Proteiinien laskostusongelman ratkaiseminen syväoppivan neuroverkon (AlphaFold3) avulla

Mervi Tenhami

LuK -tutkielma
Toukokuu 2026

MATEMATIIKAN JA TILASTOTIETEEN LAITOS

Ohjaaja:
Aleksi Winstén

TURUN YLIOPISTO
Matematiikan ja tilastotieteen laitos

Mervi Tenhami: Proteiinien laskostusongelman ratkaiseminen syväoppivan neuroverkon (AlphaFold3) avulla
LuK -tutkielma, 25 s.
Matematiikka
Toukokuu 2026

Proteiinit ovat eliöiden rakenteen ja biologisten reaktioiden keskeisiä osatekijöitä. Proteiinimolekyylejä käytetään myös lääkkeinä sekä osana teollisuuden kemiallisia prosesseja. Proteiinit koostuvat yhdestä tai useammasta aminohappoketjusta, joka laskostuu kolmiulotteisesti muodostaen toiminnallisesti aktiivisen molekyylin. Proteiinin kolmiulotteisen rakenteen selvittäminen mahdollistaa molekyylin toiminnan ymmärtämisen ja sen mahdollisen muokkaamisen myös haluttua prosessia varten.

Proteiinien rakennetta on pyritty selvittämään jo 1900-luvun alkupuolelta lähtien, mutta menetelmät olivat varsin työläitä ja aikaavieviä. Tietokoneiden laskentakapasiteetin kasvaessa on kehitetty erilaisia koneoppimisalgoritmeja laskostusongelman ratkaisemisen tehostamiseksi. Perinteisillä menetelmillä selvitetyt polypeptidiketjut ja niitä vastaavat kolmiulotteiset rakenteet on tallennettu tietokantoihin ja syvät neuroverkot ovat mahdollistaneet aiemmin kertyneen datan tehokkaan käytön proteiinien laskostusongelman tutkimuksessa. V. 2024 julkaistu AlphaFold3 on syväoppivaan neuroverkkoarkkitehtuuriin perustuva tekoälymalli, joka ennustaa suurella tarkkuudella proteiinien ja niihin liittyvien ligandien kolmiulotteisen rakenteen. Keskeisiä matemaattisia osa-alueita AlphaFold3:n algoritmissa ovat datan esittämiseen ja muokkaamiseen liittyvä lineaarialgebra ja oppimisprosessiin (konvergenssi) liittyvät peräkkäiset derivoinnit ja niihin liittyvät aktivointifunktiot.

Tämän tutkielman tavoitteena on tutustuttaa lukija proteiinimolekyylin yleiseen rakenteeseen, koneoppimisen perusteisiin, AlphaFold3:n toimintaan, sekä niiden matemaattiseen taustaan.

Asiasanat: proteiinien laskostusongelma, koneoppiminen, AlphaFold3.

Sisällys

1 Johdanto

Proteiinit ovat eliöiden rakenteen ja biologisten prosessien keskeisiä osatekijöitä. Laboratorio-olosuhteissa valmistettuja proteiineja voidaan käyttää nykyisin myös lääkkeinä (insuliini ja monoklonaaliset vasta-aineet) sekä osana biologista prosessiteollisuutta (amylaasit ja sellulaasi biomassan hajotuksessa). Proteiinien toiminta perustuu niiden kolmiulotteiseen rakenteeseen, jonka mallintaminen on olennaista molekyylin toiminnan ymmärtämisen kannalta. Kolmiulotteisen rakenteen selvittäminen luo myös edellytykset rakenteen rationaaliseen muokkaamiseen haluttua toimintaa vastaavaksi. [1, 6]

Proteiinimolekyylin kolmiulotteisen rakenteen ennustamista sen aminohappojärjetyksen perusteella sanotaan proteiinin laskostumisongelmaksi (the protein folding problem). Kolmiulotteisen rakenteen määrittäminen perinteisillä kokeellisilla menetelmillä on ollut työlästä ja se on vaatinut paljon aikaa. Koneoppiminen on tuonut tehokkaan apuvälineen myös laskostumisongelman ratkaisemiseen. AlphaFold (v. 2018) ja RoseTTAFold (v. 2021) olivat ensimmäiset syväoppimiseen perustuvat järjestelmät, jotka ennustivat suurella tarkkuudella annetun polypeptidiketjun kolmiulotteisen rakenteen. Vuonna 2024 julkaistu syväoppivan neuroverkon toimintaan perustuva AlphaFold3 -arkkitehtuuri pystyy lisäksi ennustamaan proteiinikompleksin ja siihen sitoutuneen ligandin kolmiuloitteisen rakenteen pelkästään polypeptidin aminohappojärjestyksen perusteella. [1, 6]

2 Proteiinien rakenne ja toiminta

Proteiinit koostuvat aminohappojen muodostamista polypeptidiketjuista, jotka sisältävät yleensä 50-300 aminohappotähdettä. Proteiinit toimivat eliöissä rakenneosina, kuljetustehtävissä, katalyytteinä, välittäjäaineina ja immunologisissa puolustustehtävissä vasta-aineina ja komplementin osina. Proteiinin kolmiulotteinen rakenne voi olla pallomainen (pyöreä, elliptinen tai yleisesti tiivis) tai kuitumainen. Pallomaiset proteiinit, kuten elimistön toimintoja katalysoivat entsyymit ja polypeptidihormonit, ovat vesiliukoisia ja siten helpommin inaktivoituvia kuin kuitumaiset proteiinit, kuten sidekudosten kollageeni ja solun tukirangan muodostukseen osallistuva keratiini. Proteiinin rakenne liittyy olennaisesti sen biologiseen tehtävään ja vähäisetkin muutokset proteiinin rakenteessa voivat muuttaa sen toimintaa merkitsevästi (de-naturoituminen). [4, 11]

2.1 Sidokset ja termodynamiikkaa

Proteiinin muodostava polypeptidiketju pysyy koossa atomien välisten kovalenttisten sidosvoimien avulla. Polypeptidiketjun laskostumiseen, eli proteiinin kolmiulotteisen rakenteen muodostumiseen, vaikuttavavat sekä ei-kovalenttiset hydrofobiset ja hydrofiiliset voimat, että kovalenttiset aminohappotähteiden sivuketjujen välille muodostuvat rikkisillat. Aminohappotähteiden hiili-, happi-, vety-, typpi- ja rikkiatomien välinen kovalenttinen sidosenergia vaihtelee 259 - 414 kJ/mol välillä. Heikompi ei-kovalenttinen sidosenergia puolestaan vaihtelee yleensä 0,5 - 40 kJ/mol välillä. Tämän seurauksen heikommat sidosvoimat ovat helpommin kumottavissa, mikä

johtaa sidosten avautumiseen ja proteiinin kolmiulotteisen rakenteen muuttumiseen. Aminohappotähteiden välillä vaikuttavat kovalenttiset sidokset vaativat yli kymmenen kertaa suuremman energian avautuakseen ja tämän vuoksi proteiinin aminohappojärjestys on huomattavasti vakaampi kuin sen kolmiulotteinen rakenne.[4, 11]

Proteiinin vallitseva kolmiulotteinen rakenne eli konformaatio

Entalpia 1 on termodynaamisen järjestelmän suure, joka mittaa systeemiin sitoutunutta kokonaisenergiaa. Se koostuu järjestelmän sisäenergioiden summasta sekä paineen ja tilavuuden tulosta.

$$H = U + PV \tag{1}$$

jossa H on entalpia, U on sisäenergia, P on paine ja V on tilavuus. Sidosentalpia eli sidosten dissosioitumisentalpia DH ilmaistaan usein standardisena suureena ja sen yksikkö on $kJmol^{-1}$. Esimerkiksi $DH(A-B)$ tarkoittaa energiaa, joka tarvitaan kaasumaisen AB-molekyylin dissosioitumiseen A:ksi ja B:ksi, jossa A ja B ovat atomeja tai radikaaleja [17]. Atomien ja molekyylien välisten sidosten muodostuminen puolestaan vapauttaa vastaavan määrän energiaa ja tällöin molekyylim energiasisältö ja entalpia pienenee. Molekyylin laskostuminen ja sidosten muodostuminen vapauttaa siis energiaa ja molekyyli saavuttaa tällöin ympäristössään yleensä sen muodon, jonka energiasisältö eli entalpia on mahdollisimman pieni.

Proteiinin eri konformaatioiden monimuotoisuutta, konformaatioiden suhteellisia todennäköisyyksiä, toiminnallisia muutoksia ja vuorovaikutusalttiutta voidaan tutkia esim. proteiinien konformaatioiden otannalla (protein conformation sampling). Menetelmällä luodaan rakenteellisia konfiguraatioita, jotka kuvastavat proteiinin konformaatioiden jakaumaa. Metodi tuottaa optimitilassa konformaatiot x:

$$p_s(x) \approx p_{eq}(x) \tag{2}$$

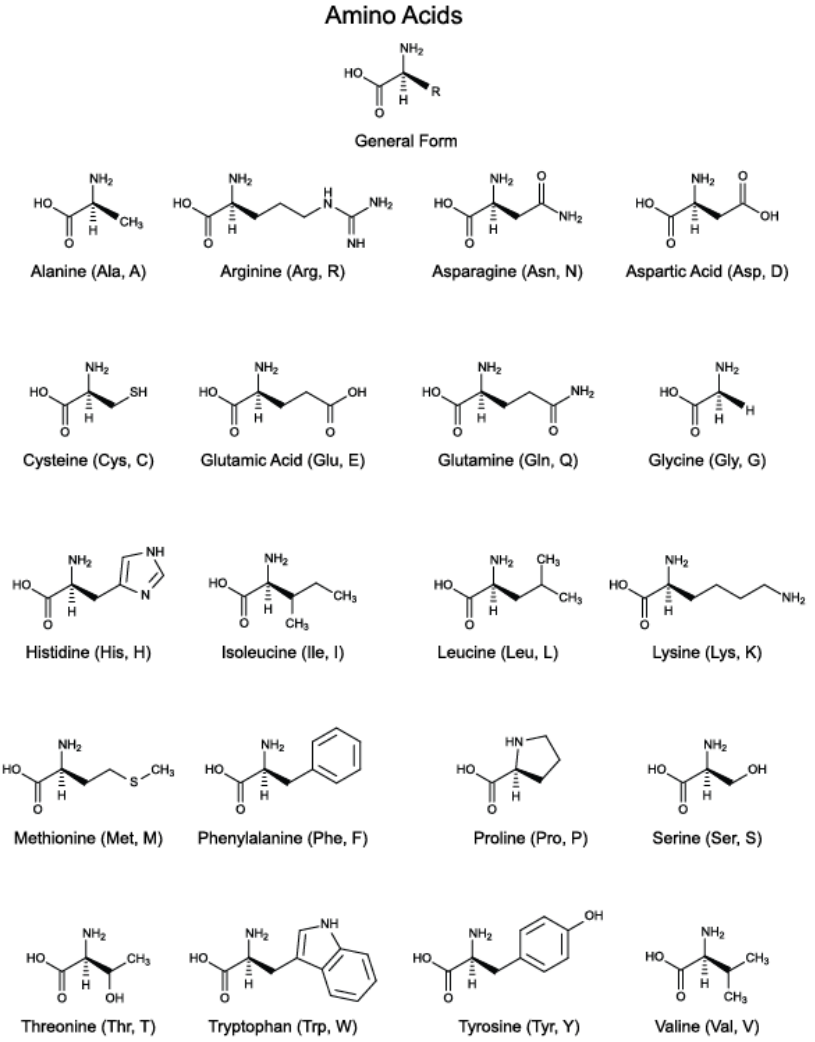
jossa $p_s(x)$ on menetelmän indusoima näytejakauma ja $p_eq(x)$ on proteiinin konformaatioiden todellinen jakauma tasapainotilassa. Tämän määrittämiseen käytetään perinteisiä molekulaarisen dynamiikan (MD) menetelmiä sekä Monte Carlo -simulaatiota. Tavoitteena on luoda edustava joukko proteiinin eri konformaatioita $x_1, x_2, ..., x_N$, joiden välistä tasapainojakaumaa voidaan kuvata Bolzmannin tasapainojakaumalla:

$$p_eq(x) = \frac{1}{Z} e^{-\beta E(x)} \tag{3}$$

jossa $E(x)$ on konformaation x potentiaalienergia, $\beta = 1/(kBT)$ on sisälämpöenergia ja Z on normalisointiin käytettävä jakaja. Yhtälön (3) perusteella konformaation todennäköisyys pienenee eksponentiaalisesti systeemin energian kasvaessa eli matalaenergiset konformaatiot ovat eksponentiaalisesti todennäköisempiä kuin korkeaenergiset muodot. [2]

2.2 Proteiinien rakenne

Proteiinien rakenne luokitellaan polypeptidiketjun aminohappojärjestystä tarkoittavaan primäärirakenteeseen, ketjun kolmiulotteisen muodon määrittämään sekundäärirakenteeseen, sekundäärirakenneyksiöiden laskostumisen tuloksena muodostuvaan tertiääriseen rakenteeseen ja useiden laskostuneiden polypeptidiketjujen yhteenliittymisen seurauksena muodostuvaan kvartenääriseen rakenteeseen.



