НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

На правах рукопису
УДК 004.032.26, 615.072

До захисту допущено
Завідувач кафедри ММСА
_____ Оксана ТИМОЩУК
«___»_____2022 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 124 «Системний аналіз»

на тему: «Оцінка ADME-Tox властивостей молекул. Методи навчання багатозадачних нейронних мереж»

Виконав:
студент ІІ курсу, групи КА-11мп
Островський Захар Юрійович _____

Керівник:
доцент кафедри ММСА,
к.т.н., доц. Недашківська Н. І. _____

Рецензент:
доцент кафедри системного проектування
КПІ ім. Ігоря Сікорського, к.т.н., Безносик О. Ю. _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

Київ – 2022 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Рівень вищої освіти – другий (магістерський)

Спеціальність – 124 «Системний аналіз»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Оксана ТИМОЩУК
«___» _____ 2022 р.

ЗАВДАННЯ

на магістерську дисертацію студенту

Островського Захара Юрійовича

1. Тема дисертації: «Оцінка ADME-Tox властивостей молекул. Методи навчання багатозадачних нейронних мереж», комплексна робота у співавторстві із Возняком Володимиром Зіновійовичем і його роботою «Оцінка ADME-Tox властивостей молекул. Методи та моделі графових нейронних мереж», науковий керівник роботи Недашківська Надія Іванівна, доцент кафедри ММСА, кандидат технічних наук, затверджено наказом по університету №4046-с від «03» листопада 2022 р.
2. Термін подання студентом дисертації 12.12.2022.
3. Об’єкт дослідження: ADME-Tox властивості молекули.
4. Предмет дослідження: методи багатозадачного тренування.
5. Перелік завдань, які потрібно розробити:
    1) Дослідити поняття ADME-Tox властивостей молекул.
    2) Проаналізувати існуючі моделі для побудови графових нейронних мереж.

3) Проаналізувати можливі підходи для реалізації багатозадачності.
4) Обрати найкращий метод багатозадачного тренування.
5) Побудувати фінальну графову багатозадачну нейронну мережу для передбачень ADME-Tox властивостей молекул.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: презентація.

7. Дата видачі завдання: 01.09.2022 _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів роботи	Примітка
1	Затвердження теми МД	01.09.2022-07.09.2022	виконано
2	Ознайомлення зі структурою МД згідно з Положенням про державну атестацію студентів НТУУ «КПІ ім. І. Сікорського»	08.09.2022-14.09.2022	виконано
3	Ознайомлення з ДСТУ 3008-2015 та стандартами ЄСПД	15.09.2022-21.09.2022	виконано
4	Проведення дослідження за темою МД під керівництвом керівника	22.09.2022-28.09.2022	виконано
5	Завершення роботи над першим варіантом частини МД	29.09.2022-05.10.2022	виконано
6	Проведення роботи над експериментальною частиною МД	06.10.2022-12.10.2022	виконано
7	Проведення роботи над програмним продуктом	13.10.2022-02.11.2022	виконано
8	Оформлення МД та аналіз отриманих результатів	03.11.2022-13.11.2022	виконано

Студент Захар ОСТРОВСЬКИЙ

Керівник Надія НЕДАШКІВСЬКА

РЕФЕРАТ

Магістерська дисертація: 110 с., 28 табл., 43 рис., 1 додаток, 46 посилань.

ADME-TOX ВЛАСТИВОСТІ, ГРАФОВІ НЕЙРОННІ МЕРЕЖІ, БАГАТОЗАДАЧНЕ НАВЧАННЯ.

Об’єкт дослідження: ADME-Tox властивості молекули.

Мета дослідження: аналіз ADME-Tox параметрів, побудова багатозадачної нейронної мережі із використанням графових нейронних мереж для прогнозування ADME-Tox властивостей, порівняння якості цієї моделі із класичними моделями під кожен параметр у плані швидкості і якості прогнозу.

Використані методи: Рівнозначне зважування (EW), Випадкове зважування (RLW), Зважування за невизначеністю (UW), Нормалізація градієнтів (GradNorm), Динамічне усереднення коефіцієнтів (DWA), “Неконфліктуючий” градієнтний спуск (CAGrad), Неупереджене багатозадачне навчання (IMTL), Проєктування конфліктуючих градієнтів (PCGrad), Вакцинування градієнтів (GradVac).

Отриманні результати: побудовано графову багатозадачну нейронну мережу для прогнозування ADME-Tox властивостей молекул із швидкістю прогнозу 500 тисяч молекул за 280 секунд для середньостатистичної машини із графічним процесором GPU та за 550 секунд для середньостатистичної машини із процесором CPU.

У рамках подальшого дослідження пропонується підвищувати точність кожного ADME-Tox параметру, а також побудувати універсальний молекулярний декодувальник, який переводитиме будь-яку молекулу у векторний простір.

ABSTRACT

Master’s thesis: 110 p., 28 tabl., 43 fig., 1 addition, 46 sources.

ADME-TOX PROPERTIES, GRAPH NEURAL NETWORKS, MULTI-TASK LEARNING.

Research object: ADME-Tox molecular properties.

Purpose: analyze ADME-Tox parameters, construct a multi-task neural network using graph neural networks to predict ADME-Tox properties, compare the quality of this model with classical ML models for each parameter in terms of speed and quality of prediction.

Methods Used: Equal Weighting (EW), Random Weighting (RLW), Uncertainty Weighting (UW), Gradient Normalization (GradNorm), Dynamic Weight Averaging (DWA), "Conflict-Free" Gradient Descent (CAGrad), Impartial Multi-Task Learning (IMTL) , Projecting conflicting gradients (PCGrad), Vaccination of gradients (GradVac).

Results: a graph multitasking neural network is capable of predicting ADME-Tox properties of molecules with a speed of 500,000 molecules in 280 seconds for average machines with a GPU and in 550 seconds for average machines with a CPU.

As a part of further research, it is proposed to improve the accuracy of each parameter of ADME-Tox, as well as to build a universal molecular decoder that translates any molecule into a latent vector space.

ЗМІСТ

ВСТУП 8
РОЗДІЛ 1 ТЕОРЕТИЧНІ ВІДОМОСТІ ПРО ADME-TOX ВЛАСТИВОСТІ МОЛЕКУЛИ 10
1.1 Загальні теоретичні відомості про ADME-Tox 10
1.2 Аналіз кожного параметра ADME-Tox 12
1.2.1 Absorption (абсорбція) 12
1.2.2 Distribution (розповсюдження) 17
1.2.3 Metabolism (метаболізм) 19
1.2.4 Excretion (виділення) 21
1.2.5 Toxicity (токсичність) 24
Висновки 32
РОЗДІЛ 2 ТЕОРЕТИЧНІ ВІДОМОСТІ ПРО ГРАФОВІ НЕЙРОННІ МЕРЕЖІ ТА ПІДХОДИ ДО ПОБУДОВИ БАГАТОЗАДАЧНОЇ МОДЕЛІ 34
2.1 Графові нейронні мережі 34
2.1.1 Графові згорткові нейронні мережі 37
2.1.2 Графові уважні нейронні мережі 39
2.1.3 Графові нейронні мережі із передачею повідомлень 40
2.1.4 Графові нейронні мережі для молекулярних задач 41
2.2 Підходи до побудови багатозадачної нейронної мережі 52
2.2.1 Рівнозначне зважування (EW) 54
2.2.2 Випадкове зважування (RLW) 54
2.2.3 Зважування за невизначеністю (UW) 55
2.2.4 Нормалізація градієнтів (GradNorm) 55
2.2.5 Динамічне усереднення коефіцієнтів (DWA) 56
2.2.6 “Неконфліктуючий” градієнтний спуск (CAGrad) 57
2.2.7 Неупереджене багатозадачне навчання (IMTL) 58

2.2.8 Проєктування конфліктуючих градієнтів (PCGrad) 60
2.2.9 Вакцинування градієнтів (GradVac) 61
Висновки 61
РОЗДІЛ 3 ПОБУДОВА ГРАФОВОЇ БАГАТОЗАДАЧНОЇ НЕЙРОННОЇ МЕРЕЖІ 64
3.1 Опис проблеми й огляд датасетів 64
3.2 Вибір методу багатозадачного тренування 67
3.2.1 Терміни 67
3.2.2 Постановка досліджень 68
3.2.3 Методика досліджень 69
3.2.4 Встановлення взаємозалежностей 71
3.2.5 Висновки за результатами проведених експериментів по вибору методу багатозадачного тренування 80
3.3 Побудова результуючої графової багатозадачної нейронної мережі та її порівняння із класичними моделями машинного навчання 81
Висновки 87
РОЗДІЛ 4 РОЗРОБКА ВЛАСНОГО СТАРТАП-ПРОЄКТУ 89
4.1 План розробки стартапу та масштабування його на ринок 89
4.2 Опис ідеї стартап-проєкту 91
4.3 Технологічний аудит ідеї проєкту 92
4.4 Аналіз ринкових можливостей запуску стартап-проєкту 94
4.5 Розроблення ринкової стратегії стартап-проєкту 100
4.6 Розроблення маркетингової програми стартап-проєкту 103
Висновки 105
ВИСНОВКИ 106
ПЕРЕЛІК ПОСИЛАНЬ 108
ДОДАТОК А ЛІСТИНГ ПРОГРАМНОГО ПРОДУКТУ 111

ВСТУП

Актуальність обраної теми. Оцінка ADME-Tox (absorption – абсорбція, distribution – розподіл, metabolism – метаболізм, excretion – виділення, і toxicity – токсичність) властивостей молекул має надзвичайно важливе значення на сьогоднішній день у сфері розробки ліків (Drug Discovery). Дуже важливо, щоб згенерована молекула (наші нові ліки) володіла всіма властивостями, які ми їй ставимо (тобто, щоб значення кожного параметру коливалися в межах якогось діапазону, якщо це регресійна задача, або відповідали конкретному значенню, якщо це класифікаційна задача). Оскільки проста перевірка в лабораторіях таких згенерованих молекул дуже дорога й часозатратна, зараз біологи спільно із інженерами машинного навчання пробують будувати моделі, які могли із відносно невеликою похибкою передбачати ці ADME-Tox властивості. Проте таких параметрів може бути дуже багато й, побудувавши окремі моделі під конкретні параметри, прогнозування таких властивостей може теж займати доволі багато часу для, скажімо, 1 мільярда молекул. Саме тому ми вирішили побудувати багатозадачну нейронну мережу із використанням графових нейронних мереж для передбачення ADME-Tox параметрів, що дасть нам значний виграш у швидкості відносно класичних моделей під кожен параметр, проте без значного програшу в якості прогнозування.

Метою дослідження є аналіз ADME-Tox параметрів, побудова багатозадачної неройнної мережі із використанням графових нейронних мереж для прогнозування ADME-Tox параметрів, порівняння якості цієї моделі із класичними моделями під кожен параметр у плані швидкості і якості прогнозу.

Для виконання сформульованої мети дослідження, нами були поставлені такі завдання:

1. Розглянути поняття ADME-Tox параметрів.

2. Проаналізувати графові підходи для конвертування молекул у вектор чисел за допомогою графових нейронних мереж.

3. Проаналізувати можливі підходи для побудови багатозадачної нейронної мережі.

4. Побудувати багатозадачну нейронну мережу із використанням графових нейронних мереж для передбачення ADME-Tox параметрів.

5. Порівняти побудовану модель із класичними моделями під кожен параметр у плані швидкості і якості прогнозування.

6. Створити програмний продукт мовою програмування Python задля реалізації моделі.

Об’єктом дослідження є ADME-Tox властивості молекули, як одних із найважливіших показників якості згенерованої молекули.

Предметом дослідження є багатозадачна графова нейронна мережа для прогнозування ADME-Tox параметрів та її програмна реалізація.

Щодо джерельної бази досліджень, дана робота ґрунтується на аналізі науково-методичної літератури, методичних посібників, наукових статей, періодичних видань та напрацювань вітчизняних та іноземних вчених і дослідників, а також практичних розробок за темою ADME-Tox властивостей молекули та побудови моделей для їх прогнозування.

Наукова новизна полягає у побудові саме багатозадачної графової нейронної мережі для прогнозування ADME-Tox параметрів, яка не буде поступатися в якості класичним підходам для прогнозування окремих параметрів і буде значно переважати такі підходи в швидкості прогнозування.

РОЗДІЛ 1 ТЕОРЕТИЧНІ ВІДОМОСТІ ПРО ADME-TOX ВЛАСТИВОСТІ МОЛЕКУЛИ

1.1 Загальні теоретичні відомості про ADME-Tox

Важливою частиною відкриття та розробки ліків є проведення досліджень DMPK (метаболізму та фармакокінетики лікарських засобів), які часто називають дослідженнями ADME-Tox (absorption – абсорбція, distribution – розподіл, metabolism – метаболізм, excretion – виділення, і toxicity – токсичність). Ці дослідження допомагають визначити життєздатність препарату-кандидата, відповідаючи на такі ключові запитання:

- **A**bsorption (абсорбція) – скільки препарату всмоктується і як швидко?
- **D**istribution (розповсюдження) – де поширюється препарат в організмі? Яка швидкість і ступінь поширення?
- **M**etabolism (метаболізм) – як швидко метаболізується препарат? Який механізм дії? Який метаболіт утворюється, активний він чи токсичний?
- **E**xcretion (виділення) – як і як швидко виводиться препарат?
- **T**oxicity (токсичність) – чи має цей препарат токсичну дію на системи або органи організму?

Підраховано, що близько 50% кандидатів на ліки не вдаються через неприйнятну ефективність і що до 40% кандидатів на ліки в минулому зазнали невдачі через токсичність. Такі ліки, як мібефрадил, сорувідин та фенілпропаноламін гідрохлорид, були зняті з ринку через взаємодію ліків або токсичність. Як для регуляторів, так і для виробників лікарських засобів стало очевидним, що на додаток до фармакологічних властивостей дослідження ADME/Tox відіграють вирішальну роль в успіху кандидата на ліки. У зв’язку з цим впливом на кінцевий успіх ці дослідження зараз проводяться на початку процесу відкриття ліків.

Дослідження in vitro і in vivo проводяться, щоб дозволити розробнику ліків прийняти рішення про те, чи слід вибирати лікарський засіб як кандидат

на лікування, і перейти до пізніх доклінічних та клінічних програм. Властивості ADME дозволяють розробникам ліків зрозуміти безпеку та ефективність препарату-кандидата, і вони необхідні для схвалення регулятором.

Управління з контролю за продуктами і ліками (FDA) підготувало декілька рекомендаційних документів для промисловості, таких як тестування на безпеку метаболітів ліків, дослідження метаболізму in vitro та опосередковані транспортером взаємодії лікарських засобів, клінічні дослідження взаємодії лікарських засобів — дизайн дослідження, аналіз даних, клінічні наслідки тощо. Основна мета та кінцевий результат усіх досліджень ADME-Tox полягає в тому, щоб краще зрозуміти опосередковану метаболітами токсичність і профіль безпеки сполуки, щоб прийняти конкретне рішення про те, чи може сполука перейти до пізніх стадій доклінічних та клінічних досліджень, щоб дозволити подати заявку на дослідницьку експертизу. Новий лікарський засіб (IND), заявка на новий лікарський засіб (NDA) або заявка на ліцензування біологічних препаратів (BLA).

Хоча кожен препарат є унікальним, конкретні моделі та пов’язані аналізи, як визначено в керівних документах FDA, допомагають вченим визначити, які властивості ADME слід оцінити. Наприклад, моделі мікросом печінки та цілих гепатоцитів зазвичай використовуються в дослідженнях ADME in vitro, обидві моделі містять ферменти метаболізму, такі як CYP450 та UDP-глюкуронозилтрансфераза (UGT). Ці моделі in vitro можна застосувати до таких аналізів, як інгібування та індукція CYP4. Для оцінки кишкової проникності використовуються аналізи in vitro, такі як дослідження на основі клітин CACO-2 або MDCK. Під час відкриття, а також на пізніх стадіях доклінічних і доклінічних досліджень проводяться дослідження in vivo для оцінки фармакокінетичних (ФК) властивостей. ФК-дослідження in vivo проводяться з акредитованими тваринами, такими як миші та щури, акредитованими Асоціацією оцінки та акредитації лабораторних тварин (AAALAC), тоді як примати, які не є людьми, використовуються для отримання даних фармакологічної активності для оцінки таких властивостей, як кліренс ліків,

біодоступність, експозиція, період напіввиведення та об’єм розподілу4. Ці дослідження включають поєднання токсикологічних досліджень, не пов’язаних із GLP та GLP.

1.2 Аналіз кожного параметра ADME-Tox

Як ми вже згадували раніше, в ADME-Tox входять absorption (абсорбція), distribution (розподіл), metabolism (метаболізм), elimination (виведення), і toxicity (токсичність). Спробуємо проаналізувати, за допомогою яких параметрів ми зможемо охарактеризувати ці властивості.

1.2.1 Absorption (абсорбція)

1) Bioavailability (біодоступність)

Біодоступність (F) визначається як швидкість і ступінь, з якою активний компонент або активна частина лікарського засобу абсорбується з лікарського засобу та досягає циркуляції.

Біодоступність зазвичай оцінюють шляхом визначення площі під кривою концентрація-час у плазмі (рис. 1.1). Найнадійнішим показником біодоступності препарату є AUC. AUC прямо пропорційний загальній кількості незміненого препарату, що досягає системного кровообігу. Біодоступність препарату при пероральному прийомі — це частка дози препарату, яка врешті досягає терапевтичного місця дії та кількісно позначається як %F.

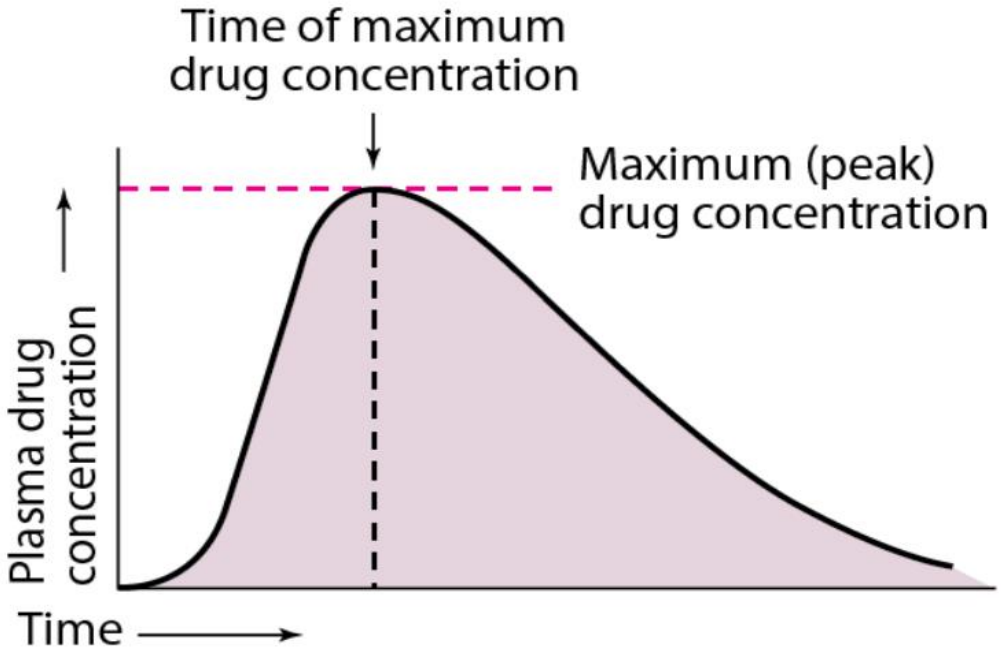

Рисунок 1.1 – Оцінка біодоступності [27]

2) Caco-2

Клітинна лінія Caco-2 є популярним сурогатом епітелію кишечника людини для оцінки проникності препарату in vivo через їх морфологічну та функціональну схожість з ентероцитами людини. Більше того, експресія транспортерів у клітинах Caco-2, таких як P-глікопротеїн (P-gp), транспортер олігопептидів із протонним зв'язком (PEPT1) та інші, пропонує великі переваги порівняно із спрощеними моделями. При оцінці фармакокінетичних властивостей in vitro для оцінки проникності кишечника людини (HIP) зазвичай використовують коефіцієнт проникності моношарової клітини Caco-2 (Papp).

Диференційовані та поляризовані клітини Caco-2 (21-денна система) висаджують на 96-лункову проникну систему у вигляді одного моношару, щоб забезпечити автоматизований високопродуктивний скринінг сполук. Транспорт ліків оцінюють в обох напрямках (від апікального до базолатерального (A-B) і від базолатерального до апікального (B-A)) через моношар клітини. Буфер, що використовується для аналізу, не містить HEPES, щоб мінімізувати інгібуючий ефект на транспортери поглинання (рис. 1.2).

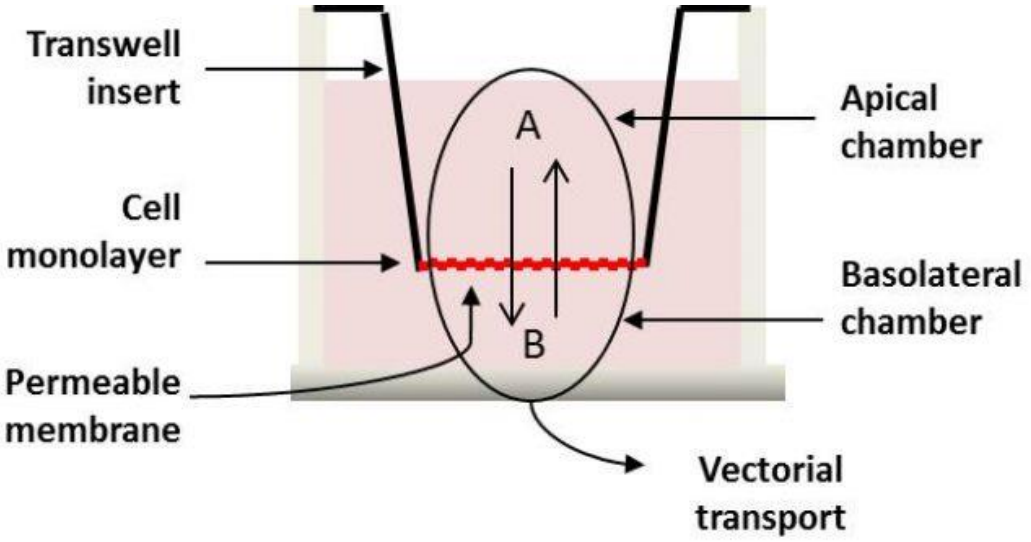

Рисунок 1.2 – Візуальне пояснення Caco-2 [27]

3) Human intestinal absorption (всмоктування в кишечнику людини) – HIA

Властивість кишкової абсорбції людини відноситься до процесу, за допомогою якого лікарські засоби, що вводяться перорально, всмоктуються з шлунково-кишкової системи в кровотік людського організму.

В експериментах кишкове всмоктування зазвичай вимірюється фракцією всмоктування, %FA, яка визначається як загальна абсорбована маса, поділена на дану дозу препарату. Серед методів, які лежать в основі отримання даних HIA, найбільш широко використовуються паралельний аналіз проникності штучної мембрани (PAMPA), клітинні лінії аденокарциноми товстої кишки людини (Caco-2) та клітини собачої нирки Мадіна Дарбі (MDCK). Зокрема, деякі дослідження показали, що існує хороший сигмовидний зв’язок між всмоктуванням препарату людиною перорально та проникністю Caco-2, але цей зв'язок здається не таким зрозумілим. Клітинні лінії Caco-2 можна використовувати як інструмент прогнозування для оцінки перорального всмоктування тільки для сполук з хорошою абсорбцією в кишечнику. У той час як для сполук з низьким або середнім поглинанням, клітинні лінії Caco-2 можуть не дати дуже гарного рейтингу для оцінки перорального всмоктування. Тому навіть якщо моделі прогнозування проникності Caco-2 розвинулися

багато, якісна та практична прогнозна модель HIA все ще користується попитом.

4) P-dp inhibitors (PGP-інгібітори)

Витікаючий транспортер Р-глікопротеїн (P-gp) кодується геном множинної лікарської стійкості (MDR1, ABCB1) і належить до сімейства транспортерів АТФ-зв'язуючих касет (ABC). Він широко поширюється в організмі і відповідає за обмеження клітинного поглинання та розподілу ксенобіотиків і токсичних речовин. Субстратами для нього є сотні різноманітних за структурою терапевтичних засобів, які перешкоджають всмоктуванню, проникності та утриманню ліків, видавлюючи їх із клітин.

Інгібування ефлюксного насоса в основному здійснюється для того, щоб покращити доставку терапевтичних агентів (рис. 1.3). Загалом, P-gp можна інгібувати трьома механізмами: 1) блокування сайту зв'язування лікарського засобу конкурентно, неконкурентно або алостерично; 2) перешкоджання гідролізу АТФ; 3) зміна цілісності ліпідів клітинної мембрани.

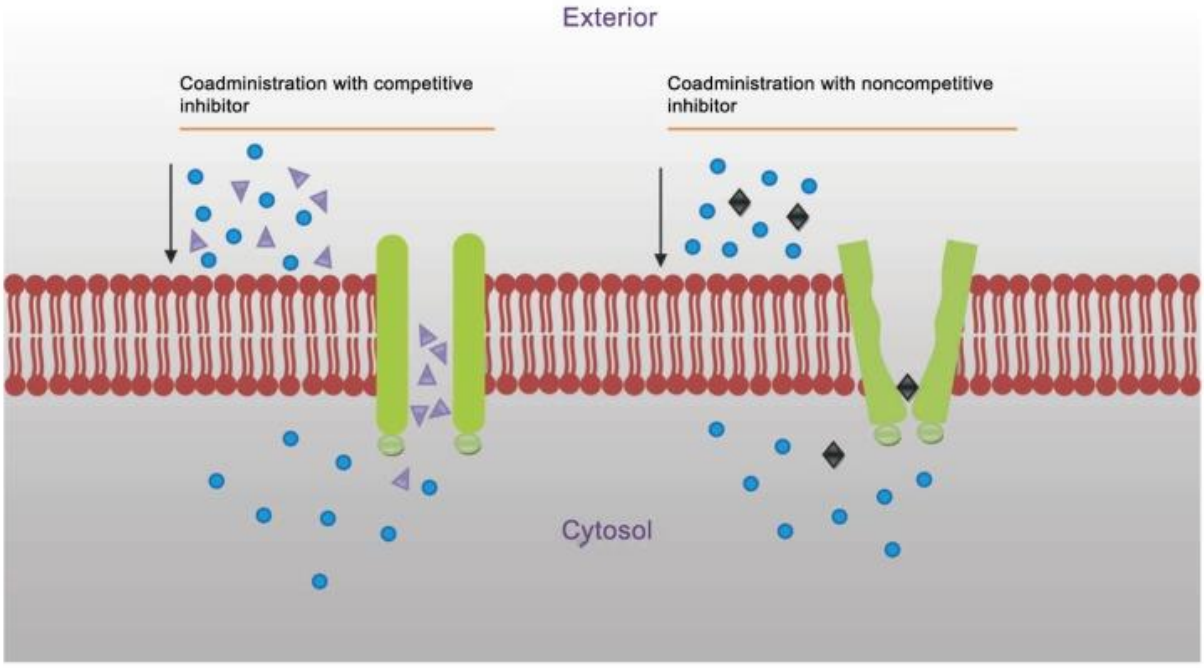

Рисунок 1.3 – Візуальне зображення P-db inhibitors [27]

Для оцінки впливу P-gp на кишкове всмоктування ліків, клітини Caco-2, клітинна лінія клітин колоректальної аденокарциноми людини, широко

використовуються в дослідженнях in vitro, оскільки Caco-2 клітини експресують кілька транспортерів переносників розчиненої речовини (SLC), транспортери ABC та метаболічні ферменти. З іншого боку, миші з нокаутом mdr1a/1b (KO) часто використовуються для досліджень in vivo функції P-gp. Для оцінки інгібування P-gp було визначено напівмаксимальну концентрацію інгібітора (IC50) і константу дисоціації для комплексу інгібітор/P-gp (Ki).

5) P-gp substrates (PGP субстрати)

Активність P-gp знижує внутрішньоклітинну концентрацію ліків від раку, що дозволяє розвивати стійкість до них. Субстрати P-gp можуть посилювати виведення ліків з організму, знижувати ефективність препарату та включати розвиток толерантності.

Субстрати, як правило, оцінюються двонаправленими транспортними аналізами, які виявляють очевидний або чистий (неефективний) транспорт. Таким чином, субстрат визначається як сполука, яка демонструє чистий відтік (тобто більший активний витік за P-gp, ніж пасивний приплив) у транспортному аналізі. Клітини MDR1-MDCK є звичайним вибором для оцінки субстратів та інгібіторів P-gp.

Збір даних проводився систематичним шляхом широкомасштабного скринінгу цитотоксичності сполук проти 60 ліній ракових клітин людини в програмі терапевтичної терапії Національного інституту раку (NCI-DTP). Панель лінії ракових клітин NCI-DTP (NCI-60) неоднорідна за рівнями експресії гена ABCB1, і, відповідно, рівні експресії ABCB1 повинні обернено корелювати з цитотоксичністю для субстратів P-gp, але не для несубстратів P-gp. Така кореляція може бути успішно використана для ідентифікації субстратів P-gp. NCI-60 містить дві клітинні лінії, які були важливими для цього дослідження: OVCAR-8 і NCI/ADR-RESa, обидві клітини яєчників, отримані від однієї особини. Ключова відмінність між двома клітинними лініями полягає у високій експресії ABCB1 в лінії клітин NCI/ADR-RES.

1.2.2 Distribution (розповсюдження)

1) BBB

Гематоенцефалічний бар'єр (ГЕБ) є регуляторним і захисним механізмом в ЦНС, який важливий для гомеостазу нервової системи і захищає її від токсинів, патогенів та інших шкідливих агентів. Проте, за оцінками, деякі молекули ліків проникні для ГВВ, щоб досягти своєї мети. З цієї причини встановлення проникності ГВВ є вирішальним для процесу відкриття та розробки ліків.

Проникність ГЕБ виміряли у людей та на моделях на тваринах за допомогою методів нейровізуалізації, включаючи динамічну перфузійну КТ (ПКТ) та динамічну магнітно-резонансну томографію з посиленим контрастом (DCEMRI). Методи МРТ були засновані на впливі витоку контрастної речовини на сигнал протонів води в позасудинному просторі і пропонують кілька переваг перед методами, заснованими на інших методах зображення, включаючи спільне отримання та простоту спільної реєстрації з іншими зображеннями МРТ, які зазвичай отримують при оцінці захворювання або травма мозку.

Для аналізів in vitro додають досліджувану сполуку та флуоресцентний зонд, а культури повертають інкубувати протягом визначеного часу. Середовище відбирається для кількісного визначення досліджуваних виробів за допомогою UPLC та для кількісного визначення флуоресцентних зондів за допомогою зчитувача планшетів (рис. 1.4).

