Міністерство освіти і науки України
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

Гарматюк Валентин Романович

«Синтез і оптимізація структур згорткових нейронних мереж / Synthesis and optimization of convolutional neural networks structures»

Студент групи КІм – 21
Гарматюк Валентин Романович
_____

Науковий керівник
д.т.н., проф. О.М. Березький
_____

Тернопіль – 2020

РЕЗЮМЕ

Випускна кваліфікаційна робота на тему «Синтез і оптимізація структур згорткових нейронних мереж» зі спеціальності 123 «Комп'ютерна інженерія» написана обсягом 101 сторінка і містить 37 рисунків, 5 таблиць, 1 додаток та 73 джерела за переліком посилань.

Метою роботи є розробка алгоритмів синтезу і оптимізації структур згорткових нейронних мереж і створення на цій основі генератора структур згорткових нейронних мереж.

Методи досліджень. Для розв'язання поставлених задач у випускній кваліфікаційній роботі використано методи: теорію штучних нейронних мереж, оптимізаційні методи, теорію алгоритмів, об'єктно-орієнтоване програмування.

Результати дослідження: на основі розроблених алгоритмів розроблено генератор структур згорткових нейронних мереж, який реалізовано в сервісі Google Colaboratory на мові програмування Python 3.6.

Орієнтовні напрямки розвитку досліджень: у подальшому розвитку даної програмної системи передбачено розробку підсистеми розпізнавання біомедичних зображень.

КЛЮЧОВІ СЛОВА: ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, СИНТЕЗ, ОПТИМІЗАЦІЯ, АЛГОРИТМ, ПРОГРАМНА СИСТЕМА.

RESUME

The qualification graduation thesis on «Synthesis and optimization of convolutional neural networks structures» from the specialty 123 «Computer engineering» is 101 pages long and contains 37 illustrations, 5 tables, 1 appendice, and 73 references.

The aim of the work is to develop algorithms for the synthesis and optimization of the structures of convolutional neural networks and to create on this basis a generator of the structures of convolutional neural networks.

Research methods. To solve tasks in the final qualification work, the following methods were used: the theory of artificial neural networks, optimization methods, the theory of algorithms, object-oriented programming.

Research results: on the basis of the developed algorithms, a generator of structures of convolutional neural networks has been developed, which is implemented in the Colaboratory service in the Python 3.6 programming language.

Approximate directions of research development: in the further development of this software system, it is envisaged to develop a subsystem for recognizing biomedical images.

KEY WORDS: CONVOLUTIONAL NEURAL NETWORK, SYNTHESIS, OPTIMIZATION, ALGORITHM, SOFTWARE SYSTEMS.

ЗМІСТ

ВСТУП

Актуальність роботи. В останні роки у штучному інтелекті велика увага приділяється напрямкам, які базуються на біологічних принципах. Цим напрямком займалися і продовжують займатися багато лабораторій і інститути в різних країнах. Широкого використання набувають прогресивні моделі штучного інтелекту. Цьому сприяють нові алгоритми навчання, які дозволяють обробляти великі обсяги інформації. В області алгоритмів машинного навчання та комп'ютерного зору появилося багато сучасних розробок, які швидко приймають рішення з точністю не гірше ніж фахівці. Класичні нейронні мережі для синтезу зображень ускладнені великою розмірністю вектора вхідних значень і витрачають великі обчислювальні ресурси на навчання і обчислення мережі. Згорткові нейронні мережі (ЗНМ) менш обтяжені такими недоліками.

Згорткові нейронні мережі (convolutional neural network, CNN) є спеціальними архітектурами штучних нейронних мереж, які входять до технології глибокого навчання (deep learning) [1]. Ця технологія входить до галузі машинного навчання, будується за аналогією з принципами роботи зорової кори головного мозку. У зоровій корі головного мозку були відкриті так звані прості клітини, що реагують на прямі лінії під різними кутами, і складні клітини, реакція яких пов'язана з активацією певного набору простих клітин [1]. Згорткові мережі набули широкого застосування в розпізнаванні зображень, рекомендаційних системах та обробці природної мови.

Мета і завдання дослідження. Метою випускної кваліфікаційної роботи є розробка алгоритмів синтезу і оптимізації структур згорткових нейронних мереж і створення на цій основі генератора структур згорткових нейронних.

У відповідності із поставленою метою випускна кваліфікаційна робота включає розв’язки таких задач:

проведення аналізу областей застосування нейронних мереж структури згорткових нейронних мереж;

аналіз методів навчання нейронних мереж;

розробка алгоритмів структурного синтезу згорткових нейронних мереж;

розробка алгоритмів оптимізації з використанням генетичних алгоритмів;

проектування та програмна реалізація генератора структур згорткових нейронних мереж і оптимізація його структури на основі генетичних алгоритмів;

тестування розробленого програмного забезпечення.

Об’єктом досліджень є згорткові нейронні мережі.

Предметом досліджень є синтез і оптимізація структур згорткових нейронних мереж.

Методи досліджень. У випускній кваліфікаційній роботі використано теорію штучних нейронних мереж, оптимізаційні методи, теорію алгоритмів, об’єктно-орієнтоване програмування.

Наукова новизна одержаних результатів полягає в розробці алгоритмів синтезу та оптимізації структур згорткових нейронних мереж .

Практичне значення одержаних результатів полягає в розробці генератора структур згорткових нейронних мереж.

Публікації результатів досліджень. За результатами досліджень опубліковані двоє тез доповідей III науково-практичної конференції молодих вчених і студентів «Інтелектуальні комп’ютерні системи та мережі» (26 листопада 2020 р., м. Тернопіль, Західноукраїнський національний університет) [2, 3]:

Гарматюк В.Р. Кухарук В.Р. Алгоритми оптимізації структур згорткових нейронних мереж: III Наук.-практ. конф. молодих вчених і студентів «Інтелектуальні комп’ютерні системи та мережі». 26 листопада 2020 р. Тернопіль, 2020. С. 49.

Кухарук В.Р., Гарматюк В. Р. Алгоритми генерування зображень на основі нейронних мереж: III Наук.-практ. конф. молодих вчених і студентів «Інтелектуальні комп’ютерні системи та мережі». 26 листопада 2020 р. Тернопіль, 2020. С. 50.

Випускна кваліфікаційна робота складається із трьох розділів, висновків, списку використаної літератури та додатку [4, 5].

У першому розділі здійснено огляд областей застосування нейронних мереж, проаналізовано структури згорткових нейронних мереж, описано методи навчання нейронних мереж, здійснено аналіз завдання на випускну кваліфікаційну роботу та постановку задач.

У другому розділі здійснено аналіз алгоритмів структурного синтезу згорткових нейронних мереж: «І/АБО» граф, метод віток і границь, побудова морфологічної матриці; алгоритмів оптимізації з використанням побудови множини Парето та алгоритмів оптимізації з використанням генетичних алгоритмів.

У третьому розділі спроектована і реалізована програмна система.

У додатку приведено світлокопії виданих публікацій.

1 АНАЛІЗ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ

1.1 Аналіз областей застосування нейронних мереж

Нейронні мережі це системи, які мають таку архітектуру, використовування якої дозволяє умовно імітувати роботу нейронів. Математична модель нейрона являє собою деякий універсальний нелінійний елемент із можливістю широкої зміни і настроювання його характеристик (параметрів). Нейронні мережі являють собою сукупність зв'язаних між собою шарів нейронів, що одержують вхідні дані, здійснюють їх обробку і генерують результат, який видають на виході.

Останнім часом зросла зацікавленість до використання нейронних мереж для вирішення різних завдань і застосування їх в різних сферах людського життя. Треба відмітити, що донедавна нейронні мережі відносилися лише до сфери людського інтелекту.

Завдяки нейронним мережам з’явилися можливості для створення систем, які, як людина, можуть вчитися, запам'ятовувати та аналізувати інформацію.

Розглянемо приклади використання нейронних мереж в: економіці, медицині, зв’язку і безпеці охоронних систем, обробці інформації тощо [6].

Успішним прикладом застосування нейромережевих технологій в галузі економіки є фінансова сфера. Для неї поширеними є системи управління кредитними ризиками. Ці системи застосовуються у деяких відомих банках США. Кредитний ризик оцінюється за вірогідністю завдання збитків від несвоєчасно повернених кредитів. Щоб запобігти втратам банки ще до видачі кредиту, проводять обчислення фінансової надійності позичальника. Ці розрахунки виходять з оцінки кредитної історії, динаміки розвитку компанії, стабільності її основних фінансових показників і багатьох інших факторів. Ці обчислення проводяться нейромережевим технологіями, що легко дозволяє встановити потенційних неплатників.

Крім вище приведеного прикладу застосування нейромережевих технологій в галузі економіки, існують системи з прогнозу ситуації на фондовому ринку, оцінки вартості нерухомості, прогнозування динаміки біржових курсів, оптимізації товарних і грошових потоків, автоматичного зчитування чеків і форм тощо [6].

Наступною галуззю за поширенням нейромережевих технологій є медицина. Найчастіше нейронні мережі використовуються в діагностиці захворювань. Наприклад, для діагностики онкологічних захворювань вчені з університету Дюка (США) розробили систему, яка успішно застосовується для діагностики раку молочної залози [7].

Для запобігання рецидивів виникнення пухлини після лікування раку молочної залози теж використовуються нейромережі. Завдяки мережам, дослідження яких проводяться на медичному факультеті Техаського університету. Вдалося покращити прогнозування рецидивів [8].

Також нейронні мережі використовуються в діагностиці серцево-судинних захворювань. Не менш відомим прикладом систем діагностики є програмний пакет для кардіодіагностики, розроблений фірмою RES Informatica спільно з Центром кардіологічних досліджень в Мілані. Система здійснює неінвазивну кардіодіагностику на основі розпізнавання спектрів тахограм. Такого типу системи успішно використовуються у деяких госпіталях Англії для попередження інфаркту міокарда та інших серцево-судинних захворювань, що дає можливість знижувати їх рівень.

Наступним прикладом використання мереж в онкології є нейромережа дослідників з медичної школи в Кагаве (Японія), яка практично безпомилково прогнозувала за передопераційними даними результати резекції печінки у хворих печінково-клітинною карциномою [8].