Kuva 1: Eliöiden 20 essentiaalista aminohappoa: päärunko on esitetty ylimpänä ja sivuketjujen kuvat aakkosjärjestyksessä sen alla. https://sciencenotes.org/printable-20-amino-acids-study-sheet/

2.2.1 Primäärirakenne

Proteiinin primäärirakenne tarkoittaa sen polypeptidiketjun aminohappojärjestystä. Taksonomisessa luokittelussa kehittyneempien eliöiden aminohapot ovat stereokemialliselta rakenteeltaan α-aminokarboksyylihappoja, joiden α-hiileen sitoutunei-

den atomien ja ryhmien rakenne vastaa kiraalisuudeltaan L-(-)glyseraldehydin rakennetta. Ihmiselimistössä on 20 erilaista aminohappoa, joiden rakenteet on esitetty kuvassa 1. Aminohapot voivat sitoutua toisiinsa karboksyylihapporyhmän ja aminoryhmän välisellä amidisidoksilla (peptidisidos), jolloin ne muodostavat polypeptidiketjun. Proteiinien polypeptidiketjut sisältävät yleensä 50 - 300 aminohappotähdettä ja erilaisten n kpl aminohappotähteitä sisältävien ketjujen määrä on tällöin 20^n. Esimerkiksi 100 aminohappotähdettä sisältävä ketju voidaan muodostaa 20^{100} erilaisella tavalla. [4, 11]

Elimistön proteiinien polypeptidiketjujen aminohappojärjestys määräytyy solun tumassa sijaitsevien geenien nukleiinihappojärjestyksen perusteella ja tieto tästä järjetyksestä siirtyy tumasta solulimaan ribosomeille transkriptiossa muodostetun lähetti-RNA:n välityksellä. Ribosomeilla lähetti-RNA:n sisältämä informaatio muutetaan aminohappojen muodostamaksi polypeptidiketjuksi (translaatio). [11]

2.2.2 Sekundäärinen rakenne

Translaatiossa muodostunut polypeptidiketju laskostuu aminohappojärjestyksensä ja liuosympäristön perusteella joko kierteiseksi α-heliksiksi, kuitumaiseksi β-levyksi tai lenkkimäiseksi (loop) -rakenteeksi (Kuva 2). Globulaarisissa proteiineissa olevista aminohapoista n. 50 - 80 % ja kuitumaisissa proteiiniessa lähes kaikki aminohapot sijaitsevat näissä sekundäärisissä rakenteissa.[11]

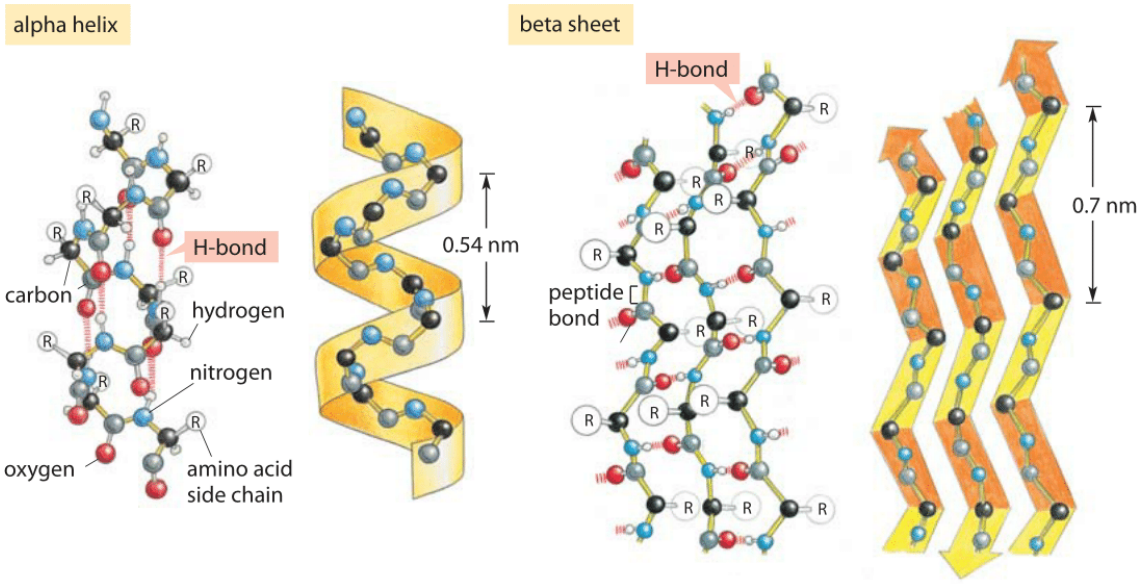
α-helikseissä rungon N-H -ryhmä sitoutuu vetysidoksella rungon C=O -ryhmään, joka sijaitsee neljä aminohappotähdettä aikaisemmin polypeptidiketjussa. Nämä heliksin akselin suuntaiset vetysidokset stabiloivat kierteisen molekyylirakenteen. Aminohappotähteiden sivuketjut suuntautuvat puolestaan kierteen ulkopuolelle.

β-levyn muodostavat polypeptidiketjun juosteet sitoutuvat toisiinsa viereisten juosteiden amidisidosten vety- ja happimolekyylien kanssa muodostuvien vetysidosten avulla. Aminohappotähteiden sivuketjut suuntautuvat tällöin levyn tasosta joko ylös- tai alaspäin. Aminohappojen sivuketjun varaus ympäröivässä liuoksessa riippuu sen aktiivisen ryhmän isoelektrisestä pisteestä. Sivuketjun varaus vaikuttaa puolestaan sekundäärirakenteiden laskostumiseen ja muodostuvan kolmiulotteisen rakenteen pysyvyyteen joko kovalenttisen sidosmuodostuksen tai heikkojen vuorovaikutusten kautta (hydrofobiset voimat vesiliuoksessa).[11]

2.2.3 Tertiäärinen rakenne

Proteiinin tertiäärisellä rakenteella tarkoitetaan koko polypeptidiketjun kolmiulotteista laskostumista, jossa edellä muodostuneet α-kierteet ja β-levyt laskostuvat suhteessa toisiinsa. Liuosympäristössä laskostuu yleensä joko termodynaamisesti vakain tai kineettisesti helpoiten saavutettavissa oleva rakenne. Proteiinien laskostumista voivat ohjata myös toiset proteiinit, joita kutsutaan kaperoneiksi (chaperon). Yleensä vain yksi laskostumismuoto on biologisesti aktiivinen eli se toimii ympäristössään tarkoituksenmukaisella tavalla. Jos proteiinin laskokset avautuvat, se denaturoituu ja menettää tällöin biologisen aktiivisuutensa. [11]

Globulaarisen proteiinin poolittomat alueet (hydrofobiset aminohappotähteiden sivuketjut) sijaitsevat vesiliuoksissa yleensä molekyylin sisäosissa ja polaariset sivuketjut molekyylin ulkopinnalla vuorovaikutuksessa vesimolekyylien kanssa. Amino-



Kuva 2: Vasemmalla on esitetty polypeptidiketjun kierteinen α-heliksi ja oikealla tasomainen β-levy. Atomit on värikoodattu seuraavasti: hiili = musta, happi = punainen, typpi = sininen, vety = valkoinen. R = aminohapon sivuketju. (https://book.bionumbers.org/wp-content/uploads/2014/08/310-f1-AlphaBeta-1.png; kuva vain esimerkki vielä)

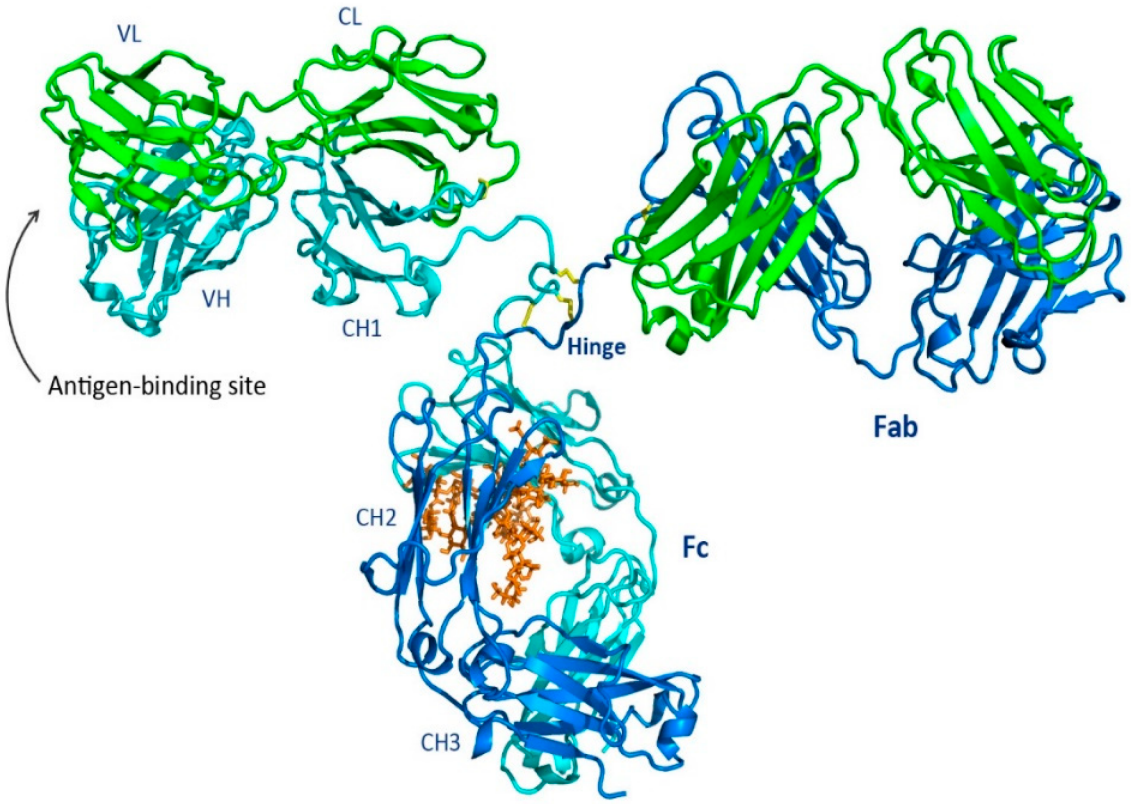
hapot voidaan jakaa sivuketjun polaarisuuden perusteella sisäisiin (Ile, Leu, Met, Phe, Val), ulkoisiin (Arg, Asn, Asp, Gln, Glu, His, Lys) ja vaihteleviin (Ala, Cys, Gly, Pro, Ser, Thr, Trp, Tyr) aminohappoihin viitaten niiden sijaintiin pallomaisessa proteiinissa. [11]