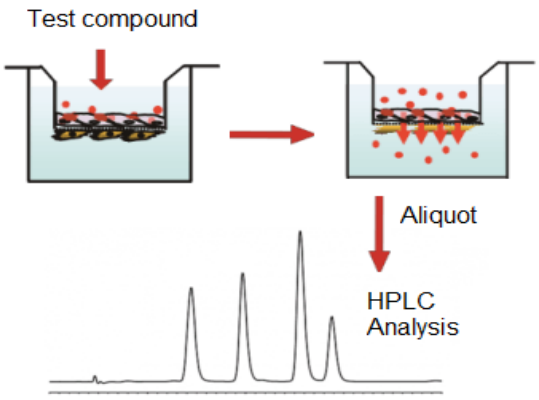

Рисунок 1.4 – Візуальне зображення BBB [27]

2) PPB

Зв’язування з білками плазми означає ступінь зв’язування ліків з білками в крові. На ефективність препарату може впливати ступінь його зв’язування. Чим менше зв’язаний препарат, тим ефективніше він може проникати через клітинні мембрани або дифундувати (рис. 1.5).

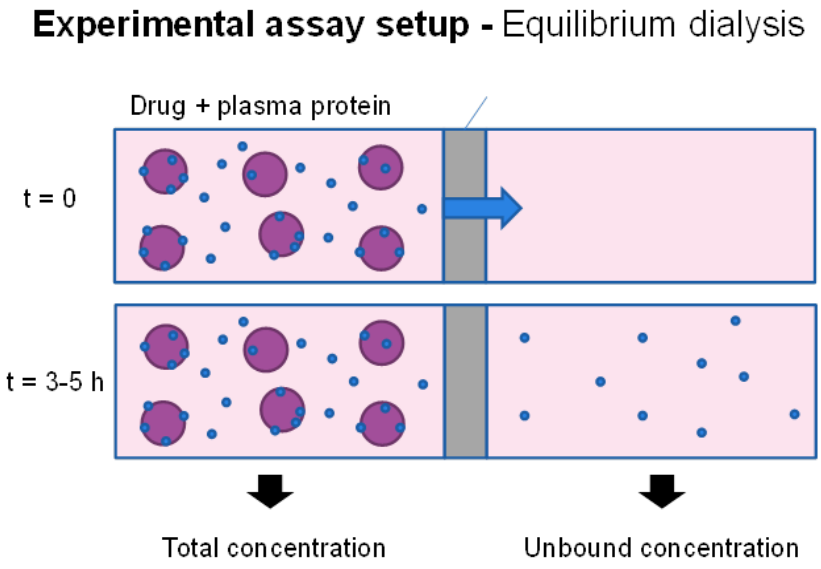

Рисунок 1.5 – Візуальне зображення PPB [27]

3) VD

Об’єм розподілу (VD) – це об’єм, на який розподіляється запитувана сполука для встановлення рівної концентрації, яка знаходиться в плазмі крові. В основному, цей параметр являє собою здатність ліків перерозповсюджуватися по тканинах. VD прямо пропорційна кількості препарату, розподіленого в тканині; більший VD вказує на більший розподіл тканин.

Концентрацію в плазмі можна спостерігати в різний час, що призводить до кількох різних можливих стратегій для розрахунку об’єму розподілу (рис. 1.6).

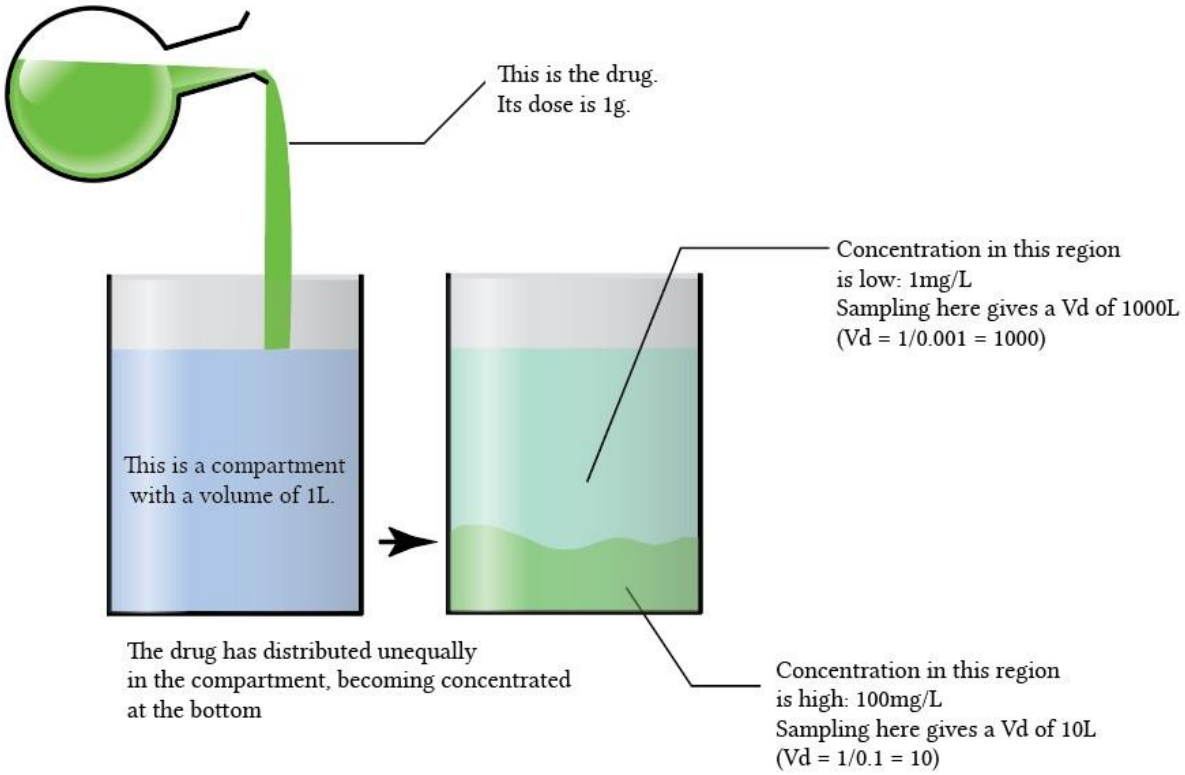

Рисунок 1.6 – Візуальне зображення VD [27]

1.2.3 Metabolism (метаболізм)

1) CYP inhibitors (CYP інхібітори)

Цитохром P450 (CYP) – це сімейство ферментів, які відіграють важливу роль у метаболізмі ліків. Оцінка потенціалу сполуки інгібувати специфічний фермент цитохрому P450 є важливою, оскільки одночасне введення сполук може призвести до того, що одне або обидва пригнічують метаболізм іншого.

Ізоформи, доступні для прогнозування: CYP1A2, CYP2C9, CYP2C19, CYP2D6 і CYP3A4.

Стандартний протокол аналізу включає використання методу LC-MS/MS аналізу. Кінцевими точками інгібування CYP є IC50 та Ki (рис 1.7).

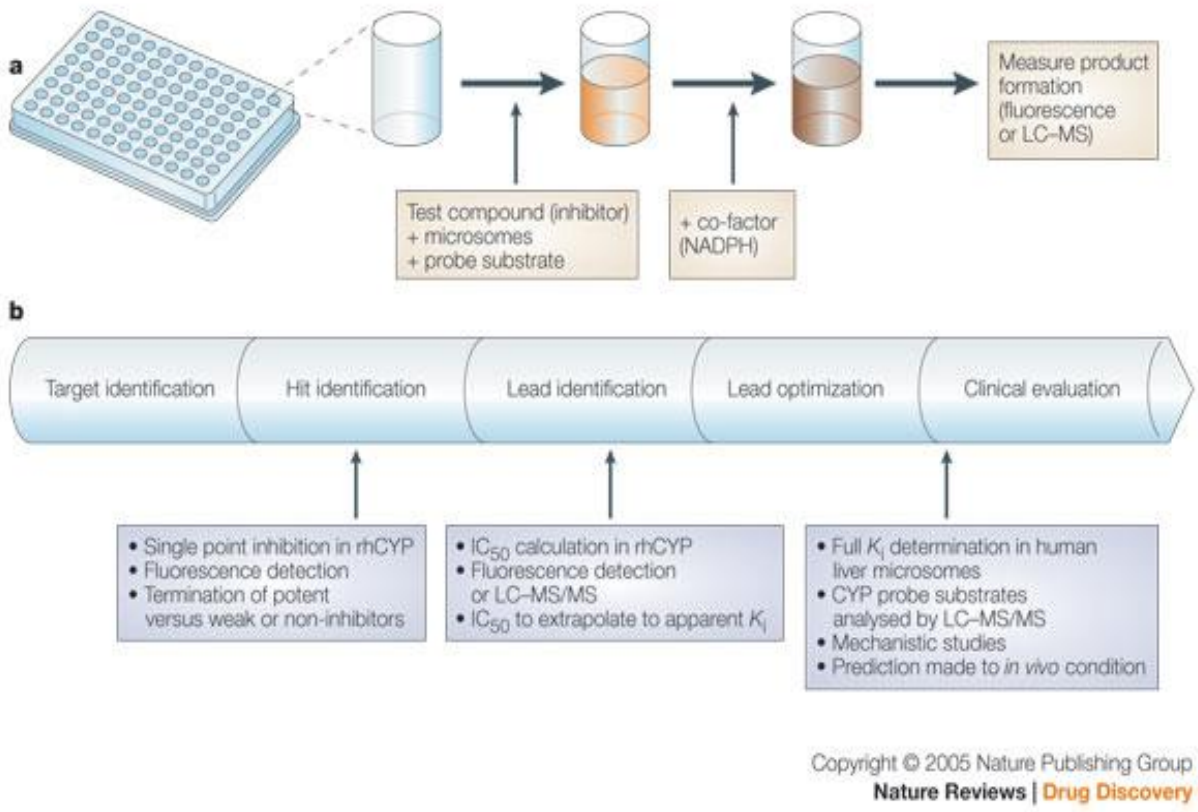

Рисунок 1.7 – Візуальне зображення CYP інхібіторів [27]

2) CYP substrates (CYP субстрати)

Субстрати контрольного цитохрому P450 (CYP) та їх відповідні метаболіти зазвичай використовуються в службах виявлення ліків на ранній стадії для визначення здатності сполуки запобігати виробленню специфічних метаболітів, щоб пришвидшити вибір кандидатів на ліки.

Стандартний протокол аналізу включає використання методу LC-MS/MS аналізу. Вихідна інформація цього аналізу відноситься до якісних результатів (рис. 1.8).

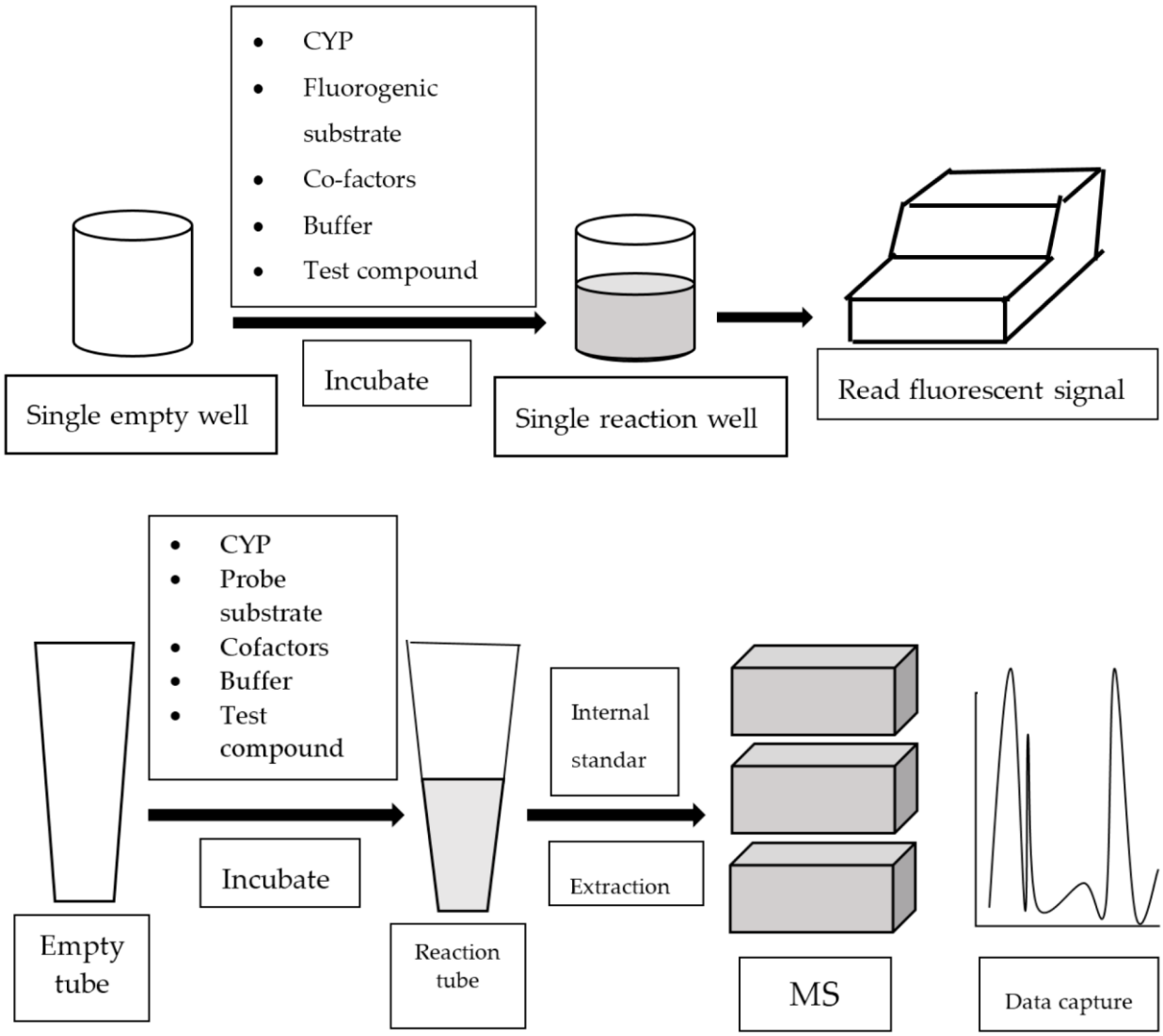

Рисунок 1.8 – Візуальне зображення CYP субстратів [27]

1.2.4 Excretion (виділення)

1) Half-life

Період напіввиведення лікарського засобу – це час, необхідний для того, щоб кількість активної речовини препарату в організмі зменшилася вдвічі.

Наприклад, якщо прийняти 100 мг препарату з періодом напіввиведення 60 хвилин, оцінюється наступне:

– через 60 хвилин після введення залишається 50 мг;

– через 120 хвилин після введення залишається 25 мг;

– через 180 хвилин після введення залишається 12,5 мг;

– через 240 хвилин після введення залишається 6,25 мг;

– через 300 хвилин після введення залишається 3,125 мг.

Період напіввиведення препарату можна визначити, використовуючи таке рівняння: t1/2 = (0,7 x Vd)/Cl, де Vd – об’єм розподілу, а Cl – кліренс. Для введення T1/2 слід взяти зразки крові. Час внутрішньовенного введення проби має бути не «занадто близьким» до часу прийому препарату і не «занадто пізнім», щоб правильно екстраполювати результати (рис. 1.9).

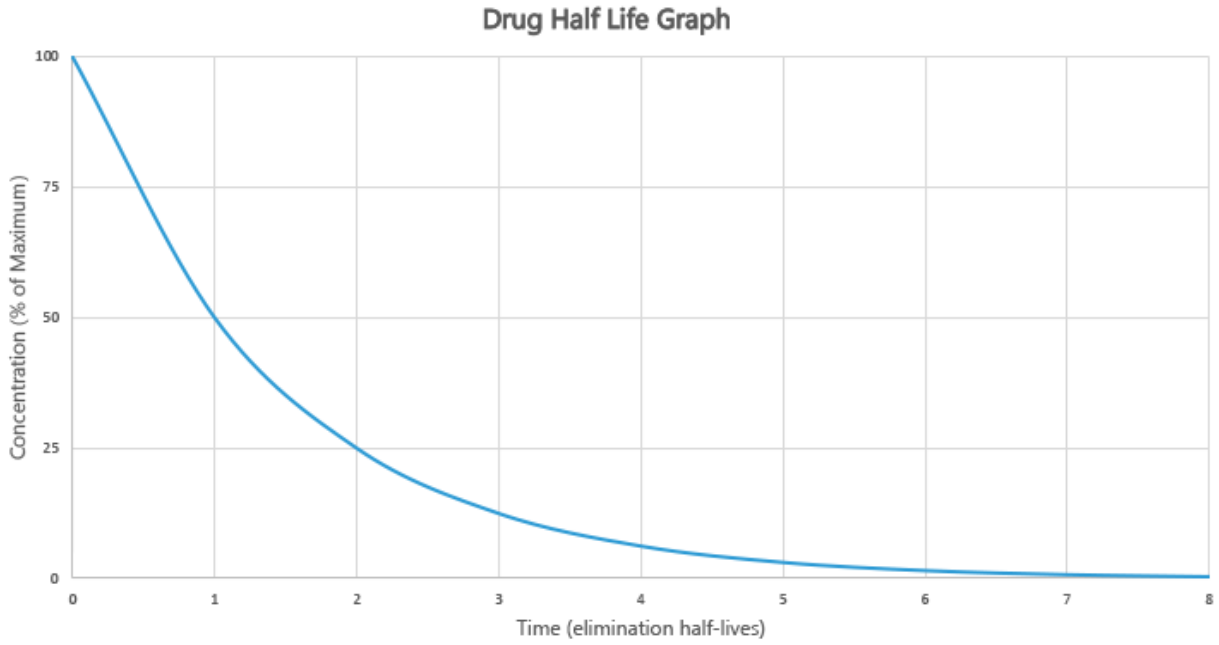

Рисунок 1.9 – Приклад Half-life [27]

2) Plasma Clearance (плазмовий кліренс)

Плазмовий кліренс відноситься до систематичного або повного кліренсу і залежить від метаболізму в різних органах, головним чином у печінці та нирках. Це важливий фармакокінетичний параметр, оскільки дані про кліренс із плазми дозволяють розрахувати дозу препарату, необхідну для отримання рівноважної концентрації.

Кліренс дорівнює швидкості, з якою препарат виводиться з плазми (мг/хв), поділеній на концентрацію цього препарату в плазмі (мг/мл). Існує важливий зв’язок між кліренсом, періодом напіввиведення та об’ємом розподілу. Константа швидкості виведення препарату Kel еквівалентна

загальному кліренсу, поділеному на об’єм розподілу (Kel=Cltot/Vd) (зверніть увагу на використання Cl, а не K, щоб не плутати з Kel). Але Kel також еквівалентний ln2, поділеному на швидкість напіввиведення $t_{1/2}$ (Kel=ln2$t_{1/2}$). Таким чином, Cltot = ln2 Vd/$t_{1/2}$. Це означає, наприклад, що збільшення загального кліренсу призводить до зменшення швидкості напіввиведення за умови постійного об’єму розподілу.

3) Renal Clearance (нирковий кліренс)

Нирковий кліренс речовини відноситься до того, як швидко певна речовина виводиться з плазми нирками і виводиться з сечею. Тож щось із високим нирковим кліренсом означає, що воно буде швидко видалено з крові, і навпаки. Нирковий кліренс є основним шляхом виведення ліків з незначним метаболізмом і виведенням з жовчі.

Нирковий кліренс (ClR) — це міра транспорту нирками в одиницях об’єму плазми за одиницю часу. Виміряний об’єм плазми – це той об’єм, для якого дана речовина (наприклад, сечовина або ліки) повністю видаляється за хвилину. Альтернативним підходом до вимірювання ниркового кліренсу є співвіднесення швидкості виведення маркера або препарату з його концентрацією в плазмі. де Ae – кількість, що виділяється із сечею в момент часу t, а C – концентрація в плазмі.

Пасивно очищені сполуки, як правило, добре прогнозуються з даних про тварин, оскільки CLr = ШКФ × fu (де CLr — нирковий кліренс, швидкість клубочкової фільтрації ШКФ, fu — незв’язана фракція). Зменшення ниркового кліренсу з метою зменшення загального кліренсу організму зазвичай досягається шляхом підвищення ліпофільності за допомогою модуляції LogP або pKa.

1.2.5 Toxicity (токсичність)

1) Acute toxicity (гостра токсичність)

Гостра токсичність описує несприятливі ефекти речовини, які є результатом одноразового або багаторазового впливу протягом короткого періоду часу (зазвичай менше 24 годин). Щоб описати як гостру токсичність, побічні ефекти повинні виникнути протягом 14 днів після введення речовини.

Призначається перорально щурам. Тестування на гостру токсичність вимагає, щоб досліджувані матеріали були надані тваринам протягом обмеженого, але короткого періоду часу, зазвичай у вигляді одноразового впливу. Середня смертельна доза (або LD50) визначається як доза досліджуваної речовини, яка є смертельною для 50% тварин у групі дозування. Значення LD50 використовувалися для порівняння відносної гострої небезпеки промислових хімікатів, особливо якщо для цих хімічних речовин немає інших токсикологічних даних (рис. 1.10).

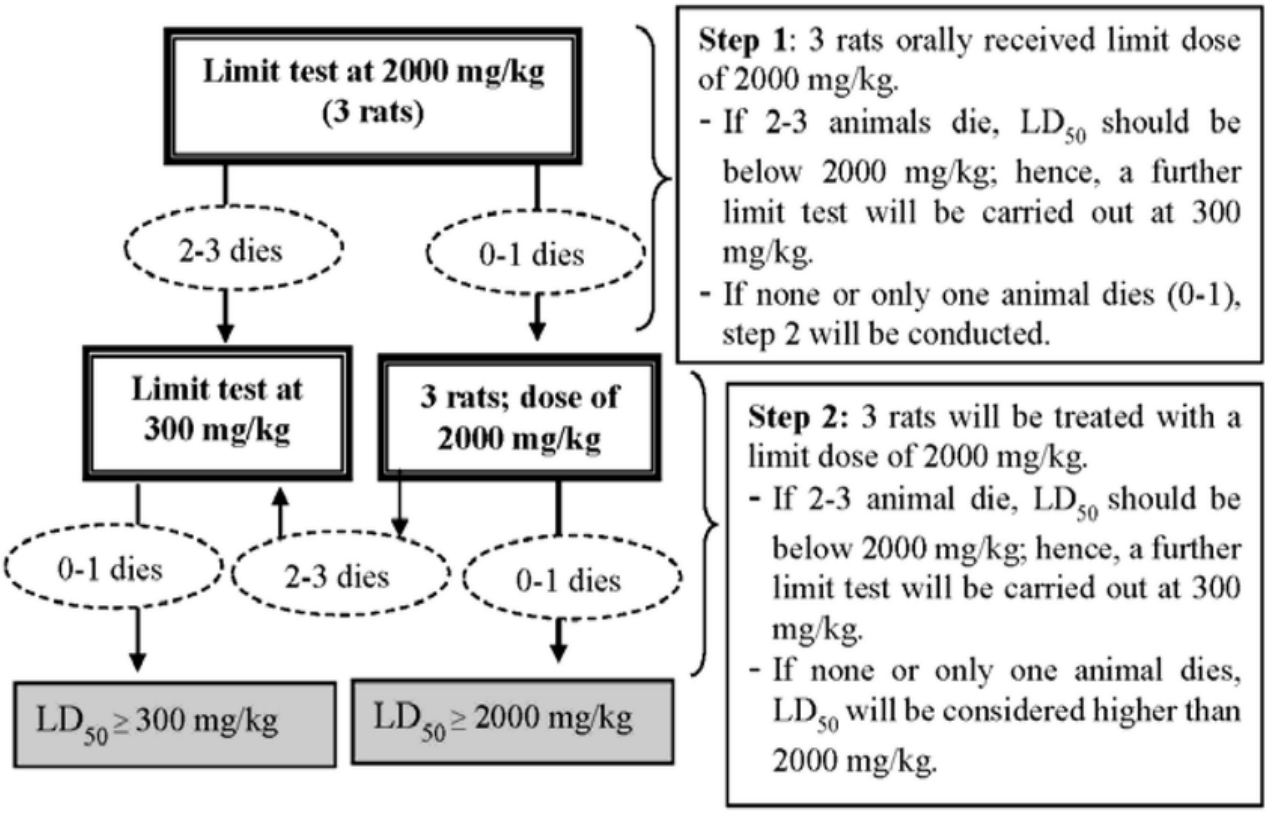

Рисунок 1.10 – Візуальне зображення гострої токсичності [27]

2) AMES Mutagenicity (AMES мутагенність)

У відкритті ліків мутагенність є проблемою, якої необхідно уникати. Виявлення мутагенності на етапах доклінічного відкриття ліків могло б зупинити розробку потенційно шкідливих ліків і сприяти розробці безпечних терапевтичних засобів. Мутагенність – це термін, який використовується для широкого опису властивості хімічних агентів або лікарських речовин викликати генетичні мутації. Іноді його використовують як синоніми з терміном генотоксичність, особливо щодо обговорення хімічних агентів, які згубно змінюють генетичний матеріал у клітині. Однак, хоча всі мутагени є генотоксичними, не всі генотоксичні речовини є мутагенними.

Тест AMES (аналіз бактеріальної зворотної мутації) є одним із еталонних біологічних аналізів для виявлення мутагенності в конвеєрі виявлення ліків. Він заснований на виявленні помилок зсуву кадрів або заміни пари основ після додавання досліджуваної сполуки до гістидин-залежних штамів Salmonella typhinurium. Під час впливу на клітини мутагену зворотні мутації відновлюють здатність штаму виробляти гістидин і дозволяють колонії рости на середовищі з дефіцитом гістидину.

Хімічна речовина визначається як позитивна за AMES (або метагенна), якщо вона значною мірою індукує ревертантний ріст колонії, хоча й у штамів сальмонели ⅕ (рис. 1.11).

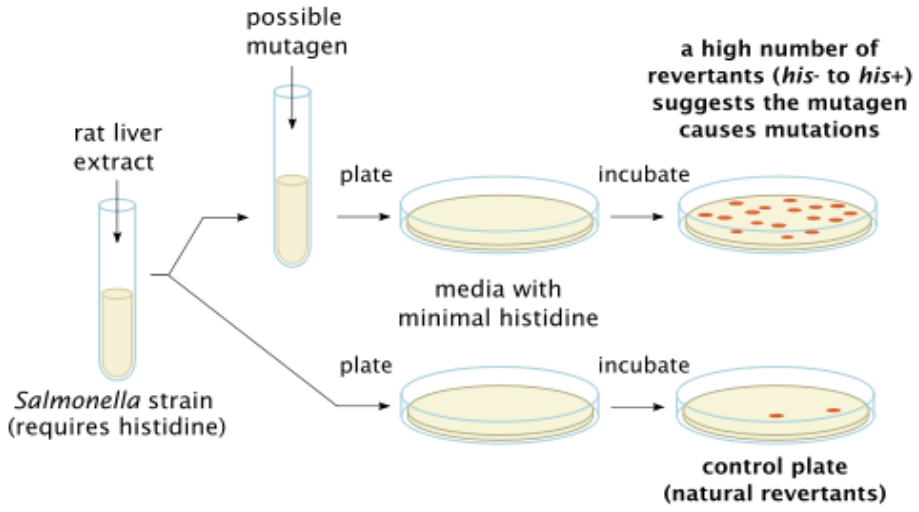

Рисунок 1.11 – Візуальне зображення AMES [27]

3) Carcinogenicity (канцерогенність)

Канцерогенність - це здатність або схильність викликати рак. Канцероген - це речовина, яка здатна збільшити частоту виникнення пухлини або скоротити час до появи пухлини. Мутаген і канцероген – це два фізичні, хімічні або біологічні фактори, які можуть викликати зміни в нормальному поділі клітин в організмах. Приблизно 90% канцерогенів є мутагенами. Мутації соматичних клітин можуть викликати рак. Основна відмінність між мутагеном і канцерогеном полягає в тому, що мутаген викликає спадкові зміни в генетичній інформації організму, тоді як канцероген викликає або сприяє розвитку раку у тварин і людей. Мутагенез - це механізм, за допомогою якого відбувається зміна генетичного матеріалу, тоді як канцерогенез - це механізм, за допомогою якого відбувається утворення пухлин внаслідок мутагенних подій.

Коефіцієнт нахилу (SF) — це оцінка верхньої межі нахилу кривої «доза-відповідь» в області низьких доз для канцерогенів і використовується для оцінки збільшення протягом життя захворюваності на рак у людей від перорального або інгаляційного впливу. до дози канцерогенної хімічної речовини. В рамках HRA ризик раку для кожної хімічної речовини (CR або Incremental Lifetime Cancer Risk, ILCR) розраховується з використанням тривалого добового споживання (CDI, мг/кг-день) і коефіцієнта нахилу (SF, (мг/кг-день).) −1); SF забезпечує специфічну для хімікатів канцерогенну дію. З цими двома значеннями ризик раку (CR, безрозмірний) виходить шляхом їх множення, як у рівнянні: CR = CDI × SF.

Ми отримали значення для коефіцієнта нахилу ротової порожнини (OSF, (мг/кг-день)-1) та для ризику одиниці вдихання (IUR, мкг/м3) відповідно (рис. 1.12).

Відповідно до Агентства з охорони навколишнього середовища США (US EPA), дані IUR були перетворені на коефіцієнт нахилу вдиху (ISF) за формулою: ISF = (IUR ∗ BW ∗ CF)/IR, де IUR = ризик одиниці вдихання [(мкг/м3)-1], BW = середня маса тіла [70 кг], IR = швидкість вдихання [20 м3/день] і CF = коефіцієнт перерахунку [1000 мкг/мг].

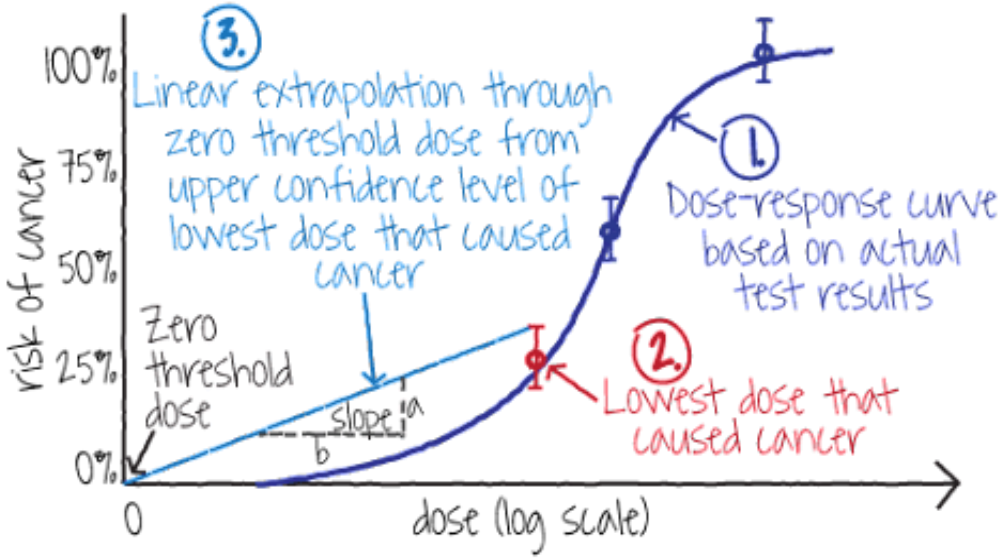

Рисунок 1.12 – Візуальне зображення канцерогенності [27]

4) Androgen Receptor Activity (Активність андрогенних рецепторів)

Рецептори андрогенів дозволяють організму відповідним чином реагувати на ці гормони. Рецептори присутні в багатьох тканинах організму, де вони приєднуються (зв'язуються) з андрогенами. Утворений андроген-рецепторний комплекс потім зв'язується з ДНК і регулює активність генів, що реагують на андроген.