Нейромережі використовуються також для передбачення дії нових засобів лікування. Так, наприклад, дослідники з Національного інституту раку в США використовували нейромережі для прогнозу поведінки препаратів, що вживалися при хіміотерапії злоякісних пухлин [8].

На думку автора [6] нейронні мережі мають вигідне застосування в сфері телекомунікацій. З їх допомогою успішно вирішується важливі завдання при проектуванні і оптимізації мереж зв'язку: знаходження оптимального шляху трафіку між вузлами, проектування нових телекомунікаційних мереж, швидке кодування та декодування даних, стиснення відеоінформації тощо.

Нейронні мережі застосовуються в галузі безпеки і охоронних системах. Вони потрібні для ідентифікації особи, розпізнавання голосу, осіб в натовпі, розпізнавання автомобільних номерів, аналіз аерокосмічних знімків, моніторингу інформаційних потоків, виявлення підробок.

Слід відмітити про використання мереж у галузі обробки інформації. Існують розроблені, вище згаданою італійською фірмою RES Informatica, нейромережеві пакети серії FlexR d, які використовуються для розпізнавання і автоматичного введення рукописних платіжних документів і податкових декларацій. Нейронні мережі застосовуються для розпізнавання кількості товарів, їх вартості, фіскальних кодів, сум податків, підписів, відбитків пальців і голосу.

Автори [9] довели необхідність використання штучних нейронних мереж в аналізі соціологічних даних. Ця технологія може бути включеною в інструменти соціолога, оскільки інші методи не здатні з такою ж високою точністю здійснити прогнозування на підставі не дуже великих вибірок.

В наступній статті [10] автори досліджують процес застосування інтелектуальних технологій на основі нейронної мережі в статистичній системі аналізу і моніторингу телекомунікаційних систем та комп’ютерних мереж. Доведено можливість використання нейронних мереж для прогнозу процесів, що протікають у телекомунікаційних мережах в часовій області. Для цього була побудована схема паралельного прогнозування параметрів мережі. Схема являє собою нейронну мережу, причому у вхідному шарі розташовано 3 нейрони.

За оцінкою іншого автора [11], область застосування штучних нейронних мереж з кожним роком все більш розширюється, на сьогоднішній день вони використовуються в таких сферах як:

- машинне навчання (machine learning), що представляє собою різновид штучного інтелекту. В основі його лежить навчання ШІ на прикладі мільйонів однотипних завдань;
- в робототехніці нейронні мережі використовуються у виробленні чисельних алгоритмів роботи роботів;
- архітектори комп'ютерних систем користуються нейронними мережами для вирішення проблеми паралельних обчислень;
- з допомогою нейронних мереж математики можуть вирішувати різні складні математичні задачі і т. д.

Одним з її найбільш прогресивних напрямків є машинне навчання, яке є підгалуззю штучного інтелекту в галузі інформатики. Машинне навчання застосовує статистичні прийоми для надання комп'ютерам здатності «навчатися» з даних без явного програмування. Цей напрямок еволюціонував з досліджень розпізнавання образів та теорії обчислювального навчання в галузі штучного інтелекту. В машинному навчанні проходить вивчення та побудова алгоритмів, які можуть навчатися і здійснювати прогнозування, використовуючи наявні дані. Такі алгоритми слідують вимогливо статичним програмним інструкціям, здійснюючи керовані даними прогнози або ухвалюють рішення через побудову моделі з вибіркових входів.

Машинне навчання використовують у задачах, для яких розробка та програмування явних алгоритмів є складною або нездійсненною. Прикладами таких задач є фільтрування електронної пошти, виявляння мережевих завад або зловмисних інсайдерів, що зливають дані, оптичне розпізнавання символів (ОРС), навчання ранжуванню та комп'ютерний зір.

Машинне навчання знайшло застосування в біоінформатиці, в медицині, в медичній діагностиці, в геології і геофізиці, в соціології, в економіці, в додатках в техніці, у технічній діагностиці, в робототехніці, в комп'ютерному зорі, в розпізнаванні мови, тексту, виявленні спаму.

Одним із підходів машинного навчання є глибинне навчання, яке ґрунтується на алгоритмах, які намагаються моделювати високорівневі

абстракції в даних, застосовуючи глибинний граф із декількома обробними шарами, що побудовано з кількох лінійних або нелінійних перетворень [12].

Існують різні архітектури глибинного навчання, серед яких: глибинні нейронні мережі, згорткові глибинні нейронні мережі, глибинні мережі переконань та рекурентні нейронні мережі.

Розглянемо завдання, які розв'язують нейронні мережі.

1. Класифікація образів. Вхідний образ представляється вектором ознак і завдання полягає у визначенні приналежності (наприклад, мовного сигналу або рукописного символу) до одного або декількох попередньо визначених класів. Відомими задачами є розпізнавання букв, розпізнавання мови, класифікація сигналу електрокардіограми, класифікація клітин крові тощо.

2. Кластеризація / категоризація. Навчальна множина на початку алгоритму не має міток класів. Алгоритм ґрунтується на подібності образів і розміщує подібні образи в один кластер. Цей процес використовується для видобутку знань, стиснення даних і дослідження властивостей даних.

3. Апроксимація функцій. Для даного завдання є навчальна вибірка пар даних вхід-вихід – $((x_1, y_1), (x_2, y_2) ..., (x_n, y_n))$. Вона генерується невідомою функцією F, що спотворена шумом. Завданням апроксимації є знаходження невідомої функції F. Апроксимація функцій використовується для вирішення багатьох інженерних і наукових задач моделювання.

4. Передбачення / прогноз. Припустимо, що є задані n дискретних значень функції $\{y(t_1), y(t_2), ..., y(t_n)\}$ в послідовні моменти часу $t_1, t_2, ..., t_n$. Завдання полягає в прогнозуванні значення $y(t_{n1})$ в момент часу t_{n1}. Прогноз має велике значення для прийняття рішень в бізнесі, науці і техніці (передбачення цін на фондовій біржі, прогноз погоди тощо).

5. Оптимізація. Нехай є проблеми, що вимагають знаходження рішення, яке задовольняє системі обмежень і максимізує або мінімізує цільову функцію. Таке завдання називається оптимізацією. Оптимізація розглядається в математиці, статистиці, техніці, науці, медицині, економіці тощо.

6. Пам'ять, що адресується за змістом. У традиційних комп'ютерах звернення до пам'яті доступно тільки за допомогою адреси, що не залежить від змісту пам'яті. Більш того, якщо допущена помилка в обчисленні адреси, то може бути знайдена зовсім інша інформація. Асоціативна пам'ять або пам'ять, що адресується за змістом, доступна за вказівкою заданого змісту. Вміст пам'яті може бути викликано навіть по частковому входу або пошкодженому змісті. Асоціативна пам'ять може бути використана в мультимедійних інформаційних базах даних.

7. Управління. Розглянемо динамічну систему, задану сукупністю $\{u(t), y(t)\}$, де $u(t)$ – вхідний керуючий вплив, а $y(t)$ – вихід системи в момент часу t. У системах управління з еталонною моделлю метою управління є розрахунок такого вхідного впливу $u(t)$, при якому система діє по бажаної траєкторії, заданої еталонної моделлю. Прикладом є оптимальне управління двигуном.

В наступному підрозділі розглянемо структури згорткових нейронних мереж.

1.2 Аналіз структур згорткових нейронних мереж

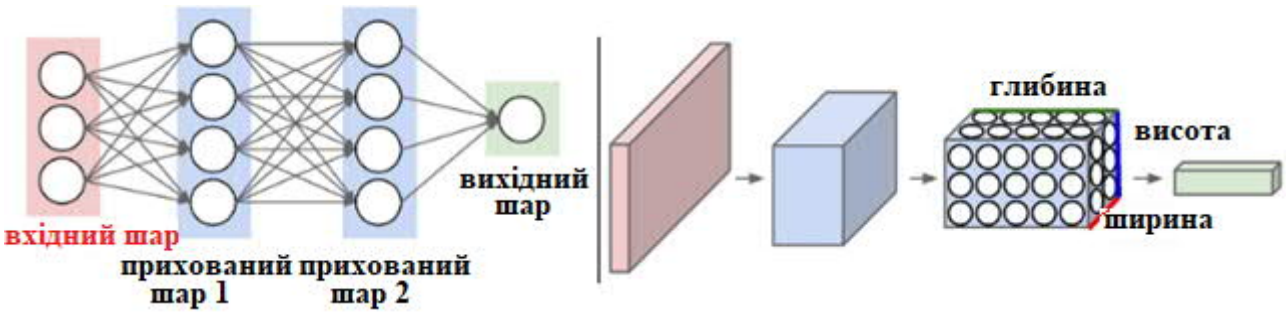
Як зазначалося в попередньому підрозділі існує багато областей застосування нейронних мереж.

Розглянемо структури згорткових нейронних мереж (ЗНМ).

ЗНМ добре підходять до масштабних зображень [13], що не можна сказати про звичайні нейронні мережі. В CIFAR-10 зображення мають розмір 32×32×3 (32 ширина, 32 висота, 3 кольорових канали). Тому один повністю пов'язаний нейрон у першому прихованому шарі звичайної нейронної мережі матиме $32 \cdot 32 \cdot 3 = 3072$ ваг. Ця кількість ще може бути керованою, але очевидно, що ця структура не є великим зображенням. Якщо зображення є більшого розміру, наприклад, 200×200×3, то нейрони, матимуть $200 \cdot 200 \cdot 3 = 120000$ ваг. Якщо

треба декілька таких нейронів, то параметри ще більше зростуть і це швидко призведе до надмірного розміщення.

В ЗНМ на вхід поступають зображення і архітектура будується більш розумним чином. Зокрема, на відміну від звичайної нейронної мережі, шари ЗНМ мають нейрони, розташовані в 3 вимірах: ширина, висота, глибина (слово глибина тут відноситься до третього виміру об'єму активації, а не до глибини повної нейронної мережі, яка може посилатися на загальну кількість шарів у мережі). Наприклад, вхідні зображення в CIFAR-10 – вхідний об'єм активацій, а об'єм має розміри 32×32×3 (відповідно ширина, висота, глибина). Нейрони в шарі будуть зв'язані лише з невеликою областю шару перед ним, замість усіх нейронів повністю пов'язаним чином. Більше того, кінцевий вихідний шар для CIFAR-10 мав би розміри 1×1×10, оскільки до кінця архітектури ЗНМ зменшується повне зображення в один векторний показник класу, розміщений уздовж глибинного виміру [1]. На рисунку 1.1а) показана звичайна тришарова нейромережа, а 1.1б) – ЗНМ розташовує свої нейрони в трьох вимірах (ширина, висота, глибина), як візуалізується в одному з шарів. Кожен шар ЗНМ перетворює об'єм введення 3D в об'єм виведення 3D в активації нейронів. У цьому прикладі червоний вхідний шар утримує зображення, тому його ширина та висота були б розмірами зображення, а глибина - 3 (Червоний, Зелений, Синій канали).



