SVEUČILIŠTE U RIJECI

**TEHNIČKI FAKULTET**

Diplomski sveučilišni studij računarstva

Diplomski rad

**ANALIZA SLIČNOSTI PEPTIDA TEMELJENA NA LATENTNOM PROSTORU AUTOENKODERA**

Rijeka, rujan 2023. Bruno Novosel   
 0069078199

SVEUČILIŠTE U RIJECI

**TEHNIČKI FAKULTET**

Diplomski sveučilišni studij računarstva

Diplomski rad

**ANALIZA SLIČNOSTI PEPTIDA TEMELJENA NA LATENTNOM PROSTORU AUTOENKODERA**

Mentor: doc. dr. sc. Goran Mauša

Rijeka, rujan 2023. Bruno Novosel   
 0069078199

IZJAVA

Izjavljujem da sam ovaj rad izradio samostalno, uz vodstvo i stručnu pomoć mentora doc. dr. sc. Gorana Mauše i asistenta Marka Njirjaka.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Bruno Novosel

ZAHVALA

Zahvaljujem mentoru doc. dr. sc. Goranu Mauši i asistentu Marku Njirjaku na pomoći i stručnom vođenju kojim su me vodili kroz izradu ovog rada i što su mi uvijek bili na raspolaganju.

Sadržaj

[1. UVOD 1](#_Toc143594426)

[2. PEPTIDI 2](#_Toc143594427)

[2.1. Što su peptidi? 2](#_Toc143594428)

[2.2. Antimikrobna svojstva 4](#_Toc143594429)

[2.3. Antiviralna svojstva 6](#_Toc143594430)

[3. AUTOENKODERI 7](#_Toc143594431)

[3.1. Autoenkoderi općenito 7](#_Toc143594432)

[3.2. Varijacijski autoenkoderi 10](#_Toc143594433)

[4. MODEL 15](#_Toc143594434)

[4.1. Alati 15](#_Toc143594435)

[4.1.1. Python 15](#_Toc143594436)

[4.1.2. Jupyter Notebook 15](#_Toc143594437)

[4.1.3. Tensorflow 17](#_Toc143594438)

[4.1.4. Keras 17](#_Toc143594439)

[4.1.5. Ostale Python knjižnice 18](#_Toc143594440)

[4.2. Podaci 19](#_Toc143594441)

[4.3. Metode 22](#_Toc143594442)

[4.3.1. Enkoder 23](#_Toc143594443)

[4.3.2. Dekoder 26](#_Toc143594444)

[4.3.3. VAE 28](#_Toc143594445)

[4.3.4. Obrada podataka 32](#_Toc143594446)

[4.3.5. Gridsearch 34](#_Toc143594447)

[4.3.6. Trening 34](#_Toc143594448)

[5. REZULTATI 35](#_Toc143594449)

[6. ZAKLJUČAK 36](#_Toc143594450)

[LITERATURA 37](#_Toc143594451)

[POPIS KRATICA 40](#_Toc143594452)

[Sažetak 41](#_Toc143594453)

[Abstract 41](#_Toc143594454)

# UVOD

Zahvaljujući napretku medicine i znanosti, otkrivena su i proizvedena mnoga cjepiva i lijekovi koji su značajno doprinijeli suzbijanju raznih bolesti kod ljudi. Unatoč tome, mnoge bolesti i dalje predstavljaju veliku prijetnju ljudskom zdravlju te se neprestano traže nova rješenja za njihovo suzbijanje. U tom kontekstu, peptidi su se istaknuli kao obećavajuće novo rješenje pri liječenju. Iako se još uvijek smatraju pomalo nekonvencionalnim pristupom liječenju raznih bolesti i infekcija i relativno su brojčano ograničeni, peptidi su pokazali obećavajuće rezultate u mnogim istraživanjima [1]. U prirodi postoje peptidi s različitim funkcijama, ali ograničena raznolikost sekvenci unutar ovih molekula ograničava i spektar njihove primjene pa se zbog toga pokušava sintetizirati nove peptide koji će imati željena svojstva. Međutim, predviđanje svojstava novo sintetiziranih peptida i poboljšanje postojećih metoda predviđanja još uvijek ima prostora za napredak.

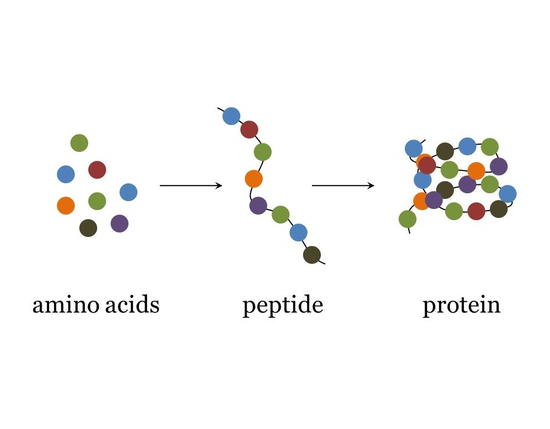
Temeljem dosadašnjih uspjeha metoda strojnog učenja na ovom području, u ovom radu koristi se varijacijski autoenkoder kako bi veliki broj peptida opisanih njihovim fizikalno-kemijskim svojstvima prikazali u latentom prostoru autoenkodera. S obizrom na to da su se brojna fizikalno-kemijska svojstva peptida pokazala kao potencijalni indikatori antimikrobne ili antiviralne učinkovitosti [2], u latentnom prostoru tada možemo promatrati sličnost među pojedinim vrstama i podvrstama peptida sa željenim značajkama. Također, može se postaviti pitanje postoji li korelacija između takve sličnosti, dobivene u latentnom prostoru autoenkodera, i sličnosti izračunate prema sekvencama pojedinih peptida.

Ovaj rad predstavlja metode i rezultate istraživanja koji prikazuju potencijal korištenja strojnog učenja za istraživanje peptida u svrhu otkrivanja novih lijekova. Korištenjem ovakvih računalnih metoda, cilj je ubrzati otkrivanje i dizajn antimikrobnih i antiviralnih peptida s poboljšanom učinkovitošću.

1. PEPTIDI

## Što su peptidi?

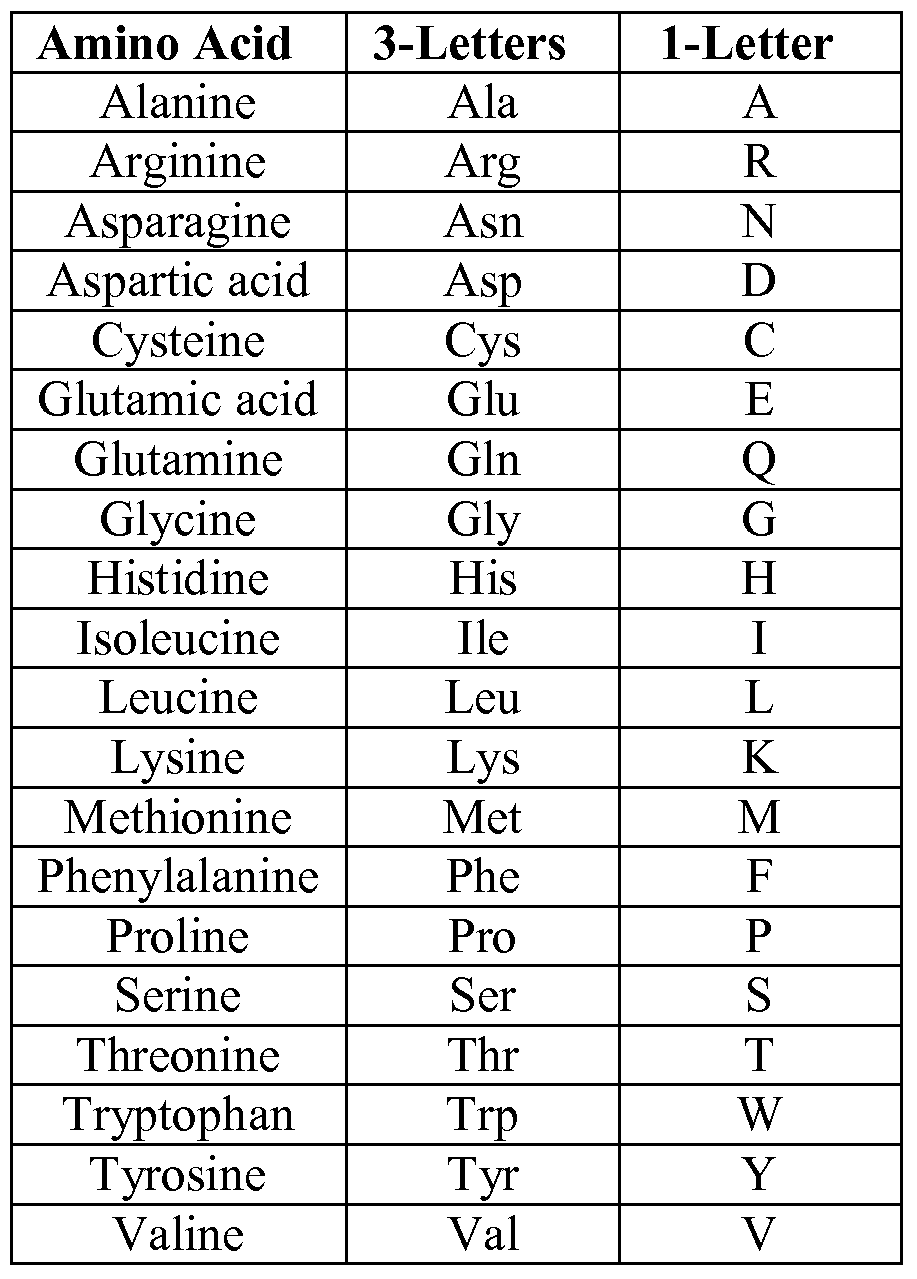
Peptidi su biomolekule prisutne u svim živim organizmima. Kao i proteini, sastavljeni su od aminokiselina povezanih peptidnim vezama u polipeptidne lance, a razlikuju se po broju aminokiselina od kojih se sastoje. Polipeptidne lance do 50 aminokiselina smatramo peptidima, a s više od 50 smatramo proteinima [3]. Pojednostavljeni prikaz aminokiselina, peptida i proteina nalazi se na slici 2.1.

