Kauno technologijos universitetas
Matematikos ir gamtos mokslų fakultetas

Naujų vaistinių molekulių generavimas giliaisiais dirbtiniais neuroniniais tinklais

Baigiamasis magistro studijų projektas

Milda Šniokaitė
Projekto autorė

Doc. Dr. Mantas Landauskas
Vadovas

Vytautas Rafanavičius
Konsultantas

Kaunas, 2019

Kauno technologijos universitetas
Matematikos ir gamtos mokslų fakultetas

Naujų vaistinių molekulių generavimas giliaisiais dirbtiniais neuroniniais tinklais

Baigiamasis magistro studijų projektas
Taikomoji matematika (6211AX006)

Milda Šniokaitė
Projekto autorė

Doc. Dr. Mantas Landauskas
Vadovas

Lekt. Dr. Daiva Petkevičiūtė-Gerlach
Recenzentė

Kaunas, 2019

Kauno technologijos universitetas
Matematikos ir gamtos mokslų fakultetas
Milda Šniokaitė

Naujų vaistinių molekulių generavimas giliaisiais dirbtiniais neuroniniais tinklais

Akademinio sąžiningumo deklaracija

Patvirtinu, kad mano, Mildos Šniokaitės, baigiamasis projektas tema „Naujų vaistinių molekulių generavimas giliaisiais dirbtiniais neuroniniais tinklais“ yra parašytas visiškai savarankiškai ir visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

(vardą ir pavardę įrašyti ranka)	(parašas)

Šniokaitė, Milda. Naujų vaistinių molekulių generavimas giliaisiais dirbtiniais neuroniniais tinklais. Magistro studijų baigiamasis projektas, vadovas doc. Dr. Mantas Landauskas; Kauno technologijos universitetas, Matematikos ir gamtos mokslų fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): Matematikos mokslai, Taikomoji matematika.

Reikšminiai žodžiai: Vaistinių molekulių atradimas, variacinis autoenkoderis, generacinis modelis, gilieji dirbtiniai neuroniniai tinklai.

Kaunas, 2019. 59 p.

Santrauka

Naujų vaistinių molekulių generavimas gali sukelti didelę visuomenės ir technologijų pažangą. Molekulių generavimas aktualus sprendžiant įvairių sričių problemas: nuo individualus gydymo iki energijos gamybos ir sandėliavimo. Naujų vaistinių molekulių generavimo svarba ir aktualumas akivaizdus, norint atrasti naujus vaistus būtina ieškoti naujų tam tinkamų molekulių. Giliųjų neuroninių tinklų naudojimas naujų vaistinių molekulių generavimui gali paspartinti naujų vaistų atradimą. Šiai užduočiai įgyvendinti pasitelkiama neuroninių tinklų struktūra – variacinis autoenkoderis: koderio pagalba molekulių reprezentacijos suspaudžiamos į latentinės erdvės vektorius, o vėliau naudojantis dekoderiu dekoduojamos į validžias molekules. Darbo tikslas – Ištirti naujų molekulių generavimo galimybes naudojantis variaciniu autoenkoderiu, nustatyti variacinio autoenkoderio tikslumo priklausomybę nuo latentinio vektoriaus bei molekulių ilgio.

Tyrime buvo apmokyti keturi variacinio autoenkoderio modeliai su skirtingais latentinių vektorių ilgiais lygiais 56, 156, 196 ir 254 bei keturi variacinio autoenkoderio modeliai, su skirtingais molekulių ilgiais, lygiais 60, 80, 100, 120. Modelių tikslumas tikrinamas trimis aspektais: mokymo ir validacijos imčių tikslumas, modelių gebėjimas atstatyti molekules tūkstančiu bandymų bei validžių molekulių generavimas. Visais aspektais mažiausiai tikslus buvo modelis su latentinės erdvės vektoriumi lygiu 56, todėl galima teigti, jog latentinės erdvės matmuo lygus 56 negali tinkamai reprezentuoti turimo molekulių duomenų rinkinio. Atitinkamai, galima teigti, jog turimą molekulių rinkinį geriausiai apibūdina latentinės erdvės matmuo lygus 156. Analizuojant modelius, su skirtingais molekulių ilgiais gauta išvada, jog modelio tikslumui atstatant molekules labai svarbi yra duomenų rinkinio, kuriuo naudojantis apmokomas modelis, struktūra.

Naujos vaistinės molekulės, su norima mažiausia inhibicijos konstanta, generuojamos dviem būdais: naudojantis variaciniu autoenkoderiu bei tiesine interpoliacija tarp latentinės erdvės vektorių. Generuojant naujas molekules variaciniu autoenkoderiu naudojant skirtingas latentinės erdvės dimensijas, gautos dvi vienodos molekules, tai parodo, jog latentinė erdvė, nepriklausomai nuo dimensijos, išdėsto panašias molekules šalia. Atlikus tiesinę septynių žingsnių interpoliaciją, nustatyta, jog pirmosiomis interpoliacijos iteracijomis, molekulės vizualiai panašesnės į pirminę molekulę, o paskutinėse iteracijose panašesnės į tikslo molekulę.

Šniokaitė, Milda. Generating de novo Drug-Like Molecules Using Deep Artificial Neural Networks. Master's Final Degree Project, supervisor assoc. Prof. Dr. Mantas Landauskas; Mathematics and Natural Sciences Faculty, Kaunas University of Technology.

Study field and area (study field group): Mathematical Science, Applied Mathematics.

Keywords: Drug-like Molecules Discovery, Variational Autoencoder, Generative Model, Deep Artificial Neural Networks.

Kaunas, 2019. 59 p.

Summary

Generating de novo drug-like molecules can lead to major societal and technological advancements. Molecule generation is relevant to solving problems in various fields: from individual treatment to energy production and storage. The importance and relevance of generating new drug-like molecules is straightforward, to find new drugs you need to look for de novo molecules. The use of deep neural networks to generate de novo drug-like molecules can accelerate the discovery of new drugs. To accomplish this task, the structure of neural networks is used - variational autoencoder: using encoder, the representations of the molecules are compressed into latent space vectors and subsequently decoded into valid molecules using a decoder. The aim of this study - to investigate the potential of new drug-like molecule generation using variational autoencoder, to determine the dependence of variational autoencoder from a length of latent vector and molecules.

In this study, four models of variational autoencoder were trained with different latent vector lengths at 56, 156, 196 and 254, and four models of variational autoencoder with different molecular lengths of 60, 80, 100, 120. The accuracy of models is checked by three aspects: training and validation samples accuracy, the ability of models to regenerate molecules with thousands of tests and the generation of valid molecules. In all respects, the least accurate model was the latent space vector at level 56, so it can be said that the latent space dimension of 56 cannot represent the molecular data set adequately. Accordingly, the available molecular set is best characterized by a latent space dimension equal to 156. The analysis of the models with different molecular lengths concluded that the structure of data set by which the trained model is used is very important for the model's accuracy in molecules recovery.

New drug-like molecules, with the desired minimum inhibition constant, are generated in two ways: using a variational autoencoder and linear interpolation between latent space vectors. By generating new molecules with a variational autoencoder using different latent space dimensions, two identical molecules were obtained, indicating that latent space, irrespective of dimension, locates similar molecules nearby. After the linear seven-step interpolation, it was found that the first iterations of interpolation molecules were visually more similar to the parent molecule, and in the last iterations they were more similar to the target molecule.

Turinys

Lentelių sąrašas

Paveikslų sąrašas

Santrumpų sąrašas

Doc. – docentas

AE – autoenkoderis (angl. *autoencoder*)

GRU – uždari rekurentiniai vienetai (angl. *gated recurrent units*)

SMILES – supaprastinta molekulinės įvesties linijinės įvedimo sistema (angl. *Simplified Molecular Input Line Entry System*)

IBM – JAV kompiuterių gamintoja, programinės ir techninės įrangos prekybos bendrovė (angl. *International Business Machines Corporation*)

JAV – Jungtinės Amerikos Valstijos

Įvadas

Naujų vaistinių molekulių generavimas gali sukelti didelę visuomenės ir technologijų pažangą. Molekulių generavimas aktualus sprendžiant įvairių sričių problemas: nuo individualus gydymo iki energijos gamybos ir sandėliavimo. Naujų vaistinių molekulių generavimo svarba ir aktualumas akivaizdus, norint atrasti naujus vaistus būtina ieškoti naujų molekulių.

Iki atsirandant cheminei informatikai, naujų molekulių kūrimas užtrukdavo ilgus metus, tad dabar pasitelkiant į pagalbą įvairius informacinių technologijų metodus naujos molekulės generuojamos efektyviau. Viena iš šiuo metu populiarių idėjų – pasitelkti dirbtinių neuroninių tinklų pagrindu veikiančius, naujų molekulių kūrimo algoritmus. Pastaraisiais dešimtmečiais dirbtinio intelekto metodai išpopuliarėjo dėl didelių pritaikymo galimybių. Neseniai pasiekta mašininio mokymosi pažanga lėmė galingus tikimybinius generuojančius modelius, kurie, išmokę realius pavyzdžius, sugeba pateikti realistiškus sintetinius mėginius. Šiame darbe, naujoms molekulėms kurti pasitelkiama neuroninių tinklų struktūra – variacinis autoenkoderis, kurio pagalba molekulės savybės yra užkuoduojamos į latentinės erdvės vektorių. Pridedant papildomą Gauso triukšmą galimas panašias savybes turinčių naujų molekulių generavimas.

Darbo tikslas – Ištirti naujų molekulių generavimo galimybes naudojantis variaciniu autoenkoderiu, nustatyti variacinio autoenkoderio tikslumo priklausomybę nuo latentinio vektoriaus bei molekulių ilgio.

Suformuluotam tikslui pasiekti darbe yra sprendžiami tokie uždaviniai:

1. Sudaryti variacinio autoenkoderio architektūrą. Apmokyti variacinį autoenkoderį su skirtingais latentinių vektorių ilgiais: 56, 156, 196, 254 bei palyginti modelių tikslumą.
2. Apmokyti variacinį autoenkoderį su tokiais molekulių ilgiais: 60, 80, 100, 120 bei palyginti modelių tikslumą.
3. Generuoti naujas vaistines molekules, su norimomis savybėmis, naudojant variacinį autoenkoderį, patikrinti sugeneruotų molekulių validumą.
4. Sugeneruoti naujas vaistines molekules naudojant interpoliacinius metodus bei patikrinti sugeneruotų molekulių validumą.

Šis baigiamasis projektas atliktas bendradarbiaujant su Baltijos pažangių technologijų institutu. Darbe pateiktas tyrimas yra didesnio projekto dalis, toliau Baltijos pažangių technologijų institute sugeneruotos naujos molekulės bus tiriamos įvairiais aspektas, išgryninamos molekulių savybės. Atlikus kompiuterizuotus tyrimus bei išrinktos tinkamiausias savybes turinčios molekulės bus susintetinamos laboratorijoje bei jų savybės tvirtinamos realiais fiziniais tyrimais. Kuomet gautos savybės bus patvirtintos, naujai sugeneruotos molekulės gali būti panaudotos naujų vaistų kūrime.

1. Literatūros apžvalga

Šioje dalyje bus analizuojama mokslinė literatūra, susijusi su molekulių generavimu bei giliaisiais dirbtiniais neuroniniais tinklais. Analizuojant literatūros šaltinius stebimos tyrimų tendencijos, kintamieji bei metodai, apibrėžiamos svarbiausios sąvokos bei teorinės išvados.

1.1. Cheminė informatika

Naujų medžiagų atradimas gali sukelti didžiulę visuomenės ir technologijų pažangą [1]. Siekiant atrasti naujas medžiagas, reikia žvelgti giliau į objektų struktūrą. Pagrindinis inovatyvių medžiagų kūrimas susideda iš naujų, dar nematytų molekulių paieškų. Naujų molekulių paieška būtina ir akivaizdžiai pagrindžiama – pažangios medžiagos yra daugelio problemų, pradedant nuo individualus gydymo baigiant energijos gamyba ir sandėliavimu, sprendimas. JAV gynybos ministerijos technologijų skyrius jau keletą metų vykdo inovatyvius, naujų molekulių paieškos metodų, projektus, grįsdamas tuo, kad efektyvus naujų molekulių atradimas ir gamyba yra būtini norint realizuoti Jungtinių Valstijų gynybos departamento pajėgumus, pradedant moderniais preparatais ir vaistais, kurie yra būtini kovojant su kylančiomis grėsmėmis [2]. Šiuo metu taikomi metodai apibūdinantys, kaip sukurti molekules tam tikrame taikyme, yra pagrįsti intuicijos pagrindu, lėtos iteracinės projektavimo bei bandymo ciklų metu ir galiausiai apribojamos specifinių chemiko žinių.

Cheminės informatikos lauką galima laikyti skaičiavimo chemijos dalimi, kurios modeliai nėra pagrįsti tikrosios fizikos ir chemijos atkūrimu, pagal kurį pasaulis veikia molekuliniu mastu [3]. Cheminėje informatikoje yra keletas pagrindinių užduočių susijusių su molekulėmis bei naujų molekulių paieška. Šie uždaviniai dažnai iškeliami ir bandomi pasiekti įvairiomis technikomis [4]:

- Duomenų bazių kūrimas – vienas iš plačiausiai siekiamų uždavinių yra sudaryti kuo geriau, lengviau ir kokybiškiau pasiekiamas duomenų bazes. Cheminė informatika suteikia prieigą prie didžiulių mastų cheminės informacijos duomenų bazėse, šis uždavinys pasiekiamas naudojantis moksline chemine literatūra bei jau atliktais tyrimais. Šiuo metu galima pasiekti daugiau kaip 90 milijonų žinomų junginių, tai būtų neįmanoma, jei ne cheminės informacijos duomenų bazės.
- Molekulių savybių prognozavimas. Svarbi cheminės informatikos dalis yra esamų ar naujai sukurtų molekulių savybių prognozavimas, naudojantis įvairiais skaičiavimo metodais ir nenaudojant realių fizinių tyrimų laboratorijoje. Šioje srityje naudojamos įvairios technikos, tokios kaip statistiniai metodai, raštų atradimas, duomenų gavyba, mašininis mokymas, dirbtiniai neuroniniai tinklai, genetiniai algoritmai, įvairūs kintamųjų parinkimo algoritmai ir įvairūs kiti.
- Analitinė chemija – dauguma analitinės chemijos užduočių apima klasifikavimo problemą, pavyzdžiui, mėginio priskyrimą konkrečiai kategorijai. Jau seniai buvo pripažinta, kad analitinių mėginių klasifikavimas gali būti palengvintas naudojantis skaičiavimo metodus.
- Vaistų kūrimas – iki šiol daugiausiai cheminės informatikos tyrimų atliekama siekiant atrasti naujus galimus vaistinius preparatus, ši užduotis glaudžiai susijusi su kitais cheminės informatikos

uždaviniais, naudojant juos procese. Šis cheminės informatikos uždavinys tiek paplitęs, kad visos pagrindinės vaistų bendrovės turi cheminės informatikos skyrius ir vystant visus naujai sukurtus vaistus buvo naudojami vieni ar kiti cheminės informatikos metodai.

Taigi, viena iš galimai svarbiausių naujų molekulių panaudojimo sričių – naujų vaistinių preparatų kūrimas. Pagrindinis vaistų atradimo uždavinys yra surasti tikslines molekules, turinčias pageidaujamų cheminių savybių, šiuo metu ši užduotis trunka ilgus metus kuriant ir tiriant chemikams ir farmakologams ekspertams [5].

Cheminė informatika prasidėjo lokaliais modeliais, paprastai kiekybinėmis struktūrų aktyvumo ar struktūrų savybių ryšių analizėmis. Ankstyvieji modeliai buvo pagrįsti tiesine, o vėliau daugialype tiesine regresine analize, paprastai šie modeliai buvo sudaromi naudojant nedaug kintamųjų ir galiojo tik nedideliam kiekiui artimai susijusių junginių [3].

Per pastarąjį dešimtmetį pastebimai padidėjo turimų aktyvių junginių ir biomedicinos duomenų kiekis, todėl kaip efektyviai panaudoti didelio masto chemijos duomenis tampa svarbiausia problema, susijusi su vaistų atradimu [6]. Chemijos ir statistinio mokymosi sąryšiai turi jau ilgą istoriją. Dirbtinis intelektas su mašininiu mokymusi turi didelį potencialą konkuruoti ir netgi viršyti įvairius tradicinius cheminės informatikos metodus, skirtus įvairioms užduotims spręsti [7].

Dirbtinio intelekto bei mašininio mokymo vienas iš svarbiausių privalumų yra plati metodų pasirinkimo galimybė. Akivaizdu, jog nėra vieno algoritmo, kuris pateiktų geriausią metodą bet kokiai problemai spręsti. Todėl santykiniai metodų gebėjimai priklauso nuo duomenų rinkinio dydžio ir pasiskirstymo cheminėje erdvėje, sprendžiamos cheminės problemos tiesiškumo ar kitokio, turimo deskriptoriaus pobūdžio ir vidinės koreliacijos ir ne lokalių duomenų tinkamumo bei daug kitų įvairių veiksnių [3].

1.2. Molekulių generavimas

Vis dar nėra tikslaus sutarimo, vertinant galimų vaistinių molekulių skaičių: priklausomai nuo jo įvertinimo būdo jis svyruoja nuo 10^{23} iki 10^{180}, P. G. Polishchukas su kolegomis straipsnyje „Vaistinių molekulių cheminės erdvės dydžio įvertinimas pagal GDB-17 duomenis“ apskaičiavo, kad potencialių objektų skaičius svyruoja nuo 10^{23} iki 10^{60} [8]. Šiandien akademinių, komercinių ir tinkamumo duomenų bazių įrašuose galima rasti tik apie 10^{8} esamų cheminių junginių struktūrų. Kadangi šios kolekcijos apsiriboja jau žinomais chemotipais, reikėtų stengtis sukurti virtualius junginius, kuriuose yra struktūrinių fragmentų, kurių dar nėra esamose struktūrose [8]. Taigi galima teigti, jog molekulių generavimas turi didelį potencialą, kadangi nėra atrasta net pusė galimų molekulių.

Šiuo atveju kuriamas atvirkštinis naujų medžiagų kūrimo procesas, kuris yra svarbi naujų sudėtingesnių medžiagų atradimo dalis. Naujų technologijų diegimo laikas nuo atradimo

laboratorijoje iki komercinio produkto, istoriškai trunka 15–20 metų. Šis procesas paprastai apima šiuos veiksmus [1]:

1. sukurti naują ar patobulintą medžiagos koncepciją ir imituoti jo potencialų tinkamumą;
2. susintetinti medžiagą;
3. įtraukti medžiagą į prietaisą ar sistemą;
4. apibūdinti ir matuoti norimas savybes.

Šis ciklas sukuria grįžtamąjį ryšį, kad kartotų, pagerintų ir atkurtų būsimus atradimų ciklus, o kiekvienas žingsnis gali užtrukti iki kelerių metų. Medžiagų inžinerijos eroje mokslininkai siekia pagreitinti šiuos ciklus, sumažindami laiką tarp žingsnių. Pagrindinis tikslas yra tuo pačiu metu pasiūlyti, kurti ir apibūdinti naujas medžiagas, kai kiekvienas komponentas vienu metu ir perduoda ir priima duomenis [1].

Pirmiausia atvirkštinio dizaino procesas prasidėjo nuo didelio našumo virtualaus parinkimo bei evoliucijos strategijos diskrečiuose optimizavimo metoduose. Kitais atvejais atvirkštinis dizainas įgyvendinamas, naudojant patobulintą *Bayeso* mėginių ėmimą su nuosekliu *Monte Carlo* metodu, invertuojančia *Hamiltonians* sistema, gaunant analitinius savybių gradientus molekulinės sistemos atžvilgiu, optimizuojant galimus cheminių sistemų energijos paviršius arba išskiriant projektavimo modelius naudojant duomenų gavybos technologijas [1].

Ir dabartinės tendencijos rodo, jog mokslininkai labai plačiai taiko gilias generacines modeliavimo technologijas molekulėms generuoti ir optimizuoti, D. C. Eltonas su kolegomis, apžvalgoje „Gilusis mokymasis generuojant ir optimizuojant molekules“ suskaičiavo 45 mokslinius straipsnius [9], publikuotus per pastaruosius dvejus metus, kuriuose įvairūs tyrėjai naudodami dirbtinių neuroninių tinklų metodus gilinasi būtent į šią cheminės informatikos sritį. Pažanga kuriant gilius generuojančius modelius sukėlė daugybę perspektyvių pasiūlymų, kaip spręsti molekulių sudarymo klausimą bei kaip atskleisti perspektyvią vaistų kūrimo kryptį. Vaistų atradimo metu labai svarbu yra molekulių, turinčių pageidaujamų molekulinių savybių, projektavimas [10].

Yra ne viena galimybė, kaip įgyvendinti molekulių generavimo užduotį, naudojant įvairias mašininio mokymo technikas. Dažniausiai literatūroje randami metodai yra:

- rekurentiniai neuroniniai tinklai [11, 12];
- sustiprintas mokymas [13];
- autoenkoderis [14, 15, 16];
- generaciniai priešiškieji neuroniniai tinklai [17, 18].

Šiame darbe molekulių generavimui bus naudojamas autoenkoderis (angl. *autoencoder*, AE), todėl tolimesnėje literatūros apžvalgoje bus gilinamasi į mokslinius straipsnius, kuriuose naudojamas būtent šis metodas, tačiau AE architektūroje galima sutikti ir kitų neuroninių tinklų objektų.

1.3. Biologiniai ir dirbtiniai neuroniniai tinklai

Biologinis neuronas buvo atrastas 1890 metais bei apibrėžtas kaip pagrindinis neuroninės sistemos vienetas, neuronai gali būti randami įvairių dydžių ir formų bei susideda iš pagrindinių dalių: dendritų, aksonų, ląstelės kūno bei sinapsinių ryšių [19]. Šį atradimą ir įrodymą pateikė ispanų mokslininkas S. Ramonas y Cajalas, laikomas šiuolaikinės neurologijos tėvu [20]. Po šio mokslininko pasiekimo, neuronai tapo įtvirtinta ir visuotinai naudojama sąvoka.

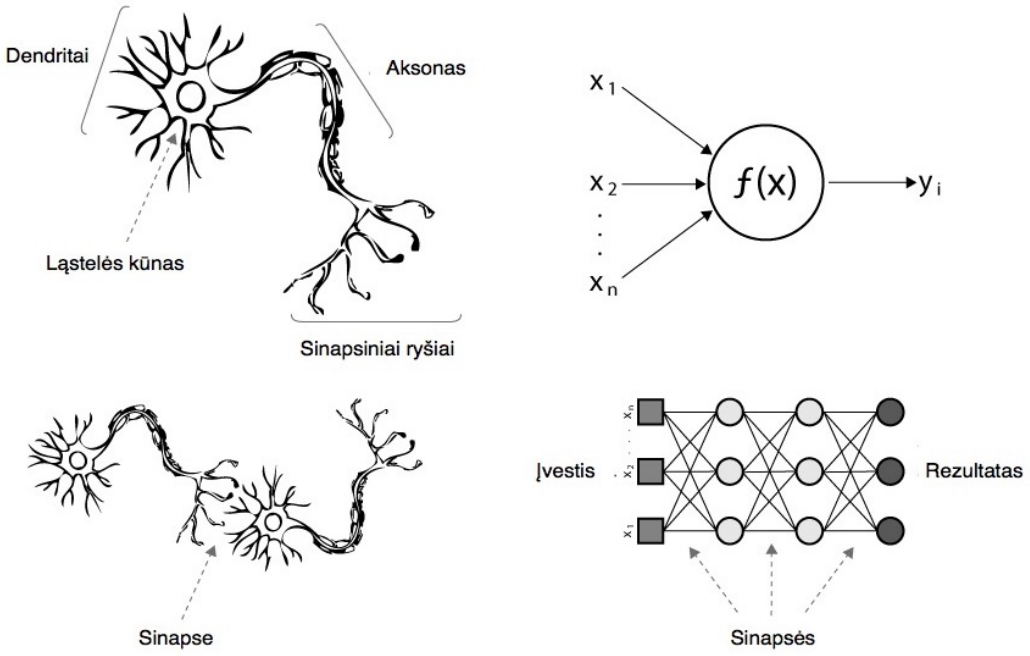
Vėliau buvo įrodyta, jog žmogaus nervinę sistemą sudaro milijardai įvairių tipų ir ilgių neuronų, priklausomai nuo jų vietos organizme [21]. Paprasčiausias biologinis neuronas turi tris pagrindinius funkcinius vienetus – dendritus, ląstelės kūną ir aksoną. Ląstelės kūnas turi branduolį, saugantį informaciją apie paveldimumo požymius ir plazmą, kurioje yra molekulinė įranga, naudojama neuronui reikalingų medžiagų gamybai. Dendritai gauna signalus iš kitų neuronų ir perduoda juos ląstelės kūnui. Aksonas, kuris skaidomas į dalis, gauna signalus iš ląstelės kūno ir per sinapses perneša juos į kaimyninių neuronų dendritus. Impulsas, elektrinio signalo pavidalu, tokiu būdu keliauja ir per tolimesnius neuronus. Signalo kiekis, praeinantis per priimantį neuroną, priklauso nuo signalo intensyvumo, jo sinapsinės stiprybės ir priimančio neurono slenksčio. Kadangi neuronas turi daug dendritų ir sinapsių, jis gali priimti ir perduoti daug signalų vienu metu. Šie signalai gali sužadinti arba slopinti neuroną [21].

