Математичка гимназија

Матурски рад

-из предмета Програмирање и програмски језици-

на тему

Примена графовских неуралних мрежа на предвиђање молекуларних својстава

Ученик:
Ања Вујачић, IVa

Ментор:
др Петар Величковић

Београд, 28.5.2021.

Садржај

1 Увод **6**

1.1 Процес откривања лека . . . 6

1.1.1 Истраживање и идентификација мете . . . 6

1.1.2 Претклиничка испитивања . . . 7

1.1.3 Претклиничка *In Vitro* испитивања . . . 7

1.1.4 *In Vivo* испитивања на животињама . . . 7

1.1.5 Клиничка испитивања . . . 7

1.1.6 Комерцијализација . . . 8

2 Графови **10**

2.1 Основни појмови и представљање графова . . . 10

2.1.1 Рачунарска репрезентација графова . . . 11

3 Неуралне мреже **13**

3.1 Кратак увод о машинском учењу . . . 13

3.2 Неурон . . . 14

3.3 Потпуно повезане неуралне мреже . . . 16

3.4 Тренирање неуралне мреже . . . 18

3.4.1 Иницијализација параметара . . . 18

3.4.2 Оптимизација . . . 18

3.5 Конволуцијске неуралне мреже . . . 19

3.5.1 Конволуцијски слојеви . . . 20

3.5.2 Агрегациони слојеви . . . 21

3.6 Графовске неуралне мреже . . . 22

3.6.1 Графовске конволуцијске неуралне мреже . . . 23

3.6.2 Графовске неуралне мреже са механизмом пажње . . . 24

3.6.3 Графовске неуралне мреже са преношењем порука . . . 25

4 Обрада података и учење машинског модела **27**
4.1 Опис библиотека 27
4.2 Скуп података и опис модела и резултата 28

5 Закључак **29**

Литература **29**

Глава 1

Увод

1.1 Процес откривања лека

Људска врста, иако није ни најснажнија ни најиздржљивија у поређењу са осталима, успела је да преживи веома дуго. Откриће антибиотика је умањило смртност становништва од болести као што су сифилис, сепса и многе друге а коришћење пеницилина у лечењу постало је свакодневница. Јасно је да је проналазак лекова од кључне важности за побољшање квалитета живота као и његово продужавање. Међутим, откривање, регистрација и доступност лека који је ефикасан а истовремено безбедан за пацијента, представља дуг и комплексан процес. Неопходно је да свака супстанца која претендује да постане лек прође опсежна претклиничка и клиничка испитивања, у којима се испитује сигурност, ефикасност и квалитет лека.

1.1.1 Истраживање и идентификација мете

Дужина: 3+ година

Први корак у развоју лека је свакако темељно истраживање које се обично своди на проналажење мете тј. одређеног макромолекула у телу (најчешће неки протеин или нуклеинска киселина) на који болест делује. При овом процесу истраживачи настоје да буду што сигурнији да је мета на неки начин укључена у болест или стање, и једном када је идентификују она пролази кроз процес валидације .

1.1.2 Претклиничка испитивања

Дужина: 1-2+ година

Када је мета пронађена и потврђена, следи потрага за супстанцом која би могла утицати на њу. Овај процес ће подразумевати лабораторијско испитивање огромног броја једињења, често око 10.000 или више, а вероватноћа да се пронађе савршени кандидат који ће
одговарати је веома мала. Као резултат испитивања идентификоваће се једињења која показују неку активност према мети, а на медицинским хемичарима остаје да одговарајућа једињења прилагоде и побољшају. Циљ овог корака јесте да се значајно смањи број супстанци које потенцијално могу бити лек.

1.1.3 Претклиничка *In Vitro* испитивања

Дужина: 1-2+ година

Овај корак је први сусрет могућих лекова са ћелијом и обично ће само око 250 једињења од оних која су првобитно разматрана доћи до овде. *In Vitro* укључује тестове на ћелијама или молекулима изван њиховог уобичајеног биолошког окружења (односно у епрувети или петријевој шољи). Ова метода проверава како ће супстанца утицати на ћелије и узрочника болести. Нажалост, не пружа информације о начинима на које може утицати на организме.

1.1.4 *In Vivo* испитивања на животињама

Дужина: 1-2+ година

Супстанце које се покажу као обећавајуће у *In Vitro* испитивању ступају у четврту фазу истраживања које се обично раде на мишевима и сисарима, за чији рад је потребно много новца. Врше се токсиколошка истраживања, посматрају се ефекти и ефикасност супстанце. Уколико се супстанца покаже као неодговарајућа, штете по живу средину али и по буџет научника су огромне. Иако се чине напори да се смањи количина оваквих испитивања, она се и даље користи јер дају важне информације о ефектима лекова које *in Vitro* не може.

1.1.5 Клиничка испитивања

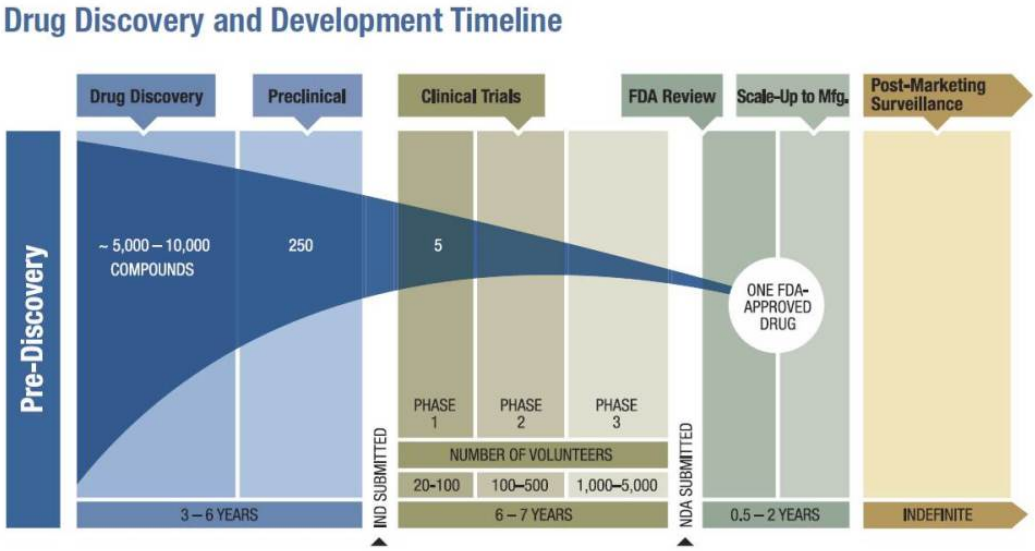
Дужина: 6-7 година

Након што успешно прођу све претходне кораке, супстанце су спремне за проверу безбедности и ефикасности на људском организму. Овај корак се састоји из три

подкорака од којих сваки тежи да испита понашање лека на здравим добровољцима и пацијентима.