  
*Slika 2.1. Pojednostavljeni prikaz aminokiselina, peptida i proteina (preuzeto iz [4])*

U prirodi se pojavljuje dvadeset standardnih aminokiselina. Svaka aminokiselina ima jedinstvenu strukturu i svojstva te one među sobom mogu stvarati peptidne veze i tako se povezivati u lance. Popis svih prirodnih aminokiselina prikazan je slikom 2.2.

Ovisno o broju aminokiselina od kojih se sastoje, peptidi se mogu klasificirati kao dipeptidi, tripeptidi, oligopeptidi ili polipeptidi. Također se mogu podijeliti na linearne i cikličke, ovisno o načinu povezivanja aminokiselina.

Peptidi imaju raznolike uloge u biološkim sustavima, kao što su signalizacija, obrana od patogena, regulacija enzimskih aktivnosti i strukturna podrška. Jedna od posebno bitnih karakteristika za ovaj rad je da peptidi mogu imati antimikrobnu aktivnost, boreći se na taj način protiv bakterija, gljivica, parazita i virusa. Takvi peptidi, poznati kao antimikrobni peptidi (AMP), imaju široki spektar djelovanja i važni su u razvoju novih lijekova.



*Slika 2.2. Popis svih prirodnih aminokiselina i njihove oznake (preuzeto iz [5])*

Interes istraživača za razvoj novih peptida i proučavanje struktura i funkcija ciljnih receptora znatno je porastao posljednjih godina. Suvremeno medicinsko i biokemijsko istraživanje ima značajan fokus na peptide zbog njihove selektivnosti, specifičnosti i snažne interakcije s ciljanim proteinima. Svojom veličinom i površinom omogućuju preciznije vezanje na ciljne molekule. Mogu imati vrlo selektivno djelovanje, smanjujući rizik od nuspojava, ali brzo se metaboliziraju te imaju kratko djelovanje u tijelu. Aktivnost peptida može se produljiti uvođenjem raznih modifikacija.

Peptidi također imaju neke prednosti u usporedbi s proteinima kada govorimo o liječenju, iako proteini zauzimaju sve veći udio na farmaceutskom tržištu posljednjih nekoliko godina. Proteini, iako su često vrlo sigurni i učinkoviti, moraju se proizvoditi u bioreaktorima koji koriste cijele stanice, a njihovo pročišćenje i strukturna analiza često su složeni i skupi. S druge strane, peptidi se često mogu dobiti kemijski, a njihovo pročišćenje i analiza su mnogo jednostavniji. Također, sve je više primjera oralno djelotvornih peptida, što ih čini poželjnijima jer se lijekovi zasnovani na proteinima gotovo uvijek moraju ubrizgati u tijelo [3].

## Antimikrobna svojstva

Antimikrobni peptidi (AMP) su peptidi koji imaju antimikrobna svojstva i time igraju ključnu ulogu u obrani organizama od mikrobnih infekcija. Često se nazivaju i kationskim peptidima jer su kao cjelina pozitivno nabijeni, iako su zapravo samo podvrsta kationskih peptida. Simboličan prikaz podjele kationskih peptida u skupine prema njihovim funkcijama, među kojima se nalaze i AMP prikazan je slikom 2.4. Oni su dio prirodnog imunološkog sustava prisutnog kod gotovo svih živih organizama, uključujući ljude, životinje, biljke i mikroorganizme. AMP se ističu svojom sposobnošću da selektivno ciljaju i ubiju mikrobe, uključujući bakterije, gljivice, parazite i viruse.

A circular diagram of different types of bacteria

Description automatically generated

*Slika 2.3. Podjela kationskih peptida prema funkcijama (preuzeto iz [6])*

AMP imaju raznoliku strukturu koja može uključivati linearno raspoređene aminokiseline, cikličke strukture, kao i kombinaciju različitih strukturnih motiva. Ovi peptidi često sadrže hidrofobne i hidrofilne regije, što im omogućuje interakciju s membranama mikroba. Mnogi AMP imaju pozitivno nabijene aminokiseline, kao što su lizin i arginin, što im pomaže u vezivanju za negativno nabijene komponente mikrobnih membrana [7].

AMP djeluju na mikrobe na različite načine, ciljajući njihove vitalne strukture i funkcije. Neki peptidi djeluju na mikrobne membrane, oštećujući njihovu strukturu i uzrokujući curenje unutarstaničnih tvari. Drugi peptidi prodiru u unutrašnjost mikroba i ciljaju vitalne procese kao što su sinteza proteina i DNA replikacija. Pojednostavljeni primjer djelovanja peptida na ciljnu molekulu prikazan je slikom 2.3. Također, neki peptidi imaju imunomodulatorna svojstva, potičući imunološki odgovor organizma i potencijalno regulirajući upalne procese.

Kada govorimo o razvoju lijekova, jedna od ključnih prednosti AMP je to što su mikroorganizmi manje skloni razvoju otpornosti na njihovo djelovanje za razliku od ostalih lijekova. Otpornost na antimikrobne lijekove postala je globalni problem, ali mnogi AMP djeluju na mikrobe putem mehanizama koji se razlikuju od konvencionalnih antibiotika, što smanjuje vjerojatnost razvoja otpornosti.

Diagram of a cell membrane

Description automatically generated with medium confidence

*Slika 2.4. Primjer djelovanja AMP na membranu ciljne molekule (preuzeto iz [8])*

## Antiviralna svojstva

Antiviralni peptidi (AVP) su podvrsta AMP koja ima gotovo sve iste karakteristike, ali su specifični po tome što imaju sposobnost suzbijanja virusnih infekcija. Kao i kod AMP, AVP koriste razne mehanizme kojima suzbijaju infekcije, npr. mogu blokirati vezanje virusa, spriječiti spajanje virusa sa stanicama domaćina, prekinuti proces signaliziranja virusa ili spriječiti razmnožavanje virusa u stanicama domaćina [1]. Osim toga, AVP mogu modulirati imunološki odgovor domaćina i poticati stvaranje protuvirusnih citokina.

Predviđanje antiviralnih svojstava prije sinteze novih peptida izuzetno je važno za razvoj lijekova i prevenciju virusnih infekcija. Mnoge metode strojnog učenja i računalnog modeliranja koriste se za predviđanje antiviralnih svojstava peptida na temelju njihovih fizičko-kemijskih svojstava i strukture. Ova područja istraživanja doprinose razvoju boljih modela za predviđanje antiviralne aktivnosti peptida i optimizaciju njihovih svojstava.

# AUTOENKODERI

## Autoenkoderi općenito

Autoenkoderi su posebna vrsta neuronskih mreža čiji je cilj što vjernije rekonstruirati podatke sa ulaza na izlazu. Sastoje se od tri glavna dijela: enkodera, latentnog prostora i dekodera, vrlo pojednostavljeno prikazano na slici 3.1. Možemo reći da autoenkoder uči kako kroz enkoder sažeti podatke u latentni prostor i prikazati ih pomoću manjeg broja značajki, a pritom minimizirati grešku rekonstrukcije na izlazu dekodera.

A network with blue and yellow circles

Description automatically generated

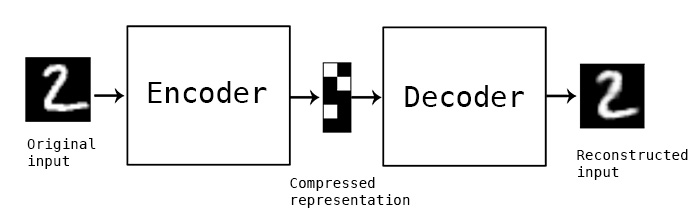
*Slika 3.1. Pojednostavljeni prikaz autoenkodera (enkoder – lijevi sloj, latentni prostor – srednji sloj, dekoder – desni sloj) (preuzeto iz [https://towardsdatascience.com/a-high-level-guide-to-autoencoders-b103ccd45924 LITR])*

Međutim, autoenkoder ne bi imao svrhu kada bi samo preslikavao podatke s ulaza na izlaz. Ono što ga čini korisnim je to da kroz proces rekonstrukcije podataka model uči koje su najvažnije značajke ulaznih podataka te kada je istreniran može se koristiti kako bi na temelju naučenog stvorio nove podatke uz dodatak ili izuzetak određenih značajki[https://www.unite.ai/what-is-an-autoencoder/ LITR]. Takav primjer možemo vidjeti i na slici 3.2. gdje autoenkoder uzima sliku znamenke napisane rukom, sažima ju u mali broj dimenzija, uči koje su najvažnije značajke te na kraju pomoću tih značajki rekonstruira sliku koja je vrlo slična originalnoj slici na ulazu.