Žmogaus nervinę sistemą galima laikytis biologiniu kompiuteriu, susidedančiu iš daug bei įvairių neuronų. Visi šie neuronai sujungti tarpusavyje sudaro didelį bendrą tinklą. Šio biologinio neuroninio tinklo centrinė dalis yra smegenų žievė (smegenys), kuri yra 2–3 milimetrų storio plokščia, masyviai tarpusavyje sujungtų neuronų plokštė, kurios apytikslis paviršiaus plotas yra 2200 kvadratinių centimetrų [21]. Kiekvienas neuronas yra prijungtas prie 1 000–10 000 kitų neuronų, sudarant maždaug 10^{14}–10^{15} jungčių, lyginant, dirbtinius neuroninius tinklus sudaro nuo 10 iki 10 000 neuronų, kurių ryšių tankis svyruoja nuo 5 iki 100 jungčių per neuroną [21].

Kalbant apie jų veikimą ir vidinę struktūrą, dirbtiniai neuroniniai tinklai laikomi vienarūšiais ir dažnai veikia deterministiškai, o žmogaus smegenų žievė yra labai heterogeninė ir veikia sudėtingu deterministiniu ir stochastiniu pobūdžiu. Kalbant apie funkcionalumą, dirbtiniai neuroniniai tinklai yra sukurti imituoti smegenų skaičiavimo savybes, tokias kaip adaptyvumas, triukšmo duomenyse ir klaidų tolerancija [21]. Neapdorota analogija tarp dirbtinio ir biologinio neurono yra ta, kad ryšiai tarp dirbtinių neuronų yra aksonų ir dendritų atitikmuo, svoriai neuronuose atitinka sinapses, o slenkstis aproksimuoja aktyvumą (žr. 1 pav.). Taip pat svarbus aspektas, jog ir biologinis tinklas, ir dirbtiniai neuroniniai tinklai mokosi laipsniškai koreguojant svorių ar sinapsių dydžius [21].

Žmogaus smegenys, kaip ir kompiuteris, sudarytos iš daugelio paprastų elementų bei vykdo įvairias skaičiavimo funkcijas, operuoja elektroniniais signalais, mokosi, priima abstrakčius sprendimus ir daro klaidas. Tačiau galima įvardinti ir daug įvairių skirtumų tarp žmogaus smegenų

ir kompiuterio. Biologinių neuronų tarpusavio komunikacijos greitis yra pakankamai lėtas lyginant su šiuolaikinio kompiuterio galimybėmis. Tačiau sudėtingi uždaviniai žmogaus smegenimis sprendžiami greitai, kai lyginama su kompiuteriu. Taip yra todėl, nes žmogaus smegenys gali atlikti užduotis vykdant skaičiavimus lygiagrečiai, t. y., uždavinių sprendime, žmogaus smegenyse dalyvauja labai didelis kiekis neuronų, o kompiuteriai neturi tokių didelių išteklių. Dar vienas aspektas, kuo panašūs dirbtiniai neuroniniai tinklai bei biologiniai neuroniniai tinklai, jog abejuose yra aktyvacijos savybė – nors ir nėra tiksliai žinoma, kas įvyksta somoje su įvesties duomenimis, tačiau gaunant pakankamą kiekį stimuliacijos iš dendritų, neuronas gali aktyvuotis ir perteikti elektrocheminį signalą per aksoną toliau, jeigu stimuliacija nepakankamai didelė, neuronas nesugeneruoja jokio rezultato ir signalas nuslopinamas.



1 pav. Biologinių ir dirbtinių neuronų sąsaja

Dirbtinis neuronas yra pagrindinis bet kurio dirbtinio neuroninio tinklo elementas, kaip minėta, jo dizainas ir funkcionalumas gaunamas stebint biologinį neuroną, kuris yra pagrindinis biologinių nervinių tinklų, apimančių smegenis ir nugaros smegenis, blokas (žr. 1 pav.).

Paprasčiausias dirbtinis neuronas, tai matematinis modelis – funkcija, turinti tris paprastus taisyklių rinkinius: dauginimą, sumavimą ir aktyvavimą [22]. Dirbtinio neurono įėjime įvestys „pasveriamos“, o tai reiškia, kad kiekviena įvesties vertė dauginama iš individualaus svorio. Vidutinėje dirbtinio neurono dalyje yra sumos funkcija, sudedanti visas svertines įvestis ir poslinkius. Kuomet išeinama iš dirbtinio neurono, vidutinėje dalyje apskaičiuota reikšmė praleidžiama per aktyvacijos funkciją, kuri taip pat vadinama perdavimo funkcija. Dirbtinio neurono modelio paprastumą galima suvokti žemiau pateiktame matematiniame aprašyme [22]:

$$y(k) = \sigma\left(\sum_{i=0}^{m} w_i(k)x_i(k) + b\right), \quad (1)$$

kur $y(k)$ – rezultatas, $w_i(k)$ – svorio reikšmė, $x_i(k)$ – įvestis, b – poslinkis, o σ aktyvacijos funkcija.

Kuomet derinama du ar daugiau dirbtiniai neuronai, gaunamas dirbtinis neuroninis tinklas, kaip pavaizduota 1 paveiksle. Jeigu galime teigti, jog vienas dirbtinis neuronas iš esmės neturi naudos sprendžiant gyvenimiškas problemas, dirbtinių neuroninių tinklų nauda – akivaizdi. Iš tiesų, dirbtiniai neuroniniai tinklai gali išspręsti sudėtingas, kompleksiškas, realaus gyvenimo problemas, apdorodami informaciją savo pagrindiniuose blokuose – dirbtiniuose neuronuose netiesiniu, paskirstytu, lygiagrečiu ir vietiniu būdu [22].

Atskirų dirbtinių neuronų tarpusavio sąveika vadinama dirbtinių neuroninių tinklų topologija arba architektūra [22]. Tai, kad sujungimą galima atlikti daugeliu būdų, sukuria daugybę galimų topologijų. Reikia paminėti, kad būtų galima dirbtinį neuroninį tinklą lengviau valdyti ir matematiškai apibūdinti, atskiri neuronai grupuojami į sluoksnius, kurie skirstomi taip: įvesties, paslėpti bei išvesties sluoksniai.

Kuomet jau parenkama tinkama dirbtinio neuroninio tinklo architektūra, atlikta tik dalis užduoties, kad būtų galima panaudoti šį dirbtinį neuroninį tinklą problemoms spręsti. Kaip ir biologiniai neuroniniai tinklai turi išmokti tinkamai reaguoti į tam tikrus aplinkos šaltinius, taip ir dirbtiniai neuroniniai tinklai turi daryti tą patį. Taigi kitas žingsnis yra išmokyti dirbtinį neuroninį tinklą tinkamai reaguoti, o tai gali būti pasiekta mokantis. Nesvarbu koks būdas naudojamas mokyme, mokymosi užduotis yra nustatyti svorių ir poslinkių reikšmes mokymosi duomenis, kad būtų sumažinta pasirinkta tikslo funkcija [22].

1.4. Dirbtinių neuroninių tinklų mokymas

Kaip jau minėta, dirbtinio neuroninio tinklo gebėjimas mokytis yra viena iš svarbiausių dalių. Skirtingi turimi duomenys ir skirtingos tinklų struktūros reikalauja ir skirtingų mokymo būdų. Galima išskirti tris neuroninių tinklų mokymo tipus:

- mokymas su mokytoju arba prižiūrimas mokymas (angl. *supervised learning*);
- mokymas be mokytojo arba neprižiūrimas mokymas (angl. *unsupervised learning*);
- hibridinis mokymas (angl. *hybrid learning*).

Kai naudojamas mokymas be mokytojo, tinklui paduodamas duomenų rinkinys, turintis daug įvairių ypatybių, o gaunamas rezultatas – naudingos duomenų rinkinių struktūros savybės [23]. Gilaus mokymo kontekste, norima sužinoti visą tikimybinį pasiskirstymą, kuris sukūrė turimą duomenų rinkinį, nesvarbu, ar tai bus tankis, ar netiesioginės užduotys, tokios kaip sintezė ar denoizavimas [23]. Kai kurie kiti neprižiūrimo mokymosi algoritmai gali atlikti ir kitas įvairias užduotis, pvz., klasterizavimą, grupavimą ar panašiai.

Kai naudojamas mokymas su mokytoju, tinklui paduodamas duomenų rinkinys, turintis daug įvairių ypatybių, tačiau taip pat kiekvienas elementas yra susijęs su žymės taikiniu, t. y., žinomos norimos išeities reikšme [23]. Šiuo atveju mokymasis vyksta ieškant tokių svorių, kurie sumažintų skirtumą tarp norimų išėjimo reikšmių ir tinklo prognozuojamo rezultato iki minimumo.

Hibridinio mokymosi atveju dalis dirbtinio neuroninio tinklo svoriai nustatomi pagal prižiūrimą mokymą, kita dalis pagal neprižiūrimą mokymą.

Jau yra sukurta daug dirbinių neuroninių tinklų mokymosi algoritmų, tačiau klaidos sklidimo atgal algoritmas dirbtinių neuroninių tinklų mokyme yra populiariausiai naudojamas [24], taip pat šis algoritmas naudojamas ir AE mokyme. Nors AE priskiriama neprižiūrimam mokymui, t. y., mokant šį dirbtinį neuroninį tinklą turimi duomenys nėra žymėti, tačiau naudojantis įvesties duomenimis galima tinklą mokyti klaidos sklidimo atgal algoritmu. Šiam algoritmui būtina žinoti teisingą rezultatą, t. y., jis priskiriamas mokymui su mokytoju. Klaidos sklidimo atgal algoritmas veikia matuojant išėjimo klaidą, apskaičiuojant šios klaidos gradientą ir koreguojant neuroninio tinklo svorius (ir paklaidas) mažėjančioje gradiento kryptyje. Šis algoritmas remiasi gradientinio nusileidimo optimizavimo metodu [25].

1.5. Dirbtinių neuroninių tinklų pradžia

Per pastaruosius du dešimtmečius mašininis mokymasis smarkiai progresavo, nuo pradinių eksperimentų laboratorijose iki praktinės technologijos plačiai paplitusiam komerciniam naudojimui. Dirbtinio intelekto srityje mašininis mokymasis tapo dažnai pasirenkamu metodu kuriant praktinę programinę įrangą, skirtą kompiuterinei vizijai, kalbos atpažinimui, natūralios kalbos apdorojimui, robotų valdymui ir kitoms reikmėms. Daugelis dirbtinio intelekto sistemų kūrėjų jau pritaria, kad daugelių problemų sprendimui, lengviau mokyti sistemą, parodant norimo įvesties – išvesties elgesio pavyzdžius, nei programuoti ją rankiniu būdu, numatant norimą atsaką visoms galimoms įvestims [26]. Mašininio mokymosi poveikis jaučiamas kompiuterių moksle, įvairiose pramonės šakose bei įvairiuose empiriniuose moksluose nuo biologijos, socialogijos iki kosmologijos, susijusiose su didelės apimties duomenimis, nes mašininis mokymasis pateikia naujus būdus, kaip interpretuoti duomenis [26].

Žvelgiant į dirbtinio intelekto, mašininio mokymo bei neuroninių tinklų istoriją, akivaizdu, jog pirmaisiais žingsniais galima laikyti statistikos, kaip mokslo, atsiradimą, kadangi šie metodai glaudžiai susiję su įvairiomis statistikos formomis. Tačiau pirmieji realūs žingsniai link mašininio mokymo, tokio, kokį turime šiandien, prasidėjo nuo D. O. Hebbio 1949 metais, kuomet remiantis neuropsichologija buvo pristatyta mokymosi formuluotė [27]. Vėliau 1952 metais A. Samuelis sukūrė IBM programą, kuri žaidė šaškes [27]. Kompiuteris sugebėjo stebėti pozicijas ir išmokti tam tikrus elgesio modelius, kurie padėdavo žaisti geriau, po daugelio sužaistų partijų, A. Samuelis pastebėjo, jog programa per tam tikrą laiką sugebėjo išmokti geriau žaisti. 1954 metais M. Minskis pateikė daktaro disertaciją, susijusią su mašininiu mokymu [28]. Vėliau 1957 metais F.

Rosenblatas sukūrė pirmąjį neuroninį tinklą kompiuteriams – perceptroną, kuris simuliavo žmogaus smegenų minties procesus [27]. Po trijų metų H. Widrowas pasiūlė Delta mokymosi taisyklę, taip pat žinomą, kaip mažiausių kvadratų metodas (angl. *least square*), kuri tuo metu buvo naudojama perceptronų mokymui. Apskritai, 1960-aisiais, kai buvo pateikta labai optimistinių pretenzijų dėl neuronų tinklo pajėgumų, susidomėjimas padidėjo, tačiau iki to laiko M. Minskis buvo nusivylęs vykstančiu tyrimu. Jis nustatė, kad daugelis dėsningumų negali būti išmokti paprasto vieno sluoksnio neuroniniu tinklu, o daugiasluoksniai tinklai vis dar buvo prastai suprantami. Nebuvo teorijos, susijusios su daugiasluoksnių tinklų mokymu ir nebuvo įrodymų, kad daugiasluoksniai tinklai iš esmės buvo pranašesni už vieno sluoksnio tinklus [28].

Vėliau, M. Minskis su kolega S. Papertu parašė knygą, pavadinimu „Perceptronai“, pristatydami išsamų teorinį vieno sluoksnio tinklo apdorojimą ir įrodydami, kad šie tinklai negali išspręsti paprastų problemų ir nepasižymi niekuo išskirtiniu [29]. M. Minskio prestižas, kartu su įtikinamais teoriniais aspektais aprašytais knygoje, kitus 15 metų veiksmingai nužudė valstybinį neuroninių tinklų tyrimų finansavimą. Be to, Sovietų Sąjunga atšaukė keletą labai perspektyvių projektų, o mokslininkai visame pasaulyje ieškojo daugiau žadančių sričių. Nepaisant paramos praradimo, keli mokslininkai tęsė darbą ir palaipsniui sukūrė teorinius daugiasluoksnių tinklų pagrindus ir parengė praktinius jų įgyvendinimo algoritmus [28]. Deja, jie patyrė rimtų sunkumų skelbdami mokslinius straipsnius, todėl jų darbai yra išsklaidyti tarp daugelio neaiškių žurnalų – tai dar labiau apsunkina šios srities mokslinius tyrimus.

Bet kokiu atveju, mokymosi algoritmai su įrodytomis konvergencijos charakteristikomis buvo sukurti daugiasluoksniams tinklams, nepaisant M. Minskio pesimizmo. Šie tinklai suteikė galimybę išspręsti išskirtines problemas, o su tinkamu sudėtingumu demonstravo labai puikius mokymosi gebėjimus.

Didelis pokytis įvyko, kuomet P. J. Werbosas 1981 metais pristatė daugiasluoksnį perceptroną su atgaliniu klaidos sklidimo algoritmu, kuris ir šiomis dienomis yra vienas iš plačiausiai naudojamų neuroninių tinklų struktūrų. Po šio metodo pristatymo, mašininio mokymo sritis vėl suklestėjo ir buvo kuriami nauji algoritmai bei metodai. 1986 metais J. R. Quinlanas pasiūlė labai gerai žinomą mašininio mokymo algoritmą – sprendimų medžius [28]. Apskritai jau 1990-aisiais darbas mašininio mokymosi srityje pereina nuo žiniomis paremto požiūrio į duomenis prie gilesnio mašininio mokymo, mokslininkai pradeda kurti kompiuterių programas, skirtas analizuoti didelius duomenų kiekius ir padaryti išvadas – mokytis iš rezultatų. Tolimesnis mašininio mokymosi tobulėjimas siekia ir šiandienos visuomenę, dabar dirbtinis intelektas, mašininis mokymas bei neuroniniai tinklai plačiai naudojami įvairiose srityse, o ir paprastas žmogus su juo susiduria kasdieną.

1.6. Autoenkoderiai

AE yra neuroninis tinklas, išmokytas mėginti nukopijuoti įvestį į išvestį [23], t. y., šio neuroninio tinklo įvestis ir išvesti turi sutapti kaip įmanoma labiau. Pagrindinė AE idėja yra suspausti informaciją automatiniu būdu. AE pirmą kartą pristatyti 1980-aisiais, G. E. Hintono ir jo mokslo grupės [30], šiuo neuroniniu tinklu buvo siekiama išspręsti atgalinio sklidimo algoritmo be mokytojo problemą, naudojant įvesties duomenis, kaip mokytoją. Nuo tada galima sakyti, kad AE atlieka esminį vaidmenį neprižiūrimame mokyme ir giliose neuroninių tinklų architektūrose [31].

AE galima suskirstyti į dvi dalis: koderį ir dekoderį. Tarp jų yra vadinamasis butelio kaklelis (angl. *bottleneck*), t. y., informacijos susiaurėjimas. Priklausomai nuo AE rūšies, dažniausiai turimas vektorius, kuriame yra informacija, artimiausia duomenims. Koduotojas priima informaciją ir neuroniniu tinklu suspaudžia informaciją į savybių vektorių, kuriame kiekvienas matmuo rodo tam tikrą išmoktą požymį apie duomenis. Dekoderis atlieka atvirkštinį darbą, kaip įvestį ima savybių vektorių ir grąžina iškoduotą duomenų eilutę. Visa AE esmė yra savybių vektoriuje. Tradiciškai AE buvo naudojami dimensijų mažinimui arba savybių mokymui, o neseniai teoriniai ryšiai tarp AE ir latentinių kintamųjų įtraukė AE į generacinio modeliavimo sritį, kaip lyderius [23].

Per pastaruosius dešimt metų galima atrasti įvairių puikiai pritaikytų AE įvairiausiose srityse. AE gali būti pritaikytas organams identifikuoti magnetinio rezonanso medicininiuose vaizduose, pasižyminčiuose vizualiais ir laikinais hierarchiniais požymiais, išmokusiais kategorizuoti objektų klases iš nepažymėto daugiarūšio duomenų rinkinio ir tam reikalingas tik silpnai prižiūrimas mokymas [32]. Taip pat demonstruojamas iš dalies tikimybinis metodas, skirtas objektams aptikti, kuris naudojamas organų lokalizavimui. Tyrimas parodo būtent šios rūšies neuroninių tinklų modelių potencialą taikyti medicininiams vaizdams, nepaisant to, kad sunku gauti teisingai pažymėtų mokymo duomenų rinkinius [32].

Kita AE pritaikymo galimybė buvo pateikta 2013 metais, šis gilus AE naudojamas triukšmui mažinti ir kalbai stiprinti balso įrašų duomenų rinkiniuose [33]. Šiame tyrime mokant AE buvo naudojamas triukšmo mažinimo procesas, naudojant triukšmingas ir švarias kalbų poras. Šio metodo privalumas yra tas, kad gilus AE automatiškai žino statistinį skirtumą tarp kalbos ir triukšmo, kuris padeda atskirti šias dvi klases, siekiant stiprinti balsą įraše. Be to, palyginti su tradiciniais kalbos tobulinimo metodais, AE gali ištirti netiesinę ir aukštos eilės statistinę informaciją, kuri yra svarbi kalbėjimui stiprinti [33].

Kitas AE pritaikymo pavyzdys yra susijęs su erdvėlaivių telemetrijos duomenimis. Straipsnyje siūloma naudoti AE su netiesiniu matmenų mažinimu anomalijos aptikimo užduotyje [34]. Autoriai taiko dimensijos matmens mažinimą, naudodami AE ir lygina jį su linijine ir branduoline pagrindinių komponenčių analize. Šiame tyrime parodyta, kad AE gali aptikti

subtilias anomalijas, kurių linijinei pagrindinių komponenčių analizei nepavyksta, be to, AE naudingi kaip netiesiniai metodai be kompleksinio skaičiavimo, kuris yra branduolinės pagrindinių komponenčių analizes minusas. Taip pat AE įtaisai gali padidinti anomalijų aptikimo tikslumą, naudojant triukšmo mažinimo metodus [34].

AE taip pat taikomi tokiuose tyrimuose kaip efektyvusis gedimų diagnozavimas, kadangi puikiai nustato sveikatos būklę signalams, kuriuose yra aplinkos triukšmo bei darbo sąlygų svyravimų [35]. Žvelgiant į kitokias problemas, 2014 metais pristatytas naujas vaizdu grįstas 3D formos paieškos metodas, naudojant AE, kuris naudojamas 3D formoms atpažinti, tai naudinga plėtojant 3D spausdintuvus bei lazerinius skaitytuvus [36]. AE, kaip ir visi dirbtinio intelekto modeliai, taip pat naudojami robotų manipuliavimo užduotyse. Šiuo atveju, naudojant AE įgyjamas savybių taškų rinkinys, apibūdinantis dabartinės užduoties aplinką, o naudojant išmoktą informaciją apie objektų pozicijas, sistema mokosi judėjimo įgūdžių [37].

AE plačiai naudojamas įvairioms problemoms spręsti, taip pat AE plačiai naudojami ir molekulėms generuoti. Tiksliau, dažniausiai naudojama AE modifikacija – variacinis AE, kadangi variacinis AE yra tikimybinis ir generuoja šiek tiek skirtingus rezultatus iš latentinės erdvės vektoriaus, todėl variacinis AE žymiai populiaresnis, nei bet koks kitas AE molekulių generavimui. Naudojant variacinį AE, nereikalinga papildoma sistema, jog būtų galima generuoti naujas molekules. Toliau apžvelgsime mokslinius tyrimus, kuriuose ir naudojamas variacinis AE.

1.7. Variacinių autoenkoderių pavyzdžiai, generuojant molekules

Vienas iš daugiausiai cituojamų mokslinės literatūros šaltinių, skirtų molekulių generavimui – R. Gomezo-Bombarellio ir kolegų straipsnis „Automatinis cheminis dizainas, naudojant duomenų srautinį nuolatinį molekulių vaizdavimą“ [16]. Straipsnyje pirmą kartą pristatytas variacinis AE, siekiant generuoti naujas molekules, kurios būtų tinkamos naujoviškų vaistų kūrimui. Nuo to laiko sparčiai vystėsi gilių generuojančių sistemų modeliavimas, naudojant keletą skirtingų giliųjų mokymosi architektūrų ir kitaip patobulintų variantų.

Šio cheminio variacinio AE architektūrą sudaro koderis, dekoderis ir prognozavimo sluoksnis. Kodavimo tikslas – konvertuoti atskiras molekulių reprezentacijas į realiai vertinamus fiksuoto dydžio nepertraukiamus vektorius – latentinę erdvę. Tada dekoderis transformuoja vektorius į SMILES eilutes. Variacinis AE naudojamas siekiant sumažinti rekonstrukcijos klaidą mokymo metu. Tam, kad būtų užtikrintas dekoduotų SMILES eilučių galiojimas, AE pridedamas Gauso triukšmas, užtikrinantis galiojantį dekodavimą, verčiantį dekoderį išmokti dekoduoti taškus latentinėje erdvėje. Duomenys, naudojami variaciniam AE mokymui, buvo gauti iš QM9 [38] (su 108 000 molekulių mažiau nei devyniais sunkiais atomais) ir ZINC [39] duomenų bazių (su 250 000 vaistinių junginių). Taip pat verta paminėti, kad abu: kodavimo ir dekodavimo procesai yra tikimybiniai. Latentinių taškų paėmimo tikimybė yra atvirkščiai proporcinga Euklido atstumui nuo pradinių taškų, taip pat iškodavimo koeficientas, rodantis galimą molekulinių struktūrų koreliaciją

latentinėje erdvėje [10]. Statistiškai galima rasti apie 30 latentinių molekulių netoli vieno natūralaus latentinio taško.

Bendrai mokant variacinį AE, naudojant pusiau prižiūrimą mokymąsi, latentinė molekulinė erdvė pati persitvarko taip, kad molekulės su panašiomis savybėmis būtų artimos viena kitai [1]. Tokiu atveju naudojantis Gauso triukšmu, kuris buvo įtrauktas į AE, galima pademonstruoti vietinio arba globalaus optimizavimo galimybes per visą sukurtą paskirstymą. Kita panaudota galimybė, kaip generuoti naujas molekules, buvo sferinė interpoliacija latentinėje erdvėje. Šiuo atveju norint pasiūlyti naujas perspektyvias molekules, pradedama nuo vienos molekulės latentinio vektoriaus bei judama ta kryptimi, kuria labiausiai tikėtina, kad pagerins norimą atributą. Tuomet sukurti nauji latentiniai vektoriai gali būti iškoduoti į SMILES molekulės reprezentacijas, naudojant dekoderį.

Remiantis R. Gomezo-Bombarellio ir kt. straipsniu T. Blaschkesas su kolegomis atliko tyrimą bei išmokė ir palygino keturis skirtingus AE: variacinį AE, kuris nenaudoja priverstinio mokymo, variacinį AE, kuris naudoja priverstinį mokymą bei du priešpriešinius AE, kurie buvo išmokyti sekti Gauso arba Tolydųjį pasiskirstymą [40]. Variacinių AE architektūra šiuo atveju yra labai panaši į R. Gomezo-Bombarellio straipsnyje pristatytą struktūrą. Visi modeliai buvo mokomi naudojant ChEMBL [41] duomenų bazę su 1,3 milijono SMILES eilučių. Latentinės erdvės vektoriaus dimensija buvo parinkta 56 visiems AE. Visi metodai parodė puikią rekonstrukciją naudojant SMILES sekų eilutes modelio apmokyme, ne mažiau kaip 95% visų simbolių yra teisingai rekonstruoti. Variacinis AE su priverstiniu mokymu parodė 1–2% geresnį atlikimą, lyginant su variaciniu AE, nenaudojančiu priverstinio mokymo. Tačiau, abu priešpriešiniai AE pasiekė didesnį tikslumą, nei variacinis AE su priverstiniu mokymu, tai rodo, kad AE dekoderiai labiau priklauso nuo informacijos, esančios latentinėje erdvėje, nei nuo ankstesnio sugeneruoto simbolio.