а) б)

Рисунок 1.1 – Звичайна тришарова нейромережа (а), згорткова нейронна мережа (б)

ЗНМ є класом глибинних штучних нейронних мереж прямого поширення, який застосовується до аналізу зображень. Ці мережі були створені для біологічних процесів, вони є варіаціями багатошарових перцептронів, і вимагають мінімальної попередньої обробки.

ЗНМ складається з шарів входу та виходу, а також із декількох прихованих шарів. Зазвичай вхідний шар лише трансформує двовимірне зображення у тривимірну матрицю. Тому на вхід окремого шару подається тривимірна матриця, проте на виході із шару її розмірність може бути змінена. Приховані шари ЗНМ зазвичай складаються зі згорткових шарів, агрегувальних шарів, повноз'єднаних шарів, шарів субдискретизації та шарів нормалізації [14].

Згорткові шари є основними компонентами цього типу штучних нейронних мереж, вони застосовують до входу операцію згортки і передають результат наступному шару. Згортка імітує реакцію окремого нейрону на зоровий стимул. Кожен згортковий шар має такі характеристики:

- F – розмір рецептивного поля. Він відповідає за кількість нейронів попереднього шару, які будуть зв’язані з одним нейроном у цьому шарі;
- K – глибина, кількість нейронів у цьому шарі (такою і буде глибина матриці на виході);
- S – крок. Він контролює як сильно будуть перекриватися рецептивні поля;
- P – розмір нульового доповнення. Він дозволяє контролювати розмір вихідної матриці. При потребі вхід на краях буде доповнено нулями.

Кожен нейрон згорткового шару представлений фільтром (ядром). Це є матриця, що має розмір $F \times F \times F$, її параметри змінюються під час навчання. За допомогою цього фільтра формується двомірна карта активації, яка є зрізом вихідної матриці шару.

ЗНМ використовують спільні ваги в згорткових шарах, що означає, що для кожного рецептивного поля шару використовується один і той же фільтр (банк ваг); це зменшує обсяг необхідної пам'яті та поліпшує продуктивність.

За допомогою повнозв’язного шару визначається клас на основі отриманих ознак. Цей шар працює повністю аналогічно як традиційна нейронна мережа багатошарового перцептрону. Повнозв'язні шари з'єднують кожен нейрон одного шару з кожним нейроном наступного шару.

Шар субдискретизації (підвибірки) змінює розмірність карт ознак. Це дозволяє спростити обчислення, а також може запобігати перенавчанню. Вважається, що факт наявності ознаки є важливішим за точне знання її координат. Група пікселів (найчастіше розміру 2×2) стискається до одного пікселя, проходячи нелінійне перетворення. За рахунок цього для кожного зрізу вхідної матриці застосовується фільтр розміром 2×2 і заміняються 4 числа на максимальне з них. На виході отримується матриця з у 2 рази меншою розмірністю [15]. Таке перетворення зачіпає прямокутники або квадрати, що не перетинаються. Дана операція пулінгу суттєво зменшує просторовий об’єм зображення.

Крім стиснення з функцією максимуму використовують і інші функції – середнього значення, $L2$-нормування, але, як показує практика, пулінг з функцією максимуму є найкращим.

Згорткові нейронні мережі використовуються в задачах класифікації супутникових даних [16], оскільки вони призначені для оброблення даних, які подаються у вигляді багатовимірних масивів. Автори запропонували архітектуру згорткової нейронної мережі (рисунок 1.2). Визначено чотири генеральні властивості, які забезпечують перевагу використання згорткових нейронних мереж для оброблення геопросторових даних: локальні з’єднання, спільні ваги (shared weights), підвибірка (pooling) та використання багатьох шарів. Типова згорткова мережа містить декілька рівнів оброблення інформації (рисунок 1.2) [17].

Декілька перших рівнів складаються з двох, що чергуються, типів шарів: згорткових та підвибіркових. Ці шари безпосередньо пов’язані з класичними представленнями про прості та складні клітини в нейробіології, а архітектура в цілому нагадує організацією зорової системи тварин [17].

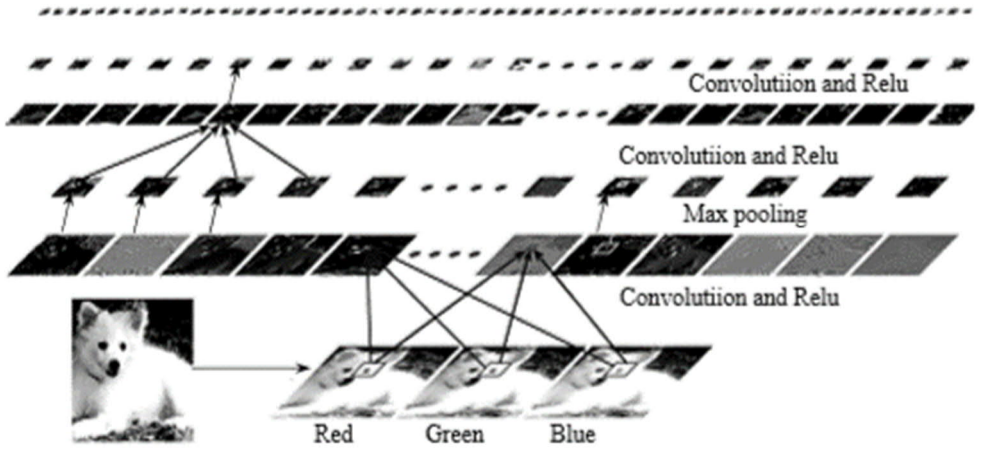


Рисунок 1.2 – Архітектура типової згорткової нейронної мережі

Параметри згорткового шару складаються з набору фільтрів або ядер, які мають невелике рецептивне поле по ширині та довжині, але простягаються на всю глибину вхідного багатовимірного масиву. Під час прямого проходження кожен фільтр здійснює згортку по ширині та висоті вхідного масиву даних, обчислюючи скалярний добуток вхідних даних та фільтра, формуючи двовимірну карту активації цього фільтра. Об’єднання всіх активаційних карт, дає вихідний масив згорткового шару.

Після кожного згорткового шару застосовується нелінійна функція активації (найпоширенішою є maxpooling) – це підвибірковий шар. Вона поділяє вхідне зображення на множину квадратів, які не перетинаються, і для кожного з них залишає максимальне значення. Зміст в тому, що точне розташування знайденої ознаки не так важливе, як її грубе положення відносно інших ознак. Підвибірковий шар забезпечує інваріантність відносно зсуву та спотворення об’єкта на вхідному зображенні.

Після декількох послідовних чергувань згорткового та підвибіркового шарів використовують один або декілька повнозв’язних шарів (fullyconnected) нейронів. Нейрони у повнозв’язному шарі з’єднані з усіма нейронами попереднього шару, як у звичайних нейронних мережах прямого поширення. Звичайний метод зворотного поширення похибки дозволяє навчати згорткову нейронну мережу, підбираючи ваги у фільтрах.

В 2014 році була розроблена модель нейронної мережі SRCNN (Super-Resolution Convolutional Neural Network), яка стала однією з найбільш відомих моделей згорткових мереж, що застосовуються для вирішення завдання збільшення зображень [18]. Перед тим, як здійснювати збільшення роздільності зображення за допомогою даної моделі, зображення попередньо збільшується за допомогою методів інтерполяції до необхідних розмірів. Даний метод інтерполяції і є згорткою. Архітектура мережі SRCNN зображена на рисунку 1.3.

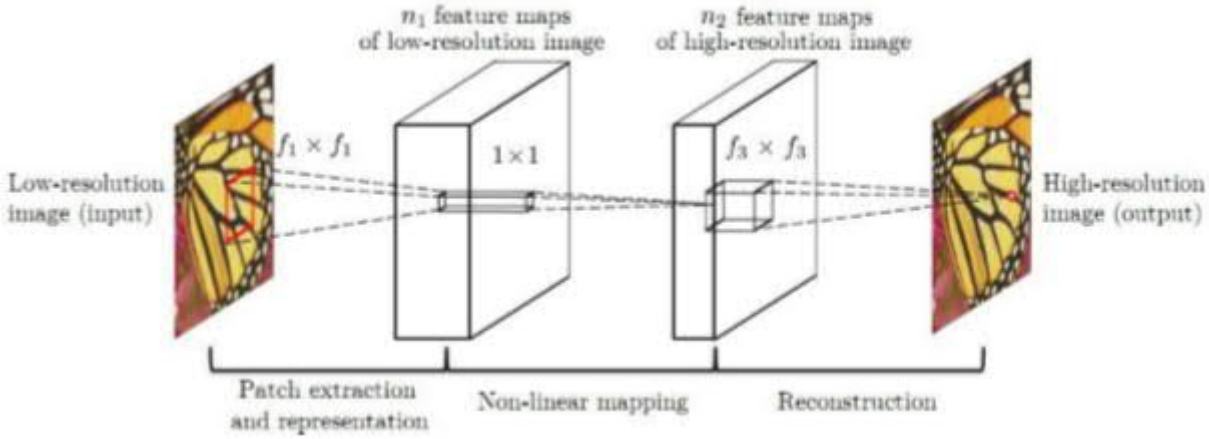


Рисунок 1.3 – Архітектура мережі SRCNN

З рисунку видно, що вхід поступає зображення низької роздільності «Low resolution image (input)» (*Y*). В результаті роботи нейронної мережі необхідно отримати зображення «High-resolution image (output)» (*F*(*Y*)). Це зображення має високу роздільність і з ним буде здійснюватися порівняння до вхідного зображення *X*. Завданням є знаходження функції *F*. Пошук складається з наступних 3-х кроків:

1. «Patch extraction and representation». Отримання патчів (окремих ділянок зображення, що складаються з декількох пікселів). В результаті цього кроку маємо патчі з *Y*. Потім кожен патч представляється у вигляді багатовимірного вектора, що містить набір карт ознак, розмір яких дорівнює розмірності вектора (n_1 feature maps of low-resolution image).

