se na tenzore reda l gdje je $|l_1 - l_2| \leq l \leq l_1 + l_2$. Komponente tenzora za dani l dobiju se Clebsch-Gordanovom dekompozicijom

$$h_{lm} = \sum_{m_1=-l_1}^{l_1} \sum_{m_2=-l_2}^{l_2} C^{lm}_{l_1 m_1 l_2 m_2} h_{l_1 m_1} h_{l_2 m_2} \tag{6.4}$$

[3]Indeks i označava indeks vrha dok μ je indeks komponente tenzora x_i u nekoj bazi

Uzmemo li u obzir transformacijska svojstva sustava s obzirom na zrcaljenje osi koordinatnog sustava, tada se radi o grupi O(3). Prostori ireducibilnih reprezentacija su u tom slučaju identificirani s dva indeksa (l, p). Indeks l isti je kao i kod grupe SO(3), dok indeks $p = \pm 1$ označava paritetnu transformaciju reprezetacije na koju se odnosi. Tenzorski produkt i njegova Clebsch-Gordanova dekompozcija na grupi O(3) identična onoj iz izraza 6.4, osim što je paritet rezultata umnožak pariteta inicijalnih reprezentacija.

6.1.2 Ugradnja nelinearnosti

Jednom kada prepoznamo strukturu ireducibilnih reprezentacija vektora značajki vrhova, više ne možemo na svaku komponentu primijeniti aktivacijsku funkciju, kao što bismo radili u odjeljku 4.2.1 jer bi to narušilo ekvivarijanost. Stoga moramo pronaći drugi način za ugradnju nelinearnosti koja nam je potrebna za izgradnju univerzalnih aproksimatora. Niz radova postiže nelinearnost oslanjajući se na izgradnju skalara, primjenjujući nelinearnost na njih te s dobivenim skalarnim množe ostale ireducibilne reprezentacije [6, 31, 32]. Naime skalarne veličine su jedine na koje se može primijeniti proizvoljna nelinearnost bez narušavanja ekvivarijantnosti.

Na primjer, u slučaju grupe O(3) uzmimo da se vektorski prostor značajki može dekomponirati na direktnu sumu $\mathcal{X} = 16 \times \mathbf{0}_e \oplus 4 \times \mathbf{1}_o \oplus 8 \times \mathbf{2}_e$. Otisnute veličine $\mathbf{0}$, $\mathbf{1}$, $\mathbf{2}$ redom označavaju prostor skalara, vektora i tenzora reda 2, tj. odnose se na broj l. Subskripti (o ili e) na otisnutim veličinama odnose se na paritet reprezentacije pri čemu o označava neparni paritet -1, a e parni 1. Brojke ispred otisnutih veličina označavaju multiplicitet danog prostora, dakle u navedenom primjeru prostor $\mathcal{X}$ se dekomponira na 16 skalara, 4 vektora te 8 tenzora reda 2. Na 16 skalara bez problema primjenjuje se nelinearnost, dok se od za svaki od preostalih prostora grade skalari, kojih će u ovom primjeru biti $4 + 8 = 12$. Na svaki od dobivenih skalara primjenjuje se nelinearnost te se taj transformirani skalar množi sa vektorima pripadajućeg prostora. Recimo, za svaki od 4 vektora bismo pronašli normu, tu normu predali na ulaz potpuno povezanoj neuronskoj mreži ili samo aktivacijskoj funkciji te s njome pomnožili svaki vektor. Također, moguće je i spremati 12 dodatnih skalara koji bi služili za nelinearnost i množenje neskalarnih tenzora.

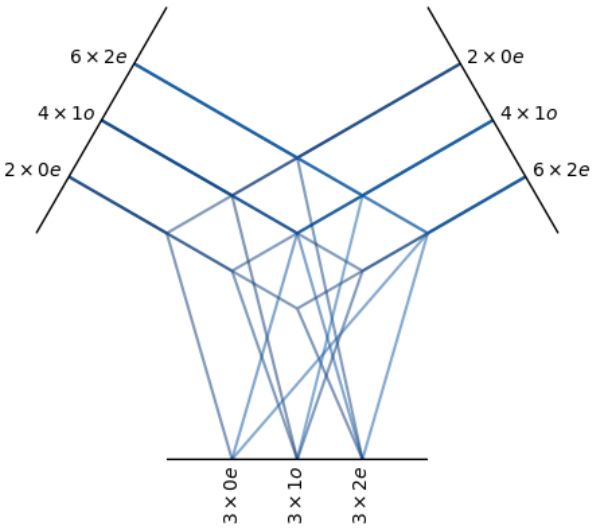
Općenito nelinearnost se ugrađuje putem izraza

$$\Phi(\{\mathbf{x}\},\{\mathbf{g}\},\{\mathbf{y}\}) = \left(\bigoplus_i \phi_i(x_i)\right) \oplus \left(\bigoplus_j \phi_j(g_j)y_j\right) \tag{6.5}$$

Simbol $\{\mathbf{x}\}$ predstavlja skup skalara koje čuvamo i nakon nelinearnosti, $\{\mathbf{y}\}$ je skup ne skalarnih tenzora, a $\{\mathbf{g}\}$ skup skalara koji se propuštaju kroz nelinearnost i množe s neskalarnim tenzorima. Ovakva nelinearnost često se naziva propusnica (*eng.*gate).

6.1.3 Potpuno povezani tenzorski produkt

U prošlom potpoglavlju 6.1.2, naglasili smo kako vektori značajki vrhova mogu biti dekomponirani na više vektora ireducibilnih reprezentacija od kojih svaki posljeduje neki multiplicitet. U tom slučaju tenzorski produkt poput onog opisanog u 6.1.1 nije jedinstven, odnosno imamo više načina (*puteva*) kojim možemo dobiti vektor iste reprezentacije h_{lmp}. Tenzorski produkt je bilinearna operacija, stoga na *ulazu* prima dvije veličine, odnosno možemo unaprijed definirati kakvu strukturu reprezentacija očekujemo na jednom ulazu, kakvu na drugom i kakvu na izlazu. Tenzorski produkt



Slika 6.1: Vizualizacija *potpuno povezanog tenzorskog produkta*, na ulazima tenzorskog produkta očekuju dva identična vektorska prostora koji su dekomponirani na $2\times\mathbf{0}_e\oplus 4\times\mathbf{1}_o\oplus 6\times\mathbf{2}_e$, dok na izlazu konstruiramo vektorski prostor $3\times\mathbf{0}_e\oplus 3\times\mathbf{1}_o\oplus 3\times\mathbf{2}_e$. Svijetlo plavom bojom naznačeni su putevi tenzorskog produkta, dakle sve moguće kombinacije reprezentacija na ulazu koje daju neku reprezentaciju na izlazu. U ovom primjeru ukupno ima 588 puteva.

možemo promatrati kao

$$Tp : V_{in1} \times V_{in2} \longrightarrow V_{out} \tag{6.6}$$

Svaki od navedenih vektorskih prostora direktna je suma prostora ireducibilnih reprezentacija $V = \bigoplus_{l,p} \mu_{l,p} V_{l,p}$, tako da se svaki prostor reprezentacija izlaza može povezati s određenim brojem parova prostora reprezentacija ulaza. Svaku takvu identifikaciju para dvaju prostora ulaza s prostorom izlaza nazivamo *putem*. U tenzorski produkt opisan 6.1.1 mogu se ugraditi parametri koji su prilagodljivi podatcima i to tako da svakom putu pridružimo prilagodljivu težinu. Ovime dobijemo potpuno povezani tenzorski produkt opisan u [32].