Analogiškai, J. Lim ir kt. straipsnyje „Molekulinis generacinis modelis, pagrįstas sąlyginiu variaciniu AE naujų molekulių dizainui“ pristatė sąlyginį variacinį AE [14]. Pasirinktas sąlyginis variacinis AE yra vienas iš populiariausių generuojančių modelių, kuris sukuria panašius, bet ne identiškus objektus. Nuo variacinio AE jis skiriasi, kadangi galima įvesti tam tikras sąlygas kodavimo ir dekodavimo procesuose. Šiame tyrime kaip sąlyginis vektorius koderiui ir dekoderiui buvo pateikta molekulinės savybės, kurias norima kontroliuoti. Gautas rezultatas – sąlyginis variacinis AE gali generuoti molekules su tikslinėmis savybėmis, kurias nustato sąlyginis vektorius. Šiame tyrime sąlyginio variacinio AE mokymui buvo naudota atsitiktinai atrinktų 500 000 ZINC [39] duomenų bazės molekulių. Nepaisant to, kad iš esmės sąlyginis variacinis AE buvo sėkmingai pritaikytas molekulių, su norimomis savybėmis, generavimui, reikia paminėt, kad pagrindinis metodo trūkumas yra tai, jog pageidaujamų molekulių generavimo sėkmės rodiklis yra labai mažas [14].

Kitą variacinio AE pritaikymo galimybę molekulių generavimui pristatė S. Kangas ir K. Cho straipsnyje „Sąlyginis molekulinis dizainas su giliais generuojančiais modeliais“ [15]. Šiame tyrime sukurtas sąlyginis molekulinio dizaino modelis, kuris tuo pačiu metu atlieka tiek savybių prognozavimą, tiek molekulių generavimą, naudojant pusiau prižiūrimą variacinį AE. Atsižvelgiant į specifinių savybių rinkinį, sąlyginė molekulinė konstrukcija atliekama tiesiogiai imant naujas molekules iš sąlyginio generacinio paskirstymo, nenaudojant jokios papildomos optimizavimo procedūros. Vienas iš naudingų pusiau prižiūrimo variacinio AE aspektų, jog šis modelis gali efektyviai išnaudoti nepažymėtas molekules, ši savybė naudinga, ypač kai nedidelė dalis molekulių yra pažymėtos, o tai yra gana dažna problema, kylanti dėl didelių molekulių savybių žymėjimo kaštų. Eksperimentai, naudojant iš ZINC [39] duomenų bazės paimtas vaistines molekules (310 000 vienetų), sėkmingai parodė veiksmingumą tiek savybių prognozavimo, tiek sąlyginio molekulinio dizaino požiūriu. Pusiau prižiūrimo variacinio AE architektūrą sudaro trys rekurentiniai neuroniniai tinklai: savybių prognozavimo tinklas, koderio tinklas bei dekoderio tinklas. Latentinės erdvės vektoriaus dimensija buvo parinkta 100 [15].

Dar vieną išplėsto AE versiją 2018 metais pristatė S. Harel ir K. Radinsky [42]. Variacinis AE buvo išplėstas, kad būtų galima sąlyginė atranka, t. y., taškų ėmimas iš vaistinių molekulių duomenų pasiskirstymo, kurie yra arčiau konkretaus įvesties objekto. Tai leidžia sugeneruotas molekules priartinti prie vaisto prototipo ir tokiu būdu padidinti tikimybę sugeneruoti galiojantį vaistą su panašiomis savybėmis. Be to, pridedamas įvairovės komponentas, kuris leidžia sugeneruotoms molekulėms šiek tiek skirtis nuo vaisto prototipo. Rezultatai rodo, kad šis modelis generuoja molekules panašias į vaistų prototipą, tačiau skirtingas tarpusavyje. Šis modelis buvo išmokytas naudojant 200 000 atsitiktinai atrinktų vaistinių molekulių, iš ZINC [39] duomenų bazės, o generavimo metu išplėstinis variacinis AE intuityviai ieško naujų molekulių arčiau vaistų prototipų, tai sukelia didesnę tikimybę sukurti naujus vaistus. Pateikti rezultatai rodo, kad daug sugeneruotų molekulių yra tinkamos ir naujos. Taip pat stebint sugeneruotas naujas molekules, pastebėta, jog išplėstinis variacinis AE sugebėjo sugeneruoti jau žinomus vaistus, su kuriais sistema niekada nebuvo susidūrusi.

Ir daugiau variacinių AE buvo sukurtų, naudojantis R. Gomezo-Bombarellio ir kt. straipsniu, darant papildomus pakeitimus, ar naudojant kitokius duomenis. Vienas iš populiarių pakeitimų – naudoti variacinį AE pritaikant molekulių grafinei reprezentacijai. Remiantis panašia architektūra, M. Simonovskio ir N. Komodakio straipsnyje „Grafinis variacinis AE: smulkių grafų generavimas, naudojant variacinį AE“ [43], kuriamas AE, kuris iš molekulės grafo gali išskirti latentinį vektorių, o vėliau dekoderio pagalba iš latentinės erdvės taško grįžti prie molekulės grafo. AE struktūra modifikuota, koderį sudaro du konvoliuciniai sluoksniai, bei pilnai sujungtas sluoksnis, dekoderį sudaro trys pilnai sujungti sluoksniai, taip pat naudojama populiariausia aktyvacijos funkcija – ReLU. Tyrime naudojami du duomenų rinkiniai: QM9 [38] su 134 000

molekulių bei ZINC [39] duomenų bazės rinkinys su 250 000 molekulių. Šis tyrimas parodė, kad AE gali kokybiškai įsisavinti mažų molekulių grafus ir juos generuoti, tačiau dekoderis sunkiai užfiksavo sudėtingas chemines sąveikas didesnėms molekulėms. Taip pat variacinis AE grafams buvo pritaikytas ir B. Samanta ir kolegų darbe „Gilusis generacinis molekulių grafų modelis“ [44]. Lyginant, su M. Simonovskio ir N. Komodakio modeliu, šis AE papildytas taip, kad galėtų generuoti naujas molekules su norimomis savybėmis. Dar vienas įdomus, AE pritaikymas, molekulių generavimui pristatytas D. Kuzminykhio ir kolegų straipsnyje „3D molekulinės reprezentacijos, pagrįstos bangos transformacija, konvoliuciniams neuroniniams tinklams“ [45]. Šis AE sudarytas iš konvoliucinių ir pilnai sujungtų sluoksnių, o įvestis – molekulių reprezentacija 3D grafais, pagrįstais bangos transformacija. AE pagalba sėkmingai buvo sugeneruotos naujos validžios molekulės reprezentuotos 3D grafais [45].

Nuo 2016 metais R. Gomezo-Bombarellio ir kt. pristatyto variacinio AE molekulių generavimui plačiai naudojami ir kuriami nauji mašininio mokymo metodai. Galima pastebėti, jog populiariausia naudojama molekulių reprezentacija variaciniame AE – SMILES eilutės, todėl tolimesniame skyrelyje bus aprašyta SMILES eilutės bei jų sudarymas. Taip pat turimi duomenys – molekulės yra reprezentuotos SMILES eilutėmis.

1.8. SMILES notacija

Šiame darbe reprezentuoti vaistines molekules buvo pasirinkta naudojant SMILES notacija. SMILES, tai supaprastinta molekulinės įvesties linijinės įvedimo sistema. Ši cheminių notacijų kalba buvo sukurta palengvinti vartotojo ir kompiuterio sąsają ir pirmą kartą pristatyta 1988 metais, D. Weiningerio pateiktame straipsnyje [46]. Ji lengvai prieinama ir suprantama chemikams, tačiau tuo pačiu pakankamai lanksti, kad būtų galima interpretuoti ir generuoti cheminius molekulių žymėjimus, nepriklausomai nuo naudojamos kompiuterinės sistemos.

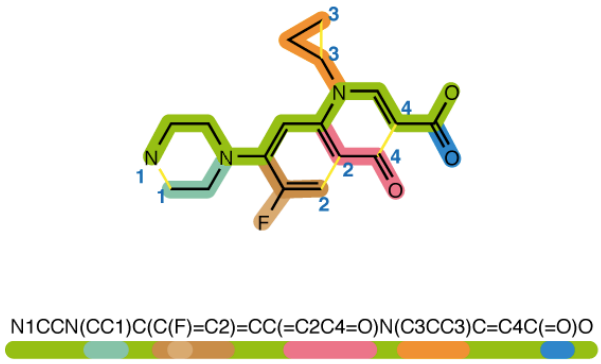
Dabar SMILES yra plačiausiai naudojamas cheminės eilutės žymėjimas, nors buvo pasiūlyta ir kitokių [47]. Kadangi panašiai kaip ir tradicinis cheminis žymėjimas, ji pagreitina įprastinius programinės įrangos metodus, sparčiau ir geriau naudoja kompiuterio pajėgumus [46]. Molekulinės struktūros naudojant SMILES notacijų sistemą unikaliai ir tiksliai apibrėžiamos ir gali būti naudojamos cheminėse duomenų bazėse. Kitas svarbus aspektas, kodėl ši sistema yra tokia populiari, kadangi manoma, kad ji yra geriausias kompromisas tarp žmogaus ir kompiuterio, cheminio žymėjimo aspektu [46]. Žmogus lengvai gali suprati žymėjimą, kadangi yra minimalus skaičius paprastų taisyklių, o molekulinės struktūros linijinė simbolių eilutė panaši į natūralią kalbą (žr. 2 pav.).

Cheminių junginių SMILES žymenys susideda iš atomų, ryšių, skliaustelių ir numerių [49]:

- atomai: Atomai yra atvaizduojami naudojant jų atominius simbolius. Pvz. C anglies, N azoto, arba S sieros. Aromatiniams atomams naudojamos mažosios raidės, kitais atvejais – didžiosios raidės. Atomai, turintys dviejų raidžių simbolius, tokius kaip chloras (Cl) arba bromas

(Br), visada rašomi iš didžiosios raidės, o antrą raidę galima rašyti didžiąja arba mažąja raide. Išskyrus keletą išimčių, vandenilio atomai nėra įtraukti į molekulės eilutę;

- ryšiai: SMILES kalba vartojamos keturios pagrindinės ryšių rūšys: viengubi, dvigubi, trigubi ir aromatiniai ryšiai, žymimi simboliais: „-“, „=“, „#“ ir „:“. SMILES eilutėse paprastai praleidžiama viengubi ir aromatiniai ryšiai. Nepriklausantys pagrindiniams keturiems ryšiams, tai yra joniniai arba atjungiamieji ryšiai reprezentuojami „.“;
- atsišakojimai: atsišakojimai nurodomi pridedant skliaustelius, „(“ ir „)“ bei nurodant šonines struktūras. Atsišakojimai gali ir dažnai turi ir kitų šakų;
- ciklai: ciklinės struktūros yra pateikiamos nutraukiant vieną jungtį kiekviename žiede. Atomai, esantys šalia ryšio, gauna tą patį skaičių. Čia nurodomi numeriai, parodantys ciklines junginio struktūras kaip ciklinius ryšių numerius. Šie cikliniai nuorodų numeriai nėra unikalūs molekulės SMILES vaizde.



2 pav. SMILES notacijos pavyzdys

Apie 90 proc. mokslinių tyrimų naudoja SMILES eilučių reprezentacijas, nors yra ir labiau tinkami formatai, taip yra todėl, kad pasiekiama pusiausvyra tarp tikslumo (atsižvengiant į pateiktų molekulių niuansus) ir sudėtingumo (kompiuteriui lengva interpretuoti duomenis).

Literatūroje galima rasti daug molekulių generavimo pavyzdžių, kuomet naudojama SMILES notacijų sistema [14, 15, 16, 40, 42], todėl šiame darbe ir pasirinkta molekules reprezentuoti būtent SMILES notacijų sistema.

1.9. Baigiamojo projekto temos ir uždavinių pagrindimas

Molekulių generavimas svarbus norint atrasti naujus vaistinius preparatus. Apskirtai, naujų molekulių atradimas gali lemti didelius pokyčius ir naudingas įžvalgas įvairių problemų sprendimui. Naujos molekulės gali būti ieškomos ir rankiniu būdu, tačiau efektyviau pasitelkti į pagalbą kompiuterinius metodus. Pastaruoju metu stipriai išpopuliarėjus dirbtiniams neuroniniams tinklams, tam tikri metodai gali būti pritaikyti ir siekiant atrasti naujas molekules. Šis baigiamasis projektas atliktas bendradarbiaujant su Baltijos pažangių technologijų institutu.

Kadangi variacinis AE yra viena iš populiariausių priemonių naudojamų molekulėms generuoti pastaruoju metu, baigiamajame darbe bus kuriamas variacinis AE bei siekiama

sugeneruoti naujas molekules su norimomis savybėmis. Variacinio AE architektūra parinkta remiantis R. Gomezo-Bombarellio ir kolegų straipsnyje „Automatinis cheminis dizainas, naudojant duomenų srautinį nuolatinį molekulių vaizdavimą“ pateikta struktūra [16]. Kadangi literatūroje nėra visuotinai pritariamo latentinio vektoriaus ilgio, baigiamajame darbe apmokomi keturi variacinio AE modeliai su vienodomis struktūromis, tik skirtingais latentinio vektoriaus ilgiais, siekiant nustatyti variacinio AE tikslumo priklausomybę, nuo latentinio vektoriaus ilgio. Pasirinkti keturi skirtingi latentinio vektoriaus ilgiai: 156 ir 196, pagal R. Gomezo-Bombarellio [16] ir kolegų straipsnyje naudojamus ilgius, taip pat latentinio vektoriaus ilgis, didesnis 30 proc., nei naudojama minėtame straipsnyje – 254 bei 56, pagal T. Blaschkeso su kolegomis atliktą tyrimą [40]. Taip pat apmokomi modeliai, su keliais molekulių ilgių apribojimais, kadangi literatūroje dažniausiai pasirenkamas molekulės ilgis iki 120, išbandomi dar keli molekulių ilgiai: 60, 80, 100 ir tikrinama modelių tikslumo priklausomybė nuo molekulių ilgio, naudojant modelius įvairių molekulių ilgių kodavimui. Modelių tikslumas turi būti tikrinamas keliais aspektais: mokymo ir validacijos tikslumas, tikslumas dekoduojant molekules iš latentinio vektoriaus (1000 molekulių dekoduojama 1000 kartų [16]), validžių molekulių dekodavimas iš latentinės erdvės taškų. Naujų vaistinių molekulių generavimas su norimomis savybėmis atliekamas dviem būdais: naudojantis variaciniu AE bei naudojant interpoliaciją tarp molekulių.

Darbe pateiktas tyrimas yra didesnio projekto dalis, toliau Baltijos pažangių technologijų institute sugeneruotos naujos molekulės bus tiriamos įvairiais aspektas, išgryninamos molekulių savybės. Atlikus kompiuterizuotus tyrimus bei išrinktos tinkamiausias savybes turinčios molekulės bus susintetinamos laboratorijoje bei jų savybės tvirtinamos realiais fiziniais tyrimais. Kuomet gautos savybės bus patvirtintos, naujai sugeneruotos molekulės gali būti panaudotos naujų vaistų kūrime.

2. Metodologija

Šiame skyriuje aprašomas tyrimo vykdymas, duomenys, tyrime naudojami matematiniai metodai. Apžvelgiama tyrime naudojama programinė įranga bei jos ypatybės.

2.1. Duomenys

AE modelio mokymui naudojamos 250 000 atsitiktinai atrinktų SMILES molekulių reprezentacijų, iš ZINC vaistinių molekulių duomenų bazės [39]. Lyginant su literatūroje nagrinėtais tyrimais, tai vidutinis duomenų rinkinio dydis. Kuomet naudojama daugiau duomenų, apmokymo laikas didėja, kai naudojamas mažesnis duomenų kiekis – mažėja tikslumas ir didėja reikiamų mokymo epochų skaičius, tad pasirinktas vidutinio dydžio duomenų rinkinys atitinka poreikius tarp mokymo laiko ilgio bei tikslumo ir epochų skaičiaus. ZINC [39] duomenų bazę sudaro komerciškai prieinami junginiai, skirti struktūriniam virtualiam patikrinimui. Šiuo metu ji turi apie 1 milijardą junginių, kuriuos galima pasiekti per internetinę svetainę. Visa informacija teikiama paruoštuose 3D formatuose, turinčiose biologiškai svarbias molekules. Ji prieinama pogrupiais įvairiems tikslams: bendriesiems, tiksliniams, chemoterapiniams tyrimams bei komerciniams tikslams. Šią duomenų bazę ir paslaugą teikia Kalifornijos San Francisko universiteto Farmacijos chemijos katedros laboratorija [39]. Tyrimo metu, modelių apmokyme 90 proc. duomenų naudojama mokymui, o likusieji 10 proc. – modelio testavimui. Molekulių SMILES eilutės kanonizuojamos naudojant RDKit paketą, kadangi SMILES eilutės notacija nėra unikali. Kanonizuotos SMILES eilutes – tai vienintelė ir unikali molekulės reprezentacija.

Kuomet siekiama generuoti naujas molekules, imamas kitas duomenų rinkinys. Šiuo atveju reikalinga papildoma informacija – molekulės savybės. Pasirinkta molekulių savybė, pagal kurią bus generuojamos naujos molekulės – inhibicijos konstanta (Ki). Inhibicijos konstanta rodo, koks stiprus inhibitorius yra. Kuo mažesnė yra inhibicijos konstantos vertė, tuo mažesnis vaistų kiekis reikalingas fermento aktyvumui slopinti. Todėl tyrime siekiama sugeneruoti molekules, kurių inhibicijos konstantos reikšmė būtų maža. Šiam uždaviniui spręsti naudojamas Binding duomenų bazės molekulių rinkinys. Visų šio duomenų rinkinio molekulės sužymėtos atitinkama Ki reikšme. BindingDB, pradėta internete 2000 metais, yra pirmoji viešai prieinama duomenų bazė, kurią sudaro eksperimentinių baltymų ir mažų molekulių sąveikų duomenys [50]. Ši duomenų kolekcija, kurioje yra daugiau nei milijonas duomenų, visų pirma atsiranda iš mokslinių straipsnių ir vis daugiau iš JAV patentų. BindingDB svetainė yra plačiai ir nuolatinai naudojama maždaug 9000 vartotojų sesijų per mėnesį, o per sesiją žiūrima vidutiniškai šeši puslapiai. Paskelbus, šie duomenys tampa vertingu šaltiniu mokslininkams, studijuojantiems tą pačią makromolekulinę užduotį, taip pat tiems, kurie siekia sukurti geresnius kompiuterinius molekulinio atpažinimo modelius [51]. Bazė sukurta taip, kad palaikytų prieigą prie tikslinių duomenų rinkinių, tokių kaip, su konkrečiu vaistinių objektu susijusių sąveikos duomenų, taip pat plataus masto ir augančio duomenų rinkinio išsamumo analizių [50]. Molekulių generavime naudojamas 25 000 molekulių

rinkinys, kuriame yra ir informacija susijusi su savybėmis. Duomenų rinkinys išfiltruojamas, kad molekulės ilgis būtų ne ilgesnis nei 120.

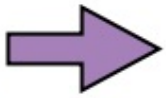
2.2. Vieno vektoriaus kodavimas

Vieno vektoriaus kodavimas yra kategorinių kintamųjų kaip dvejetainių vektorių vaizdavimas. Šio darbo atveju, tekstinės SMILES eilutės pakeičiamos į naujus binarinius požymius, naudojant vieno vektoriaus kodavimą. Vieno vektoriaus kodavimo atveju kiekvienam struktūros vienetui naudojama atskira būsena [52].

Daugelis mašininio mokymosi algoritmų tiesiogiai negali dirbti su kategoriniais duomenimis, todėl kategorijos turi būti konvertuojamos į skaičius, tai reikalinga tiek įvesties, tiek išvesties kintamiesiems. Todėl dėl savo efektyvumo ir paprastumo, vieno vektoriaus kodavimas vis dar yra labiausiai paplitusi procedūra skirta daugiareikšmių klasifikavimo užduočių sprendimui [53].

Koduojant duomenis, pirmiausia reikia, kad kategorinės vertės būtų priskirtos sveikiesiems skaičiams. Tada kiekvienas sveikasis skaičius yra pateikiamas kaip dvejetainis vektorius, kuriame yra visos nulinės vertės, išskyrus sveikojo skaičiaus indeksą, pažymėtą 1. Toliau pateikiamas pavyzdys, kaip gali būti koduojamas vektorius, naudojant vieno vektoriaus kodavimą:

Vartotojo interesai
Technologijos
Mada
Mada
Technologijos
Sportas
Technologijos
Sportas



Interesai: Technologijos	Interesai: Mada	Interesai: Sportas
1	0	0
0	1	0
0	1	0
1	0	0
0	0	1
1	0	0
0	0	1

3 pav. Vieno vektoriaus transformacijos pavyzdys

Kai kuriais atvejais galima tiesiogiai naudoti sveikų skaičių kodavimą, jei reikia, skalę. Tai galimas atvejis, kai tarp kategorijų yra natūralus didėjantis ar mažėjantis ryšys, tuomet jos numeruojamos atitinkamai sveikaisiais skaičiais, pvz. temperatūros būtų numeruojamos: „šalta“ – 0, „šilta“ – 1 ir „karšta“ – 2. Tačiau problema kyla, kai nėra jokių įprastų santykių, tačiau eiliškumas yra svarbus, kaip antai SMILES eilutėse. Taip pat naudojant vieno vektoriaus kodavimą, kitos būsenos ir išvesties logika dažnai yra paprastesnė, todėl reikia mažiau resursų [52].

Kadangi tyrime turimi duomenys yra SMILES eilutės, o kompiuteris ir neuroniniai tinklai negali apdoroti teksto, turimos molekulių išraiškos naudojant vieno vektoriaus kodavimą yra užšifruojamos į matricas. SMILES aprašyta molekulė išskaidoma atskirai po vieną simbolį ir kiekvienam simboliui priskiriamas dvejetainis vektorius, kuriame kiekvienas elementas yra lygus

nuliui, išskyrus vieną vienetą, kuris žymi būtent šį elementą. Toliau, 4 paveiksle pateikiama vienos SMILES eilutės užkoduota matrica, koduojama molekulė – NCC(O)=O.

Molekulė
N
C
C
(
O
)
=
O
...

...	(	)	...	=	...	C	...	N	...	O	...
0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0

4 pav. Molekulės NCC(O)=O kodavimas

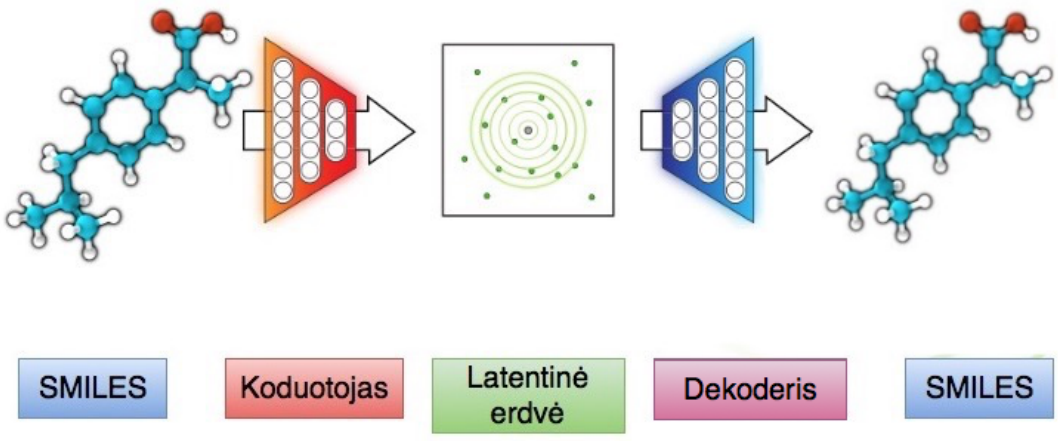
Neuroniniam tinklui reikalingos vienodų dimensijų įvestys, todėl, visų molekulių ilgiai suvienodinami gale pridedant reikalingą kiekį nulinių vektorių.

2.3. Neuroninio tinklo architektūra

Šiame skyrelyje bus pristatyta metodologija, naudota AE architektūroje. Literatūroje galima sutikti daug skirtingų AE architektūros pavyzdžių [14, 15, 40, 42], šie variantai svarbūs, siekiant užtikrinti, kad suspaustas atvaizdavimas atspindėtų prasmingus pradinio duomenų įvesties atributus. Svarbu parinkti tinkamą architektūrą, kad modelis iš tiesų išmoktų prasmingą ir apibendrintą latentinę erdvę.

2.3.1. Autoenkoderis

AE sudarytas iš dviejų dirbtinių giliųjų neuroninių tinklų: koduotojo, skirto kiekvienai sekai paversti fiksuoto dydžio vektoriumi bei dekoderio, skirtu vektorius konvertuoti atgal į sekas (žr. 5 pav.). AE mokomas taip, kad sumažintų klaidą atkuriant pradinę eilutę, t. y., bando išmokti tapatybės funkciją. AE dizaino esmė, tai SMILES eilučių suspaudimas į informacijos „butelio kaklelį.“ Ši fiksuoto ilgio nepertraukiamo vektoriaus kliūtis skatina tinklą išmokti suspaustą molekulės vaizdą, kuris užfiksuoja labiausiai statistiškai svarbią informaciją [16]. Vektoriumi koduota molekulė vadinama latentiniu vektoriumi.