1.1.6 Комерцијализација

Ако лек успешно прође сва клиничка испитивања и буде одобрен од стране светске здравствене организације, произвођач може добити дозволу за производњу и стављање лека у промет. Добро је познато да су почетне цене новодобијених лекова изразито високе како би се новац утрошен у истраживања и проналазак терапије вратио.



Слика 1.1: Илустрација процеса откривања лекова

Очигледно је да је развој успешног лека дуг и мукотрпан процес који се чешће сусреће са неуспехом него са успехом. Медицински хемичар Дерек Лове има занимљив чланак у којем разматра вероватноћу да медицински хемичар развије лек спреман за тржиште, и довољно је рећи да је то прилично мала цифра. Отуда се намеће питање да ли је могуће наћи иновативни приступ и тако убрзати процес. Развој нових лекова у великој мери се ослања на познавање различитих молекуларних својстава потенцијалних кандидата за лекове.

Модели дубоког учења омогућавају хемичарима да брзо филтрирају огромне библиотеке података о молекулима, и идентификују потенцијалне кандидате. Последњих година, машинско учење је почело да игра све значајнију улогу у предвиђању молекуларних својстава, јер пружа брже и јефтиније решење.

У математичкој хемији, молекули су приказани преко структурне формуле, где врхови, одговарају атомима а ивице хемијским везама. Интуитивно би се могло замислити да атоме у молекулу третирамо као чворове графа, а везе као његове ивице. Специфичност репрезентације молекула и његове зависности од појединачних атома и њихових међусобних веза отоворили су пут ка графовским неуралним мрежама (енг. *graph neural networks*) које су коришћене у овом раду.

Глава 2

Графови

У овом одељку ћемо увести неке основне елементе теорије графова који ће нам бити од великог значаја за даљи рад и разумевање.

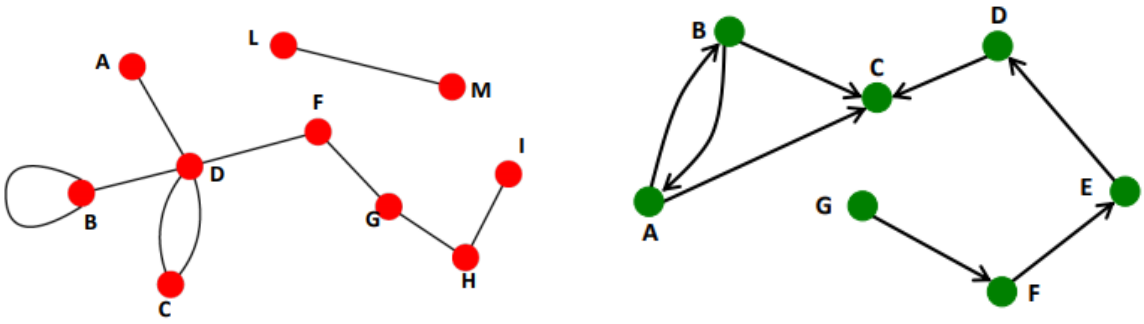
2.1 Основни појмови и представљање графова

- Граф $G = (V, E)$ је дефинисан скупом чворова V и скупом грана E између њих. Грану која иде од чвора $u \in V$ до чвора $v \in V$ означавамо као $(u, v) \in E$. Граф може бити усмерен и неусмерен.

- Код неусмереног графа гране које повезују чворове су двосмерне тј. (грана $(v \to u)$ је исто што и грана $(u \to v)$.

- Супотно, граф чије гране имају правац назива се усмерен граф и код њега су гране $(v \to u)$ и $(u \to v)$ различите.

- Степен чвора, у ознаци $d(v)$ је број грана суседних чвору v (односно број грана које тај чвор повезују са неким другим). Код усмерених графова се дефинишу улазни и излазни степен као број грана које улазе у дати чвор, односно број грана које излазе из датог чвора

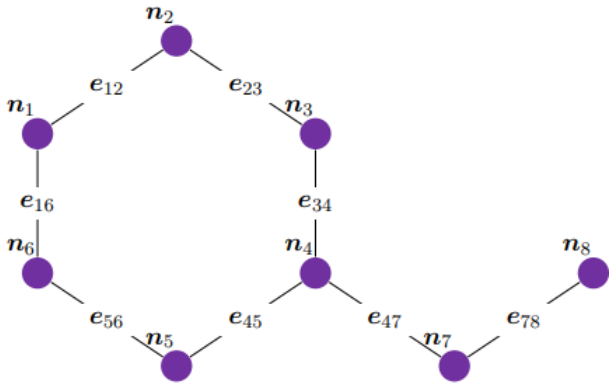


(a) Неусмерен граф (b) Усмерен граф

Слика 2.1

Сада је интуитивно јасно како би изглдао приказ молекула као неусмереног графа. Атоми су представљени као чворови, док су хемијске везе између атома представљене као гране. Карактеристике атома и њихови веза можемо једноставно представити преко два скупа вектора, $\{ e_{ij} \mid (v, w) \in E\}$ и $\{n_v | v \in V\}$.

Треба обратити пажњу да редослед чворова у графу V није једнозначно одређен, стога ни једна операција која се над њим извршава не сме да зависи од редоследа којим се чворови обрађују. За операције са овим својством, које нам дају идентични резултат независно од редоследа, кажемо да су инварјантне у односу на редослед. Такве су на пример сабирање, множење, просек...



Слика 2.2: Приказ структуре којом се представља молекул: неусмерени граф, вектор особина за сваки чвор, и вектор особина за сваку грану.

2.1.1 Рачунарска репрезентација графова

Графови се обично представљају графички, цртањем тачака као чворова и линија између чворова ако су они повезани. С обзиром на велики број података и проблема који се

могу јавити при креирању графова, поставља се питање њихове рачунарске репрезентације. До сада, вероватно најпопуларнија репрезентација графа је матрица повезаности.

Дефиниција 1 *Матрица повезаности A коначног графа G са n чворова је $n \times n$ (квадратна) матрица где, A_{ij} узима вредност 1 уколико између i и j постоји грана, док у супротном узима вредност 0.*

Лако је приметити да ће матрица повезаности неусмереног графа бити симетрична у односу на главну дијагоналу, стога неће бити потребно да чувамо све њене елементе већ само њену горњу-троугаону половину. Међутим, у пракси се испоставило да овај начин чувања није увек најпрактичнији. Није тешко увидети да ће матрица повезаности графа чији је број грана мали садржати много већи број нула него јединица, тј. матрица ће бити *ретка* (енг. *sparse matrix*). Рад са оваквим матрицама може захтевати јако пуно времена и меморијског простора због чега се обично прибегава некој од следећих метода:

- Креирање листе листи (енг. *Using List of Lists*) - Једна листа је коришћена за представљање редова матрице, где сваки ред садржи листу са индексом колоне и вредности 0 или 1.
- Креирањем дневника кључева (енг. *Using Dictionary of keys*) - Индекс колоне и реда се мапирају тако да показују на одговарајућу вредност 0 или 1.