Svaki dio autoenkodera ima svoju konkretnu ulogu u cijelome procesu. Enkoder je gusto povezana mreža i ulaz u autoenkoder. Zadaća mu je da uzme ulazne podatke te ih sažme i prikaže u latentnom prostoru, čineći tako novi prikaz podataka koji ima smanjenu dimenzionalnost.

Latentni prostor radi sa sažetim prikazom podataka. Konstruiran je tako da odredi najvažnije dijelove promatranih podataka odnosno značajke podataka koje su najvažnije za dobru rekonstrukciju. Cilj mu je odrediti koje značajke podataka trebaju biti očuvane, a koje se mogu ukloniti. Također treba uzeti u obzir i napraviti dobar omjer između veličine latentnog prikaza odnosno koliko će on biti sažet te relevantnosti značajki.

Dekoder je zadužen da uzme sažete podatke iz latentnog prostora koje je kreirao enkoder i pretvori ih ponovno u prikaz sa istim brojem dimenzija kao što su imali i početni ulazni podaci.



*Slika 3.2. Slika prolazi kroz enkoder, postaje sažeti prikaz podataka te na izlazu postaje rekonstruirana ulazna slika (preuzeto iz [https://blog.keras.io/building-autoencoders-in-keras.html LITR]*

Postoje tri vrlo bitne značajke koje definiraju rad autoenkodera [https://blog.keras.io/building-autoencoders-in-keras.html LITR]:

1. Specifični su za podatke na kojima uče – bit će u mogućnosti raditi samo s podacima koji su vrlo slični onima na kojima je model treniran, npr. autoenkoder koji je treniran na slikama na kojima se nalaze ljudska lica će raditi vrlo loše sa slikama na kojima se nalazi drveće jer su značajke koje je autoenkoder naučio specifične za ljudska lica
2. Imaju gubitke – podaci na izlazu koji su prošli kroz dekompresiju će biti lošije kvalitete u usporedbi s ulaznim podacima
3. Uče automatski iz primjera podataka – vrlo je lako trenirati algoritam da radi dobro na specifičnom tipu ulaznih podataka, odnosno nije potrebno mijenjati algoritam već je dovoljno algoritmu dati prikladne podatke za trening

Autoenkoderi imaju vrlo široki spektar primjene, ali najčešće su sljedeće[https://www.unite.ai/what-is-an-autoencoder/ LITR]:

* Smanjenje dimenzionalnosti - prolaskom kroz enkoder podaci dolaze u latentni prostor gdje su prikazani pomoću manjeg broja dimenzija u odnosu na izvorne ulazne podatke te se tako mogu lakše razumjeti ili vizualizirati; također može pomoći mrežama visokog kapaciteta naučiti korisne značajke slika što upućuje na to da autoenkoderi mogu biti korišteni za poboljšanje treninga drugih tipova neuronskih mreža
* Uklanjanje šuma iz podataka - često se koristi na slikama; osim uklanjanja šuma, autoenkoderi mogu ispraviti i druge tipove oštećenja na slikama kao što su mutne slike ili dopuniti dijelove slike koji nedostaju
* Ekstrakcija značajki - kao i smanjenje dimenzionalnosti može biti korisno za poboljšanje treninga drugih tipova neuronskih mreža jer se mogu koristiti za prepoznavanje značajki skupova podataka koji se koriste za treniranje drugih modela
* Generiranje slika - autoenkoderi mogu generirati nove, do sad nepostojeće, slike ljudi ili animiranih likova što pomaže pri konstruiranju sustava za prepoznavanje lica ili automatiziranja određenih aspekata animacije
* Predviđanje sekvence - mogu se koristiti za određivanje promjene strukture podataka kroz vrijeme, što znači da mogu generirati idući podatak u sekvenci i to dovodi do toga da autoenkoderi mogu generirati videozapise
* Stvaranje sustava preporuke – koriste se duboki autoenkoderi, a sustave stvaraju tako što će prepoznati uzorke korištenja i povezati ih s interesima korisnika; u ovoj primjeni enkoder analizira podatke o načinu interakcije, a dekoder stvara preporuke koje odgovaraju prepoznatim uzorcima

## Varijacijski autoenkoderi

U osnovi, varijacijski autoenkoder (VAE) je autoenkoder čija je distribucija enkodiranih podataka regularizirana tijekom treninga kako bi osigurali da latentni prostor ima dovoljno dobra svojstva kako mi mogli generirati nove podatke [https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73 LITR].

„Obični“ autoenkoderi često nemaju svojstva koja su potrebna za generiranje novih podataka. Kako bi to prikazali možemo zamisliti primjer gdje postoje enkoder i dekoder koji su dovoljno moćni da postave bilo koji N početni skup podataka na realnu os, odnosno svaku točku podataka enkodiraju kao realnu vrijednost u jednoj dimenziji, i dekodiraju ih bez ikakvih gubitaka pri rekonstrukciji. U tom slučaju visoki stupanj slobode koji omogućava autoenkoderu enkodiranje i dekodiranje bez ikakvih gubitaka informacija dovodi do jakog „overfittinga“ što znači da će neke točke latentnog prostora nakon dekodiranja vraćati besmislen i nerazumljiv sadržaj. Također treba uzeti u obzir da je cilj smanjenja dimenzionalnosti, uz smanjenje broja dimenzija podataka, i zadržavanje većine informacija u sažetim prikazima, a sažimanjem podataka u samo jednu dimenziju gubimo jako velik dio informacija što je vrlo pojednostavljeno prikazano slikom 3.3. Iako je ovaj primjer s jednom dimenzijom u latentnom prostoru poprilično ekstreman, on demonstrira kako autoenkoderi općenito imaju problem s regularnosti latentnog prostora i da na to treba pripaziti.

Ovaj problem nedostatka strukture u enkodiranim podacima u latentnom prostoru je poprilično razumljiv s obzirom na to da zadatak za koji je autoenkoder treniran ne zahtijeva takvu organizaciju. Autoenkoder je treniran isključivo da enkodira i dekodira podatke sa što manje gubitaka moguće, bez obzira na to kako je latentni prostor organiziran. Stoga, ako se arhitekturi autoenkodera ne priodaje posebna pažnja, normalno je da mreža koristi sve mogućnosti za „overfitting“ koje ima kako bi postigla što bolji rezultat.

A screen shot of a game

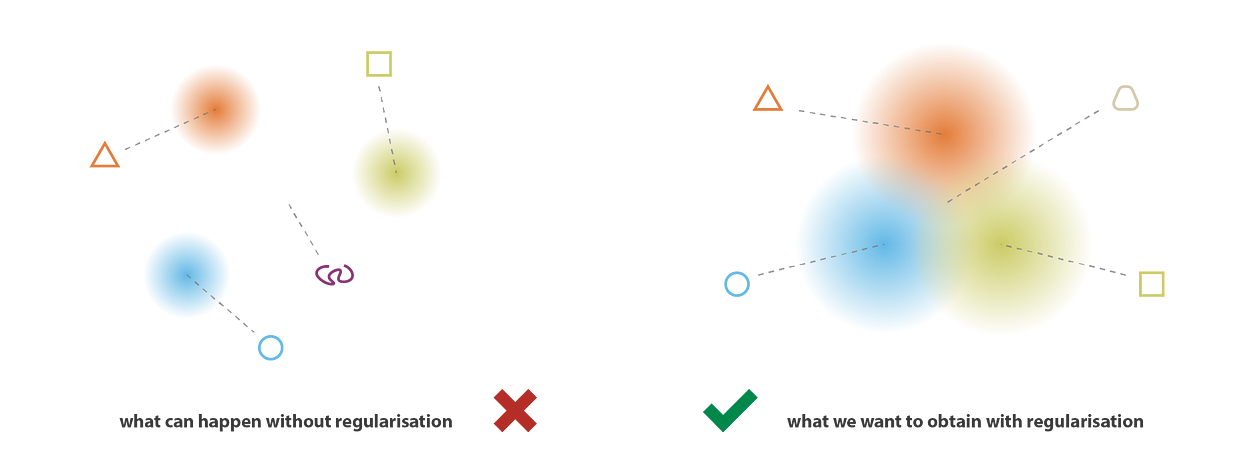
Description automatically generated

*Slika 3.3. Početni skup podataka (sredina) enkodiran u jednu dimenziju (lijevo) i u dvije dimenzije (desno) (preuzeto iz* [*https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73*](https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73) *LITR)*

Dakle, kako bi mogli koristiti dekoder autoenkodera u svrhu generiranja novih podataka moramo osigurati regularnost latentnog prostora. Jedno od rješenja za postizanje regularnosti je da tijekom procesa treninga eksplicitno uvedemo regularizaciju. Na taj način dobivamo varijacijski autoenkoder. Njegova arhitektura je ista kao i kod standardnog autoenkodera, ali postoji razlika kod procesa enkodiranja: umjesto enkodiranja ulaznih podataka kao pojedinačne točke, enkodiramo ih kao distribuciju u latentnom prostoru. Time dolazimo do sljedećeg načina treniranja modela:

1. Ulazni podaci se enkodiraju kao distribucija u latentnom prostoru
2. Točka iz latentnog prostora se uzima kao uzorak iz te distribucije
3. Uzorkovana točka se dekodira i izračunava se greška rekonstrukcije
4. Greška rekonstrukcije se vraća unazad kroz mrežu

Regularnost koja se očekuje od latentnog prostora kako bi se omogućio proces generiranja novih podataka zahtijeva da latentni prostor bude kontinuiran i potpun. Kontinuiranost osigurava da dvije točke koje su blizu jedna drugoj u latentnom prostoru ne vraćaju potpuno drugačiji sadržaj kada se dekodiraju. Potpunost osigurava da za odabranu distribuciju svaka točka koja se uzme kao uzorak iz latentnog prostora i dekodira vraća smisleni sadržaj. Važnost regularizacije latentnog prostora pojednostavljeno je prikazana slikom 3.4.