5 pav. Autoenkoderis [16]

Architektūriniu požiūriu, paprasčiausia AE forma yra labai panaši į daugelį vieno sluoksnio perceptronų, kurie sudaro daugiasluoksnį perceptroną, turintį įvesties sluoksnį, išvesties sluoksnį ir vieną ar daugiau paslėptų sluoksnių – tačiau išvesties sluoksnis turi tą patį neuronų skaičių, kaip ir įvesties sluoksnis, ir siekia rekonstruoti savo įvestį. Todėl AE iš esmės yra priskiriami prie neprižiūrimų mašininio mokymosi metodų. Jei koderį pažymime p , o dekoderį – q, tai visas AE procesas gali būti aprašytas taip [54]:

$$\begin{aligned} p&: \mathrm{X} \to \mathcal{F} \\ q&: \mathcal{F} \to \mathrm{X} \\ p, q &= \arg\min_{\phi,\varphi} \|\mathrm{X} - (\varphi \circ \phi)\mathrm{X}\|^2 \end{aligned} \tag{2}$$

Paprasčiausiu atveju, kuomet AE sudaro vienas paslėptas sluoksnis, koduotojas priima įvestį $x \in R^d = \mathrm{X}$ ir užfiksuoja jį į $z \in R^p = \mathcal{F}$:

$$z = \sigma(Wx + b) \tag{3}$$

Šis vaizdas z – latentinis vektorius, o σ yra aktyvacijos funkcija, pvz., logistinė sigmoidinė ar hiperbolinio tangento funkcija, W yra svorio matrica, o b yra poslinkio vektorius. Toliau dekoderio fazėje, nuo latentinio vektoriaus z pereinama prie rekonstruoto x', tokių pačių dimensijų, kaip ir x:

$$x' = \sigma'(W'z + b'), \tag{4}$$

kur σ', W' ir b' žymi tuos pačius objektus, tačiau dažnu atveju skiriasi nuo naudojamų kodavime, visa tai priklauso nuo pasirinktos AE architektūros. Atliekant šias užduotis AE siekia sumažinti rekonstravimo klaidą.

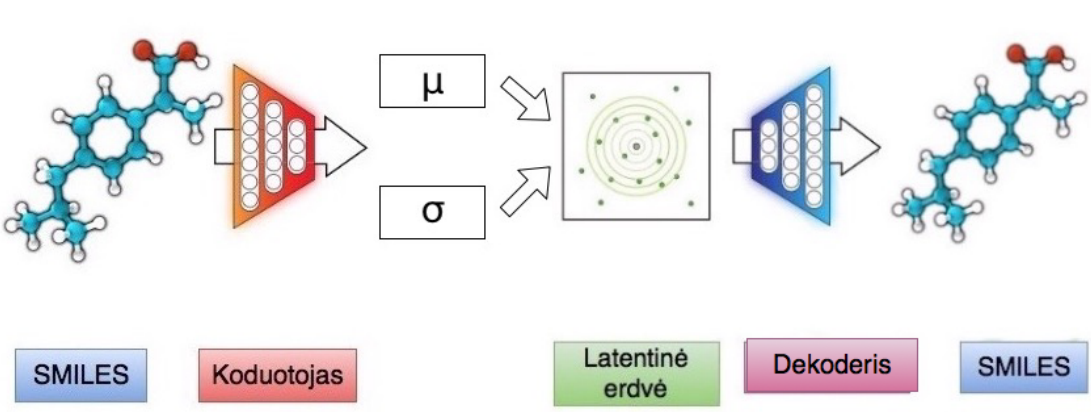
Norint, kad latentinėje erdvėje veiktų nesuvaržytas optimizavimas, latentinės erdvės taškai turi būti dekoduoti į galiojančias SMILES eilutes, kurios užfiksuoja mokymo duomenų cheminį pobūdį [16]. Be šio suvaržymo latentinė erdvė, kurią išmoko AE, gali būti reta ir turėti didelių „negyvų vietų“, kurios dekoduoja netinkamas SMILES eilutes. Siekdami užtikrinti, kad latentinėje erdvėje esantys taškai atitiktų galiojančias realistines molekules, pasirenkama naudoti variacinį AE [16].

2.3.2. Variacinis autoenkoderis

Variacinis AE apibendrina AE pridedant stochastiškumo dekoderiams, kurie kartu su bausmės sąlyga paskatina, jog visi latentinės erdvės taškai būtų iškoduoti į galiojančias išeitis. Iš esmės, pridėjus triukšmo koduotoms molekulėms, dekoderis verčiamas išmokti iššifruoti įvairesnius latentinius taškus ir surasti patikimesnius vaizdus [16].

Variacinis AE modifikuoja AE architektūrą, pakeisdama deterministinę funkciją ϕ su išmoktu aposterioriniu atpažinimo modeliu $q(z|x)$. Šis modelis parodo aposteriorinio diagonalinio Gauso pasiskirstymo apytikslį poslinkį per z su neuroniniu tinklu, kuris yra sąlygotas x. Intuityviai, variacinis AE mokosi reprezentacijas ne kaip atskirus taškus, bet kaip plastiškus

elipsoidinius regionus latentinėje erdvėje, verčiant vektorius užpildyti erdvę, o ne įsiminti mokymo duomenis kaip izoliuotas reprezentacijas.



6 pav. Variacinis autoenkoderis [16]

Jei variacinis AE būtų apmokytas naudojant standartinio AE rekonstrukcijos tikslo funkciją, jis išmoktų koduoti savo įeitis deterministiškai, o $q(z|x)$ dispersija taptų išnykstamai maža. Vietoj to, variacinis AE naudoja tikslo funkciją, kuri skatina modelį išlaikyti savo aposteriorinį pasiskirstymą netoli ankstesnio $p(z)$, paprastai standartinio Gauso ($\mu = 0, \sigma = 1$). Be to, šios tikslo funkcijos apatinė riba validi bei susijusi su tikrąja duomenų logaritmine tikimybe, todėl modelis yra generacinis. Ši tikslo funkcija aprašoma taip [55]:

$$\mathcal{L}(\theta; x) = -KL(q_\theta(z|x)||p(z)) + \mathbb{E}_{q_\theta(z|x)}\big[\log\big(p_\theta(x|z)\big)\big] \leq \log\big(p(x)\big). \qquad (5)$$

Pirmasis lygties terminas yra reguliatorius, kurį mes įtraukiame į tikslo funkciją. Tai yra *Kullback-Leibler* skirtumas tarp koduotojo pasiskirstymo $q_\theta(z|x)$ ir $p(z)$. Šis skirtumas matuoja, kiek informacijos prarandama, kai naudojama q reprezentuoti p, galima sakyti, p ir q artumo matas.

Antrasis terminas yra rekonstrukcijos praradimas arba tikėtina logaritminė tikimybė. Skaičiuojama vidutinė reikšmė, atsižvelgiant į koduotojo pasiskirstymą pagal reprezentaciją. Šis terminas skatina dekoderį išmokti rekonstruoti duomenis. Jei dekoderio išvesties duomenys nėra gerai atkurti, statistiškai, dekoderis keičia logaritminės tikimybės pasiskirstymo parametrus, kadangi tikimybėmis neatspindimi tikrieji duomenys. Prasta rekonstrukcija sukels didelį šios nuostolių funkcijos pokytį.

Jei koduotojo išvestys z labai skiriasi nuo standartinio normaliojo pasiskirstymo, tuomet tikslo funkcija papildoma bauda. Šis reguliatoriaus terminas padeda saugoti kiekvienos įvesties z reprezentacijas pakankamai skirtingas, tačiau ir toje pačioje latentinėje erdvėje. Jei nebūtų reguliatoriaus, koderis galėtų išmokti apgauti ir kiekvieną duomenų tašką reprezentuoti kitame Euklido erdvės regione. Tai būtų blogai, kadangi du to paties objekto vaizdai galėtų būti labai skirtingi. Norima, kad z reprezentacinė erdvė būtų prasminga, todėl į tikslo funkciją įtraukiamas

ir baudos faktorius. Taip pat tai lemia, jog panašūs objektai reprezentuojami šalia latentinėje erdvėje.

2.3.3. Konvoliucinis sluoksnis

Konvoluciniai neuroniniai tinklai dažniausiai naudojami vaizdo apdorojimui, bet taip pat gali būti naudojami kitiems įvesties tipams, pavyzdžiui garsui, ar kitiems didelių dimensijų objektams. Šie neuroniniai tinklai buvo įkvėpti 1986 metais atlikto tyrimo, kuriame buvo nagrinėjamas vaizdo apdorojimas katės smegenyse. Buvo atrasta, kad egzistuoja persiklojantys regionai, padengiantys visą regėjimo lauką, kurie veikia kaip filtrai, kurie apdoroja paveikslėlius, o vėliau informacija perduodama į kitus sluoksnius. Įspūdingas konvoliucinių neuroninių tinklų veikimas vaizdo atpažinimo užduotyse rodo, kad jie puikiai tinka mokytis ir iš kitų tipų erdvinių duomenų, tokių kaip molekulės struktūros. Skirtingai nuo ankstesnių mašininio mokymosi metodų, konvoliucinių neuroninių tinklų taškų skaičiavimo metodas nereikalauja išgauti savybių iš struktūros. Vietoj to, metodas automatiškai identifikuoja labiausiai informatyvias funkcijas, reikalingas sėkmingam taškų skaičiavimui [56]. Pavadinimas konvoliuciniai neuroniniai tinklai rodo, kad tinklas naudoja matematinę operaciją, vadinamą konvoliucija – specializuota linijinės operacijos rūšis [23]. Konvoliucijos funkcija, kuomet laiko indeksas laikomas sveikaisiais skaičiais, gali būti aprašyta:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a), \tag{6}$$

kur t – laiko indeksas, w – svorių funkcija, x – funkcija, o a – laiko pokyčio matmuo.

Kuomet įvesties objekto dimensija siekia 200 x 200, nenorima, jog sluoksnis būtų 40 000 neuronų. Tokiu atveju kuriamas, pavyzdžiui, 20 x 20 skenavimo įvesties sluoksnis – filtras (taip pat gali būti vadinamas branduoliu), kuriam pateikiama pirmieji 20 x 20 elementų (paprastai prasideda viršutiniame kairiajame kampe) [57]. Šis filtras taip pat yra skaičių masyvas, kurio elementai yra neuroninio sluoksnio svoriai. Kuomet filtras stumiamas per objektą, jis daugina svorius su pradinio objekto taškų reikšmėmis – ši operacija gali būti interpretuojama kaip daugyba panariui. Visi šie skaičiavimai apibendrinami ir įrašomi į išeities matricos pirmąjį elementą, kuris yra filtro viršutinio kairiojo objekto kampo reprezentacija. Toliau šis procesas kartojamas kiekvienai įvesties apimties vietai, filtrą slenkant į dešinę pusę. Kiekviena unikali įvesties apimties vieta sukuria naują elementą išvesties matricoje, kuri vadinama aktyvinimo žemėlapiu arba funkcijų žemėlapiu. Klasikiniuose konvoliuciniuose neuroniniuose tinkluose atgalinio sklidimo algoritmas paprastai atnaujina branduolio svorius, kad kiekvienam funkcijų matmeniui būtų galima nustatyti ryšį tarp kaimyninių neuronų. Tada jis apibendrina visų filtrų signalus, kad sukurtų paslėptų sluoksnių aktyvavimą [58].

Literatūroje galima aptikti daug mokslinių tyrimų, kurių metu naudojami konvoliuciniai neuroniniai tinklai, taip pat šis metodas labai populiarus ir AE struktūrose, bei moksliniuose straipsniuose, kuriuose pristatomas naujų molekulių generavimas [14, 15, 16].

2.3.4. Hiperbolinio tangento aktyvacijos funkcija

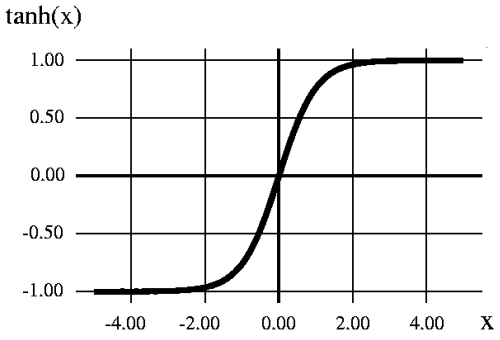
Konstruojant dirbtinius neuroninius tinklus, vienas iš svarbiausių svarstymų yra diferencijuojamų paslėptų ir išvesties sluoksnių aktyvacijos funkcijos pasirinkimas. Taip yra todėl, nes apskaičiuojant atgalinės sklaidos vertės klaidos signalą, naudojamą nustatant dirbtinių neuroninių tinklų parametrų atnaujinimą, reikia nustatyti aktyvavimo funkcijos gradientą. Dažniausiai naudojamos trys aktyvavimo funkcijos: sureguliuoti linijiniai vienetai (angl. *rectified linear units*), logistinė sigmoidinė bei hiperbolinio tangento funkcijos. Paslėptų sluoksnių projektavimas bei čia svarbi aktyvacijos funkcija yra labai plačiai nagrinėjama mokslinių tyrimų sritis ir dar neturi daug galutinių teorinių principų. Prieš įvedant sureguliuotus linijinius vienetus, dauguma dirbtinių neuronų tinklų naudojo logistinės sigmoidės ir hiperbolinio tangento aktyvinimo funkcijas [23].

Hiperbolinio tangento funkcija išreiškiama taip:

$$g(z) = \tanh(z) = \frac{\sinh(z)}{\cosh(z)} = 2\sigma(2z) - 1 = \frac{e^z - e^{-z}}{e^z + e^{-z}}, \tag{7}$$

kur $\sigma(z)$ – logistinė sigmoidinė funkcija.

Kaip ir logistinė sigmoidė, hiperbolinio tangento funkcija taip pat yra sigmoidinė („s“ formos), tačiau leidžia rezultatams kisti intervale $(-1, 1)$. Taigi, stipriai neigiamoms įvesties objekto reikšmėms bus priskirtas neigimas rezultatas. Be to, tik nulinės įvesties reikšmės priskiriamos prie nulinių rezultatų. Dėl šių priežasčių dirbtinis neuroninis tinklas, mokymosi metu, mažiau linkęs „užstrigti“ ir toliau nebesimokyti.



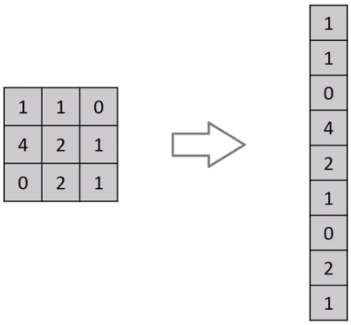
7 pav. Hiperbolinio tangento aktyvacijos funkcija

Kuomet dirbtinių neuroninių tinklų architektūroje riekia naudoti sigmoidinę aktyvinimo funkciją, hiperbolinio tangento funkcija paprastai veikia geriau nei logistinė sigmoidinė funkcija, taip pat hiperbolinio tangento funkcija panašesnė į tapatybės funkciją. Kuomet įvesties objekto reikšmės mažos hiperbolinio tangento aktyvacijos funkcija elgiasi panašiai kaip ir identifikacijos funkcija, tai palengvina giliųjų tinklų mokymąsi ir padaro panašų į tiesinį modelį [23].

2.3.5. Suliejimas

Pertvarkymo operacijos yra vienas svarbiausių objektų operacijų tipas. Taip yra todėl, nes objekto forma suteikia mums kažką konkretaus, ką galime panaudoti, norėdami formuoti intuiciją apie objektą.

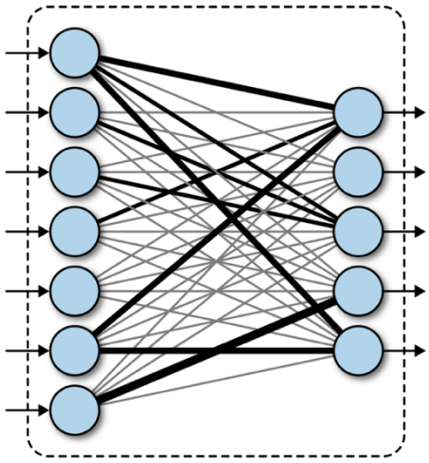
Objekto suliejimo (angl. *flatten*) metu pašalinami visi matmenys, išskyrus vieną, iš esmės pereiname nuo dviejų dimensijų prie vienos. Po konvoliucijos sluoksnio dažniausiai turimas pilnai sujungtas neuroninis sluoksnis, kuris gali priimti tik vieno matmens duomenis. Kai po konvoliucijos sluoksnio gaunama bendra išeities matrica, kitas žingsnis yra ją sulieti į vieną vektorių, kuris vėliau perduodamas neuroniniam tinklui tolimesniam apdorojimui.



8 pav. Suliejimo pavyzdys

2.3.6. Pilnai sujungtas sluoksnis

Pilnai sujungtas sluoksnis (angl. *dense* arba *fully connected layer*) tai neuroninio tinklo architektūros dalis, kuriame visi neuronai viename sluoksnyje yra sujungti su visais kito sluoksnio neuronais, įskaitant įvesties ir išvesties sluoksnius. Tokie sluoksniai sudaro etaloninio daugiasluoksnio perceptrono neuroninio tinklo architektūrą. Pilnai sujungtą sluoksnį galima laikyti funkcija, kuri pereina nuo $\mathbb{R}^m$ dimensijos iki $\mathbb{R}^n$. Reikia paminėti, kad kiekvienas išeities matmuo priklauso nuo kiekvieno įvesties matmens.



9 pav. Pilnai sujungtas neuroninio tinklo sluoksnis

Laikant, kad $x \in \mathbb{R}^m$ yra visiškai sujungto sluoksnio įvestis, tuomet $y_i \in \mathbb{R}$ i-toji išvestis apskaičiuojama taip:

$$y_i = \sigma(w_1 x_1 + \cdots + w_m x_m), \tag{8}$$

kur σ yra netiesinė funkcija (dažnai logaritmine sigmoidinė, hiperbolinio tangento ar panašiai), o w yra neuroninio tinklo mokomieji svoriai. Tuomet pilna y išvestis gali būti aprašoma taip:

$$y = \begin{pmatrix} \sigma\left(w_{1,1} x_1 + \cdots + w_{1,m} x_m\right) \\ \vdots \\ \sigma\left(w_{n,1} x_1 + \cdots + w_{n,m} x_m\right) \end{pmatrix} \tag{9}$$

Šią procedūrą galima suvokti, kaip kelių vektorių daugybą panariui, todėl skaičiuojant neuroninio tinklo svorius, efektyvumo tikslais, y apskaičiuojama kaip x vektoriaus bei svorių matricos W daugyba:

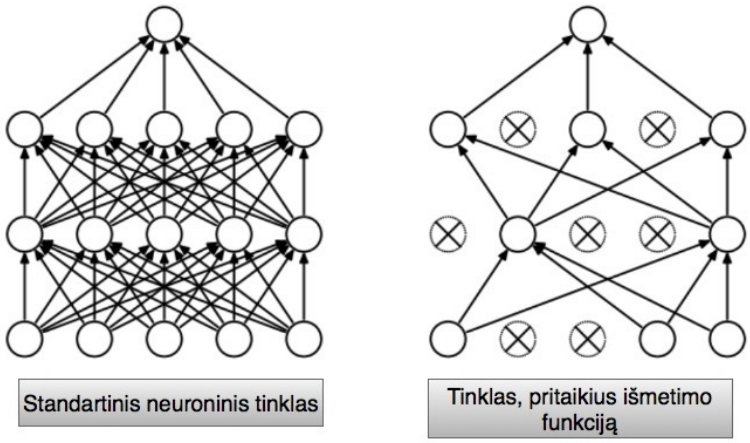
$$y = \sigma(W \circ x), \tag{10}$$

kur σ – netiesinė aktyvacijos funkcija.

Pilnai sujungtas sluoksnis visuomet naudojamas po konvoliucinio sluoksnio, keičiant dimensiją bei dažnu atveju jame atliekamas klasifikavimas ar pagrindinė neuroninio tinklo užduotis.

2.3.7. Išmetimo technika

Gilūs neuroniniai tinklai, turintys daug parametrų, yra labai galingos mašininio mokymosi sistemos, tačiau persimokymas didelė šių tinklų problema, todėl 2014 metais N. Srivastavas su kolegomis straipsnyje „Išmetimas: paprastas būdas apsaugoti neuroninius tinklus nuo persimokymo" pateikė išmetimo techniką (angl. *dropout*), kuri padeda susidoroti su dirbtinio neuroninio tinklo persimokymu [59].



10 pav. Standartinis neuroninis tinklas bei tinklas, pritaikius išmetimo funkciją [60]

Išmetimo technika, tai neuronų (paslėptų ir matomų) išjungimas neuroniniame tinkle. Išmesti neuroną, reiškia laikiną jo pašalinimą iš tinklo, kartu su visais įeinančiais ir išeinančiais ryšiais (žr. 10 pav.). Pasirinkimas, kuriuos neuronus išmesti mokymo metu yra atsitiktinis. Paprasčiausiu atveju kiekvienas neuronas yra išsaugotas su fiksuota tikimybe p, nepriklausoma nuo kitų neuronų, kur p gali būti pasirinkta naudojant patvirtinimo rinkinį arba tiesiog gali būti nustatyta 0.5, tai yra optimalu daugeliui dirbtinių neuroninių tinklų ir užduočių [23]. Kadangi naudojant šią techniką

yra mažinamas neuroninio tinklo sudėtingumas, mažinamas ryšių tarp neuronų kiekis kiekvienai iteracijai, todėl neuronai negali būti maksimaliai priklausomi nuo kažkurio tam tikro, išskirtinio neurono. Išmetimo technika yra naudojama tik mokymo metu – ši technika nėra naudojama vėlesniame neuroninio tinklo eksploatavime ar testavimo etape.

Išmetimo technikos motyvacija pagrįsta lyties vaidmens evoliucijoje teorija. Reprodukcijos metu įtraukiami pusė vieno iš tėvų genų bei pusė kito, pridedant labai nedidelį atsitiktinės mutacijos kiekį ir juos derinant palikuonyje, esmė yra sukurti palikuonį, turintį šiek tiek mutavusių tėvų genų kopiją [59].

Išmetimo technika žymiai pagerino standartinių neuroninių tinklų veikimą, turint įvairias architektūras bei duomenų rinkinius. Vienas iš didžiausių išmetimo technikos trūkumų yra tas, jog šios operacijos metu didinamas mokymo laikas. Tinklo su išmetimo technika apmokymas paprastai trunka 2–3 kartus ilgiau, nei standartinis apmokymas su tokia pačia struktūra. Pagrindinė šio padidėjimo priežastis yra ta, kad parametrų atnaujinimai yra labai triukšmingi. Kiekvienas apmokymo atvejis efektyviai bando mokyti kitą atsitiktinę architektūrą. Todėl skaičiuojami gradientai nėra galutinės architektūros gradientai, kurie bus naudojami testavimo metu. Todėl nenuostabu, kad mokymas trunka ilgai. Tačiau pagrįsta, jog šis stochastiškumas padeda neuroniniam tinklui nepersimokyti [59].

2.3.8. Uždari rekurentiniai vienetai

K. Cho su kolegomis 2014 metais pasiūlė uždarus rekurentinius vienetus (angl. *gated recurrent unit*s, GRU), kad kiekvienas rekurentinis vienetas užfiksuotų skirtingų laiko skalių priklausomybę. GRU turi blokavimo įrenginius, kurie moduliuoja informacijos srautą įrenginio viduje, tačiau neturi atskirų atminties elementų [60]. Pirmiausia aprašoma rekurentiniai neuroniniai tinklai – GRU pagrindas.

Rekurentiniai neuroniniai tinklai

Rekurentiniai neuroniniai tinklai (angl. *recurrent neural networks*) – tai dirbtiniai neuroniniai tinklai, susidedantys iš paslėptos būsenos h ir pasirinktinio išėjimo y, veikiančio kintamo ilgio seką $x = (x_1, \dots, x_T)$. Kiekvienu laiko momentu t, paslėpta rekurentinio neuroninio tinklo būsena $h_{\langle t \rangle}$ atnaujinama taip [61]:

$$h_{\langle t \rangle} = f\left(h_{\langle t-1 \rangle}, x_t\right), \tag{11}$$

kur f yra netiesinė aktyvacijos funkcija. Aktyvacijos funkcija gali būti tokia paprasta kaip logistinė sigmoidinė funkcija panariui ar sudėtinga kaip ilgos trumpalaikės atminties (LSTM) vienetas. Rekurentiniai neuroniniai tinklai gali išmokti tikimybinį pasiskirstymą, išmokstant numatyti kitą simbolį sekoje. Tokiu atveju kiekvienas rezultatas laiko momentu t yra sąlyginis pasiskirstymas $p(x_t|x_{t-1}, \dots, x_1)$. Pavyzdžiui, polinominį pasiskirstymą galima gauti naudojant „softmax“ aktyvacijos funkciją [61]:

$$p(x_t = 1|x_{t-1}, \dots, x_1) = \frac{\exp(w_j h_{\langle t \rangle})}{\sum_{j'=1}^{K} \exp(w_{j'} h_{\langle t \rangle})}, \quad (12)$$

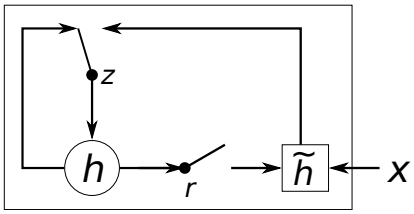
visiems galimiems simboliams j = 1, … , K, kur w_j yra svorio matricos W eilutės. Derinant šias tikimybes, galime apskaičiuoti sekos x tikimybę naudojant [61]:

$$p(x) = \prod_{t=1}^{T} p(x_t|x_{t-1}, \dots, x_1). \quad (13)$$

Iš šio išmokto pasiskirstymo galima gauti naują seką iteratyviai imant simbolį kiekvienu laiko momentu [61].

Uždari rekurentiniai vienetai

Toliau pateikiama GRU reprezentacija, pagal tai, kaip buvo pristatyta pirmąjį kartą, K. Cho ir kolegų straipsnyje „Frazių reprezentacijų mokymasis naudojant rekurentinių neuroninių tinklų koduotoją – dekoderį statistiniam mašinų vertimui“ [61]. Uždaro rekurentinio vieneto vizualizacija pateikiama 11 paveiksle.