Глава 3

Неуралне мреже

Ово поглавље садржи кратак увод у машинско учење, након чега ћемо представити три типа неуралних мрежа и детаљније их описати.

3.1 Кратак увод о машинском учењу

Машинско учење је подобласт вештачке интелигенције која се бави конструисањем алгоритама и рачунарских система који су способни да се адаптирају на нове ситуације и уче на основу претходног искуства. Ове методе су обично класификоване у неколико класа:

- Надгледано учење (енг. Supervised learning);
- Ненадгледано учење (енг. Unsupervised learning);
- Учење са појачањем (енг. Reinforcement learning).

Међутим, ове класе се међусобно не искључују и постоји много веза између њих.

Надгледано учење је вероватно најзаступљенији вид машсинског учења чији је примарни циљ да уочи правилности и конструише нову функцију $f : X \to Y$ којом ће решавати нове улазе, користећи унапред дати скуп података D са паровима улаза и излаза

$$D = \{(x_i, y_i), i = 1 \dots n\},$$

где $x_i \in X$ и $y_i \in Y$. Овај модел се обично примењује при решавању проблема класификације и регресије.

У случају ненадгледаног учења наш скуп података D састоји се само од сирових улаза, док нам њихови излази остају непознати, тј.

$$D = \{x_i, i = 1 \dots n\}.$$

Циљ метода ненадгледаног учења јесте да алгоритам сам уочи везе и заједничке карактеристике на основу структуре. Кластеровање, откривање аномалија и технике смањења димензионалности су неке од уобичајених метода ненадгледаног учења.

Коначно, учење са појачањем има за циљ да научи оптималну политику за агента у окружење, користећи принцип доделе награде. Иако постоје многе формулације, основна и најприроднија би била

$$D = \{(s_i, a_i, r_i), i = 1 \dots n\},$$

где је $s_i \in S$ стање система окружења, $a_i \in A$ акција коју је агент предузео, и $r_i \in R$ награда која се даје агенту за предузимање акције a_i у стању s_i. На овај начин мрежа се усавршава и постаје све боља у решавању датог проблема. Теоријски, овај облик учења би се могао применити на проблем класификације међутим, поставља се питање ефикасности. Испоставља се да је време прикупљања података, док модел не постане довољно поуздан, енормно и да алгоритам споро конвергира према решењу.

Фокус овог рада биће на надгледаном учењу и примени модела дубоког учења за предвиђање својстава молекула. У остатку овог одељка представићемо две категорије алгоритама дубоког учења, конволуцијске и графовске неуралне мреже.

3.2 Неурон

Неурална мрежа је модел машинског учења који се састоји од међусобно повезане мреже рачунских јединица познатих као „неурони“ или „перцептрони“. Неурон прима n улазних вредности и трансформише их у једну излазну вредност као линеарну комбинацију својих улаза ($R_n \to R_1$). Вредност излаза може се представити као:

$$y = (w_1, \dots, w_n) \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \sum_{i=1}^{n} w_i X^i \tag{3.1}$$

где су w_i променљиви параметара који се називају тежинама (енг. *weights*) . Вредности тежина се мењају у току тренинга (о чему ће бити речи касније) како би се постигла тачнија предвиђања.

Наиме, како линеарни модели не могу довољно добро апроксимирати неке функције, неопходно је обезбедити нелинеарност, што се постиже коришћењем функције активације. Једноставно говорећи, резултати неурона са линеарном функцијом активације били би само линеарна комбинација улазних параметара што нам не би дало прецизно предвиђање.
Користећи функцију актиавације, повратна вредност неурона је :

$$y = \sigma\left(b + \sum_{i=1}^{n} w_i X^i\right) = \sigma(b + \vec{w}^T \vec{x}), \tag{3.2}$$

где је σ функција активације а b додатни слободни члан.

Постоји велики број функција активације, али најсавременије мреже најчешће примењују неке од следећих:

- ***Логистичка функција:***

  $$\sigma(x) = \frac{1}{1 + e^{-x}}$$

  Реч *сигмоид* значи облик слова s што описује график ове функције. Логистичка функција позната је још и као функција згњечења, јер пресликава читав скуп реалних бројева $z \in R$ у интервал $[0, 1]$ што даје могућност моделовања Бернулијеве расподеле случајних величина (нпр. за бинарну класификацију). Ова једноставна функција има две корисне особине: (1) може се користити за моделирање условне расподеле вероватноће и (2) има једноставан извод.

- ***Тангенс хиперболичка функција:***

  $$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

  Ова функција, као и логистичка, пресликава скуп реалних вредности $z \in$R у интервал $[-1, 1]$, међутим даје боље резултате када се ради о вишеслојним неуронским мрежама. Углавном се користи код проблема регресије.