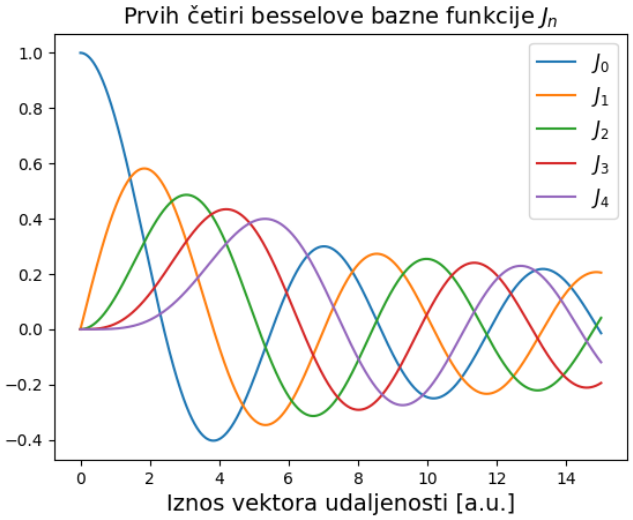
$$h_{lmp} = \sum_{k}^{N_{putevi}} W_k \sum_{m_1=-l_1}^{l_1} \sum_{m_2=-l_2}^{l_2} C^{lm}_{l_1 m_1 l_2 m_2} h_{l_1(k) m_1(k) p_1} h_{l_2(k) m_2(k) p_2} \tag{6.7}$$

U izrazu 6.7 naglašeno je kako komponente ulaznih reprezentacija ovise o indeksu puta k. Potpuno povezani tenzorski produkt i nelinearnost korištenjem propusnice čine temelj ekvivarijantnih modela koje koristimo u ovom radu i šire [4, 33, 34]. Dodatni detalji o teoriji i računalnoj implementaciji tenzorskog produkta ove vrste mogu se pronaći u [32].

6.2 Ugradnja translacijske simetrije

Budući da slanje poruka modelira interakciju između dijelova kompleksnog sustava (atoma, dijelova materijala, čestica itd.), ta interakcija ne smije ovisiti o apsolutnoj poziciji i/ili brzini vrha u nekom koordinatnom sustavu. Stoga u odsustvu vanjskih pobuda koje bi narušile translacijsku simetriju, ne koristimo apsolutne pozicije i apsolutne brzine kao značajke vrhova, već koristimo relativne pozicije i relativne brzine kao značajke bridova grafa. Na primjeru relativnih pozicija, objasnit ćemo kako se ugrađuje invarijantnost na translaciju globalnog koordinatnog sustava, analogan pristup može se koristiti i za tretiranje Galileijevih potisaka. Ugradnja Poincaréove simetrije se detaljnije proučava u [5–7].

Kao što je spomenuto umjesto apsolutnih pozicija vrhova grafa $\mathbf{r}_i$, radije se koristi skup relativnih pozicija $\mathcal{E} = \{\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j \,|\, \forall\, i \in V\,, \forall\, j \in \mathcal{N}_i\}$. Zatim iz skupa relativnih udaljenosti računamo iznos r_{ij} i vektor smjera $\hat{r}_{ij}$ za svaki vektor relativne udaljenost iz $\mathcal{E}$. Tretman vektora smjera, diskutiramo u potpoglavlju 6.3, dok su



Slika 6.2: Prvih četiri Besselovih baznih funkcija prve vrste.

iznosi relativnih pozicija već translacijski i rotacijski invarijantni. Ti iznosi se razviju po nekoj od baznih funkcija, npr. Fourier, Bessel, Gausijan itd. U ovom radu koristili smo za bazne funkcije Besselove funkcije prve vrste. Intuicija iza ovog razvoja jest da se dobije bolja rezolucija iznosa relativnih udaljenosti na većem broju prostornih skala.

6.3 *Sferni harmonici*

Jedinični vektori smjera već su translacijski invarijantne i rotacijski ekvivarijantne veličine, međutim što ako od njih želimo konstruirati tenzore višeg reda? Tenzorski produkt iz izraza 6.1 je definiran za značajke vrhova i nije lako prilagodljiv na značajke bridova a vektori relativne udaljenosti vrhova su očito značajke bridova. Iz tog razloga za konstrukciju tenzora višeg ili nižeg reda možemo koristiti sferne harmonike. Sferni su harmonici članovi vektorskih prostora ireducibilnih reprezentacija grupe SO(3) te su njihova rotacijska svojstva su dobro poznata. Neka je R proizvoljna matrica rotacija kojom djelujemo na vektor smjera $\hat{r}$. Tada se $Y_{lm}(\hat{(}r))$ transformira putem Wignerovih-D matrica.

$$\hat{r}' = R\hat{r} \implies Y_{lm}(\hat{r}') = \sum_{m'} D^{l}_{mm'}(R) Y_{lm'}(\hat{r}) \tag{6.8}$$

Bitno je za naglasiti da vrijednosti sfernih harmonika koje djeluju na zarotirane jedinične vektore *ostaju* u svom prostoru reprezentacija, odnosno ne miješaju se s har-

monicima različitih l. Tako da vektor $\hat{r}$ možemo preslikati u vektorski prostor proizvoljno visokih sfernih harmonika. Prednost toga jest povećanje sferne rezolucije, pod cijenu povećanja dimenzionalnosti vektora značajki bridova. Pokazuje se kako je dovoljno razviti po svim sfernim harmonicima s $l \leq 2$ [4, 34, 35].

7 O(3)-transformer

U prethodnim smo poglavljima objasnili sve potrebne komponente za izgradnju modela te iznijeli osnovne informacije o skupu podataka te ciljeve koje pokušavamo ostvariti. U ovom poglavlju sve od navedenog povezujemo u arhitekturu modela, objašnjavamo postupak učenja modela, koje metrike pratimo kako bismo ocijenili kvalitetu modela i konačno rezultate na skupu podataka za provjeru generalizacije (sile na atome molekula paracetamola). Također komentirat ćemo performansu modela s obzirom na promjenu različitih hiperparametara.

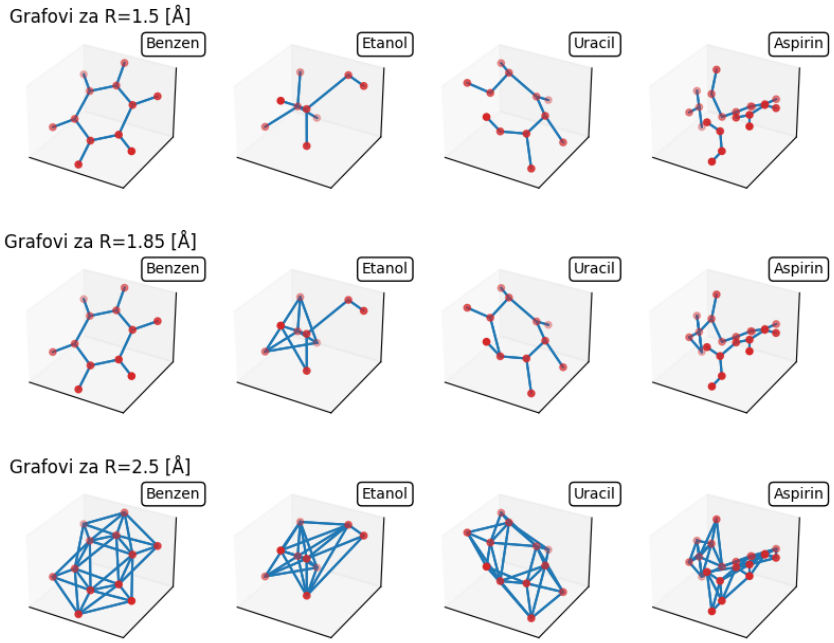
7.1 Izgradnja grafa

Iz skupa podataka MD17 dobivamo samo informaciju o tipovima atoma i njihovim pozicijama, iz ovih grubih podataka potrebno je izgraditi graf, odnosno odrediti povezanost atoma putem skupa bridova E. Susjedstvo pojedinog atoma definiramo kao skup svih ostalih atoma koji se nalaze na udaljenosti manjoj od nekog radijusa R od njega.