Хімічні речовини, що порушують роботу ендокринної системи (EDC) — це ксенобіотики, які імітують взаємодію природних гормонів і змінюють синтез, транспорт або метаболічні шляхи. Перспектива того, що EDC спричиняють несприятливий вплив на здоров'я людей та дикої природи, призвело до розвитку наукових та нормативних підходів до оцінки біоактивності.

Дослідження HTS (high-throughput screening або високопродуктивний скринінг) in vitro. Дослідження HTS in vitro є швидшими та рентабельнішими, ніж традиційні тести на токсичність in vivo, і вони дозволяють уникнути етичних проблем, пов'язаних із випробуваннями на тваринах. Однак жодного окремого аналізу наразі не достатньо (через недостатню точність, цитотоксичність, проблеми з розчинністю тощо) для скринінгу всіх класів

хімічних речовин для даної молекулярної мішені та режиму активності (наприклад, агоніст або антагоніст) для точної оцінки потенціалу шкоди здоров'ю людей та навколишньому середовищу (рис. 1.13).

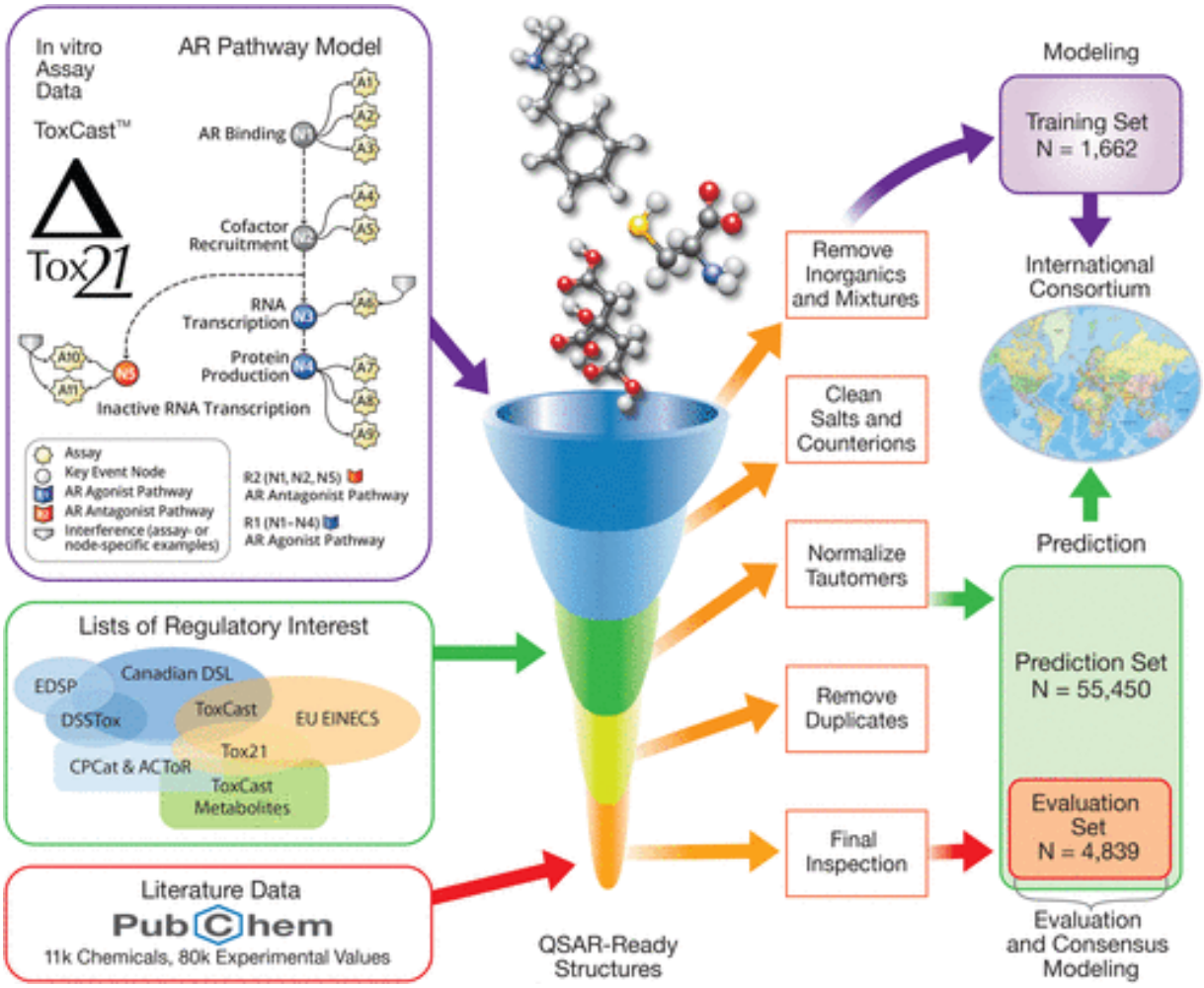

Рисунок 1.13 – Візуальне зображення активності андрогенних рецепторів [27]

5) Estrogen Receptor Activity (активність естрогенових рецепторів)

Естроген є стероїдним гормоном, який відіграє важливу роль у репродуктивному розвитку, гомеостазі кісток, серцево-судинному ремоделюванні та функціях мозку. Однак естроген також сприяє пухлинному утворенню молочних залоз, яєчників і ендометрія. Антагоністи естрогену та препарати, що знижують біосинтез естрогену, стали дуже успішними терапевтичними засобами для хворих на рак молочної залози. Вплив естрогену значною мірою опосередковується рецепторами естрогену (ER) α і ERβ, які є членами суперсімейства ядерних рецепторів факторів транскрипції.

Вплив EDC може призвести до несприятливих наслідків для здоров'я, пов'язаних з розвитком, неврологічної, репродуктивної, метаболічної, серцево-судинної та імунної систем у людей та дикої природи.

Рецептор естрогену (ER) є однією з найбільш детально вивчених мішеней, пов'язаних з ефектами EDC. Занепокоєння з приводу естрогеноподібної активності штучних хімічних речовин пояснюється їх потенціалом негативно впливати на репродуктивну функцію. Поява занепокоєння з приводу EDC призвело до введення правил, які вимагають оцінки хімічних речовин на естрогенну активність. Існують численні протоколи in vitro та in vivo для виявлення потенційних ефектів хімічних речовин, опосередкованих ендокринними шляхами, включаючи взаємодію з рецепторами гормонів (рис. 1.14).

HTS аналізи in vitro. Дані аналізу ToxCast™/Tox21 ER HTS були використані для розробки прогнозних обчислювальних моделей.

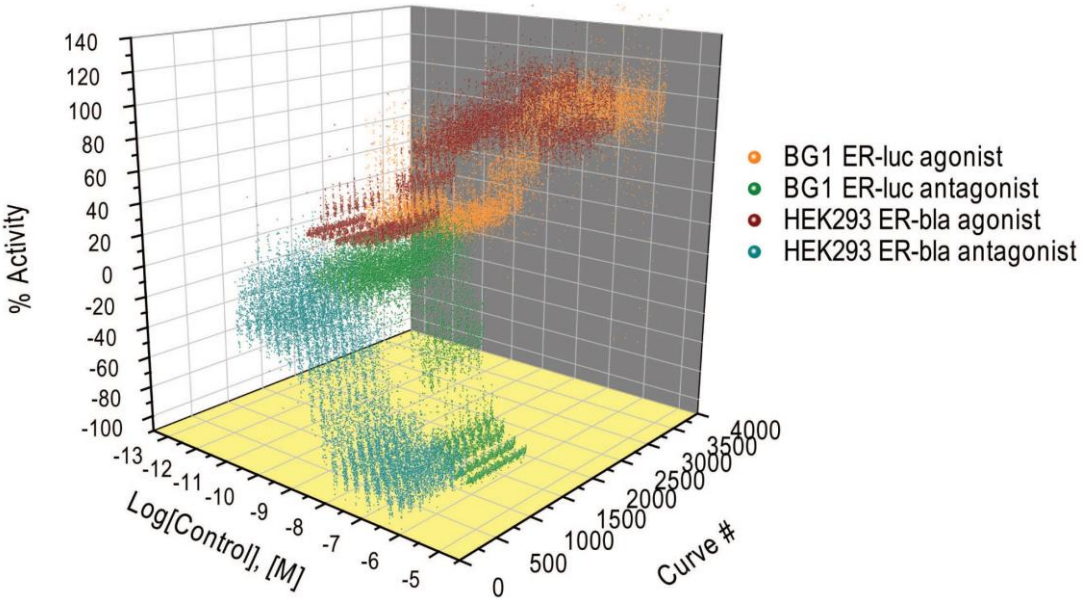

Рисунок 1.14 – Візуальне зображення активності естрогенових рецепторів [27]

6) Cardiotoxicity HERG (кардіотоксичність HERG)

Кардіотоксичність – це виникнення електрофізіологічної дисфункції серця або пошкодження м'язів. Лікарська кардіотоксичність є важливою причиною виснаження сполук у доклінічній та клінічній розробці. Він є одним

з найсерйозніших побічних ефектів, пов’язаних з розробкою нових ліків, і, як відомо, є одним із основних токсичних ефектів, викликаних кількома типами ліків.

Типовий механізм кардіотоксичності включає зв’язування сполук із серцевими калієвими каналами, що кодуються геном, пов’язаним з ефіром людини (hERG), що призводить до синдрому подовженого інтервалу QT (LQTS) і в кінцевому підсумку призводить до фатальних шлуночкових аритмій і раптова смерть.

Аналіз інгібування каналів hERG є високочутливим вимірюванням, яке визначить сполуки, що виявляють кардіотоксичність, пов’язану з інгібуванням hERG in vivo. Однак важливо зазначити, що не всі сполуки, які інгібують активність hERG in vitro, продовжують викликати кардіотоксичність in vivo.

Для нашої моделі активність IC50 в основному визначали на основі клітинних ліній ссавців, таких як HEK, CHO та COS. Дані IC50, отримані з лінії клітин не ссавців, XO (ооцити Xenopus laevis), були включені в набір даних, коли не було були доступні дані лінії клітин ссавців (рис. 1.15).

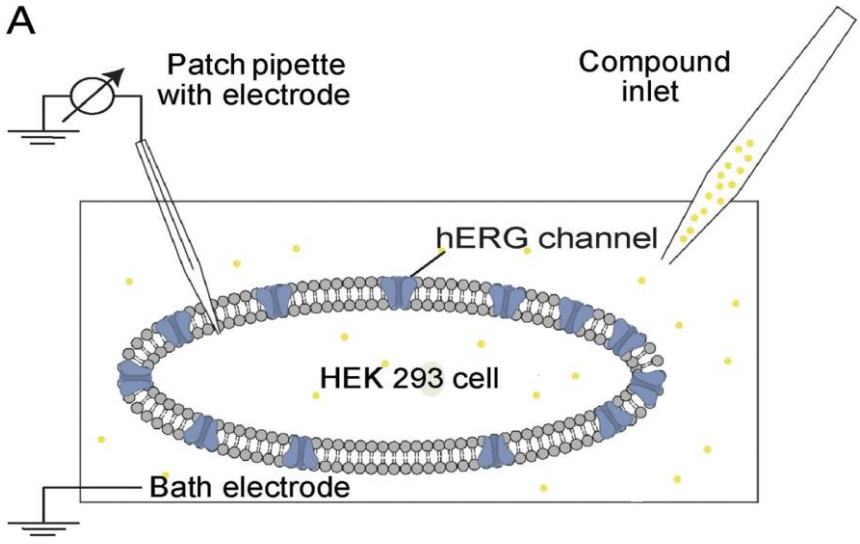

Рисунок 1.15 – Візуальне зображення кардіотоксичності HERG [27]

7) Hepatotoxicity DILI (гепатотоксичність DILI)

Гепатотоксичність визначається як пошкодження або пошкодження печінки, спричинене впливом ліків або інших нефармакологічних засобів.

Лікарська гепатотоксичність або медикаментозне ураження печінки (DILI) є гострою або хронічною реакцією на природну або виготовлену сполуку. Підводячи підсумок, можна сказати, що гепатотоксичність є основною причиною того, що ліки повертаються з ринку або не випускаються повністю.

Гепатотоксичність лікарських засобів є основною причиною вилучення ліків з фармацевтичного ринку та припинення розвитку нових молекул. Біомаркери корисні в кількох ситуаціях. У разі підозри на медикаментозне ураження печінки (ДПЗП), біомаркери можна використовувати для підтвердження ураження печінки, його тяжкості, прогнозу, підтвердження причинно-наслідкового зв'язку з прийомом препарату або визначення типу ХПЗ.

Традиційно в клінічній практиці біомаркери, що використовуються для виявлення ураження печінки, вимірюють зміну нормальної функції печінки, зміни цілісності тканин і клітин, які визначають за рівнем аланінамінотрансферази (АЛТ), аспартатамінотрансферази, лужної фосфатази (АЛФ) у сироватці крові. і гамма-глутамілтрансфераза. Корисність сироваткового білірубіну як діагностичного маркера залежить від його рівня. Коли загальний білірубін нижче 40 мкмоль/л без значення кон'югованого білірубіну, його можна сплутати з підвищенням некон'югованого білірубіну, як це спостерігається при хворобі Жильбера. Однак коли його значення значно вище цього рівня або коли підвищення пов'язане з підвищенням кон'югованого білірубіну, це є діагностичною ознакою ураження печінки. Тим не менш, використання білірубіну є скоріше критерієм тяжкості (рис. 1.16).

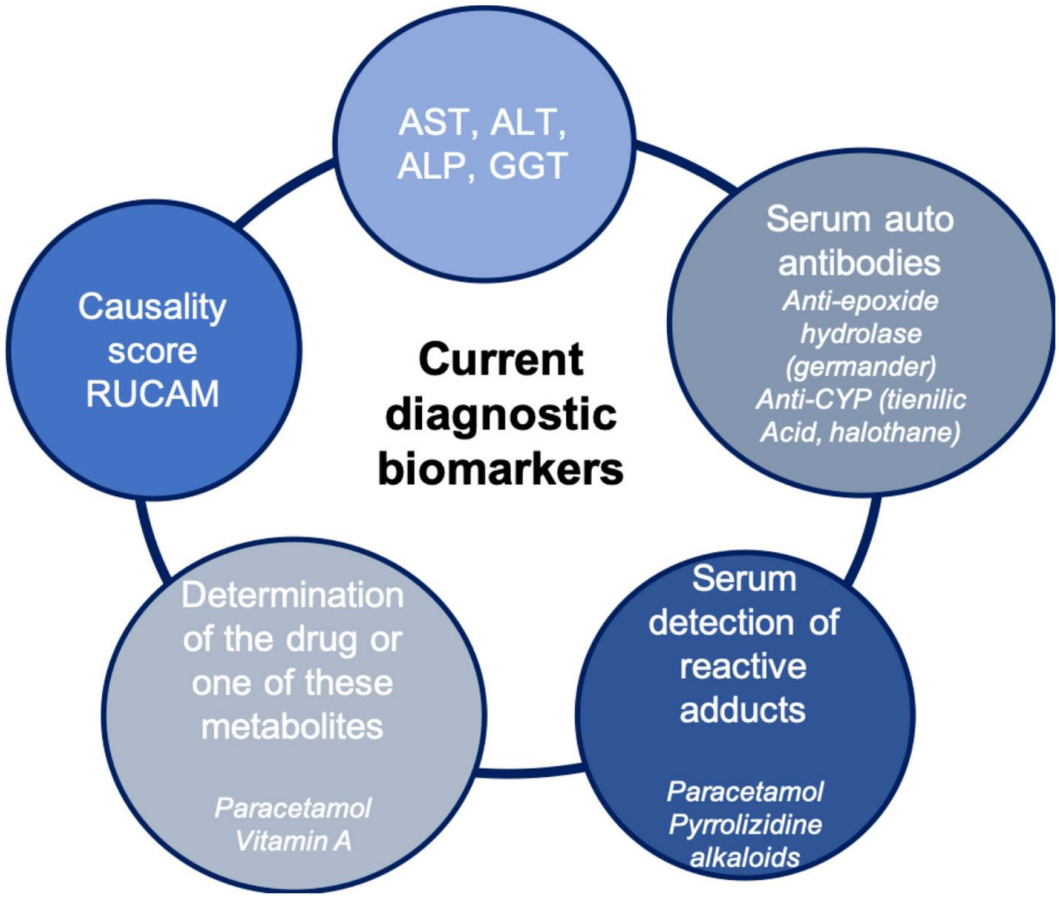

Рисунок 1.16 – Візуальне зображення гепатотоксичності DILI [27]

Висновки

Важливою частиною відкриття та розробки ліків є проведення досліджень DMPK (метаболізму та фармакокінетики лікарських засобів), які часто називають дослідженнями ADME-Tox (absorption – абсорбція, distribution – розподіл, metabolism – метаболізм, excretion – виділення, і toxicity – токсичність). Ці дослідження допомагають визначити життєздатність препарату-кандидата.

Підраховано, що близько 50% кандидатів на ліки не вдаються через неприйнятну ефективність і що до 40% кандидатів на ліки в минулому зазнали невдачі через токсичність. Такі ліки, як мібефрадил, сорувідин та фенілпропаноламін гідрохлорид, були зняті з ринку через взаємодію ліків або

токсичність. Як для регуляторів, так і для виробників лікарських засобів стало очевидним, що на додаток до фармакологічних властивостей дослідження ADME-Tox відіграють вирішальну роль в успіху кандидата на ліки. У зв'язку з цим впливом на кінцевий успіх ці дослідження зараз проводяться на початку процесу відкриття ліків.

Дослідження in vitro і in vivo проводяться, щоб дозволити розробнику ліків прийняти рішення про те, чи слід вибирати лікарський засіб як кандидат на лікування, і перейти до пізніх доклінічних та клінічних програм. Властивості ADME дозволяють розробникам ліків зрозуміти безпеку та ефективність препарату-кандидата, і вони необхідні для схвалення регулятором.

РОЗДІЛ 2 ТЕОРЕТИЧНІ ВІДОМОСТІ ПРО ГРАФОВІ НЕЙРОННІ МЕРЕЖІ ТА ПІДХОДИ ДО ПОБУДОВИ БАГАТОЗАДАЧНОЇ МОДЕЛІ

2.1 Графові нейронні мережі

Перед тим, як приступати до огляду методів та моделей графових нейронних мереж, необхідно розглянути саме поняття “граф”, його властивості й способи зміни признаків (фіч) вершин.

Нехай наш граф не містить жодних зв’язків (ребер, еджів), а містить лише множину вершин υ. Також нехай $x_i \in \mathbb{R}^k$ – це фічі вершини i. Ми можемо сконкатенувати їх разом й отримати матрицю фіч вершин даного графа розміром n x k:

$$X = (x_1, \dots, x_n)^T$$

Важливо розуміти, що, пронумерувавши вершини якось по-іншому, дана матриця фіч вершин також буде іншою, тож ми маємо зробити так, щоб результат нашої нейронної мережі не залежав від перестановки вершин.

Для цього введемо поняття інваріантності та еквариантності функції відносно перестановок.

Функція $f(X)$ називається інварінтною відносно перестановок для всіх матриць перестановок P, якщо

$$f(PX) = f(X)$$

Один звичний приклад – це Deep Sets модель [38]:

$$f(X) = \phi\left(\sum_{i\epsilon\upsilon} \psi(x_i)\right),$$

де ψ та φ – це функції, які навчаються (наприклад, MLP – багатошарові перцептрони), а агрегатор сумації може бути замінений на агрегатор максимуму, середнього тощо.

Функція $f(X)$ називається еквіваріантною відносно перестановок для всіх матриць перестановок P, якщо:

$$f(PX) = Pf(X)$$

Таким чином ми можемо ввести еквіваріантну функцію, яка буде обробляти фічі кожної вершини й видавати нові фічі незалежно від порядку розташування самих вершин у графі:

$$h_i = \psi(x_i)$$

І так само ми можемо ввести інваріантну функцію, яка оброблятиме еквіваріантні функції й таким чином видаватиме нам результат про весь граф у вигляді якогось вектора чисел, а не матриці вершин фіч:

$$f(X) = \phi\left(\sum_{i \epsilon v} \psi(x_i)\right),$$

де ψ – еквіваріантна функція, ϕ – інваріантна функція, а агрегатор сумації може бути замінений на агрегатор максимуму, середнього тощо.

Тепер розглянемо граф, який містить множину вершин, а також ребра між цими вершинами, де $\mathrm{E} \subseteq VxV$. Ми можемо представити ці ребра за допомогою матриці суміжності так, що

$$a_{ij} = \begin{cases} 1, (i,j)\ \epsilon\ \mathrm{E} \\ 0, \text{інакше} \end{cases}$$

Усі властивості інварінтності й еквіваріантності функцій залишаються, проте тепер для ребер вони набудуть трохи іншого вигляду.

Інваріантність:

$$f(PX, PAP^T) = f(X, A)$$

Еквіваріантність:

$$f(PX, PAP^T) = Pf(X, A)$$

Для графів важливо також враховувати контекст кожної вершини, тобто її сусідів. Для вершини i, 1-рівневі сусіди визначаються як:

$$\mathrm{N}_i = \{j : (i,j)\ \epsilon\ E\ \vee (j,i)\ \epsilon\ E\}$$

Звідси випливає, що ми можемо ввести матрицю всіх фіч вершин, які є сусідами конкретної вершини:

$$X_{N_i} = \left\{\{x_j : j\ \epsilon\ N_i\}\right\}$$

Тепер ми можемо ввести еквіваріантну функцію відносно перестановок *f(X, A)*, просто застосувавши локальну функцію g для всіх сусідств вершин:

$$f(X,A) = \begin{bmatrix} - g(x_1, X_{N_1}) - \\ - g(x_2, X_{N_2}) - \\ \vdots \\ - g(x_n, X_{N_n}) - \end{bmatrix}$$

Щоб отримати еквіваріантність функції *f*, функція g не має залежати від порядку розташування вершин у X_{N_i}, а тому g має бути інваріантною відносно перестановок.

На рисунку 2.1 зображено детальніше, що саме робить функція g. Тобто вона для кожної вершини, враховуючи фічі самої вершини й усіх її сусідів модифікує фічі цієї вершини. Це в точності те ж саме, що робить один шар будь-якої графової нейронної мережі.

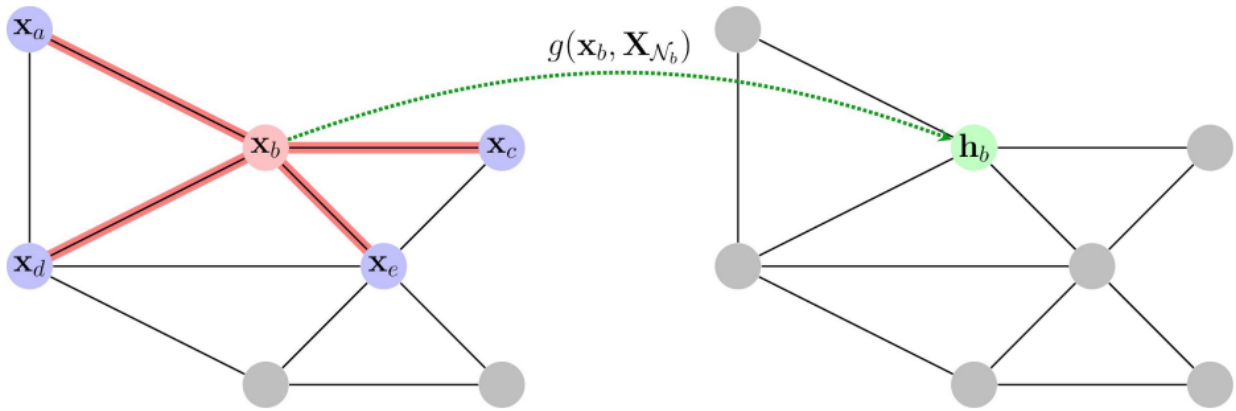

Рисунок 2.1 – Візуалізація роботи функції g [34]

Таким чином, використовуючи одну й ту саму функцію *f*, яку ми визначили вище, можна вирішувати різні задачі, що стосуються графів, як-от класифікація вершин (якщо в нас є достатньо великий граф і по ньому потрібно навчитися визначати, до якого класу належатиме кожна вершина, якщо розмічених вершин у нас не так багато, наприклад, передбачення того, чи буде людина ходити на секцію футболу, маючи граф, де кожна вершина – це людина, а ребра – дружні зв'язки між цими людьми), графова класифікація (якщо в нас є багато подібних за структурою графів, і ми маємо класифікувати кожен граф, наприклад, якраз наша задача в цій роботі – передбачення якогось

параметру для молекулярних задач, маючи різні графи, де кожен граф – це окрема молекула із атомами як вершини й зв'язками між цими атомами як ребра), передбачення зв'язків (трохи подібна проблема до класифікації вершин у тому плані, що в нас є великий граф, для якого потрібно передбачати тепер не приналежність кожної вершини до якогось класу, а передбачати можливі пропущені зв'язки між вершинами цього графа, наприклад, можливі зв'язки між ліками і їх таргетом у великому графі знань із біологічними даними). На рисунку 2.2 зображено формульну візуалізацію цих трьох проблем.

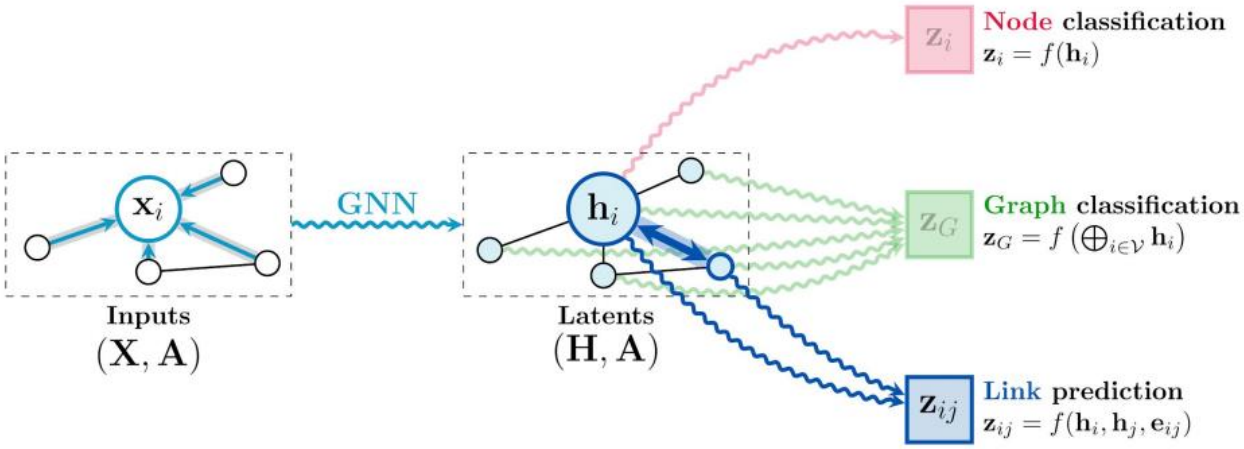

Рисунок 2.2 – Різновидності задач із графами [34]

Взагалі, існує 3 основних підходи для реалізації функції g, яку ми визначили вище. Це графові згорткові (convolutional) та уважні (attentional) нейронні мережі, а також нейронні мережі із передачею повідомлень (message-passing). Саме ці підходи ми й розберемо в наступних підрозділах, а також наведемо детальніший спосіб побудови графової нейронної мережі для біологічних задач.

2.1.1 Графові згорткові нейронні мережі

Для згорткових нейронних мереж фічі для сусідів вершин агрегуються за допомогою константних ваг c_{ij}:

$$h_i = \phi\left(x_i, \sum_{i \epsilon N_i} c_{ij}\psi(x_j)\right)$$

Зазвичай такі ваги прораховуються на основі даних всього графа (і не використовують фічі вершин як такі). На рисунку 2.3 зображено візуалізацію такого підходу.

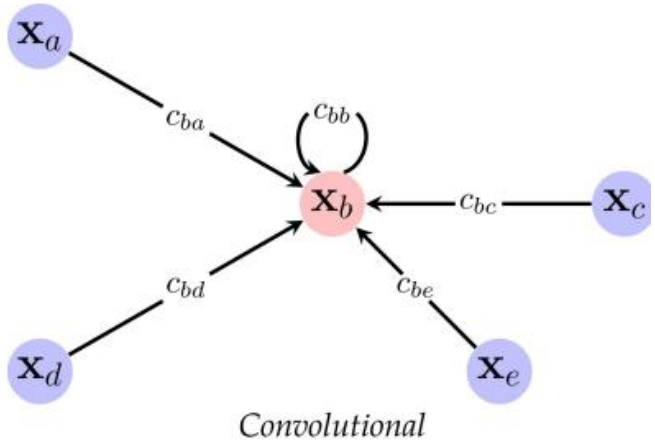

Рисунок 2.3 – Візуалізація згорткового підходу [34]

Такі архітектури доцільно використовувати, коли в нас є гомофілійні графи, тобто такі, подібні вершини яких зв'язуються ребрами із набагато більшою ймовірністю, ніж неподібні. Однією із найбільш переваг цього підходу – швидкість обчислень, оскільки ваги обраховуються лише раз і є константними скалярними значеннями.

Однією із найпоширеніших архітектур цього підходу є власне GCN, у назві якої якраз є слово convolutional, тобто згортковий [17]. Ця стаття також є однією із перших у сфері побудови графових нейронних мереж. Зміна фіч кожної вершини відбувається за наступною формулою:

$$h_i = \Theta^T \sum_{j \epsilon N_i} \frac{e_{j,i}}{\sqrt{\widehat{d}_j \widehat{d}_i}} x_j,$$

де Θ^T – функція, яка навчається, $\widehat{d}_i = 1 + \sum_{j \epsilon N_i} e_{j,i}$, де $e_{j,i}$ означає вагу серцевої вершини j до таргет-вершини i (1 за замовчуванням).

Також є архітектури ChebyNet та SGC, проте в даній роботі ми їх розглядати не будемо.

2.1.2 Графові уважні нейронні мережі

Для уважних нейронних мереж фічі для сусідів вершин агрегуються за допомогою неявних ваг $a_{ij} = a(x_i, x_j)$:

$$h_i = \phi\left(x_i, \sum_{i \epsilon N_i} a(x_i, x_j)\psi(x_j)\right)$$

На відміну від графових згорткових нейронних мереж, такі мережі вже використовують фічі вершин, проте все ще в неявному вигляді. На рисунку 2.4 зображено візуалізацію такого підходу.