- ***ReLU - Исправљена линеарна јединица:***

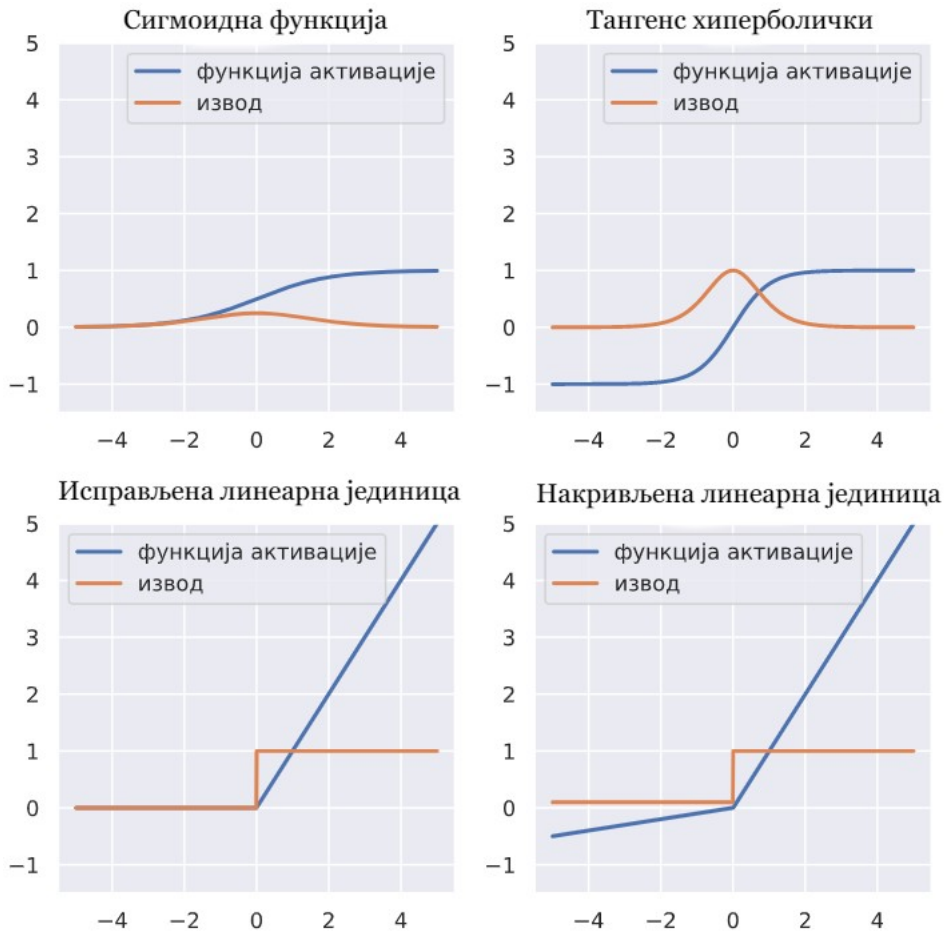
  $$\sigma(x) = max(x, 0)$$

  Тренутно је једна од најпопуларнијих функција активације захваљујући својој рачунарској ефикасности и неким својствима због којих је смањена вероватноћа да се јави нестајући градијент.

- ***LeakyReLU - Накривљена линеарна јединица:***

$$\sigma(x) = \begin{cases} x, & x \geq 0, \\ \alpha x, & x < 0 \end{cases}$$

Уместо константе за вредности мање од нула користи се линеарна функција константног нагиба. Коефицијент α је обично врло мала вредност (реда величине 0.01) што решава проблем нестајућег градијента.

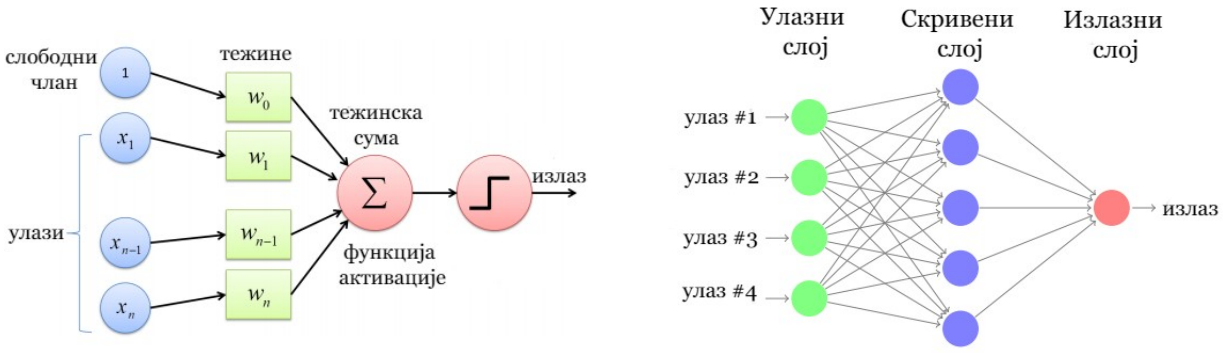


Слика 3.1:
Графички приказ неких функција активације и њихових извода

3.3 Потпуно повезане неуралне мреже

Оно што чини праве неуралне мреже знатно сложенијим јесте комбиновање већег броја неурона заједно. Потпуно повезана неуралне мрежа састоји се од већег броја неуро-

на који су организовани по слојевевима. Први слој се назива *улазним* слојем и он се састоји од улазних вектора атрибута. Последњи слој је *излазни* слој који представља предвиђање које мрежа даје за улазне атрибуте. Поред улазног и излазног слоја може постојати један или више *скривених* слојева који извршавају додатне нелинеарне трансформације над улазним подацима. Улазни подаци се обрађују и прослеђују из једног слоја у други тј. излаз претходног слоја се прослеђују као улаз у наредни слој. Иако је број чворова у улазном и излазном слоју одређен улазно/излазним типовима података, скривени слој може имати произвољан број неурона. Приказ примера потпуно повезане неуралне мреже и неурона дат је на слици 3.2.



Слика 3.2: **Лево:** Модел неурона, са улазним вектором , тежинама $\vec{w}$, слободним чланом b и функцијом активације σ. **Десно**: Приказ потпуно повезане неуралне мреже са једним скривеним слојем од 5 неурона.

Наиме, ако означимо улаз као $\vec{x}$, онда j-ти неурон датог слоја израчунава следеће:

$$y_j = \sigma(b_j + \vec{w_j}^T \vec{x}), \tag{3.3}$$

где су $\vec{w_j}^T$ и b_j тежине и слободни члан неурона респективно. Ради једноставности, једначина (3.3) се често записује у матричном облику:

$$\vec{y} = \sigma(W\vec{x} + \vec{b}), \tag{3.4}$$

где су W и $\vec{b}$ матрица тежина и вектор слободних чланова респективно.

Дубина мреже може се повећати спајањем више оваквих слојева, где је сваки слој одређен сопственим параметрима, на пример:

$$\vec{y} = \sigma_2(W_2\sigma_1(W\vec{x} + \vec{b}) + \vec{b_2}), \tag{3.5}$$

Са повећањем броја слојева и неурона у њима повећава се и сложеност функције коју модел може да предвиди.

3.4 Тренирање неуралне мреже

Сада када смо се упознали са архитектуром неуралне мреже остало је да пронађемо оптималне параметре који карактеришу модел. Почетне вредности параметара иницијализујемо по сопственом избору а у току тренинга их прилагођавамо моделу како би извршио прецизнију и бољу апроксимацију.

3.4.1 Иницијализација параметара

Оптимизације неуронских мрежа се обично врши постепено, у више мањих корака, ажурирањем почетних параметара. Правилан одабир почетних параметара је први и значајан корак јер од њега зависе перформансе алгоритма тј. колико брзо ће конвергирати и ка ком локалном минимуму. На одабир могу утицати разни фактори па је зато битно добро размислити како би се избегли проблеми нестајућег и експлодирајућег градијента.

3.4.2 Оптимизација

Када се тежине иницијализују на одговарајуће вредности модел је спреман да крене да учи како да што тачније апроксимира зависност улаза и излаза. То се обично постиже коришћењем функције губитка која мери одступање предвиђеног излаза од његове праве вредности и тако нам говори колико су постављени параметари погодни за задатак.