11 pav. Uždaras rekurentinis vienetas [61]

Pirmiausia aprašoma, kaip apskaičiuojama j-ojo paslėpto vieneto aktyvacija. Pirma, atstatymo vartai r_j apskaičiuojami taip [61]:

$$r_j = \sigma\left([W_r x]_j + \left[U_r h_{\langle t-1 \rangle}\right]_j\right), \quad (14)$$

kur σ yra logistinė sigmoidinė funkcija, o $[.]_j$ žymi vektoriaus j-tąjį elementą, x – įvesties objektas, $h_{\langle t-1 \rangle}$ ankstesnė paslėpta būsena, W_r ir U_r yra jau išmoktos svorių matricos.

Panašiai, atnaujinimo vartai z_j apskaičiuojami taip [61]:

$$r_j = \sigma\left([W_z x]_j + \left[U_z h_{\langle t-1 \rangle}\right]_j\right). \quad (15)$$

Tada faktinė siūlomo vieneto $h_{\langle t \rangle}$ aktyvacija apskaičiuojama:

$$h_j^{\langle t \rangle} = z_j h_j^{\langle t-1 \rangle} + (1 - z_j)\tilde{h}_j^{\langle t \rangle}, \quad (16)$$

kur

$$\tilde{h}_j^{\langle t \rangle} = \phi\left([Wx]_j + \left[U(r \odot h_{\langle t-1 \rangle})\right]_j\right). \quad (17)$$

Šioje formuluotėje, kai atstatymo vartai yra arti 0, paslėpta būsena yra priversta ignoruoti ankstesnę paslėptą būseną ir iš naujo nustatyti tik dabartinę įvestį. Tai veiksmingai leidžia paslėptai būsenai atsisakyti bet kokios informacijos, kuri, kaip nustatyta, ateityje yra nereikšminga, tokiu

būdu suteikiant galimybę į kompaktiškesnį vaizdą. Kita vertus, atnaujinimo vartai kontroliuoja, kiek informacijos iš ankstesnės paslėptos būsenos bus perkelta į dabartinę paslėptą būseną.

Kadangi kiekvienas paslėptas vienetas turi atskirus atstatymo ir atnaujinimo vartus, kiekvienas paslėptas įrenginys išmoks užfiksuoti priklausomybes skirtingais laiko momentais. Tie vienetai, kurie mokosi užfiksuoti trumpalaikes priklausomybes, linkę turėti atstatymo vartus, kurie dažniau būna aktyvūs. Priešingai, tie vienetai kurie mokosi užfiksuoti ilgalaikes priklausomybes turės dažniau aktyvius atnaujinimo vartus [61].

2.4. Baigiamojo projekto įgyvendinimas

Projekto įgyvendinimui pasirinkta Python programavimo kalba. Python yra interpretuojama, interaktyvi programavimo kalba sukurta Guido van Rossumo 1990 metais [62]. Python kūrėjų tikslas buvo sukurti kalbą, kuri yra lengvai skaitoma, išraiškinga bei paprasta. Python programavimo kalba kuriama kaip atviro kodo projektas, todėl nauji algoritmai realizuojami greitai, sukurta daug įvairių bibliotekų bei paketų padedančių spręsti didelį spektrą problemų.

Dabar Python yra viena populiariausių programavimo kalbų, naudojama mašininio mokymo srityje. Darbe AE įgyvendinimui, naudojama Keras biblioteka. Keras [63] yra aukšto lygio neuroninių tinklų programavimo sąsaja, parašyta Python programavimo kalba ir galinti veikti naudojantis vienu iš populiariausių karkasų TensorFlow [64]. Ji buvo sukurta sutelkiant dėmesį į greitą eksperimentavimą, galimybė pereiti nuo idėjos prie rezultato su kuo mažesniu atotrūkiu yra raktas į gerus tyrimus. Didžiausias Keras bibliotekos privalumas, jog ji leidžia lengvai ir greitai atlikti prototipų kūrimą. Taip pat naudojantis TensorFlow karkasu galima naudoti NVIDIA kurtą cuDNN biblioteką, kurios dėka yra įgalinamos efektyvios konvoliucijos bei telkimo operacijos, kurios yra kritiškai svarbios tinklo mokymui. Tokiu būdu, programos veikia daug greičiau naudojantis NVIDIA grafikos apdorojimo įrenginiu. Vizualizavimui naudojama viena iš populiariausių bibliotekų – Matplotlib [65], duomenų įkėlimui, apdorojimui, optimizavimui naudojami Pandas [66], NumPy [67], SciPy [68] paketai. Molekulių validacijai ir vizualizacijai buvo naudojama cheminės informatikos programinė įranga RDKit [69] bei MolVS [70]. Neuroninių tinklų mokymas vyko atgaliniu klaidos sklidimo algoritmu, naudojant Adam optimizavimo algoritmą.

Programos realizavimui buvo naudojamas Lenovo Legion Y520 nešiojamas kompiuteris (FHD-IPS AG i7-7700HQ/16G/PCIe SSD 256GB su 1 TB HDD, vaizdo plokštė GeForce GTX1060_6GB)

3. Tyrimas

Šioje projekto dalyje pateikiama projekto eiga, atliktų tyrimų rezultatai bei jų interpretacija. Pateikiamos gautos išvados ir apibendrinimai.

3.1. Projekto eiga

1. Parsisiunčiama SMILES molekulių duomenų bazė. Duomenys minimaliai turi susidėti iš tokių stulpelių: molekulės pavadinimas, SMILES eilutė, parametras (inhibicijos konstanta).

2. Molekulės apibūdinimas SMILES koduojamas naudojant vieno vektoriaus kodavimą. Molekulių ilgiai suvienodinami. Gaunamos matricos, toliau naudojamos variacinio AE įvesčiai.

3. Kadangi duomenų kiekis yra didelis, o algoritmai skaitantys tekstinį formatą veikia lėtai, visa duomenų bazė saugoma kaip binariniai vektoriai .h5 failo formate.

4. Sudaroma variacinio AE struktūra. Variacinis AE apmokomas naudojant jau užkoduotas SMILES eilutes. Išbandomi kelios variacinio AE parametrų reikšmės.

4.1. Apmokomas variacinis AE, SMILES ilgis suvienodinamas. Molekulių ilgis iki 120 simbolių.

4.2. Apmokomas variacinis AE su skirtingais molekulių ilgiais. Pasirinkti maksimalūs molekulių ilgiai : 60, 80, 100, 120. Modeliams priskiriami pavadinimai atitinkamai: M60, M80, M100, M120.

4.3. Apmokomas variacinis AE su skirtingais latentinio vektoriaus ilgiais. Pasirinkti latentinio vektoriaus ilgiai: 56, 156, 196, 254. Modeliams priskiriami pavadinimai, atitinkamai: L56, L156, L196, L254.

5. Patikrinamas variacinio AE gebėjimas užkoduoti bei iškoduoti SMILES eilutę.

6. Generuojamos naujos molekulės, naudojant variacinį AE. Patikrinamas molekulių validumas. Tikrinama ar sugeneruotos vienodos molekulės, naudojantis skirtingais modeliais.

7. Naudojant latentinius vektorius atliekama interpoliacija tarp dviejų vektorių.

8. Latentiniai vektoriai perleidžiami per variacinį AE dalį – dekoderį. Gaunamos vieno vektoriaus kodavimo matricos.

9. Iš vieno vektoriaus kodavimo matricos iškoduojamos į molekulių reprezentaciją – SMILES eilutes.

10. Naudojant cheminės informatikos paketus Python programinėje įrangoje, patikrinama, kurios molekulės validžios.

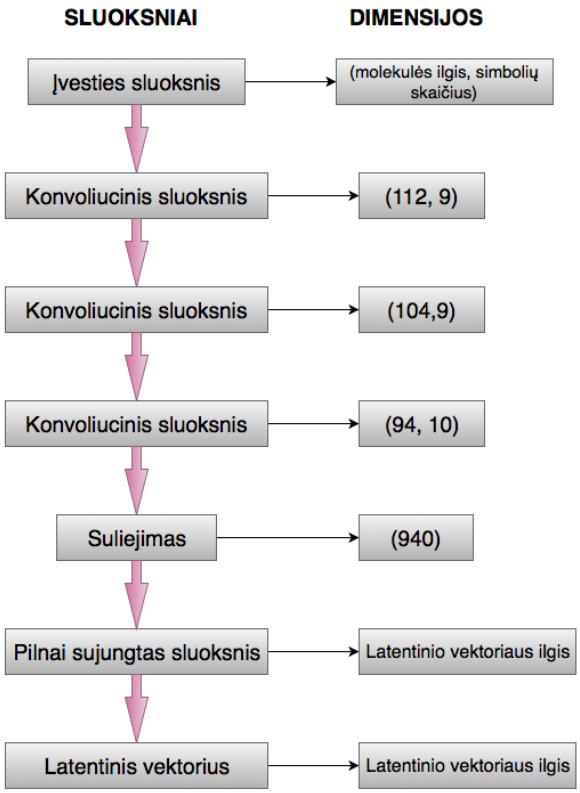
3.2. Variacinio autoenkoderio architektūra

Šiame skyrelyje bus aptarta tyrime naudojamo variacinio AE architektūra. Pasirinkta architektūra remiasi literatūros apžvalgoje aprašytame R. Gomezo-Bombarellio ir kolegų straipsnyje „Automatinis cheminis dizainas, naudojant duomenų srautinį nuolatinį molekulių vaizdavimą“ naudojama struktūra [16].

3.2.1. Koderio architektūra

Koderio architektūra – svarbus aspektas, siekiant kokybiškai išgauti latentinę erdvę. Šiuo atveju, koderio įvestis yra molekulės SMILES reprezentacija, jau užkoduota vieno vektoriaus kodavimu, todėl įvestį sudaro matrica, kurios dydis yra molekulės ilgis bei simbolių imties dydis. Viso tyrimo metu pasirinkta molekules riboti 120 simbolių, jei molekulė yra trumpesnė, nei 120 simbolių, matrica papildoma nuliniais vektoriais. Simbolių imtis sudaroma pagal turimus duomenis bei SMILES eilutėse naudojamus simbolius, šioje vietoje pasirinkta sudarinėti simbolių imtį, o ne naudoti visus galimus SMILES eilučių simbolius tam, kad sumažinti įvesties matricą, taip sumažinant mokymo, kodavimo ir dekodavimo laiką.

Toliau koderį sudaro trys konvoliuciniai sluoksniai, kuriuose naudojamas 8×8 filtras. Kiekviename sluoksnyje taip pat naudojama hiperbolinio tangento aktyvacijos funkcija. Kiekvieno išeities sluoksnio dimensijos pavaizduotos 12 paveiksle. Konvoliucinis sluoksnis atlieka informacijos ištraukimo funkciją. Tačiau rezultatas, po šių sluoksnių yra matrica, todėl tolimesnis sluoksnis, einant link latentinės erdvės, yra suliejimas. Šiame sluoksnyje iš matricos perdaromas vektorius, todėl nėra nežinomų parametrų.



12 pav. Koderio architektūra

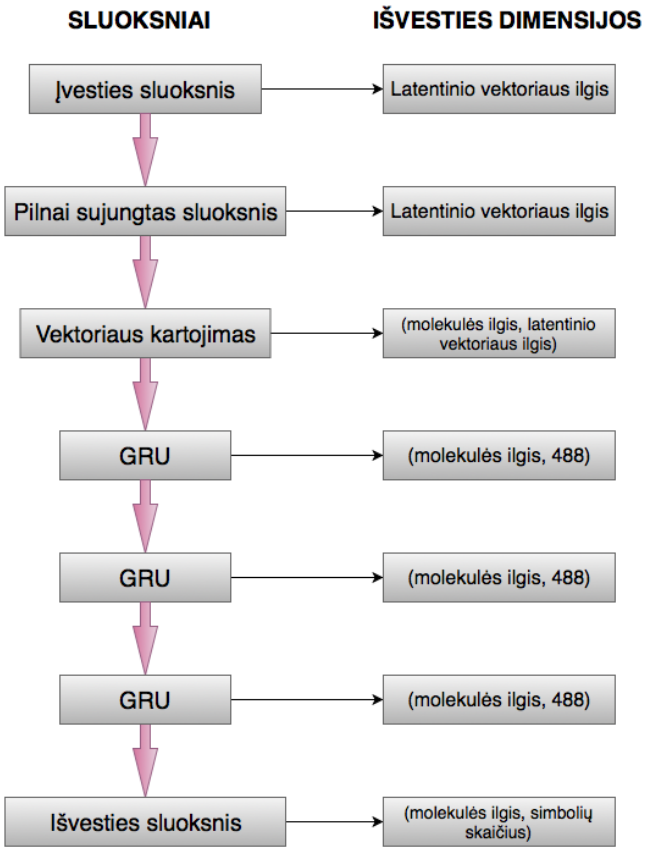
Tradiciškai, po konvoliucinio sluoksnio seka pilnai sujungtas sluoksnis, kuris susistemina išgautą informaciją. Pilnai sujungtame sluoksnyje yra latentinio vektoriaus ilgio neuronų skaičius, tad šis sluoksnis skiriasi, priklausomai nuo modelio. Taip pat pilnai sujungtame sluoksnyje

naudojama išmetimo technika, jog koderis nepersimokytų. Po šio sluoksnio seka latentinis vektorius, kuris yra vidutinis pasiskirstymo ėminys.

3.2.2. Dekoderio architektūra

Dekoderio tikslas – iškoduoti molekulę iš latentinio vektoriaus. Šiuo variacinio AE atveju, dekoderis yra tikimybinis. Dekoderio įvesties sluoksnis yra latentinis vektorius, todėl ši įvestis priklauso nuo modelio bei pasirinkto latentinio vektoriaus ilgio.

Dažnu atveju, dekoderio architektūra gali būti pasirinkta simetriška koderio topologijai, tačiau sekant R. Gomezo-Bombarellio ir kolegų straipsnyje aprašytą architektūrą, dekoderyje naudojami ne konvoliuciniai sluoksniai, o GRU [16]. Kiekvieno išeities sluoksnio dimensijos pavaizduotos 13 paveiksle. Po įvesties sluoksnio, naudojamas pilnai sujungtas sluoksnis, kurį sudaro tiek neuronų, koks yra latentinio vektoriaus ilgis. Šiame sluoksnyje, taip pat kaip ir koderyje, naudojama išmetimo technika, dėl analogiškos priežasties, kad dekoderis nepersimokytų.



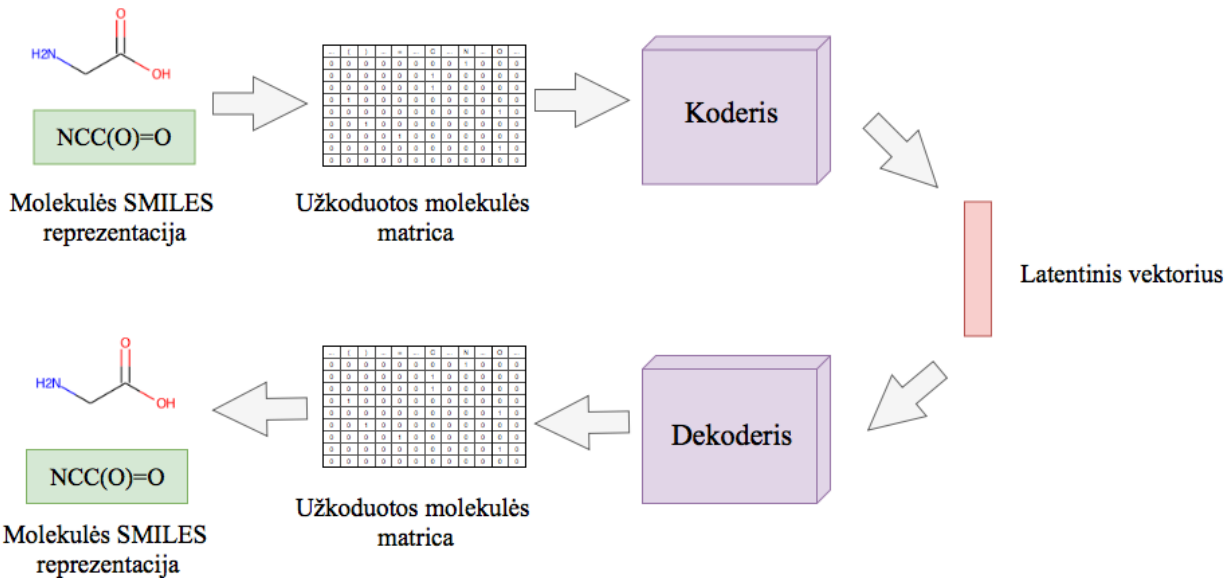
13 pav. Dekoderio architektūra

Toliau seka trys GRU sluoksniai, kuriuos kiekvieną sudaro 488 blokai. Šiuose sluoksniuose išskiriamos molekulių savybių reprezentacijos, tai reiškia, jog šie sluoksniai atsakingi už molekulių SMILES reprezentacijos simbolių atstatymą. Išvesties sluoksnis, panašus į GRU, tačiau papildomai įtraukiamas Gauso triukšmas, kad būtų galima generuoti naujas molekules. Galutinis

dekoderio rezultatas – matrica, kurios dimensijos priklauso nuo molekulės ilgio bei simbolių imties dydžio.

3.2.3. Variacinio autoenkoderio schema

Pristačius koderio ir dekoderio architektūras, toliau pristatoma bendra variacinio autoenkoderio veikimo schema (žr. 14 pav.). Turimi duomenys – molekulės, reprezentuotos SMILES notacija. SMILES simbolių eilutės naudojant vieno vektoriaus kodavimą užkoduojamos į matricas. Molekulių matrica – koderio įvestis, ši matrica perleidžiama per koderio architektūrą (pristatytą 3.2.1. Koderio architektūra skyrelyje) ir gaunamas rezultatas – latentinis vektorius. Vėliau, latentinis vektorius perleidžiamas per dekoderio architektūrą (pristatytą 3.2.2. Dekoderio architektūra skyrelyje) ir gaunama molekulės matrica. Matrica, naudojantis atvirkštiniu vieno vektoriaus kodavimu, iškoduojama į molekulės SMILES reprezentaciją.



14 pav. Variacinio autoenkoderio schema

3.3. Modelių mokymo laikas

Modelių apmokymo laikai pateikti 1-oje lentelėje. Vieno modelio apmokymas vidutiniškai truko vienuolika su puse valandos.

1 lentelė. Modelių mokymo laikas

Modelis	Mokymo laikas (valandomis)
M60	12,4725
M80	11,4511
M100	11,2732
L56	11,485
L156	11,2002
L196	11,2508
L254	11,5516

Lyginant visų modelių apmokymo laikus tarpusavyje, nepastebimas labai didelis skirtumas. Ilgiausiai užtruko išmokyti M60 modelį. Kadangi nepastebima kita tendencija, galima teigti, jog

šio modelio ilgesnis mokymo laikas yra atsitiktinis, galimai priklauso nuo papildomų tuo metu kompiuteryje vykdytų užduočių.

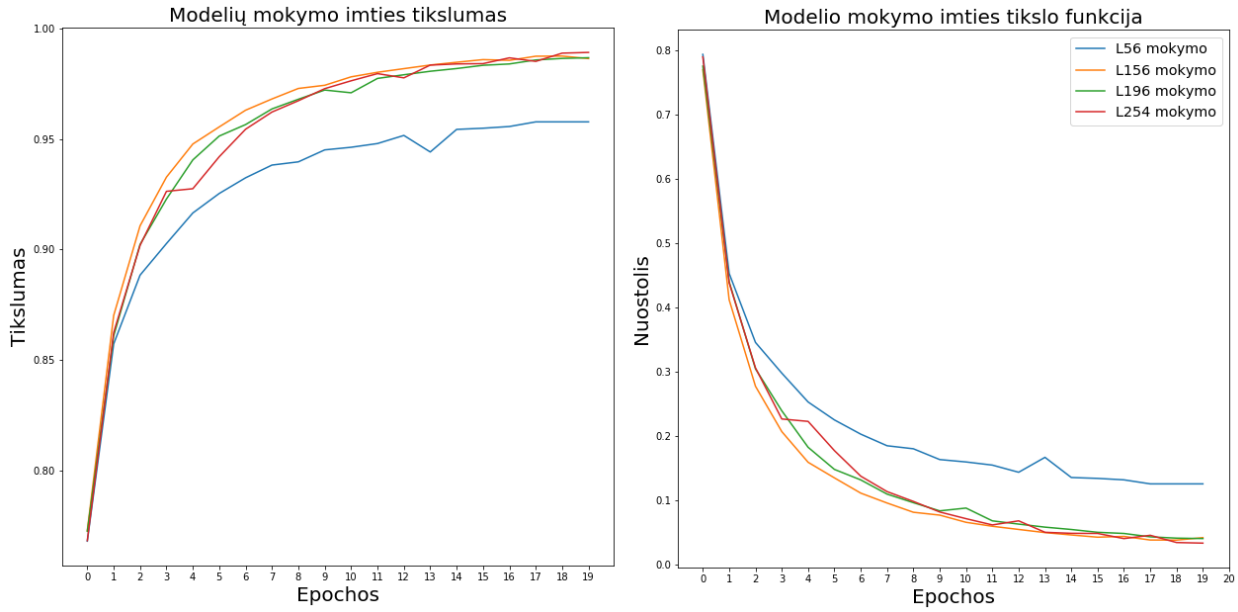
Modelių mokymo laikas, kai turimas skirtingas molekulių ilgis, taip pat iš esmės nesiskiria, kadangi įvesties matrica vis tiek yra tokio paties dydžio. Jei matricos nebūtų papildomos nuliniais vektoriais, mokymo laikas skirtųsi ir trumpesnių molekulių ilgių modelių apmokymas užtruktų trumpiau, kadangi variacinio AE įvesties bei išvesties sluoksnių matricos būtų mažesnės bei reikėtų mažiau nežinomų parametrų skaičiaus šiuose sluoksniuose. Lyginant modelius su skirtingais latentinių vektorių ilgiais, nepastebima mokymo laiko kitimo tendencija, trumpiausiai mokytas modelis su 156 latentinio vektoriaus ilgiu, o ilgiausiai su – 254. Galima teigti, jog mokymo laikas nepriklauso nuo latentinio vektoriaus ilgio ir visiems modeliams yra gana panašus.

3.4. Modelių mokymo tikslumas

Šiame skyrelyje pateikiama modelių mokymo ir validavimo metu gautos tikslo funkcijų reikšmės, bei modelių tikslumas.

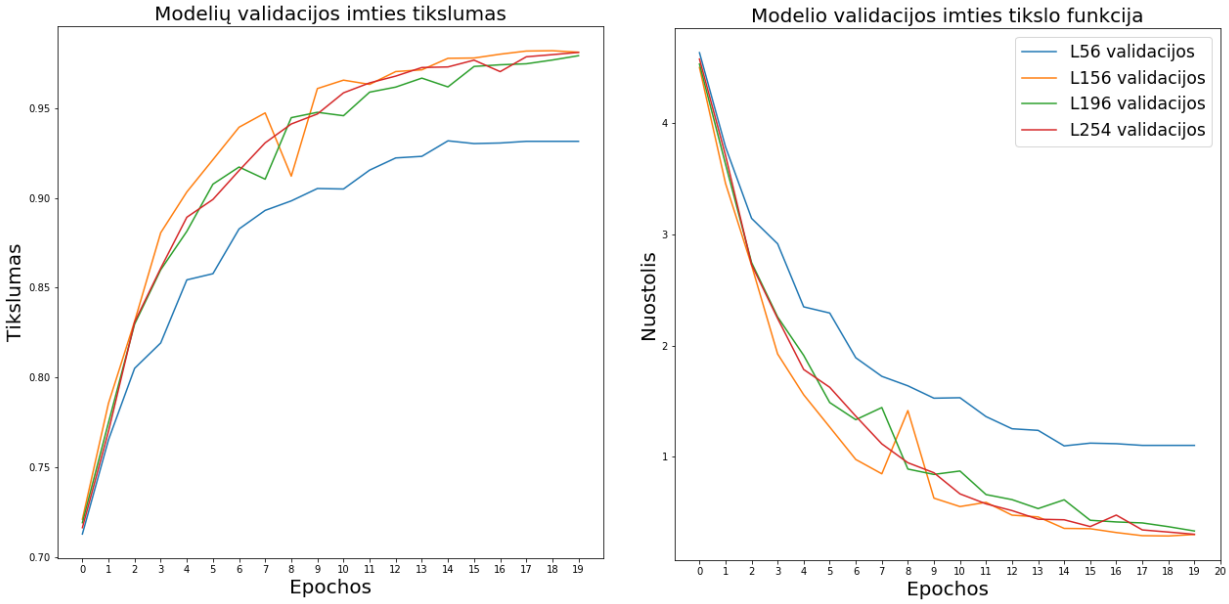
3.4.1. Modelių, su skirtingais latentiniais vektoriais, mokymo tikslumas

Grafike 14 pav. pateikiama modelių, su skirtingais latentiniais vektoriais, tikslumo bei tikslo funkcijos modelių mokymo imties kreivės, o 15 pav. pateikiamos validavimo imties kreivės, skirtingų epochų metu. 90 proc. duomenų paskiriama modelio mokymui, o likusieji 10 proc. – validacijai. Svarbu vienu metu analizuoti modelio ir mokymo ir validacijos kreives, kadangi jų pagalba galima įvertinti modelio mokymo tikslumą. Modelio tikslumo bei modelio tikslo funkcijos kreivės iš esmės yra simetriškos, tik tikslumo kreivės krypties koeficientas teigiamas, o tikslo funkcijos krypties koeficientas neigiamas. Kuomet tikslo funkcija mažėja, gaunamas didesnis tikslumas.