Globulaarisen proteiinin itsenäisesti laskostuvaa alayksikköä sanotaan domeeniksi. Yksi domeeni koostuu yhdestä tai useammasta rakenneaihiosta eli motiivista, jossa tietyt sekundäärirakenteen elementit laskostuvat proteiineissa hyvin samalla tavoin muodostaen geometrisesti samankaltaisia rakenteita. Motiiveita ovat esim. heliksi-loop-heliksi -rakenne, joka nimensä mukaisesti muodostuu kahdesta α-kierteestä ja niitä yhdistävästä lenkkimäisestä polypeptidirakenteesta sekä β-tynnyri, joka muodostuu kahdeksasta vierekkäisestä putkimaiseen muotoon asettuvasta β-levystä. Motiiveissa toistuvien samankaltaisten aminohapposekvenssien ja niille tyypillisten kolmiulotteisten laskostumismuotojen takia niitä voidaan käyttää proteiinien 3D-rakenteen selvittämisessä. [11]

2.2.4 Kvartenäärinen rakenne

Proteiinin kvartenäärinen rakenne on kokonaisuus, jossa useita polypeptidiketjuja on liittynyt yhteen vetysidoksin, hydrofobisin voimin tai suolasidoksin. Biologisesti aktiivinen kvarternäärinen rakenne edustaa todennäköisesti termodynaamisesti vakainta alayksiköiden yhteenliittymää. Sekundaarinen, tertiäärinen ja kvartenäärinen rakenne muodostavat siis kokonaisuutena proteiinin konformaation. Kuvassa 3 on esimerkkinä kvartenäärisestä rakenteesta hiiren immunoglobuliini G2 -luokan

(IgG2) vasta-ainemolekyyli. joka koostuu neljästä yhteenliittyneestä polypeptidiketjusta, jotka ovat liittyneet toisiinsa kovalenttisin rikkisilloin. [11]



Kuva 3: Hiiren immunnoglobuliini IgG2 -molekyyli, joka koostuu neljästä yhteen liittyneestä polypeptidiketjusta. Sekundäärirakenteista voidaan nähdä useita beetalevyjä, muutamia alfa-kierteitä ja useita samankaltaisia motiiveja. (https://www.mdpi.com/2073-4468/8/4/55)

2.3 Proteiinien rakenteen määrittämisessä käytettyjä menetelmiä

Proteiinien rakenteiden selvittämiseksi ja mallintamiseksi on käytetty perinteisesti röntgenkristallografiaa, ydinmagneettiresonanssi(NMR) -spektroskopiaa ja kryoelektronimikroskopiaa, jotka ovat luoneet pohjan atomitason mallien rakentamiselle. Röntgenkristallografia perustuu röntgensäteiden diffraktioon kiteisessä proteiinissa, ja sitä on käytetty proteiinien atomitason mallintamiseen 1950-luvulta alkaen. NMR-spektroskopia perustuu atomien ydinten energiatasojen muutoksiin magneettikentässä liuosympäristössä. Ytimillä on erilaisissa ympäristöissä erilaiset resonanssitaajuudet, joiden avulla voidaan selvittää niiden kytkeytymistä viereisiin atomeihin. Kryoelektronimikroskopiassa proteiinikompleksin rakennetta voidaan tutkia natiivissa muodossa elektronisuihkun avulla hyvin kylmissä olosuhteissa. [8, 15, 18]

Myöhemmin kehitettyjä mallinnusmenetelmiä ovat pienimmän sisäenergian saavuttamiseen perustuvat menetelmät ja vertaileva mallinnus. Pienimmän sisäener-

gian saavuttamiseen perustuvia menetelmiä on esim. *build up* -menetelmä, jossa polypeptidi rakennetaan asteittain aminohapporakenneyksiköistä niiden atomien välisten sidosten sallittuihin kiertokulmiin perustuvan Ramachandran kartan avulla. Kehittyneemmissä malleissa huomioidaan myös elektrostaattiset vuorovaikutukset molekyylien välillä. *Hilamallissa* proteiini mallinnetaan sarjana hydrofobisia ja hydrofiilisiä monomeerejä. Sarja rakennetaan kaksiulotteiselle hilalle ja muodostuvan konformaation energia lasketaan summaamalla vierekkäisten ei-kovalenttisesti sitoutuneiden monomeeriparien vuorovaikutukset. *Monte Carlo -simulaatiossa* proteiinien kolmiulotteisia rakenteita luodaan satunnaisotantamenetelmällä ja lasketaan rakenteiden sisäenergia. *Sääntöpohjaisissa malleissa* etsitään puolestaan ensin ne polypeptidiketjun alueet, jotka aminohappojärjetyksensä perusteella muodostavat tunnettuja sekundäärirakenteita ja määritetään näiden perusteella vierekkäin asettuvat rakenteet (esim. alfaheliksi asettuu beetalevyä vasten). [8]

Vertailevaan mallinnukseen perustuvissa menetelmissä pyritään määrittämään, mihin proteiiniperheeseen tutkittava proteiini kuuluu. Menetelmässä vertaillaan polypeptidiketjun aminohappojärjestystä tietokannoissa oleviin yhdistelmiin. Tietyt rakennemotiivit viittaavat proteiinissa tiettyihin toimintoihin ja rakenteellisiin piirteisiin. *Sequence alignment -algoritmissa* asetetaan eri proteiinien samankaltaiset aminohapposekvenssit rinnakkain. Ne vastaavat proteiineissa yleensä samankaltaisia rakenteita tai toimintoja. Nämä menetelmät ovat kuitenkin olleet laskennallisesti työläitä ja sen vuoksi on kehitetty myös heuristisia menetelmiä kuten BLAST (Basic logical alignment search tool) ja FASTA (FAST-AII). [8]

Edellä kuvatuilla menetelmillä mallinnettujen proteiinien aminohapposekvenssit ja kolmiulotteiset rakenteet on tallennettu Protein Data Bank -tietokantaan, josta niitä on voitu jatkossa hyödyntää muiden samankaltaiseksi oletettujen proteiinien rakenneyksiköiden selvittämiseen [6].

3 Koneoppimisesta

3.1 Yleistä

Tekoälyllä tarkoitetaan ohjelmakoodia ja tietokonearkkitehtuuria, joka oppii syötteenä saamastaan datasta ja tuottaa siitä tulosteen. Tekoälyn käytön tavoitteena on automatisoida ja nopeuttaa tiedon tulkintaa ja kehittää tehokkaita sovelluksia esim. teollisuuden ja liiketoiminnan tarpeisiin.[16]

Koneoppiminen on tekoälyn osa-alue, jossa ohjelmiston toimintaa optimoidaan ensin pohjatiedon ja sitten käyttäjän toistuvan toiminnan perusteella. Koneoppiminen on perinteisesti luokiteltu ohjattuun, ohjaamattomaan ja vahvistettuun oppimiseen, mutta nykyään kaikki mallit eivät enää sovi tähän luokitteluun.[16]

Neuroverkot ovat yksi koneoppimiseen osa-alue. Ne ovat laskennallisista neuroneista (kts. perseptroni 3.3.1) ja niiden välisistä kytkennöistä muodostuva arkkitehtuuri, joka prosessoi siihen syötettävän datan matemaattisen mallin mukaisesti halutuntyyppiseksi tulosteeksi. Laskennallinen neuroni saa yhden tai useamman syötesignaalin eli lukuarvon ja lähettää eteenpäin yhden lukuarvon. Neuroverkko koostuu syötekerroksesta (input layer), yhdestä tai useammasta piilokerroksesta (hidden layer) ja ulostulokerroksesta (output layer) (kts. Kuva 4), ja se vaatii toimiakseen

suorituskykyisen laitteistoarkkitehtuurin. Neuroverkkoja käytetään esim. kuvan- ja puheentunnistuksissa sekä käännössovelluksissa. Neuroverkkojen toiminnan taustalla olevia matematiikan osa-alueita ovat mm. vektorianalyysi, lineaarialgebra ja todennäköisyyslaskenta, ja neuroverkkojen ohjelmointiin käytettyjä johtavia ohjelmointikirjastoja ovat esim. PyTorch ja TensorFlow. [3, 5, 16, 24]

3.2 Mallit

Koneoppimismalli koulutetaan ensin tunnetulla harjoitusdatalla varsinaista myöhempää käyttöä varten. Koulutuksella pyritään optimoimaan verkon parametrit eli minimoimaan parametreista aiheutuva virhefunktio (eli kuinka kaukana yksittäinen ennuste on oikeasta vastauksesta). Virhefunktion minimoinnissa käytettävät optimointialgoritmit perustuvat usein iteratiiviseen gradienttimenetelmään, jossa lasketaan funktion derivaatta nykyisessä sijainnissa ja siirrytään sitten askel negatiivisimman gradientin suuntaan. Sijainti päivitetään siirtymän pituuden mukaan ja askelpituutta kutsutaan algoritmin oppimisnopeudeksi. Yhtälössä (4) x_n on nykyinen sijainti, η oppimisnopeus ja $\nabla f(x_n)$ funktion gradientti. Tällöin uusi sijainti saadaan seuraavasti:

$$x_{n+1} = x_n - \eta \nabla f(x_n) \tag{4}$$

Gradientti voidaan kirjoittaa pystyvektorina, joka sisältää funktion osittaisderivaatat. Esimerkiksi $\mathbb{R}^3$:ssa gradientin esitysmuoto on:

$$\nabla f(x, y, z) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \end{bmatrix} \tag{5}$$

Koneoppimisalgoritmin virhefunktio pienenee tyypillisesti sitä mukaa, kun algoritmi prosessoi lisää dataa esimerkiksi vuorovaikuttamalla ympäristönsä kanssa.[3, 16, 24]

Koneoppimisen suuntauksia ovat ohjattu oppiminen, ohjaamaton oppiminen ja vahvistusoppiminen. Ohjatussa oppimisessa koneelle annetaan syötteenä pareja, jotka muodostuvat esimerkistä ja siihen liittyvästä luokasta. Koulutusvaiheessa kone oppii ennustamaan annetun syötteen luokan esimerkkien avulla. Ohjattu oppiminen jaetaan diskreettiä dataa prosessoivaan luokitteluun ja jatkuvaa numeerista dataa prosessoivaan regressioon. Ohjaamattomassa oppimisessa koneelle annetaan pelkkiä esimerkkejä, jotka kone luokittelee ryhmiin (klusterointi). Vahvistusoppimisessa (reinforcement learning) kone oppii yrityksen ja erehdyksen kautta jatkuvassa vuorovaikutuksessa ympäristön kanssa.[3, 16, 24]

3.3 Syväoppivat neuroverkot

Mikäli neuroverkossa on useita piilokerroksia, verkkoa sanotaan syväksi neuroverkoksi. Perinteisessä koneoppimisessa ihminen määrittää datasta tunnistettavat piir-

teet (syötteet), joiden perusteella malli generoi ennusteen. Syvä neuroverkko puolestaan oppii piirteet automaattisesti. Syväoppivien neuroverkkojen koulutukseen tarvittavat datamäärät ovat isoja ja niiden prosessointi vaatii tehokkaan laskenta-arkkitehtuurin. Perinteiseen koneoppimiseen riittävät sen sijaan pienemmät datamäärät ja tavallinen tietokone. [16, 24]
Syvien neuroverkkojen yleisimpiä tyyppejä ovat:

- Konvoluutioneuroverkot (CNN, convolution neural networks), joita käytetään esim. kuvantunnistuksessa.
- Takaisinkytketyt neuroverkot (RNN, recurrent neural networks), jotka on suunniteltu sarjamuotoisen datan analysointiin.
- Muuntimet eli transformer-verkot (BERT), joiden toimintaan perustuvat esim. suuret kielimallit ja kääntäjät.
- Generatiiviset kilpailevat verkot (GAN, generative adversial networks), jossa arkkitehtuuri koostuu kahdesta keskenään kilpailevasta verkosta. GAN:ia käytetään usein uuden datan luomiseen esim. grafiikoiden suunnittelussa.