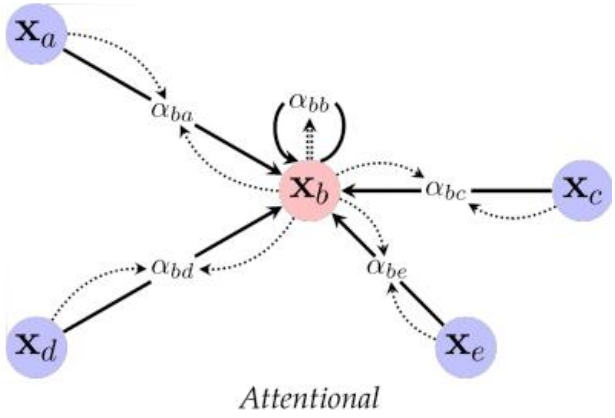

Рисунок 2.4 – Візуалізація уважного підходу [34]

Такі архітектури доцільно використовувати, якщо ребра не обов'язково мають представляти гомофілію, тобто подібність вершин. Також цей підхід все одно підраховує скалярні величини для кожного ребра, що позитивно впливає на обчислювальну швидкість.

Однією із найпоширеніших архітектур цього підходу є власне GAT, у назві якої якраз є слово attention, тобто уважний [33]. Зміна фіч кожної вершини відбувається за наступною формулою:

$$h_i = a_{i,i}\Theta x_i + \sum_{j \epsilon N_i} a_{i,j}\Theta x_j\,,$$

де уважні коефіцієнти $a_{i,j}$ обчислюються за формулою:

$$a_{i,j} = \frac{\exp\left(LeakyRELU(a^T[\Theta x_i || \Theta x_j])\right)}{\sum_{k \epsilon N_i} \exp\left(LeakyRELU(a^T[\Theta x_i || \Theta x_k])\right)}$$

Також є модифікація даного методу, яка має назву GATv2 [1] і містить дещо модифіковані уважні коефіцієнти:

$$a_{i,j} = \frac{\exp\left(a^T LeakyRELU(\Theta[x_i||x_j])\right)}{\sum_{k\epsilon N_i} \exp\left(a^T LeakyRELU(\Theta[x_i||x_k])\right)}$$

2.1.3 Графові нейронні мережі із передачею повідомлень

Для графових нейронних мереж із передачею повідомлень фічі для сусідів вершин агрегуються за допомогою "повідомлень" $m_{ij} = \psi(x_i, x_j)$:

$$h_i = \phi\left(x_i, \sum_{i\epsilon N_i} \psi(x_i, x_j)\right)$$

Цей підхід використовує всю інформацію про таргет-вершину та вершину-сусіда й динамічно обчислює необхідні ваги. На рисунку 2.5 зображено візуалізацію такого підходу.

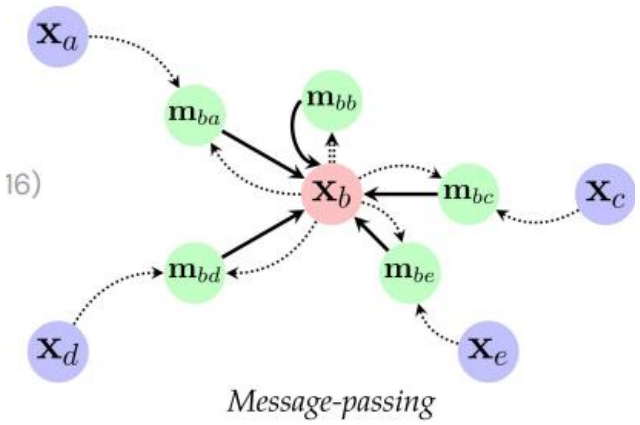

Рисунок 2.5 – Візуалізація підходу із передачею повідомлень [34]

Такі архітектури можуть відчувати проблеми із швидкістю тренування, оскільки динамічно обчислюють інформацію про кожне ребро. Проте саме такі архітектури ідеально використовувати для біолого-хімічних задач (саме така, як у нас), а також для причинних (reasoning) задач та задач симуляції.

У даній роботі ми розглянемо дві архітектури-представника цього підходу: GIN [36] та GINE [13].

У GIN зміна фіч кожної вершини відбувається за наступною формулою:

$$h_i = \theta\left(x_i + \sum_{j \epsilon N_i} x_j\right),$$

де θ – це нейронна мережа, наприклад MLP – багатошаровий перцептрон.

У GINE зміна фіч кожної вершини відбувається за наступною формулою:

$$h_i = \theta\left(x_i + \sum_{j \epsilon N_i} ReLU(x_j + e_{j,i})\right),$$

де θ – це нейронна мережа, наприклад MLP – багатошаровий перцептрон, а $e_{j,i}$ – це ребреві фічі конкретного зв'язку між вершинами.

Таким чином маємо, що GINE – трішки модифікована версія GIN, яка також бере до уваги й ребреві фічі, а не лише фічі вершин.

2.1.4 Графові нейронні мережі для молекулярних задач

Розглянемо один із графових підходів для переведння молекул із графового простору в числовий вектор признаків – метод MF (Molecular Fingerprints).

Нещодавні роботи з проектування матеріалів використовували нейронні мережі для прогнозування властивостей нових молекул шляхом узагальнення на прикладах. Одними із найбільших труднощів із цим завданням є те, що вхідні дані для предиктора, у даному випадку молекули, можуть мати довільний розмір і форму. Наразі більшість пайплайнів машинного навчання можуть лише обробляти входи фіксованого розміру. Нинішній "state of the art" полягає у використанні стандартного програмного забезпечення для фінгерпринтів, щоб обчислювати вектори ознак фіксованого розміру та використовувати ці ознаки як вхідні дані для багатозв'язної глибокої нейронної мережі або іншого стандартного методу машинного навчання. Цього шаблону дотримувався [39, 14, 6]. Під час навчання молекулярні вектори фігнгерпринтів розглядалися як фіксовані.

У підході MF замінюється нижній шар цього стека – функцією, яка обчислює молекулярні вектори фінгерпринтів – з диференційованою нейронною мережею, вхідним сигналом якої є граф, що представляє оригінальна молекула. У цьому графі вершини представляють окремі атоми, а ребра — зв’язки. Нижні шари цієї мережі є згортковими в тому сенсі, що до кожного атома та його сусідів застосовується один і той же локальний фільтр. Після кількох таких шарів крок глобального пулінгу об’єднує признаки з усіма атомами в молекулі.

Ці нейронномережеві графові фінгерпринти мають ряд переваг над фіксованими фінгерпринтами.

Прогнозна здатність. Використовуючи дані, адаптовані до поставленої задачі, оптимізовані для машини фінгерпринти можуть забезпечити значно кращу прогнозну здатність, ніж фіксовані. Ми показуємо, що фінгерпринти нейронного графа відповідають або перевершують прогнозну здатність стандартних фінгепринтів щодо розчинності, ефективності ліків та органічної фотоелектричної ефективності.

Ощадливість. Фіксовані фінгепринти повинні бути надзвичайно великими, щоб закодувати всі можливі підструктури без перекриття. Наприклад, [39] використав вектор фінгерпринтів розміром 43000 після видалення признаків, які рідко зустрічалися. Диференційовані фінгерпринти можна оптимізувати для кодування тільки релевантних признаків, зменшуючи вимоги до обчислень та регуляризації.

Інтерпретація. Стандартні фінгерпринти кодують кожен можливий фрагмент повністю чітко, без поняття подібності між фрагментами. Навпаки, кожен признак нейронномережевого графового фінгерпринта може бути активований подібними, але різними молекулярними фрагментами, що робить представлення ознак більш змістовним.

Сучасні молекулярні фінгерпринти – це кругові фінгерпринти з розширеною зв’язністю (ECFP) [29]. Кругові фінгерпринти [8] є удосконаленням алгоритму Моргана [24], розробленого щоб закодувати, які

підструктури присутні в молекулі, способом, який є інваріантним до перемічення атомів.

Кругові фінгерпринти генерують особливості кожного шару, застосовуючи фіксовану хеш-функцію до об’єднаних признаків сусідів на попередньому шарі. Результати цих хешів потім обробляються як цілі індекси, де 1 записується у вектор фінгерпринта за індексом, заданим вектором признаків у кожному вузлі графа. На рисунку 2.6 (ліворуч) показано ескіз цієї обчислювальної архітектури. Ігноруючи колізії, кожен індекс фінгерпринта означає наявність певної підструктури. Розмір підструктур, представлених кожним індексом, залежить від глибини мережі. Таким чином кількість шарів називається «радіусом» фінгерпринта.

Кругові фінгерпринти аналогічні згортковим мережам тим, що застосовують скрізь ту саму операцію локально та об’єднують інформацію в кроці глобального пулінгу.

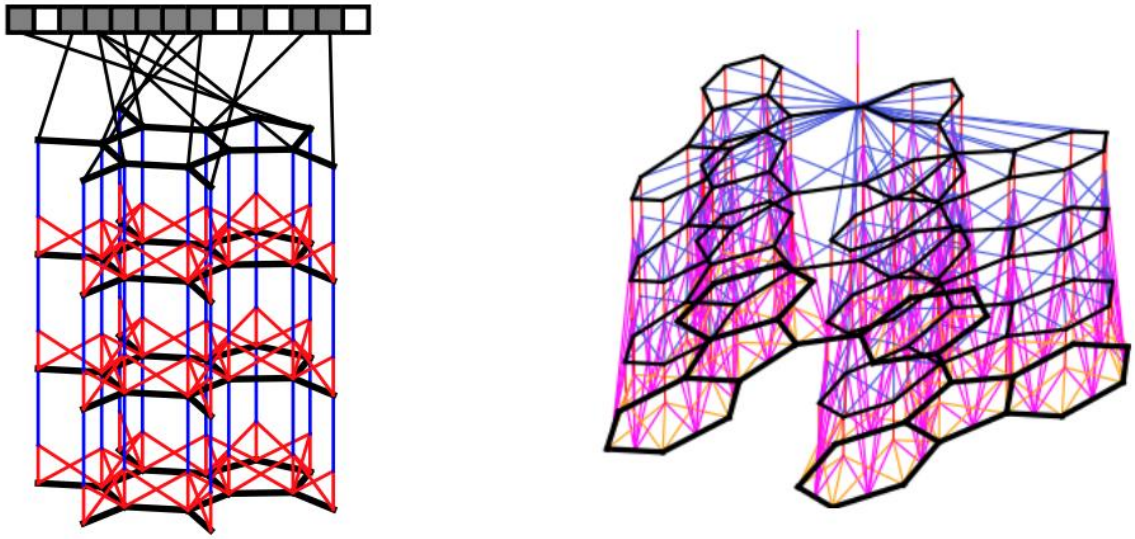

Рисунок 2.6 – Ліворуч: візуальне представлення обчислювального графа як стандартних кругових фінгерпринтів, так і нейронних фінгерпринтів. Праворуч: більш детальний ескіз, включаючи інформацію зв’язків, що використовується в кожній операції [6]

Простір можливих мережевих архітектур великий. Відштовхуючись від добре відомої конфігурації, ми розробили диференційоване узагальнення

кругових фінгерпринтів. Цей розділ описує нашу заміну кожної дискретної операції в кругових фінгерпринтах на диференційований аналог.

Хешування. Метою хеш-функцій, що застосовуються до кожного шару кругових фінгерпринтів, є об'єднання інформації про кожен атом і сусідні з ним підструктури. Це гарантує, що будь-яка зміна фрагмента, незалежно від його розміру, призведе до активації іншого індексу фінгерпринта. Ми замінюємо хеш-операцію на один шар нейронної мережі. Використання гладкої функції дозволяє активаціям бути подібними, коли локальна молекулярна структура змінюється в неважливий нам спосіб.

Індексування. Кругові фінгерпринти використовують операцію індексування для об’єднання векторів ознак усіх вузлів в один фінгерпринт всієї молекули. Кожен вузол встановлює один біт відбитка пальця в 1, за індексом, що визначається хешем його вектора ознак. Ця операція, подібна до об’єднання, перетворює граф довільного розміру у вектор фіксованого розміру. Для малих молекул і великої довжини фінгерпринтів, фінгерпринти завжди рідкісні. Ми використовуємо операцію softmax як диференційований аналог індексації. По суті, кожному атому пропонується віднести себе до однієї із категорій. Сума з усіх цих векторів класифікації міток створює остаточний фінгерпринт. Ця операція аналогічна операції об'єднання в стандартних згорткових нейронних мережах.

Каноніалізація. Кругові фінгерпринти ідентичні незалежно від впорядкування атомів у кожному сусідстві. Ця інваріантність досягається шляхом сортування сусідніх атомів за їх особливостями та особливостями зв'язку. Ми експериментували з цією схемою сортування, а також із застосуванням локальних перетворень ознак на всіх можливих перестановках локального сусідства. Альтернатива каноніалізації полягає в застосуванні інваріантної перестановки функції, наприклад підсумовування. В інтересах простоти й масштабованості, ми обрали підсумовування.

Кругові фінгерпринти можна інтерпретувати як окремий випадок фінгерпринтів нейронного графа, які мають великі випадкові ваги. Це

пояснюється тим, що в межах великих вхідних ваг нелінійності tanh наближається кроковими функціями, які при з'єднанні утворюють просту хеш-функцію. Крім того, на межі великих вхідних ваг, softmax наближається до one-hot-coded argmax, що є аналогом операції індексування.

На рисунку 2.7 алгоритми 1 і 2 підсумовують ці два підходи та виділяють їх відмінності. Враховуючи довжину фінгерпринтів L і F на кожному шарі, параметри фінгерпринтів нейронного графа складаються з окремої вихідної матриці вагів розміром F × L для кожного шару, а також набору hidden-to-hidden вагової матриці розміру F × F на кожному шарі, по одній на кожне можливе число зв'язків, яке може мати атом (до 5 в органічних молекулах).

Algorithm 1 Circular fingerprints

Input: molecule, radius R, fingerprint length S
Initialize: fingerprint vector $\mathbf{f} \leftarrow \mathbf{0}_S$
for each atom a in molecule
  $\mathbf{r}_a \leftarrow g(a)$ ▷ lookup atom features
for $L = 1$ to R ▷ for each layer
  for each atom a in molecule
    $\mathbf{r}_1 \dots \mathbf{r}_N = \text{neighbors}(a)$
    $\mathbf{v} \leftarrow [\mathbf{r}_a, \mathbf{r}_1, \dots, \mathbf{r}_N]$ ▷ concatenate
    $\mathbf{r}_a \leftarrow \text{hash}(\mathbf{v})$ ▷ hash function
    $i \leftarrow \text{mod}(r_a, S)$ ▷ convert to index
    $\mathbf{f}_i \leftarrow 1$ ▷ Write 1 at index
Return: binary vector $\mathbf{f}$

Algorithm 2 Neural graph fingerprints

Input: molecule, radius R, hidden weights $H_1^1 \dots H_R^5$, output weights $W_1 \dots W_R$
Initialize: fingerprint vector $\mathbf{f} \leftarrow \mathbf{0}_S$
for each atom a in molecule
  $\mathbf{r}_a \leftarrow g(a)$ ▷ lookup atom features
for $L = 1$ to R ▷ for each layer
  for each atom a in molecule
    $\mathbf{r}_1 \dots \mathbf{r}_N = \text{neighbors}(a)$
    $\mathbf{v} \leftarrow \mathbf{r}_a + \sum_{i=1}^{N} \mathbf{r}_i$ ▷ sum
    $\mathbf{r}_a \leftarrow \sigma(\mathbf{v} H_L^N)$ ▷ smooth function
    $\mathbf{i} \leftarrow \text{softmax}(\mathbf{r}_a W_L)$ ▷ sparsify
    $\mathbf{f} \leftarrow \mathbf{f} + \mathbf{i}$ ▷ add to fingerprint
Return: real-valued vector $\mathbf{f}$

Рисунок 2.7 – Псевдокод кругових фінгерпринтів (ліворуч) і нейронних фінгерпринтів (праворуч). Відмінності виділені синім кольором. Кожна недиференційована операція замінюється диференційованим аналогом

Було проведено два експерименти, щоб продемонструвати, що нейронні фінгерпринти з великими випадковими вагами поводяться так само, як кругові. Спочатку ми перевірили, чи відстані між круговими фінгерпринтами подібні до відстаней на основі нейронних фінгерпринтів. На рисунку 2.8 (ліворуч) показана діаграма розсіювання попарних відстаней між круговими та нейронними фінгерпринтами. Фінгерпринти мали довжину 2048, і були розраховані на парах молекул із набору даних про розчинність [18]. Відстань

вимірювалася, використовуючи безперервне узагальнення міри подібності Танімото (або відповідний коефіцієнт Жаккара), заданої за формулою:

$$\text{distance}(\mathbf{x}, \mathbf{y}) = 1 - \sum \min(x_i, y_i) \Big/ \sum \max(x_i, y_i)$$

Між відстанями існує кореляція r = 0,823. Лінія точок праворуч від графіка показує, що для деяких пар молекул бінарні відбитки ECFP мають рівно нульове перекриття.

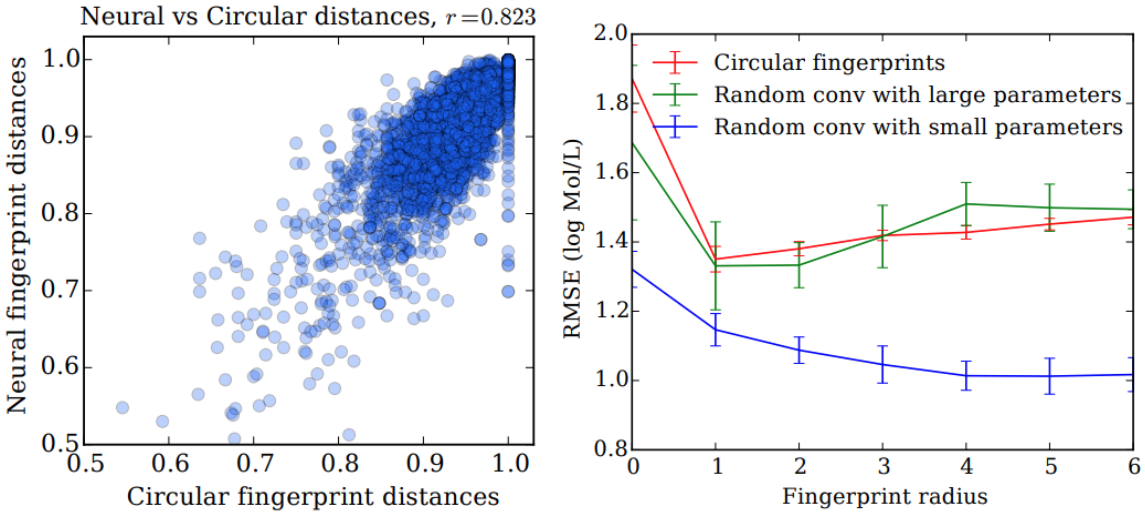

Рисунок 2.8 – Ліворуч: порівняння попарних відстаней між молекулами, виміряних за допомогою кругових і нейронних фінгерпринтів із великими випадковими вагами. Праворуч: прогнозна здатність кругових (червоний графік), нейронних із фіксованими великими випадковими вагами (зелений графік) та нейронних із фіксованими малими випадковими вагами (синій графік) фінгерпринтами. Продуктивність нейронних фінгерпринтів із великими випадковими вагами майже відповідає показникам кругових фінгерпринтів [6]

По-друге, було досліджено прогнозну здатність нейронних фінгерпринтів із великими випадковими ваговими вагами порівняно із круговими. Рисунок 2.8 (праворуч) показує середню продуктивність прогнозування для набору даних розчинності з використанням лінійної регресії поверх фінгерпринтів. Ефективність обох методів унаслідується подібними кривими. На відміну від них, продуктивність нейронних фінгерпринтів із невеликими випадковими

ваговими показниками йде за іншою кривою і є значно кращою. Це говорить про те, що навіть із випадковими вагами, відносно плавна активація нейронних фінгерпринтів допомагає продуктивності узагальнення.

Щоб продемонструвати, що фінгерпринти нейронного графа можна інтерпретувати, ми показуємо підструктури, які найбільше активують окремі функції у векторі фінгерпринтів. Кожен признак кругового вектора фінгерпринтів може бути активований лише одним фрагментом одного радіуса, за винятком випадкових зіткнень. На відміну від цього, признаки фінгерпринтів нейронного графа можуть бути активовані варіаціями тієї самої структури, роблячи їх більш інтерпретованими та дозволяючи коротші вектори ознак.

Характеристики розчинності. На рисунку 2.9 показані фрагменти, які максимально активують найбільш передбачувані признаки фінгерпринтів. Мережа фінгерпринтів була навчена як вхідні дані для лінійної моделі, що прогнозує розчинність, як виміряно в [5]. Признак, показаний у верхньому рядку, має позитивний прогнозний зв'язок з розчинністю і найбільше активується фрагментами, що містять гідрофільну групу R-OH, стандартний показник розчинності. Признак, показаний в нижньому рядку, що сильно вказує на нерозчинність, активується неполярними повторюваними кільцевими структурами.

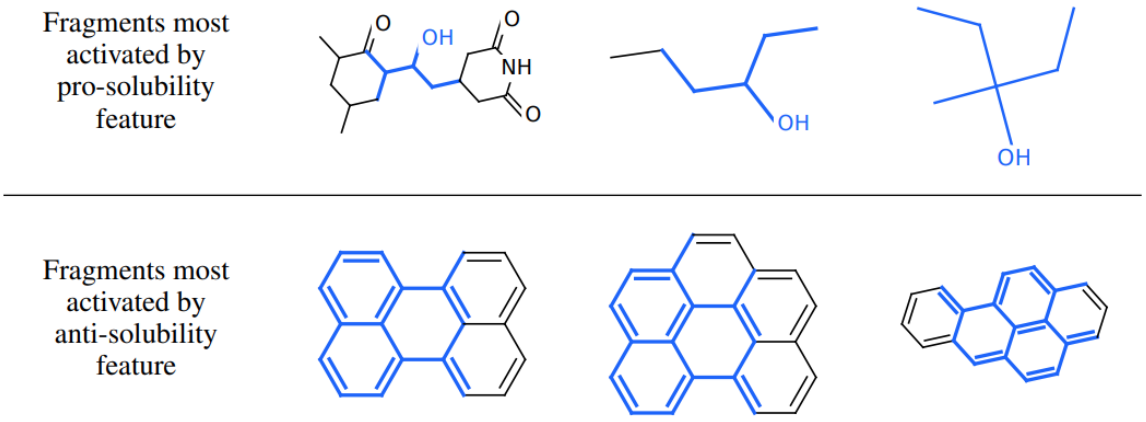

Рисунок 2.9 – Дослідження відбитків пальців оптимізовано для прогнозування розчинності. Показані тут репрезентативні приклади молекулярних фрагментів (виділені синім кольором), які найбільше активують різні ознаки відбиток

пальця. Верхній рядок: признак, який найбільше прогнозує розчинність. Нижній рядок: признак, який найбільше прогнозує нерозчинність [6]

Характеристики токсичності. Було навчено ту саму архітектуру моделі для прогнозування токсичності, виміряної в двох різних наборах даних. На рисунку 2.10 показані фрагменти, які найбільше активують функцію прогнозу токсичності, у двох окремих наборах даних.

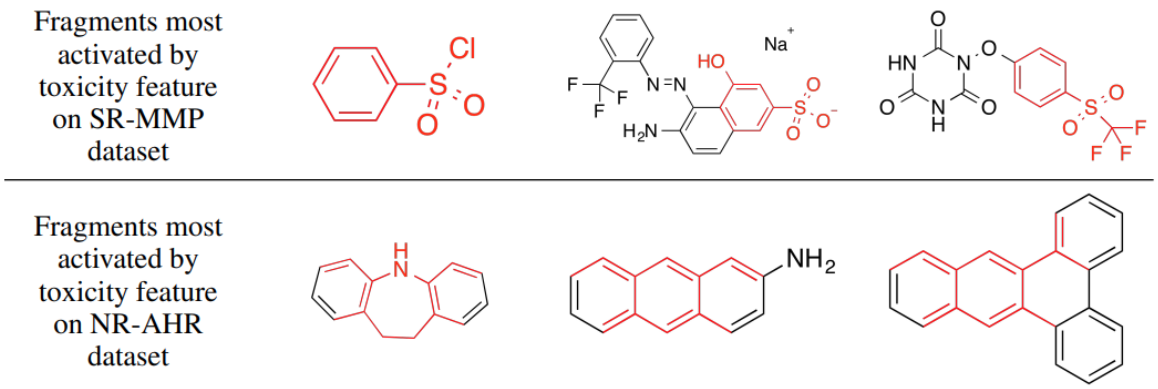

Рисунок 2.10 – Візуалізація фінгерпринтів оптимізована для прогнозування токсичності. Показані тут репрезентативні зразки молекулярних фрагментів (виділені червоним), які найбільше активують ознаку, яка найбільше прогностична токсичності. Верхній рядок: найбільш передбачувана функція визначає групи, що містять приєднаний атом сірки до ароматичного кільця. Нижній ряд: найбільш передбачуваною ознакою є також злиті ароматичні кільця відомі як поліциклічні ароматичні вуглеводні, добре відомий канцероген. [6]

[32] побудували подібні візуалізації, але напівручним способом: щоб визначити, які токсичні фрагменти активували даний нейрон, вони переглянули складений вручну список токсичних підструктур і вибрали ту, яка найбільше корелює з даним нейроном. Навпаки, наші візуалізації генеруються автоматично, без необхідності обмежувати діапазон можливих відповідей заздалегідь.

Було проведено кілька експериментів, щоб порівняти прогнозну продуктивність нейронних фінгерпринтів до стандартної “state-of-the-art”

конфігурації: кругові відбитки пальців подаються в повнозв’язну нейронну мережу.

Експериментальна установка. Пайплайн приймає в якості вхідних даних кодування рядка SMILES для кожної молекули, яка потім перетворюється на графік за допомогою RDKit [26]. Ми також використовували RDKit для генерування розширених круглі фінгерпринтів, використаних в бейзлайні. Атоми водню розглядалися неявно.

У згорткових мережах початкові характеристики атома та зв’язку були обрані подібними до ECFP: початкові характеристики атома об’єднували кодування елемента атома, його ступінь, кількість приєднаних атомів водню, неявну валентність і показник ароматичності. Характеристики зв’язку полягали в тому, чи був тип зв’язку одинарним, подвійним, потрійним, або ароматичний, чи був зв'язок спряженим, і чи був зв'язок частиною кільця.

Навчання та архітектура. Під час навчання використано батч нормалізацію [14]. Також експериментували з функціями активації tanh і relu як для мережевих шарів нейронних фінгерпринтів, так і для повністю підключених мережевих шарів. relu мала невелику, але стабільну перевагу в продуктивності на валідаційному наборі. Також експериментували з dropconnect, варіантом дропауту, у якому ваги випадковим чином зануляються замість прихованих одиниць, але виявили, що це призвело до гіршої валідаційної помилки в цілому. Кожен експеримент оптимізовано для 10000 мінібатчів розміром 100 з використанням алгоритму Адама [16], варіант RMSprop, що включає імпульс.

Оптимізація гіперпараметрів. Для оптимізації гіперпараметрів ми використовували випадковий пошук. Гіперпараметри всіх методів були оптимізовані за допомогою 50 випробувань для кожного крос-валідаційного фолду. Оптимізовано такі гіперпараметри: логарифм швидкості навчання, логарифм початкової вагової шкали, логарифм L2-penalty, довжина фінгерпринта, c (до 6) і розмір прихованого шару в повнозв’язній мережі.

Датасети. Було порівняно продуктивність стандартних кругових фінгерпринтів із фінгерпринтами нейронного графа в різних доменах.

Розчинність: розчинність у воді 1144 молекул, виміряна за [5].

Ефективність препарату: напівмаксимальна ефективна концентрація (EC50) in vitro 10000 молекул проти сульфідорезистентного штаму P. falciparum, паразита, що викликає малярію, як виміряно за [7].

Органічна фотоелектрична ефективність: Гарвардський проект чистої енергії [10] використовує дорогі моделювання DFT для оцінки фотоелектричної ефективності органічних молекул. Ми використовували підмножину із 20000 молекул із цього набору даних.

Точність прогнозування. Було порівняно продуктивність кругових фінгерпринтів і нейронних фінгерпринтів за двох умов: у першій умові передбачення було зроблено лінійним шаром за допомогою фінгерпринтів як вхідних даних. У другій умові передбачення були зроблені одним прихованим шаром нейронної мережі з використанням фінгерпринтів як вхідних даних. У всіх налаштуваннях усі диференційовані параметри були оптимізовані одночасно. Результати наведені на рисунку 2.11.

Dataset Units	Solubility [4] log Mol/L	Drug efficacy [5] EC_{50} in nM	Photovoltaic efficiency [8] percent
Predict mean	4.29 ± 0.40	1.47 ± 0.07	6.40 ± 0.09
Circular FPs + linear layer	1.71 ± 0.13	$\mathbf{1.13 \pm 0.03}$	2.63 ± 0.09
Circular FPs + neural net	1.40 ± 0.13	1.36 ± 0.10	2.00 ± 0.09
Neural FPs + linear layer	0.77 ± 0.11	$\mathbf{1.15 \pm 0.02}$	2.58 ± 0.18
Neural FPs + neural net	$\mathbf{0.52 \pm 0.07}$	$\mathbf{1.16 \pm 0.03}$	$\mathbf{1.43 \pm 0.09}$

Рисунок 2.11 – Середня точність прогнозування нейронних фінгерпринтів у порівнянні із стандартними круговими

У всіх експериментах фінгерпринти нейронного графа збігалися з точністю кругових фінгерпринтів або перевершували їх, а методи з нейронною мережею поверх відбитків пальців зазвичай перевершували лінійні шари.

Обчислювальна вартість. Нейронні фінгерпринти мають однакову асимптотичну складність у кількості атомів і глибину мережі як кругові, але мають додаткові терміни через множення матриць, необхідних для

перетворення вектора ознак на кожному кроці. Якщо бути точним, складність обчислення нейронних фінгерпринтів глибини R, довжиною відбитка L молекули з N атомами за допомогою молекулярної згорткової мережі, що має F ознак на кожному рівні, дорівнює $O(RNFL + RNF^2)$. На практиці навчання нейронних мереж поверх кругових фінгерпринтів зазвичай займало кілька хвилин, у той час як навчання нейронних займало близько години на більших наборах даних.

Обмежені обчислення на кожному рівні. Наскільки складною ми маємо зробити відповідну функцію від одного шару мережі до іншого? Було обрано найпростішу можливу архітектуру: один шар нейронної мережі. Однак може бути корисним застосувати кілька шарів нелінійності між кожним кроком передачі повідомлення (як у [30]) або полегшити збереження інформації адаптацією архітектури Long Short-Term Memory [12] для передачі інформації вгору.

Обмежене поширення інформації по графі. Місцева архітектура передачі повідомлень, розроблена у [6], добре масштабується за розміром графу (через низький рівень органічних молекул), але її здатність поширювати інформацію по графу обмежена глибиною мережі. Це може бути доречним для невеликих графів, таких як ті, що представляють малі органічні молекули. Щоб уникнути проблему обмеження поширення інформації по графу, [2] запропонував ієрархічну кластеризацію підструктур графа. А деревоподібна мережа може досліджувати структуру всього графіка, використовуючи лише шари log(N), але для цього знадобиться навчитися аналізувати молекули. Прийоми обробки природної мови [31] можуть бути адаптовані до цієї області.