2. «Non-linear mapping» (нелінійне відображення). Результатом цього кроку є нелінійне відображення кожного багатовимірного вектора на інші

вектори. Кожен відображений вектор являє собою патч з високою роздільною здатністю. Ці вектори містять інший набір карт ознак (n_2 feature maps of high-resolution image).

3. «Reconstruction» (реконструкція). Патчі, отримані з нелінійного відображення використовуються для того, щоб згенерувати зображення високої роздільної здатності $F(Y)$, яке буде об'єктивно схожим з оригінальним зображенням X.

Дана модель містить три шари [18].

Перший шар нейронної мережі, який використовується для отримання патчів:

$$F_1(Y) = \max(0, W_1 * Y + B_1),$$

де W_1 – фільтри, B_1 – ваги, * – операція згортки.

W_1 визначається так:

$$W_1 = c \times f_1 \times f_1,$$

де c – кількість каналів зображення Y, f_1 – розмір фільтра у просторі.

Фільтри W_1 здійснюють n_1 операцій згортки зображення, і кожна згортка має ядро згортки розміру $c \times f_1 \times f_1$. Вихідні дані шару містять карти ознак в кількості n_1.

Ваги B_1 це n_1-мірний вектор. Кожен елемент його співставлено з елементом фільтра W_1.

Функція активації першого шару (ReLU (Rectified Linear Unit, «випрямляч»)):

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

де x – вхідний сигнал.

В другому шарі здійснюється операція нелінійного відображення n_1-мірних векторів на n_2-мірні. Використовуються фільтри 1 × 1. Другий шар ШНМ, здійснює операцію:

$$F_2(Y) = \max(0, W_2 * F_1(Y) + B_2),$$

де W_2 – фільтри, B_2 – ваги, * – операція згортки; B_2 це n_2-мірний вектор, W_2 здійснюють n_2 операцій

Функція активації теж ReLU.

Операція реконструкції здійснюється третім шаром:

$$F(Y) = W_3 * F_2(Y) + B_3,$$

де W_3 – це фільтри розмірністю $n_2 \times f_3 \times f_3$, B_3 – c-мірний вектор.

На виході після обробки даних третім шаром виходить зображення з високою роздільною здатністю.

Функція активації шару:

$$f(x) = x,$$

де x – це вхідний сигнал.

Для оцінки втрат використовується середня квадратична помилка MSE:

$$L(\theta) = \sum_{i=1}^{n} | F*(Y_i) - X_i |,$$

де θ – це параметри нейронної мережі $\{W_1, W_2, W_3, B_1, B_2, B_3\}$,

n –кількість зразків в навчальній вибірці.

Для даної моделі було з’ясовано, що найбільш оптимальними є наступні вхідні дані: $f_1 = 9$, $f_2 = 1$, $f_3 = 5$, $n_1 = 64$ і $n_2 = 32$ [18].

Подібний метод застосовується в розробленому компанією Google методі RAISR [19]. Алгоритм навчання і роботи RAISR відображено на рисунку 1.4.

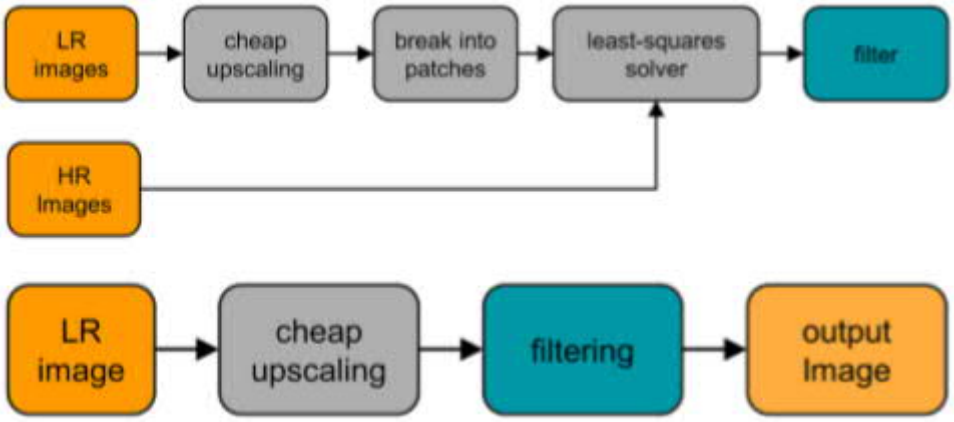


Рисунок 1.4 – Схема роботи і навчання методу RAISR

Саме навчання відбувається за кроками:

1. На вході маємо зображення низької роздільності.

2. Роздільність підвищується за допомогою білінійної інтерполяції.

3. За допомогою фільтрів оброблене зображення з кроку 2, ділиться на 4 різних патчі.

4. За допомогою фільтрів оригінальне зображення ділиться на 4 різних патчі.

5. За допомогою отриманих патчів та методу найменших квадратів проводиться регресійний аналіз.

Існують безліч модифікацій даної моделі, одними з найшвидших за швидкістю і ефективних за якістю є VDSR (Very Deep Convolutional Network) [20] і FSRCNN (Fast Super-Resolution Convolutional Neural Network) [21]. Також заслуговує на увагу модель ESPCN (Efficient SubPixel Convolutional Neural Network) [22], яка здатна в реальному часі збільшувати зображення. На відміну від SRCNN, на вхід подається не 18 збільшене за допомогою інтерполяції зображення, а зображення низького дозвіл. Збільшення розмірів зображення відбувається в результаті роботи нейронної мережі.

Розглянемо наступну мережу – LeNet. Розроблена була в 1990-і роки Яном Лекуном. Архітектура LeNet застосовувалася для зчитування поштових індексів, цифр і т. д. [23] (рисунок 2.5).

Рисунок 1.5 – Архітектура мережі LeNet

Отже, в даному підрозділі розглянено структури згорткових нейронних мереж.

1.3 Методи навчання нейронних мереж

Важливим етапом практичної побудови штучної нейронної мережі є її навчання. Навчання нейронної мережі – це налаштування параметрів нейронної мережі за допомогою моделювання середовища, в яку ця мережа вбудована (рисунок 1.6). Тип навчання визначається способом налаштування параметрів. Коли мережа добре навчена, то при поданні вхідних значень на вхід ШНМ, отримуємо правильні вихідні значення ШНМ. Для визначення якості роботи нейронної мережі використовують функцію втрат (помилки). Зазвичай за таку функцію обирають евклідову відстань, середньоквадратичну похибку або функцію кросентропії [16, 24]. Мережа вважається навченою, якщо функція втрат набуває мінімального значення.

Розрізняють два класи навчальних методів [25]: детерміністський і стохастичний.

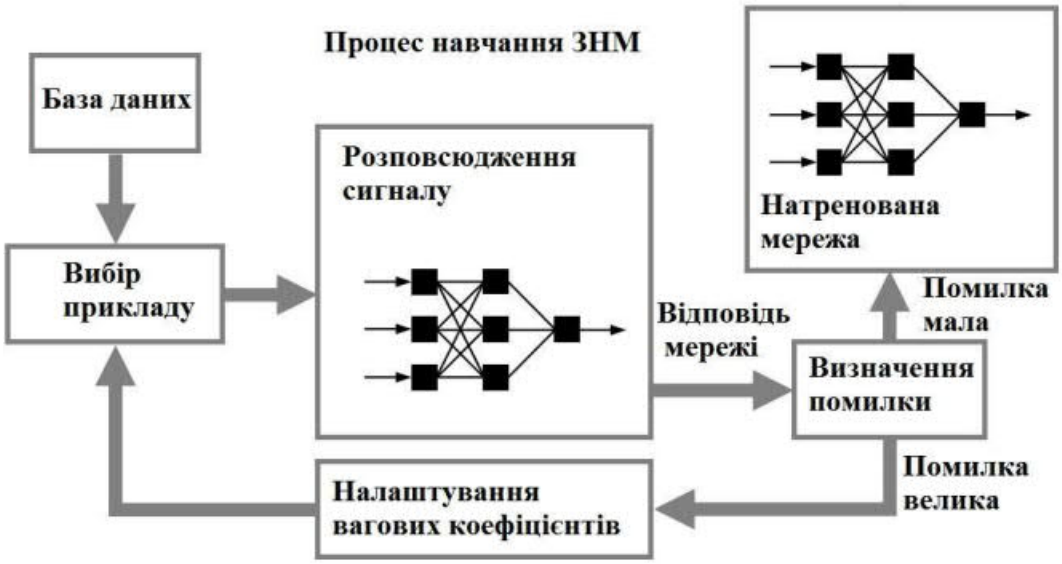


Рисунок 1.6 – Процес навчання нейронної мережі

Перший поетапно здійснює процедуру корекції ваг, яка базується на використанні їх поточних значень, значень входів, фактичних виходів і бажаних виходів.

Другий – здійснює псевдовипадкові зміни значень параметрів ШНМ, які спонукають до покращення значень на виході.

Обидва класи мають навчання з учителем та без учителя.

Взагалі існує три підходи до навчання нейронних мереж [26]: навчання з учителем (supervised learning), навчання без вчителя (unsupervised learning) і навчання з підкріпленням (reinforcement learning).

При навчанні з учителем на вхід мережі подаються набори вхідних сигналів (об'єктів), для яких заздалегідь відома правильна відповідь (Навчальна множина). Ваги змінюються за певними правилами в залежності від того, чи правильний вихідний сигнал видала мережа. Аналізуючи відомий навчальний набір даних, алгоритм навчання створює передбачувану функцію для прогнозування вихідних значень. Алгоритм навчається через набір навчальних даних, який керує машиною (рисунок 1.7) [27]. У контексті ШНМ процес навчання проходить як налаштування архітектури мережі і ваг зв’язків для ефективного виконання спеціального завдання. Функціонування мережі поліпшується із ітеративним налаштуванням вагових коефіцієнтів. Після

достатнього навчання система здатна забезпечити цілі для будь-якого нового вводу. Якщо при порівнянні своїх вихідних даних з правильними, алгоритм навчання знаходить помилки, то відповідним чином змінюється модель.