Избор ове функције најчешће зависи од типа проблема који се решава (на пример, уколико је познато да је излаз из мреже слика, користиће се другачија функција губитка у односу на случај када је излаз из мреже јединични вектор при процесу класификације). Дакле, специфичност проблема који се решава диктира избор функције губитка. Нека је дато N парова улаза и излаза у векторском облику $\{(\vec{x_k},\, \vec{y_k})\}$, $k \in \{1 \dots N\}$. На основу датих атрибута $\vec{x_k}$ модел ће извршити предвиђање и дати $\hat{y_k}$ као излаз из последњег слоја.

За једноставне регресионе проблеме најчешће се користи функција средњеквадратне грешке:

$$\mathcal{L} = \frac{1}{N} \sum_{k=1}^{N} \quad \|\vec{y_k} - \hat{y_k}\|^2 ,$$

док је у проблемима класификације уобичајено коришћење функције унакрсне ентропије:

$$\mathcal{L} = \frac{1}{N} \sum_{k=1}^{N} \sum_{j=1}^{M} y_{kj} \log(\hat{y_{kj}}),$$

где M означава број класа.

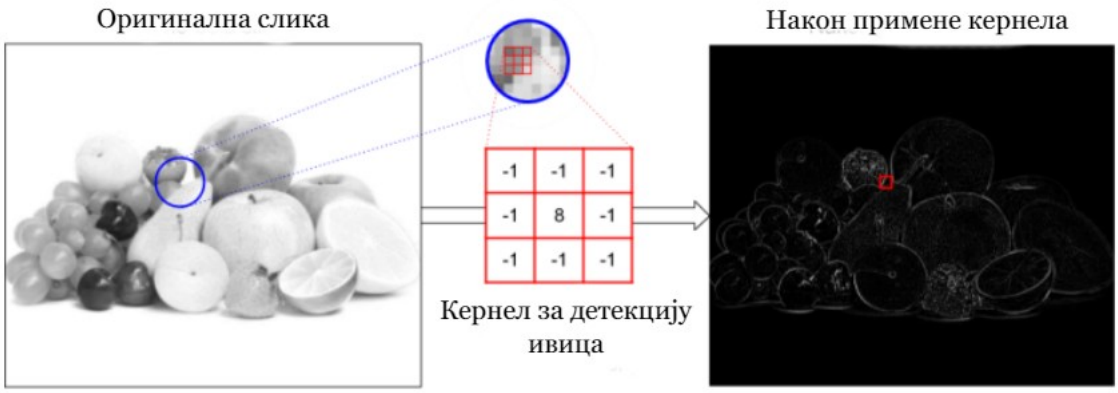
Поред ових функција губитка користе се и многе друге, од којих су најчешће:функција средње апсолутне грешке, преломни губитак, Хуберова функција губитка... У принципу, функцијом губитка се може прогласити било која математичка функција, али је неопходно да њена оптимизација утиче повољно на подешавање параметара мреже.

3.5 Конволуцијске неуралне мреже

Након што смо се упознали са вишеслојним неуралним мрежама, у овом одељку ћемо описати конволуцијске неуралне мреже које су се показале као нарочито успешне при обради слика (2-димензионална мрежа пиксела) и видеа (3-димензионална мрежа пиксела).

Порекло конволуцијских неуронских мрежа сеже у 1970-те, али рад који је успоставио савремени предмет конволуцијских мрежа био је рад из 1998. године, *"Gradient-based learning applied to document recognition"*. Један од аутора је тада дао занимљиву примедбу на терминологију конволуцијских мрежа рекавши да је њихова биолошка (неуронска) инспирација у моделима врло слаба, иако користе многе исте идеје као и неуронске мреже које смо до сада проучавали.

Архитектура потпуно повезане неуралне мреже подразумева улазе у облику једнодимензионалног вектора. Представљањем слике у том облику, редови пиксела би се надовезали у вектор чиме би се изгубиле информације о просторној структури слике. Како би се то избегло, идеја је да оригиналну слику смањимо тако да буде довољно мала да вишеслојна неуралн мрежа може да је обради. Како потенцијално не бисмо избацили неке корисне информације, потребно је да пазимо на просторну структуру слике док екстрахујемо информације, а логични кандидат за овакав задатак јесте управо оператор конволуције.



Слика 3.3: Пример препознавања ивица слике помоћу оператора конволуције.

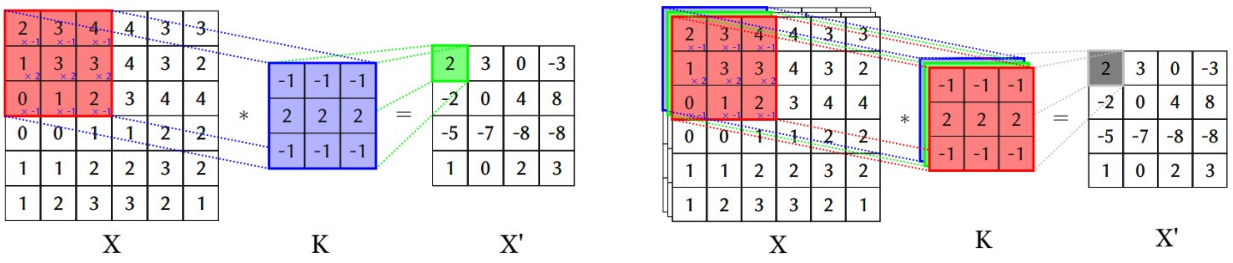
3.5.1 Конволуцијски слојеви

У терминологији конволуцијских неуралних мрежа, конволуција је оператор који на основу две матрице, где је прва наша слика $X \in R^{h \times w}$, а друга такозвани кернел $K \in R^{n \times m}$, формира трећу матрицу X', при чему важи $n \leq h$ и $m \leq w$. Елементи матрице X' добијају се постављањем кернела на све могуће начине преко полазне матрице и рачунањем сума производа:

$$X'_{kl} = \sum_{i=1}^{n} \sum_{j=1}^{m} K_{ij} \cdot X_{k+i-1,l+j-1}$$

Савремене архитектуре мрежа обично узимају $n = m = 3$. Димензије овако добијене матрице су $(h - n + 1) \times (w - m + 1)$. У пракси је уобичајено додавање нула на ивице матрице (енг. *padding*) како би се осигурало да величина излаза одговара величини улаза.

Улазна слика је углавном представљена у три канала (као тродимензионални низ вредности за сваку од нијанси црвене, зелене и плаве). Тада се за сваки канал дефинише посебан кернел чиме настаје тензоркси кернел (енг. *kernel tensor*) који садржи три кернела димензија $m \times n$.