*Slika 3.4. Slikoviti prikaz latentnog prostora i podataka bez regularizacije (lijevo) i s regularizacijom (desno) (preuzeto iz [https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73 LITR])*

Sama činjenica da VAE enkodira ulazne podatke kao distribucije umjesto pojedinačnih točaka nije dovoljno da osigura kontinuitet i potpunost. Zbog toga je potrebno regularizirati matricu kovarijance i srednju vrijednost distribucija koje vraća enkoder. U praksi se to postiže osiguravanjem da distribucija bude što sličnija normalnoj distribuciji (centrirana i reducirana). Na taj način se osigurava da matrica kovarijance bude blizu jediničnoj, spriječavajući točkaste distribucije, i da srednja vrijednost bude blizu nuli, spriječavajući enkodirane distribucije da budu predaleko jedne od drugih.

Cijeli postupak regularizacije dovodi do toga da spriječavamo model da enkodira podatke daleko jedne od drugih u latentnom prostoru i potiče enkodirane distribucije da se preklapaju koliko god je to moguće, zadovoljavajući na taj način uvjete kontinuiteta i potpunosti i stvarajući gradijent u podacima enkodiranim u latentnom prostoru. Npr. točka u latentnom prostoru koja je na pola puta između srednjih vrijednosti dvije enkodirane distribucije koje dolaze od različitih podataka u treningu trebala bi biti dekodirana u nešto što bi odgovaralo vrijednosti između podatka koji je dao prvu distribuciju i podatka koji je dao drugu distribuciju. Ovo je vrlo jednostavno i slikovito prikazano na slici 3.5.

A group of colorful circles and squares

Description automatically generated

*Slika 3.5. Latentni prostor i dekodirani podaci uz prisutnu regularizaciju (preuzeto iz [https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73 LITR])*

Dodatno treba napomenuti dvije funkcije koje računaju gubitke kod VAE, a koje osiguravaju regularnost i bit će spomenute u kasnijim poglavljima: gubici rekonstrukcije i Kullback-Leibler divergencija (KL divergencija). Kada se minimiziraju gubici rekonstrukcije, to za mrežu znači da mora učiniti da izlazni podaci izgledaju što više kao ulazni podaci i da pri tome može koristiti sve što ima na raspolaganju. To će osigurati da model grupira uzorke koji su slični jedni drugima, odnosno svaka klasa će biti u jednoj skupini, a slične klase će biti enkodirane blizu jedna drugoj te je tako zadovoljen kontinuitet latentnog prostora.

Međutim, kada uzimamo u obzir samo gubitke rekonstrukcije nemamo pravilo koje osigurava da se grupacije donekle preklapaju i da su na taj način povezane jer u namjeri modela da minimizira gubitke rekonstrukcije, on može u potpunosti razdvojiti grupacije enkodiranih podataka te tada nemamo zadovoljenu potpunost latentnog prostora. U tom slučaju, uz izračun gubitaka rekonstrukcije dodajemo i KL divergenciju. KL divergencija je specifična za VAE i ne postoji kod klasičnih autoenkodera. KL divergencija zapravo izračunava divergenciju između dvije distribucije vjerojatnosti, odnosno njen gubitak se povećava kada se distribucija vjerojatnosti koju je generirao enkoder razlikuje od standardne normalne distribucije. To znači da je model prisiljen učiti tako da enkoder „gura“ distribucije vjerojatnosti odnosno uzorke što je bliže moguće jedne drugima i tako zadovoljava potpunost latentnog prostora [https://github.com/christianversloot/machine-learning-articles/blob/main/what-is-a-variational-autoencoder-vae.md LITR].

# MODEL

## Alati

Alati pomoću kojih je izrađen model, napravljena vizualizacija i analiza rezultata odabrani su tako da nude što veću moć i više mogućnosti, a da budu relativno jednostavni za korištenje i tumačenje. Kod je napisan u programskom jeziku Python unutar Jupyter Notebook sučelja pomoću knjižnica Keras i Tensorflow koje su specijalizirane za strojno učenje.

### Python

Python je jedan od najpopularnijih programskih jezika u svijetu već dugi niz godina. S obzirom na to da je Python jezik opće namjene, može se koristiti za razne stvari kao što su izgradnja web stranica i softvera, automatizacija procesa ili analiza podataka. Ono što ga također čini toliko popularnim je činjenica da je vrlo intuitivan i lagan za učenje kod početnika. U ovom radu korištena je verzija Pythona 3.4.10.

Python je također vrlo moćan jezik kada govorimo o strojnom učenju. Sposoban je provoditi kompleksne statističke izračune, graditi algoritme strojnog učenja, manipulirati podacima i analizirati ih, stvarati razne vizualizacije podataka te izvoditi razne druge zadatke vezane uz rad s podacima. Također, postoje mnoge knjižnice koje dodatno olakšavaju pisanje koda za analizu podataka i strojno učenje te čine kod bržim i efikasnijim, kao što su Tensorflow i Keras [https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python LITR].

### Jupyter Notebook

Jupyter Notebook je aplikacija koja nam pruža okruženje za interaktivno programiranje, analizu podataka i prikaz rezultata. Omogućava korisnicima da kombiniraju kod, tekst, vizualizacije i matematičke formule u jednom dokumentu, stvarajući dinamične i interaktivne „notebookove“ odnosno bilježnice. Alat je u potpunosti otvorenog koda. [https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what\_is\_jupyter.html LITR]

Blokovi koda, koji se nazivaju ćelijama, mogu se pokretati zasebno što omogućava izvršavanje željenih dijelova koda u stvarnom vremenu i time interaktivno istraživanje podataka. Ovo je vrlo korisno za eksperimentiranje s različitim pristupima analize podataka ili provjeravanje ispravnosti koda. U konkretnom slučaju ovog rada, Jupyter Notebook je imao veliku prednost zbog toga što su se analiza podataka, arhitektura modela, izračuni, trening, vizualizacije i interpretacija rezultata, sve moglo prikazati u jednom dokumentu. Slikom 4.1. prikazan je izgled Jupyter Notebook sučelja i kratki kod koji pokazuje mogućnost da se izračuni, obrada podataka i vizualizacija sve obavlja na jednom mjestu.

A screen shot of a computer

Description automatically generated

*Slika 4.1. Izgled sučelja i demonstracija mogućnosti Jupyter Notebooka [ preuzeto iz* [*https://docs.bokeh.org/en/latest/\_images/notebook\_inline.png*](https://docs.bokeh.org/en/latest/_images/notebook_inline.png) *LITR]*

### Tensorflow

Tensorflow je besplatna knjižnica otvorenog koda za strojno učenje i umjetnu inteligenciju. Ima vrlo široki spektar primjene, ali posebno se fokusira na rad s dubokim neuronskim mrežama. Prva verzija knjižnica objavljena je 2015. godine, a nova i unaprijeđena verzija pod imenom „Tensorflow 2.0“ objavljena je 2019. godine i ona je korištena u ovom radu. Tensorflow se može koristiti u raznim programskim jezicima, kao što su Python, Javascript, C++ i Java, što mu osigurava veliku fleksibilnost. [https://en.wikipedia.org/wiki/TensorFlow LITR]

Tensorflow ima praktičnu primjenu najčešće za klasifikaciju i detekciju (prepoznavanje objekata na slikama i analiza teksta), generiranje i sintezu (stvaranje umjetnih slika, glazbe ili teksta) te „reinforcement learning“ (inteligentni sustavi koji uče putem interakcije s okolinom). Tensorflow uglavnom radi s višedimenzionalnim poljima koja se nazivaju tenzorima i postoje kao zasebni objekti. Neke od prednosti su mu da omogućava rad na više uređaja, uključujući CPU, GPU i TPU (Tensor Processing Unit), velika prilagodljivost i kontrola prilikom izgradnje modela te opširna i detaljna dokumentacija.

### Keras

Keras je knjižnica otvorenog koda koja pruža Python sučelje za strojno učenje i rad s neuronskim mrežama uz fokus na duboko učenje, odnosno služi kao sučelje za Tensorflow knjižnicu. Keras pokriva sve korake strojnog učenja, od procesiranja podataka preko podešavanja hiperparametara do konačnog modela. Razvijen je s fokusom na omogućavanje brzog eksperimentiranja.