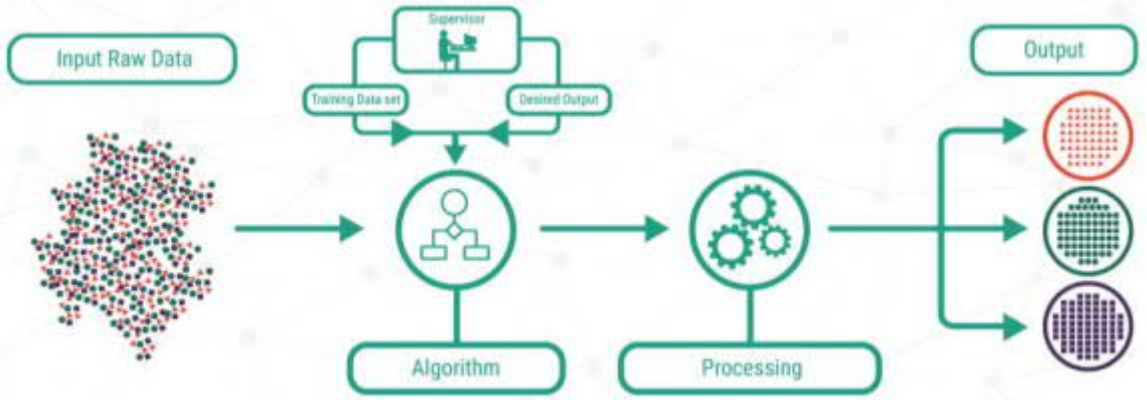


Рисунок 1.7 – Схема навчання з учителем

При навчанні без учителя на вхід мережі подаються об'єкти, для яких правильний вихідний сигнал заздалегідь невідомий, застосовується налаштування і корекція ваг зв’язків за визначеним алгоритмом. При цьому еталонні виходи відсутні. Інформація, що використовується для тренування, не є ні класифікованою, ні маркованою (рисунок 1.8) [27]. Навчання без вчителя вивчає, як системи можуть вивести функцію для опису прихованої структури з немаркованих даних. Система не визначає правильний висновок, але вона досліджує дані і може зробити висновки з наборів даних, щоб описати приховані структури з немічених даних.

Навчання з підкріпленням передбачає наявність зовнішнього середовища, з яким взаємодіє мережа. Навчання відбувається на підставі сигналів, отриманих від цього середовища.

Якщо заглянути в історію розвитку нейронних мереж, то спочатку мережі (мережі Маккаллок-Піттса) не навчалися. Ваги для всіх входів нейронів задавалися заздалегідь.

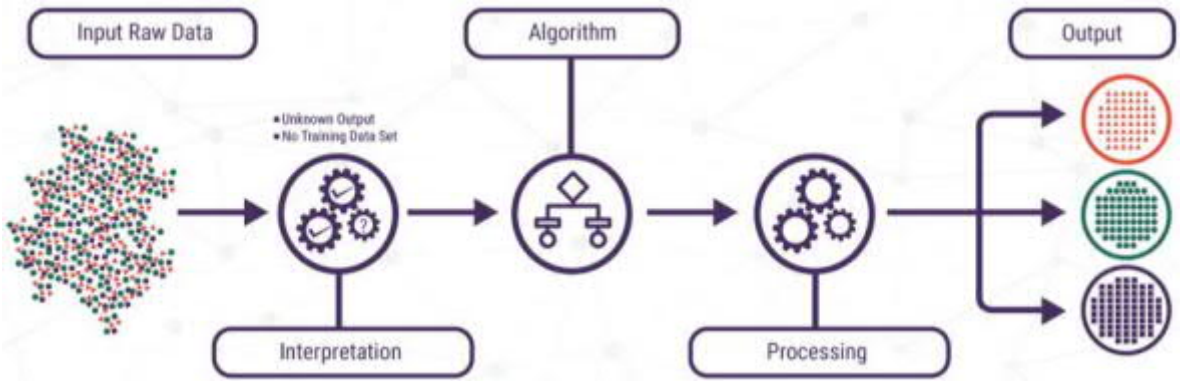


Рисунок 1.8 – Схема навчання без учителя

Вперше ідею навчання нейронних мереж запропонував Дональд Хеб в 1949 році [28, 29]. Згідно з Хебом, зв'язки нейронів, які активуються разом, повинні посилюватися, а зв'язки нейронів, які спрацьовують окремо один від одного, повинні слабшати. Хеб запропонував правила зміни ваги вхідних сигналів нейронів відповідно до того, правильну чи ні відповідь видала мережа [28].

В 1962 році А.В. Новиков довів збіжність запропонованого методу навчання нейрона на основі правил Хеба [30], за умови, що вибірка об'єктів лінійно роздільна. Згодом було запропоновано кілька аналогічних правил як для навчанням з учителем [31-33], так і без вчителя [34-38].

У 1970 році А.Г. Івахненко розробив метод групового урахування аргументів [39, 40], що дозволяє не тільки обчислювати ваги зв'язків між нейронами, а й визначати кількість шарів в мережі і нейронів в них в залежності від потреб прикладної задачі. Рівні мережі будуються і навчаються на основі навчальної множини з використанням регресійного аналізу. Потім відбувається етап спрощення мережі із застосуванням перевірочної множини об'єктів з відомими правильними відповідями. Для виключення непотрібних нейронів з мережі використовується регуляризація. В роботі [40] описано застосування методу групового врахування аргументів для навчання глибокої нейронної мережі, що складається з восьми шарів.

На даний час для навчання нейронних мереж часто використовується алгоритм зворотного поширення помилки, що ґрунтується на методі

градієнтного спуску. Цей спосіб мінімізації функції помилки запропонували Румельхарт, Гінт та Вільямс [25, 41].

При навчанні на кожній ітерації проходить корегування параметрів в напрямку антиградієнта $\nabla E(P)$

$$\Delta P = -\varepsilon \nabla E(P), \tag{1.1}$$

де $E(P)$ – функція помилки від параметрів P, $P = (W, Q)$ – параметри, які є множиною W ваг семантичних зв'язків у нейронній мережі та множиною Q порогових рівнів реакції нейронів.

Цей градієнтний метод гарантує збіжність до локальних мінімумів.

На кожній ітерації відбувається корегування параметрів – обчислюється градієнт функції похибки та оптимальний крок.

Функція помилки представляється у вигляді складної функції $E(W, Q)$ і послідовно обчислюються часткові похідні. Для вагових коефіцієнтів формула (1.1) матиме вигляд:

$$\Delta w_{ij}^{k} = -\varepsilon \left(\frac{\partial E(W,Q)}{\partial w_{ij}^{k}} \right)\Bigg|_{W,Q},$$

де похідна обчислюється для поточних значень параметрів W, Q.

Для обчислення нових значень ваг застосовується формула:

$$w_{ij}^{k'} = w_{ij}^{k} + \Delta w_{ij}^{k},$$

де i та j визначають семантичний зв'язок, k – прошарок нейронів. Аналогічні корекції вводяться для порогових рівнів.

Навчання алгоритмом зворотного поширення помилки передбачає два проходи по всім шарам мережі: прямий і зворотний. При прямому проході вхідний вектор подається на вхідний прошарок нейронної мережі, після чого поширюється по мережі від шару до шару. В результаті генерується набір вихідних сигналів, який і є фактичною реакцією мережі на даний вхідний образ. Під час прямого проходу всі синаптичні ваги мережі фіксовані. Під час зворотного проходу всі синаптичні ваги налаштовуються відповідно до правила корекції помилок, а саме: фактичний вихід мережі віднімається з бажаного, в результаті чого формується сигнал помилки. Цей сигнал згодом поширюється по мережі в напрямку, протилежному напрямку синаптичних зв'язків. Звідси і назва – алгоритм зворотного поширення помилки. Синаптичні ваги налаштовуються з метою максимального наближення вихідного сигналу мережі до бажаного.

Відзначимо, що алгоритм зворотного поширення помилки був запропонований без зв'язку з нейронними мережами в 1970 році в роботі [42]. Перший раз застосували цей алгоритм для навчання нейронних мереж в роботі [43]. Після цього з'явилося ще кілька робіт на цю тему [41, 44].

У глибокій нейронній мережі з декількома прихованими шарами проводиться розрахунок помилки, яка передається від одного шару до іншого. На першому етапі розраховується значення помилки на виході нейронної мережі, для якого ми знаємо правильні відповіді. Потім розраховується помилка на вході в вихідний шар мережі, яка буде використовуватися як помилка на виході прихованого шару. Таким способом розрахунок триває до того моменту, коли буде відома помилка на вхідному шарі. Саме тому алгоритм має назву зворотне поширення помилки.

Можливо кілька варіантів реалізації навчання нейронних мереж за допомогою алгоритму зворотного поширення помилки. При повному навчанні градієнт розраховується для всіх об'єктів навчальної вибірки. Однак такий підхід часто не є ефективним в разі, коли навчальна множина велика і для обробки всіх її елементів потрібно чимало часу. Альтернативний варіант – використання методу стохастичного градієнтного спуску, при якому ваги змінюються при

обробці одного елемента навчальної множини (онлайн-навчання) або декількох елементів (навчання на пакетах або міні-вибірках). На практиці для навчання нейронних мереж найчастіше використовується саме метод стохастичного градієнтного спуску або його модифікації [45-46].

1.4 Постановка задач дослідження

Метою випускної кваліфікаційної роботи є програмна реалізація синтезу і оптимізації структур згорткових нейронних мереж.

Для досягнення поставленої мети необхідно:

- провести аналіз областей застосування нейронних мереж;
- проаналізувати структури згорткових нейронних мереж;
- провести аналіз методів навчання нейронних мереж;
- проаналізувати алгоритми структурного синтезу;
- провести аналіз алгоритмів оптимізації структур нейронних мереж;
- програмно реалізувати автоматизований синтез структур ЗНМ;
- провести тестування реалізованих алгоритмів.

1.5 Висновки до розділу 1

Результатами першого розділу є:

- проведений аналіз областей застосування нейронних мереж;
- проведений аналіз структур згорткових нейронних мереж;
- проведений опис методів навчання нейронних мереж.