Нездатність розрізняти стереоізомери. Для розрізнення між ними необхідна спеціальна бухгалтерія стереоізомерів, включаючи енантомери (дзеркальні зображення молекул) і цис/транс ізомери (обертання навколо подвійних зв’язків). Більшість реалізацій кругових фінгерпринтів мають можливість зробити ці відмінності. Нейронні фінгерпринти можуть бути

розширені, щоб бути чутливими до стереоізомерів, але це залишається завданням для майбутньої роботи.

2.2 Підходи до побудови багатозадачної нейронної мережі

У багатьох практичних задачах постійно виникає наступна проблема: маючи вхідні дані необхідно побудувати для них кілька моделей, які виконують різні завдання. Часто кількість необхідних моделей може коливатися від кількох десятків до кількох сотень. В таких умовах підхід “окреме завдання - окрема модель” є дуже неефективним і непрактичним, адже це вимагає дуже багато часу та ресурсів на тренування кожної окремої моделі, підтримки їх, а також суттєво сповільнює процес інференса.

Підхід, який вирішує цю проблему, називається Багатозадачним Навчанням (Multi-task learning). Його ідея полягає в тому, щоб вирішувати множину задач “однією” моделлю. Існують різні архітектури для реалізації Багатозадачного Навчання, проте у випадках, коли важлива швидкість інференсу моделі, найпопулярнішим є підхід Жорсткого Обміну Параметрами (Hard Parameter Sharing). При ЖОП архітектурі (якби смішно не звучала ця абревіатура) існує частина нейронної мережі, яка є спільною для всіх задач (такий собі енкодер), а також індивідуальних для кожної задачі частин (такі собі декодери), які на вхід отримують вихід спільної частини.

При ЖОП архітектурі навчання відбувається через оптимізацію функції втрат, яка є лінійною комбінацією функцій втрат індивідуальних задач. За таких умов градієнт для параметрів спільної частини вираховується відносно усіх задач, які використовуються під час тренування, а градієнти для індивідуальних частин - лише на відносно відповідних задач.

Найпоширенішими проблемами, які виникають під час Багатозадачного Навчання, є проблема конфліктуючих градієнтів та нерівномірність діапазону значень функцій втрат індивідуальних задач.

Суть конфліктуючих градієнтів полягає в наступному. Оскільки загальна функція втрат, що оптимізується, є лінійною комбінацією індивідуальний функцій втрат, то загальний градієнт є лінійною комбінацією градієнтів відносно індивідуальних функцій втрат. Якщо кут між індивідуальними градієнтами більше 90°, то такі градієнти вважаються конфліктуючими, адже вказують в протилежних напрямках. Якщо ми додамо два конфліктуючі градієнти, то результуючий градієнт буде вказувати у напрямку, що не сприятиме покращенню функції втрат по жодній із задач, тому оновлення ваг таким градієнтом провокує Негативне Перенесення (Negative Transfer) - значення функцій втрат починає погіршуватися. Це, у свою чергу, взагалі може призвести до проблем зі збіжністю алгоритму оптимізації. Проблема конфліктуючих градієнтів зазвичай виникає тоді, коли дослідник неправильно підбрав задачі для сумісного тренування. Це можуть бути дуже різнорідні задачі, які мають дуже різні вхідні дані і слабку взаємопов’язаність між задачами.

Нерівномірність діапазону значень функцій втрат індивідуальних задач впливає на норму градієнтів. Очевидно, що градієнти певних задач можуть мати норму значно більшу за градієнти по інших задачах лиш через те, що їх функції втрат приймають значення з вищих діапазонів. Таким чином, оптимізація може бути упередженою на користь задач з вищими значеннями функції втрат, оскільки градієнти по цих задачах будуть відігравати основну роль у визначені напрямку загального градієнту.

Тим не менш, якщо правильно підібрати задачі для сумісного тренування та коефіцієнти в лінійній комбінації функцій втрат, то в результаті Багатозадачного Навчання можна покращити метрики по кожній задачі у порівнянні з випадком, коли моделі тренуються окремо. Це стається за рахунок того, що ми збільшуємо розмір і різноманітність тренувальної вибірки та

отримуємо більш інформативні та узагальнюючі фічі на виході спільної енкодерної частини.

Отож, розглянемо, які існують методи формування загальної функції втрат, аналізуючи ідеї побудови методу, опису його гіперпараметрів і порівняння з іншими методами.

2.2.1 Рівнозначне зважування (EW)

Повна назва: рівнозначне зважування (EW - Equal Weighting).

Ідея: це – найпростіший варіант реалізації, при якому коефіцієнти при лінійній комбінації рівні 1 для всіх задач, тому загальна функція втрат є простою сумою індивідуальних функцій втрат.

Гіперпараметри: відсутні.

Порівняння з іншими методами: це – базовий метод реалізації Багатозадачного Навчання, який можна брати за “початкову точку відліку” для порівняння ефективності інших методів.

2.2.2 Випадкове зважування (RLW)

Повна назва: випадкове зважування (RLW - Random Loss Weighting) [5].

Дата публікації статті: 20 Листопада 2021.

Ідея: генерує кожної ітерації нові випадкові ваги для кожного завдання, які нормалізуються за допомогою Softmax, щоб в сумі давати константу.

Гіперпараметри: можна задавати випадковий розподіл для генерації ваг.

Порівняння з іншими методами: у статті наводиться екстенсивне порівняння RLW (з різними випадковими розподілами) з існуючими методами, такими як EW, UW, GradNorm, MGDA, DWA, PCGrad, GradDrop, IMTL, GradVac. Дивує, що результативність абсолютно випадкового зважування знаходиться приблизно на одному рівні з більш складними методами, такими як

IMTL. Незважаючи на те, що IMTL в більшості випадків все ж таки показав найкращі результати, RLW зумів перевершити його на деяких задачах.

2.2.3 Зважування за невизначеністю (UW)

Повна назва: зважування за невизначеністю (UW - Uncertainty Weighting) [15].

Дата публікації статті: 2018.

Ідея: робиться припущення, що вихід моделі для кожної задачі відповідає певному статистичному розподілу, який має невизначеність конкретної задачі. Потім, використовуючи Метод Максимальної Правдоподібності, ми максимізуємо log-likelihood відносно параметрів моделі W та параметра шуму спостереження σ (який відповідає за невизначеність, притаманну задачі).

Гіперпараметри: відсутні.

Порівняння з іншими методами: це був один із перших алгоритмів, де реалізовано автоматичний пошук оптимальних коефіцієнтів для функцій втрат. Тому, у статті його порівнюють лише з алгоритмом повного перебору сітки значень і показують, що значення отримані методом зважування за невизначеністю дуже близькі до найкращих значень отриманих методом перебору.

2.2.4 Нормалізація градієнтів (GradNorm)

Повна назва: нормалізація градієнтів (GradNorm – Gradient Normalization) [3].

Дата публікації статті: 2018.

Ідея: основна ідея полягає в тому, щоб через підібране значення коефіцієнту для функції втрат корегувати норму градієнта кожної окремої задачі. Тоді, задачі, що повільно навчаються, повинні мати більшу норму, щоб

посилити їх вплив у загальний градієнт, тоді як вплив задач, що швидко навчаються, має бути зменшено через зменшення норми їх градієнтів.

Знаходження оптимальних коефіцієнтів досягається шляхом оптимізації допоміжної функції втрат лише відносно коефіцієнтів.

$$L_{\text{grad}} = \sum_i |G_W^{(i)}(t) - \overline{G}_W(t) \times [r_i(t)]^\alpha|_1$$

Після оновлення значень коефіцієнтів виконується стандартний крок зворотнього проходу назад відносно параметрів нейронної мережі. Значення коефіцієнтів тоді нормалізуються, щоб в сумі давати T, де T - кількість задач.

Гіперпараметри: α - асиметрія [повинна бути > 0]. α встановлює міру відновлювальної сили, яка повертає швидкість, з якою тренується окрема задача, до середньої швидкості тренування. Коли задача особливо вирізняється за своєю складністю, це призводить до різної динаміки навчання між задачами. У такому випадку слід використовувати більше значення α, щоб забезпечити сильніше збалансування швидкості навчання. Коли завдання є більш однорідними - доречним є нижче значення α. Зауважте, що α = 0 завжди намагатиметься встановити норми градієнтів для зворотного поширення для кожної задачі такими, щоб вони були рівними.

Порівняння з іншими методами: автори статті порівнювали свій метод з UW і EW і було показано, що GradNorm у всіх випадках або працює однаково, або краще за ці методи. GradNorm також має значну перевагу над UW завдяки нормалізації коефіцієнтів, оскільки це запобігає необмеженому росту або спаду значень коефіцієнтів.

2.2.5 Динамічне усереднення коефіцієнтів (DWA)

Повна назва: динамічне усереднення коефіцієнтів (DWA - Dynamic Weight Average) [21].

Дата публікації статті: 2019.

Ідея: натхнена алгоритмом GradNorm, але працює набагато простіше. Коефіцієнти для кожної задачі для епохи t обчислюються як нормалізовані відношення функцій втрат за попередні епохи L_k(t-1)/L_k(t-2), де L_k(t) - значення функції втрат для задачі k на t-й епосі. Нормалізація проводиться через Softmax по всіх задачах, а потім отримані значення множаться на кількість завдань.

Гіперпараметри: відсутні.

Порівняння з іншими методами: у статті автори порівнюють свій метод лише з EW та UW (також порівнюючи з не ЖОП-архітектурами, такими як Cross-Stich). На 1 наборі даних DWA не досягла найкращих результатів серед жодного із завдань, а на іншому в більшості задач досягла успіху.

Дивно те, що автори надихалися алгоритмом GradNorm, проте не порівняли з ним. Однак автори RLW у своїй статті наводять порівняння DWA з іншими методами. Цікаво, що він значною мірою домінує над GradNorm і в цілому показує результати вище середнього.

2.2.6 “Неконфліктуючий” градієнтний спуск (CAGrad)

Повна назва: “неконфліктуючий” градієнтний спуск (CAGrad - Conflict-Averse Gradient Descent for Multi-task Learning) [19].

Дата публікації статті: 26 жовтня 2021.

Ідея: цей метод зосереджується на боротьбі з проблемою конфліктуючих градієнтів і розглядає сумісну функцію втрат у вигляді звичайного усереднення індивідуальних функцій втрат. За такої функції втрат загальний градієнт оновлення буде середнім значенням градієнтів для окремих задач. Очевидно, що це не є найоптимальнішим рішенням. Тому автори алгоритму пропонують замінити усереднений градієнт вектором із його c-околиці, що максимізує найгірше покращення функції втрат серед задач (тобто, намагається підбрати такий вектор для градієнту, що максимально “допоможе” задачі, в якої значення функції втрат покращилося найменшими темпами).

Гіперпараметри: c - контролює околиці усередненого градієнта для пошуку нового вектора оновлення. Щоб гарантувати, що CAGrad сходиться до оптимуму L0(θ), ми повинні забезпечити виконання умови $0 \leq c < 1$.

Порівняння з іншими методами: у статті [19] автори порівнюють свій метод з MGDA і PCGrad (іншими старішими алгоритмами, які теж зосереджені на подолані проблеми конфліктуючих градієнтів). Основна перевага CAGrad над цими методами полягає в наступному (цитата зі статті): "…вони (MGDA і PCGrad & co) маніпулюють градієнтом, оптимізуючи модифіковану функцію втрат L_. Таким чином, ці методи в принципі можуть сходитися до будь-якої точки Парето фронту відносно початкової функції втрат L_0. У свою чергу CAGrad мінімізує саме незмінну початкову функцію втрат L_0, використовуючи конфліктуючі градієнті для регляризації траєкторії оптимізації та забезпечує збіжність до оптимуму L_0 за умови $0 \leq c < 1$".

2.2.7 Неупереджене багатозадачне навчання (IMTL)

Повна назва: неупереджене багатозадачне навчання (IMTL - Impartial Multi-Task Learning) [20].

Дата публікації статті: 2021.

Ідея: унікальність запропонованого методу полягає в тому, що він вирішує одразу 2 проблеми: балансування градієнтів (IMTL-G) і масштабування значень функцій втрат (IMTL-L).

IMTL-G – тренує коефіцієнти кожної функції втрат таким чином, щоб агрегований градієнт мав рівні проекції на вихідні градієнти окремих задач. Якщо накласти умови, що сума коефіцієнтів має бути рівна 1, можна отримати розв'язок цієї задачі в закритій формі.

IMTL-L - масштабує функції втрат за допомогою експоненти, де її степінь є параметром, який оптимізується під час навчання.

IMTL поєднує IMTL-G і IMTL-L в одному алгоритмі, так що всі параметри обчислюються автоматично.

До речі, у статті автори запропонували трюк для підвищення обчислювальної ефективності. Вони пропонують розрахувати градієнти для IMTL-G відносно виходів спільної частини нейронної мережі (енкодера) замість того, щоб розраховувати градієнт відносно параметрів енкодера. Інтуїтивно, у виходах енкодерної частини вже закладено результат математичних операцій з її параметрами, тому вони апроксимують собою функцію усього енкодера.

Перевага полягає в тому, що розмірність вектору градієнтів, обчислених у такий спосіб, дорівнює розміру останнього вихідного шару енкодера. Тому такий підхід робить алгоритм масштабованим для великих нейромереж і заощаджує дуже багато часу під час обчислення коефіцієнтів функцій втрат (оскільки розв’язок у закритій формі включає в себе обрахунок оберненої матриці).

Цікаво, що автори статті наводять порівняння двох реалізацій: з трюком і “в лоб” (тобто чесне обчислення градієнтів відносно параметрів спільної енкодерної частини). Вражає, але “чесний” варіант дає трохи гірші метрики, при тому що займає аж в 1,6 разів більше часу.

Гіперпараметри: відсутні.

Порівняння з іншими методами: у статті представлено дуже детальне гарне порівняння з іншими методами, такими як UW, MGDA, GLS, GradNorm, PCGrad.

Традиційно тестування алгоритмів проводилося на багатозадачних наборах даних із зображеннями. Причому майже в усіх задачах IMTL досягнув найкращих результатів, тоді як лише по 1 задачі UW був найкращим.

Крім того, однією з найголовніших переваг IMTL автори вважають відсутність упередженості в алгоритмі.

Ось приклади упереджень в інших алгоритмах:

- MGDA. Зосереджується на задачах з малими значеннями градієнта;
- PCGrad. Задачі з більшою нормою градієнта навчаються краще, що може спровокувати ті ж проблеми, що виникають при EW, яке наївно

додає всі функції втрат, незважаючи на те, що масштаби значень цих функцій втрат дуже різні;

- GLS (замість простого усереднення, як у випадку EW, розглядає середнє геометричне функцій втрат) чутливий до змін масштабу втрат {L_t} і дуже нестабільний при збільшені кількості задач > 20.
- GradNorm прагне оптимізувати «більшість задачі», напрямки градієнтів яких ближче до середнього напрямку Sum_t(u_t), що призводить до небажаного упередження під час навчання.
- UW 1) зважує кожну задачу незалежно; 2) може надати перевагу регресійним задачам перед класифікаційними; 3) накладає припущення щодо розподілу і, таким чином, не може застосовуватися до різних функцій втрат; 4) вводить наближення, щоб отримати формулу розрахунку коефіцієнтів для певних функцій втрат (наприклад, таких як перехресна ентропія), які можуть бути неоптимальними.

2.2.8 Проєктування конфліктуючих градієнтів (PCGrad)

Повна назва: проєктування конфліктуючих градієнтів (PCGrad - Projecting Conflicting Gradients) [37].

Дата публікації статті: 22 грудня 2020.

Ідея: PCGrad шукає пари конфліктуючих градієнтів і бореться з конфліктом, проектуючи один градієнт на нормальну площину іншого (якщо кут між градієнтами був > 90, робить його рівним 90).

Гіперпараметри: відсутні.

Порівняння з іншими методами: єдина стаття з наведених у цій доповіді, яку ми не розбирали. Але вирішили включити в перелік методів, оскільки помітили в кількох статтях порівняння з цим методом. PCGrrad був основою для інших вдосконалених методів, таких як GradVac.

2.2.9 Вакцинування градієнтів (GradVac)

Повна назва: вакцинування градієнтів (GradVac - Gradient Vaccine) [35].

Дата публікації статті: 2021.

Ідея: дуже нагадує ідею PCGrad, але вносить додаткові вдосконалення. GradVac обробляє не лише ситуацію з негативним перенесенням (коли кут між градієнтами більше 90 градусів), а й випадок позитивної подібності (cos кута між градієнтами задач > 0). Таким чином, GradVac проактивно реагує, коли задачі подібні і cos кута між їхніми градієнтами хоч і більший за 0, але все ще недостатньо великий, щоб з максимальною користю використати наявну подібність між задачами.

Гіперпараметри: beta - коефіцієнт експоненціального ковзного середнього (попередніх значень косинуса кутів між градієнтами задач).

Порівняння з іншими методами: автори розробили алгоритм GradVac для навчання мовних моделей, тому всі їх порівняння з іншими методами проводяться на текстових наборах даних. Варто зазначити, що попередні методи були розроблені для багатозадачних завдань CV та оцінені на тих наборах даних із зображенями. Так, на наборах мовних даних GradVac показав найкращі результати на більшості завдань (при цьому GradNorm перевершив в одному завданні). Порівняння відбувалося з алгоритмами GradNorm, PCGrad і MGDA.

Висновки

У даному розділі ми розглянули інтуіцію до побудови графової нейронної мережі, які є види задач для графів, а також навели основні підходи для реалізації графових нейронних мереж: графові згорткові (convolutional) та

уважні (attentional) нейронні мережі, а також нейронні мережі із передачею повідомлень (message-passing).

MF підхід узагальнює наявні молекулярні признаки, створені вручну, щоб дозволити їх оптимізацію для різноманітних завдань. Роблячи кожну операцію в пайплайні диференційованою, можна використовувати стандартні нейронномережеві методи навчання для масштабованої оптимізації параметрів цих нейронних молекулярних фінгерпринтів. Було продемонстровано інтерпретацію та прогнозну продуктивність цих нових фінгерпринтів.

Признаки, керовані даними, вже замінили ручні признаки в розпізнаванні мовлення, машинному баченні та обробці природної мови. Виконання того ж завдання для віртуального скринінгу, дизайну ліків і дизайну матеріалів є природним наступним кроком.

У результаті проведеного аналізу, на нашу думку, стосовно вибору найкращого методу для тренування багатозадачної нейронної мережі, то це IMTL, який, до речі, є одним із останніх опублікованих. Ця стаття заслуговує особливої уваги й довіри, адже автори дуже ретельно аналізують існуючі підходи та їх недоліки, на основі цього пропонують дуже цікаву власну ідею, яка долає вразливості попередніх алгоритмів, і зрештою наводять дуже детальне порівняння ефективності свого методу з існуючими.

Алгоритми, по типу PCGrad, CAGrad, GradVac & Co, ідея яких базується на прямій корекції напрямків існуючих градієнтів, а не на регулюванню їх норм, не виправдовують себе через кілька причин. По-перше, вони розглядають загальну функцію втрат як просту суму індивідуальних функцій втрат. Це суттєвє обмеження, оскільки очевидно, що так робити небезпечно, коли різні типи задач мають різну функцію втрат, які приймають значення з різних діапазонів. Адже частина задач може почати домінувати під час навчання просто через те, що їхні функції втрат приймають більших значень і відповідно градієнти будуть мати більшу норму. То ж безпечно застосовувати ці алгоритми лише для сумісного тренування однорідних задач. По-друге, ці алгоритми вимагають велику кількість оперативної пам’яті та обчислень. Тому

швидкість тренування поступається іншим алгоритмам, а сам процес навчання ускладнюється зі зростанням складності моделей.

Якщо найбільш критичним для вас є швидкість тренування і простота реалізації, то варто експериментувати з алгоритмами EW, RLW, UW і DWA - вони є лідерами по цих параметрах.

РОЗДІЛ 3 ПОБУДОВА ГРАФОВОЇ БАГАТОЗАДАЧНОЇ НЕЙРОННОЇ МЕРЕЖІ

3.1 Опис проблеми й огляд датасетів

Як уже згадувалося в попередніх розділах, ми спробуємо побудувати графову багатозадачну нейронну мережу для передбачення ADME-Tox властивостей молекул.

По-перше, чому саме графові нейронні мережі? Тому що саме такі архітектури здатні сповна перевести молекулу із її реального молекулярного простору в математичний графовий простір, а звідти у вектор чисел, за допомогою якого ми вже здатні передбачати будь-яку регресійну, чи класифікаційну величину. Тобто кожну молекулу ми можемо розглядати як граф, де вершини (ноди) графа – атоми молекули, а ребра (еджі) – зв’язки між цими атомами.

По-друге, чому саме багатозадачність? Взагалі, дещо раніше ми будували різні моделі класичного МЛ, використовуючи авторський АвтоМЛ (до речі, отримані результати ми будемо порівнювати саме із цими результатми, отриманими за рахунок класичного МЛ). Проте такий підхід працює необ’єктивно довго, враховуючи той факт, що для всіх ADME-Tox властивостей (параметрів) і для всіх цих моделей ми працюємо в одній площині, а саме із молекулами (при чому, деякі молекули можуть перетинатися майже для всіх параметрів, тобто одна й та сама молекула в одному випадку може говорити показник одного параметра, в іншому - іншого й т.д.). Тому було вирішено спробувати побудувати багатозадачну (мульти-таск) модель, яка допоможе пришвидшити предікт у кілька, а то й у кількасот разів, оскільки буде видавати умовно універсальний набір фічів для всіх параметрів одночасно, а не для кожного параметра окремо, як це було при побудові незалежних моделей під кожен параметр. Отже, основна задача багатозадачності – пришвидшити час предікту, оскільки потім ми будемо працювати із

мільйонами, а то й мільярдами молекул, для яких захочемо отримати результати предікту всіх ADME-Tox властивостей. Також ціль полягає в тому, щоб отримати результати по кожному із параметрів не гірші, ніж були при побудові звичайних моделей під кожен параметр.

Що стосується самих ADME-Tox властивостей та датасетів під кожен параметр, то всього в нас їх 41, проте деякі із них можуть містити передбачення одночасно і регресійної, і бінарної величини. Тобто для таких параметрів вводився якийсь катоф, який перетворив регресійну величину в бінарну, проте ця бінарна величина все одно залишається дуже інформативною із біологічної точки зору. На жаль, ми не можемо поділитися самими даними, оскільки побудова цієї нейронної мережі – комерційний проєкт, а також не будемо наводити кількість молекул у кожному датасеті задля безпеки. У табл. 3.1 наведений список усіх ADME-Tox параметрів та їх приналежність до бінарної класифікації, або регресії, або і того, і того одразу.

Таблиця 3.1 – Список ADME-Tox параметрів та їх тип задачі

Назва параметру	Тип	Бінарна класифікація	Регресія
Bioavailability	Absorption	+	-
HIA	Absorption	+	-
P-gp inhibitors	Absorption	+	-
P-gp substrates	Absorption	+	-
Caco-2	Absorption	-	+
BBB	Distribution	+	-
PPB	Distribution	-	+
VD	Distribution	-	+
CYP1A2 inhibitor	Metabolism	+	+
CYP2C9 inhibitor	Metabolism	+	+
CYP2C19 inhibitor	Metabolism	+	+

Продовження таблиці 3.1

CYP2D6 inhibitor	Metabolism	+	+
CYP3A4 inhibitor	Metabolism	+	+
CYP1A2 substrate	Metabolism	+	-
CYP2C9 substrate	Metabolism	+	-
CYP2C19 substrate	Metabolism	+	-
CYP2D6 substrate	Metabolism	+	-
CYP3A4 substrate	Metabolism	+	-
Half-life	Excretion	-	+
Plasma Clearance	Excretion	-	+
Renal Clearance	Excretion	-	+
Acute	Toxicity	-	+
AMES	Toxicity	+	-
Carcinogenicity ISF	Toxicity	+	+
Carcinogenicity OSF	Toxicity	+	+
Androgen antagonist	Toxicity	+	-
Androgen agonist	Toxicity	+	-
Androgen binding	Toxicity	+	-
Estrogen antagonist	Toxicity	+	-
Estrogen agonist	Toxicity	+	-
Estrogen binding	Toxicity	+	-
hERG	Toxicity	+	+
DILI	Toxicity	+	-

Отже, всього ми маємо 33 універсальні ADME-Tox властивості, 8 із яких містять і бінарну, і регресійну величини, тобто всього виходить 26 задач бінарної класифікації та 15 задач регресії.

На цьому етапі цікавим для аналізу буде порівняння датасетів на вміст однакових молекул. На рисунку 3.1 зображено попарний молекулярний перетин

усіх датасетів із відносним показником, щось на подобі матриці кореляцій. Варто відзначити, що матриця – не симетрична. Чим вищий показник – тим більше молекул із датасету рядка присутня в датасеті стовпця.

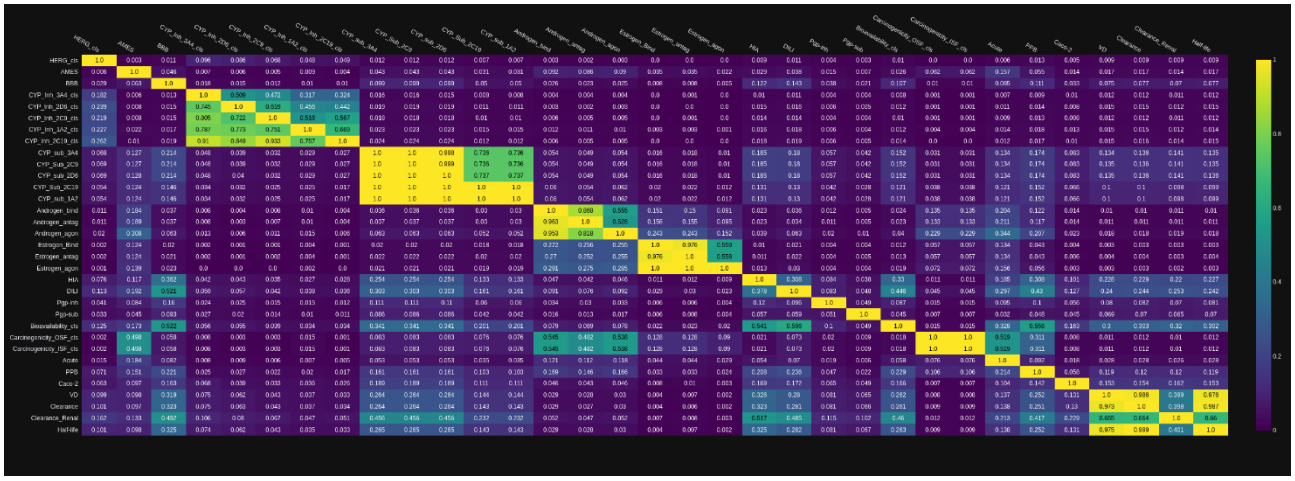

Рисунок 3.1 – Попарний молекулярний перетин усіх датасетів

Як ми бачимо, більшість датасетів містять універсальні молекули, проте існують деякі кластери властивостей, які містять або дуже схожий, або повністю однаковий набір молекул. Наприклад, усі CYP інхібітори, усі CYP субстрати, андрогени, естрогени, обидві карциногенності, що впринципі логічно, оскільки ці властивості дуже сильно перетинаються в біології. А от велика подібність VD до датасетів із Excretion (обидва кліренси та Half-life) не є настільки очевидною й потрібно це враховувати при поділі датасетів на тренувальну, валідаційну та тестові вибірки.

3.2 Вибір методу багатозадачного тренування

3.2.1 Терміни

Базові значення метрик – значення метрик моделей, натренованих за допомогою AutoML Pipeline в режимі “окремий датасет - окрема модель”.

AutoML Pipeline – внутрішньо розробрлений пайплайн для автоматизації тренування моделей класичного машинного навчання, який включає етапи автоматичного генерування молекулярних дескрипторів, їх відбору та підбору гіперпараметрів за допомогою алгоритмів баєсовської оптимізації (рис. 3.2).

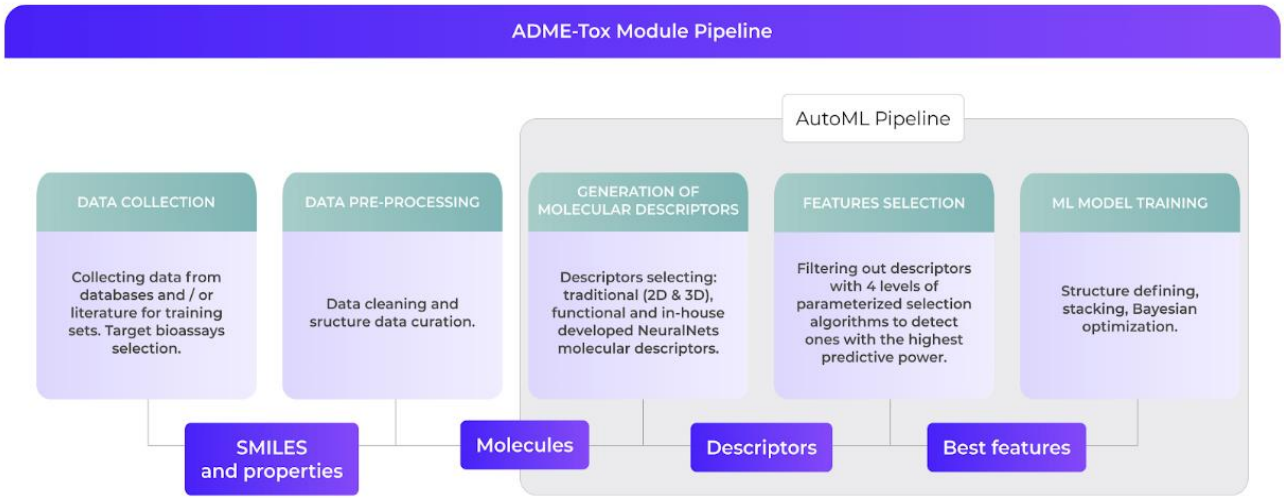

Рисунок 3.2 – AutoML Pipeline

Якість датасету – це введена для зручності характеристика датасету, яка може мати значення “висока якість”, “середня якість” та “низька якість”. Всі наявні молекулярні датасети були проранжовані по базових значеннях цільової метрики та відповідно погруповані на три класи якості.

3.2.2 Постановка досліджень

Отож, на початку досліджень було відомо наступне:

1. Більше 40 датасетів з різними розмірами та якістю.
2. Базові значення метрик моделей, для кожного датасету.
3. 9 методів багатозадачного навчання нейронних мереж.
4. Проблеми, які виникають при багатозадачному навчанні, а саме “конфліктуючі градієнти” та різна область значень індивудальних функцій втрат.

Задачі проведення досліджень:

1. Метод багатозадачного тренування, який буде найкращим компромісом між якістю отриманої моделі та швидкістю тренування.
2. Дослідити залежність метрик багатозадачної нейронної мережі та стабільність її тренування від обраного методу багатозадачного тренування, якості, кількості та типу датасетів, відібраних для сумісного тренування.
3. Визначити оптимальний набір датасетів для тренування багатозадачної нейронної мережі.