Kuvassa 4 on esitetty kaavamaisesti syötekerroksesta, yhdestä piilokerroksesta ja ulostulokerroksesta koostuva neuroverkko, jonka tuottama tulos t saadaan seuraavasti:

Ensimmäisessä vaiheessa verkko kertoo syötevektorin x komponentit piilokerroksen neuroneiden painoilla w_{ij}^1, laskee sitten tulot yhteen ja lisää summaan neuronin i vakiotermin d_i

Esimerkiksi piilokerroksen neuroniin 1 tuleva summa on tällöin:

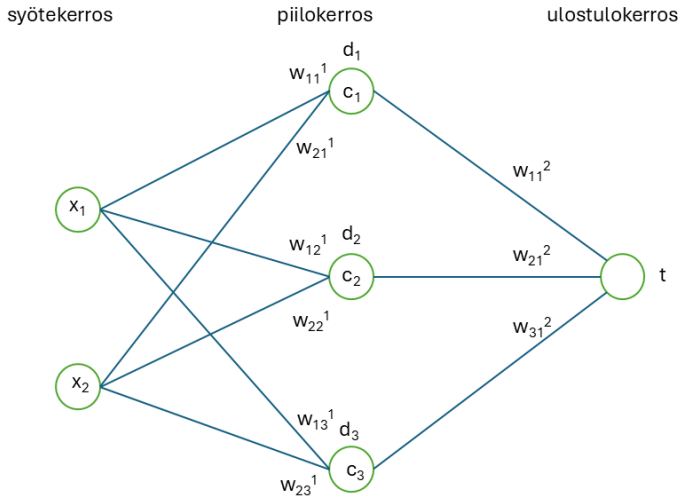
$$s_1^1 = w_{11}^1 x_1 + w_{21}^1 x_2 + d_1 = \sum_{i=1}^{2} w_{i1}^1 x_i + d_1 \tag{6}$$

Toisessa vaiheessa neuroniin tullut summa sijoitetaan aktivointifunktioon ψ (esim. ReLu, kts. kappale 3.3.2). Tällöin neuronin 1 ulostulokerrokselle antama signaali c_1 on:

$$c_1 = \psi(s_1) \tag{7}$$

Lopullinen tulos t saadaan kertomalla piilokerrosten neuronien 1-3 antamat syötteet ulostulokerroksen painoilla w_{i1}^2 ja syöttämällä niiden summa piilokerroksen ja ulostulokerroksen väliseen aktivointifunktioon:

$$t = \psi(s_1^2) = \psi(\sum_{i=1}^{3} w_{i1}^2 c_i) \tag{8}$$



Kuva 4: Neuroverkko, jonka syöte on vektori $x = (x_1, x_2) \in \mathbb{R}^2$. Verkossa on yksi piilokerros, jossa on kolme neuronia. Symboli w_{ij}^1 on piilokerroksen neuronin j paino ja d_i on piilokerroksen neuronin j vakiotermi. Ulostulokerroksessa on yksi neuroni. Tästä neuronista saadaan syötteestä x laskettu tulos t.

3.3.1 Perseptroni eli klassinen neuronimalli

Perseptroni on neuronverkon laskennallinen yksikkö, jonka toimintamalli perustuu hermosolun toimintaan. Perseptroni muodostuu syötekerroksesta ja yhdestä neuronista ja se saa syötteet $x = (x_1, x_2, ..., x_n) \in \mathbb{R}^n$, jotka kerrotaan painojen muodostamalla vektorilla $w = (w_1, w_2, ..., w_n) \in R^n$. Kun aktivointifunktio on yksiportainen $h : \mathbb{R} \to 0, 1$

$$h(s) = \begin{cases} 1, \text{jos } s > 0 \\ 0, \text{jos } s \leq 0 \end{cases} \tag{9}$$

perseptronia vastaa funktio $P : \mathbb{R}^n \to [0, 1]$,

$$P(x) = \begin{cases} 1, \text{jos } w \cdot x + b > 0 \\ 0, \text{jos } w \cdot x + b \leq 0 \end{cases} \tag{10}$$

jossa b on vakiotermi.

Tällöin joukot $A \subset \mathbb{R}^n$ ja $B \subset \mathbb{R}^n$ voidaan erottaa lineaarisesti, jos on olemassa vakiot $c_1, c_2, ...c_n \in \mathbb{R}$ ja vakio $b \in \mathbb{R}$, joille

$$\sum_{i=1}^{n} c_i x_i > b, \forall x \in A \tag{11}$$

$$\sum_{i=1}^{n} c_i x_i \leq b, \forall x \in B \tag{12}$$

Tasossa $\mathbb{R}^2$ tämä tarkoittaa, että joukkoja A ja B vastaavat pisteet voidaan erottaa suoralla ja avaruudessa $\mathbb{R}^3$ pisteet voidaan erottaa tason avulla. [16]

Perseptroni ei kykene ratkaisemaan epälineaarisia ongelmia. Tämä voidaan ratkaista yhdistämällä useampi perseptroni verkkorakenteeksi (multilayer perceptron = MLP), jossa edellisen kerroksen tulokset toimivat seuraavan kerroksen syötteenä. (mm. XOR-ongelman ratkaisu). [16, 24]

3.3.2 Aktivointifunktioista φ

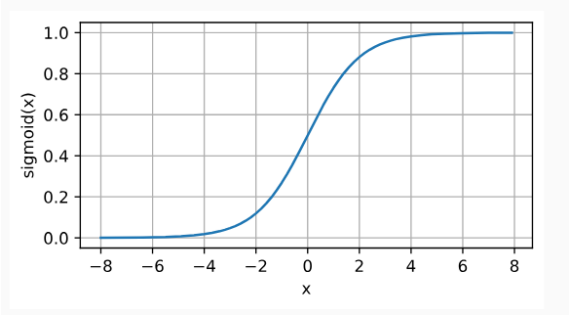
Aktivointifunktioita käytetään syvien neuroverkkojen piilo- ja ulostulokerroksissa. Kun piilo- tai ulostulokerroksen neuroniin saapuneiden syötteiden painotettu summa on laskettu yhteen, se viedään aktivointifunktioon φ: $\mathbb{R} \to \mathbb{R}$, joka muuntaa lineaarisen syötteen epälineaariseksi, mikä puolestaan mahdollistaa epälineaarisen datan moniulotteisten relaatioiden oppimisen paremmin kuin pelkästään lineaaristen syötteiden summana. Jatkuvasti derivoituva aktivointifunktio (esim. sigmoidaalinen aktivointifunktio, 13) mahdollistaa sen käytön myös vastavirta-algoritmissa (backpropagation) ja muissa virhefunktion minimointimenetelmissä. Jos aktivointifunktio on rajoitettu, korjausaskeleet ovat pienempiä ja verkon oppiminen on vakaata mutta oppiminen hidastuu, kun korjausaskeleen pituus lyhenee. Rajoittamattomilla aktivaatiofunktioilla oppiminen on puolestaan tehokkaampaa, koska kerralla voidaan ottaa isompia muutosaskeleita. Suurten korjaukseen pyrkivien askeleiden seurauksena parametrien optimoinnista eli verkon oppimisesta voi kuitenkin tulla isojen heilahdusten seurauksena epävakaata. [16]

Aktivointifunktion valinta verkon arkkitehtuurissa perustuu yleensä siihen, minkälaisen datan käsittelyyn verkko on suunniteltu ja minkätyyppisiä tuloksia ulostulokerroksen odotetaan tuottavan. Yleisiä aktivointifunktioita ovat sigmoidaalinen aktivointifunktio (logistinen funktio), hyperbolinen tangentti (Tanh), rectified linear unit (ReLU) ja Softmax. Tutkielman lopussa esiteltävän AlphaFold3:n arkkitehtuurissa käytetään useimmissa moduuleissa SwiGLU-aktivointifunktiota.[1, 14, 16]

Sigmoidaalinen aktivointifunktio

$$f : \mathbb{R} \to]0, 1[, \quad \sigma(x) = \frac{1}{1 + e^{-x}} \tag{13}$$

on rajoitettu, jatkuva, aidosti kasvava ja sen derivaatta on jatkuva (kuva 5). Tulokset asettuvat välille 0 - 1, joten sitä voidaan käyttää todennäköisyyksien laskemiseen. Funktio ei ole symmetrinen nollan suhteen ja sitä käytetään nykyisin yleensä ulostulokerroksessa. [16]

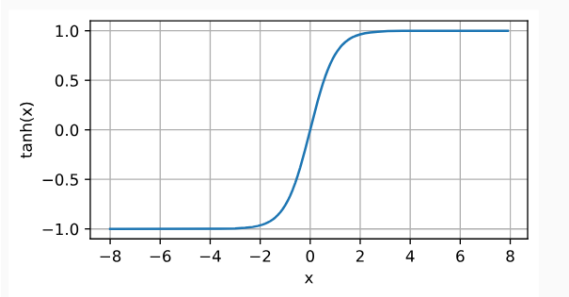


Kuva 5: Sigmoid-funktio $f : \mathbb{R} \rightarrow]0, 1[, \sigma(x) = \frac{1}{1+e^{-x}}$ [12]

Hyperbolinen tangentti (Tanh)

$$g : \mathbb{R} \rightarrow]-1, 1[, \quad \tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{14}$$

on rajoitettu, jatkuva, aidosti kasvava ja sen derivaatta on jatkuva (kuva 6). Toisin kuin sigmoidaalinen aktivointifunktio, hyperbolinen tangentti on symmetrinen nollan suhteen ja se kasvaa nopeammin nollan läheisyydessä, jolloin sen derivaatta on isompi. Pienillä ja isoilla x:n arvoilla gradientti on pieni, mikä hidastaa oppimista näillä arvoilla. [16]



Kuva 6: Tanh-funktio $\mathbb{R} \rightarrow]-1, 1[, \quad \tanh(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$ [12]

Rectified linear unit (ReLu)

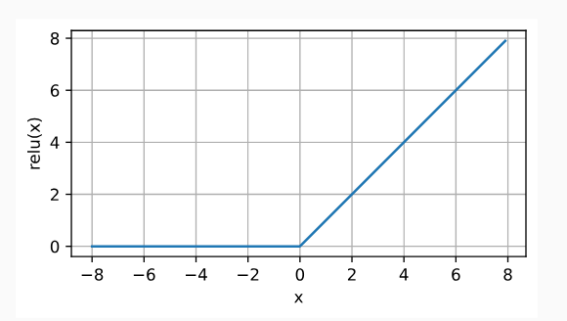
$$ReLU(x) = x^+ = max\{0, x\} = \frac{x + |x|}{2} = \begin{cases} \text{x, kun x} > 0 \\ 0, \text{kun} x < 0 \end{cases} \tag{15}$$

ReLu (kuva 7) on yleisimpiä syvien verkkojen piilokerroksissa käytettyjä aktivaatiofunktiota. Funktion derivaatta on syötteen positiivisilla arvoilla vakio, minkä seurauksena virheen korjaamiseen käytetty gradientti ei pienene suurillakaan syötteen arvoilla toisin kuin katoavan gradientin ongelmassa. Funktio on epäjatkuva nollassa, eikä siis ole jatkuvasti derivoituva. Negatiivisilla syötteen x arvoilla funktion

derivaatta on nolla, mikä johtaa esim. vastavirta-algoritmissa neuronin inaktivoitumiseen. Tämän seurauksena informaatiota häviää, joskin laskentaprosessi nopeutuu. Ongelma voidaan ratkaista muuttamalla funktiota siten, että syötteen x negatiivisilla arvoilla funktioon lisätään syötteelle pieni kerroin α = 0,01 - 0,3, jolloin aktivointifunktiota kutsutaan nimellä Leaky ReLU (16). Muita ReLUsta johdettuja aktivointifunktioita ovat esim. Softmax ja Swish-funktio. [16, 20, 24]

$$f(x) = \begin{cases} \text{x, x} > 0 \\ \alpha x, x \leq 0 \end{cases} \tag{16}$$

$$f'(x) = \begin{cases} 1, \text{x} > 0 \\ \alpha, x \leq 0 \end{cases} \tag{17}$$