Таким чином, на основі проведеного аналізу показано актуальність проведення наступних досліджень та здійснено постановку задач на наступні розділи.

2 АЛГОРИТМИ СИНТЕЗУ ТА ОПТИМІЗАЦІЇ СТРУКТУР НЕЙРОННИХ МЕРЕЖ

2.1 Алгоритми структурного синтезу

Розглянемо методи, які є зручними для представлення тих задач, які розкладаються на ряд взаємно незалежних підзадач і є декілька варіантів рішення.

2.1.1 «І/АБО» граф

Першим способом є побудова «І/АБО» графу. Будується цей граф процесом перебору, який успішно завершується тоді, коли знаходиться вирішальний граф. Цей спосіб є вигідним тоді, коли задача окрім того, що ділиться на підзадачі, ще й має декілька варіантів розв'язку. Якщо припустити, задача *A* розв'язується через розв'язання множини задач *X* і *Y*, або через розв'язання множини задач *Z* і *V*, або через множину з однієї задачі *W*, тоді отримуємо опис задачі наступною структурою (рисунок 2.1).

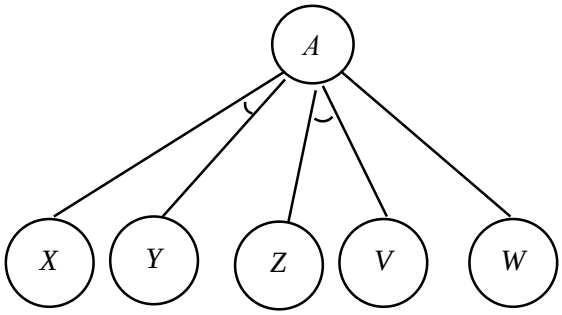


Рисунок 2.1 – Структура, що показує різні множини підзадач для задачі *A*

Якщо в кожну множину задач, що містить більш ніж одну результуючу задачу, додати додаткову породжуючу (батьківську) вершину, то структура рисунку 2.1 матиме вигляд, як показано на рисунку 2.2.

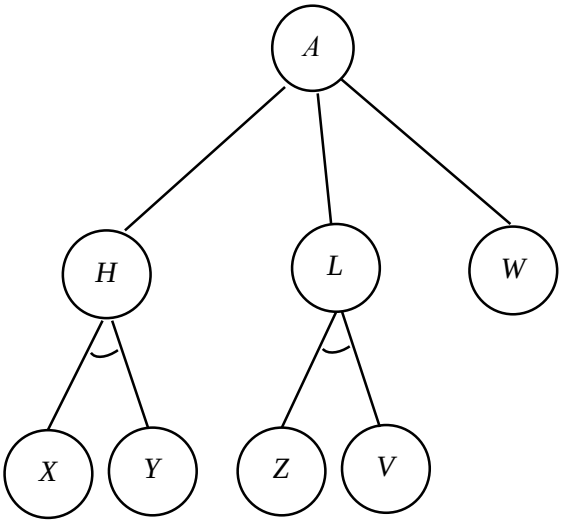


Рисунок 2.2 – «І/АБО» граф

Вершини *H, L, W* називаються «АБО» вершинами і вони є альтернативними підзадачами задачі *A*. Вершини *X* і *Y* (аналогічно *Z* і *V*) мають бути обидві розв'язані, щоб була розв'язана задача *H* (*L*). Вони належать до типу «І». На дугах цих вершин ставиться значок з'єднання. Вершини, що слідують з деякої вершини, називаються дочірними, а вершина, з якої вони слідують, є для них батьківською [47, 48].

Початковою вершиною графа, називається та, яка відповідає опису вихідної задачі, заключними ті, що відповідають опису елементарних задач.

Мета процесу пошуку на «І/АБО» графі – показати, що початкова вершина є вирішуваною. Представимо рекурсивно загальне визначення вирішуваності вершини на «І/АБО» графі:

Заключні вершини вирішувані.

Вершина, яка не є заключною, і має дочірні вершини типу «АБО», буде вирішуваною тоді і тільки тоді, коли вирішуваною є хоча б одна з дочірніх вершин.

Вершина, яка не є заключною, і має дочірні вершини типу «І», буде вирішуваною тоді і тільки тоді, коли вирішуваною є кожна з дочірніх вершин.

Представимо рекурсивно загальне визначення невирішуваності вершини на «І/АБО» графі:

Вершини, що не є заключними і що не мають дочірніх вершин, невирішувані.

Вершина, яка не є заключною, і має дочірні вершини типу «АБО», буде невирішуваною тоді і тільки тоді, коли невирішуваною є кожна з дочірніх вершин.

Вершина, яка не є заключною, і має дочірні вершини типу «І», буде невирішуваною тоді і тільки тоді, коли невирішуваною є хоча б одна з дочірніх вершин.

Всі процеси перебору включають наступні етапи:

1) початкова вершина відповідає початковому опису задачі;

2) процес розкриття вершини: будуються множини дочірніх вершин для початкової вершини з допомогою операторів зведення задач до підзадач;

3) процес встановлення вказівників: від кожної дочірньої вершини проводяться вказівники назад до батьківської вершини, які використовуються при спробі розмітити вирішувані і невирішувані вершини;

4) процеси 2 і 3 продовжуються доти, доки початкова вершина не буде помічена як вирішувана або як невирішувана.

І снує декілька методів перебору на графах «І/АБО». Відрізняються вони в основному тим, яким чином в них упорядковуються вершини перед розкриттям.

В методі повного перебору вершини розкриваються в тому порядку, в якому вони були побудовані. Послідовність кроків є довгою і велика кількість її зв’язана з перевірками того, чи не повинна дана процедура закінчити свою роботу.

Блок-схема методу повного перебору приведена на рисунку 2.3.

Всередині «І/АБО» дерева необхідно знайти дерево з мінімальною вартістю – оптимальне дерево. Позначимо через $h(s)$ вартість оптимального дерева рішення з коренем в початковій вершині s, $h(n)$ – вартість дерева рішення з коренем в вершині n, $c(n_i, n_j)$ – вартість дуги між вершиною n_i і дочірною вершиною n_j.

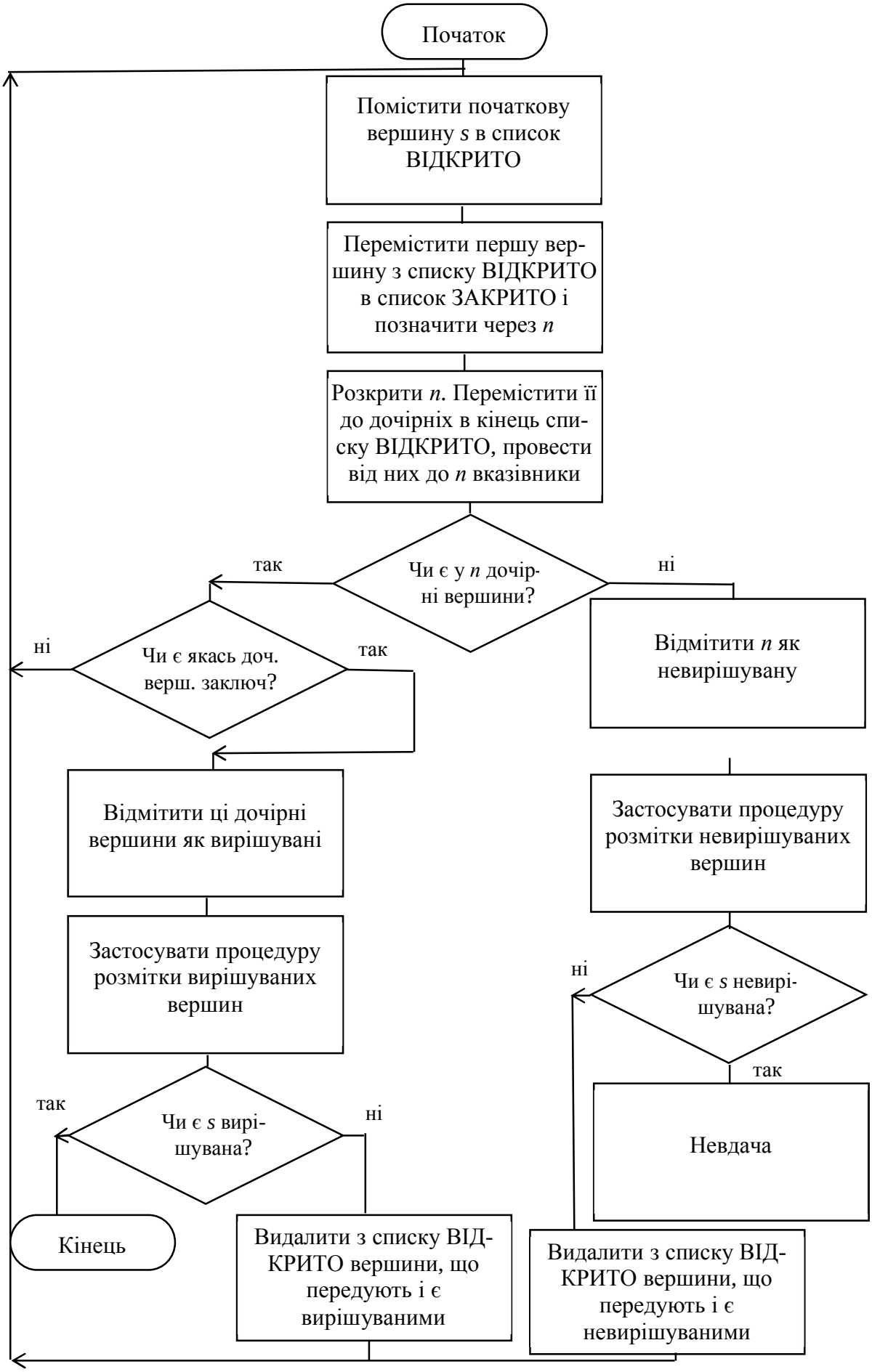


Рисунок 2.3 – Блок-схема методу повного перебору

1. Якщо n – завершальна вершина (елементарна задача), то

$$h(n) = 0.$$

2. Якщо n не є завершальною вершиною і має в якості своїх дочірніх вершин $n_1, \ldots, n_k$, що належать до типу «АБО», то

$$h(n) = \min_i [c(n, n_i) + h(n_i)].$$

3. Якщо n не є завершальною вершиною і має в якості своїх дочірніх вершин $n_1, \ldots, n_k$, що належать до типу «І», то для сумарних вартостей

$$h(n) = \sum_{i=1}^{k} [c(n, n_i) + h(n_i)],$$

а для максимальних вартостей

$$h(n) = \max_i [c(n, n_i) + h(n_i)].$$

2.1.2 Метод віток і границь

Нехай є задача, що має множину варіантів розв'язків $U = \{1, ..., m\}$. Якість розв'язку в i-му варіанті ($i = \overline{1, m}$) характеризується значенням деякого параметра q_i, а оптимальним розв'язком буде такий, який дає мінімальне значення:

$$q^* = \min_{i \in U} q_i$$

з множини $\{q_1, ..., q_m\}$.