Radijus je hiperparametar modela te smo ga kroz razne numeričke eksperimente varirali od $1{,}5\,$Å do $2{,}5\,$Å. Iznos radijusa bliže $1{,}5\,$Å često bi rezultirala ne povezanim grafom, odnosno postojale bi regije koje su potpuno izolirane jedna od druge. Na ovakvim grafovima modeli su često podnaučeni i ne uspijevaju pravilno opisati interakciju atoma. S druge strane, vrijednosti radijusa bliže $2{,}5\,$Å često stvaraju potpuno povezan graf, dakle svaki vrh je susjed svakom drugom vrhu. Rezultat ovoga su prenaučeni modeli, koji slabo generaliziraju na podatke koji se ne nalaze u skupu podataka za učenje. Dobri rezultati dobivaju radijuse u rasponu od $1{,}8$ do $1{,}9$ Å, s najboljom generalizacijskom sposobnošću pri radijusu od $1{,}85\,$Å. Primjeri grafova s različitim radijusima susjedstva mogu se vidjeti na slici 7.1.

7.2 Arhitektura modela

U ovom potpoglavlju opisat ćemo arhitekturu modela. U dodatku A je implementacija čitavog modela u programskom jeziku Python uz paket za izradu modela dubokog učenja Pytorch. U dodatku B nudimo implementacije pojedinih sastavnih cjelina modela također korištenjem Python-a i Pytorch-a. Također čitava računalna imple-

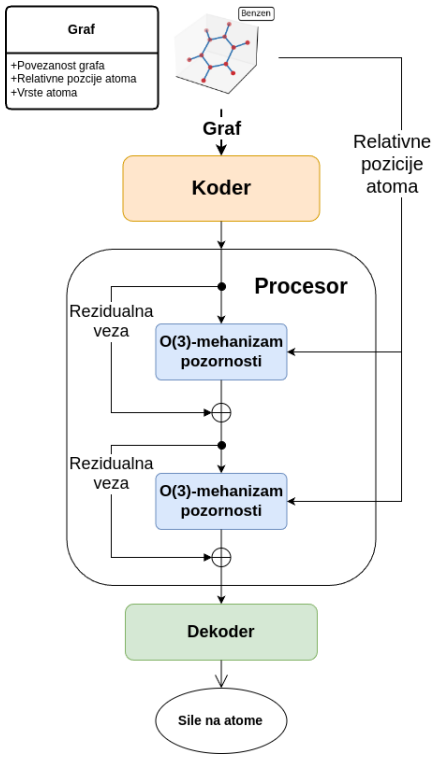


Slika 7.1: Prikaz vrhova i bridova grafa za benzen, etanol, uracil i aspirin. Crvenim točkama označeni su vrhovi grafa na pozicijama atomskih jezgara, dok su plavim linijama označeni bridovi grafa. Simbol R označava radijus unutar kojeg se moraju nalaziti svi susjedi danog atoma. Korištenje manjih radijusa stvara grafove s nepovezanim regijama, dok veći radijusi stvaraju potpuno povezane grafove.

mentacija projekta: tretiranje podataka, izgradnja modela, trening modela i testiranje modela objavljena je na javnom repozitoriju `https://github.com/mirxonius/complex-sys-gnn`. Globalna struktura modela slična je kao ona u radovima [2, 3], odnosno sastoji se od tri veće cjeline: kodera, procesora i dekodera. Dijagramski prikaz modela dan je na slici 7.2.

7.2.1 Koder

Zadatak kodera jest iz podataka ulaznog grafa stvoriti kompleksnije vektore značajki vrhova koje ovise o relativnim pozicijama susjednih atoma te vrsti susjednih atoma. Za tretman vektora udaljenosti atoma koristimo postupak opisanim u potpoglavljima 6.3 i 6.2. Dakle iznose vektora udaljenosti razvijamo po Besselovim baznim funkcijama te ih pružamo potpuno povezanoj neuronskoj mreži. Jedinične vektore udaljenosti razvijamo po sfernim harmonicima s l u rasponu: $0 \leq l \leq L_{max}$. Svakoj vrsti



Slika 7.2: Na dijagramu je prikazana struktura O(3)-transformera, koji se sastoji od kodera, procesora i dekodera. Strelice prikazuju tok podataka. Svakom bloku O(3) mehanizma pozornosti ponovno se pruži informacija o relativnim pozcijama atoma. Između dva susjedna bloka O(3) mehanizma pozornosti postoji rezidualna veza.

atoma pridružujemo M-dimenzionalni vektor koji se može prilagoditi podatcima, na primjer svi atomi ugljika dobit će isti vektor, koji se pak razlikuje od vektora kojeg smo pridružili atomima vodika. Kako smo spomenuli u potpoglavlju 6.3 vektori međusobne udaljenosti atoma, a tako i njihov razvoj po sfernih harmonicima spadaju u značajke bridova, stoga da dobijemo značajke pridružene pojedinom vrhu možemo ih jednostavno agregirati po svakom susjedstvu. Kodiranje značajki vrhova može se sročiti formulom

$$h_{i,nlm} = \sum_{j \in \mathcal{N}_i} R_n(r_{ij}) Y_{lm}(\hat{r}_{ij}) W_{z_j} \tag{7.1}$$

Oznaka $R_n(r_{ij})$ odnosi se na kompoziciju razvoja iznosa vektora udaljenosti po Besselovim funkcija i potpuno povezanu neuronsku mrežu, index n odgovara multiplicitetu ireducibilne reprezentacije s brojem l. Vektor W_{z_j} pridružen je vrhu j s atomskim brojem z_j, dok su Y_{lm} sferni harmonici.

7.2.2 Procesor

Procesor se sastoji od nekoliko slojeva slanja poruka, konkretno koristili smo ekvivarijantni mehanizam pozornosti, koji je analogan onome iz potpoglavlja 5.3. Bitna je razlika što koristimo isključivo operacije koje čuvaju O(3)-ekvivarijantnost. Pokazuje se u [4, 33] kako modeli funkcioniraju bolje, ako se u svakom koraku procesora ponovno ugradi informacija o relativnim pozicijama vrhova. To se radi tako da na vektore značajki i razvoj relativnih pozicija po sfernim harmonicima primijenimo potpuno povezani tenzorski produkt. Za težine puteva tenzorskog produkta koristimo radijalnu neuronsku mrežu R_n poput one u potpoglavlju 7.1, mreža ima jednako izlaza koliko postoji puteva u tenzorskom produktu.

$$\tilde{h}_{ij,nlm} = \sum_{k}^{N_{putevi}} R_{n(k)}(r_{ij}) \sum_{m_1 m_2} C^{lm}_{l_1 m_1 l_2 m_2} h_{j,nl_1 m_1} Y(\hat{r}_{ij})_{l_2 m_2} \tag{7.2}$$

Slična se operacija provodi za konstrukciju vektora ključeva $k_{ij,nlm}$, dok se za vektore upita q_i, nlm primjenjuje linearna transformacija W^Q koja ne miješa prostore različitih reprezentacija (W^Q je blok dijagonalna matrica).

$$k_{ij,nlm} = \sum_{k}^{N_{putevi}} R^K_{n(k)}(r_{ij}) \sum_{m_1 m_2} C^{lm}_{l_1 m_1 l_2 m_2} h_{j,nl_1 m_1} Y_{l_2 m_2}(\hat{r}_{ij}) \tag{7.3}$$

$$q_{i,nlm} = W^Q h_{i,nlm} \tag{7.4}$$

Mjere sličnosti s_{ij} su skalarne veličine te kako moramo poštivati O(3)-ekvivarijantnost običan skalarni produkt između kartezijevih vektora neće biti prikladan već ih trebamo vezati koristeći tenzorski produkt oblika

$$s_{ij} = \sum_{k}^{N_{putevi}} \sum_{m_1 m_1} C^{00}_{l_1 m_1 l_2 m_2} q_{i,n(k) l_1 m_1} k_{j,n(k) l_2 m_2} \tag{7.5}$$

Normirane mjere sličnosti α_{ij} dobivaju se izrazom 5.4. Označimo li indeksom t redni broj sloja slanja poruka, ažurirane značajke vrhova možemo dobiti izrazom

$$h^{t+1}_{i,nlm} = h^t_{i,nlm} + \sum_{j \in \mathcal{N}_i} \alpha_{ij} \tilde{h}^t_{ij,nlm} \tag{7.6}$$

Ponovimo li postupak N puta dobit ćemo procesor s N slojeva slanja poruka, u nastavku pokazujemo kako broj slojeva ne mora biti velik te da se najbolji rezultati dobivaju za $N = 2$.