15 pav. Modelių, su skirtingais latentinio vektoriaus ilgiais, mokymo imties tikslumas bei tikslo funkcija skirtingų epochų metu

Iš 15 pav. ir 16 pav. matoma, jog mažiausiai tikslus modelis tiek mokymo tiek validacijos kreivių atžvilgiu yra L56 – modelis, kuomet latentinio vektoriaus ilgis lygus 56. Jau iš modelio mokymo tikslumo galima teigti, jog pasirinktas latentinio vektoriaus ilgis – 56, yra per trumpas naudoti variaciniame AE, siekiant kuo geresnių rezultatų. Matoma, jog paskutinių epochų metu modelio tikslumas nebedidėjo, reiškia didesnis epochų skaičius taip pat nebūtų naudingas.

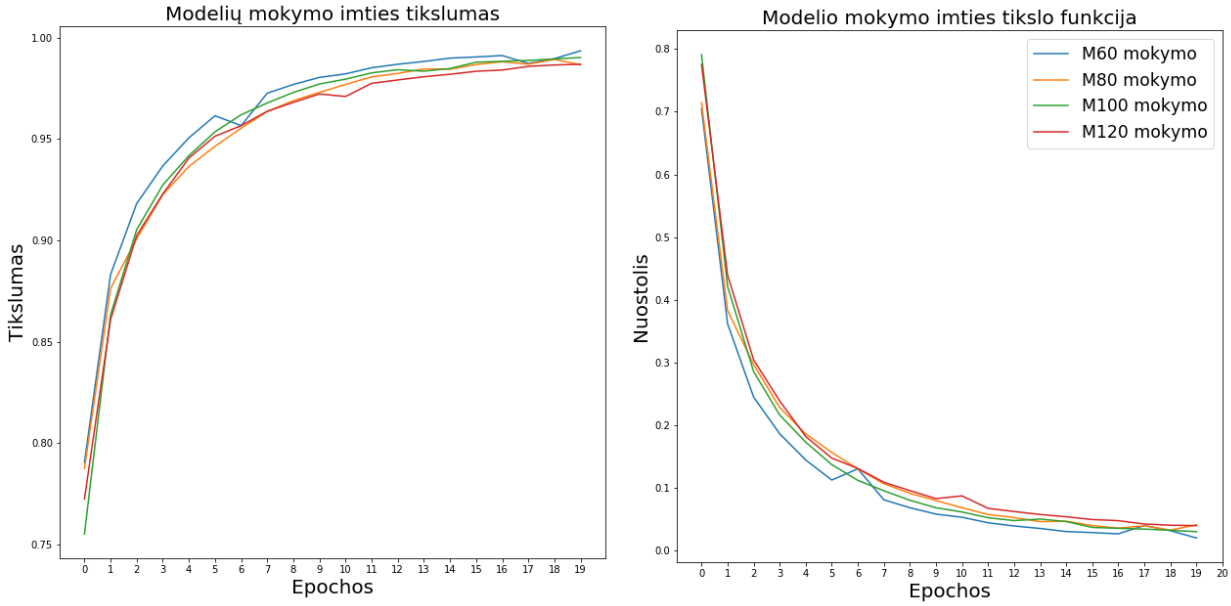


16 pav. Modelių, su skirtingais latentinio vektoriaus ilgiais, validacijos imties tikslumas bei tikslo funkcija skirtingų epochų metu

Lyginant kitus tris modelius tarpusavyje, modelių mokymo tikslumas bei tikslo funkcijos reikšmės gana panašios, geriausioje epochoje modelių L156, L196, L254 tikslumas atitinkamai siekia 98,76 proc., 98,68 proc. bei 98,92 proc. (žr. 1 priedą), o modelių tikslo funkcija, atitinkamai 0,0416, 0,0402, 0,0331 (žr. 2 priedą). Šiuo modelių mokymo atveju, galima būtų teigti, jog labiausiai išsiskiria modelis, kurio latentinio vektoriaus ilgis lygus 254. Toliau analizuojama modelių validacijos tikslumas bei tikslo funkcija. Kai validacijos tikslumas yra gerokai prastesnis, nei modelio mokymo tikslumas, galima teigti, jog modelis persimokė ir išmoko turimus duomenis, tačiau dažniausiai validacijos tikslumas yra šiek tiek mažesnis, nei modelio mokymo tikslumas. Visų trijų modelių validacijos tikslumo kreivės yra žemiau, nei mokymo tikslumo kreivės. Modelio, su latentinio vektoriaus ilgiu lygiu 196, tikslumas, geriausios epochos metu, siekia 97,93 proc., šis modelis, palyginus su kitais dviem yra mažiausiai tikslus (žr. 1 priedą). Lyginant L156 ir L254 modelius, validacijos tikslumas yra panašus ir atitinkamai siekia 98.12 proc. bei 98,11 proc. (žr. 1 priedą). Lyginant apmokytų modelių tikslumą, su literatūroje pateikiamų modelių tikslumu, galima teigti, jog gautas rezultatas geras. Pavyzdžiui, visi modeliai, validacijos imtyje yra tikslesni, nei T. Blaschkeso straipsnyje pateikti variaciniai AE su mokytoju ar be mokytojo bei priešpriešiniai AE: su Gauso ir tolydžiuoju pasiskirstymais [40]. Vienareikšmiškai pasakyti, kuris iš modelių yra geresnis negalima, abu modeliai pasiekia pakankamą tikslumą mokymo ir validavimo metu.

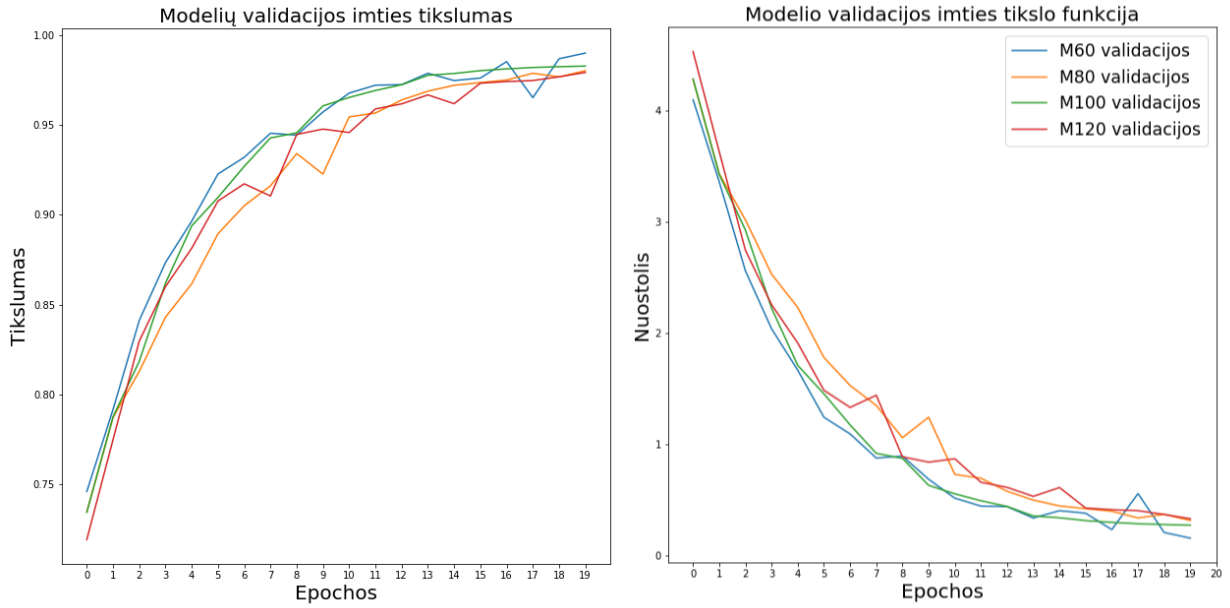
3.4.2. Modelių, su skirtingais molekulių ilgiais, mokymo tikslumas

Grafikuose 17 pav. ir 18 pav. pateikiama modelių, su skirtingais molekulių ilgiais, tikslumo bei tikslo funkcijos modelių mokymo ir validavimo kreivės, skirtingų epochų metu.



17 pav. Modelių, su skirtingais molekulių ilgiais, mokymo imties tikslumas bei tikslo funkcija skirtingų epochų metu

Visi modeliai rodo panašią modelių mokymo tendenciją. Tikslumas visų modelių atveju, mokymo imties, geriausios epochos metu pasiekia daugiau nei 98 proc. tikslumo (žr. 3 priedą), atitinkamai mokymo imties tikslo funkcija nukrito visų modelių atveju žemiau nei 0,05 (žr. 4 priedą).



18 pav. Modelių, su skirtingais molekulių ilgiais, validacijos imties tikslumas bei tikslo funkcija skirtingų epochų metu

Matoma panaši tendencija ir validacijos imties kreivėse. Modelių tikslumą, geriausios epochos metu, pagal mokymo ir validacijos imtis galima išdėstyti didėjimo tvarka taip: M120,

M80, M100, M60. Reikia paminėti, jog visų modelių atveju, molekulių ilgiai papildomi iki 120 simbolių nuliniais vektoriais, o modelio tikslo funkcija iš dalies apibūdina atspėtų simbolių kiekį. Modelis, su trumpiausiu molekulės ilgiu, yra tiksliausias, nes jam reikia atspėti tik pusę elementų, kadangi variaciniam AE nesunku išmokti, jog antra įvesties pusė – nuliniai vektoriai. Analogiškai, kai molekulės ilgis didėja, tuo daugiau simbolių atsiranda, kur galima lengviau suklysti, todėl modelyje duomenų molekulių ilgiui didėjant, tikslumas mažėja.

Natūralu, kad tiksliausias modelis, su molekulės ilgiu lygiu 60, mokymo tikslumas siekia 99,35 proc., o modelio validacijos tikslumas – 99,01 proc. (žr. 3 priedą). Antrasis pagal gerumą modelis, su molekulės ilgiu 100, mokymo imties kreivių atveju pasiekia 99,02 proc. tikslumo, o validacijos kreivė pasiekia šiek tiek mažiau – 98,28 proc. tikslumo (žr. 3 priedą). Modelis, su molekulės ilgiu 80, geriausios epochos metu pasiekia 98,92 proc. mokymo imties tikslumo, o validacijos kreivė pasiekia šiek tiek mažiau – 98,02 proc. (žr. 3 priedą). Modelio, su molekulės ilgiu 120, tikslumas tiek mokymo tiek validacijos imtyse mažesnis, tačiau taip pat pakankamai tikslus. Šio modelio mokymo tikslumas siekia 98,68 proc., o validacijos – 97,92 proc. (žr. 3 priedą). Iš esmės, galima teigti, jog visi modeliai pakankamai tikslūs realiems duomenims, tačiau, kaip tiksliausią galima išskirti modelį, su trumpiausiu molekulės ilgiu.

3.5. Modelių tikslumas, atstatant molekules

Šiame skyrelyje pristatomas kitas modelių tikslumas. Atsitiktinai atrenkamas 1000 molekulių, kiekviena iš jų užkoduojama į latentinį vektorių ir vėl dekoduojama į SMILES reprezentaciją 1000 kartų. Toliau tikrinama kiek molekulių buvo pilnai teisingai atstatyta bent vieną kartą.

3.5.1. Modelių, su skirtingais latentiniais vektoriais, molekulių atstatymo tikslumas

Kitoks tikslumas šioje užduotyje svarbus aspektas, kadangi mokymo ir validacijos tikslumas labiau apibūdina molekulės SMILES reprezentacijos eilutės simbolių skaičiaus teisingumą, kai tuo tarpu net ir vienas, neteisingas simbolis lemia kitokią molekulę ir gali sugeneruoti netinkamą molekulę. Šis tikslumo tikrinimas taip pat atliekamas ir literatūroje [16]. Kadangi dekoderis yra tikimybinis ir priklauso ir nuo Gauso triukšmo, reikalingas didelis bandymų skaičius, kad užtikrinti, jog būtų sugeneruotas rezultatas, lygus įvesčiai.

Mažiausiai tokių pačių molekulių sugeneravo modelis, su latentinio vektoriaus ilgiu lygiu 56. Šis modelis, kaip ir modelio mokymo bei validacijos tikslumo tikrinime, yra prasčiausias, todėl galima teigti, jog latentinio vektoriaus ilgis 56, negali išsaugoti pakankamą informacijos kiekį, kad vėliau molekulė galėtų būti atstatyta į SMILES reprezentacinę eilutę. Modelis, su latentinio vektoriaus ilgiu lygiu 196, taip pat kaip ir modelio mokymo ir validacijos tikslume, pasirodė prasčiau. Iš esmės, atstatytas nemažas kiekis molekulių reprezentacijų – 87,8 proc.. Modeliai su latentinio vektoriaus ilgiu lygiu 156 ir 254 vėl yra geriausi, tačiau tiksliai išskirti, kuris geresnis sunku, kadangi modelis L156 su tūkstančiu bandymų, atstatė 935 molekules iš 1000, o modelis

L254 tik dvejomis mažiau – 933. Skirtumas nėra reikšmingas. Lyginant šių modelių tikslumą, su pateikiamu tikslumu literatūroje, gautas rezultatas labai geras ir pavyzdžiui, pralenkia R. Gomezo-Bombarellio straipsnyje pateiktą tikslumą – 91 proc. [16].

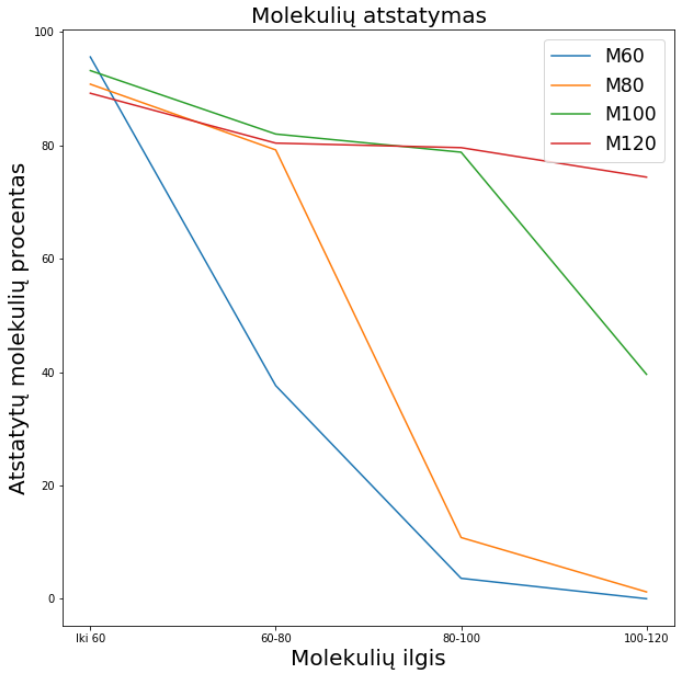
2 lentelė. Modelių, su skirtingais latentiniais vektoriais, molekulių atstatymo tikslumas

Modelis	Teisingai atstatytų molekulių skaičius	Neteisingai atstatytų molekulių skaičius	Molekulių atstatymo tikslumas, proc.
L56	625	375	62,5
L156	935	65	93,5
L196	878	122	87,8
L254	933	67	93,3

Vėl gi, pasakyti tiksliai kuris modelis yra geriausias sunku, tačiau šis tikslumo tikrinimas parodė panašius rezultatus, kaip ir modelių mokymo ir validacijos tikslumas. Tad galima teigti, jog modeliai su latentinio vektoriaus ilgiais 156 ir 254 yra tiksliausi. Modelis L156 abiejuose tikslumo tikrinimuose pasirodo šiek tiek geriau.

3.5.2. Modelių, su skirtingais molekulių ilgiais, molekulių atstatymo tikslumas

Kuomet tikrinamas modelių, su skirtingais molekulių ilgiais, molekulių atstatymo tikslumas, turimą tūkstantį molekulių sudaro keturios lygios grupės, kuriose molekulių ilgis yra atitinkamai iki 60 simbolių, tarp 60 ir 80 simbolių, tarp 80 ir 100 simbolių bei tarp 100 ir 120 simbolių. Tuomet tikrinama kiek molekulių modelis geba atstatyti visose keturiose grupėse. Toliau, 19 paveiksle pateikiama kiekvieno modelio atstatytų molekulių kiekis procentais, priklausomai nuo molekulių ilgio.



19 pav. Modelių, su skirtingais molekulių ilgiais, atstatymo tikslumas

Kai molekulės ilgis yra iki 60, visi modeliai pasiekia apie arba daugiau 90 proc. atstatymą. Šioje grupėje pasiekiamas aukščiausias atstatymo sėkmės rodiklio lygis, visų modelių atveju,

galima teigti, jog trumpas molekules lengviau atstatyti, kadangi didelę dalį molekulę apibūdinančios matricos sudaro nuliniai vektoriai.

Stebint modelius, matoma tendencija, jog modelis geriausiai atstato molekules trumpesnio ilgio, nei buvo duomenų imtis, t. y., modelis, apmokytas su molekulių ilgiu iki 60 simbolių geriausiai atstato būtent tokio ilgio molekules ir pasiekia 95,6 proc. Tuo tarpu kiti lyginami modeliai, su molekulės ilgiu iki 80, 100 ir 120 šios grupės molekules atstato šiek tiek prasčiau, atitinkamai 90,8 proc., 93,2 proc. ir 89,2 proc. Kai molekulės ilgis yra tarp 60 ir 80, geriausiai molekules atstatė modelis, su molekulės ilgiu lygiu 100 – 82 proc., nors kiti modeliai, su molekulės ilgiais lygiais 80 ir 120 atsilieka nedaug ir atstatė atitinkamai 79,2 proc. ir 80,4 proc. molekulių. Tuo tarpu, modelio M60 atstatymo sėkmės rodiklis žymiai krenta iki 37,6 proc. Lyginant paskutinės molekulių grupės atstatymo sėkmės rodiklį, akivaizdu, jog labiausiai tinkamas modelis yra M120. Kadangi kiti modeliai buvo apmokyti naudojant trumpesnes molekules, jie neišmoko teisingai atstatyti ilgesnių molekulių, jei molekulė yra nedaug ilgesnė – atstatymo sėkmės rodiklis šiek tiek didesnis, kaip yra su M100 modeliu, kuris atstatė 39,6 proc. molekulių. Tačiau jei molekulė ryškiai ilgesnė, atstatymo rodiklis krenta iki 0. Modelis, su molekulės ilgiu lygiu 120, gana statiškas visų turimų molekulių grupių atžvilgiu, nors trumpesnių molekulių atstatymas yra šiek tiek geresnis. Taip yra todėl, nes kai molekulę sudaro mažiau simbolių, tai ir tikimybė suklysti mažesnė bei reikia atkreipti dėmesį, kad net ir kai modelis apmokomas su molekulių ilgiu iki 120 simbolių, didesnę dalį duomenų rinkinio sudaro gerokai trumpesnės molekulės.

Galima padaryti išvadą, jog modelio tikslumui atstatant molekules labai svarbi yra duomenų rinkinio struktūra, kuriuo naudojantis apmokomas modelis. Norint turėti modelį, kuris gali kokybiškai užkoduoti ilgesnes molekules, būtina modelio apmokyme naudoti atitinkamo ilgio molekules. Jei turimos trumpesnės molekulės ir tokio ilgio užtenka, naudojant tokio paties dydžio duomenų rinkinį ir tiek pat epochų modelio apmokymo metu, pasiekiamas didesnis tikslumas.

3.6. Modelių validžių molekulių sugeneravimas

Naudojant variacinį AE molekulių generavimui, labai svarbu, jog modelis gebėtų sugeneruoti validžias molekules. Dažniausiai sugeneruotų, validžių molekulių skaičius krenta dėl SMILES reprezentacijos eilutėse neteisingai sudėtų skliaustelių, t. y., trūksta atidarančių ar uždarančių skliaustelių, kurių pagalba žymimi atsišakojimai molekulėse. Molekulių validumas tikrinamas naudojantis programinės įrangos paketais, molekulė bus validi, jei atitiks visas SMILES reprezentacijos taisykles, t. y., jos sintaksinė išraiška bus tinkama. Taip pat validumo metu tikrinamos įvairios molekulės savybės, tokios kaip svoris, atomų skaičius bei ryšiai, jei šios savybės neatitinka realybės – molekulė bus laikoma nevalidžia.

Stebint validžių molekulių sugeneravimą, pastebima panaši modelių gerumo tendencija, mažiausiai validžių molekulių sugeneravo modelis, su latentinio vektoriaus ilgiu lygiu 56, tik 27,11 proc. Daugiau validžių molekulių sugeneravo modelis, su latentinio vektoriaus ilgiu lygiu

196 – 54,68 proc. Vėl gi, geriausi modeliai šiuo aspektu – L156 ir L254. Modelis, su latentinio vektoriaus ilgiu 254 sugeneravo 709 137 validžias molekules, iš 1 000 000 bandymų, t. y., 70,91 proc. Daugiausiai validžių molekulių sugeneravo modelis, su latentinio vektoriaus ilgiu lygiu 156 – 719 758 validžias molekules, iš 1 000 000 bandymų, t. y., 71,98 proc.

3 lentelė. Validžių molekulių generavimas

Modelis	Validžių molekulių sugeneravimas, proc.	Sugeneruotų validžių molekulių skaičius
L56	27,14	271446
L156	71,98	719758
L196	54,68	546787
L254	70,91	709137

Sugeneruotų validžių molekulių skaičius gali būti vertinamas, kaip tinkamas, kadangi literatūroje aprašyti modeliai pasiekia panašų ar mažesnį sugeneravimo sėkmės rodiklį. Pavyzdžiui, S. Harel ir K. Radinsky [42] aprašyto variacinio AE validžių molekulių generavimo sėkmės rodiklis siekia 58 proc., o J. Lim ir kt. [14] sąlyginis variacinis AE sugeneruoja 46 proc. validžių molekulių.

Po trijų modelių tikslumo ir naudingumo vertinimo aspektų, galima teigti, jog geriausi yra modeliai, su latentiniais vektorių ilgiais lygiais 156 ir 254. Šie modeliai tikslumo aspektais yra labai panašūs, L156 šiek tiek tikslesnis. Naudingumo aspektu geresnis modelis L156, t. y., daugiau validžių molekulių sugeneruota naudojantis modeliu, su latentinio vektoriaus ilgiu lygiu 156. Todėl galima laikyti, jog latentinės erdvės matmuo lygus 156 geriausiai apibūdina turimo duomenų rinkinio molekules.

3.7. Molekulių generavimas variaciniu autoenkoderiu

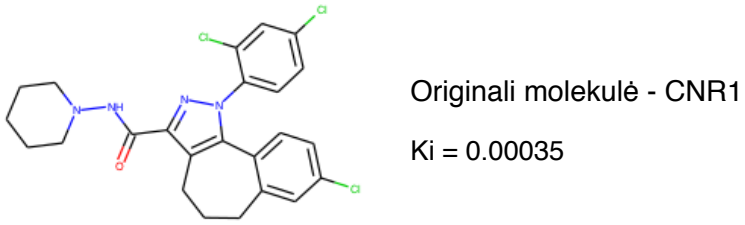
Generuojant molekules su variaciniu AE, molekulės latentinis vektorius su geriausia inhibicijos konstanta $Ki = 0.00035$, buvo dekoduojamas 1000 kartų, vėliau tikrinama ar molekulės validžios. Generavimas atliekamas analogiškai naudojant skirtingus latentinio vektoriaus ilgius. Reikia pastebėti, jog kiekis, kiek validžių molekulių bus sugeneruota yra visiškai atsitiktinis. Galimas variantas, jog generavimą atlikus dar kartą, validžių molekulių skaičius skirtųsi. Atliekama tik vienas bandymas, kadangi priešingu atveju būtų sugeneruota labai daug molekulių, kurias būtų sunku atvaizduoti.

Toliau, 4 lentelėje pateikiama, kiek bandymo metu buvo sugeneruota validžių skirtingų molekulių, pagal latentinių vektorių ilgį.

4 lentelė. Validžių molekulių skaičius, pagal latentinio vektoriaus ilgį

Latentinio vektoriaus ilgis	Validžių naujų molekulių skaičius
56	9
156	24
196	3
254	8

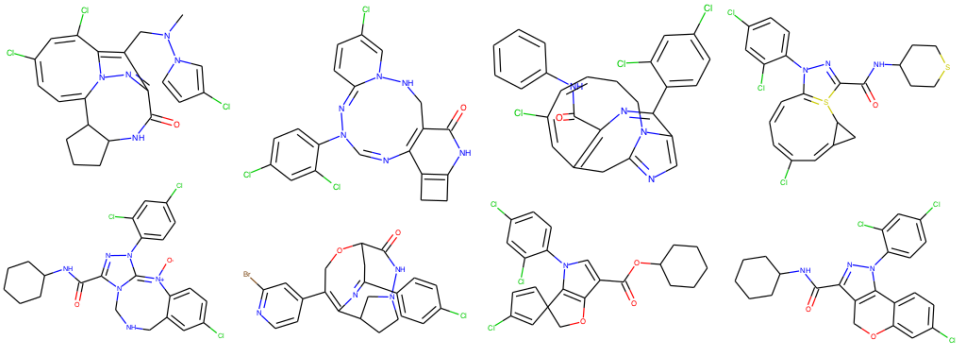
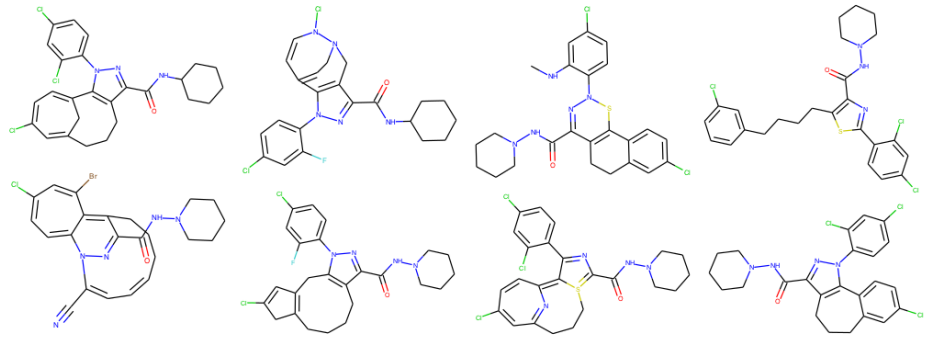
Iš viso sugeneruota 44 naujos molekulės. Toliau tikrinama, ar tarp skirtingų latentinio vektoriaus ilgių modelių yra sugeneruotų tokių pačių molekulių. Tarp 44 naujai sugeneruotų molekulių, pastebėtos dvi vienodos, kurias sugeneravo modeliai su latentinių vektorių ilgiais lygiais 156 bei 254. Tad rezultatas, generuojant naujas molekules su keturias skirtingų latentinių vektorių ilgių modeliais: gautos 42 naujos, unikalios molekulės. Gavus rezultatą, jog dvi vienodos molekulės buvo sugeneruotos naudojantis skirtingomis latentinės erdvės dimensijomis, galima prieiti išvadą, jog nesvarbu kokia latentinės erdvės dimensija, bet tarpusavyje panašios molekulės išdėstomos šalia. Kadangi visos molekulės dalinasi tuo pačiu latentiniu vektoriumi, kaip ir originali molekulė, galima teigti, jog ir šių molekulių inhibicijos konstanta $Ki = 0{,}00035$.