На початку дослідження була висунута наступна гіпотеза:

Враховуючи різну якість та розміри датасетів, найкращі результати багатозадачна нейронна мережа може досягнути тренуючись на їх підвибірці. Ця гіпотеза є підґрунтям для задачі 3.

3.2.3 Методика досліджень

Для отримання інформації необхідної для розв’язку поставлених задач пропонується провести серію експериментів.

Експеримент, в цьому контексті, визначається набором тренувальних датасетів (множини датасетів, на яких навчається багатозадачна нейронна мережа) та набором зовнішніх датасетів (множини датасетів, які не використовувалися для тренування багатозадачної нейронної мережі і слугують критерієм її узагальнюючої здатності).

Під час кожного експерименту нейромережа навчається на тому ж наборі тренувальних датасетів кожним із наперед визначених методоів багатозадачного тренування.

Проведені експерименти мають вимірність 3, тобто є 3 осі параметрів по яким визначається експеримент, а саме:

1. Кількість датасетів у тренувальному наборі.

2. Типи датасетів у тренувальному наборі. Тобто, наявність лише класифікаційних, лише регресійних або датасетів обох типів.
3. Якість датасетів включених до тренувального набору. Наприклад, лише високої якості або наявність датасетів різної якості.

Таким чином, задаючи різні конфігурації експериментів та порівнюючи їх результати між собою, отримуємо можливість встановити шукані взаємозалежності, сформульовані у постановці задач.

Таблиця 3.2 – Конфігурації експериментів

Тренувальний набір датасетів	Зовнішній набір датасетів	№ експерименту
AMES, HERG_cls	Bioavailability, Carcinogenicity_ISF, CYP_inh_2D6 (cls + rgr)	0
Acute, CYP_inh_3A4_rgr	Bioavailability, Carcinogenicity_ISF, CYP_inh_2D6 (cls + rgr)	1
AMES, HERG_cls, BBB	Bioavailability, Carcinogenicity_ISF, CYP_inh_2D6 (cls + rgr)	2
Acute, CYP_inh_3A4_rgr, Caco-2	Bioavailability, Carcinogenicity_ISF, CYP_inh_2D6 (cls + rgr)	3
AMES, HERG_cls, BBB, CYP_inh_3A4_cls	Bioavailability, Carcinogenicity_ISF, CYP_inh_2D6 (cls + rgr)	4
Acute, CYP_inh_3A4_rgr, Caco-2, HERG_rgr	Bioavailability, Carcinogenicity_ISF, CYP_inh_2D6 (cls + rgr)	5
AMES, Acute	Bioavailability, Carcinogenicity_ISF, CYP_inh_2D6 (cls + rgr)	6
AMES, HERG_cls, Acute, CYP_inh_3A4_rgr	Bioavailability, Carcinogenicity_ISF, CYP_inh_2D6 (cls + rgr)	7

Продовження таблиці 3.2

AMES, HERG_cls, BBB, CYP_inh_3A4_cls, Acute, CYP_inh_3A4_rgr, Caco-2, HERG_rgr	Bioavailability, Carcinogenicity_ISF, CYP_inh_2D6 (cls + rgr)	8
AMES, HERG_cls, BBB, CYP_inh_3A4_cls, HIA, Bioavailability, CYP_sub_2D6, Carcinogenicity_ISF	Bioavailability, Carcinogenicity_ISF, CYP_inh_2D6 (cls + rgr)	13
HERG_rgr, HERG_cls, CYP_inh_3A4_rgr, CYP_inh_3A4_cls, CYP_inh_2D6_rgr, CYP_inh_2D6_cls	Bioavailability, Carcinogenicity_ISF, CYP_inh_2D6 (cls + rgr)	17
HIA, Clearance, CYP_sub_2D6	Bioavailability, Carcinogenicity_ISF, CYP_inh_2D6 (cls + rgr)	18
AMES, HERG_cls, BBB, CYP_inh_3A4_cls, Acute, CYP_inh_3A4_rgr, Caco-2, HERG_rgr, HIA, Clearance, Bioavailability_cls, Bioavailability_rgr, CYP_sub_2D6, Carcinogenicity_ISF_rgr, Carcinogenicity_ISF_cls, PGP_sub, CYP_inh_2D6_cls, CYP_inh_2D6_rgr	Bioavailability, Carcinogenicity_ISF, CYP_inh_2D6 (cls + rgr)	19

3.2.4 Встановлення взаємозалежностей

3.2.4.1 Вплив кількості датасетів у тренувальному наборі на якість моделі

Для проведення перших експериментів, розглянемо лише датасети з найвищою якістю та найбільшою кількістю молекул. При цьому, розглянемо

групи експериментів з лише бінарними і лише регресійними датасетами, щоб мінімізувати вплив проблеми різної область значень функцій втрат.

Для окремо класифікаційних задач, розглянемо експерименти 0, 2 і 4 (рис. 3.3-3.4).

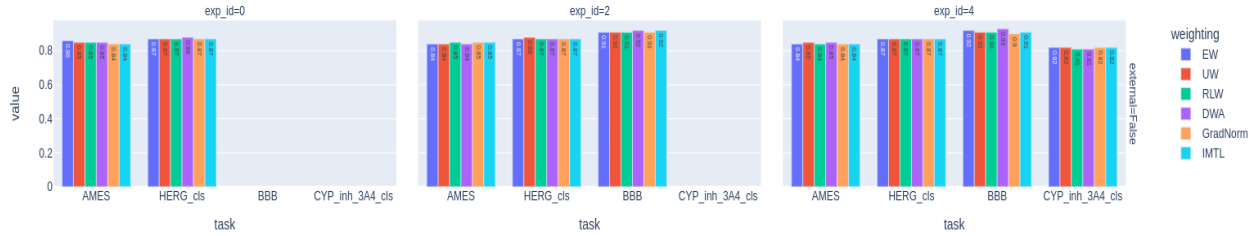

Рисунок 3.3 – Метрики на тестових частинах датасетів з тренувального набору експериментів 0, 2 і 4

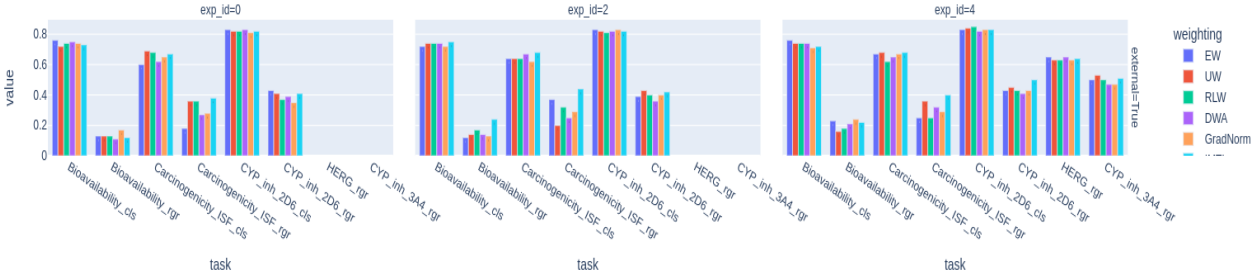

Рисунок 3.4 – Метрики на тестових частинах датасетів із зовнішнього набору експериментів 0, 2 і 4

Висновки:

1. для класифікаційних задач збільшення кількості датасетів не дає значного покращення у значеннях метрик;
2. для датасетів з тренувальної вибірки результативність різних методів багатозадачного навчання приблизно однакова.
3. для датасетів із зовнішньої вибірки IMTL або не поступається іншим методам, або значно перевершує їх.

Для окремо регресійних задач, розглянемо експерименти 1, 3 і 5 (рис. 3.5-3.6).

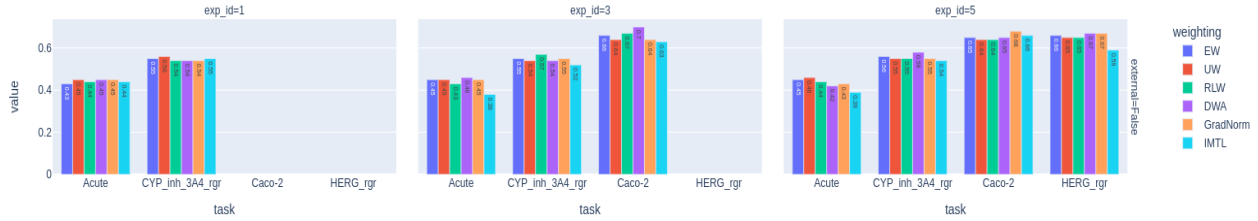

Рисунок 3.5 – Метрики на тестових частинах датасетів з тренувального набору експериментів 1, 3 і 5

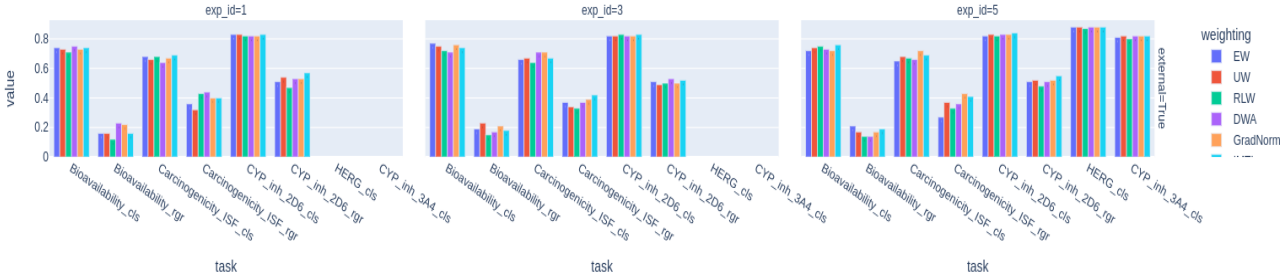

Рисунок 3.6 – Метрики на тестових частинах датасетів із зовнішнього набору експериментів 1, 3 і 5

Висновки:

1. для регресійних задач збільшення кількості датасетів теж не дає значного покращення у значеннях метрик;
2. для датасетів з тренувальної вибірки результативність різних методів багатозадачного навчання відмінна - для кожного датасету можна чітко визначити, який метод дав найкращі/гірші метрики. Проте, найкращий/гірший метод не є узгодженим між завданнями, тобто для різних задач один і той самий метод багатозадачного тренування може давати як найкращі, так і найгірші результати з-поміж інших методів;
3. для задач з тренувальної вибірки IMTL зазвичай показує гірші результати порівняно з іншими методами.

Загальний висновок: як показали експерименти, принаймні для датасетів високої якості і великих розмірів, включення додаткових датасетів у тренувальний набір не викликає систематичного просідання чи підвищення

метрик. Тобто, найголовніше, що процес тренування є стабільним. Це означає, що задачі добре узгоджуються між собою і тому проблеми конфліктуючих градієнтів не виникає.

3.2.4.2 Вплив збільшення кількості датасетів у тренувальному наборі на якість моделі у випадку, коли датасети різних типів

На практиці нейронна мережа повинна використовуватися як для класифікаційних, так і для регресійних задач. Тому потрібно дослідити процес сумісного тренування регресійних і класифікаційних датасетів. Отож, наступні експеримименти подібні, до розглянутих у попередньому пункті, з відмінністю, що різні типи задач тепер тренуються сумісно.

Для цього проаналізуємо результати експериментів 6, 7 і 8 (рис. 3.7-3.8).

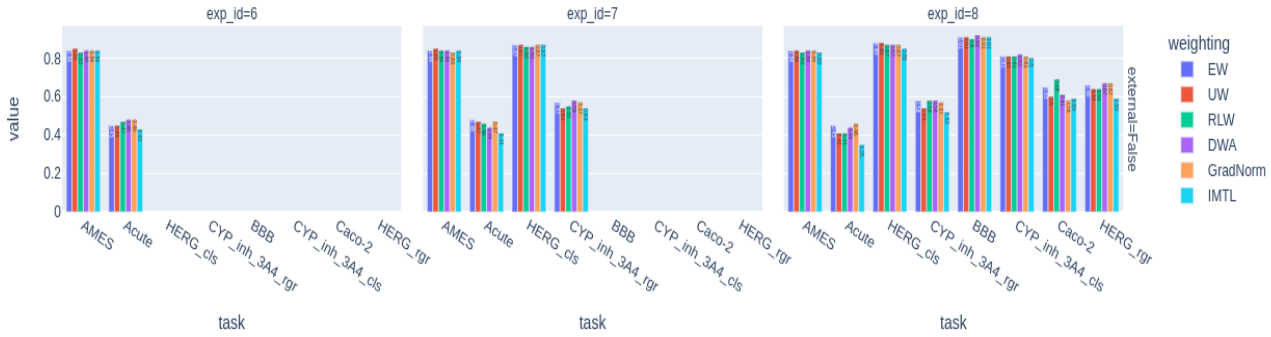

Рисунок 3.7 – Метрики на тестових частинах датасетів з тренувального набору експериментів 6, 7 і 8

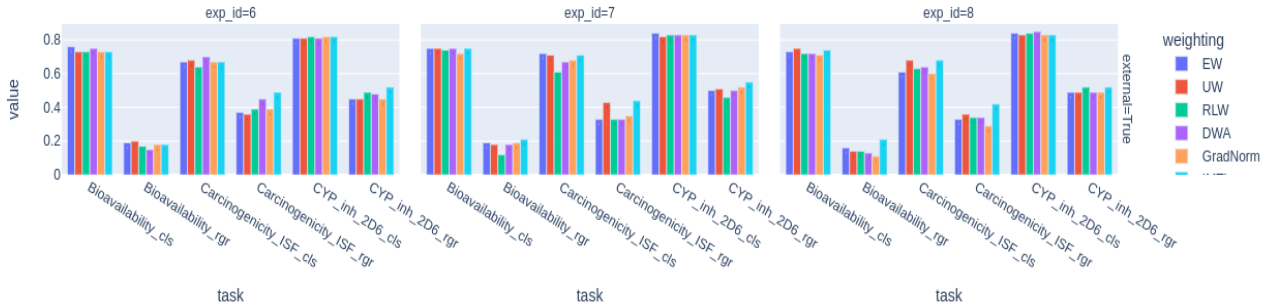

Рисунок 3.8 – Метрики на тестових частинах датасетів із зовнішнього набору експериментів 6, 7 і 8

Висновки:

1. включення додаткових датасетів не вносить суттєвих систематичних змін у значення метрик по жодному з датасетів;
2. оскільки тренування стабільне, а метрики як по класифікаційних, так і по регресійних датасетах залишаються високими, то це означає, що проблеми з різними діапазонами значень функцій втрат не виникає;
3. також, що досить неочікувано, не простежується жодного підходу багатозадачного навчання, який би систематично чи принаймні у більшості випадків давав би кращі результати;
4. для кожної задачі, в залежності від експериментів, найвищі значення метрик і найкращий метод багатозадачного тренування є різним.

3.2.4.3 Вплив включення датасетів низької та середньої якості у тренувальний набір на якість моделі

В попередніх пунктах увага була зосереджена виключно на датасетах високої якості, оскільки це були перші експерименти і теоретично вони мали давати найкращі результати. Проте у нас наявні також датасети гіршої якості і менших розмірів. Тому поставало питання, наскільки робастним є процес багатозадчного тренування до якості і розмірів елементів тренувального набору. Також цікавим питанням є чи покращаться метрики на датасетах гіршої якості від того, що модель навчатиметься разом із датасетами високої якості.

Для відповіді на ці питання порівняємо наступні пари експериментів.

Пара 1 (рис. 3.9-3.10) – експеримент 4 (найкращі класифікаційні датасети) і 13 (класифікаційні датасети різних якостей).

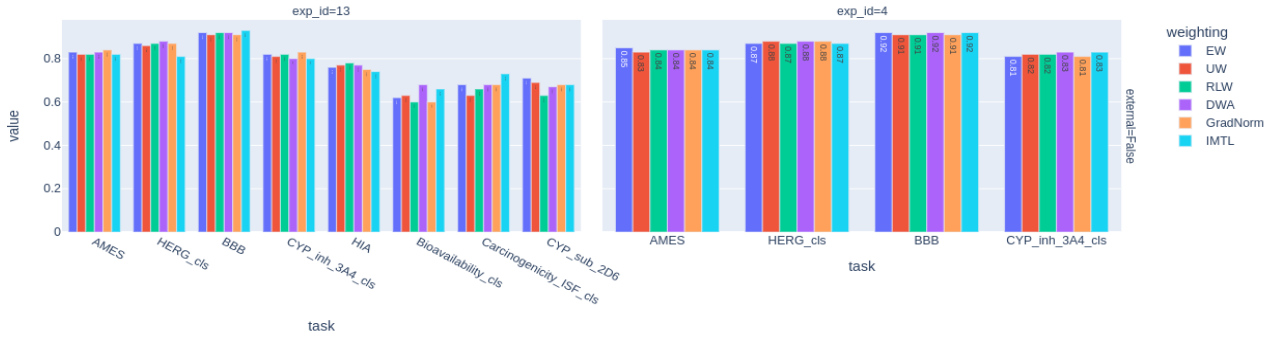

Рисунок 3.9 – Метрики на тестових частинах датасетів з тренувального набору експериментів 4 і 13

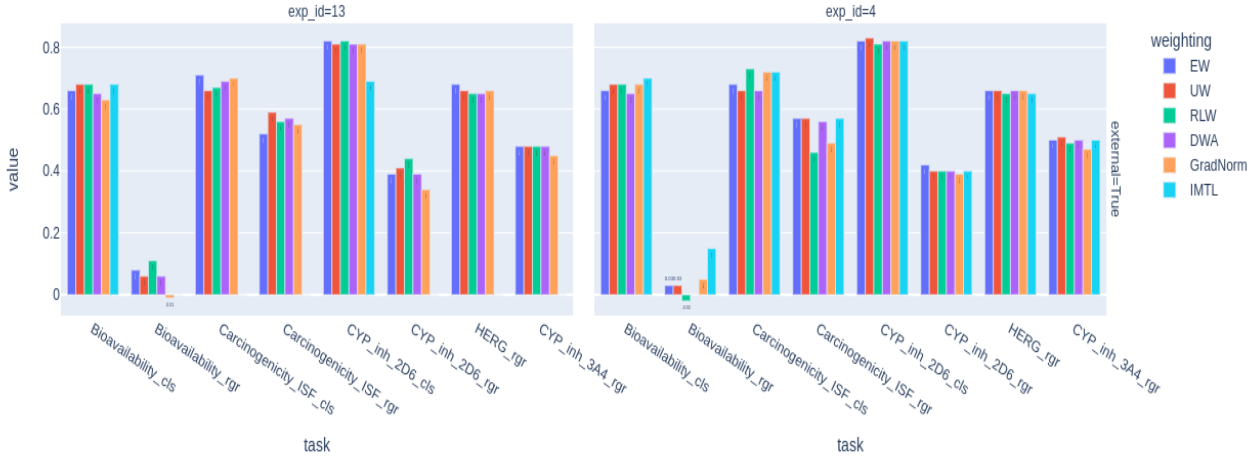

Рисунок 3.10 – Метрики на тестових частинах датасетів із зовнішнього набору експериментів 4 і 13

Висновки:

1. сумісне навчання з датасетами низької якості веде до погіршення метрик на датасетах високої якості. Найвищі значення метрик на датасетах високої якості досягаються саме під час тренування з датасетами аналогічної якості. Також, найгірші значення метрик практично не відрізняються від найкращих, при тому, що в експерименті 13 ця різниця дуже помітна;
2. узагальнююча здатність багатозадачної нейронної мережі також вища у випадку тренування виключно на датасетах високої якості, що видно з аналізу метрик на зовнішніх датасетах (у випадку експеримента 4 вони вищі, ніж у випадку експеримента 13).

Пара 2 (рис. 3.11-3.12) – експеримент 6 (найкращі класифікаційні та регресійні датасети) та 19 (додатково до експ. 6 містить класифікаційні та регресійні датасети середньої та низької якості).

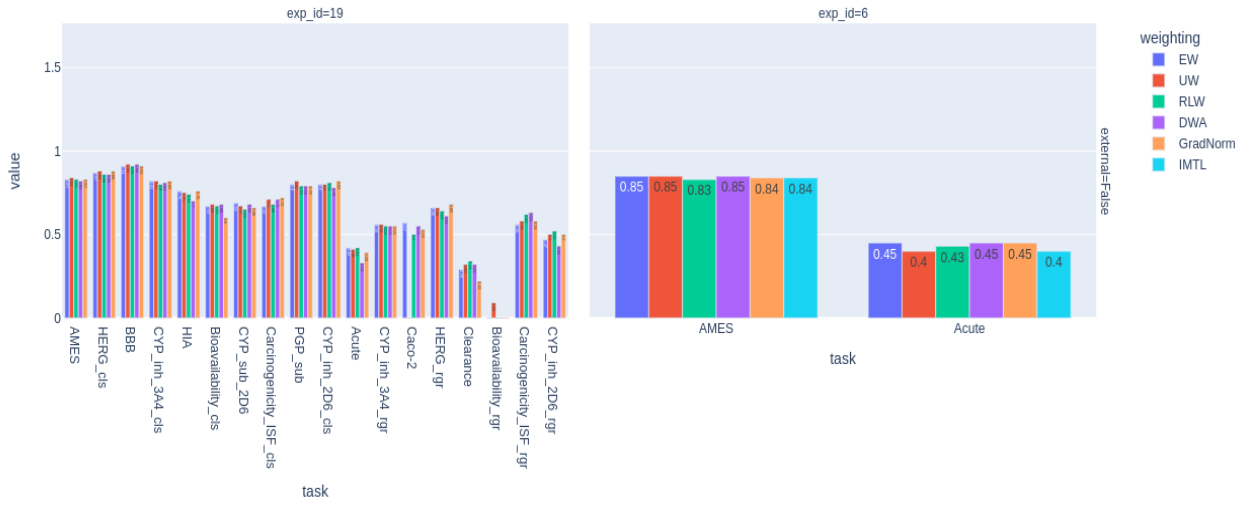

Рисунок 3.11 – Метрики на тестових частинах датасетів з тренувального набору експериментів 6 і 19

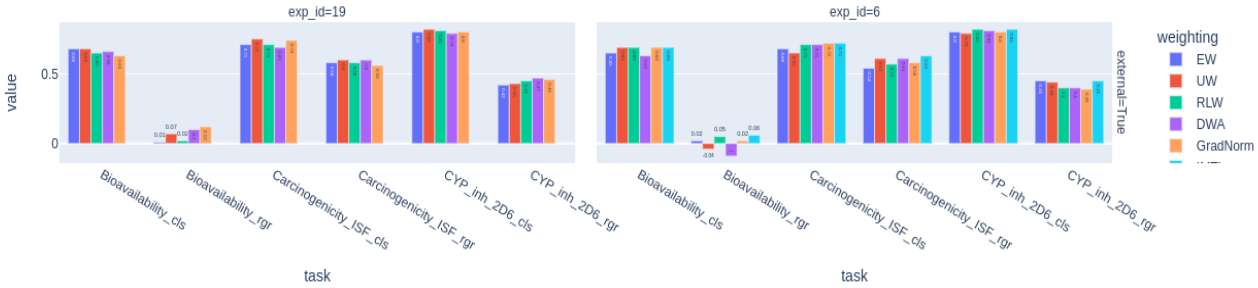

Рисунок 3.12 – Метрики на тестових частинах датасетів із зовнішнього набору експериментів 6 і 19

Висновки:

1. як бачимо, висновки з попередньої пари експериментів справджуються і в цьому випадку;
2. варто зауважити те, метрики на зовнішніх датасетах свідчать про те, що узагальнююча якість при тренуванні на 2 датасетах є достатньо високою. Включення датасетів різної якості може несуттєво негативно впливати на метрики на зовнішніх вибірках.

3.2.4.4 Вплив обрання датасетів, які містять як класифікаційні, так і регресійні значення таргету на якість моделі

Як зазначалося раніше, у нас присутні датасети, в яких відомими є одночасно і бінарні, і регресійні значення таргет-змінної для кожної молекули. Тому, ми висунули гіпотезу, що тренування на таких наборах може допомогти нейронній мережі побудувати якісніше векторне представлення молекули і в результаті підвищити метрики.

Для перевірки гіпотези проаналізуємо наступні пари експериментів (рис. 3.13-3.14).

Експеримент 17 (містить датасети високої якості з бінарними і регресійними таргет-значеннями для тих же молекул) і експеримент 9 (регресійні і класифікаційні датасети різні, присутні датасети низької якості).

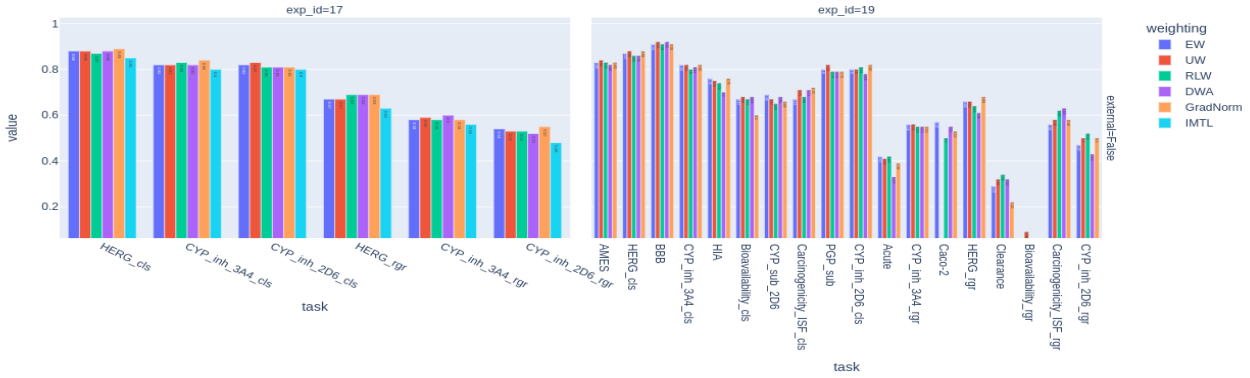

Рисунок 3.13 – Метрики на тестових частинах датасетів з тренувального набору експериментів 9 і 17

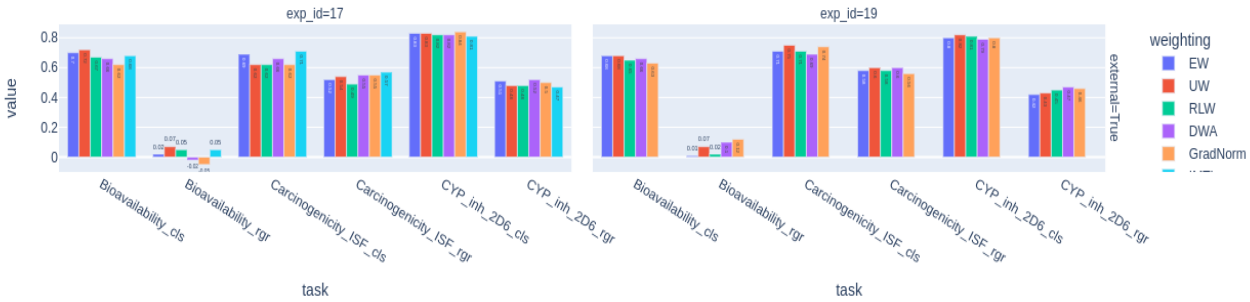

Рисунок 3.14 – Метрики на тестових частинах датасетів із зовнішнього набору експериментів 9 і 17

Висновки:

1. такий тренувальний набір дійсно дещо покращує метрики на датасетах, використаних під час тренування, а також на зовнішніх, що свідчить про поліпшення узагальнюючої здатності моделі.

Розглянемо також пару експериментів 17 та 8 (рис. 3.15-3.16).

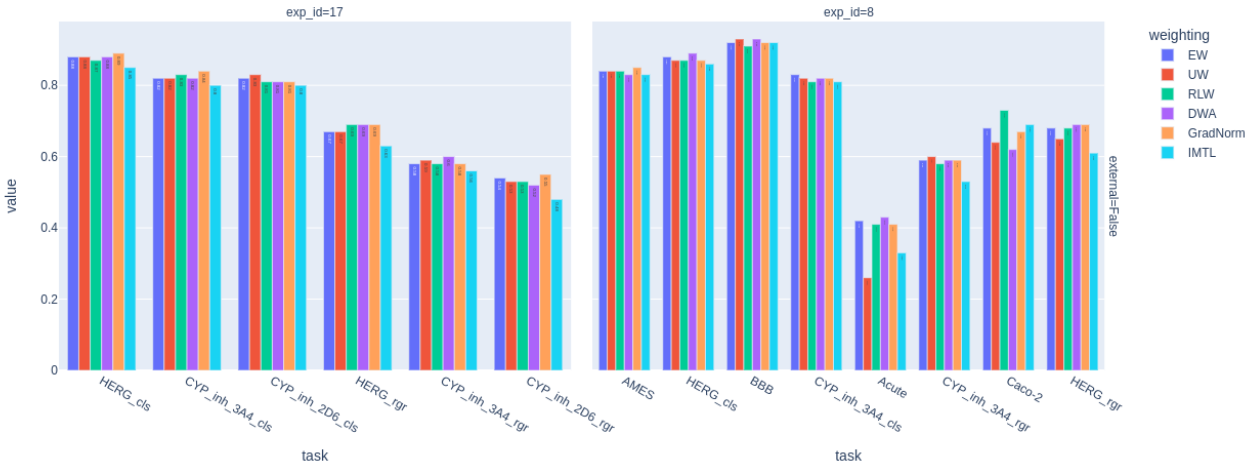

Рисунок 3.15 – Метрики на тестових частинах датасетів з тренувального набору експериментів 17 і 8

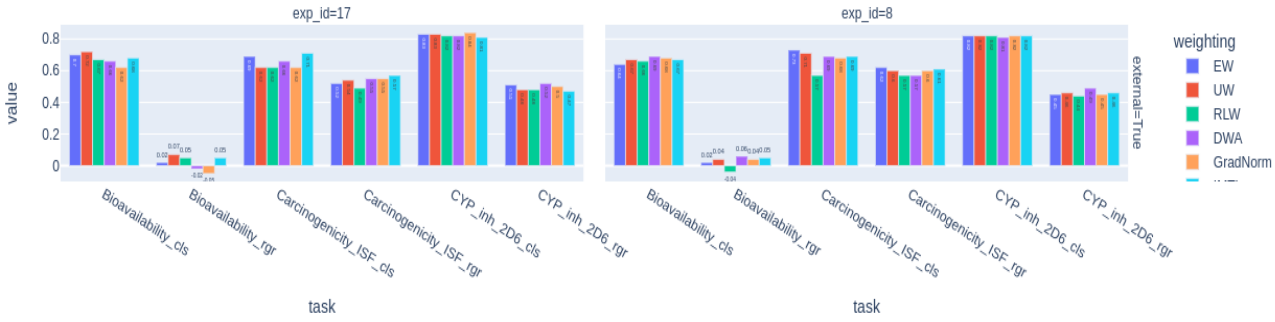

Рисунок 3.16 – Метрики на тестових частинах датасетів із зовнішнього набору експериментів 17 і 8

Висновки:

1. із отриманих результатів маємо, що тренування виключно на датасетах із одночасно регресійними та бінарними таргет-змінними не є принциповим. Вирішальну роль грає саме наявність у тренувальному наборі як класифікаційних датасетів, так і регресійних, при чому високої якості.