Оцінкою знизу для множини U є таке значення $q = \hat{q}$, яке відповідає співвідношенню [45]

$$\hat{q} < q^* \text{ або } \hat{q} \le q^*.$$

Оцінка $\hat{q}$ називається досяжною, якщо в вище наведеній нерівності стоїть знак $\le$.

Метод віток і границь полягає в послідовному покращенні оцінки $\hat{q}$ і наближенні її до значення q^*.

Нехай є можливість знайти оцінку знизу як для всієї множини *U*, так і для її різних підмножин.

Розіб'ємо множину U на дві підмножини A та B з точними нижніми границями критерію якості q_A^* і q_B^*, для яких виконується співвідношення:

$$q^* = \min(q_A^*, q_B^*).$$

Множини A та B є меншими, ніж множина *U*, то є можливим отримати оцінки $\hat{q}_A$, $\hat{q}_B$ більш близькі до значень q_A^*, q_B^*, ніж в початковій множині *U*, що означає, що $\min(\hat{q}_A, \hat{q}_B)$ буде більш близьким до q^*, ніж оцінка $\hat{q}$.

Найімовірніше, що q^*, буде в тій з підмножин A та B, яка має меншу оцінку знизу. Тому варто розбивати множину U на множини A та B так, щоби оцінки знизу для цих підмножин відрізнялися як можливо більше.

Якщо виявиться, що $\hat{q}_A < \hat{q}_B$, то оптимальний варіант входить в множину A і множину A треба досліджувати детальніше. Розбиваємо її в свою чергу на дві підмножини C і D так, щоб оцінки $\hat{q}_C$ і $\hat{q}_D$ розрізнялися якнайбільше. Якщо виявиться, що $\hat{q}_C < \hat{q}_D$, то розбиваємо множину C і т. д. до тих пір, поки не прийдемо до підмножини, що складається всього з одного елемента зі значенням параметра $q = q_0$. Процедура розбиття представлена на рисунку 2.4.

Знайдений елемент q_0 необхідно порівняти з елементами нерозглядуваних множин *B, D*, щоб переконатися, що він дійсно дає мінімальне значення.

Приведемо алгоритм методу віток і границь.

1. Розбиття множини *U* на множини *A* та *B*.
2. Перевірка кількості елементів в множинах *A* та *B*.
3. Якщо $n_A = 1$, то $\hat{q}_A = q_A$, якщо $n_B = 1$, то $\hat{q}_B = q_B$, інакше перехід до кроку 4.
4. Знаходження $\hat{q}_A$, $\hat{q}_B$.
5. Якщо $\hat{q}_A < \hat{q}_B$, то *U* := *A*, інакше *U* := *B*.
6. Перехід до кроку 1.

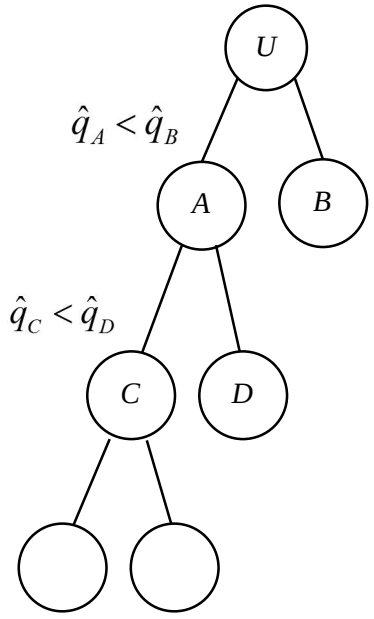


Рисунок 2.4 – Побудова дерева розбиття

2.1.3 Побудова морфологічної матриці

Морфологічний аналіз ґрунтується на комбінаторному принципі пошуку рішення. Суть методу полягає у створенні на етапі аналізу так званої морфологічної множини рішень, яка являє собою опис всіх потенційно можливих розв'язків цієї задачі. Передбачається, що таку повну морфологічну множину рішень можна побудувати на основі її частини – відомих описів технічних систем, що належать до цього класу (реалізують певну головну функцію в певних умовах). Для цього застосовують подання морфологічної множини рішень у вигляді морфологічної таблиці або морфологічного дерева, для побудови якої в найпростішому варіанті морфоаналізу визначають основні елементи (функції) об'єкта проектування та їх носіїв, виявлення якомога більшої кількості варіантів реалізації кожного елемента як незалежного від інших і систематичний перебір всіх можливих варіантів комбінацій реалізації елементів об'єкта проектування.

В більшості випадків, морфологічна таблиця [34] – це класифікаційна таблиця, кожний рядок якої являє собою класифікацію множини досліджуваних систем за певною істотною ознакою. Фактично морфологічна таблиця є сукупністю розділених на певні значення класифікаційних ознак функціональних систем морфологічної множини. Результатом побудови морфологічної таблиці є охоплення якомога більшої сукупності імовірних розв'язань задачі на основі закономірностей побудови (морфології) об'єкта проектування. Морфологічна таблиця МТ є результатом багатовимірної класифікації; вона призначена для різнобічного опису систем досліджуваної морфологічної множини рішень. Якщо на кожному рядку такої таблиці взяти по одній точці, тобто по одному значенню кожної з відібраних ознак , то сукупність точок дасть опис однієї з можливих технічних систем, що належать до морфологічної множини рішень.

Можна в кожному рядку морфологічної множини рішень задавати не тільки відомі, але й усі допустимі значення ознак. Тоді отримуються принципово нові технічні рішення. Деякі з них виявляться неправильними, а деякі –

принципово новими.

Морфологічний аналіз ділять на такі етапи.

1. Детальне вивчення об’єкта проектування і точне формулювання проблеми, шо підлягає розв’язанню. На цьому етапі надзвичайно важливо дати точний опис об’єкта дослідження.

2. Виявлення і створення переліку основних істотних характеристик (частин, ознак, параметрів, властивостей, функцій) об’єкта дослідження чи процесу, сукупність яких є необхідною і достатньою для функціонування об’єкта в заданих задачею умовах, тобто виділення класифікаційних ознак.

3. Визначення всіх відомих можливих варіантів реалізації чи зміни по кожній із істотних характеристик (частин, ознак, параметрів, властивостей, функцій), віднесених до класифікаційних ознак.

4. Зведення сукупності всіх отриманих варіантів у морфологічну таблицю.

5. Комбінація всіх можливих поєднань ознак і розгляд всіх варіантів вирішення проблеми, які виникають.

У морфологічному аналізі головним є саме отримання сполучень ознак. Морфологічний аналіз виконують за допомогою багатовимірних таблиць, в яких вибрані характеристики або компоненти об’єкта мають значення основних осей. Такі таблиці називають морфологічними таблицями, кожна комірка її відповідає варіантові рішення з певною комбінацією компонентів і характеристик. Такі таблиці полегшують пошук нових комбінацій рішень.

Переваги і недоліки методу. Морфологічний аналіз є, напевно, найсильнішим з неалгоритмічних методів. Він забезпечує розширення зони пошуку, гарантує “непропускання” жодного з можливих рішень. Високу ефективність методу пояснює уникнення елемента випадковості у перебиранні варіантів. Але метод має і такі недоліки:

1) потребує досить багато часу для отримання великого набору варіантів. Вибір ознак є досить суб’єктивним;

2) для виявлення морфологічних ознак потрібно знати структуру

проблеми, як метод сам по собі не розкриває;

3) в методі відсутні правила відбору найефективніших поєднань варіантів морфологічних ознак;

4) методу притаманна внутрішня суперечність: з одного боку, чим повніше враховані всі характеристики об’єкта, тим більшим буде вибір варіантів рішення і тим більша імовірність виявлення серед цих варіантів того, що задовольняє нас. З іншого боку, чим більше буде варіантів, тим складніше їх аналізувати. Для 20 ознак і 10 варіантів з кожної ознаки таблиця буде містити 10 у двадцятому степені можливостей реалізації технічної системи. Систематичний перебір цих варіантів без критерію ефективності рішення займе багато років;

5) вибір характеристики не виходить за межі вектора психологічної інерції, тобто отримати за цим методом принципово нове рішення неможливо.

Метод використовується для успішного пошуку рішень будь-яких інженерних і економічних проблем. Найкращі результати досягаються при дослідженні обмеженої зони пошуку, а не погано вивченої, розмитої, неясно сформульованої.

2.2 Алгоритми оптимізації з використанням побудови множини Парето

Під терміном «оптимізація», як правило, в техніці розуміють процес, який дає можливість отримати оптимальне, або близьке до оптимального рішення в заданих умовах.

Оцінку «оптимального рішення» приймає проектувальник (інженер), який має значний досвід в цій області чи колектив досвідчених науковців.

Слід відмітити, що оптимальне рішення буде таким «в заданих умовах», а в інших умовах отриманий результат вже може бути і не оптимальним.

Як показує практика, в більшості випадків в процесі розв’язання практичних оптимізаційних задач вдається лише покращити вже існуюче

рішення, а не знайти найкраще. Адже процес оптимізації є надзвичайно складним, на нього впливають різні за природою параметри кількість яких є надзвичайно велика. І, крім цього, досить часто, найкраще рішення може взагалі не існувати в даній області.

В процесі вивчення та формалізації оптимізаційних задач оперують такими складовими, як критерій оптимізації та цільова функція.

Критерій оптимізації – показник чи система показників якості роботи деякої системи, значення якої має бути мінімізовано (максимізовано) [50]).