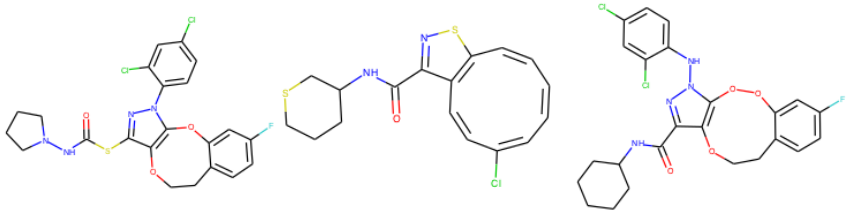
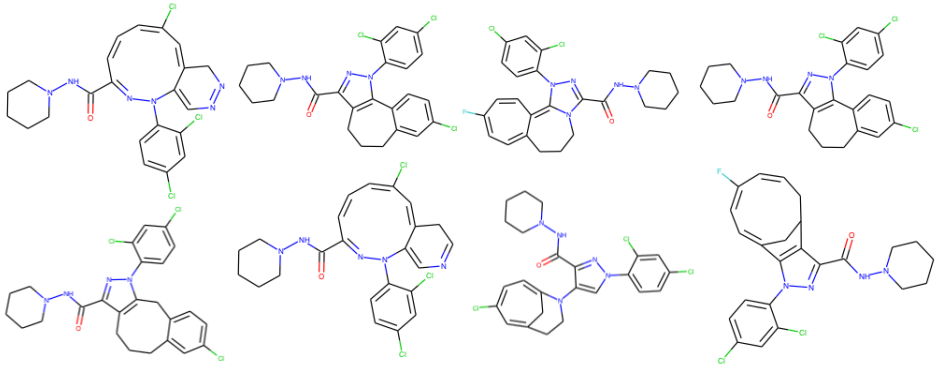


20 pav. Originali molekulė

Toliau, 5 lentelėje pateikiama sugeneruotų molekulių pavyzdžiai, prie skirtingų latentinio vektoriaus ilgių, o 20 paveiksle matoma originali molekulė, kurios latentinio vektoriaus pagrindu buvo generuojamos naujos molekulės.

5 lentelė. Sugeneruotų molekulių pavyzdžiai, su skirtingais latentinio vektoriaus ilgiais

Latentinio vektoriaus ilgis	Sugeneruotų molekulių pavyzdžiai
56	
156	

196	
254	

3.8. Naujų molekulių generavimas, naudojant interpoliaciją

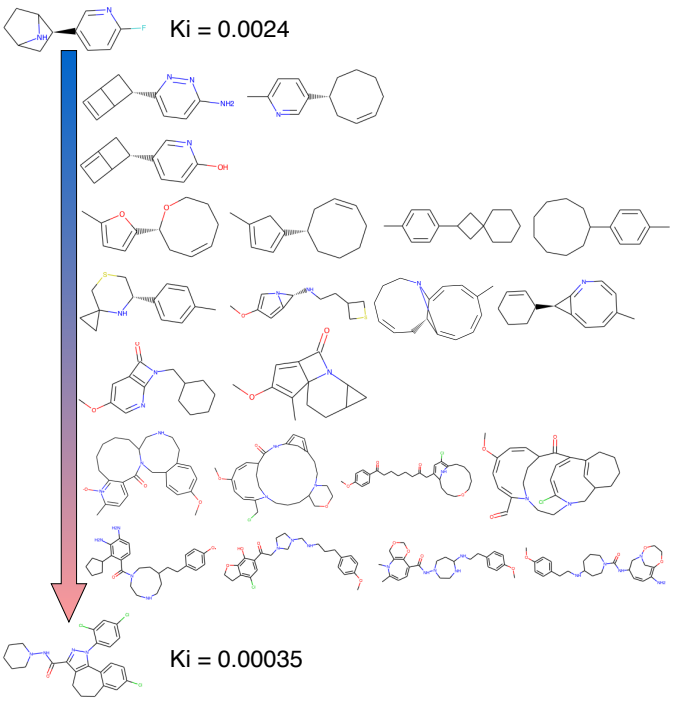
Vientisa latentinė erdvė leidžia molekulių interpoliavimą sekant trumpiausiu Euklido keliu tarp latentinių vektorių [16]. Atsitiktinai parenkamos dvi molekulės, tarp kurių vyks tiesinė interpoliacija. Generuojami nauji latentinės erdvės taškai judės nuo molekulės, kurios inhibicijos konstanta $Ki = 0{,}0024$, iki molekulės su mažesne jungimosi savybe – $Ki = 0{,}00035$. Kadangi latentinė erdvė yra vientisa ir išsidėsčiusi pagal molekulių savybes, galima teigti, jog sugeneruotos naujos molekulės turės inhibicijos konstantą $0{,}00035 < Ki < 0{,}0024$. Naujos sugeneruotos molekulės, pirmose iteracijose bus panašesnės į pradinę molekulę, o galutinėse iteracijose, sugeneruotos molekulės bus panašesnės į tikslo molekulę.

Pirmiausia sugeneruojami pradžios ir tikslo molekulių latentiniai vektoriai, tuomet apskaičiuojamas žingsnis tarp jų ir iteraciniu būdu apskaičiuojami naujų molekulių latentiniai vektoriai:

$$L(i) = L_{pr} + \left(\frac{L_t - L_{pr}}{n} \cdot i\right), \tag{18}$$

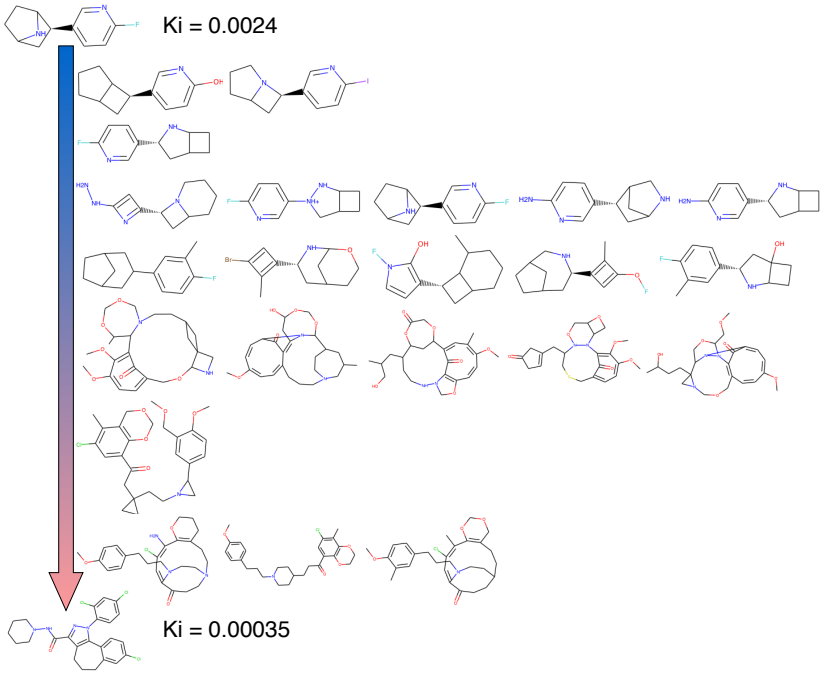
kur $L(i)$ – naujos i-tosios molekulės latentinis vektorius, L_{pr} – pradinės molekulės latentinis vektorius, L_t – tikslo molekulės latentinis vektorius, n – interpoliacijos žingsnių skaičius.

Toliau pateikiama naujų molekulių generavimo interpoliacijos vizualizacija, naudojant modelius, su skirtingais latentinio vektoriaus ilgiais. Pasirinktas interpoliacijos žingsnis $n = 7$.



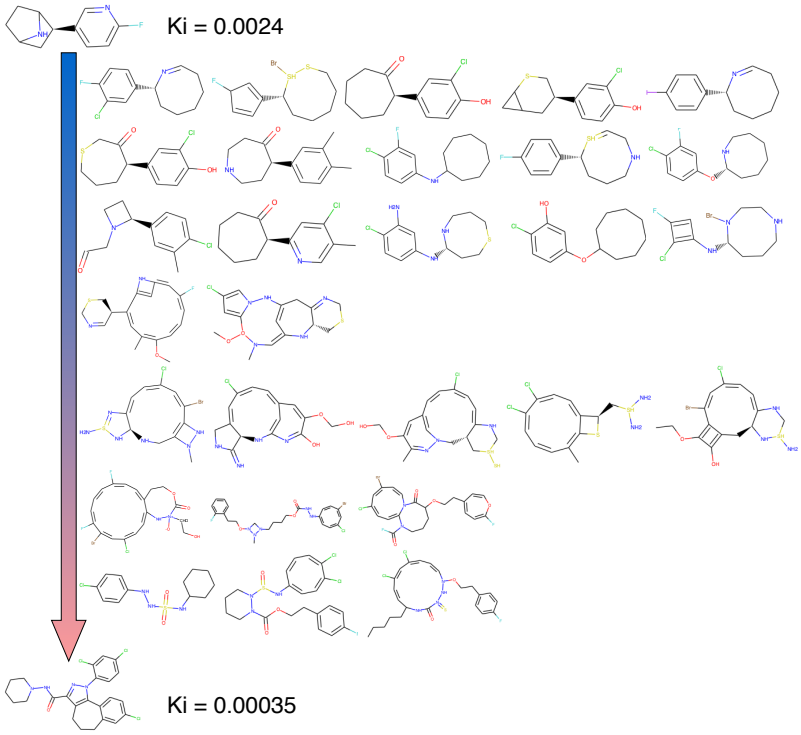
21 pav. Molekulių generavimas naudojant interpoliaciją, naudojamas L56 modelis

Interpoliacijos rezultatų vizualizacija, kuomet naudojamas modelis, su latentinio vektoriaus ilgiu lygiu 56, pateikta 21 paveiksle. Galima pastebėti, jog pirmųjų iteracijų metu molekulės vizualiai panašesnės į pirminę molekulę, o galutinėse iteracijos, į tikslo molekulę, t. y. pirmųjų iteracijų metu sugeneruotos molekulės turi mažai ir didesnius atomų junginius, kai tuo tarpu tolimesnėse iteracijos molekulės turi daugiau atsišakojimų, mažesnius ir daugiau atomų junginių. Antrojoje iteracijoje iš latentinio vektoriaus sugeneruota tik viena validi molekulė, tuo tarpu pirmoje ir penktoje iteracijose – dvi validžios molekulės, likusiose iteracijose sugeneruota daugiau nei keturios molekulės.



22 pav. Molekulių generavimas naudojant interpoliaciją, naudojamas L156 modelis

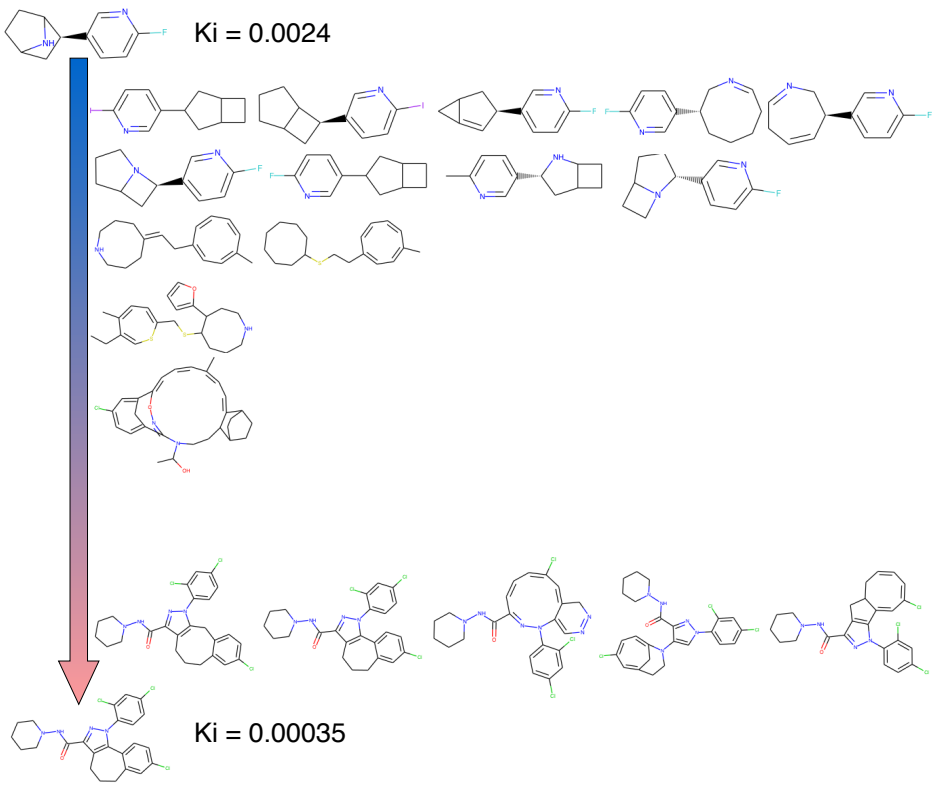
Interpoliacijos rezultatų vizualizacija, kuomet naudojamas modelis, su latentinio vektoriaus ilgiu lygiu 156, pateikta 22 paveiksle. Kaip ir ankstesniame paveiksle, matoma, jog vizualiai pirmųjų iteracijų metu molekulės panašesnės į pirminę molekulę, o galutinėse iteracijos – į tikslo molekulę. Šiuo atveju antrojoje ir šeštojoje iteracijose buvo sugeneruotos po vieną validžią molekulę, kitose iteracijose sugeneruota daugiau nei viena validi molekulė.



23 pav. Molekulių generavimas naudojant interpoliaciją, naudojamas L196 modelis

Interpoliacijos rezultatų vizualizacija, kuomet naudojamas modelis, su latentinio vektoriaus ilgiu lygiu 196, pateikta 23 paveiksle. Naudojant šį modelį, sugeneruota daugiausiai naujų validžių molekulių. Mažiausiai sugeneruota validžių molekulių ketvirtoje iteracijoje – dvi bei šeštoje ir septintoje iteracijose sugeneruotos po tris validžias molekules. Vėl gi, galima pastebėti molekulių vizualaus panašumo tendenciją, kintant iteracijoms.

Interpoliacijos rezultatų vizualizacija, kuomet naudojamas modelis, su latentinio vektoriaus ilgiu lygiu 254, pateikta 24 paveiksle. Naudojant šį modelį, šeštoje iteracijoje nepavyko sugeneruoti nei vienos naujos validžios molekulės. Kadangi latentinė erdvė yra vientisa, reiškia bet kokiame latentinės erdvės taške galima sugeneruoti molekulę, todėl priežastis, kodėl interpoliacijos atveju nebuvo sugeneruota nauja molekulė yra per mažas bandymų skaičius. Kituose taškuose molekulės sėkmingai buvo sugeneruotos. Šio modelio atveju, paskutinės iteracijos metu sugeneruotos molekulės vizualiai yra labiausiai artimos tikslo molekulei.



24 pav. Molekulių generavimas naudojant interpoliaciją, naudojamas L254 modelis

Generuojant naujas molekules, naudojant interpoliaciją, nebuvo sugeneruotų vienodų molekulių, nei tarp modelių, nei tarp artimų iteracijų. Tai reiškia, kad kiekvieno modelio latentinė erdvė yra unikali, o septynių žingsnių interpoliacija tarp parinktų molekulių yra pakankama, jog iš dviejų latentinių vektorių nebūtų sugeneruotos tos pačios molekulės.

Išvados

1. Buvo apmokyti keturi variacinio autoenkoderio modeliai su skirtingais latentinių vektorių ilgiais lygiais 56, 156, 196 ir 254.

- Mokymo ir validavimo tikslume geriausiai pasirodė modeliai, su latentiniais vektorių ilgiais lygiais 156 ir 254, šie modeliai pasiekė virš 98 proc. tikslumą, tiek validavimo tiek mokymo imtyse.
- Lyginant pagal modelių gebėjimą atstatyti molekules, generuojant tūkstantį bandymų, taip pat išsiskyrė modeliai, su latentiniais vektoriais lygiais 156 ir 254, modeliai, atitinkamai atstatė 93,5 ir 93,3 proc. molekulių.
- Trečiasis tikslumo aspektas, validžių molekulių generavimas, vėl gi kaip geriausius išskyrė modelius, su latentinio vektoriaus ilgiais lygiais 156 ir 254. Šiek tiek geriau pasirodė modelis, su latentinės erdvės vektoriumi lygiu 156 ir jo validžių molekulių generavimo sėkmės rodiklis siekia 71,98 procentus.

Visais aspektais mažiausiai tikslus buvo modelis su latentinės erdvės vektoriumi lygiu 56, todėl galima teigti, jog latentinės erdvės matmuo lygus 56 negali tinkamai reprezentuoti turimo molekulių duomenų rinkinio. Atitinkamai, galima teigti, jog turimą molekulių rinkinį geriausiai apibūdina latentinės erdvės matmuo lygus 156.

2. Buvo apmokyti keturi variacinio autoenkoderio modeliai, su skirtingais molekulių ilgiais, lygiais 60, 80, 100, 120.

- Mokymo ir validavimo tikslume geriausiai pasirodė modelis, su molekulių ilgiu lygiu 60, kuris tiek mokymo, tiek validacijos imtyse pasiekė virš 99 proc. tikslumo.
- Lyginant pagal modelių gebėjimą atstatyti skirtingo ilgio molekules, gauta, jog modelis su molekulės ilgiu lygiu 120, yra stabiliausias visų molekulių ilgiams. Kai molekulės ilgis yra iki 60 simbolių, daugiausiai molekulių atstatė modelis, su molekulių ilgiu lygiu 60 – 95,6 proc.

Gaunama išvada, jog modelio tikslumui atstatant molekules labai svarbi yra duomenų rinkinio, kuriuo naudojantis apmokomas modelis, struktūra.

3. Naudojantis variaciniu autoenkoderiu, buvo sugeneruotos molekulės, su norima mažiausia inhibicijos konstanta. Molekulės generuotos naudojantis modeliais, su skirtingais latentiniais vektorių ilgiais. Gautos 42 naujos, unikalios molekulės, modeliai su latentinio vektoriaus ilgiais lygiais 156 ir 254 sugeneravo dvi vienodas molekules, tai parodo, jog latentinė erdvė, nepriklausomai nuo dimensijos, išdėsto panašias molekules šalia.

4. Tarp molekulių latentinių vektorių buvo atlikta tiesinė septynių žingsnių interpoliacija. Gauti latentinės erdvės vektoriai buvo dekoduoti į validžias molekules, kurių inhibicijos konstanta yra tarp 0,0024 ir 0,00035. Nustatyta, jog pirmosiomis interpoliacijos iteracijomis, molekulės vizualiai panašesnės į pirminę molekulę, o paskutinėse iteracijose panašesnės į tikslo molekulę.

Padėka

Norėčiau padėkoti darbo vadovui doc. dr. M. Landauskui už lankstumą bei pagalbą ir konsultacijas atliekant darbe iškeltas užduotis.

Taip pat norėčiau padėkoti Baltijos pažangių technologijų instituto mokslininkams už pasiūlytą įdomią ir aktualią temą bei pagalbą, padedant įsigilinti į dirbtinių neuroninių tinklų sritį. Nuoširdžiai dėkoju darbo konsultantui V. Rafanavičiui už malonų bendradarbiavimą ir suteiktas konsultacijas ir įžvalgas viso tyrimo metu.

Pranešimas konferencijoje

M. Šniokaitė, M. Landauskas, V. Rafanavičius – Naujų vaistinių molekulių generavimas giliaisiais dirbtiniais neuroniniais tinklais („Matematika ir matematikos dėstymas – 2019“, 2019-04-26)

Literatūros sąrašas

1. SANCHEZ-LENGELING, B., A. ASPURU-GUZIK. Inverse molecular design using machine learning: Generative models for matter engineering. *Science* [interaktyvus]. 2018. Vol. **361**(6400), p. 360–365 [žiūrėta 2019-02-28]. doi: 10.1126/science.aat2663
2. Accelerated Molecular Discovery [interaktyvus]. [žiūrėta 2019-02-28]. Prieiga per internetą: https://www.darpa.mil/program/accelerated-molecular-discovery.
3. MITCHELL, J.B.O. Machine learning methods in chemoinformatics: Machine learning methods in chemoinformatics. *Wiley Interdisciplinary Reviews: Computational Molecular Science* [interaktyvus]. 2014. Vol. **4**(5), p. 468–481 [žiūrėta 2019-02-28]. doi: 10.1002/wcms.1183
4. GASTEIGER, J. Chemoinformatics: Achievements and Challenges, a Personal View. *Molecules* [interaktyvus]. 2016. Vol. **21**(2), p. 151 [žiūrėta 2019-02-25]. doi: 10.3390/molecules21020151
5. JIN, W. ir kt. Junction Tree Variational Autoencoder for Molecular Graph Generation. *arXiv:1802.04364* [interaktyvus]. 2018. [žiūrėta 2019-02-18]. Prieiga per internetą: http://arxiv.org/abs/1802.04364.
6. CHEN, H. ir kt. The rise of deep learning in drug discovery. *Drug Discovery Today* [interaktyvus]. 2018. Vol. **23**(6), p. 1241–1250 [žiūrėta 2019-02-20]. ISSN 1359-6446. doi: 10.1016/j.drudis.2018.01.039.
7. WU, Z. ir kt. MoleculeNet: a benchmark for molecular machine learning. *Chemical Science* [interaktyvus]. 2018. Vol. **9**(2), p. 513–530 [žiūrėta 2019-02-20]. doi: 10.1039/C7SC02664A
8. POLISHCHUK, P.G. ir kt. Estimation of the size of drug-like chemical space based on GDB-17 data. *Journal of Computer-Aided Molecular Design* [interaktyvus]. 2013. Vol. **27**(8), p. 675–679 [žiūrėta 2019-04-30]. doi: 10.1007/s10822-013-9672-4
9. ELTON, D.C. ir kt. Deep learning for molecular generation and optimization - a review of the state of the art. *arXiv:1903.04388* [interaktyvus]. 2019. [žiūrėta 2019-03-18]. Prieiga per internetą: http://arxiv.org/abs/1903.04388.
10. XU, Y. ir kt. Deep learning for molecular generation. *Future Medicinal Chemistry* [interaktyvus]. 2019. Vol. **11**(6), p. 567-597 [žiūrėta 2019-03-18]. doi: 10.4155/fmc-2018-0358
11. BJERRUM, E.J., R. THRELFALL. Molecular Generation with Recurrent Neural Networks (RNNs). *arXiv:1705.04612* [interaktyvus]. 2017. [žiūrėta 2019-03-20]. Prieiga per internetą: http://arxiv.org/abs/1705.04612.
12. SEGLER, M.H.S. ir kt. Generating Focussed Molecule Libraries for Drug Discovery with Recurrent Neural Networks. *arXiv:1701.01329* [interaktyvus]. 2017. [žiūrėta 2019-02-25]. Prieiga per internetą: http://arxiv.org/abs/1701.01329.
13. PUTIN, E. ir kt. Reinforced Adversarial Neural Computer for *de Novo* Molecular Design. *Journal of Chemical Information and Modeling* [interaktyvus]. 2018. Vol. **58**(6), p. 1194–1204 [žiūrėta 2019-04-15]. doi: 10.1021/acs.jcim.7b00690
14. LIM, J. ir kt. Molecular generative model based on conditional variational autoencoder for de novo molecular design. *Journal of Cheminformatics* [interaktyvus]. 2018. Vol. **10**(1). [žiūrėta 2019-03-06]. doi: 10.1186/s13321-018-0286-7
15. KANG, S., K. CHO. Conditional Molecular Design with Deep Generative Models. *Journal of Chemical Information and Modeling* [interaktyvus]. 2019. Vol. **59**(1), p. 43–52 [žiūrėta 2019-04-20]. doi: 10.1021/acs.jcim.8b00263
16. GÓMEZ-BOMBARELLI, R. ir kt. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science* [interaktyvus]. 2018. Vol. **4**(2), p. 268–276 [žiūrėta 2019-04-20]. doi: 10.1021/acscentsci.7b00572
17. KADURIN, A. ir kt. druGAN: An Advanced Generative Adversarial Autoencoder Model for de Novo Generation of New Molecules with Desired Molecular Properties in Silico. *Molecular Pharmaceutics* [interaktyvus]. 2017. Vol. **14**(9), p. 3098–3104 [žiūrėta 2019-04-20]. doi: 10.1021/acs.molpharmaceut.7b00346
18. KADURIN, A. ir kt. The cornucopia of meaningful leads: Applying deep adversarial autoencoders for new molecule development in oncology. *Oncotarget* [interaktyvus]. 2017. Vol. **8**(7). [žiūrėta 2019-02-26]. doi: 10.18632/oncotarget.14073
19. BATES, J. Glossary of physiological terms. *Transactions of the IRE Professional Group on Information Theory* [interaktyvus]. 1953. Vol. **1**(1), p. 5–8 [žiūrėta 2019-04-10]. doi: 10.1109/TIT.1953.1188574
20. VENKATARAMANI, P.V. Santiago Ramón y Cajal: Father of neurosciences. *Resonance* [interaktyvus]. 2010. Vol. **15**(11), p. 968–976 [žiūrėta 2019-04-10]. doi: 10.1007/s12045-010-0113-6
21. BASHEER, I.A., M. HAJMEER. Artificial neural networks: fundamentals, computing, design, and application. *Journal of Microbiological Methods* [interaktyvus]. 2000. Vol. **43**(1), p. 3–31 [žiūrėta 2019-04-10]. doi:10.1016/S0167-7012(00)00201-3
22. GROSSI, E., M. BUSCEMA. Introduction to artificial neural networks. *European Journal of Gastroenterology & Hepatology* [interaktyvus]. 2007. Vol. **19**(12), p. 1046 [žiūrėta 2019-04-30]. Doi: 10.1097/MEG.0b013e3282f198a0
23. GOODFELLOW, I., Y. BENGIO ir A. COURVILLE. *Deep Learning book* [interaktyvus]. [žiūrėta 2019-03-05]. MIT Press, 2016 [žiūrėta 2019-04-30]. Prieiga per internetą: http://www.deeplearningbook.org.
24. WILAMOWSKI, B. Neural network architectures and learning algorithms. *IEEE Industrial Electronics Magazine* [interaktyvus]. 2009. Vol. **3**(4), p. 56–63 [žiūrėta 2019-04-04]. doi: 10.1109/MIE.2009.934790
25. KHAN, K., SAHAI, A. A Comparison of BA, GA, PSO, BP and LM for Training Feed forward Neural Networks in e-Learning Context. *International Journal of Intelligent Systems and Applications* [interaktyvus]. 2012. Vol. **4**(7), p. 23–29 [žiūrėta 2019-04-04]. doi: 10.5815/ijisa.2012.07.03