Uz Keras možemo u potpunosti iskoristiti skalabilnost i kompatibilnost s više platformi koju nudi Tensorflow. Keras je dizajniran da smanji kognitivno opterećenje programera tako što će [https://www.tensorflow.org/guide/keras LITR]:

* Ponuditi jednostavno, dosljedno sučelje
* Minimizirati broj potrebnih radnji za najčešće slučajeve korištenja
* Pružiti jasne poruke kod grešaka
* Pratiti princip progresivnog otkrivanja: lako je započeti s korištenjem, a zahtjevnije zadatke je moguće riješiti usputnim učenjem dok radimo na modelu
* Pomoći pisati jasan i čitljiv kod

Glavne strukture podataka u Kerasu su slojevi i modeli. Slojevi podrazumijevaju ulazne i izlazne transformacije podataka dok su modeli objekti koji povezuju slojeve i koji se tada mogu trenirati na podacima.

### Ostale Python knjižnice

Pandas je moćna i fleksibilna knjižnica otvorenog koda za analizu i manipulaciju podataka. Glavna struktura s kojom radi Pandas je DataFrame, odnosno dvodimenzionalna tablica s označenim redovima i stupcima. U ovom radu je Pandas knjižnica korištena za učitavanje podataka i obradu prije treniranja modela.

Scikit-learn ili skraćeno sklearn je knjižnica za strojno učenje i analizu podataka. Sadrži razne algoritme strojnog učenja za klasifikaciju, regresiju i „clustering“ te alate za obradu i pripremu podataka. U ovom radu je uz Pandas knjižnicu korištena za obradu podataka prije treniranja modela.

Numpy je knjižnica koja omogućava brzu i učinkovitu manipulaciju numeričkim podacima. Pruža podršku za rad s velikim, višedimenzionalnim poljima i matricama, zajedno s velikim brojem matematičkih funkcija koje olakšavaju rad s tim poljima.

Matplotlib je knjižnica koja se koristi za stvaranje različitih vrsta grafova, grafikona i vizualnih prikaza podataka. Pomoću nje mogu se kreirati statičke, animirane i interaktivne vizualizacije kako bi lakše predstavili i razumjeli svoje podatke.

Scipy je knjižnica za Python koja proširuje njegove mogućnosti za znanstvenu i tehničku analizu. Sadrži razne module za optimizaciju, linearnu algebru, statistiku, integraciju, interpolaciju, procesiranje signala i slika te druge znanstvene proračune. U ovom radu su iz Scipy knjižnice korišteni moduli za korelacijsku analizu kako bi obradili krajnje rezultate rada.

Biopython je knjižnica za Python koja nudi niz alata za bioinformatičku analizu i manipulaciju bioloških podataka. Neki od alata omogućavaju rad s biološkim sekvencama, analizu genoma, analizu strukture proteina i slično. Također omogućava da se programski pristupi bazama podataka koje sadrže biološke informacije. U ovom radu knjižnica Biopython je korištena za izračun sličnosti među peptidima temeljene na njihovim sekvencama.

## Podaci

Skup podataka koji je korišten za trening i testiranje modela sastoji se od dva manja skupa podataka. Prvi sadrži 1050 sekvenci peptida zajedno s oznakom antiviralne aktivnosti (603 pozitivnih i 447 negativnih) gdje „1“ označava prisutnost, a „0“ odsutnost antiviralne aktivnosti. Njemu je pridodan drugi skup podataka koji sadrži 9409 sekvenci peptida s oznakom antimikrobne aktivnosti (3708 pozitivnih i 5701 negativnih) gdje „3“ označava prisutnost, a „2“ odsutnost antimikrobne aktivnosti. Konačni skup dakle sadrži ukupno 10459 sekvenci peptida, a radi bolje preglednosti informacije o podjeli podataka prikazane su i tablicom 4.1.

*Tablica 4.1. Sadržaj skupa podataka korištenog za treniranje i testiranje modela*

|  |  |  |  |
| --- | --- | --- | --- |
|  | Pozitivni | Negativni | Ukupno |
| Antiviralna svojstva | 603 | 447 | 1050 |
| Antimikrobna svojstva | 3708 | 5701 | 9409 |
| Ukupno | 4311 | 6148 | 10459 |

Sljedeći korak bio je izračunavanje značajki svih peptida u skupu podataka. Za to je korišten Python paket pod imenom „peptides“ te je izračunato 88 značajki za svaki peptid. U nastavku se nalazi sažeti popis svih značajki koje su dostupne u paketu „peptides“ i koje su izračunate za peptide u korištenom skupu podataka [https://peptides.readthedocs.io/en/stable/ LITR]:

* Statistike aminokiselina:
  + Broj pojavljivanja određene aminokiseline u sekvenci peptida
  + Učestalost pojavljivanja u sekvenci peptida
* QSAR deskriptori:
  + BLOSUM indeksi
  + Cruciani svojstva
  + FASGAI vektori
  + Kidera faktori
  + MS-WHIM rezultati
  + PCP deskriptori
  + ProtFP deskriptori
  + Sneath vektori
  + ST-skale
  + T-skale
  + VHSE-skale
  + Z-skale
* Profili sekvenci:
  + Profil hidrofobnosti
  + Profil momenta hidrofobnosti
  + Pozicija membrane
* Fizikalno-kemijska svojstva:
  + Alifatski indeks
  + Indeks nestabilnosti
  + Ukupni teoretski naboj
  + Izoelektrična točka
  + Molekularna masa
* Biološka svojstva
  + Strukturna klasa

Konačni skup podataka sa izračunatim značajkama svih peptida spremljen je u datoteku *features.csv* s konačnim dimenzijama od 10460 redaka gdje se u prvom retku nalaze nazivi stupaca, a ostalo su peptidi, te 89 stupaca gdje se u prvom stupcu nalaze sekvence peptida, a u ostalima su izračunate značajke. Slikom 4.2. prikazan je isječak programskog koda pomoću kojeg su izračunate značajke peptida. Iz datoteke *amp+avp.csv* učitane su sekvence peptida, za njih su izračunate značajke te je sve skupa spremljeno u datoteku *features.csv*. Izlaz nakon zadnje linije prikazanog koda prikazan je na slici 4.3., a u kojem je vidljiv izgled konačnog skupa podataka, njegove dimenzije i neke od izračunatih značajki.

A screenshot of a computer code

Description automatically generated

*Slika 4.2. Isječak programskog koda kojim su izračunate značajke za sve peptide u skupu podataka*

A table of numbers and letters

Description automatically generated

*Slika 4.3. Konačni skup podataka objedinjen u datoteci features.csv*

Treba napomenuti da uz glavni skup podataka postoji i skup podataka koji se nalazi u datoteci *labels.csv* te koji sadrži ranije spomenute oznake aktivnosti svih peptida u istom poretku kao i u datoteci *features.csv*. Oznaka „0“ označava odsutnost viralne aktivnosti, oznaka „1“ prisutnost antiviralne aktivnosti, oznaka „2“ odsutnost antimikrobne aktivnosti, a oznaka „3“ prisutnost antimikrobne aktivnosti. Oznake ovisno o aktivnosti i vrsti peptida su prikazane i tablicom 4.2. Ovaj skup podataka neće biti korišten prilikom treniranja modela jer autoenkoderima nisu potrebni *labeli* za učenje, ali će biti vrlo koristan prilikom vizualizacije kako bi mogli prikazati peptide odgovarajućom bojom ovisno o vrsti njihove aktivnosti.

*Tablica 4.2. Oznake peptida ovisno o njihovoj vrsti i aktivnosti u skupu podataka labels.csv*

|  |  |  |
| --- | --- | --- |
|  | Neaktivan | Aktivan |
| **AVP** | 0 | 1 |
| **AMP** | 2 | 3 |

## Metode

U ovom poglavlju bit će prikazane i objašnjene sve metode kojima se došlo do konačnih rezultata. To uključuje izgradnju modela VAE, enkodera i dekodera, obradu podataka kako bi ih VAE mogao koristiti za trening, optimizaciju parametara modela i sami trening te njegovu uspješnost. Većina ovog dijela koda je napisana po uzoru na Kerasov tutorial za VAE [https://keras.io/examples/generative/vae/ LITR] i članak o izgradnji VAE sa GitHuba koji je napisao Christian Versloot [https://github.com/christianversloot/machine-learning-articles/blob/main/how-to-create-a-variational-autoencoder-with-keras.md LITR].

Na početku je naravno potrebno napraviti *import* glavnih knjižnica i njihovih pojedinih dijelova koji će biti korišteni u kodu, Numpy, Tensorflow, Keras, Pandas, Scikit-learn, koje su navedene i objašnjene u poglavlju 4.1., a što je prikazano slikom 4.4.

A screen shot of a computer code

Description automatically generated

*Slika 4.4. „Import“ glavnih knjižnica i njihovih dijelova koji će biti korišteni u kodu*

Sljedeći dio koji se mora definirati prije nego što se krene u izgradnju same arhitekture VAE je sloj za uzorkovanje, odnosno „sampling layer“. Kao što mu ime govori, njegov zadatak je da uzima uzorke z iz latentnog prostora, gdje je z vektor koji predstavlja enkodirani peptid odnosno njegove značajke. U ovom dijelu definiramo i varijablu latent\_dim koja označava broj dimenzija koje će sadržavati latentni prostor. Slikom 4.5. prikazan je ovaj dio koda.