Kuva 7: ReLU-aktivointifunktio. Kuvasta nähdään, että funktion derivaatta on negatiivisilla syötteen arvoilla 0 ja positiivisilla arvoilla 1. Leaky ReLU:n derivaatta saa negatiivisilla syötteillä pienen positiivisen arvon yleensä väliltä $0,01 - 0,3$, jolloin signaali ei koskaan nollaudu täysin ja optimointialgoritmilla pystytään jatkamaan painojen optimointia.[12]

Softmax

$$\sigma : \mathbb{R} \to (0,1)^K, K > 1, z \in \mathbb{R}^K \quad \sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{18}$$

Softmax (18) on johdettu ReLu:sta. Neuroneiden arvojen summa on 1, jolloin sitä voidaan käyttää esim. todennäköisyyksien ennustamiseen monen luokan luokittelutehtävissä. Yleensä Softmax:ia käytetäänkin neuroverkon ulostulokerroksessa.[16, 21]

SwiGLU

Tässä kappaleessa on käytetty lähteinä julkaisuja [1, 10, 12, 13]. SwiGLU-aktivointifunktio on muodostettu yhdistämällä Swish- ja GLU (gated linear unit) -aktivointifunktiot. Swish-aktivointifunktio (19) on samankaltainen ReLu:n kanssa: sen arvo on lähellä nollaa, kun x saa negatiivisia arvoja ja se on lähes lineaarinen, kun x saa positiivisia arvoja. Yhtälössä (19) $\sigma(x)$ on määritelty yhtälössä (13) ja β on opittava parametri.

$$i : \mathbb{R} \to [0, \infty), \quad Swish(x) = x * \sigma(\beta x) \tag{19}$$

GLU-aktivointifunktiossa (20) syöte jaetaan kahteen osaan. Toiselle osalle suoritetaan tavallinen lineaarimuunnos ja toinen osa kulkee sigmoidaalisen aktivointifunktion läpi. Jälkimmäinen prosessi ("portti") määrittää, kuinka suuri osa varsinaisesta tiedosta siirtyy verkon seuraavaan kerrokseen.

$$GLU(x, W, V, b, c) = \sigma(xW + b) \otimes (xV + c) \tag{20}$$

Yhtälössä (20) x on syöte eli kerrokseen tuleva datavektori, W ja V ovat painokerroinmatriiseja, joiden kertoimia optimoidaan oppimisprosessin aikana, ja b ja c ovat vakioita ("bias").

Yhdistetyssä SwiGLU:ssa (21) β on (opittava tai vakio) kynnysparametri, joka määrittää miten jyrkästi funktio reagoi syötteeseen. Kun kun $\beta \to \infty$, funktio muistuttaa ReLUa. Isoissa kielimalleissa käytettään oletusarvoa $\beta = 1$.

$$SwiGLU(x, W, V, b, c, \beta) = Swish_\beta(xW + b) \otimes (xV + c) \tag{21}$$

Yhtälössä (21) $xV + c$ on pääpolku, jossa syöte kulkee lineaarisen muunnoksen läpi. $Swish_\beta(xW + b)$ on portti, jossa syöte kulkee erillisen lineaarisen muunnoksen ja aktivaatiofunktion läpi.

SwiGLU:ssa pienet negatiiviset arvot parantavat optimoinnin tulosta ja verkon koulutus on vakaampaa. Esim. ReLU:ssa negatiiviset arvot kuvautuvat nollaksi ja oppiminen pysähtyy kyseisen neuronin osalta. GLU-rakenne toimii porttina, jolla hallitaan informaatiovirtaa. SwiGLU on laskennallisesti työläs, mutta sen ansiosta verkon konvergenssi on nopeaa ja suorituskyky ovat parempi kuin monilla muilla aktivointifunktioilla. SwiGLU-aktivointifunktiota käytetään proteiinien kolmiulotteista rakennetta ennustavassa AlphaFold3-algoritmissa syötteen upotusvaiheessa ja MSA-moduulissa.

3.3.3 Vastavirta-algoritmi (backpropagation)

Vastavirta-algoritmilla pyritään pienentämään verkon tuottaman tuloksen virhettä. Koulutus- ja testausvaiheessa verkon syötteestä tuottamaa tulosta verrataan tavoiteltuun tulokseen ja lasketaan tämän perusteella virhefunktion arvo. Virhefunktio on kaikkien neuronien painojen ja vakiotermien funktio, jonka arvo pyritään minimoimaan käyttäen sopivaa menetelmää (esim. gradienttimenetelmä). [16, 24]

Syötettä x_k vastaavan tavoitellun tuloksen $y_k \in R^m$ ja verkon tuottaman tuloksen $t_k \in R^m$ virhefunktiona E (22) käytetään usein tavoitellun tuloksen ja verkon tuottaman tuloksen erotuksen euklidisen normin neliötä:

$$E = \frac{1}{2}||t - y||^2 = \frac{1}{2}\sum_{k=1}^{m}(t_k - y_k)^2 \tag{22}$$

Verkon tuottama tulos on mahdollisimman lähellä oikeaa tulosta silloin, kun virhefunktion arvo on mahdollisimman pieni. Virhefunktion arvoa voidaan minimoida optimoimalla neuronien painot w (kts. yhtälö (6) ja kuva 4). Monissa minimointimenetelmissä tämä tehdään laskemalla vastavirta-algoritmin avulla virhefunktion E osittaisderivaatat $\frac{\partial E}{\partial w}$ verkon painojen ja $\frac{\partial E}{\partial d}$ verkon vakiotermien suhteen (kts. yhtälöt (6) ja (8), joissa näkyvät muuttujat w ja d). Esim. gradienttimenetelmässä verkko käydään läpi vastakkaisessa suunnassa tuloskerroksesta syötekerrokseen analyysin ketjusäännön mukaan ja jokaiselle neuronille lasketaan sekä uusi paino w että vakiotermi d muuttamalla edeltäviä arvoja virhefunktion nopeimman pienenemisen suuntaan. [16, 19, 24]

3.4 Datasta

Syväoppivan neuroverkon koulutuksessa käytettävän datan laatu vaikuttaa merkitsevästi generoitavan mallin suorituskykyyn. Hyvälaatuinen, oikeaa tietoa sisältävä ja kattava data tuottaa tarkemman ennusteen kuin puutteellinen, virheitä sisältävä ja kapea-alainen data. Mitä enemmän hyvälaatuista dataa on käytettävissä, sitä vahvempi ja tarkempi malli voidaan yleisesti kouluttaa. Mallin suorituskyvyn mittaamiseen tarvitaan aina koulutusdatasta erillinen testiaineisto, [24]

Data esitetään perinteisesti numeerisena ja yleensä taulukkomuodossa, jossa rivi edustaa yksilöä ja sarake edustaa tiettyä ominaisuutta/piirrettä. Nykyisin koneoppimiseen perustuvat sovellukset käsittelevät myös ei-taulukkomuotoista dataa, kuten geenisekvenssejä, tekstiä, kuvia, ääntä ja kaavioita. [3, 24]

Jokainen datapiste (neuroverkon syöte) x_n on vektori avaruudessa $\mathbb{R}^n$. Yleisesti vektorit ovat avaruudellisia objekteja, joita voidaan laskea yhteen ja kertoa skalaareilla. Myös objektia, joka täyttää nämä vaatimukset, voidaan pitää vektorina. Näitä ovat esimerkiksi geometriset vektorit, polynomit, audiosignaalit ja avaruuden $\mathbb{R}^n$:n alkiot. [3, 9]

3.4.1 Koneoppimisen lineaarialgebraa

Tässä kappaleessa on käytetty lähteinä teoksia [5, 9]. Koneoppimisalgoritmeja varten data on yleensä muokattu numeeriseen muotoon, ja ratkaistava ongelma voidaan usein esittää lineaarisena yhtälönä:

$$a_1x_1 + a_2x_2 + ... + a_nx_n = b, \tag{23}$$

missä $x_1, x_2, ..., x_n$ ovat tuntemattomia vektoriavaruudessa $\mathbb{R}^n$, $a_1, a_2, ..., a_n \in \mathbb{R}$ ovat vakiokertoimia ja $b \in \mathbb{R}$ on vakio.

Lineaarinen yhtälöryhmä esitetään muodossa:

$$\begin{aligned} a_{11}x_1 + a_{12} + \cdots + a_{1n}x_n &= b_1 \\ &\vdots \\ a_{m1}x_1 + a_{m2} + \cdots + a_{mn}x_n &= b_m, \end{aligned}$$

missä $a_{ij} \in \mathbb{R}$ ja $b_1 \in \mathbb{R}$. Kun kertoimet a_{ij} kootaan vektoreiksi saadaan:

$$x_1 \begin{bmatrix} a_{11} \\ \vdots \\ a_{m1} \end{bmatrix} + x_2 \begin{bmatrix} a_{12} \\ \vdots \\ a_{m2} \end{bmatrix} + \cdots + x_n \begin{bmatrix} a_{1n} \\ \vdots \\ a_{mn} \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

ja edelleen kokoamalla vektorit matriisiin, saadaan:

$$\begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}.$$

Matriiseja voidaan käyttää sekä datan että transformaatioiden (kuvauksen) esittämiseen. Data voidaan esittää datamatriisina, missä jokainen rivi on yksi dataobjekti (esim. yksilö) ja jokainen sarake vastaa yhtä attribuuttia (esim. yksilön ominaisuus).

Merkitään matriisi

$$A = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix},$$

$$B = \begin{bmatrix} b_{11} & \dots & b_{1n} \\ \vdots & & \vdots \\ b_{m1} & \dots & b_{mn} \end{bmatrix}$$

missä $a_{ij}, b_{ij} \in \mathbb{R}$ on matriisin alkio. Tällöin matriisin $A \in \mathbb{R}^{m \times n}$ ja matriisin $B \in \mathbb{R}^{m \times n}$ summa:

$$A + B = (a_{ij}) + (b_{ij}) = (a_{ij} + b_{ij}) = \begin{bmatrix} a_{11} + b_{11} & \dots & a_{1n} + b_{1n} \\ \vdots & & \vdots \\ a_{m1} + b_{m1} & \dots & a_{mn} + b_{mn} \end{bmatrix}$$

Reaaliluvun r ja matriisin A tulo eli matriisin A skalaarimonikerta:

$$rA = r(a_{ij}) = (ra_{ij}) = \begin{bmatrix} ra_{11} & \dots & ra_{1n} \\ \vdots & & \vdots \\ ra_{m1} & \dots & ra_{mn} \end{bmatrix}$$

Matriisien $A \in \mathbb{R}^{m \times n}$ ja $B \in \mathbb{R}^{n \times k}$ tulo on $C = AB \in \mathbb{R}^{m \times k}$. Matriisin C alkiot määritellään seuraavasti:

$$c_{ij} = \sum_{l=1}^{n} a_{il} b_{lj} = a_{i1} b_{1j} + a_{i2} b_{2j} + ... + a_{in} b_{nj}. \tag{24}$$

3.4.2 Lineaarikuvauksesta

Tämä kappale perustuu Prof. Metsänkylän luentomonisteeseen [9]. Lineaarikuvaus on kuvaus kahden eriulotteisen vektoriavaruuden välillä. Vektoriavaruuksien U ja V välistä kuvausta $t : U \to V$ sanotaan lineaarikuvaukseksi, jos se täyttää ehdot:

$$t(X_1 + X_2) = tX_1 + tX_2, \forall X_1, X_2 \in U \tag{25}$$

ja

$$t(aX) = atX, \forall a \in \mathbb{R}, X \in U \tag{26}$$

Kun U ja V ovat vektoriavaruuksia, $B_1, ..., B_n$ on U:n kanta ja $Y_1, ..., Y_n$ ovat mielivaltaisia V:n vektoreita, on olemassa yksikäsitteinen lineaarikuvaus $t : U \to V$, joka täyttää ehdon

$$tB_i = Y_i, (i = 1, ..., n) \tag{27}$$

Jos t ja s ovat lineaarikuvauksia $U \to V$, niin myös niiden summa ja skalaarimonikerrat ovat lineaarisia:

$$t + s : U \to V, (t + s)X = tX + sX \tag{28}$$

$$ct : U \to V, (ct)X = c * tX, (c \in \mathbb{R}) \tag{29}$$

Samoin, jos t on lineaarikuvaus $U \to V$ ja s on lineaarikuvaus $V \to W$, niis yhdistetty kuvaus $s \circ t : U \to W$ on lineaarinen.