26. JORDAN, M.I., T.M. MITCHELL. Machine learning: Trends, perspectives, and prospects. *Science* [interaktyvus]. 2015. Vol. **349**(6245), p. 255–260 [žiūrėta 2019-04-06]. doi: 10.1126/science.aac4520
27. MARR, B. A Short History of Machine Learning -- Every Manager Should Read. *Forbes* [interaktyvus]. [žiūrėta 2019-04-13]. Prieiga per internetą: https://www.forbes.com/sites/bernardmarr/2016/02/19/a-short-history-of-machine-learning-every-manager-should-read/.
28. WASSERMAN, P.D., T. SCHWARTZ. Neural networks. II. What are they and why is everybody so interested in them now? *IEEE Expert* [interaktyvus]. 1988. Vol. **3**(1), p. 10–15 [žiūrėta 2019-04-13]. doi: 10.1109/64.2091
29. MARVIN L. MINSKY. *Perceptrons: An introduction to computational geometry* [interaktyvus]. 1988. 308 p. [žiūrėta 2019-04-20]. ISBN 978-0-262-53477-2.
30. RUMELHART, D.E. ir kt. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* [interaktyvus]. Cambridge, MA, USA: MIT Press, 1986. Vol **1**, p. 318–362. [žiūrėta 2019-03-22]. ISBN 978-0-262-68053-0 Prieiga per internetą: http://dl.acm.org/citation.cfm?id=104279.104293.
31. BALDI, P. Autoencoders, Unsupervised Learning, and Deep Architectures. JMLR: Workshop and Conference Proceedings [interaktyvus]. 27:37-50, 2012. p. 14 [žiūrėta 2019-04-03].
32. HOO-CHANG SHIN ir kt. Stacked Autoencoders for Unsupervised Feature Learning and Multiple Organ Detection in a Pilot Study Using 4D Patient Data. *IEEE Transactions on Pattern Analysis and Machine Intelligence* [interaktyvus]. 2013. Vol. **35**(8), p. 1930–1943 [žiūrėta 2019-04-03]. doi: 10.1109/TPAMI.2012.277
33. LU, X. ir kt. Speech Enhancement Based on Deep Denoising Autoencoder. *INTERSPEECH 2013* [interaktyvus]. 2013. p. 5 [žiūrėta 2019-04-13].
34. SAKURADA, M., T. YAIRI. Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction. *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis - MLSDA'14* [interaktyvus]. Gold Coast, Australija QLD, Australija: ACM Press, 2014. p. 4–11. [žiūrėta 2019-03-27]. doi: 10.1145/2689746.2689747
35. LU, C. ir kt. Fault diagnosis of rotary machinery components using a stacked denoising autoencoder-based health state identification. *Signal Processing* [interaktyvus]. 2017. Vol. **130**, p. 377–388 [žiūrėta 2019-04-14]. doi: 10.1016/j.sigpro.2016.07.028
36. ZHU, Z. ir kt. Deep Learning Representation using Autoencoder for 3D Shape Retrieval. *arXiv:1409.7164* [interaktyvus]. 2014. [žiūrėta 2019-03-27]. doi: 10.1016/j.neucom.2015.08.127
37. FINN, C. ir kt. Deep Spatial Autoencoders for Visuomotor Learning. *arXiv:1509.06113* [interaktyvus]. 2015. [žiūrėta 2019-03-27]. Prieiga per internetą: http://arxiv.org/abs/1509.06113.
38. Quantum-Machine.org: [interaktyvus]. [žiūrėta 2019-04-02]. Prieiga per internetą: http://quantum-machine.org/.
39. ZINC subset Clean Drug-Like | ZINC Is Not Commercial - A database of commercially-available compounds [interaktyvus]. [žiūrėta 2019-02-08]. Prieiga per internetą: http://zinc.docking.org/subsets/clean-drug-like.
40. BLASCHKE, T. ir kt. Application of Generative Autoencoder in *De Novo* Molecular Design. *Molecular Informatics* [interaktyvus]. 2018. Vol. **37**(1–2), p. 1700123 [žiūrėta 2019-04-06]. doi: 10.1002/minf.201700123
41. ChEMBL Database [interaktyvus]. [žiūrėta 2019-04-02]. Prieiga per internetą: https://www.ebi.ac.uk/chembl/.
42. HAREL, S., K. RADINSKY. Prototype-Based Compound Discovery Using Deep Generative Models. *Molecular Pharmaceutics* [interaktyvus]. 2018. Vol. **15**(10), p. 4406–4416 [žiūrėta 2019-04-06]. doi: 10.1021/acs.molpharmaceut.8b00474
43. SIMONOVSKY, M., N. KOMODAKIS. GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders. *arXiv:1802.03480* [interaktyvus]. 2018. [žiūrėta 2019-03-18]. Prieiga per internetą: http://arxiv.org/abs/1802.03480.
44. SAMANTA, B. ir kt. NeVAE: A Deep Generative Model for Molecular Graphs. *arXiv:1802.05283* [interaktyvus]. 2018. [žiūrėta 2019-04-25]. Prieiga per internetą: http://arxiv.org/abs/1802.05283.
45. KUZMINYKH, D. ir kt. 3D Molecular Representations Based on the Wave Transform for Convolutional Neural Networks. *Molecular Pharmaceutics* [interaktyvus]. 2018. Vol. **15**(10), p. 4378–4385 [žiūrėta 2019-04-30]. doi: 10.1021/acs.molpharmaceut.7b01134.
46. WEININGER, D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of Chemical Information and Modeling* [interaktyvus]. 1988. Vol. **28**(1), p. 31–36 [žiūrėta 2019-04-30]. doi: 10.1021/ci00057a005
47. BONE, R.G.A. ir kt. SMILES Extensions for Pattern Matching and Molecular Transformations: Applications in Chemoinformatics. *Journal of Chemical Information and Computer Sciences* [interaktyvus]. 1999. Vol. **39**(5), p. 846–860 [žiūrėta 2019-04-30]. doi: 10.1021/ci990422w
48. BROWNLEE, J. How to One Hot Encode Sequence Data in Python [interaktyvus]. 2017. [žiūrėta 2019-03-14]. Prieiga per internetą: https://machinelearningmastery.com/how-to-one-hot-encode-sequence-data-in-python/.
49. BRINGMANN, B., KARWATH A. Frequent SMILES. *ResearchGate* [interaktyvus]. [žiūrėta 2019-03-04]. Prieiga per internetą: https://www.researchgate.net/publication/221146866_Frequent_SMILES.
50. GILSON, M.K. ir kt. BindingDB in 2015: A public database for medicinal chemistry, computational chemistry and systems pharmacology. *Nucleic Acids Research* [interaktyvus]. 2016. Vol. **44**(D1), p. D1045–D1053 [žiūrėta 2019-04-11]. doi: 10.1093/nar/gkv1072
51. LIU, T. ir kt. BindingDB: a web-accessible database of experimentally determined protein-ligand binding affinities. *Nucleic Acids Research* [interaktyvus]. 2007. Vol. **35**(Database), p. D198–D201 [žiūrėta 2019-04-30]. doi: 10.1093/nar/gkl999
52. HARRIS, S., D. HARRIS. *Digital Design and Computer Architecture* [interaktyvus]. Elsevier, 2013. ISBN: 978-0-12-394424-5 [žiūrėta 2019-04-11].

53. RODRÍGUEZ, P. ir kt. Beyond One-hot Encoding: lower dimensional target embedding. *Image and Vision Computing* [interaktyvus]. 2018. Vol. **75**, p. 21–31 [žiūrėta 2019-04-11]. doi: 10.1016/j.imavis.2018.04.004
54. MEYER, D. Introduction to Autoencoders [interaktyvus]. 2015. p. 8 [žiūrėta 2019-04-30].
55. BOWMAN, S.R. ir kt. Generating Sentences from a Continuous Space. *arXiv:1511.06349* [interaktyvus]. 2015. [žiūrėta 2019-02-25]. Prieiga per internetą: http://arxiv.org/abs/1511.06349.
56. RAGOZA, M. ir kt. Protein-Ligand Scoring with Convolutional Neural Networks. *arXiv:1612.02751* [interaktyvus]. 2016. [žiūrėta 2019-02-27]. Prieiga per internetą: http://arxiv.org/abs/1612.02751.
57. LECUN, Y. ir kt. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* [interaktyvus]. 1998. Vol. **86**(11), p. 2278–2324 [žiūrėta 2019-04-15]. doi: 10.1109/5.726791
58. LI, R. ir kt. Adaptive Graph Convolutional Neural Networks. *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)* [interaktyvus]. 2018. p. 8 [žiūrėta 2019-04-13].
59. SRIVASTAVA, N. ir kt. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* [interaktyvus]. 2014. Vol. **15**, p. 1929-1958 [žiūrėta 2019-04-13].
60. CHUNG, J. ir kt. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv:1412.3555* [interaktyvus]. 2014. [žiūrėta 2019-03-05]. Prieiga per internetą: http://arxiv.org/abs/1412.3555.
61. CHO, K. ir kt. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv:1406.1078* [interaktyvus]. 2014. [žiūrėta 2019-03-05]. Prieiga per internetą: http://arxiv.org/abs/1406.1078.
62. General Python FAQ — Python 3.7.3 documentation [interaktyvus]. [žiūrėta 2019-04-30]. Prieiga per internetą: https://docs.python.org/3/faq/general.html.
63. Keras Documentation [interaktyvus]. [žiūrėta 2019-04-30]. Prieiga per internetą: https://keras.io/.
64. TensorFlow [interaktyvus]. [žiūrėta 2019-04-30]. Prieiga per internetą: https://www.tensorflow.org/.
65. Matplotlib: Python plotting — Matplotlib 3.0.3 documentation [interaktyvus]. [žiūrėta 2019-04-30]. Prieiga per internetą: https://matplotlib.org/.
66. Python Data Analysis Library — pandas: Python Data Analysis Library [interaktyvus]. [žiūrėta 2019-04-30]. Prieiga per internetą: http://pandas.pydata.org/.
67. NumPy [interaktyvus]. [žiūrėta 2019-04-30]. Prieiga per internetą: http://www.numpy.org/.
68. SciPy.org [interaktyvus]. [žiūrėta 2019-04-30]. Prieiga per internetą: https://www.scipy.org/.
69. RDKit [interaktyvus]. [žiūrėta 2019-04-30]. Prieiga per internetą: https://www.rdkit.org/.
70. MolVS: Molecule Validation and Standardization — MolVS 0.1.1 documentation [interaktyvus]. [žiūrėta 2019-04-30]. Prieiga per internetą: https://molvs.readthedocs.io/en/latest/.

Priedai

1 priedas. Modelių, su skirtingais latentiniais ilgiais, mokymo tikslumas

Epochos	L56_validacijos	L156_validacijos	L196_validacijos	L254_validacijos	L56_mokymo	L156_mokymo	L196_mokymo	L254_mokymo
0	0.712832651	0.7210809635127196	0.719145599	0.716374829	0.768026125	0.7728078458181602	0.772488706	0.768252206
1	0.765312722	0.7856410560520014	0.774889878	0.770039885	0.857291754	0.8701754874348091	0.860936233	0.862441821
2	0.805117823	0.8312866940476378	0.829748156	0.830869447	0.88841566	0.910740787624031	0.90233606	0.901864291
3	0.819259702	0.8806482325501156	0.8600206	0.860866675	0.902704911	0.9327739019364623	0.922862229	0.926356045
4	0.854398904	0.9033630031594483	0.881420854	0.889275988	0.916552534	0.9477635423647582	0.940575433	0.92753328
5	0.857840496	0.9213493944313119	0.907697309	0.899196333	0.925375697	0.9555002558371744	0.951363758	0.942001946
6	0.882754578	0.9394086272485794	0.917256481	0.915270523	0.932477146	0.9630721184881418	0.956566178	0.954459357
7	0.89309612	0.9473887236986293	0.910477222	0.930718994	0.938237587	0.9681590958796461	0.963656759	0.962241047
8	0.898383781	0.9121647941352036	0.944744131	0.941219941	0.939693362	0.9729141657070447	0.968030446	0.967355862
9	0.90527033	0.9609493841224003	0.947723072	0.946791353	0.945108653	0.9743545288558624	0.972197049	0.972824661
10	0.905013089	0.9656381466971015	0.945828245	0.958569037	0.946290768	0.9781674154892496	0.970953605	0.976376823
11	0.915519832	0.9632791354908922	0.958948189	0.964116063	0.947980714	0.9802346827430842	0.977455515	0.979623493
12	0.922308242	0.9704152952690828	0.961773204	0.967890481	0.951650761	0.9818764042073009	0.979107023	0.977708717
13	0.923193626	0.9713954796439491	0.966765838	0.972674946	0.944154115	0.9835159949809542	0.980694865	0.983489208
14	0.931842722	0.9777937520484221	0.96186921	0.97299649	0.954382995	0.9847653268608019	0.981924255	0.984011704
15	0.930305413	0.9779656496465481	0.973271709	0.976782185	0.95491267	0.9859604765010136	0.983392522	0.984146147
16	0.930627867	0.9801015603926874	0.974200986	0.970380852	0.955688103	0.9856397121976842	0.984001917	0.986769978
17	0.931564764	0.9818497876967153	0.974755688	0.978648973	0.957790556	0.9875035181267714	0.985810425	0.985165914
18	0.931564765	0.9820283868345797	0.976847711	0.979847371	0.957790557	0.9875692152635172	0.986501975	0.988892442
19	0.931564766	0.9812070073619965	0.979260055	0.981103379	0.957790558	0.986438646447152	0.986822773	0.989221234

2 priedas. Modelių, su skirtingais latentiniais ilgiais, tikslo funkcija

Epochos	L56_tikslo_fun	L156_tikslo_fun	L196_tikslo_fun	L254_tikslo_fun	L56_val_tikslo_fun	L156_val_tikslo_fun	L196_val_tikslo_fun	L254_val_tikslo_fun
0	0.793590829	0.7696721837939328	0.775344666	0.789603879	4.628582882	4.495636001709969	4.526815621	4.571470201
1	0.452806349	0.411764069086945	0.439765326	0.438216393	3.782710138	3.4550418524148827	3.628340172	3.706517616
2	0.345381204	0.27721016305840696	0.304338783	0.305784419	3.141128604	2.7193322225649785	2.74413051	2.726062375
3	0.297631397	0.20663009715184028	0.239004261	0.226271559	2.913184197	1.9237192374769994	2.256192211	2.242565015
4	0.252700013	0.15890216123535886	0.182240422	0.222508727	2.346808153	1.557600913509246	1.911261539	1.784637332
5	0.224789655	0.1346227769447606	0.147822607	0.177021918	2.29133651	1.2677048796332926	1.4877457	1.624765641
6	0.202671004	0.11090507852729015	0.131493527	0.13731457	1.889774828	0.9766103930187665	1.333665782	1.365666665
7	0.184624391	0.09550641487217597	0.109372806	0.113251344	1.72308952	0.8479969204845517	1.442945321	1.116667886
8	0.179915173	0.08112265093321686	0.096023917	0.097999277	1.637858848	1.41573041155591	0.890624322	0.947419228
9	0.163138223	0.0766790126871434	0.083303223	0.081552916	1.526867432	0.6294325209707708	0.842616307	0.857630724
10	0.15941387	0.06550806055214548	0.087647402	0.071148794	1.53101629	0.5538607556973735	0.873155371	0.667805507
11	0.154517986	0.059332037421827494	0.067763785	0.061412583	1.361666701	0.5918822855718674	0.661691812	0.578392757
12	0.143379206	0.05438249564886521	0.06271679	0.067788622	1.252265981	0.47687097710947834	0.616168764	0.517575397
13	0.166593756	0.04948773694783068	0.057978584	0.049956359	1.237996748	0.461070636862434	0.535696731	0.440462267
14	0.135216757	0.04579746178210667	0.054322527	0.04834891	1.098604882	0.3579597496491973	0.614641991	0.435297811
15	0.133834901	0.04234833564061845	0.049948938	0.048169831	1.123402366	0.3551978581512029	0.430876387	0.374311663
16	0.131573297	0.04351829961171165	0.048148362	0.040156057	1.118244965	0.3208061734102838	0.415937935	0.477540297
17	0.125299391	0.037933802732857326	0.042881594	0.045297169	1.103171936	0.2926713479554049	0.407049783	0.344340224
18	0.125299391	0.03780195679891372	0.040861595	0.034027802	1.103171936	0.28987263870953417	0.373406927	0.325127164
19	0.125299391	0.041627650594084124	0.040208594	0.03311228	1.103171936	0.3032212087086269	0.334629705	0.305012642

3 priedas. Modelių, su skirtingais molekulių ilgiais, mokymo tikslumas

Epochos	M60_validacijos	M80_validacijos	M100_validacijos	M120_validacijos	M60_mokymo	M80_mokymo	M100_mokymo	M120_mokymo
0	0.7459227790212417	0.7346802722712803	0.734365598236	0.719145599	0.7904565987447878	0.7874925447634424	0.755130629227	0.772488706
1	0.7916274530470638	0.787174278845167	0.787539963765	0.774889878	0.8831257202018196	0.8762703990864718	0.863207108113	0.860936233
2	0.8411992403423839	0.8128033547658022	0.81848614713	0.829748156	0.9181336723845741	0.9008041872806464	0.905278123159	0.90233606
3	0.8734696665152306	0.8430508003641137	0.861987815042	0.8600206	0.9367019068067702	0.9225248953233426	0.927320352431	0.922862229
4	0.8965413188185927	0.8616265798898021	0.893866153576	0.881420854	0.950572182501239	0.9364663405141692	0.941773528543	0.940575433
5	0.9227214388783203	0.8894450234190765	0.909684921327	0.907697309	0.9614572333895963	0.9463677946002916	0.953538120449	0.951363758
6	0.9320168925507721	0.9050469743296704	0.926978573938	0.917256481	0.9565760193615331	0.9553099080226969	0.961943855221	0.956566178
7	0.9454149547713755	0.9161714473112816	0.94280771263	0.910477222	0.9725821903790755	0.9635289410700376	0.967765556716	0.963656759
8	0.9443781079198212	0.9341077906133882	0.945631463431	0.944744131	0.9768829874660803	0.9689050158779283	0.972962307149	0.968030446
9	0.9572455850951874	0.9226914900835319	0.960643521072	0.947723072	0.9803646236374832	0.9729582037312678	0.977110779721	0.972197049
10	0.967767701822546	0.9545360204884824	0.965358571087	0.945828245	0.9821132824026149	0.9768986704052538	0.979513073725	0.970953605
11	0.9721846887883584	0.9566811935249465	0.969222733258	0.958948189	0.9851263658054594	0.9806403913695911	0.982609629273	0.977455515
12	0.9724785955497502	0.9639432828522584	0.972412465399	0.961773204	0.986853519399623	0.9823205957059684	0.984163548333	0.979107023
13	0.9788206917288057	0.9688508120352912	0.977697538955	0.966765838	0.9882298044528647	0.9845571979515072	0.983435506788	0.980694865
14	0.9747521962820147	0.9721114301360776	0.978688412985	0.96186921	0.9898742110863992	0.9844081172828617	0.984750746041	0.981924255
15	0.9761535370296427	0.9736275525905627	0.980270085313	0.973271709	0.9904527089606529	0.9867232960423807	0.987873507238	0.983392522
16	0.9852962427075134	0.9750520405213394	0.981252955749	0.974200986	0.9911486593826584	0.9881019293218329	0.988303650016	0.984001917
17	0.965229556164934	0.9788100134096873	0.982015465942	0.974755688	0.9872415428939254	0.986881607887207	0.988742526619	0.985810425
18	0.9868936340905091	0.9768024573411642	0.982479318642	0.976847711	0.9896246490626409	0.9892251758709021	0.989460975543	0.986501975
19	0.990057562231483	0.980235091506633	0.982816529648	0.979260055	0.9935062524973004	0.9866210954257284	0.990179258415	0.986822773

4 priedas. Modelių, su skirtingais molekulių ilgiais, tikslo funkcija

Epochos	M60_tikslo_fun	M80_tikslo_fun	M100_tikslo_fun	M120_tikslo_fun	L60_val_tikslo_fun	L80_val_tikslo_fun	L100_val_tikslo_fun	L120_val_tikslo_fun
0	0.7044058180767754	0.7142150201548213	0.790951839025	0.775344666	4.095238897297949	4.276436915846683	4.28151360969	4.526815621
1	0.3620431404253314	0.3831135553679149	0.421465319908	0.439765326	3.3585630927919805	3.430344381674523	3.42442153388	3.628340172
2	0.2450526554161695	0.2970732161705347	0.285713271566	0.304338783	2.559566144986003	3.0172578743220444	2.92565734932	2.74413051
3	0.18671494067878114	0.22837084493767087	0.216669175045	0.239004261	2.039424313023486	2.5297276214633824	2.22449437546	2.256192211
4	0.14488093048826106	0.18635828184031916	0.173264707787	0.182240422	1.667549078774559	2.230302927205381	1.71067712393	1.911261539
5	0.1128770991518773	0.15693311764112766	0.137551802793	0.147822607	1.2455807956344878	1.7819426423231048	1.45571423111	1.4877457
6	0.13080472125298623	0.13065361540173817	0.112429096818	0.131493527	1.0957613838627733	1.530464875858461	1.17696456123	1.333665782
7	0.08131355178421769	0.10705075204640761	0.0957209724055	0.109372806	0.8798064322215025	1.3511461719803746	0.921839582813	1.442945321
8	0.0687487433632354	0.09170312865666594	0.0806233541609	0.096023917	0.8965103393712921	1.0620574673195056	0.876322496098	0.890624322
9	0.05864430251174894	0.08024453977612092	0.0687032389553	0.083303223	0.6891278715411644	1.2460687721791288	0.63435383268	0.842616307
10	0.05358581582066594	0.06877113640189111	0.0619910059943	0.087647402	0.5195343555890926	0.7328035558820305	0.55836390313	0.873155371
11	0.04482959469164414	0.057941924176808296	0.0528693160954	0.067763785	0.44834099596391347	0.6982219572024495	0.496080430634	0.661691812
12	0.03966507121357785	0.05305083480669654	0.0482759261229	0.06271679	0.44360699260715947	0.5811824680978407	0.444674861939	0.616168764
13	0.03572202559510221	0.04663425330454675	0.0507535921403	0.057978584	0.34140353820249103	0.5020976641252971	0.359501189448	0.535696731
14	0.030786932118538918	0.04722838754439097	0.0467563009485	0.054322527	0.40700536498574397	0.44956714984012824	0.343558747672	0.614641991
15	0.029190330713495367	0.04036243611656916	0.0373539135742	0.049948938	0.3844311039544007	0.4251478708645688	0.318082453758	0.430876387
16	0.027136501156840995	0.036283384015287024	0.0360014803368	0.048148362	0.2371087060915515	0.4022312189698754	0.302287006926	0.415937935
17	0.04036675039315415	0.04000958738165566	0.0347169295527	0.042881594	0.5606314205802609	0.34171913022952227	0.290054598063	0.407049783
18	0.03217039138298591	0.03305088342533164	0.0327783304883	0.040861595	0.21154775009427904	0.3741869071807562	0.282652143539	0.373406927
19	0.020523500876159236	0.04149962869053963	0.0306365569739	0.040208594	0.16070021592047182	0.3190196264752358	0.277351362422	0.334629705