A computer screen shot of a computer code

Description automatically generated

*Slika 4.5. Definiranje sloja za uzorkovanje i varijable koja definira broj dimenzija u latentnom prostoru*

### Enkoder

Kreiranje enkodera je proces koji se sastoji od tri koraka: prvo ga moramo definirati, drugi korak je „trik reparametrizacije“ koji nam omogućava da povežemo enkoder s dekoderom i tako definiramo VAE kao cjelinu i na kraju treći korak u kojem instanciramo enkoder. Prvi korak prikazan je slikom 4.6., na kojoj vidimo na koji način se definira enkoder i kako se povezuju slojevi neuronske mreže.

A computer code with text

Description automatically generated with medium confidence

*Slika 4.6. Definiranje enkodera*

U nastavku su pojašnjeni svi slojevi enkodera:

* **encoder\_inputs = keras.Input(shape=(88, 1))** - prvi sloj je ulazni sloj koji prihvaća ulazne podatke u obliku (88, 1), dakle jedan po jedan peptid opisan sa 88 izračunatih značajki
* **x=layers.Conv1D(8,3,activation='relu',padding='same',dilation\_rate=2)(encoder\_inputs)**- drugi sloj je jednodimenzionalni konvolucijski sloj s 8 filtera; vrijednost 2 parametra *dilation\_rate* označava da postoje praznine između vrijednosti u konvolucijskom kernelu; izlazni tenzor sada ima oblik (None, 88, 8)
* **x = layers.MaxPooling1D(2)(x)** – ovaj sloj obavlja operaciju *max pooling* na izlaznom tenzoru prethodnog sloja i tako mu smanjuje dimenzije na pola pa je sada izlazni tenzor oblika (None, 44, 8); ova operacija smanjuje dimenzionalnost podataka primjenom max filtera na podregije izvornih podataka; time spriječava *over-fitting* jer pruža apstraktni oblik izvornih podataka i smanjuje računalnu zahtjevnost modela smanjivanjem broja parametara koje treba učiti; slikom 4.7. prikazan je princip *max poolinga*
* **x = layers.Conv1D(16, 3, activation='relu', padding='same', dilation\_rate=2)(x)** – još jedan jednodimenzionalni konvolucijski sloj, ali ovaj put sa 16 filtera što dovodi do oblika izlaznog tenzora (None, 44, 16)
* **x = layers.MaxPooling1D(2)(x)** – još jedan *max pooling* sloj koji dodatno smanjuje dimenzije podataka na oblik (None, 22, 16)
* **x = layers.Flatten()(x)** – ovaj sloj služi da „izravna“ višedimenzionalne tenzore u jednu dimenziju te je sada oblik podataka (None, 352); ovo moramo napraviti zato što idući slojevi zahtijevaju da podaci budu takvog oblika
* **encoder = layers.Dense(128, activation='relu')(x)** – *Dense* sloj se klasificira kao potpuno povezani sloj i on mapira tenzor iz prethodnog sloja u prikaz s manje dimenzija u latentnom prostoru te ovdje dolazimo do „najužeg“ dijela odnosno sredine autoenkodera; izlazni podaci su oblika (None, 128)
* **z\_mean = layers.Dense(latent\_dim, name="z\_mean")(encoder)** – još jedan potpuno povezani sloj koji mapira tenzor *encoder* kao srednje vrijednosti u latentnom prostoru; oblik podataka je sada (None, latent\_dim) odnosno (None, 2) jer smo na početku definirali varijablu *latent\_dim = 2*
* **z\_log\_var = layers.Dense(latent\_dim, name="z\_log\_var")(encoder)** – slično kao prethodni sloj, još jedan potpuno povezani sloj koji mapira tenzor *encoder* ali ovaj put kao logaritam varijance u latentnom prostoru; izlaz je također (None, 2)
* **z = Sampling()([z\_mean, z\_log\_var]) ­–** koristimo prethodno definirani sloj za uzorkovanje koji uzima *z\_mean* i *z\_log\_var* kao ulaz i generira uzorak iz latentnog prostora koristeći „trik reparametrizacije“; trik reparametrizacije služi da izmijenimo parametre uzorka u oblik koji je potreban da bi mogli koristiti gradijentni spust za preciznu procjenu gradijenata
* **encoder = keras.Model(encoder\_inputs, [z\_mean, z\_log\_var, z], name="encoder")** – instanciramo model enkodera uzimajući *encoder\_inputs* kao ulaz te [*z\_mean*, *z\_log\_var*, *z*] kao izlaz
* **encoder.summary()** – ova linija ispisuje kratki pregled enkodera, prikazujući slojeve i njihove oblike izlaznih podataka; izlaz ove linije prikazan je slikom 4.8.

A close-up of a grid

Description automatically generated

*Slika 4.7. Princip max pooling operacije na dvodimenzionalnoj matrici (preuzeto iz [https://computersciencewiki.org/index.php/Max-pooling\_/\_Pooling LITR])*

A screenshot of a computer program

Description automatically generated

*Slika 4.8. Kratki pregled enkodera*

### Dekoder

Kreiranje dekodera je jednostavniji proces od kreiranja enkodera i sastoji se samo od dva koraka: definicije i instanciranja dekodera. Prvi korak prikazan je slikom 4.9., na kojoj vidimo na koji način se definira dekoder i kako se povezuju slojevi neuronske mreže.

A screen shot of a computer code

Description automatically generated

*Slika 4.9. Definiranje dekodera*

U nastavku su pojašnjeni svi slojevi dekodera:

* **latent\_inputs = keras.Input(shape=(latent\_dim,))** – prvi sloj je ulazni sloj dekodera koji kao ulaz očekuje točku iz latentnog prostora; podatak koji ovaj sloj dobiva na ulaz je oblika (None, 2) kao i u zadnjem sloju enkodera
* **x = layers.Dense(128, activation='relu')(latent\_inputs) –** ovaj potpuno povezani sloj mapira točku iz latentnog prostora u prikaz s više dimenzija; izlazni tenzor je oblika (None, 128)
* **x = layers.Dense(88 \* 8)(x) –** još jedan potpuno povezani sloj koji mapira tenzor iz prethodnog sloja u prikaz s još više dimenzija što rezultira izlaznim oblikom (None, 704)
* **x = layers.Reshape((88, 8))(x) –** ovaj sloj preoblikuje prethodni tenzor u tenzor s dimenzijom više i oblikom (None, 88, 8)
* **x = layers.Conv1D(16, 3, activation='relu', padding='same')(x) –** jednodimenzionalni konvolucijski sloj sa 16 filtera nakon kojeg izlazni tenzor ima oblik (None, 88, 16)
* **x = layers.UpSampling1D(2)(x)** – ovaj sloj obavlja povećanje uzorkovanja odnosno *upsampling* na prethodnom tenzoru i udvostručuje njegovu duljinu što na kraju daje oblik (None, 176, 16)
* **x = layers.Conv1D(8, 3, activation='relu', padding='same')(x)** – još jedan jednodimenzionalni konvolucijski sloj, ovaj put s 8 filtera što dovodi do oblika (None, 176, 8)
* **x = layers.UpSampling1D(2)(x) –** još jedan sloj za povećanje uzorkovanja koji ponovno udvostručava duljinu tenzora i daje mu oblik (None, 352, 8)
* **decoded = layers.Conv1D(1, 3, activation='sigmoid', padding='same')(x) –** konačni jednodimenzionalni konvolucijski sloj s jednim filterom što izlaznom tenzoru *decoded* daje oblik (None, 352, 1)
* **decoded = layers.Cropping1D(cropping=(0, 264))(decoded) –** sloj za izrezivanje koji uklanja potreban broj dimenzija da bi konačan tenzor *decoded* imao oblik (None, 88, 1)
* **decoder = keras.Model(latent\_inputs, decoded, name="decoder") -** instanciramo model dekodera uzimajući *latent\_inputs* kao ulaz te tenzor *decoded* kao izlaz
* **decoder.summary()** - ova linija ispisuje kratki pregled dekodera, prikazujući slojeve i njihove oblike izlaznih podataka; izlaz ove linije prikazan je slikom 4.10.

A screenshot of a computer program

Description automatically generated

*Slika 4.10. Kratki pregled dekodera*

### VAE

U trenutku kada imamo definiran enkoder i dekoder možemo ih povezati i tako dobiti konačnu arhitekturu varijacijskog autoenkodera. Kao što vidimo prema kratkom isječku koda i njegovom ispisu na slici 4.11., ulaz u VAE, odnosno enkoder, je oblika (None, 88, 1). Izlaz iz enkodera, odnosno ulaz u dekoder je oblika (None, 2). Izlaz iz VAE, odnosno dekodera, je također oblika (None, 88, 1). Dakle možemo reći da je izlaz cijelog VAE zapravo izvorni ulaz, enkodiran enkoderom i dekodiran dekoderom.