7.2.3 Dekoder

Dekoder je linearna transformacija koja ne miješa prostore različitih ireducibilnih reprezentacija, dakle blok dijagonalna matrica poput one u izrazu 7.4. U zadatku predviđanja sila trebaju nam samo prostori s $l = 1$ te $p = -1$ tako da se prostori s $l \neq 1$ ili $p = 1$ ne koriste, a konačni rezultat se dobije usrednjavanjem po svim prostorima gdje su $l = 1$ i $p = -1$.

7.3 Učenje, evaluacija modela i rezultati

7.3.1 Postupak učenja

Postupak učenja dobrim dijelom je opisan u potpoglavlju 4.2.2, ovdje iznosimo tipične postavke provedenih eksperimenata. Optimizaciju smo provodili Adam optimizacijskim algoritmom [21], uz $L2$ regularizaciju s regularizaijskim faktorom $\lambda = 10^{-3}$. Također korišten je eksponencijalni usklađivač stope učenja (*eng.*scheduler) s multiplikativnim faktorom trnjenja stope učenja od $0{,}999$. Stopu učenja smo varirali od $0{,}5$ do $0{,}001$ s najboljim rezultatima i najvećom brzinom konvergencije uz stopu učenja od $0{,}01$. Najbolja veličina minigrupe bila je 256. Broj epoha smo također varirali od 200 do 1000, bitno veći broj epoha od 200 rezultiralo bi prenaučenim modelima. Za funkciju pogreške koristilo se srednje kvadratno odstupanje $\mathcal{L} = \frac{1}{N_b}\sum_{i=1}^{N_b} ||\mathbf{y}_i - \hat{\mathbf{y}}_i||_2^2$, gdje je N_b broj primjera u mini grupi. Treniranje je provedeno na NVIDIA A40 grafičkoj kartici.

7.3.2 Metrike

Za procjenu performanse modela koristi se skup podataka za validaciju koji isto kao skup podataka za trening sadrži molekule benzena, etanola, aspirina i uracila. Nije a priori jasno koje veličine daju najkvalitetniju informaciju u kvaliteti modela. Naivno mogli bismo pratiti samo vrijednost funkcije gubitka, međutim njene vrline proizlaze iz statističkih svojstava maksimizacije izglednosti podataka, dok je njen iznos poprilično neintuitivna mjera kvalitete modela. Jasno je da je manja funkcija gubitka

bolja od veće, no koliko malen kvadratni gubitak daje dobar model? Primjerice, kvadratno odstupanje sile od 3 kcal^2 mol^{-2}Å^{-2} može biti velika ili mala ovisno o stvarnoj vrijednosti sile. Srednje apsolutno odstupanje pati od istog problema. Ovi nedostatci su pogotovo izraženi u situacijama u kojim iznosi regresijske mete $\mathbf{y}$, variraju kroz nekoliko redova veličina. Na slici 3.2 vidi se da je upravo to situacija sa silama na atome molekula, koje variraju kroz svega tri reda veličine.

Kako bismo riješili navedeni problem koristili smo R^2 mjeru koja se dobiva korištenjem izraza

$$R^2 = 1 - \frac{\sum_i (||\mathbf{y}_i - \hat{\mathbf{y}}_i||)^2}{\sum_i (||\mathbf{y}_i - \langle \mathbf{y} \rangle||)^2} \tag{7.7}$$

Brojnik u izrazu 7.7 odgovara ukupnoj kvadratnoj pogrešci, dok je nazivnik proporcionalan varijanci skupa podataka do na faktor $\frac{1}{N}$. Upravo to skaliranje čini ovu mjeru otpornom na veliko variranje podataka. R^2 faktor je bolji što je bliži 1. Tijekom treninga, evaluacije i testiranja pratimo srednje kvadratno odstupanje, srednje apsolutno odstupanje i R^2 metriku.

7.4 Rezultati

Trenirali smo niz O(3)-ekvivarijantnih transformera s različitim brojem slojeva slanja poruka, različitim maksimalnim rangom sferičnih tenzora L_{max}, te različitim ireducibilnim reprezentacijama kodiranih značajki vrhova. U tablici 7.1 navodimo najuspješnije modele i hiperparametre koje smo varirali, dok u tablici 7.2 navodimo rezultate za svaki model.

model	L_{max}	IRREPS	Broj slanja poruka
O(3)-transformer	2	$32 \times \mathbf{0}_e \oplus 32 \times \mathbf{1}_o \oplus 32 \times \mathbf{2}_e$	2
big L	4	$32 \times \mathbf{0}_e \oplus 16 \times \mathbf{1}_o \oplus 8 \times \mathbf{2}_e \oplus 8 \times \mathbf{3}_o$	2
duboki O(3)-transformer	2	$32 \times \mathbf{0}_e \oplus 32 \times \mathbf{1}_o \oplus 32 \times \mathbf{2}_e$	3

Tablica 7.1: Modeli i varirani hiperparametri: L_{max} maksimalni l u razvoju po sfernim harmonicima, IRREPS prostor ireducibilnih reprezentacija kodiranih značajki vrhova te broj slojeva slanja poruka.

Na slikama 7.3, 7.4, 7.5 redom su prikazani prikazani su grafovi projekcija pozicija atoma te stvarnih i predviđenih sila na atome benzena, uracila i paracetamola u $x-y$, $x-z$ i $y-z$ ravnini. Prikazujemo djelovanje modela na atomima benzena jer je on nejjednostavnija molekula u skupu podataka te služi kao osnovna provjera rada

O(3)-transformer	MAE kcal mol^{-1}Å^{-1}	MSE kcal^2 mol^{-2}Å^{-2}	R^2
Validacija	2.54	13.30	0.982
Testiranje	2.55	13.77	0.982
Generalizacija	5.44	65.69	0.923
big L	MAE kcal mol^{-1}Å^{-1}	MSE kcal^2 mol^{-2}Å^{-2}	R^2
Validacija	2.99	18.90	0.975
Testiranje	2.95	18.69	0.975
Generalizacija	5.75	91.94	0.89
duboki O(3)-transformer	MAE kcal mol^{-1}Å^{-1}	MSE kcal^2 mol^{-2}Å^{-2}	R^2
Validacija	2.67	16.70	0.978
Testiranje	2.92	18.14	0.980
Generalizacija	6.00	79.90	0.91

Tablica 7.2: Metrike za tri najbolja modela: Srednja apsolutna greška (MAE), srednja kvadratna greška (MSE) i R^2 mjera. Najbolji je model na svim skupovima podataka je obični O(3)-transformer. Validacija i testiranje su obavljeni su nad molekulama benzena, etanola, uracila i aspirina, dok je generalizacija provjerena na skupu podataka s paracetamolom.