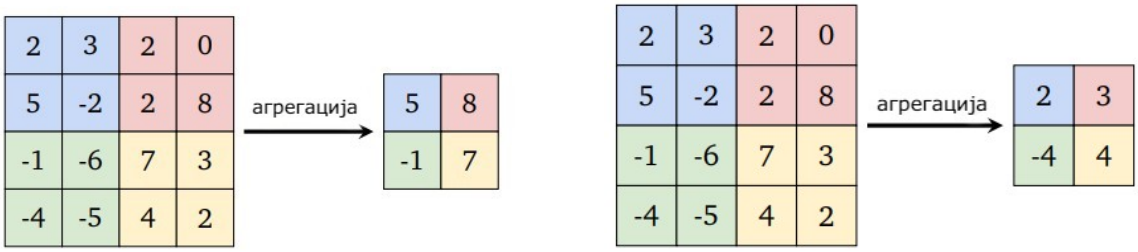
Слика 3.4: **Лево:** Приказ тензорског кернела примењеног на једноканалну слику. σ. **Десно**: Приказ тензорског кернела примењеног на троканалну слику.

Уопштено, ако улаз има више канала, димензије кернела се проширују тако да одговарају улазу. Сваки излазни канал захтева засебан тензорски кернел. Поред операција конволуције, за добијање коначног излаза се на канале може применити и функција активације. Тако се добијају такозване мапе карактеристика (енг. *feature maps*) над којима се даље примењује агрегација која смањује димензије улазне слике.

3.5.2 Агрегациони слојеви

Након што се конволуција над сликом изврши довољан број пута, може се користити агрегација. Поред тога што драстично смањује димензију улаза овај слој утиче и на смањење броја параметара, што олакшава рачунање и повољно утиче на проблем прекомерног прилагођавања.

У зависности од типа агрегације, бира се пиксел са одређеном вредношћу у одређеном региону који је задан величином филтера. Два најчешћа примера агрегације су агрегација максимумом и агрегација просечном вредношћу. У наставку дајемо визуелизацију споменутих типова агрегације.



(a) Агрегација узимањем максималног елемента са филтером величине 2×2

(b) Агрегација узимањем просечне вредности са филтером величине 2×2

Слика 3.5

Класична конволуцијска неуронска мрежа састоји се од низа конволуцијских и агрегационих слојева који извлаче нека интересантна својства слике и смањују је како би

је проследили вишеслојној неуралној мрежи која ће извршити финалну класификацију.

3.6 Графовске неуралне мреже

Стандардни алгоритми дубоког учења садрже мноштво метода за учење и предвиђање резултата, али не постоји подразумевана метода која ради на произвољним структурама. Потребни су нам модели са експлицитним приказом објеката и релација и алгоритми учења који проналазе правила по којима су повезани. Важно је узети у обзир да ентитети у свету немају устаљен редослед, већ је он одређен својствима њихових односа. На пример, односи између величина предмета у скупу могли би се потенцијално користити за њихово сортирање, као што би могле и њихове масе, године, цене, или токсичност (у случају молекула). Инваријантност уређења, осим у случају ако су објекти повезани, је својство које би требало да се одрази методама дубоког учења.

Графови представљају природни приказ система описаног објектима чији редослед није дефинисан или се сматра небитним. Ради лакшег разумевања размотримо задатак предвиђања центра масе Сунчевог соларног система који се састоји из n планета чији су атрибути (маса, положај, брзина...) $(x_1, x_2 \dots x_n)$ редом. У овом задатку редослед којим посматрамо планете није значајан, јер се стање може описати прикупљањем атрибута и рачунањем њихове просечне вредности. Међутим ако бисмо за овај задатак користили вишеслојну неуралну мрежу добили бисмо различита предвиђања за исте улазе чији је једино редослед аргумената другачији. Како постоји $n!$ могућих пермутација улаза, у најгорем случају неуралне мрежа би сваку од пермутација могла сматрати суштински другачијом, а самим тим и захтевати експоненцијални број примера улазно/излазног тренинга за учење функције предвиђања. Природан начин да решимо ову комбинаторну експлозију је да омогућимо да предвиђање зависи само од симетричних функција улазних атрибута. То би се могло извршити рачунањем заједничких карактеристика објеката $(f(x_1), f(x_2) \dots f(x_n))$ које се затим агрегирају узимајући њихову средњу вредност. Такав приступ је детаљније описан у одељку 3.6.3

Наравно, поред инваријантности на пермутације постоји још битних особина на које треба обратити пажњу. На пример, на сваки објекат могу утицати интеракције са осталим објектима у скупу. У нашем сценарију са планетама, задатак је предвидети положај сваке планете након временског интервала Δт. У овом случају, коришћење агрегираних, просечених информација није довољно, јер кретање сваке планете зависи од сила које друге планете врше на њу. Уместо тога, могли бисмо израчунати стање сваког објекта као $x_i' = f(x_i \sum_j g(x_i, x_j))$ где g рачуна силу индуковану j-том планетом на i-ој планети,

а f будуће стање i-те планете.

Горњи примери Сунчевог система илуструју две релационе структуре: једну у којој не постоје односи између објеката и другу у којој су сви објекти међусобно повезани. Многи системи из стварног света имају релациону структуру негде између ове две крајности, односно неки објекти су међусобно повезани а неки не. У пракси то значи да ћемо у обзир узимати интеракција између само неких објеката. Структура где i-ти објекат комуницира само са подскупом осталих објеката, описаним његовим суседством одговара управо графу.

Графови су, генерално, приказ који подржава произвољну релациону структуру, а прорачуни преко графа дају боље предвиђање од оног које могу пружити конволуциони слојеви.

Претпоставља се да сваки граф припада некој класи $y \in \mathcal{Y}$ из скупа унапред задатих класа $\mathcal{Y}$. Циљ задатка класификације је да изведе функцију која ће тачно предвидети класу којој припада посматрани граф. Најраније идеје за решавање овог проблема базирале су се на процени сличности графова и уочавању специфичних карактеристика на основу којих би га сврстали у одређену класу. Једна од најчешћих метода се заснивала на коришћењу такозваних графовских кернел, а један од примера био би графовски кернел случајне шетње (енг. *Random Walk Graph Kernel*).

Формална дефиниција кернела случајне шетње над графом је:

$$k_{graph}(G, G') = \sum_{walk \in G} \sum_{walk \in G'} k_{walk}(walk, walk'), \tag{3.6}$$

где су G и G' графови које поредимо а $walk \in G$, $walk' \in G$ шетње исте дужине. Посматра се сваки пар шетњи и на основу њихових чворова и ивица се одређује број који оцењује колико су шетње сличне. На крају, збир свих оцена сличности представља меру сличности два графа. Оваква метода није погодна за рад са великим графовима јер ће чување репрезентације у том случају заузети пуно меморије. Данашњи алгоритми дубоког учења нуде ефикаснија решења за класификационе проблеме.