A screenshot of a computer program

Description automatically generated

*Slika 4.11. Instanciranje i kratki pregled varijacijskog autoenkodera*

Međutim, treba napomenuti kako je za izgradnju modela definirana nova klasa „VAE“, što je vidljivo i iz linije vae = VAE(encoder, decoder) na slici 4.11, koja nasljeđuje klasu keras.Model jer su neke metode trebale biti prilagođene za ovaj specifičan slučaj. U nastavku će biti objašnjeni i prikazani slikama dijelovi koda kojima se definirala klasa VAE i njene metode.

Na slici 4.12., kao što je već spomenuto, definiramo novu klasu VAE koja nasljeđuje klasu keras.Model. Slijedi konstruktor metoda u kojoj ćemo spojiti enkoder i dekoder u jedan model. Ona prihvaća dva argumenta, naravno enkoder i dekoder, koje smo instancirali ranije. „\*\*kwargs“ u argumentima služi kako bi prihvatio bilo kakve dodatne ključne riječi kao argumente. Zatim pozivamo konstruktor klase roditelja keras.Model i prosljeđujemo dodatne argumente ako ih je bilo. Slijedi dodjela ranije instanciranih enkodera i dekodera atributima encoder i decoder. Nakon toga moramo inicijalizirati atribute metrika koje će pratiti gubitke pri treniranju modela. Metrike koje ćemo koristiti su ukupan gubitak, gubitak rekonstrukcije i KL gubitak, odnosno KL divergencija. Na slici 4.12. je osim opisanog do sada prikazana i *metrics* metoda koja se definira kao *property* klase VAE. Njena zadaća je da vraća listu prethodno definiranih metrika: total\_loss\_tracker, reconstruction\_loss\_tracker i kl\_loss\_tracker.

A computer screen shot of a program

Description automatically generated

*Slika 4.12. Definiranje klase VAE, konstruktora i metode metrics*

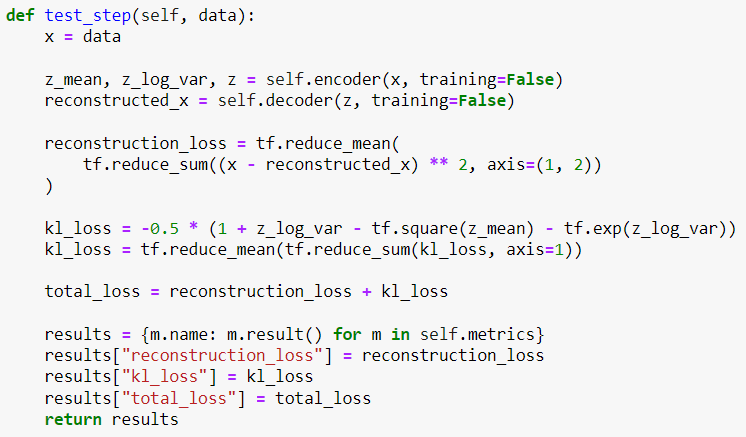
Slijedi definiranje *train\_step* i *test\_step* metoda kako bi prilagodili procese treniranja i testiranja. Prva na redu je *train\_step* metoda, prikazana slikom 4.13. Ona definira kako izgleda jedan korak procesa treniranja modela VAE. Pomoću linije *with tf.GradientTape() as tape* postavljamo kontekst gdje će operacije biti “spremljene na vrpcu” što će nam kasnije omogućiti računanje gradijenata. Unutar tog konteksta prvo prosljeđujemo ulazne podatke u enkoder kako bi dobili latentne varijable *z\_mean, z\_log\_var* i *z*,a zatim prosljeđujemo latentnu varijablu *z* u dekoder kako bi dobili renkonstruirane podatke. Slijede izračuni gubitaka: gubitak rekonstrukcije je srednja kvadrirana greška između ulaznih i rekonstruiranih podataka sumirana preko odgovarajućih osi, gubitak KL divergencije predstavlja razliku između naučene distribucije i standardne Gaussove distribucije, a *total\_loss* odnosno ukupni gubici su suma gubitaka rekonstrukcije i gubitaka KL divergencije. Sada izvan postavljenog konteksta izračunavamo gradijente ukupnih gubitaka s obzirom na težine modela koje se mogu trenirati te ih primjenjujemo na te težine. Zatim ažuriramo stanja metrika koje mjere gubitke kako bi ih mogli pratiti i na kraju vraćamo rječnik koji sadrži sve definirane metrike za mjerenje gubitaka i koje ćemo koristiti za praćenje procesa treniranja.

A computer screen shot of a program

Description automatically generated

*Slika 4.13. Definiranje train\_step metode*

Nakon train\_step metode slijedi test\_step metoda. Ona definira kako izgleda jedan korak testiranja odnosno validacije modela VAE i vrlo je slična train\_step metodi, osim što ne izračunava gradijente i ne ažurira težine s obzirom da se radi o fazi evaluacije. Najprije postavljamo ulazne podatke kao x te izračunavamo latentne varijable z\_mean, z\_log\_var i z pomoću enkodera. Argument training=False osigurava da slojevi enkodera rade u načinu rada za zaključivanje umjesto za treniranje. Zatim rekonstruiramo podatke x pomoću enkodera i latentne varijable z. Slijede izračuni gubitaka na isti način kao i u train\_step metodi. Na kraju imamo rječnik results koji sadrži izračunate gubitke i metrike, a pomoću kojih će se pratiti proces evaluacije.



*Slika 4.13. Definiranje test\_step metode*

### Obrada podataka

Nakon definirane arhitekture modela, potrebno je pripremiti ulazne podatke na način da ih model VAE može prihvatiti i efikasno učiti na njima. Slikom 4.14. prikazan je dio koda u kojem je odrađeno učitavanje i obrada ulaznih podataka, a koji će biti objašnjen u nastavku.

Najprije radimo *import* funkcije za skaliranje podataka MinMaxScaler iz knjižnice Scikit-learn pomoću koje ćemo odraditi normalizaciju podataka. Zatim učitavamo podatke iz .csv datoteka koje su opisane u poglavlju 4.2., *features.csv* i *labels.csv*, a koje sadrže sekvence i izračunate značajke peptida te oznake aktivnosti peptida. Slijedi funkcija *train\_test\_split* iz knjižnice Scikit-learn koja automatski dijeli podatke u skupove za trening i testiranje. Funkcija odmah dijeli i podatke *x* koji sadrže izračunate značajke peptida i podatke *y* koji sadrže oznake peptida koje će biti korištene isključivo prilikom vizualizacije. Argument *test\_size=0.2* unutar te funkcije određuje da će se podaci podijeliti na način da 20% podataka bude korišteno za testiranje, a ostalih 80% će se koristiti za trening. Zatim imamo dvije linije koje služe kako bi odvojili prvi stupac glavnog skupa podataka, u kojem se nalaze sekvence peptida, u zasebne varijable *x\_train\_seq* i *x\_test­\_seq* koje će nam koristiti kasnije pri izračunu sličnosti između peptida temeljene na njihovim sekvencama. Slično tome, slijede dvije linije koje odvajaju sve stupce osim prvog, u kojima se nalaze izračunate značajke peptida, u zasebne varijable *x\_train\_features* i *x\_test\_features* jer želimo da model uči isključivo pomoći izračunatih značajki, bez znanja o sekvencama peptida. Nakon toga instanciramo objekt MinMaxScaler funkcije, primjenjujemo ga na varijable u kojima su spremljene značajke peptida i spremamo u varijable *x\_train* i *x\_test*. MinMaxScaler normalizira podatke, odnosno postavlja ih u raspon od 0 do 1 na način da najmanju vrijednost značajke postavi kao 0, najveću vrijednost značajke postavi kao 1, a ostatak vrijednosti skalira u odnosu na minimum i maksimum. To radi za svaki stupac posebno tako da smo sigurni da će sve značajke biti ispravno normalizirane bez obzira na moguće razlike u njihovim izvornim rasponima vrijednosti. Na *x\_train* koristimo *fit\_transform* kako bi funkcija izračunala parametre za skaliranje na temelju podataka za treniranje i tek tada primijenila skaliranje. Na *x\_test* koristimo običan *transform* kako bi na testnim podacima primijenili skaliranje s parametrima naučenim iz podataka za trening. Na kraju svim glavnim varijablama: *x\_train*, *x\_test*, *y\_train*, i *y\_test*, dodajemo još jednu dimenziju odnosno novu os kako bi oblik podataka odgovarao onome što model očekuje na ulazu.

A screenshot of a computer program

Description automatically generated

*Slika 4.14. Učitavanje i obrada ulaznih podataka*

### Gridsearch

### Trening

# REZULTATI

# ZAKLJUČAK

Peptidi s antiviralnim svojstvima sve više obećavaju u liječenju bolesti izazvanih virusima, a da bi se mogli koristiti i primijeniti treba ih prvo otkriti. Kako ne bi morali testirati svaki peptid za svaki virus, što je vremenski zahtjevno, skupo i neefikasno, tu je vrlo obećavajuće postalo predviđanje antiviralne aktivnosti peptida prema njihovim fizikalno-kemijskim svojstvima.