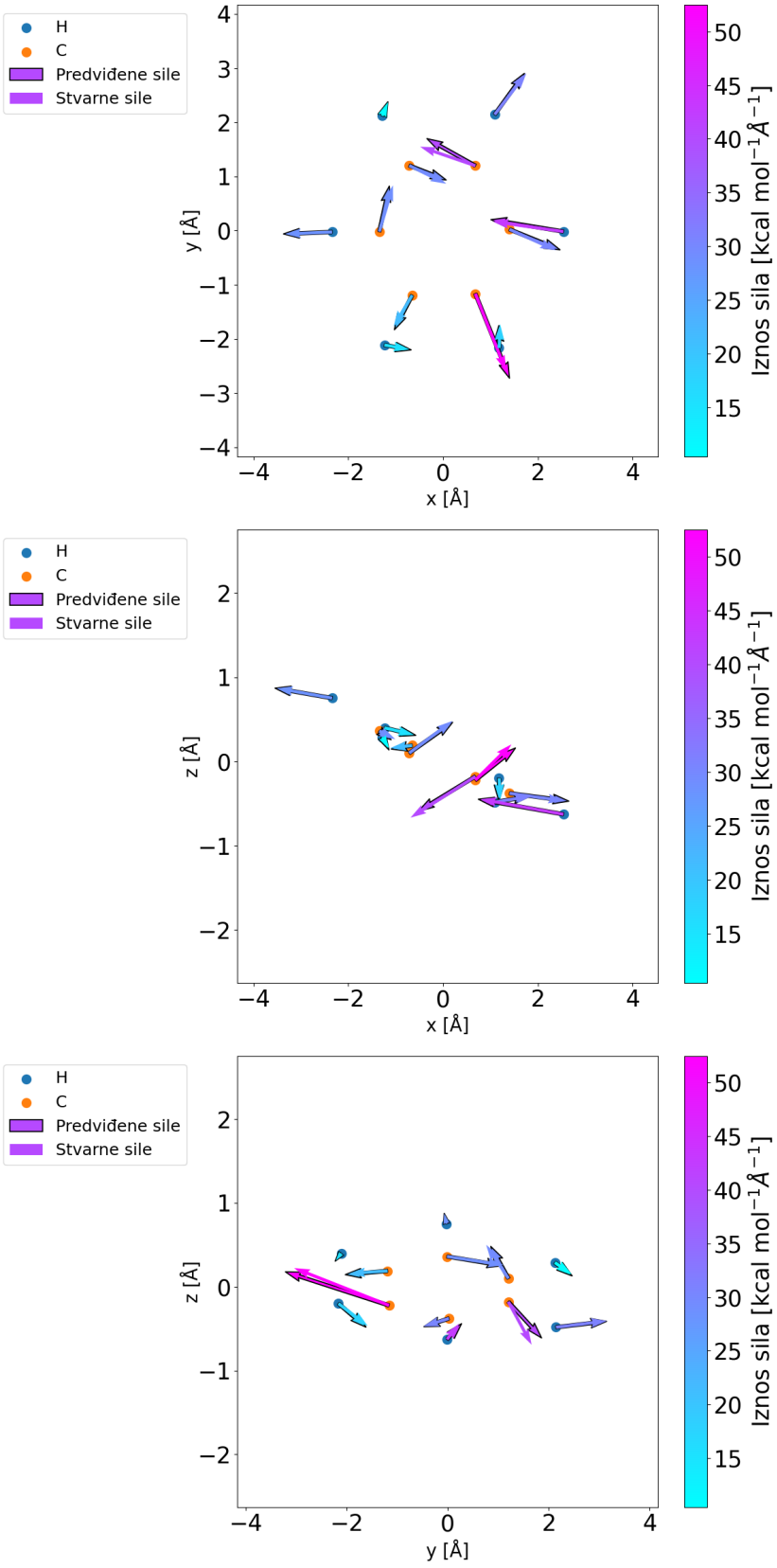
modela. Na slici 7.3 vidi se gotovo savršeno podudaranje stvarnih i predviđenih sila na svaki atom. Molekula uracila nam je zanimljiva jer smatramo da je najkompliciranija molekula na kojoj model treniran. Slika 7.4 prikazuje stvarne i predviđene sile na atome uracila. Predviđanja i stvarne vrijednosti ponovno se dobro podudaraju iako manje dobro nego u slučaju benzena. Čini se kako atomi dušika (koji su najmanje prisutni u skupu podataka za učenje) imaju veća odstupanja između stvarne i previđene sile. Primjer generalizacije modela na atomima molekule paracetamola prikazan je na slici 7.5. Situacija je slična kao na primjeru uracila, dakle većinom dobro slaganje stvarne i predviđene sile s najvećim odstupanjima sila na atomu dušika. Ovaj se problem lako riješi treniranjem novog modela s podatcima koji imaju ravnomjerniju zastupljenost atoma. Iako su odstupanja stvarnih i predviđenih sila na atomu dušika velika, ona su izolirana, odnosno loše predviđanje modela se ne širi na dušikove susjedne atome. Smatramo da to znači da se model lako prilagođava različitim geometrijama molekula. Također smatramo da bi se mogli dobiti još bolji rezultati uvođenjem dodatnog člana u funkciju gubitka koji bi kažnjavao ne samo iznos razlike predviđenih i stvarnih sila, već koji bi kažnjavao kut između tih sila. Naime u slučaju malenih iznosa sila, srednje kvadratno odstupanje bit će malo i za dva suprotna vektora, ako su sličnih duljina te član koji kažnjava kut između predviđene i stvarne sile bi *uočio* ovaj problem.

Pokazalo se kako su najbitniji prostori ireducibilnih reprezentacija koji koriste sve prostore reprezetacija s $l \leq 2$, odnosno korištenje sfernih harmonika ili građenje ireducibilnih reprezentacija s visokim l neće poboljšati performansu modela, također visoki l-ovi usporavaju računanje i značajno povećavaju broj parametara. Vidi se i blago opadanje kvalitete modela za veće brojeve slanja poruka, čini se da su 2 sloja slanja poruka idealna za ovaj zadatak.

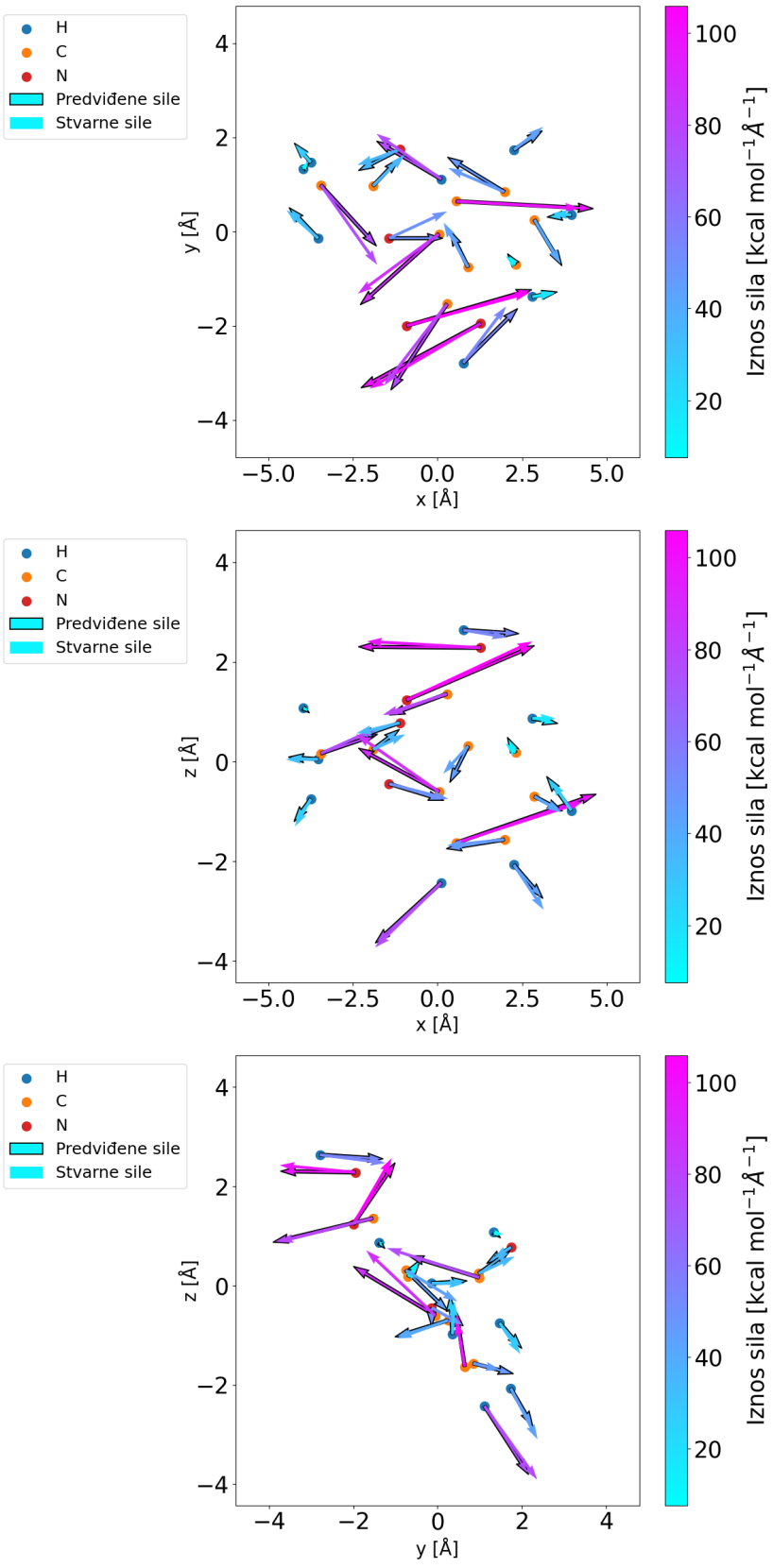
Uspjeh običnog O(3)-transformera motivirao nas je da isprobamo kako radi u režimu malog broja podataka. Od originalnog skupa za trening uzeli smo desetinu primjera od svake molekule, dakle eksperiment smo ponovili sa deset puta manjim skupom podataka. Ponovno smo trenirali O(3)-transformer na manjem skupu podataka, a testiranje i provjeru generalizacije smo proveli na skupu podataka pune veličine. Rezultati ovog eksperimenta dani su u tablici 7.3. Malen pad preciznosti modela, sa značajnim padom broja podataka pripisujemo ugradnji O(3) simetrije u model. Smatramo da je ovakvo modeliranje prilagodljivo na niz različitih fizikalnih sustava te je robusno čak i u situacijama kada podataka nema puno. Također smatramo da ovime dobivamo pouzdan način za preliminarnu analizu kompleksnih sustava koji su do sad bili računalno ne rješivi. Dobivanje pouzdanih inicijalnih rezultata smatramo bitnim korakom u stvaranju intuicije i osnovnog razumijevanja velikog skupa kompleksnih sustava.