Lineaarikuvausten käsittely voidaan muuttaa matriisilaskennaksi seuraavasti: Jos $t : U \to V$ on lineaarikuvaus, $B = B_1, ..., B_n$ jokin U:n kanta ja $C = C_1, ..., C_m$ V:n kanta. Tällöin vektorien tB_j kantaesitykset ovat:

$$\begin{cases} tB_1 = a_{11} C_1 + ... + a_{m1} C_m \\ ... \\ tB_n = a_{1n} C_1 + ... + a_{mn} C_m \end{cases} \tag{30}$$

Lineaarikuvausta tarvitaan usein koneoppimisalgoritmeissa, kun datamatriisit muutetaan yhteensopiviksi. Kun siirrytään avaruudesta $\mathbb{R}^n$ avaruuteen $\mathbb{R}^m$, kuvaus t voidaan esittää transponoidulla $m \times n$-matriisilla A:

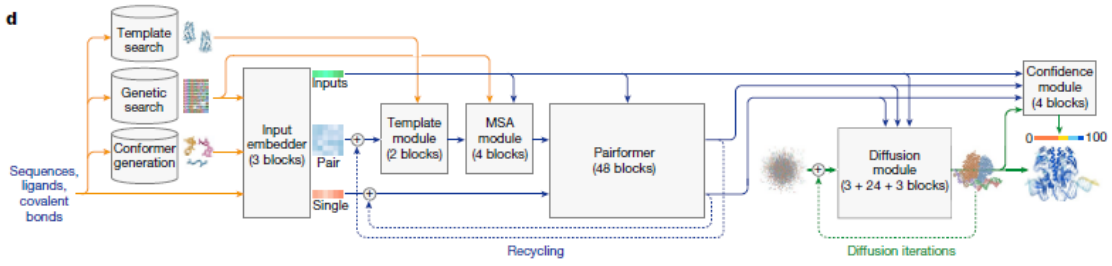
$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \tag{31}$$

jossa lähtöavaruuden ($\mathbb{R}^n$) vektori x sisältää n lukua ja maalijoukon avaruus ($\mathbb{R}^m$) sisältää m kpl lukuja, jolloin lopputulos on $y = Ax$. [9]

Matriisi A ei ole koneoppimisprosessissa vakio, vaan sen sisältämät painokertoimet $a_{11}, ..., a_{mn}$ alustetaan koneoppimisprosessin alussa ja niitä muutetaan oppimisprosessin aikana niin, että algoritmin tuottaman ennusteen virhe pienenee konvergenssin edetessä.

4 AlphaFold3

AlphaFold 3 (AF3) on v. 2024 julkaistu syväoppivaan neuroverkkoarkkitehtuuriin perustuva tekoälymalli, joka ennustaa proteiinien ja niihin liittyvien nukleiinihappojen, ligandien ja ionien muodostaman kompleksin kolmiulotteista rakennetta. AF3:n arkkitehtuuri jaetaan syötteen valmisteluvaiheeseen, esitystapojen oppimisvaiheeseen ja kompleksin rakenteen ennustavaan vaiheeseen (kuva 8). Mallin moduulit ja niiden toiminta esitellään pääpiirteidensä osalta seuraavissa kappaleissa. [1]



Kuva 8: AlphaFold3:n arkkitehtuuri. [1]

Aiemmin esitellyn proteiinien rakenteen sekä syväoppivien neuroverkkojen taustalla olevan matemaattisen teorian lisäksi AF3:n moduulien ja toiminnan ymmärtämisen kannalta olennaisia termejä ovat:

tokeni (token) on AF3:n käyttämä proteiinikompleksin rakenteen perusyksikkö, joka voi olla proteiinien tavanomainen aminohappo tai nukleiinihappoketjun tavanomainen nukleotidi. Mikäli kyseessä on modifioitu (non-standard) aminohappo tai nukleotidi tai ligandi (esim. metalli-ioni tai lääkemolekyyli), käytetään tokenina kyseisen molekyylin atomia, jolloin ko. molekyyli koostuu useista tokeneista. (kts. Kuva 9) [14]

tokenisointi on vaihe, jossa syötteenä annetut molekyylirakenteet muutetaan diskreeteiksi yksiköiksi eli tokeneiksi. AF3:ssa tämä tarkoittaa aminohappojen, nukleotidien ja ligandien muuntamista numeerisiksi tokeneiksi, jolle mallin algoritmit voivat laskea esimerkiksi paikkatietoa, interaktioita ja muita kolmiulotteisen rakenteen muodostamisen kannalta olennaisia arvoja. [14]

tokenin upotus: Ennen upotusta jokaiselle tokenisoinnin tuloksena muodostetulle yksikölle (esim. aminohappo) listataan kolmiulotteisen rakenteen muodostamiseen vaikuttavia arvoja (esim. etäisyys viereisistä tokeneista, varaus, tokenien välisten sidosten kiertokulmia), jotka kuvaavat kyseisen tokenin ominaisuuksia eli attribuutteja. Tuloksena saadaan vektori, jota käytetään algoritmin syötteenä (kuva 4)ja jota muokataan prosessin aikana (usein käytetään ilmaisua: "tensori kulkee verkon läpi"). Upotus sijoittaa samankaltaiset vektorit lähelle toisiaan multilineaarisessa laskenta-avaruudessa.

tensori on objekti, joka yleistää skalaarin, vektorin, matriisin ja bilineaarisen muodon käsitteet. Skalaari on 0., vektori on 1. ja matriisi on 2. kertaluvun tensori. Korkeamman kertaluvun tensorit ovat lineaarikuvauksia. [22]

templaatti eli malli tarkoittaa kokeellisesti määritettyä proteiinikolmpleksin 3D-rakennetta, jota käytetään AF3:n ennusteen pohjana tai vertailukohtana.[14]

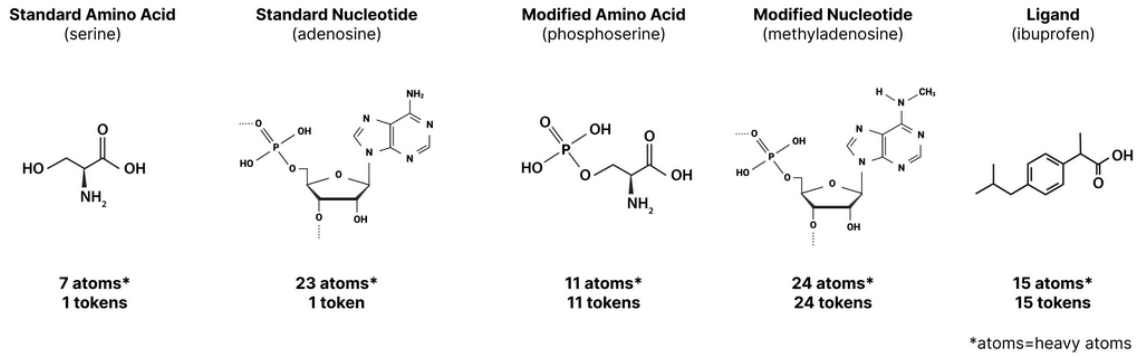
yksikköesitys (single representation): tarkoittaa tietorakennetta (vektoria tai matriisia tai tensoria), joka kuvaa yksittäisen aminohapon tai tokenin ominaisuuksia. (Gemini)

pariesitys (pair representation) kuvaa matriisia/tensoria, jossa jokainen alkio (i, j) vastaa kahden eri aminohapon tai tokenin välistä suhdetta (esim. etäisyys, sidovoimat, kulmat). Pariesitystä päivitetään AF3:n eri kerroksissa. (Gemini)

huomiomekanismi (attetion) koneoppimisessa huomiomekanismi on menetelmä, joka laskee jokaiselle sarjassa olevalle tekijälle (tokeni) tärkeyden/painoarvon suhteessa muihin sarjassa oleviin tekijöihin. Tällöin malli pystyy esim. yhdistämään asioita, jotka ovat sarjassa kaukana toisistaan, mutta liittyvät olennaisesti yhteen (algoritmin muisti). [23]

4.1 Syötteen valmistelu (Input preparation)

Syötteen valmisteluvaihe koostuu vastinehaun ja geneettisen haun moduuleista, sekä konformaation luomisen ja syötteen upotuksen sisältävistä vaiheista. Valmisteluvaiheessa valitaan tunnetuista tietokannoista mallinnuksen kohteena olevan proteiinin kannalta samankaltaisten proteiinien aminohapposekvenssejä, jotka muutetaan tokenoimalla numeerisiksi tensoreiksi. Tämän lisäksi malliin tuodaan syötteenä kokoelma molekyylejä, joiden kolmiulotteisen rakenteen oletetaan olevan samankaltainen valittujen molekyylien rakenteen kanssa. Valmisteluvaihe esittää nämä molekyylit erillisinä tensoreina.



Kuva 9: Tokeni tarkoittaa joko tavanomaista aminohappoa tai nukleotidiä tai epätavanomaisen aminohapon tai nukleotidin tai ligandin yksittäistä atomia. Kuva esittää erityyppisten molekyylien tokenit ja niiden lukumäärän. [14]

4.1.1 Vastinehaku (Template search)

Syötteen valmisteluvaiheessa vastinehaun avulla etsitään Protein Data Bank - tietokannasta sellaisia proteiinirakenteita, jotka ovat aminohappojärjestykseltään samankaltaisia tutkimuksen kohteena olevan proteiinin aminohappojärjestyksen kanssa. The Protein Data Bank (https://www.wwpdb.org/) on tietokanta, johon on tallennettu perinteisillä menetelmillä (esim. röntgenkristallografia, NMR-tekniikka ja kryoelektronimikroskopia) avulla määritettyjä proteiinien kolmiulotteisia rakenteita. [14]

4.1.2 Geneettinen haku (Genetic search)

Geneettisen haun moduulissa algoritmi etsii tutkittavan proteiinin aminohapposekvenssien kanssa samantyyppisiä sekvenssejä muiden eliölajien proteiineista. Monisekvenssirinnastuksessa (multiple sequence alignment, MSA) algoritmi linjaa löydetyt samankaltaiset sekvenssit rinnakkain, jolloin voidaan havaita, miten eri kohdat aminohappoketjussa ovat joko säilyneet tai muuttuneet evoluution aikana. Proteiini voidaan esittää matriisina, jossa jokainen rivi edustaa vastaavaa proteiinia jostain toisesta eliölajista. Evoluutiossa muuttumattomina pysyneet aminohapposekvenssialueet (matriisissa samaa aminohappoa sisältävät sarakkeet) ovat yleensä kriittisiä proteiinin rakenteen ja toiminnan kannalta. Mikäli kaksi aminohapposekvenssiä ovat proteiinin kolmiulotteisessa rakenteessa vuorovaikutuksessa keskenään, muutokset niiden sisältämissä aminohapoissa tapahtuvat usein toisiaan vastaavasti pareittain, jolloin 3D-rakenne säilyy samankaltaisena. [14]