3.6.1 Графовске конволуцијске неуралне мреже

Идеја графовских конволуцијских мрежа, скраћено *GCN*, јавила се из жеље да се прослављени конволутивни приступ примени на графовску структуру података. Аналогно коволуцијском, графовски конволутивни слој се добија комбиновањем атрибута суседних чворова. Репрезентација чвора може се записати на следећи начин:

$$h_v^{(k)} = g^{(k)}(h_v^{(k-1)}, a_v^{(k)})$$

где је a резултат функције агрегације примењене на суседе посматраног чвора i. Исто ово може се записати у матричном облику преко матрице суседства и матрице тежина:

$$H^{(k)} = \sigma(AH^{(k-1)}W^{(k-1)})$$

где је $W^{(k-1)}$ матрица тежина иста за све чворове графа, а σ функција активације. Како се не би изгубила главна информација о стању чворова, мора се додати веза чвора са самим собом тако да матрица суседства има облик $\tilde{A} = A + I_m$. Друго, пошто се услед множења матрицом у суми појављује онолико сабирака колико има чворова у суседству, врши се *нормализација* како би ред величине остао исти.

Коначно, модел графовске конволуцијске мреже се може записати као:

$$h_{(v_j)}^{(k)} = \sigma(\sum_j \frac{1}{c_{ij}} h_{v_j}^{(k-1)} W^{(K-1)}),$$

где је j индекс суседа чвора v_i а c_{ij} константа нормализације за ивицу (v_i, v_j).

Слично као код графовских конволуцијских мрежа, да би модел извршио класификацију графа неопходно је коришћење такозваних *pooling* оператора (најчешће збир или просек) који ће објединити карактеристике чворова и извршити предвиђање.

3.6.2 Графовске неуралне мреже са механизмом пажње

Графовске неуралне мреже са механизмом пажње, скраћено *GAT*, унапређују основну верзију графовских конволуцијских мрежа. Новина коју доносе јесте да при доношењу одлуке не третирамо све чворове подједнако битним, односно, коефицијенати пажње који су додељени сваком чвору одређују у којој мери ће он утицати на доношење одлуке. Ивице такође могу бити одређене коефицијентима.

GAT слој

Улаз у *GAT* слој чини скуп карактеристика чворова, $H = (\vec{h_1}, \vec{h_2}, \vec{h_3} \dots \vec{h_N}), h_i \in R^F$, где је N број чворова и F број карактеристика сваког чвора. На основу улаза изводи се нови скуп атрибута (потенцијално са различитим бројем елемената) карактеристика чвора $H' = (\vec{h'_1}, \vec{h'_2}, \vec{h'_3} \dots \vec{h'_N}), h'_i \in R^{F'}$ као излаз.

Да би се постигли ефекти дубоког учења неопходно је извршити барем једну линеарну трансформацију. У те сврхе, уводи се матрица тежина $W \in R^{F \times F'}$ заједничка за све чворове. Затим се примењује унутрашњи механизам пажње дефинисан као $a : R^{F \times F'} \to R$

који рачуна коефицијт пажње

$$e_{ij} = a(W\vec{h_i}, W\vec{h_j})$$

који одређује значај карактеристика чвора j за чвор i. Како би коефицијенти различитих чворова могли лако да се пореде неопходно је да им ред величине буде исти. То се постиже њиховом нормализацијом помоћу *softmax* функције:

$$\alpha_{ij} = \frac{\exp e_{ij}}{\sum_{k \in N_i} \exp e_{ik}}$$

Механизам пажње a који користимо у овом раду функција представља потпуно повезану неуралну мрежу, одређенз вектором тежина и функцијом активације.

$$\alpha_{ij} = \frac{\exp\left(LeakyReLU(\vec{a}^T[\mathbf{W}\vec{h_i} \parallel \mathbf{W}\vec{h_j}])\right)}{\sum_{k \in N_i} \exp\left(LeakyReLU(\vec{a}^T[\mathbf{W}\vec{h_i}] \parallel [\mathbf{W}\vec{h_k}]\right)}$$

Овако нормализовани коефицијенти користе се даље за рачунање атрибута чворова у наредном слоју.

$$\vec{h_i'} = \sigma(\sum_{j \in N_i} \alpha_{ij} \mathbf{W}\vec{h_j})$$

3.6.3 Графовске неуралне мреже са преношењем порука

Неуралне мреже са механизмом преношења порука, скраћено *MPNN*, представљају архитектуру дубоког учења која је највећу примену пронашла у хемијском и фармацеутском свету.

Рад *MPNN* модела се састоји из два фазе, прослеђивање поруке и очитавање. У првом кораку се кроз неколико итерација, коришћењем информација суседних чворова ажурирају чворови графа како би се изградила неуронска репрезентација молекула, а затим се тај коначни приказ молеукла користи за предвиђање његових својстава. Прецизније, фаза преношења поруке се састоји из m корака, где се при сваком кораку ажурирају скривена стања h_v^t и поруке m_v^tпомоћу функције агрегације M^t која генерише нову поруку и функције U^t која ажурира чворове. Може се записати као:

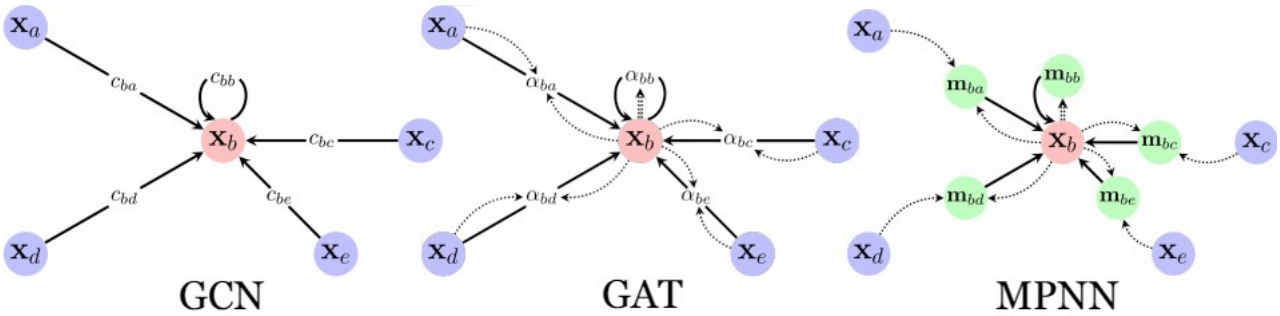
$$\begin{aligned} h_v^{(t+1)} &= U^t\left(h_v^{(t)}, M^{(t)}(h_v^{(t)}, \forall v \in N_{(v)})\right) \\ &= U^{(t)}\left(h_v^{(t)}, m_{N_{(v)}}^{(t)}\right) \end{aligned}$$

$N_{(v)}$ представља скуп суседа чвора v. Битно је истаћи да функција агрегације мора бити инваријантна у односу на редослед чворова у графу како се резултат не би мењао у зависности од редоследа којим обрађујемо чворове. Неке од најчешће коришћених функција су збир, просек и максимум.