O(3)-transformer	MAE kcal mol^{-1}Å^{-1}	MSE kcal^2 mol^{-2}Å^{-2}	R^2
Validacija	6.17	81.70	0.89
Testiranje	6.17	81.65	0.89
Generalizacija	8.36	147.47	0.83

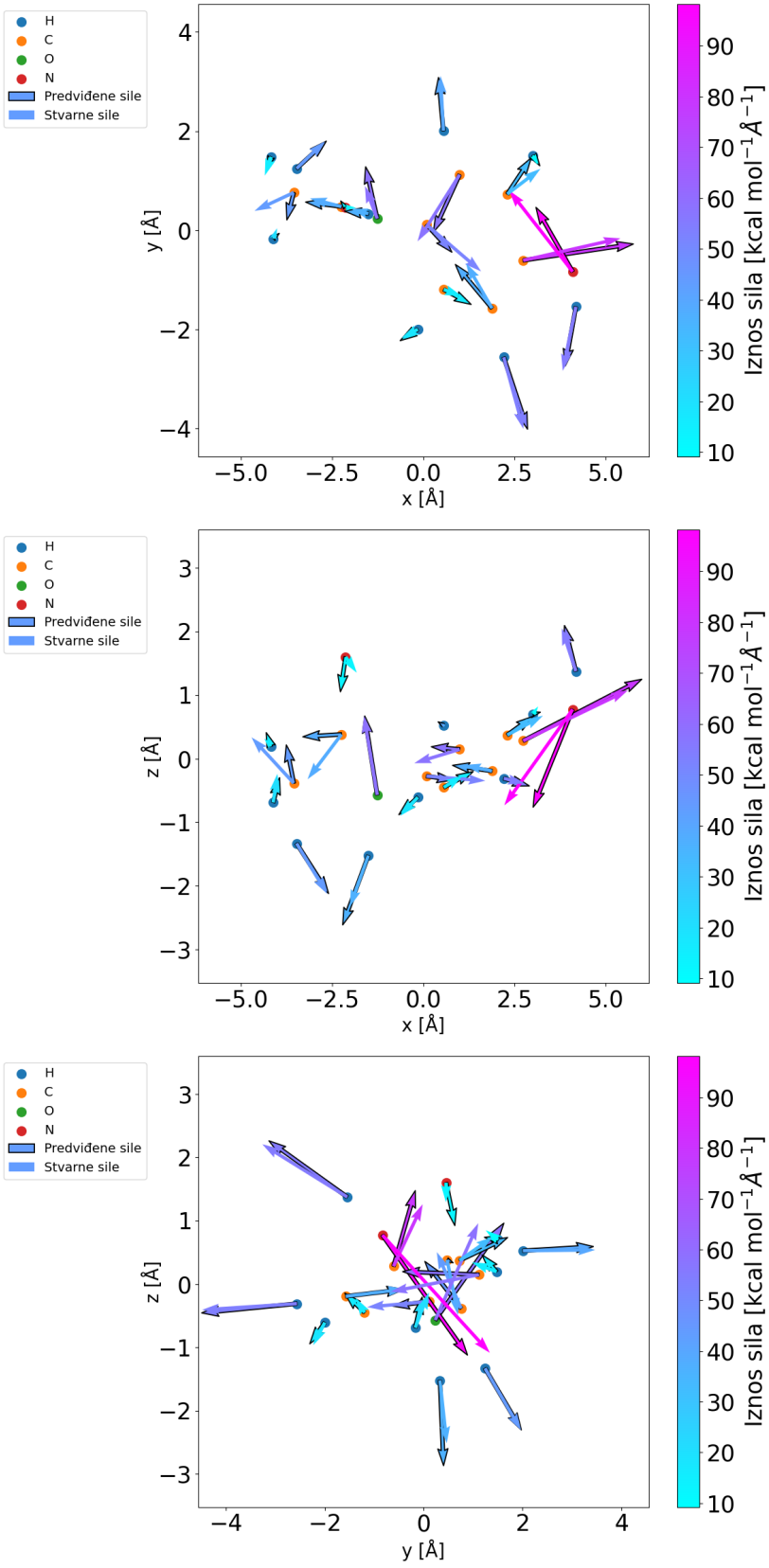
Tablica 7.3: Rezultati O(3)-transformera u režimu malog broja podataka. Testiranje, validacija i provjera generalizacije modela provedena je na istim podatcima kao i za modele iz tablice 7.2. Nije primijećen značajan pad preciznosti, dapače čini se da model s ugrađenom simetrijom ostaje robustan.



Slika 7.3: Prikazane su vrste atoma benzena njihove pozicije te stvarne i predviđene sile na pojedini atom. Stvarne sile su naznačene crnim obrubom oko strelice dok, predviđene sile nemaju taj obrub. Iznos sila naznačen je bojom, gdje toplije boje imaju veći iznos sile. Vrsta atoma također je naznačena bojom. Vidi se dobro podudaranje predviđene i stvarne sile.



Slika 7.4: Prikazana je molekularna konfiguracija uracila u $x-y$, $x-z$, $y-z$ ravninama. Stvarne sile su naznačene crnim obrubom oko strelice dok, predviđene sile nemaju taj obrub. Iznos sila naznačen je bojom, gdje toplije boje imaju veći iznos sile. Vrsta atoma također je naznačena bojom. Vidi se dobro podudaranje predviđene i stvarne sile, unatoč činjenici što molekula sadrži dušik koji je jako rijedak u skupu podataka za trening.