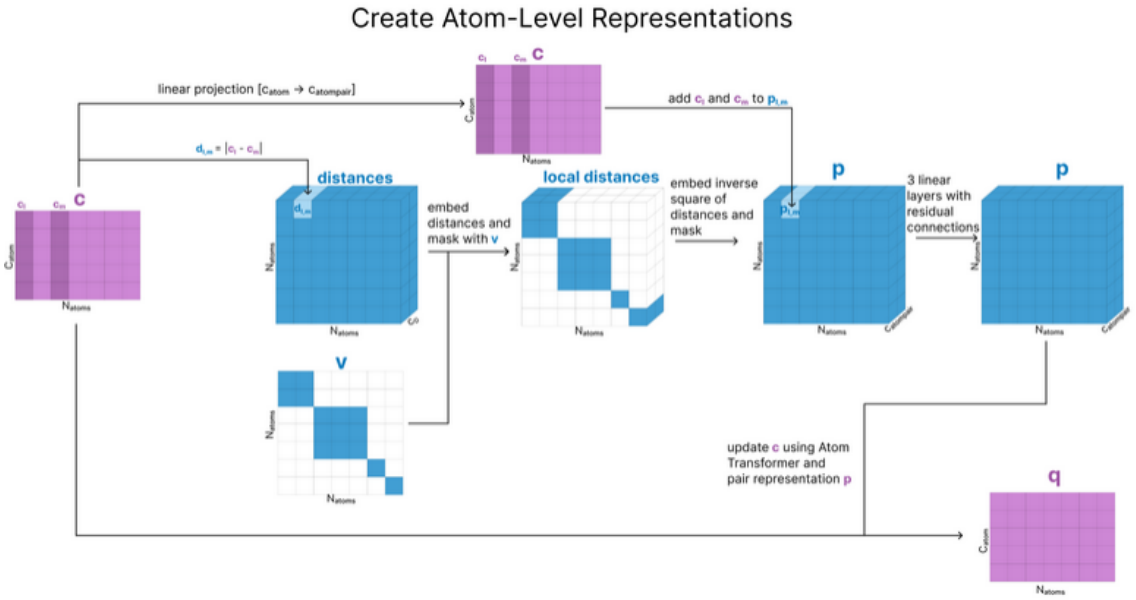
4.1.3 Konformaation luominen (Conformer generation)

Kolmiulotteisen rakenteen luovassa moduulissa lasketaan molekyylille vertailukonformaatio, mikä tarkoittaa molekyylin kolmiulotteista muotoa, joka luodaan muuttamalla atomien välisten sidosten kiertokulmia. (kuva 10). [7, 14]

Aluksi lasketaan vertailukonformaatio jokaiselle mallinnettavan proteiinin aminohapolle, nukleotidille ja ligandille. Jokaisella aminohapolla on tyypillisesti va-

kiokonformaatio, joka on jokin niistä matalaenergisistä muodoista, joissa aminohappo voi esiintyä (kts. kpl 2.1). Konformaatioita lasketaan RDKit's ETKDGv3 -algoritmilla, joka yhdistää sekä kokeellisen tiedon, että energiatasoiltaan matalimmat atomien kiertokulmat. Tuloksena saadaan ennakko-oletus mallin kolmiulotteiselle rakenteelle.

Seuraavassa vaiheessa edellä laskettu data yhdistetään atomin ominaisuuksiin (esim. varaukseen ja järjestysnumeroon). Tästä saadaan tuloksena kuvan 10 matriisi **c**. Matriisia **c** käytetään pohjana 3-tensorille **p**, jossa esitetään pareittain atomien väliset suhteelliset etäisyydet (pariesitys). Lopputuloksena valmisteluvaiheesta saadaan atomitason vektoriesitys **q**, jota päivitetään prosessin seuraavissa vaiheissa. [14]



Kuva 10: Konformaation luovan moduulin toiminta syötteen valmisteluvaiheessa [14]

4.1.4 Syötteen upotus (Input embedder)

Kaikki atomit esittävä matriisi **q** ja kaikki atomit parittain esittävä 3-tensori **p** päivitetään syötteen upotusvaihdeessa läheisten atomien ja niiden ominaisuuksien perusteella. Tämä laskentaprosessi tapahtuu atomitransformerissa, jossa transformeriarkkitehtuuri päivittää matriisia **q** huomiomekanismilla tensorin **p** ja alkuperäisen matriisin **c** avulla. Atomitransformeri noudattaa pääosin tavaomaista transformerrakennetta ja se mallintaa atomien välisiä vuorovaikutuksia. Tässä vaiheessa kerrosnormalisointi (layer norm) skaalaa uudelleen matriisin **q** tokenien parametreihin liittyviä keskiarvoja ja keskihajontaa, ja huomioverkko päivittää tokenien välisiä suhteita käyttäen Softmax-aktivointifunktiota. Lopuksi monikerrosperseptronisiirtymä (MLP-transition) tarkentaa edellä laskettuja piirteitä SwiGLU-aktivointifunktiota käyttäen. [14]

Tuloksena saadaan seuraavat matriisit: monisekvenssirinnastusmatriisi (MSA), rakennemallimatriisi, alkuperäinen atomitason yksikköesitysmatriisi, päivitetty atomitason yksikköesitysmatriisi, tokenitason yksikköesitysmatriisi, atomitason pariesitysmatriisi ja tokenitason pariesitysmatriisi (11). Nämä tensorit toimivat syötteenä seuraavalle moduulille (Piirteiden oppiminen).

Kuva 11: Syötteet upotettuna tensoreiksi, joita käytetään arkkitehtuurin representation learning - ja conformation generation -vaiheissa. [14]

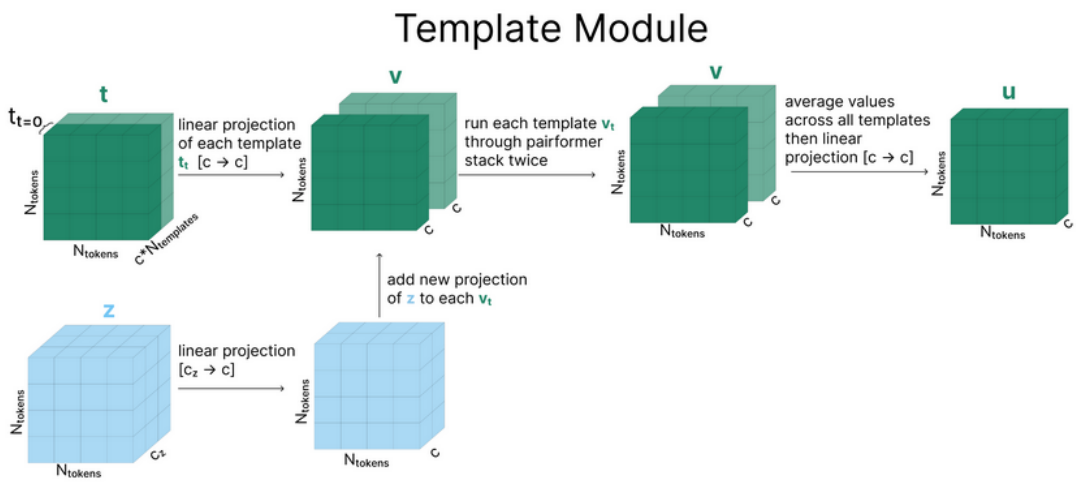
[14]

4.2 Piirteiden oppiminen (Representation learning)

Piirteiden oppimisvaihe koostuu mallimoduulista (template module), MSA-moduulista ja pairformer-yksiköstä, joissa tarkennetaan valmisteluvaiheessa alustetun atomi- ja tokenitason tensoreita (yksikkö- ja parimatriisit). Suurin osa arkkitehtuurin laskennallisesta toiminnasta sijoittuu tähän vaiheeseen. [14]

4.2.1 Templaattimoduuli

Tässä yksikössä jokainen kokeellinen malli muunnetaan lineaarikuvauksella yhteensopivaksi atomien tai tokenien välisiä suhteita kuvaavan matriisin (pariesitys) kanssa ja lasketaan sen kanssa yhteen (kuva 12). Tällä tavoin muodostettu matriisi voidaan siirtää Pairformer-moduulipinoon, joka puolestaan mallintaa molekyylissä olevien atomi- tai tokeniparien välisiä etäisyyksiä ja sidoskulmia. Muusta arkkitehtuurista poiketen viimeisessä lineaarisessa kerroksessa käytetään aktivaatiofunktiona ReLu:a, jolloin datan negatiivinen kohina saadaan poistettua ja data saadaan yhteensopivaksi pariesityksen kanssa. [14]



Kuva 12: Templaattimoduuli, jossa matriisit muutetaan lineaarikuvauksella yhteensopiviksi ja lasketaan yhteen. [14]

4.2.2 MSA-moduuli

MSA-moduulissa parannetaan samanaikaisesti MSA-matriisia ja pariesitysmatriiseja. Pariesityksessä jokainen $\overline{z}_{ij}$ on vektori, joka sisältää informaatiota tokenien i ja j välisistä suhteista. Kun 3-tensori z projisoidaan matriisiksi, jokainen $\overline{z}_{ij}$ vektori muuttuu skalaariksi, joka kuvaa, kuinka paljon tokeni i vaikuttaa tokeniin j. Kun matriisiin sovelletaan riveittäin Softmax-aktivointifunktiota, saadaan huomiomekanismilla pisteet (attention scores), joita käytetään painotetun keskiarvon muodostamiseen huomiokartassa tai painotusmatriisissa. Eli tässä vaiheessa lasketaan, miten malli kohdistaa painokertoimensa eri tokenien välisiin suhteisiin eli kuinka suurella painolla kukin sekvenssin osa vaikuttaa muihin osiin laskennan aikana. Lopuksi pariesitys päivitetään triangle-päivitysten ja huomiomekanismin avulla. Osassa transitioblokkeja käytetään SwiGLU-aktivointifunktiota matriisin projisointiin. [14]

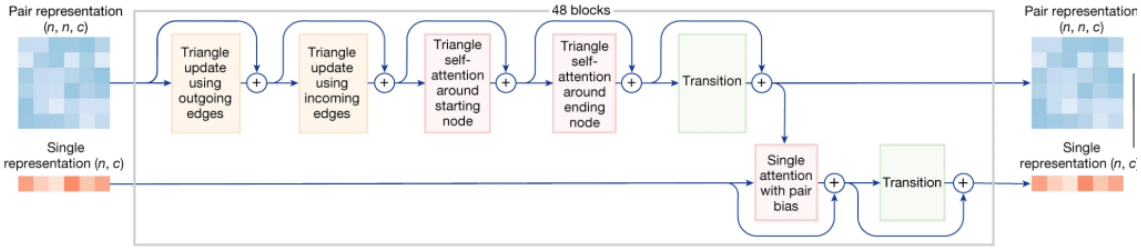
4.2.3 Pairformer-moduuli

Pairformer-moduulissa (kuva 13) tokenitason pariesitys ja yksikköesitys päivitetään geometriaan perustuvan huomiomekanismin avulla. Moduulin toiminta perustuu kolmioepäyhtälöön:

$$||u + v|| \leq ||u|| + ||v|| \tag{32}$$

jossa vektorien u ja v summan normi on pienempi tai yhtäsuuri kuin samojen vektorien normien summa. Pariesityksessä vektori $\overline{z}_{ij}$ edustaa useita erilaisia aminohappojen i ja j välisiä vuorovaikutuksia ja parametrejä (esim. etäisyys toisistaan, sidosvuorovaikutukset, evolutiivinen kytkentä). Yksiulotteisesti voidaan ajatella, että $\overline{z}_{ij}$ kuvaa aminohappojen i ja j välistä etäisyyttä, ja $\overline{z}_{jk}$ kuvaa puolestaan ami-

nohappojen j ja k välistä etäisyyttä. Kolmioepäyhtälön perusteella aminohappojen i ja k välinen etäisyys on pienempi tai yhtäsuuri kuin etäisyyksien $\overline{z}_{ij}$ ja $\overline{z}_{jk}$ summa. Todellisuudessa tensori z edustaa monimutkaisia fysikaalisia suhteita tokenien välillä, mistä seuraa, että kolmioepäyhtälöön perustuvat rajoitukset ovat suuntaa-antavia. Geometriset rajoitukset tuodaan malliin kolmiopäivitysten ja huomiomekanismin avulla. Pairformerin laskentaprosessi toistetaan 48 kertaa ja edellisellä kerralla saatua tulosta käytetään syötteenä seuraavalle optimointikierrokselle. Tuloksena saadaan tensorit s^{trunk} ja z^{trunk}. Tensori s^{trunk} kuvaa yksittäisen tokenin i ominaisuuksia sekä ohjaa tokenin sijaintia 3D-avaruudessa. z^{trunk} puolestaan sisältää tiedot tokenien i ja j välisistä suhteista (esim. etäisyys toisistaan, sidosvuorovaikutukset, evolutiivinen kytkentä). [14]