Takvo predviđanje može biti vrlo učinkovito, ako se dobro izvede. Tako je i ovim jednostavnim modelom strojnog učenja postignuta točnost od preko 80% što je vrlo dobro za neke jednostavne potrebe. Ugađanjem nekih parametara i uključivanjem nekih dodatnih svojstava prema kojima bi se mogla predviđati antiviralna aktivnost, npr. sekundarnih struktura, koje su bile zamišljene kao dio ovog modela, ali zbog određenih problema na kraju nisu uključene u model, mogla bi se i povećati točnost predviđanja i tako bi ono moglo imati i širu primjenu i to čak za neka veća i važnija znanstvena istraživanja. Predviđanje antiviralne aktivnosti peptida predstavlja jedan težak, kompleksan, ali i zanimljiv zadatak te kada bi se shvatio i iskoristio potencijal koji nudi, to bi moglo vrlo pozitivno utjecati na budućnost i ostvariti velike korake u medicini, a samim time i ljudskim životima.

VAE designed to learn a meaningful latent representation of peptide sequences and compare peptide similarity based on both their original sequences and their representations in the latent space. It's a powerful approach for exploring structural relationships among peptides and can have applications in various fields such as bioinformatics, drug discovery, and molecular biology.

# LITERATURA

[1] Kuan Y. Chang, Je-Ruei Yang: Analysis and prediction of highly effective antiviral peptides based on random forests, s Interneta, <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3734225/>, 7. kolovoza 2023.

[2] Vilas Boas LCP, Campos ML, Berlanda RLA, de Carvalho Neves N, Franco OL: Antiviral peptides as promising therapeutic drugs, s Interneta, <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7079787/>, 8. kolovoza 2023.

[3] Peptides Guide, s Interneta, <https://peptidesguide.com>, 18. srpnja 2023.

[4] Peptide Sciences, s Interneta, <https://www.peptidesciences.com/information/peptides-vs-proteins/>, 18. srpnja 2023.

[5] Courtney Simons: Structure of amino acids and proteins, s Interneta, <https://cwsimons.com/structure-of-amino-acids-and-proteins/>, 5. srpnja 2020.

[6] Hitchhiking with Nature: Snake Venom Peptides to Fight Cancer and Superbugs - Scientific Figure on ResearchGate, s Interneta, <https://www.researchgate.net/figure/Principal-functions-of-cationic-peptides-ACP-anticancer-peptides-AMP-antimicrobial_fig1_340674862>, 7. kolovoza 2023.

[7] Sara Wilcox: Cationic peptides: A new hope, s Interneta, <https://www.scq.ubc.ca/cationic-peptides-a-new-hope/>, 8. kolovoza 2023.

[8] Protein Esterification: Design, Antibacterial and Safety Assessment - Scientific Figure on ResearchGate. Available from: <https://www.researchgate.net/figure/Fig-1-Model-for-the-mechanism-of-action-of-cationic-antimicrobial-peptides_fig1_282186504>, 7. kolovoza 2023.

----------

[1] Kuan Y. Chang, Je-Ruei Yang, s Interneta, <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3734225/>, 25. lipnja 2020.

[2] Liana Costa Pereira Vilas Boas, Marcelo Lattarulo Campos, Rhayfa Lorrayne Araujo Berlanda, Natan de Carvalho Neves, Octávio Luiz Franco, s Interneta, <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7079787/>, 26. lipnja 2020.

[3] Wikipedia, s Interneta, <https://en.wikipedia.org/wiki/Peptide>, 25. lipnja 2020.

[4] World of Peptides, s Interneta, [world-of-peptides.com](http://www.world-of-peptides.com/what-is-the-difference-between-a-peptide-and-a-protein/), 5. srpnja 2020.

[5] Courtney Simons, s Interneta, <https://cwsimons.com/structure-of-amino-acids-and-proteins/>, 5. srpnja 2020.

[6] [Ali Adem Bahar](https://www.ncbi.nlm.nih.gov/pubmed/?term=Bahar%20AA%5BAuthor%5D&cauthor=true&cauthor_uid=24287494),[Dacheng Ren](https://www.ncbi.nlm.nih.gov/pubmed/?term=Ren%20D%5BAuthor%5D&cauthor=true&cauthor_uid=24287494), s Interneta, <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3873676/>, 26. lipnja 2020.

[7] Wikipedia, s Interneta, <https://en.wikipedia.org/wiki/Protein_secondary_structure>, 27. lipnja 2020.

[8] ThoughtCo., s Interneta, <https://www.thoughtco.com/protein-structure-373563>, 5. srpnja 2020.

[9] Golifescience, s Interneta, <https://www.golifescience.com/secondary-structure-of-proteins/>, 27. lipnja 2020

[10] C.Mirabello, G.Pollastri, s Interneta, <http://distillf.ucd.ie/porterpaleale/>, 29. lipnja 2020.

[11] Wikipedia, s Interneta, <https://en.wikipedia.org/wiki/Relative_accessible_surface_area>, 6. srpnja 2020.

[12] Claus AF Andersen & Burkhard Rost, s Interneta, <https://rostlab.org/papers/2003_rev_assign/paper_slow.html>, 6. srpnja 2020.

[13] Guy Yachdav, Peter Hoenigschmid, Laszlo Kajan, s Interneta, <https://github.com/Rostlab/reprof>, 29. lipnja 2020.

[14] RPBS Web portal, s Interneta, <https://mobyle.rpbs.univ-paris-diderot.fr/cgi-bin/portal.py#forms::psipred>, 29. lipnja 2020.

[15] J Yang, R Yan, A Roy, D Xu, J Poisson, Y Zhang. The I-TASSER Suite: Protein structure and function prediction. Nature Methods, 12: 7-8 (2015),

A Roy, A Kucukural, Y Zhang. I-TASSER: a unified platform for automated protein structure and function prediction. Nature Protocols, 5: 725-738 (2010),

Y Zhang. I-TASSER server for protein 3D structure prediction. BMC Bioinformatics, vol 9, 40 (2008), s Interneta, <https://zhanglab.ccmb.med.umich.edu/I-TASSER/>, 29. lipnja 2020.

[16] Niklas Donges, s Interneta, <https://builtin.com/data-science/random-forest-algorithm>, 30. lipnja 2020.

[17] Wikipedia, s Interneta, <https://en.wikipedia.org/wiki/Decision_tree>, 5. srpnja 2020.

[18] Tutorialspoint, s Interneta, <https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_classification_algorithms_random_forest.htm>, 30. lipnja 2020.

[19] Tony Yiu, s Interneta, <https://towardsdatascience.com/understanding-random-forest-58381e0602d2>, 6. srpnja 2020.

[20] Abilash R., s Interneta, <https://medium.com/@ar.ingenious/applying-random-forest-classification-machine-learning-algorithm-from-scratch-with-real-24ff198a1c57>, 5. srpnja 2020.

[21] Wikipedia, s Interneta, <https://en.wikipedia.org/wiki/Weka_(machine_learning)>, 30. lipnja 2020.

[22] Eibe Frank, Mark A. Hall, and Ian H. Witten (2016). The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann, Fourth Edition, 2016.

[23] Thakur N, Qureshi A, Kumar M (2012) [AVPpred: collection and prediction of highly effective antiviral peptides.](http://nar.oxfordjournals.org/content/40/W1/W199.long) Nucleic Acids Res. 2012 Jul;40(Web Server issue):W199-204. doi: 10.1093/nar/gks450.  
PMID: [22638580](http://www.ncbi.nlm.nih.gov/pubmed/22638580), s Interneta, <http://crdd.osdd.net/servers/avppred/>, 30. lipnja 2020.

[24] Modlamp dokumentacija, s Interneta, <https://modlamp.org/modlamp.html#module-modlamp.descriptors>, 6. srpnja 2020.

[25] Gufosowa - Own work, CC BY-SA 4.0, <https://commons.wikimedia.org/wiki/File:K-fold_cross_validation_EN.svg>, 1. srpnja 2020.

# POPIS KRATICA

AVP – antiviralni peptidi

AMP – antimikrobni peptidi

VAE – varijacijski autoenkoder

KL divergencija – Kullback-Leibler divergencija

# Sažetak

Virusi i dalje predstavljaju glavni uzrok bolesti kod ljudi usprkos masovnoj proizvodnji cjepiva i antivirusnih lijekova, a velik potencijal za nove pomake u tom području predstavljaju peptidi s antiviralnom aktivnošću. Kako bi efikasno otkrivali nove takve peptide potrebno je koristiti neku vrstu predviđanja, a pokazano je da je to moguće na osnovi njihovih fizikalno-kemijskih svojstava. Takvo predviđanje, ako se dobro izvede, može biti vrlo učinkovito. Posebno se istaknulo strojno učenje metodom *random forest*, kakvo je implementirano i u ovom radu. Daljnjim trudom i ulaganjem u ovakva istraživanja može se ostvariti velik napredak u medicini i poboljšati kvaliteta ljudskog života.

Ključne riječi: antiviralni peptidi, predviđanje sekundarne strukture, algoritam *random forest*, strojno učenje

# Abstract

Viruses are still the main cause of human diseases despite the mass production of vaccines and antiviral drugs. Antiviral peptides have a great potential to change things for the better in that particular field. For efficient discovery of such new peptides, some kind of prediction should be used and it was shown that this was possible based on their physicochemical properties. Such prediction, if done right, can be very effective. Machine learning using the random forest method was proven especially efficient, which was also implemented in this study. With further effort and investments in such research, a great progress in medicine can be achieved and the quality of human lives improved.

Keywords: antiviral peptides, secondary structure prediction, random forest algorithm, machine learning