Slika 7.5: Prikazana je molekularna konfiguracija paracetamola u $x-y$, $x-z$, $y-z$ ravninama. Stvarne sile su naznačene crnim obrubom oko strelice dok. Iznos sila naznačen je bojom, gdje toplije boje imaju veći iznos sile. Vrsta atoma također je naznačena bojom. Budući da je paracetamol korišten za provjeru generalizacije modela, podudaranje predviđenih i stvarnih slika je malo lošije nego u slučaju benzena i uracila.

8 Zaključak

U ovom radu smo objasnili temeljne principe modeliranja kompleksnih sustava pomoću neuronskih mreža nad grafovima. Konkretno objasnili smo kako se GNN-ovi prilagođavaju na komplicirane i nepravilne fizikalne domene koje su gotovo uvijek prisutne u dinamici kompleksnih sustava. Naglasili smo važnost ugradnji simetrija fizikalnog sustava u model te detaljno objasnili postupak ugradnje tih simetrija na grupi E(3).

Sve od navedenog smo primijenili na problemu molekularne dinamike, konkretno u predviđanju sila među atomima. Cilj treniranja dubokih modela nad postojećim fizikalnim simulacijama jest *recikliranje* podataka simulacija kako bi se brže i efikasnije mogli dobiti rezultati za slične fizikalne sustave. Osim toga testirali smo sposobnost modela da generalizira na veće i do sad ne viđene sustave. Svi podatci nalazili su se u Md17 skupu podataka. Za trening smo koristili molekule benzena, etanola, uracila i aspirina, dok smo generalizaciju provjeravali na podatcima simulacija dinamike paracetamola. Modeli su imali samo informaciju o pozicijama i tipovima atoma te grafove smo gradili povezivanjem svih vrhova grafa koji se nalaze na međusobnoj udaljenosti manjoj od nekog zadanog radijusa.

Korišteno je više inačica O(3)-transformera te smo pokazali visoku razinu uspješnosti ovog modela u predviđanju sila na atome svih navedenih molekula. Također smo testirali robusnost modela u režimu malog broja podataka, pri čemu se model i dalje pokazao uspješnim. Kao ključ njegovog uspjeha prepoznajemo ugrađenu O(3) simetriju u svim računalnim operacijama modela. Smatramo da se ekvivarijantni modeli mogu koristiti za brzo dobivanje preliminarnih, ali pouzdanih rezultata kada se istražuju novi ili puno veći sustavi.

Budući rad vezan uz ovu temu može se baviti predviđanjem ostalih molekularnih svojstava, simuliranjem elektronskih gustoća molekula ili više sustava molekula. Naglašavamo kako mehanizam slanja poruka temeljen na E(3) grupi simetrija nije ograničen na atomske sustave, već se lako prilagođava na klasične sustave poput mehanike fluida i mehanike deformabilnih tijela.

Dodaci

Dodatak A PyTorch implementacija O(3)-transformera

```
import torch
from e3nn import o3
from torch_geometric.data import Data

from .blocks import O3AttentionLayer, NodeEncoder
from utils.model_utils import MeanOnGraph, IdentityOnGraph

class O3GraphAttentionNetwork(torch.nn.Module):
    def __init__(
        self,
        num_layers: int,
        output_irreps: str | o3.Irreps,
        hidden_irreps: str | o3.Irreps,
        lmax: int,
        num_basis: int,
        aggregate: bool = False,
        max_radius: float = 2.5,
        num_atom_types: int = 4,
        embedding_size: int = 64,
    ):
        self.max_radius = max_radius
        super().__init__()
        if aggregate:
            self.aggregate = MeanOnGraph()
        else:
            self.aggregate = IdentityOnGraph()
        self.embedding_layer = torch.nn.Linear(
            num_atom_types,
            embedding_size
```

```
            )
        self.node_encoder = NodeEncoder(
            embedding_irreps=hidden_irreps,
            num_basis=num_basis,
            max_radius=self.max_radius,
        )
        layers = []
        for i in range(num_layers):
            layers.append(
                O3AttentionLayer(
                    input_irreps=hidden_irreps,
                    key_irreps=hidden_irreps,
                    query_irreps=hidden_irreps,
                    value_irreps=hidden_irreps,
                    lmax=lmax,
                    num_basis=num_basis,
                    max_radius=self.max_radius,
                )
            )
        self.layers = torch.nn.ModuleList(layers)
        self.decoder = o3.Linear(
        irreps_in=hidden_irreps,
        irreps_out=output_irreps
        )

    def forward(self, graph: Data) -> torch.Tensor:
        graph.x = self.node_encoder(graph)
        for layer in self.layers:
            updated_node_features = layer(graph)
            graph.x += updated_node_features
        graph.x = self.decoder(graph.x)
        return self.aggregate(graph.x, batch_index=graph.batch)
```

Dodatak B PyTorch implemetacije kodera i O(3)-evivarijantne pozornosti na grafu

```
import torch
from e3nn import o3, nn, math
from e3nn.util.jit import compile_mode
from torch_scatter import scatter
from torch_geometric.data import Data
from torch_scatter import scatter

from utils.model_utils import softmax_on_graph

class NodeEncoder(torch.nn.Module):
    def __init__(
        self,
        num_atom_types: int = 4,
        atom_embedding_size: int = 64,
        embedding_irreps: str | o3.Irreps = "32x0e + 32x1o + 32x2e",
        lmax: int = 2,
        num_basis: int = 32,
        max_radius: float = 2.0,
    ):
        super().__init__()
        self.max_radius = max_radius
        self.irreps_sph = o3.Irreps.spherical_harmonics(lmax=lmax)
        self.spherical_projection = o3.Linear(
            irreps_in=self.irreps_sph, irreps_out=embedding_irreps
        )
        self.atom_embedding = torch.nn.Linear(
            num_atom_types, self.spherical_projection.irreps_out.dim
        )

        self.num_basis = num_basis
```

```
        self.radial_embedding_net = nn.FullyConnectedNet(
            [self.num_basis,
            32,
            self.spherical_projection.irreps_out.dim],
            act=torch.nn.functional.silu,
        )

    def forward(self, graph: Data) -> Data:
        src, dst = graph.edge_index
        vec = graph.pos[src] - graph.pos[dst]
        vec_len = vec.norm(dim=1)
        vec_sph = o3.spherical_harmonics(
            self.irreps_sph,
            vec,
            normalize=True,
            normalization="component"
        )
        spherical_embedding = self.spherical_projection(vec_sph)
        radial_embedding = math.soft_one_hot_linspace(
            vec_len,
            start=0.0,
            end=self.max_radius,
            number=self.num_basis,
            basis="bessel",
            cutoff=True,
        )
        atom_embedding = self.atom_embedding(graph.z)[src]
        radial_embedding = self.radial_embedding_net(radial_embedding)
        node_emb = radial_embedding
        * spherical_embedding
        * atom_embedding
        return scatter(src=node_emb, index=dst, dim=0)
```

```python
@compile_mode("script")
class O3AttentionLayer(torch.nn.Module):
    def __init__(
        self,
        input_irreps: str | o3.Irreps,
        key_irreps: str | o3.Irreps,
        query_irreps: str | o3.Irreps,
        value_irreps: str | o3.Irreps,
        lmax: int = 2,
        num_basis: int = 32,
        max_radius: float = 2.5,
        num_neighbors: int = 32,
    ) -> None:
        super().__init__()
        self.num_neighbors = num_neighbors
        self.num_basis = num_basis
        self.max_radius = max_radius
        self.irreps_sph = o3.Irreps.spherical_harmonics(lmax=lmax)
        self.tp_value = o3.FullyConnectedTensorProduct(
            irreps_in1=input_irreps,
            irreps_in2=self.irreps_sph,
            irreps_out=value_irreps,
            shared_weights=False,
        )
        self.value_basis_net = nn.FullyConnectedNet(
            [self.num_basis, 32, self.tp_value.weight_numel],
            act=torch.nn.functional.silu,
        )

        self.tp_key = o3.FullyConnectedTensorProduct(
            irreps_in1=input_irreps,
            irreps_in2=self.irreps_sph,
            irreps_out=key_irreps,
```

```
            shared_weights=False,
        )
        self.key_basis_net = nn.FullyConnectedNet(
            [self.num_basis, 32, self.tp_key.weight_numel],
            act=torch.nn.functional.silu,
        )

        # Calculates similarity metric between keys and queries
        self.similarity_tp = o3.FullyConnectedTensorProduct(
            irreps_in1=query_irreps,
            irreps_in2=key_irreps,
            irreps_out="0e"
        )

        self.query_projection = o3.Linear(
            irreps_in=input_irreps, irreps_out=query_irreps
        )

    def forward(self, graph: Data):
        src, dst = graph.edge_index
        vec = graph.pos[src] - graph.pos[dst]
        vec_len = vec.norm(dim=1)

        radial_embedding = math.soft_one_hot_linspace(
            vec_len,
            start=0.0,
            end=self.max_radius,
            number=self.num_basis,
            basis="bessel",
            cutoff=True,
        )

        edge_weight_cutoff = math.soft_unit_step(
```

```
10 * (1 - vec_len / self.max_radius)
)

radial_embedding = radial_embedding.mul(self.num_basis**0.5)
vec_sph = o3.spherical_harmonics(
    self.irreps_sph,
    vec,
    normalize=True,
    normalization="component"
)

query = self.query_projection(graph.x)
key = self.tp_key(
            graph.x[src],
            vec_sph,
            self.key_basis_net(radial_embedding)
            )
values = self.tp_value(
            graph.x[src],
            vec_sph,
            self.value_basis_net(radial_embedding)
)
similarity = self.similarity_tp(query[src], key)
attn_score = softmax_on_graph(
    input=edge_weight_cutoff.unsqueeze(-1) * similarity,
    index=src,
    dim=0,
)

return (
    scatter(
        src=attn_score.relu().sqrt() * values,
        index=dst,
```

```
            dim=0,
            dim_size=len(graph.x),
        )
        / self.num_neighbors
    )
```