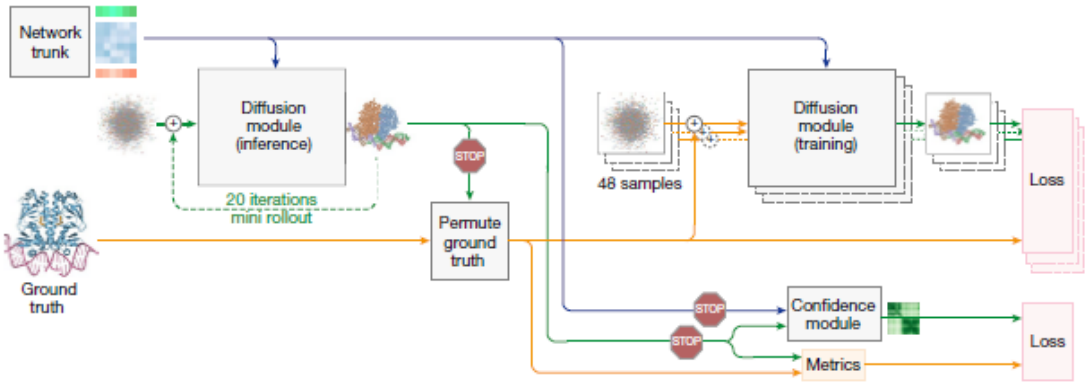


Kuva 13: Pairformer-moduuli päivittää pariesityksen (dimensiot n,n,c) ja yksikköesityksen (dimensiot n,c), jossa $n =$ tokenien lkm, $c =$ kanavien lkm (128 kpl pariesitykselle, 384 yksikköesitykselle). Jokaisella 48 lohkolla on erillinen koulutettavien parametrien kokoelma. Tuloksena saadaan tokenin tiedot sisältävä s^{trunk}-tensori ja tokenien välisestä vuorovaikutuksesta kertova z^{trunk} -tensori, jossa jokainen solu i, j sisältää vektorin, joka puolestaan sisältää tiedon tokenien i ja j välisistä vuorovaikutuksista. [1, 14]

4.3 Proteiinin rakenteen ennustaminen

Molekyylikompleksin rakenne-ennuste lasketaan diffuusiomoduulissa (kuva 14). Moduuliin tuodaan ensin todellinen data aiemmista moduuleista ja siihen liitetään hallitusti kohinaa, kunnes alkuperäisen mallin rakenne on hävinnyt. Tämän jälkeen algoritmi oppii poistamaan kohinan asteittain ja ennustamaan atomien koordinaatit $x, y, z \in \mathbb{R}^3$ neljässä eri vaiheessa. Tähän malli hyödyntää pairformer-moduulista saatua dataa tokenien ja atomien välisistä etäisyyksistä, sidosvoimista ja -kulmista. Lopputuloksena saadaan atomien etäisyyksiä kuvastava pistepilvi. Diffuusiomekanismilla tuotetaan useita mahdollisia rakenteita, joilla pyritään kuvaamaan molekyylin luonnollista dynaamisuutta ja tuloksen epävarmuutta. [1, 14]

Lopuksi Confidence-moduulilla lasketaan ennustemallin luotettavuus ja tarkkuus sekä parametrit, jotka kuvaavat mitkä osat molekyylikompleksista ovat luotettavia ja mitkä ovat epävarmoja. [1, 14]



Kuva 14: Diffuusiomoduuli. [1]

5 Diskussio

Koneoppiminen ja syväoppivat neuroverkot ovat mahdollistaneet proteiinien laskostusongelman tehokkaan ratkaisemisen 2020-luvulla. Proteiinien kolmiulotteisen rakenteen selvittäminen mahdollistaa niiden muokkaamisen tavoitehakuisesti esim. lääkekehityksessä. AlphaFold3:n tuottaman rakenne-ennusteen tarkkuutta mitataan RMSD-arvolla (root mean square deviation), joka kertoo, kuinka paljon ennustettu malli eroaa kokeellisesti määritetystä rakenteesta. Useimmissa molekyylirakenteissa AF3:n algoritmi saavuttaa atomien sijainnissa 1,5 Å:n tarkkuuden, mikä vastaa hiiliatomien välisen sidoksen pituutta. AF3:ssa esiintyy kuitenkin stereokemiaan liittyviä ongelmia, joita ei ole pystytty välttämään algoritmia korjaamalla. Malli ei esimerkiksi pysty aina huomioimaan molekyylissä esiintyvää kiraalisuutta, se saattaa myös sijoittaa kookkailla proteiini-nukleiinihappo -komplekseilla samaan paikkaan kaksi erilaista nukleiinihappo- tai aminohappoketjua ja se on voinut esittää ligandeja sitovat molekyylit suljetussa muodossa, vaikka ligandi ei olisikaan kiinnittynyt sitä sitovaan kohtaan. Lisäksi on huomioitava, että malli ennustaa proteiinin staattisen rakenteen eikä se pysty ennustamaan molekyylin dynaamista käyttäytymistä liuosympäristössä. [1] AF3:n ennusteen tarkkuutta on pyritty parantamaan algoritmin korjaamisen lisäksi tuottamalla useita ennusteita, jotka on asetettu todennäköisyyden perusteella paremmuusjärjestykseen. Tämä on vaatinut kuitenkin lisää aikaa ja laskentavoimaa, ja nostanut siten kustannuksia. Erityisesti vasta-aine-antigeeni -kompleksien rakenteet ovat olleet haasteellisia (esim. kuva 3, jossa on esitetty hiiren IgG2-molekyyli eli vasta-aine ilman antigeenia). Kun huomioidaan ennusteita tuottavien tekoälyalgoritmien tulosten nopea tarkentuminen ja datan määrän lisääntyminen, voidaan olettaa, että AF3:n ennusteissa esiintyvät epätarkkuudet saadaan korjattua lähivuosina, mikäli algoritmien kehitys jatkuu aiemman kaltaisena.

Viitteet

[1] Abramson, J., Adler, J., Dunger, J., Evans, R., Green, T., Pritzel, A., Ronneberger, O., Willmore, L., Ballard, A. J., Bambrick, J., Bodenstein, S. W., Evans,

D. A., Hung, C.-C., O'Neill, M., Reiman, D., Tunyasuvunakool, K., Žídek, A., Žídeková, A., Cascella, L. N., ... & Hassabis, D. (2024). Accurate structure prediction on biomolecular interactions with AlphaFold 3. *Nature*, *630*(8015), 54–62. https://doi.org/10.1038/s41586-024-07487-w

[2] Dao, T. & Rahman, R. (2025). Deep generative modeling of protein conformations: A comprehensive review. *BioChem*, *5*(3), 32. https://doi.org/10.3390/biochem5030032

[3] Deisenroth, M. P., Faisal, A. A. & Ong, C. S. (2020). *Mathematics for machine learning*. Cambridge University Press. https://doi.org/10.1017/9781108679930

[4] Ege, S. (1994). *Organic chemistry: Structure and reactivity* (3. painos). D.C. Heath and Company. ISBN 0-669-34161-4.

[5] Hautala, A. (2020). *Koneoppiminen ja matematiikka, johdanto aiheeseen ja kertausmateriaali* [Pro gradu -tutkielma, Helsingin yliopisto]. Helda.

[6] Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Demir, K., Senior, A. W., Skinner, A. T., Ide, L., Bolaños, I., Cowie, Y., Reiman, D., Ostrovsky, M., Zhong, R., Pandiatan, M., Antonoglou, I., King, A., Tunyasuvunakool, K., ... & Hassabis, D. (2021). Highly accurate protein structure prediction with AlphaFold. *Nature*, *596*(7873), 583–589. https://doi.org/10.1038/s41586-021-03819-2

[7] Landrum, G. (2026). *RDKit: Open-source cheminformatics*. http://www.rdkit.org

[8] Leach, A. R. (2001). *Molecular modeling: Principles and applications* (2. painos). Prentice Hall. ISBN 0-582-38210-6.

[9] Metsänkylä, T. (1997). *Lineaarialgebra*. Turun yliopisto.

[10] Ramachandran, P., Zoph, B. & Le, Q. V. (2017). *Searching for activation functions* [Manuscript submitted for publication]. arXiv. https://doi.org/10.48550/arXiv.1710.05941

[11] Ritter, P. (1996). *Biochemistry: A foundation*. Brooks/Cole Publishing Company. ISBN 0-534-33865-8.

[12] Roldán, C. (18. huhtikuuta 2024). *What is SwiGLU?* Jcarlosroldan. Haettu 18.4.2026 osoitteesta https://jcarlosroldan.com/post/348

[13] Shazeer, N. (2020). *GLU variants improve transformer* [Manuscript submitted for publication]. arXiv. https://doi.org/10.48550/arXiv.2002.05202

[14] Simon, E. & Silberg, J. (maaliskuu 2026). *The illustrated AlphaFold*. Elanapearl. Haettu 1.–31.3.2026 osoitteesta https://elanapearl.github.io/blog/2024/the-illustrated-alphafold/

[15] Stryer, L. (2000). *Biochemistry* (4. painos). W.H. Freeman and Company. ISBN 0-7167-2009-4.

[16] Tuominen, H. & Neittaanmäki, P. (2019). *Tekoälyn perusteita ja sovelluksia*. Jyväskylän yliopisto. ISBN 978-951-39-7796-2.

[17] Wikipedia-muokkaajat. (15. marraskuuta 2025). Entalpia. *Wikipedia*. Haettu 15.11.2025 osoitteesta `https://fi.wikipedia.org/wiki/Entalpia`

[18] Wikipedia-muokkaajat. (1. maaliskuuta 2026). Kryoelektronimikroskopia. *Wikipedia*. Haettu 1.3.2026 osoitteesta `https://fi.wikipedia.org/wiki/Kryoelektronimikroskopia`

[19] Wikipedia contributors. (8 March 2026). Backpropagation. *Wikipedia*. Retrieved March 8, 2026, from `https://en.wikipedia.org/wiki/Backpropagation`

[20] Wikipedia contributors. (8 March 2026). Rectified linear unit. *Wikipedia*. Retrieved March 8, 2026, from `https://en.wikipedia.org/wiki/Rectifier_(neural_networks)`

[21] Wikipedia contributors. (8 March 2026). Softmax function. *Wikipedia*. Retrieved March 8, 2026, from `https://en.wikipedia.org/wiki/Softmax_function`

[22] Wikipedia contributors. (17 March 2026). Tensor. *Wikipedia*. Retrieved March 17, 2026, from `https://en.wikipedia.org/wiki/Tensor`

[23] Wikipedia contributors. (21 March 2026). Attention (machine learning). *Wikipedia*. Retrieved March 21, 2026, from `https://en.wikipedia.org/wiki/Attention_(machine_learning)`

[24] Zhang, A., Lipton, Z. C., Li, M. & Smola, A. J. (2023). *Dive into deep learning*. `https://d2l.ai/index.html`