Фаза очитавања затим користи функцију очитавања R да би направила предвиђање $\hat{y}$ на основу коначних скривених стања:

$$\hat{y} = R\left(h_v^{(T)}\right)$$

Излаз може бити скалар или вектор, у зависности од тога да ли је *MPNN* дизајнирана да предвиђа једно или више својстава молекула.



Слика 3.6: .

Глава 4

Обрада података и учење машинског модела

Ова глава садржи преглед метода коришћених за одређивање да ли посматрани молекул инхибира репликацију ХИВ-а или не. У ту сврху коришћена је *MPNN* неуралнa мрежа која се састоји из улазног слоја,?? скривена свлоја и излазног слоја. За излазни слој коришћена је *softmax* функција док је за функцију губика узета функција унакрсне ентропије.

4.1 Опис библиотека

За потребе имплементације овог рада коришћена је библиотека *JAX* развијена од стране *Google Research* тима. Ова библиотека дизајнирана је да олакша рад са дубоким неуралним мрежама и алгоритмима машинског учења који су генерално тешки за имплементацију од нуле. *JAX* је саставни део програмског језика *Python*, а сам рад библиотеке заснован је на библиотеци *NumPy*. Неке занимљива проширења којима се *JAX* одликује су :

Диференцијацција: Подржава аутомацко диференцирање произвољних нумеричких функција унапред и уназад, помоћу трансформација функција као што су *grad* или *hessian*, па самим тим олакшава оптимизацију.

Векторизација: У истраживањима машинског учења често је примена једне функције на велику количину података (рецимо при израчунавању грешке кроз серију). Оно што *JAX* нуди ради поједноствљења овог пробелема јесте функција *vamp* која ово ради аутомацки.

4.2 Скуп података и опис модела и резултата

За потребе демонстрације модела коришћеном у овом раду коришћен је *ogbg-molhiv*. Целокупан скуп података је заправо део *Open graph benchmark-a*. Скуп се састоји од 41, 127 графова, где сваки граф представља репрезентацију неког молекула. Чворови графа представљају атоме, док ивице представљају везе између атома. Чворови и везе садрже атрибуте у облику 9-димензионалних вектора који карактеришу атоме и везе (нпр. садрже информацију о атомском броју, хиралности, наелектрисању и многим другим карактеристикама).
Зарад валидације модела потребно је поделити податке на тренинг и тест примере. Тренинг подаци се користе за обучавање модела, а тест подаци се користе у тестирању већ обученог модела. Тренинг алгоритам се одвија у више етапа: иницијализација, тренирање модела, обучен модел.

У овом раду коришћена је *Xavier* иницијализација у којој се сваки параметар w_i од тежина $\vec{w}$ неког неурона насумично бира из нормалне униформне расподеле ограничене са $-\frac{6}{n_{in}+n_{out}}$ и $\frac{6}{n_{in}+n_{out}}$, где је n_{in} број улазних неурона, а n_{out} број излазних неурона. Показало се да ова иницијализација повољно утиче на проблем нестајућег
градијента. Између конволутивних слојева је коришћена активацијска функција $ReLU$, док је као метод оптимизације коришћен *Adam* са стопом учења 0.001. Као резултат, модел враћа вектор са вероватноћама које се пореде са истинитим вредностима помоћу бинарне унакрсне ентропије. Узимањем највеће вероватноће у низу одређује се класа којој посматрани молекул припада, односно да ли је инхибитор или не. Тако је извршена финална класификација.

Имплементирани алгоритам је добро детектовао 61,31%, што значи да модел није нарочито прецизан. За већу прецизност неопходно је променити вредности параметара који ће утицати на крајњи исход.

Глава 5

Закључак

Главни циљ овог матурског рада био је да прикаже теоријску основу потребну за разумевање принципа рада и имплементацију неуронске мреже, као и њихове примене на графовске структуре података. Такође, имплементирана је графовска неурална мрежа која релативно успешно (али не и толико прецизно) предвиђа нека интересантна својства молекула.

Могућности за даљи рад јесу даље тренирање мреже и подешавање коефицијената у циљу постизања веће прецизности при класификацији. Такође интересантан задатак био би конструкција *GAT* модела за решавање поменутог проблема класификације и упоређивање перформанси.

На самом крају бих желела да изразим посебну захвалност

- **др Петру Величковићу** - мом ментору, докторанду Универзитета у Кембрицу, који ми је значајно помогао, пре свега несебичним издвајањем времена и свеопштом присутности у овом раду. Поред тога дао ми је низ сугестија, предлог теме, упутио на одговарајућу литературу и био максимално на располагању за све што ми није било јасно у процесу учења.
- **Снежани Јелић** - мојој професорки информатике која је потпомогла у развијању мог информатичког, али и свеукупног знања и додатно ме мотивисала на даљи рад.
- Свима осталима - који нису наведени а допринели су у унапређењу мог информатичког знања.

Литература

[1] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio: *GRAPH ATTENTION NETWORKS* Доступно на: https://arxiv.org/pdf/1710.10903.pdf [Прегледано 28.5.2021].

[2] Petar Veličković: *Theoretical Foundations of Graph Neural Networks* Доступно на: https://petar-v.com/talks/GNN-Wednesday.pdf [Прегледано 28.5.2021].

[3] Joey Mach: *Unlocking Drug Discovery With Machine Learning* Доступно на:https://towardsdatascience.com/unlocking-drug-discovery-through-machine-learning-part-1-8b2a64333e07[Прегледано 28.5.2021].

[4] Petar Veličković: *Neuralne mreže* Доступно на:http://ni.mg.edu.rs/static/resources/v3.0/sre1_neuralne_pv.pdf[Прегледано 28.5.2021].

[5] Kevin Yang, Kyle Swanson, Wengong Jin, Connor Coley, Philipp Eiden, Hua Gao, Angel Guzman-Perez, Timothy Hopper, Brian Kelley, Miriam Mathea, Andrew Palmer, Volker Settels, Tommi Jaakkola, Klavs Jensen, Regina Barzilay: *Analyzing Learned Molecular Representations for Property Prediction* Доступно на: https://pubs.acs.org/doi/pdf/10.1021/acs.jcim.9b00237 [Прегледано 28.5.2021].

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville: *Deep learning* Доступно на: https://www.deeplearningbook.org/ [Прегледано 28.5.2021].

[7] Christopher Thomas: *An introduction to Convolutional Neural Networks* Доступно на:https://towardsdatascience.com/an-introduction-to-convolutional-neural-networks-eb0b60b58fd7 [Прегледано 28.5.2021].

[8] William L. Hamilton: *Graph Representation Learning* Доступно на: https://www.cs.mcgill.ca/~wlh/grl_book/files/GRL_Book.pdf[Прегледано 28.5.2021].

[9] Daniel Godoy: *Understanding binary cross-entropy/log loss: a visual explanation* Доступно на: https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a.