Literatura

[1] Uwe C Täuber. *Critical dynamics: a field theory approach to equilibrium and non-equilibrium scaling behavior*. Cambridge University Press, Cambridge, 2014.

[2] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning mesh-based simulation with graph networks, 2021.

[3] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to simulate complex physics with graph networks, 2020.

[4] Ilyes Batatia, Dávid Péter Kovács, Gregor N. C. Simm, Christoph Ortner, and Gábor Csányi. Mace: Higher order equivariant message passing neural networks for fast and accurate force fields, 2023.

[5] Jose M Munoz, Ilyes Batatia, and Christoph Ortner. Boost invariant polynomials for efficient jet tagging. *Machine Learning: Science and Technology*, 3(4):04LT05, December 2022.

[6] Shiqi Gong, Qi Meng, Jue Zhang, Huilin Qu, Congqiao Li, Sitian Qian, Weitao Du, Zhi-Ming Ma, and Tie-Yan Liu. An efficient lorentz equivariant graph neural network for jet tagging. *Journal of High Energy Physics*, 2022(7), July 2022.

[7] Ilyes Batatia, Mario Geiger, Jose Munoz, Tess Smidt, Lior Silberman, and Christoph Ortner. A general framework for equivariant neural networks on reductive lie groups, 2023.

[8] Joshua A. Rackers, Lucas Tecot, Mario Geiger, and Tess E. Smidt. Cracking the quantum scaling limit with machine learned electron densities, 2022.

[9] Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data, 2018.

[10] Simon Batzner, Albert Musaelian, Lixin Sun, Mario Geiger, Jonathan P. Mailoa, Mordechai Kornbluth, Nicola Molinari, Tess E. Smidt, and Boris Kozinsky. E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature Communications*, 13(1), May 2022.

[11] Gennaro Auletta, Mauro Fortunato, and Giorgio Parisi. *Quantum Mechanics*. Cambridge University Press, 2009.

[12] Eberhard Engel and Reiner M. Dreizler. *Density functional theory An advanced course*. Springer, Germany, 2011. CLASSICAL AND QUANTUM MECHANICS, GENERAL PHYSICS.

[13] David S. Sholl and Janice A. Steckel. *Density functional theory : a practical introduction*. Wiley Hoboken, N.J., Hoboken, N.J., 2009.

[14] Ilyes Batatia, Simon Batzner, Dávid Péter Kovács, Albert Musaelian, Gregor N. C. Simm, Ralf Drautz, Christoph Ortner, Boris Kozinsky, and Gábor Csányi. The design space of e(3)-equivariant atom-centered interatomic potentials, 2022.

[15] Md17 dataset. `https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.datasets.MD17.html`. [Pristupljeno: 10-2-2024].

[16] Stefan Chmiela, Alexandre Tkatchenko, Huziel E. Sauceda, Igor Poltavsky, Kristof T. Schütt, and Klaus-Robert Müller. Machine learning of accurate energy-conserving molecular force fields. *Science Advances*, 3(5), May 2017.

[17] Pankaj Mehta, Marin Bukov, Ching-Hao Wang, Alexandre G.R. Day, Clint Richardson, Charles K. Fisher, and David J. Schwab. A high-bias, low-variance introduction to machine learning for physicists. *Physics Reports*, 810:1–124, May 2019.

[18] Neurips 2022 papers. `https://neurips.cc/virtual/2022/papers.html?filter=titles`. Pristupljeno: 12-12-2023.

[19] The universal approximation theorem. `https://www.deep-mind.org/2023/03/26/the-universal-approximation-theorem/#Universal_Approximation_Theorem`. [Pristupljeno: 8-2-2024].

[20] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[21] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[22] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, Dec 1989.

[23] Hrushikesh Narhar Mhaskar. Approximation properties of a multilayered feedforward artificial neural network. *Advances in Computational Mathematics*, 1:61–80, 1993.

[24] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks, 2015.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[26] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges, 2021.

[27] Robin M. Schmidt. Recurrent neural networks (rnns): A gentle introduction and overview, 2019.

[28] The stanford 3d scanning repository. `https://neurips.cc/virtual/2022/papers.html?filter=titles`. Pristupljeno: 15-12-2023.

[29] Alessandro Rossi, Marco Barbiero, Paolo Scremin, and Ruggero Carli. Robust visibility surface determination in object space via plücker coordinates. *Journal of Imaging*, 7:96, 06 2021.

[30] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.

[31] Soledad Villar, David W. Hogg, Kate Storey-Fisher, Weichi Yao, and Ben Blum-Smith. Scalars are universal: Equivariant machine learning, structured like classical physics, 2023.

[32] Mario Geiger and Tess Smidt. e3nn: Euclidean neural networks, 2022.

[33] Fabian B. Fuchs, Daniel E. Worrall, Volker Fischer, and Max Welling. Se(3)-transformers: 3d roto-translation equivariant attention networks, 2020.

[34] James P. Darby, Dávid P. Kovács, Ilyes Batatia, Miguel A. Caro, Gus L. W. Hart, Christoph Ortner, and Gábor Csányi. Tensor-reduced atomic density representations. *Phys. Rev. Lett.*, 131:028001, Jul 2023.

[35] Guillem Simeon and Gianni de Fabritiis. Tensornet: Cartesian tensor representations for efficient learning of molecular potentials, 2023.

[36] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to simulate complex physics with graph networks, 2020.

[37] Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) equivariant graph neural networks, 2022.

[38] Anthony Zee. *Group Theory in a Nutshell for Physicists*. Princeton University Press, USA, 3 2016.

[39] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[40] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2017.

[41] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations, 2021.

[42] Jan E. Gerken, Jimmy Aronsson, Oscar Carlsson, Hampus Linander, Fredrik Ohlsson, Christoffer Petersson, and Daniel Persson. Geometric deep learning and equivariant neural networks, 2021.

[43] H. F. Jones. *Groups, representations and physics*. 1990.