3.2.5 Висновки за результатами проведених експериментів по вибору методу багатозадачного тренування

Тренування багатозадачної нейронної мережі на наявних молекулярних датасетах є стабільним.

Найкращі метрики досягаються, якщо включати в тренувальний набір, як регресійні, так і класифікаційні датасети високої якості і приблизно одного розміру. Тобто, гіпотеза висунута на початку дослідження, підтвердилася – для отримання найкращих результатів використовувати всі наявні датасети для багатозадачного тренування не потрібно.

Аналізуючи результати не було виявлено жодного методу багатозадачного навчання, який би систематично давав кращі результати. Тому для того, щоб обрати єдиний метод, який буде використовуватися для навчання фінальної моделі, було прийнято рішення зібрати статистику по всіх експериментах і визначити, які саме методи найчастіше давали для кожної із задач найвищі результати.

Отримані результати наведені у таблиці 3.3.

Таблиця 3.3 – Кількість разів, коли метод багатозадачного тренування дав найкращі з-поміж інших методів результати для датасету

Метод	Тестові вибірки		Валідаційні вибірки		Всього
	Тренувальний набір	Зовнішній набір	Тренувальний набір	Зовнішній набір	
DWA	16	19	17	19	71
EW	15	22	18	23	78
GradNorm	24	17	25	11	77
IMTL	8	30	4	42	84
RLW	18	15	11	14	58
UW	15	25	21	23	84

Отож, топ-3 методи, які найчастіше давали найкращі результати це IMTL, UW та EW. З огляду на те, що IMTL має чітко виражене зміщення в сторону зовнішніх наборів та є найповільнішим з усіх методів, його кандидатуру вирішено було відхилити. Між UW та EW немає принципової відмінності в статистиці, проте EW – найшвидший метод для тренування, і найпростіший. У свою чергу ідея підходу UW виглядає менш робастною до потенційно нестандартних комбінацій функцій втрат. Тому, було прийнято рішення тренувати багатозадачну нейронну мережу саме звичайним EW методом.

3.3 Побудова результуючої графової багатозадачної нейронної мережі та її порівняння із класичними моделями машинного навчання

У попередніх підрозділах даної роботи ми змогли підібрати найкращу графову архітектуру, а також обрати найкращий підхід для тренування багатозадачної нейронної мережі. Прийшов час натренувати фінальну модель на 41 задачі (26 бінарних класифікацій та 15 регресій). Варто нагадати, що для тренування багатозадачності, у процесі якого ми отримуємо універсальний графовий енкодер, ми використовували не всі задачі, а лише деяку підмножину (вибрану також експериментальним шляхом). Найкраща підмножина виявилася доволі простою – датасети із найбільшою кількістю молекул (бінарні класифікації: AMES, BBB, CYP2C9 inhibitor, CYP2D6 inhibitor, CYP3A4 inhibitor, hERG; регресії: Acute, CYP2C9 inhibitor, CYP2D6 inhibitor, CYP3A4 inhibitor, hERG). Далі ми використовували вже навчений графовий енкодер (на задачах наведених вище) і використовували його вихід (у нашому випадку для кінцевої архітектури – це 512 нейронів) для транування інших задач, використовуючи лише лінійні шари. Таким чином ми отримали багатозадачну модель, яка на вхід приймає молекулу, перетворює її у математичний граф, кодує за допомогою енкодера у вектор фіч розмірності 512, потім “проганяє” ці

512 фіч через 41 незалежні лінійні частини (під кожну задачу своя лінійна частина) і видає числове значення для конкретної задачі (готовий результат для регресії й логіту для бінарної класифікації, яку, якщо прогнати через сігмоїду, то отримаємо ймовірність приналежності до класу 1).

Як ми неодноразово наголошували вище, основна мета цієї нейронної мережі – це її швидкодія при не втраченій точності в порівнянні із класичними моделями машинного навчання. Нагадаємо, що раніше (ці тренування не стосуються цієї роботи) ми тренували кожну із 41 задачі окремо за допомогою моделей класичного МЛ (деревоподібні, лінійні моделі, звичайні персептрони й тд). При такому підході для кожної моделі вибиралися власні найкращі фічі в процесі тренування і їх генерація окремо для кожної моделі займала дуже багато часу. Через це передбачення на нових молекулах теж відбувалося доволі довго. Саме через цю проблему ми й вирішили натренувати багатозадачну нейронну мережу, яка б видавала нам універсальний набір фіч під усі задачі. Проте на кожному етапі ми порівнювали результати нашо моделі із результатами, отриманими за допомогою класичного МЛ, оскільки саме не втрачена точність є одним із критеріїв успішності побудованої моделі. У табл. 3.9 якраз зображено всі задачі, їх основні метрики для класичного МЛ і багатозадачної графової нейронної мережі та їх різницю. Для зручності вирішили подати всі задачі в одній таблиці, проте потрібно пам’ятати, що для регресійних задач основною метрикою є R2, у той час як для класифікацій – F1. Також варто відзначити, що представлення різниці – це усереднена різниця для крос-валідаційного й тестового результатів між багатозадачною моделлю та класичним МЛ, тобто чим більше це значення, тим краще справляється наша модель із даною задачею, ніж модель класичного МЛ.

Таблиця 3.9 – Порівняння фінальної графової багатозадачної нейронної мережі й моделей класичного машинного навчання

ADME-Tox param	Problem	CV		Test		Diff
		Classic	MultiTask	Classic	MultiTask	
AMES	binary	0.859	0.851	0.844	0.845	-0.003
Acute	regression	0.515	0.461	0.531	0.425	-0.08
Androgen agonist	binary	0.941	0.922	0.93	0.942	-0.003
Androgen antagonist	binary	0.908	0.906	0.894	0.908	0.006
Androgen binding	binary	0.906	0.887	0.892	0.899	-0.006
BBB	binary	0.9	0.917	0.894	0.924	0.024
Bioavailability	binary	0.773	0.721	0.69	0.651	-0.045
CYP1A2 inhibitor	binary	0.851	0.865	0.84	0.839	0.007
CYP1A2 inhibitor	regression	0.559	0.537	0.584	0.585	-0.01
CYP2C19 inhibitor	binary	0.828	0.858	0.808	0.859	0.041
CYP2C19 inhibitor	regression	0.443	0.499	0.39	0.536	0.101
CYP2C9 inhibitor	binary	0.808	0.83	0.82	0.795	-0.001
CYP2C9 inhibitor	regression	0.477	0.513	0.495	0.392	-0.033
CYP2D6 inhibitor	binary	0.836	0.857	0.843	0.832	0.005
CYP2D6 inhibitor	regression	0.53	0.604	0.567	0.58	0.044
CYP3A4 inhibitor	binary	0.833	0.834	0.842	0.835	-0.003
CYP3A4 inhibitor	regression	0.613	0.623	0.592	0.626	0.022
CYP2C19 substrate	binary	0.712	0.692	0.72	0.742	0.001
CYP2C9 substrate	binary	0.62	0.635	0.608	0.544	-0.024
CYP1A2 substrate	binary	0.722	0.626	0.705	0.778	-0.011
CYP2D6 substrate	binary	0.764	0.715	0.724	0.729	-0.022
CYP3A4 substrate	binary	0.814	0.814	0.788	0.774	-0.007
Caco-2	regression	0.78	0.657	0.715	0.727	-0.055
Carcinogenicity ISF	binary	0.718	0.71	0.636	0.684	0.02

Продовження таблиці 3.9

Carcinogenicity ISF	regression	0.578	0.532	0.436	0.648	0.083
Carcinogenicity OSF	binary	0.722	0.684	0.631	0.644	-0.012
Carcinogenicity OSF	regression	0.374	0.495	0.363	0.4	0.079
Plasma Clearance	regression	0.382	0.252	0.168	0.398	0.05
Renal Clearance	regression	0.312	0.222	0.149	0.401	0.081
DILI	binary	0.802	0.758	0.793	0.742	-0.047
Estrogen binding	binary	0.9	0.881	0.843	0.85	-0.006
Estrogen agonist	binary	0.479	0.522	0.479	0.551	0.058
Estrogen antagonist	binary	0.9	0.879	0.827	0.85	0.001
hERG	binary	0.879	0.886	0.884	0.886	0.005
hERG	regression	0.701	0.675	0.741	0.702	-0.032
HIA	binary	0.657	0.742	0.622	0.762	0.113
Half-life	regression	0.354	0.295	0.26	0.209	-0.055
PPB	regression	0.742	0.71	0.686	0.732	0.007
Pgp inhibitors	binary	0.914	0.886	0.888	0.875	-0.02
Pgp substrates	binary	0.855	0.819	0.896	0.82	-0.056
VD	regression	0.477	0.486	0.468	0.533	0.037

Як ми бачимо, більшість задач мають доволі непогані метрики й невелику різницю із моделями класичного МЛ. Серед найгірших це Acute, Caco-2, Half-life та Pgp substrates. Серед найкращих це регресійний CYP2C19 inhibitor та HIA. Загалом можемо виділити таку тенденцію, що регресійні задачі навчаються трохи гірше в порівнянні із класифікаційними. У табл. 3.10 наведено узагальнене порівняння між нашою моделлю й класичним підходом (просто середнє для стовпця Diff із таблиці вище).

Таблиця 3.10 – Узагальнене порівняння фінальної графової багатозадачної нейронної мережі й класичного підходу

	MultiTask - Classic
CV	-0.0117
Test	0.0236
All	0.006

Як ми бачимо, середня різниця між нашою моделлю й моделями класичного МЛ є мінімально від'ємною на крос-валідації й мінімально додатною на тесті. Тобто можемо зробити висновок, що в середньому наша модель має точність таку саму, як і моделі класичного МЛ. Проте варто наголосити, що такий підхід дає нам виграш у швидкості в 100 разів, тобто передбачення ADME-Tox властивостей за допомогою побудованої нами багатозадачної графової нейронної мережі відбувається в 100 разів швидше, ніж окремими моделями класичного машинного навчання. А саме, для 500 тисяч молекул наша модель видає прогноз усього лише за 280 секунд на одній машині із графовим процесором. Отже, основну мету, а саме побудову такої моделі, щоб сильно виграти в швидкості й не програти в точності порівняно із класичним МЛ, можемо вважати повністю виконаною.

Наступними кроками проведемо більш детальний аналіз. У табл. 3.11 та 3.12 наведено всі основні метрики для класифікаційних та регресійних задач відповідно.

Таблиця 3.11 – Метрики для класифікаційних задач фінальної моделі

ADME-Tox param	F1		Accuracy		Precision		Recall	
	CV	Test	CV	Test	CV	Test	CV	Test
Bioavailability	0.721	0.651	0.7	0.656	0.683	0.676	0.766	0.627
HIA	0.742	0.762	0.942	0.943	0.825	0.8	0.677	0.727
Pgp inhibitors	0.886	0.875	0.881	0.862	0.905	0.844	0.869	0.908

Продовження таблиці 3.11

Pgp substrates	0.819	0.82	0.817	0.816	0.813	0.803	0.825	0.838
BBB	0.917	0.924	0.906	0.915	0.908	0.925	0.926	0.923
CYP1A2 inhibitor	0.865	0.839	0.848	0.827	0.85	0.866	0.881	0.814
CYP2C9 inhibitor	0.83	0.795	0.805	0.788	0.782	0.831	0.885	0.762
CYP2C19 inhibitor	0.858	0.859	0.848	0.838	0.859	0.804	0.857	0.923
CYP2D6 inhibitor	0.857	0.832	0.847	0.824	0.839	0.834	0.877	0.83
CYP3A4 inhibitor	0.834	0.835	0.809	0.796	0.826	0.772	0.841	0.908
CYP1A2 substrate	0.626	0.778	0.862	0.921	0.644	0.817	0.614	0.742
CYP2C9 substrate	0.635	0.544	0.885	0.871	0.634	0.607	0.641	0.493
CYP2C19 substrate	0.692	0.742	0.9	0.91	0.709	0.708	0.678	0.78
CYP2D6 substrate	0.715	0.729	0.883	0.886	0.731	0.718	0.703	0.74
CYP3A4 substrate	0.814	0.774	0.851	0.828	0.803	0.802	0.825	0.747
AMES	0.851	0.845	0.827	0.827	0.816	0.839	0.889	0.851
Carcinogenicity ISF	0.71	0.684	0.871	0.865	0.781	0.788	0.655	0.605
Carcinogenicity OSF	0.684	0.644	0.777	0.764	0.685	0.691	0.685	0.603
Androgen antagonist	0.906	0.908	0.925	0.924	0.905	0.886	0.908	0.931
Androgen agonist	0.922	0.942	0.957	0.967	0.944	0.926	0.902	0.958
Androgen binding	0.887	0.899	0.908	0.917	0.865	0.862	0.911	0.94
Estrogen antagonist	0.879	0.85	0.95	0.94	0.961	0.958	0.811	0.764
Estrogen agonist	0.522	0.551	0.846	0.861	0.658	0.731	0.433	0.442
Estrogen binding	0.881	0.85	0.952	0.94	0.97	0.958	0.808	0.764
hERG	0.886	0.886	0.865	0.867	0.875	0.885	0.897	0.887
DILI	0.758	0.742	0.686	0.689	0.671	0.696	0.874	0.795

Таблиця 3.12 – Основні метрики для регресійних задач фінальної моделі

ADME-Tox param	R2		MSE		MAE	
	CV	Test	CV	Test	CV	Test
Caco-2	0.657	0.727	0.2	0.166	0.345	0.302
PPB	0.71	0.732	301.07	280.3	11.38	10.84
VD	0.486	0.533	1.084	1	0.799	0.766
Half-life	0.295	0.209	1.316	1.583	0.876	0.974
Plasma Clearance	0.252	0.398	1.496	1.222	0.923	0.854
Renal Clearance	0.222	0.401	2.761	2.319	1.037	0.924
Acute	0.461	0.425	1.993	2.204	1.059	1.083
CYP1A2 inhibitor	0.537	0.585	1.753	1.467	0.925	0.86
CYP2C9 inhibitor	0.513	0.392	1.19	1.457	0.786	0.785
CYP2C19 inhibitor	0.499	0.536	1.247	1.175	0.793	0.763
CYP2D6 inhibitor	0.604	0.58	1.253	1.374	0.744	0.788
CYP3A4 inhibitor	0.623	0.626	1.356	1.374	0.798	0.791
Carcinogenicity ISF	0.532	0.648	12.425	11.314	2.122	1.79
Carcinogenicity OSF	0.495	0.4	15.365	14.029	2.866	2.67
hERG	0.675	0.702	1.418	1.224	0.778	0.801

Як ми бачимо, усі значення додаткових метрик для всіх задач переважно говорять про те саме, що й основні метрики, аналіз яких ми робили вище.

Висновки

У ході даного розділу ми змогли проаналізувати наявні датасети для вирішення поставленої проблеми, знайшли можливий перетин усіх

властивостей по молекулах і врахували це при поділі на тренувальні, валідаційні й тестові вибірки.

Також ми обрали найкращу графову архітектуру за допомогою тюнінгу гіперпараметрів у бібліотеці Python optuna на невеликому наборі датасетів. Найкращою архітектурою виявився графовий підхід MF із 3 графовими шарами із 128, 256 та 512 нейронами відповідно. Лінійна ж частина складалася із 2 шарів по 512 нейронів та дропауту між ними, що дорівнював 0.15.

Наступним етапом стало обрання найкращого підходу до побудови мультизадачності. Ним виявився звичайний підхід із зрівноваженим зважуванням EW.

Як результат, ми змогли побудувати графову багатозадачну нейронну мережу для передбачення ADME-Tox властивостей молекул, яка не поступається в точності моделям класичного машинного навчання й переважає ці моделі у швидкості в 100 разів.

РОЗДІЛ 4 РОЗРОБКА ВЛАСНОГО СТАРТАП-ПРОЄКТУ

На даний момент сфера біотехнологій лише починає завойовувати свою нішу в сучасному світі. І для цього їй приходить на допомогу машинне навчання й нейронні мережі. Саме за допомогою глибинних нейронних мереж інженери можуть побудувати моделі, які здатні в тисячі разів пришвидшити генерацію корисних молекул. Замість того, щоб штучно синтезувати по кілька молекул у день біологами в лабораторіях, нейронна мережа здатна відфільтрувати мільйони за кілька годин й залишити лише ті, що дійсно варті уваги й можуть принести користь. Саме цю задачу вирішує наша побудована графова багатозадачна нейронна мережа для передбачення ADME-Tox властивостей молекул.

Все це означає, що дану нейронну мережу можна вважати основою для побудови власного стартап-проєкту, який завоює ринок біотехнологій у недалекому майбутньому.

4.1 План розробки стартапу та масштабування його на ринок

Наведемо план розробки стартапу та виведення його на ринок. Спочатку треба провести маркетинговий аналіз, який включає в себе:

- конкурентний аналіз, щоб зрозуміти, якими методами вирішення проблем вже користуються люди;
- формування ідеї самого проекту та виділення цільової аудиторії;
- розробити стратегію виведення товару на ринок, базуючись на аналізі ринкового середовища.

Наступним кроком є організація самого стартапу. На цьому етапі мають бути:

- складений весь план та побудований таймлайн розробки та запуску продукту;
- запланований обсяг виробництва та оцінений потенційний обсяг ресурсу, який буде потрібен для виконання плану;
- розраховані витрати, необхідні для реалізації проекту, та витрати на запуск проекту.

Далі необхідно виконати фінансово-економічний аналіз та оцінити ризики стартап-проекту, у межах якого:

- визначити обсяг інвестиційних втрат;
- розрахувати основні фінансово-економічні показники проекту (собівартість, ціну продукту/послуги, податковий збір та чистий прибуток) та визначити показники інвестиційної привабливості проекту (рентабельність продажів, період окупності проекту);
- визначити основні ризики проекту та способи для їх запобігання.

Фінальним кроком являється розробка заходів з комерціалізації продукту. Цей крок є важливим для масштабування та збільшення розмірів продукту.

Для того, щоб залучити інвесторів та знайти різні способи фінансування проекту, необхідно:

- провести дослідження на предмет інтересів потенційних інвесторів та бізнесів;
- скласти інвестиційну пропозицію, яка включає в себе як опис самого продукту та його теперішні розміри, так і можливі шляхи розширення та розвитку;
- обрати канали комунікації із потенційно зацікавленими персонами.

4.2 Опис ідеї стартап-проєкту

Стартап-проєкт полягає у вирішенні проблеми швидкої генерації ADME-Tox властивостей молекул. Суть продукту стартапу полягає у тому, що система бере на вхід необмежену кількість молекул, обробляє їх і видає значення бажаних для користувача ADME-Tox властивостей. У табл. 4.1 наведена інформаційна карта стартапу.

Таблиця 4.1 – Інформаційна карта стартап-проєкту

Назва проєкту	FastADMET
Автори проєкту	Возняк Володимир Зіновійович та Островський Захар Юрійович
Коротка анотація	Платформа буде обробляти вхідні молекули за допомогою графової багатозадачної нейронної мережі й видавати для користувача бажаний набір спрогнозованих ADME-Tox властивостей
Термін реалізації проєкту	12 місяців
Необхідні ресурси	Приміщення з комп'ютерами, доступом до інтернету та електромережі; програмне забезпечення для розробки, хмарне сховище для даних, антивірусні програми; фінансові кошти на оплату заробітної плати виконавцям на термін 12 місяців, а також на такі витрати як: оренда приміщення, комунальні послуги, оренда хмарного сховища тощо
Опис проблеми, яку вирішує проєкт	Швидка генерація ADME-Tox властивостей молекул

Продовження таблиці 4.1

Головні цілі та завдання проєкту	Метою проєкту є створення платформи, яка буде автоматично обробляти мільйони молекул наданих користувачем та надавати передбачені ADME-Tox властивостей
Очікувані результати	Привернення технологічних компаній до нашого стартапу, автономна система для обробки молекул

4.3 Технологічний аудит ідеї проєкту

Тепер можна розібрати ідею стартапу та провести конкурентний аналіз. У таблиці 4.2 наведений опис ідеї стартапу.

Таблиця 4.2 – Опис ідеї стартап-проєкту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Основна ідея заключається в тому, щоб приймати на вхід молекули й за допомогою графової багатозадачної нейронної мережі швидко прогнозувати ADME-Tox властивості молекул	Опрацювання молекул і передбачення для них ADME-Tox властивостей	Користувач зможе отримати спрогнозовані ADME-Tox властивості на власний розсуд для власних молекул
	Власна генерація молекул на основі ADME-Tox властивостей	Користувач зможе отримати вже конкретні молекули згенеровані нашою платформою під специфічні задачі

Далі проведемо порівняльний аналіз конкурентів проекту та наведемо результати у таблиці 4.3.

Таблиця 4.3 – Порівняльний аналіз конкурентів проєкту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів				W	N	S
		Власний проєкт	AtomWise	Insilico	Biolexis			
1	Якість прогнозованих ADME-Tox властивостей	Хороші результати	Хороші резуль-тати	Хороші резуль-тати	Хороші резуль-тати		+	
2	Швидкодія	Висока	Середня	Середня	Повільна			+
3	Доступність	Висока	Власні проєкти	Висока	Комерція		+	

Далі аналізуємо реальність технічно здійснити ідею проєкту. Усю інформацію наведено в табл. 4.4.

Таблиця 4.4 – Технологічна здійсненність продукту

№ п/п	Ідея проєкту	Технології й реалізації	Наявність технологій	Доступність технологій
1	Створення власної платформи для швидкої обробки молекул і прогнозування ADME-Tox властивостей за допомогою графової багатозадачної нейронної мережі	Мова програмування Python	Наявні	Доступні
2		Мова програмування R	Наявні, необхідні доопрацювання	Доступні
3		Мова програмування Go	Не наявні	Доступні
Обрана технологія реалізації проєкту: Python				

4.4 Аналіз ринкових можливостей запуску стартап-проєкту

Далі проведемо попередній аналіз ринку для запуску стартап-проєкту. Усю інформацію наведено в табл. 4.5.

Таблиця 4.5 – Попередня характеристика потенційного ринку стартапу

№ п/п	Показники ринку (найменування)	Характеристика
1	Кількість головних гравців, од	3
2	Загальний обсяг продаж, грн/ум.од	15000
3	Динаміка ринку (якісна оцінка)	Позитивна, зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Наявність патентів
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6	Середня норма рентабельності в галузі (або по ринку), %	16%

Тепер проведемо характеристику потенційних клієнтів, які можуть бути зацікавлені в проєкті. Усю інформацію наведено в табл. 4.6.

Таблиця 4.6 – Характеристика потенційних клієнтів стартап-проєкту

№ п/п	Потреби, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Швидкий прогноз ADME-Tox властивостей для невеликої кількості молекул	Особисте використання користувачем	Більше цікавить швидкість, ніж якість	Простота використання

Продовження таблиці 4.6

2	Прогноз ADME-Tox властивостей для великої кількості молекул	Компанії для генерації власних молекул	Більше цікавить якість, проте швидкість теж важлива	Швидкодія, якість, ціна
3	Генерація власних молекул	Лабораторії	Якість є надзвичано важливою	Якість прогнозу

Обрахуємо фактори загроз (таблиця 4.7) та можливостей (таблиця 4.8). Проаналізуємо загрози, щоб зрозуміти можливі перешкоди при запуску продукту на ринок. Фактори можливостей же треба обрахувати, щоб знати усі сприятливі умови та по можливості ними скористатися.

Таблиця 4.7 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Конкуренція	Хоча ринок є відкритим і неосвоїним, на ньому вже є кілька великих гравців, які відомі на ринку і мають свою цільову групу покупців	Знайти точки додаткової цінності для користувача
2	Ціна збуту	Конкуренти можуть коштувати менше через нижчу якість	Сфокусуватися на якості роботи застосунку та продумати маркетингову стратегію

Продовження таблиці 4.7

3	Якість аналізу	Через комплексність задачі, модель може мати проблеми на певних кластерах молекул, на яких вона не тренувалася	Мати достатній штаб і ресурси, для постійної вдосконалення нейронної мережі, щоб покривати всі можливі кластери молекул

Таблиця 4.8 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Універсальнісь	Продукт не залежить від апаратної платформи як у більшості конкурентів	Зробити акцент при маркетингу, продовжувати розвиток як окремого продукту
2	Простота у використанні	Від користувача треба лише завантажити файл із молекулами	Реалізувати зручний інтерфейс для завантаження
3	Якість та гарантії	Надавати найбільш якісні послуги та сервіси	Пропонувати моделі з найкращими результатами, а також надавати усю необхідну технічну підтримку
4	Безкоштовний сервіс при MVP	Максимально швидко набрати базу своїх клієнтів та заявити про себе на ринку	Розгорнути широкий маркетинг, а також активно боротися за клієнтів конкурентів

Далі розглянемо питання конкуренції, а саме визначимо її тип та рівень (таблиця 4.9).

Таблиця 4.9 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	У чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Тип конкуренції: недосконала конкуренція	Представлено мало продуктів та експертів	Зробити максимальним збут застосунку
2. За рівнем конкурентної боротьби: міжнародний	Наявні проєкти, розроблені та можуть бути доступні у всьому світі	Розширити цільову аудиторію, розробити інтерфейс на різних мовах
3. За галузевою ознакою: внутрішньогалузева	Можуть працювати з різними галузями	Покращити персоналізацію
4. Конкуренція за видами товарів: товарно-родова	Конкуренція з аналізами інших систем та експертів	Підтримувати та покращувати якість існуючих функцій
5. За характером конкурентних переваг: нецінова	Різні компанії пропонують різну якість	Розробляти якісніші алгоритми і моделі
6. За інтенсивністю: марочна	Вже представлені компанії із сильним брендом	Предметно створити комунікаційну стратегію для вибудови свого бренду

Далі необхідно виконаємо аналіз конкуренції за моделлю 5 сил конкуренції Майкла Портера (таблиця 4.10).

Таблиця 4.10 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товаровиробники
	Інші існуючі системи та продукти	Якість, ціни, кількість користувачів, капіталовкладення	Фактори сили постачальників	Контроль якості, порівняння цін	Сила бренду, якість, ціна, масштаби
Висновки	Конкуренція з невеликою інтенсивністю, а також підігрітий ринок	Можливості входження на ринок, нові потенційні конкуренти	Постачальники відсутні	Клієнти не диктують умови роботи на ринку	Товаровиробники відсутні

Маючи результати аналізу конкуренції (таблиця 4.10), характеристики ідеї стартап-проєкту (таблиця 4.5), характеристики потенційних клієнтів і їх вимоги до продукту (таблиця 4.6) та фактори ринкового середовища (таблиці 4.7 і 4.8) було сформульовано та обґрунтовано перелік факторів конкурентоспроможності (таблиця 4.11).

Таблиця 4.11 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Універсальність	Продукт не залежить від апаратної платформи як у більшості конкурентів

Продовження таблиці 4.11

2	Простота у використанні	Від користувача треба лише завантажити файл із молекулами
3	Якість та гарантії	Надавати найбільш якісні послуги та сервіси
4	Безкоштовний сервіс при MVP	Максимально швидко набрати базу своїх клієнтів та заявити про себе на ринку

Тепер можна провести аналіз сильних та слабких сторін продукту (таблиця 4.12).

Таблиця 4.12 – Порівняльний аналіз сильних та слабких сторін системи

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів						
			-3	-2	-1	0	1	2	3
1	Універсальність	20	+						
2	Простота у використанні	10			+				
3	Якість та гарантії	18		+					
4	Безкоштовний сервіс при MVP	15		+					

Далі проведемо SWOT-аналіз продукту (таблиця 4.13).

Таблиця 4.13 – SWOT-аналіз стартап-проєкту

Сильні сторони Універсальність Простота у використанні Якість та гарантії Безкоштовний сервіс при MVP	Слабкі сторони Відсутність сильного бренду Не сформована база клієнтів Не підключені альтернативні канали маркетингу
Можливості Покращення системи Персоналізація Інтеграція з лабораторіями	Загрози Нові системи та експерти Збут

Дякуючи проведенню SWOT-аналізу, ми змогли визначити сильні та слабкі сторони, можливості та загрози, пов'язані з конкуренцією та плануванням стартап-проекту. Далі спроєктуємо альтернативну ринкову поведінку для інтеграції стартап-проєкту на ринок та приблизний час реалізації системного комплексу, з урахуванням потенційних проектів, що можуть бути виведені на ринок та наведемо результати у таблиці 4.14.

Таблиця 4.14 – Альтернативи ринкового впровадження стартап проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Вихід на ринок з нижчою якістю	40%	3 місяці
2	Пропонувати одразу платне використання	60%	8 місяців
3	Представлення користувачам системи без інтерфейсу	50%	6 місяців

У даному пункті був проведений детальний аналіз ринку та продукту. Також відповідно до результатів проведеного конкурентного аналізу, визначених факторів ринку та його сприятливість, описання ідеї та характеристик стартап-проєкту, робимо висновок, що найкраще зробити модель із високою точністю й гарним інтерфейсом, витративши більше часу, проте одразу ж пропонувати користувачам платне користування нашою платформою.

4.5 Розроблення ринкової стратегії стартап-проєкту

Для розробки ринкової стратегії продукту, у першу чергу, необхідно проаналізувати цільову аудиторію проекту (таблиця 4.15).

Таблиця 4.15 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит у межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Персональні користувачі	Низька	10%	Висока	Висока
2	Великі бізнеси	Висока	50%	Середня	Висока
3	Малі та середні бізнеси	Висока	35%	Висока	Середня
4	Держава	Низька	5%	Низька	Висока
Які цільові групи обрано: 2, 3					

Маючи аналіз цільових груп, далі визначимо базову стратегію розвитку продукту (таблиця 4.16).

Таблиця 4.16 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	2 та 3	Диференційованого маркетингу	Масштабування та максимізація	Оптимальних витрат

Для роботи в обраних сегментах ринку сформовано базову стратегію розвитку (таблиці 4.17 та 4.18).

Таблиця 4.17 – Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
Ні	Так	Ні	Виклику лідера

Таблиця 4.18 – Визначення стратегії позиціонування

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
Універсальність Простота у використанні Якість результатів	Оптимальних витрат	Універсальність Простота у використанні Якість та гарантії Безкоштовне використання при MVP	Система, яка краще всіх покращує якість користувацьких зображень Система з простим інтерфейсом

4.6 Розроблення маркетингової програми стартап-проєкту

Після проведеного комплексного аналізу, можемо повноцінно описати ключові переваги концепції потенційного товару (таблиця 4.19) та побудувати концепцію маркетингових комунікацій (таблиця 4.20).

Таблиця 4.19 – Ключові переваги концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Якість прогнозу	Якісно спрогнозовані ADME-Tox властивості молекул	Постійне покращення та перенавчання моделей, які зможуть аналізувати більше кластерів молекул
2	Швидкодія	Швидкість прогнозу ADME-Tox властивостей молекул	Обробка великої кількості молекул і швидкий прогноз ADME-Tox властивостей молекул
3	Простий інтерфейс	Система дуже проста у використанні	Система із інтуїтивно зрозумілим інтерфейсом, який вимагає всього лише завантаження файлу із молекулами

Таблиця 4.20 – Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламно о повідомлення	Концепція рекламного звернення
1	Пошук спеціалізованих систем	b2b продажі Зв'язок через теплі контакти Таргетована реклама у соціальних мережах Публікація в спеціалізованих виданнях, журналах	Точність Якість Універсальність	Поєднати повідомлення про те, що це якісна система, яка є незалежною	Таргетована реклама на цільову аудиторію
2	Пошук доступного та дешевого продукту	Рекламні банери в Інтернеті, форуми, реклами від інфлюенсерів	Простота Безкоштовне використання MVP	Вселити довіру у бренд та продукт	Реклама у лідерів думок Вивіски в публічних місцях Таргетована реклама на цільову аудиторію

Висновки

Даний розділ був присвячений дослідженню стартап-проєкту. У якості такого була представлена платформа для передбачення ADME-Tox властивостей молекул за допомогою графової багатозадачної нейронної мережі.

У рамках розділу було досліджено розробку стратегій виходу на ринок та маркетинг-стратегії для цього. Зокрема, даний ринок являється сприятливим з невеликою кількістю представлених компаній конкурентів. Оскільки вони дають лише частину функцій, а запропонована система є універсальною та доступною, то у стартап-проєкту є всі шанси стати монополістами на ринку.

Також були опрацьовані сильні та слабкі сторони проекту, SWOT аналіз, аналіз конкурентів та цільової аудиторії. На основі всіх досліджень був сформований концепт маркетингової стратегії для обраних цільових аудиторій.

ВИСНОВКИ

У першому розділі даної роботи нами було досліджено поняття ADME-Tox властивостей, чому вони були і є надзвичайно важливими у контексті відкриття та розробки ліків. Ці властивості допомагають визначити життєздатність препарату-кандидата. Підраховано, що близько 50% кандидатів на ліки не проходять через неприйнятну ефективність і що до 40% кандидатів на ліки в минулому зазнали невдачі через токсичність. Такі ліки, як мібефрадил, сорувідин та фенілпропаноламін гідрохлорид, були зняті з ринку через взаємодію ліків або токсичність. Як для регуляторів, так і для виробників лікарських засобів стало очевидним, що на додаток до фармакологічних властивостей дослідження ADME-Tox відіграють вирішальну роль в успіху кандидата на ліки. У зв’язку з цим впливом на кінцевий успіх ці дослідження зараз проводяться на початку процесу відкриття ліків.

У другому розділі ми провели детальний аналіз моделей графових нейронних мереж, а також можливих підходів до побудови багатотозадачності. MF підхід узагальнює наявні молекулярні признаки, створені вручну, щоб дозволити їх оптимізацію для різноманітних завдань. Роблячи кожну операцію в пайплайні диференційованою, можна використовувати стандартні нейронномережеві методи навчання для масштабованої оптимізації параметрів цих нейронних молекулярних фінгерпринтів. Було продемонстровано інтерпретацію та прогнозну продуктивність цих нових фінгерпринтів. У результаті проведеного аналізу, на нашу думку, стосовно вибору найкращого методу для тренування багатозадачної нейронної мережі, то це IMTL, який, до речі, є одним із останніх опублікованих. Ця стаття заслуговує особливої уваги й довіри, адже автори дуже ретельно аналізують існуючі підходи та їх недоліки, на основі цього пропонують дуже цікаву власну ідею, яка долає вразливості попередніх алгоритмів, і зрештою наводять дуже детальне порівняння ефективності свого методу з існуючими.

У ході третього розділу ми змогли проаналізувати наявні датасети для вирішення поставленої проблеми, знайшли можливий перетин усіх властивостей по молекулах і врахували це при поділі на тренувальні, валідаційні й тестові вибірки. Також ми обрали найкращу графову архітектуру за допомогою тюнінгу гіперпараметрів у бібліотеці Python optuna на невеликому наборі датасетів. Найкращою архітектурою виявився графовий підхід MF із 3 графовими шарами із 128, 256 та 512 нейронами відповідно. Лінійна ж частина складалася із 2 шарів по 512 нейронів та дропауту між ними, що дорівнював 0.15. Наступним етапом стало обрання найкращого підходу до побудови мультизадачності. Ним виявився звичайний підхід із зрівноваженим зважуванням EW. Як результат, ми змогли побудувати графову багатозадачну нейронну мережу для передбачення ADME-Tox властивостей молекул, яка не поступається в точності моделям класичного машинного навчання й переважає ці моделі у швидкості в 100 разів.

У ході четвертого розділу ми проаналізували нашу ідею на доцільність реалізування власного стартап-проєкту. У якості такого була представлена платформа для передбачення ADME-Tox властивостей молекул за допомогою графової багатозадачної нейронної мережі. У рамках розділу було досліджено розробку стратегій виходу на ринок та маркетинг-стратегії для цього. Зокрема, даний ринок являється сприятливим з невеликою кількістю представлених компаній конкурентів. Оскільки вони дають лише частину функцій, а запропонована система є універсальною та доступною, то у стартап-проєкту є всі шанси стати монополістами на ринку. Також були опрацьовані сильні та слабкі сторони проекту, SWOT аналіз, аналіз конкурентів та цільової аудиторії. На основі всіх досліджень був сформований концепт маркетингової стратегії для обраних цільових аудиторій.

ПЕРЕЛІК ПОСИЛАНЬ

1. Brody S., Alon U., Yahav E. How Attentive are Graph Attention Networks? URL: https://arxiv.org/abs/2105.14491

2. Bruna J., Zaremba W., Szlam A., LeCun Y. Spectral networks and locally connected networks on graphs. URL: https://arxiv.org/abs/1312.6203

3. Chen Z., Badrinarayanan V. GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks. URL: https://arxiv.org/abs/1711.02257

4. Dahl G., Jaitly N., Salakhutdinov R. Multi-task neural networks for QSAR predictions. URL: https://arxiv.org/abs/1406.1231

5. Delaney J. ESOL: Estimating aqueous solubility directly from molecular structure. *Journal of Chemical Information and Computer Sciences*. 2004. Vol. 44, No. 3. P. 1000–1005.

6. Duvenaud D., Maclaurin D., Aguilera-Iparraguirre J., et. al. Convolutional Networks on Graphs for Learning Molecular Fingerprints. URL: https://arxiv.org/abs/1509.09292

7. Gamo F., Sanz L., Vidal J., et al. Thousands of chemical starting points for antimalarial lead identification. *Journal of Nature*. 2010. Vol. 465, No. 7296. P. 305–310.

8. Glem R., Bender A., Arnby C., et. al. Circular fingerprints: flexible molecular descriptors with applications from physical chemistry to ADME. *Journal of Investigational Drugs*. 2006, Vol. 9, No. 3. P. 199–204.

9. Graves A., Wayne G., Danihelka I. Neural Turing machines. URL: https://arxiv.org/abs/1410.5401

10. Hachmann J., Olivares-Amaya R., Atahan-Evrenk S., et. al. The Harvard clean energy project: large-scale computational screening and design of organic photovoltaics on the world community grid. *The Journal of Physical Chemistry Letters*. 2011. Vol. 2, No. 17. P. 2241–2251.

11. Hershey J., Roux J., Weninger F. Deep unfolding: Model-based inspiration of novel deep architectures. URL: https://arxiv.org/abs/1409.2574

12. Hochreiter S., Schmidhuber J. Long short-term memory. *Journal of Neural computation*. 1997. Vol 9, No. 8. P. 1735–1780.

13. Hu W., Liu B., Gomes J., et. al. Strategies for Pre-training Graph Neural Networks. URL: https://arxiv.org/abs/1905.12265

14. Ioffe S., Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. URL: https://arxiv.org/abs/1502.03167

15. Kendall A., Gal Y., Cipolla R. Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics. URL: https://arxiv.org/abs/1705.07115

16. Kingma D., Ba J. Adam: A method for stochastic optimization. URL: https://arxiv.org/abs/1412.6980

17. Kipf T., Welling M. Semi-Supervised Classification with Graph Convolutional Networks. URL: https://arxiv.org/abs/1609.02907

18. Lin B., Ye F., Zhang Y. A Closer Look at Loss Weighting in Multi-Task Learning. URL: https://arxiv.org/abs/2111.10603

19. Liu B., Liu X., Jin X, Stone P., et. al. Conflict-Averse Gradient Descent for Multi-task Learning. URL: https://arxiv.org/abs/2110.14048

20. Liu L., Li Y., Kuang Z., et. al. Towards Impartial Multi-Task Learning. URL: https://openreview.net/forum?id=IMPnRXEWpvr

21. Liu S., Johns E., Davison A. End-to-End Multi-Task Learning with Attention. URL: https://arxiv.org/abs/1803.10704

22. Lusci A., Pollastri G., and Baldi P. Deep architectures and deep learning in chemoinformatics: the prediction of aqueous solubility for drug-like molecules. *Journal of Chemical Information and Modelling*. 2013. Vol. 53, No. 7. P. 1563–1575.

23. Micheli A. Neural network for graphs: A contextual constructive approach. *Journal of Neural Networks*. 2009. Vol. 20, No. 3. P. 498–511.

24. Morgan H. The generation of a unique machine description for chemical structure. *Journal of Chemical Documentation*. 1965. Vol. 5, No. 2. P. 107–113.

25. Oliphant T. Python for scientific computing. *Journal of Computing in Science & Engineering*. 2007. Vol. 9, No. 3. P. 10–20.

26. RDKit: Open-source cheminformatics. URL: www.rdkit.org

27. RECEPTOR.AI Case Studies. URL: https://receptor.ai/case-studies

28. Ramsundar B., Kearnes S., Riley P., Webster D., et. al. Massively multitask networks for drug discovery. URL: https://arxiv.org/abs/1502.02072

29. Rogers R., Hahn M. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*. 2010. Vol. 50, No. 5. P. 742–754.

30. Scarselli F., Gori M., Tsoi A., et. al. The graph neural network model. *Journal of Neural Networks*. 2009. Vol. 20, No. 1. P. 61–80.

31. Tai K., Socher R., Manning C. Improved semantic representations from tree-structured long short-term memory networks. URL: https://arxiv.org/abs/1503.00075

32. Unterthiner T., Mayr A., Klambauer G., Hochreiter S. Toxicity prediction using deep learning. URL: https://arxiv.org/abs/1503.01445

33. Veličković P., Cucurull G., Casanova A., et. al. Graph Attention Networks. URL: https://arxiv.org/abs/1710.10903

34. Veličković P. Theoretical Foundations of Graph Neural Networks. URL: https://petar-v.com/talks/GNN-Wednesday.pdf

35. Wang Z., Tsvetkov Y., Firat O., Cao Y. Gradient Vaccine: Investigating and Improving Multi-task Optimization in Massively Multilingual Models. URL: https://openreview.net/forum?id=F1vEjWK-lH_

36. Xu K., Hu W., Leskovec J., Jegelka S. How Powerful are Graph Neural Networks? URL: https://arxiv.org/abs/1810.00826

37. Yu T., Kumar S., Gupta A., et. al. Gradient Surgery for Multi-Task Learning. URL: https://arxiv.org/abs/2001.06782

38. Zaheer M., Kottur S., Ravanbakhsh S, et. al. Deep Sets. URL: https://arxiv.org/abs/1703.06114

ДОДАТОК А ЛІСТИНГ ПРОГРАМНОГО ПРОДУКТУ

```
import warnings
import numpy as np
import pandas as pd
import random
import copy
import os
import time
import json
import dotenv
from collections import defaultdict
from pprint import pprint
from tqdm.notebook import tqdm
from rdkit import Chem
import boto3
import optuna
import mlflow
import pyarrow as pa

with warnings.catch_warnings():
    warnings.simplefilter("ignore")
    import plotly.express as px

from sklearn.metrics import roc_auc_score, accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import mean_squared_error, mean_absolute_error, mean_absolute_percentage_error, r2_score

import torch
import torch.nn as nn
from torch.optim import SGD, RMSprop, AdamW, Adam
from torch.optim.lr_scheduler import ReduceLROnPlateau, MultiStepLR
from torch.utils.data import Dataset
from torch_geometric.data import Data
from torch_geometric.nn import global_mean_pool, global_max_pool, global_add_pool, global_sort_pool
from torch_geometric.nn import GINConv, GATConv, GCNConv, MFConv, GINEConv, GATv2Conv
from torch_geometric.loader import DataLoader
print(torch.__version__)
print(torch.cuda.get_device_name() if torch.cuda.is_available() else "CPU")

from LibMTL.loss import AbsLoss
```

```
from LibMTL.metrics import AbsMetric
from LibMTL.weighting import EW, RLW, UW, GradNorm, DWA, PCGrad, CAGrad, GradVac, IMTL
from LibMTL.architecture import HPS
from LibMTL import weighting

def set_seed(seed):
    torch.manual_seed(seed)
    np.random.seed(seed)
    random.seed(seed)
    if torch.cuda.is_available():
        torch.cuda.manual_seed(seed)
        torch.cuda.empty_cache()
        torch.backends.cudnn.deterministic = True
        torch.backends.cudnn.benchmark = False

SEED = 5894384
DEVICE = torch.device("cuda" if torch.cuda.is_available() else "cpu")

NODE_SIZE = 14
EDGE_SIZE = 6
STUDY_NAME = "0_SingleTask_2D_v2"
DIRECTION = ["maximize", "maximize"]
CLASSIFICATION_TASKS = ["AMES", "HERG", "CYP_inh_3A4", "BBB"]
REGRESSION_TASKS = ["Acute", "HERG", "CYP_inh_3A4", "Caco-2"]
EXTERNAL_TASKS = ["CYP_inh_2D6", "HIA", "Bioavailability", "Clearance",
                  "CYP_sub_2D6", "Carcinogenicity_ISF", "PGP_sub"]

class GraphPool():
    def __init__(self, pool_type):
        self.coef_dict = {"mean": 1, "add": 1, "max": 1, "sort": 8, "mean_add": 2, "mean_max": 2, "add_max": 2, "mean_add_max": 3}
        self.type_ = pool_type
        self.coef = self.coef_dict[self.type_]

    def __call__(self, _graph_out, _graph_batch):
        if self.type_ == "mean":
            out = global_mean_pool(_graph_out, _graph_batch)
        elif self.type_ == "add":
            out = global_add_pool(_graph_out, _graph_batch)
        elif self.type_ == "max":
            out = global_max_pool(_graph_out, _graph_batch)
```

```
        elif self.type_ == "sort":
            out = global_sort_pool(_graph_out, _graph_batch, 8)
        elif self.type_ == "mean_add":
            out = torch.cat([global_mean_pool(_graph_out, _graph_batch), global_add_pool(_graph_out, _graph_batch)],
                            dim=1)
        elif self.type_ == "mean_max":
            out = torch.cat([global_mean_pool(_graph_out, _graph_batch), global_max_pool(_graph_out, _graph_batch)],
                            dim=1)
        elif self.type_ == "add_max":
            out = torch.cat([global_add_pool(_graph_out, _graph_batch), global_max_pool(_graph_out, _graph_batch)],
                            dim=1)
        elif self.type_ == "mean_add_max":
            out = torch.cat([global_mean_pool(_graph_out, _graph_batch), global_add_pool(_graph_out, _graph_batch),
                             global_max_pool(_graph_out, _graph_batch)], dim=1)
        return out



class FCLayers(torch.nn.Module):
    def __init__(self, trial, prefix, in_features, layers_range=(1, 2),
                 dropout_range=(0.1, 0.5), tune_bn=True, **kwargs):
        super(FCLayers, self).__init__()
        self.trial = trial
        self.prefix = prefix
        self.in_features = in_features
        self.layers = None
        self.n_out = None

        self.get_layers(layers_range, dropout_range, tune_bn)

    def get_layers(self, layers_range, dropout_range, tune_bn):

        in_features = self.in_features
        fc_layers = []
        if len(layers_range) == 1:
            n_fc_layers = layers_range[0]
        else:
            n_fc_layers = self.trial.suggest_int(self.prefix + "_n_layers", layers_range[0], layers_range[1])
        activation = nn.LeakyReLU()
        if tune_bn:
            use_bn = self.trial.suggest_categorical(self.prefix + "_use_bn", (True, False))
```

```
            else:
                use_bn = False
            if not use_bn:
                if len(dropout_range) == 1:
                    dropout = dropout_range[0]
                else:
                    dropout = self.trial.suggest_float(self.prefix + f"_dropout", dropout_range[0], dropout_range[1])

            for i in range(n_fc_layers):
                fc_layers.append(nn.Linear(in_features, in_features))
                if use_bn:
                    fc_layers.append(nn.BatchNorm1d(in_features))
                else:
                    fc_layers.append(nn.Dropout(dropout))
                fc_layers.append(activation)

            self.layers = nn.Sequential(*fc_layers)
            self.n_out = in_features

        def forward(self, x):
            return self.layers(x)


    class GNNLayers(torch.nn.Module):
        def __init__(self, trial, prefix, _node_features_len=None, _edge_features_len=None, use_edges_features=False):
            super(GNNLayers, self).__init__()
            self.trial = trial
            self.prefix = prefix
            self._node_features_len = _node_features_len
            self._edge_features_len = _edge_features_len
            self.use_edges_features = use_edges_features

            self.activation = nn.LeakyReLU()
            self.layers_list = None
            self.bn_list = None
            self.n_out = None

        def forward(self, data, **kwargs):
            _graph_out = data.x
            _edges_index = data.edge_index
```

```
        _edges_features = data.edge_attr if self.use_edges_features else None

        for _nn, _bn in list(zip(self.layers_list, self.bn_list))[:-1]:
            _graph_out = _nn(_graph_out, _edges_index, edge_attr=_edges_features) if self.use_edges_features else _nn(
                _graph_out, _edges_index)
            if _bn is not None:
                _graph_out = _bn(_graph_out)
            _graph_out = self.activation(_graph_out)

        if self.use_edges_features:
            _graph_out = self.layers_list[-1](_graph_out, _edges_index, edge_attr=_edges_features)
        else:
            _graph_out = self.layers_list[-1](_graph_out, _edges_index)
        if self.bn_list[-1] is not None:
            _graph_out = self.bn_list[-1](_graph_out)

        return _graph_out


class GATLayers(GNNLayers):
    def __init__(self, trial, prefix, gnn, _node_features_len, _edge_features_len, use_edges_features, layers_range,
                 heads_range, dropout_range, **kwargs):
        super(GATLayers, self).__init__(trial, prefix, _node_features_len=_node_features_len,
                                        _edge_features_len=_edge_features_len, use_edges_features=use_edges_features)
        self.gnn = gnn

        self.get_layers(layers_range=layers_range, heads_range=heads_range, dropout_range=dropout_range)

    def get_layers(self, layers_range, heads_range, dropout_range):
        _node_features_len = self._node_features_len
        _edge_features_len = self._edge_features_len

        self.use_edges_features = self.trial.suggest_categorical(self.prefix + "_use_edges_features",
                                                                 (True, False)) if self.use_edges_features else False
        if self.use_edges_features:
            _edge_features_fill = self.trial.suggest_categorical(self.prefix + "_edge_features_fill",
                                                                 ("mean", "add", "max", "mul", "min"))
        else:
            _edge_features_len = None
            _edge_features_fill = "mean"
```

```
        use_bn = self.trial.suggest_categorical(self.prefix + "_use_bn", (True, False))
        _n_layers = self.trial.suggest_int(self.prefix + "_n_layers", layers_range[0], layers_range[1])
        _heads = self.trial.suggest_int(self.prefix + f"_heads", heads_range[0], heads_range[1])
        _dropout = self.trial.suggest_float(self.prefix + f"_dropout", dropout_range[0], dropout_range[1])
        _layers_list = []
        _bn_layers_list = []
        _all_heads = 1

        for i in range(_n_layers):
            if self.use_edges_features:
                _gnn = self.gnn(_node_features_len * _all_heads, _node_features_len * _all_heads, heads=_heads,
                                dropout=_dropout, edge_dim=_edge_features_len, fill_value=_edge_features_fill)
            else:
                _gnn = self.gnn(_node_features_len * _all_heads, _node_features_len * _all_heads, heads=_heads,
                                dropout=_dropout)
            _layers_list.append(_gnn)

            _all_heads = _all_heads * _heads

            if use_bn:
                bn = nn.BatchNorm1d(_node_features_len * _all_heads)
                _bn_layers_list.append(bn)

        self.n_out = _node_features_len * _all_heads
        self.layers_list = nn.ModuleList(_layers_list)
        self.bn_list = nn.ModuleList(_bn_layers_list) if use_bn else [None] * _n_layers


class GATv1Layers(GATLayers):
    def __init__(self, trial, prefix, _node_features_len, _edge_features_len, layers_range=(1, 4), heads_range=(1, 5),
                 dropout_range=(0.0, 0.3), **kwargs):
        super(GATv1Layers, self).__init__(trial, prefix=prefix, gnn=GATConv,
                                          _node_features_len=_node_features_len, _edge_features_len=_edge_features_len,
                                          use_edges_features=True,
                                          layers_range=layers_range, heads_range=heads_range,
                                          dropout_range=dropout_range, **kwargs)


class GATv2Layers(GATLayers):
```

```
    def __init__(self, trial, prefix, _node_features_len, _edge_features_len, layers_range=(1, 4), heads_range=(1, 5),
                 dropout_range=(0.0, 0.3), **kwargs):
        super(GATv2Layers, self).__init__(trial, prefix=prefix, gnn=GATv2Conv,
                                          _node_features_len=_node_features_len, _edge_features_len=_edge_features_len,
                                          use_edges_features=False,
                                          layers_range=layers_range, heads_range=heads_range,
                                          dropout_range=dropout_range, **kwargs)


class GCNLayers(GNNLayers):
    def __init__(self, trial, prefix, _node_features_len, layers_range=(1, 4), n_units=(16, 32, 64, 128),
                 units_multiplier=(1, 1.5, 2), **kwargs):
        super(GCNLayers, self).__init__(trial, prefix, _node_features_len=_node_features_len)

        self.get_layers(layers_range=layers_range, n_units=n_units, units_multiplier=units_multiplier)

    def get_layers(self, layers_range, n_units, units_multiplier):
        _n_in = self._node_features_len

        use_bn = self.trial.suggest_categorical(self.prefix + "_use_bn", (True, False))
        _n_layers = self.trial.suggest_int(self.prefix + "_n_layers", layers_range[0], layers_range[1])
        _n_units = self.trial.suggest_categorical(self.prefix + f"_n_units", n_units)
        _units_multiplier = self.trial.suggest_categorical(self.prefix + f"_units_multiplier", units_multiplier)
        _layers_list = []
        _bn_layers_list = []

        _n_out = _n_units
        multiplier = 1

        for i in range(_n_layers):
            _n_out = int(_n_out * multiplier)
            _gnn = GCNConv(_n_in, _n_out)
            _layers_list.append(_gnn)

            if use_bn:
                bn = nn.BatchNorm1d(_n_out)
                _bn_layers_list.append(bn)

            _n_in = _n_out
            multiplier = _units_multiplier
```

```
        self.n_out = _n_out
        self.layers_list = nn.ModuleList(_layers_list)
        self.bn_list = nn.ModuleList(_bn_layers_list) if use_bn else [None] * _n_layers


class GIN_Layers(GNNLayers):
    def __init__(self, trial, prefix, gnn, _node_features_len, _edge_features_len, use_edges_features, layers_range,
                 n_units, units_multiplier, **kwargs):
        super(GIN_Layers, self).__init__(trial, prefix, _node_features_len=_node_features_len,
                                         _edge_features_len=_edge_features_len, use_edges_features=use_edges_features)
        self.gnn = gnn

        self.get_layers(layers_range=layers_range, n_units=n_units, units_multiplier=units_multiplier)

    def get_layers(self, layers_range, n_units, units_multiplier):
        _n_in = self._node_features_len

        use_bn = self.trial.suggest_categorical(self.prefix + "_use_bn", (True, False))
        _n_layers = self.trial.suggest_int(self.prefix + "_n_layers", layers_range[0], layers_range[1])
        _n_units = self.trial.suggest_categorical(self.prefix + f"_n_units", n_units)
        _units_multiplier = self.trial.suggest_categorical(self.prefix + f"_units_multiplier", units_multiplier)
        _layers_list = []
        _bn_layers_list = []

        _n_out = _n_units
        multiplier = 1

        for i in range(_n_layers):
            _n_out = int(_n_out * multiplier)
            _nn = nn.Sequential(nn.Linear(_n_in, _n_out), self.activation, nn.Linear(_n_out, _n_out))
            _gnn = self.gnn(_nn, edge_dim=self._edge_features_len) if self.use_edges_features else self.gnn(_nn)
            _layers_list.append(_gnn)

            if use_bn:
                bn = nn.BatchNorm1d(_n_out)
                _bn_layers_list.append(bn)

            _n_in = _n_out
            multiplier = _units_multiplier
```

```
        self.n_out = _n_out
        self.layers_list = nn.ModuleList(_layers_list)
        self.bn_list = nn.ModuleList(_bn_layers_list) if use_bn else [None] * _n_layers


class GINLayers(GIN_Layers):
    def __init__(self, trial, prefix, _node_features_len, _edge_features_len, layers_range=(1, 4),
                 n_units=(16, 32, 64, 128), units_multiplier=(1, 1.5, 2), **kwargs):
        super(GINLayers, self).__init__(trial, prefix=prefix, gnn=GINConv,
                                        _node_features_len=_node_features_len, _edge_features_len=_edge_features_len,
                                        use_edges_features=False,
                                        layers_range=layers_range, n_units=n_units,
                                        units_multiplier=units_multiplier, **kwargs)


class GINELayers(GIN_Layers):
    def __init__(self, trial, prefix, _node_features_len, _edge_features_len, layers_range=(1, 4),
                 n_units=(16, 32, 64, 128), units_multiplier=(1, 1.5, 2), **kwargs):
        super(GINELayers, self).__init__(trial, prefix=prefix, gnn=GINEConv,
                                         _node_features_len=_node_features_len, _edge_features_len=_edge_features_len,
                                         use_edges_features=True,
                                         layers_range=layers_range, n_units=n_units,
                                         units_multiplier=units_multiplier, **kwargs)


class GMFLayers(GNNLayers):
    def __init__(self, trial, prefix, _node_features_len, layers_range=(1, 4), n_units=(16, 32, 64, 128),
                 units_multiplier=(1, 1.5, 2), tune_bn=True, **kwargs):
        super(GMFLayers, self).__init__(trial, prefix, _node_features_len=_node_features_len)

        self.get_layers(layers_range, n_units, units_multiplier, tune_bn)

    def get_layers(self, layers_range, n_units, units_multiplier, tune_bn):
        _n_in = self._node_features_len

        if tune_bn:
            use_bn = self.trial.suggest_categorical(self.prefix + "_use_bn", (True, False))
        else:
            use_bn = False
```

```
        if len(layers_range) == 1:
            _n_layers = layers_range[0]
        else:
            _n_layers = self.trial.suggest_int(self.prefix + "_n_layers", layers_range[0], layers_range[1])
        _n_units = self.trial.suggest_categorical(self.prefix + f"_n_units", n_units)
        _units_multiplier = self.trial.suggest_categorical(self.prefix + f"_units_multiplier", units_multiplier)
        _layers_list = []
        _bn_layers_list = []

        _n_out = _n_units
        multiplier = 1

        for i in range(_n_layers):
            _n_out = int(_n_out * multiplier)
            _gnn = MFConv(_n_in, _n_out)
            _layers_list.append(_gnn)

            if use_bn:
                bn = nn.BatchNorm1d(_n_out)
                _bn_layers_list.append(bn)

            _n_in = _n_out
            multiplier = _units_multiplier

        self.n_out = _n_out
        self.layers_list = nn.ModuleList(_layers_list)
        self.bn_list = nn.ModuleList(_bn_layers_list) if use_bn else [None] * _n_layers


def main(weighting_name, architecture_name, testing, experiment_id, download):
    if testing:
        EXPERIMENT_NAME = EXPERIMENT_NAME_TEST
        EPOCHS = EPOCHS_TEST
        EXPERIMENTS = EXPERIMENTS_TEST
        TASKS  = TASKS_TEST
        test_string = "_test"
        print("[INFO] Running in testing mode...\n")
    else:
        test_string = ""

    if experiment_id > 0:
```

```
    EXPERIMENTS = EXPERIMENTS[experiment_id:]


weighting_method = getattr(weighting, weighting_name)


if architecture_name == 'new':
    nn_architecture = NN_ARCHITECTURE_NEW
    mode = 'separate'
elif architecture_name == 'main':
    nn_architecture = NN_ARCHITECTURE_MAIN
    mode = 'separate'
elif architecture_name == 'leonid':
    nn_architecture = NN_ARCHITECTURE_MAIN
    mode = 'all'
    TASKS.append("all")
elif architecture_name == 'leonid_new':
    nn_architecture = NN_ARCHITECTURE_NEW
    mode = 'all'
    TASKS.append("all")


if not os.path.exists(MODELS_LOCATION):
    os.makedirs(MODELS_LOCATION)


mlflow.set_tracking_uri(MLFLOW_DB_URL)
if not mlflow.get_experiment_by_name(EXPERIMENT_NAME):
    mlflow.create_experiment(EXPERIMENT_NAME, artifact_location=ARTIFACT_LOCATION)
mlflow.set_experiment(EXPERIMENT_NAME)


run_name = f"Stage_2_{VERSION}_{weighting_name}_{architecture_name}_architecture{test_string}"
try:
    run_id = mlflow.search_runs(experiment_names=[EXPERIMENT_NAME],
                                filter_string=f"tags.mlflow.runName = '{run_name}'",
                                max_results=1).iloc[0]['run_id']
    log_parameters = False
except IndexError:
    run_id = None
    log_parameters = True


print(f"[INFO] {DEVICE} is using\n")
print(f"[INFO] Weighting is set to `{weighting_name}`\n")
print(f"[INFO] NN architecture is set to `{architecture_name}`\n")
```

```
if download:
    print("[INFO] Downloading all files from S3...")
    download_files(TASKS)
    print("[INFO] Successfully downloaded all files!\n")

with mlflow.start_run(run_id=run_id, run_name=run_name):
    if log_parameters:
        mlflow.log_params(nn_architecture)
        mlflow.log_param("epochs", EPOCHS)
        mlflow.log_param("weighting", weighting_name)
        mlflow.log_param("architecture", architecture_name)

    for exp_id, experiment in enumerate(EXPERIMENTS):
        exp_id += experiment_id

        print(f"[INFO] Experiment {exp_id} is running...\n")

        exp_results = run_experiment(experiment, weighting_method, EPOCHS, nn_architecture, exp_id, mode)

        for task, task_dict in exp_results["best_result_simple"]["result"].items():
            for stage in task_dict:
                metric_name = f"exp_{exp_id}_{task}_{PROBLEM_TO_METRIC[TASK_TO_PROBLEM[task]]}_{stage}"
                mlflow.log_metric(metric_name, task_dict[stage])

        mlflow.log_dict(exp_results, f"exp_{exp_id}_results.json")
        mlflow.log_artifacts(f"{MODELS_LOCATION}/exp_{exp_id}", f"exp_{exp_id}_models")

        print("="*100, end='\n\n\n')
```