Для того, щоб оперувати критерієм оптимізації в процесі розв'язання оптимізаційної задачі, необхідно мати справу з його кількісною оцінкою, а не з якісною. Тому критерій оптимізації треба формалізувати, записати в математичній формі. Даний запис називається цільовою функцією.

Цільова функція – функція, найбільше чи найменше значення якої шукається в задачах математичного програмування з врахуванням наявних обмежень [50].

В процесі розв'язання оптимізаційної задачі цільова функція дає змогу кількісно порівняти два чи більше альтернативних рішення. З математичної точки зору цільова функція описує деяку $(n+1)$-вимірну поверхню. Її значення визначається проектними параметрами

$$f = f(x_1, x_2, ..., x_n).$$

В ряді оптимізаційних задач потрібне введення більше однієї цільової функції. В таких випадках розробник має ввести систему пріоритетів і поставити у відповідність кожній цільовій функції деякий безрозмірний множник. В результаті з'являється так звана «функція компромісу», що дає змогу в процесі оптимізації користуватися однією складною цільовою функцією, а розробник має справу з задачею багатокритеріальної оптимізації оскільки присутні декілька критеріїв оптимізації.

Область, визначена всіма n проектними параметрами, називається

множиною допустимих рішень. Точка простору рішень, в якій цільова функція має найбільше значення в порівнянні з її значеннями в усіх інших точках її околу, називається локальним оптимумом.

Оптимальне рішення для всієї множини допустимих рішень, називається глобальним оптимумом. Воно краще за всі інші рішення, відповідні локальному оптимуму, і саме його шукає розробник.

В процесі побудови оптимізаційної задачі необхідно вирішити такі завдання:

1) вибрати та визначити критерій оптимізації;

2) визначити проектні параметри;

3) побудувати цільову функцію;

4) визначити обмеження;

5) вибрати метод розв’язання оптимізаційної задачі.

В загальному випадку задачу багатокритеріальної оптимізації можна сформулювати наступним чином [50, 51].

Знайти такі значення проектних параметрів, для яких забезпечуються екстремальні значення цільових функцій:

$$\min(\max)\ f_i(\vec{x}),\ (i=1,n),\ \ j=1,m,$$

при виконанні наступної системи обмежень

$$g_k(\overline{X}) \le \alpha_k,\ (k=1,l),$$
$$h_m(\overline{X}) = \beta_m,\ (m=1,j),$$
$$b_1 \le x_j \le b_m,$$

де вектор розв'язків $\vec{x} = (x_1, x_2, ..., x_n)^T$ належать до не порожньої області визначення D.

Отже, задача багатокритеріальної оптимізації полягає у пошуку вектора цільових змінних, який задовольняє накладеним обмеженням та оптимізує векторну функцію, елементи якої відповідають цільовим функціям [50-52].

Розглянемо задачу багатокритеріальної оптимізації, яка полягає в тому, що на першому кроці необхідно визначити альтернативи, що належать до множини Парето, а на другому, для прикладу, згортки (адитивна чи мультиплікативна) чи на основі досвіду проектувальника аналізувати вибір альтернатив. Цей спосіб підходить для випадків, коли маємо велику кількість альтернативних рішень.

В процесі пошуку рішень задачі багатокритеріальної оптимізації кількість варіантів, які задовольняють технічне завдання може досягати значної кількості. Відповідно, необхідно вибрати найоптимальніше. Разом з тим, слід додати, що буде існувати ще більша кількість рішень, які не будуть задовольняти технічне завдання, які також треба відкинути.

Пошук оптимальних розв'язків задачі багатокритеріальної оптимізації з використанням побудови множини Парето передбачає використання ряду термінів, підходів та означень.

Задача багатокритеріальної оптимізації формально описується множиною критеріїв

$$F(\overline{X}) = (f_1(\overline{X}), f_2(\overline{X}),..., f_k(\overline{X})),$$

$$\overline{X} = (x_1, x_2,..., x_l), \quad k = \overline{1,n}, \quad l = \overline{1,j}, \tag{2.1}$$

де k – кількість критеріїв оптимальності,

j – кількість проектних змінних.

Задача багатокритеріальної оптимізації може мати безліч рішень або жодного.

В процесі розв'язання задачі багатокритеріальної оптимізації вибираються кращі розв'язки, для цього доводиться порівнювати два розв'язки задачі між собою.

Позначимо перший розв'язок задачі багатокритеріальної оптимізації через $\overline{X}^1$, а друге – $\overline{X}^2$ ($\overline{X}^1$ та $\overline{X}^2$ належать до області визначення D), $\overline{X}^1, \overline{X}^2 \in D$. Говоримо, що $\overline{X}^1$ краще (володіє перевагою) у порівнянні з $\overline{X}^2$, якщо усі значення цільових функцій $f_k(\overline{X}^1) \geq f_k(\overline{X}^2)\ \forall k = \overline{1,n}$. Разом з тим, існує хоча б один з критеріїв ($k_{\max}$) для якого $f_{k\max}(\overline{X}^1) > f_{k\max}(\overline{X}^2)$. Тобто, розв'язок задачі багатокритеріальної оптимізації $\overline{X}^1$ не гірший за цими усіма частковими критеріями і серед них існує один, який має перевагу над одним з часткових критеріїв розв'язку задачі багатокритеріальної оптимізації $\overline{X}^2$.

Розв'язок $\overline{X}^* \in D$ задачі (2.1) називається ефективним розв'язком задачі багатокритеріальної оптимізації, якщо для нього не існує розв'язку з більшою перевагою (тобто такий розв'язок $\overline{X}^*$, який не можна покращити за якимось з часткових критеріїв, не погіршивши, при цьому, значення інших критеріїв).

Множина ефективних розв'язків задач багатокритеріальної оптимізації називається множиною Парето (*Pareto*(*D*)).

Вектор значень критеріїв, обчислених для ефективного розв'язку $F_{eff}(\overline{X}^*)$, називається ефективною оцінкою. Сукупність всіх ефективних оцінок, тобто образ множини Парето в просторі критеріїв, називається множиною ефективних оцінок і, як правило, позначається як $F_{eff}(Pareto)$.

Розв'язок $\overline{X}^1 \in D$ називається слабоефективним розв'язком задачі (2.1), якщо для нього не існує розв'язку $\overline{X}^2$ такого, що $\forall i = 1,k \quad f(\overline{X}^1) > f(\overline{X}^2)$, тобто, слабоефективний розв'язок, це такий розв'язок, який не може бути покращений одночасно за всіма критеріями.

На практиці, в основному, задача багатокритеріальної оптимізації зводиться до знаходження слабоефективних розв'язків.

Нехай $D = \{d_1, ..., d_M\}$ – множина можливих альтернатив, $q_i = (d_j)$, $i = \overline{1,p}$, $j = \overline{1,M}$ – сукупність значень *p* різних критеріїв, за якими оцінюють кожну з цих альтернатив (за більшим значенням критерію).

Тоді формально множину Парето *Pa* визначають таким чином [52].

Нехай для двох альтернатив d_α і d_β ($1 \leq \alpha \leq M$, $1 \leq \beta \leq M$, $\alpha \neq \beta$)

виконується нерівність

$$q_i(d_\alpha) \geq q_i(d_\beta) \quad \forall\, i = \overline{1, p}$$

і, крім цього, існує хоча б один критерій, для якого нерівність переходить до строгої нерівності.

Тоді альтернатива d_α краща за альтернативу d_β, і d_β треба викинути з розгляду, тобто $d_\beta \notin Pa$.

Найпростіший алгоритм визначення множини Pa:

1) вибираємо альтернативу d_1 вихідної множини $D = \{d_1, ..., d_M\}$ і порівнюємо її з рештою альтернатив. Якщо ж жодна з альтернатив множини $D = \{d_1, ..., d_M\}$ не домінує над d_1, то альтернатива d_1 належить до множини Pa і переходимо до кроку 3. Якщо знайдеться альтернатива d_o, яка домінує над d_1, то викидаємо d_1 з розгляду і переходимо до кроку 2.

2) порівнюємо альтернативу d_o вихідної множини $D = \{d_1, ..., d_M\}$ з рештою альтернатив. Висновки робимо аналогічно кроку 1.

3) вибираємо альтернативу d_2 вихідної множини $D = \{d_1, ..., d_M\}$ і порівнюємо її з рештою альтернатив. Висновки робимо аналогічно кроку 1.

4) обираємо наступні елементи множини $D = \{d_1, ..., d_M\}$ до тих пір, поки не будуть перевірені всі можливі варіанти.

Принцип Парето дає змогу звузити клас можливих претендентів на остаточне розв'язання і виключити з розгляду свідомо неконкурентоздатні варіанти. А остаточний вибір здійснюється на основі додаткової інформації про переваги особи, яка приймає рішення.

Правило визначення графічним способом. Множина Парето – ефективних оцінок $Pareto(Y_D)$ являє собою «північно – східну» границю множини Y_D без тих його частин, які паралельні одній з координатних осей чи розміщені в «глибоких» провалах (рисунки 2.5, 2.6) [51].

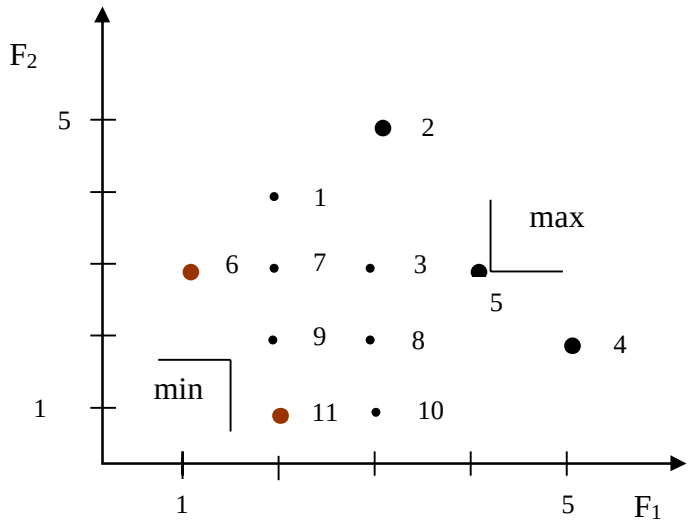


Рисунок 2.5 – Зображення правила для дискретних значень оцінок часткових критеріїв

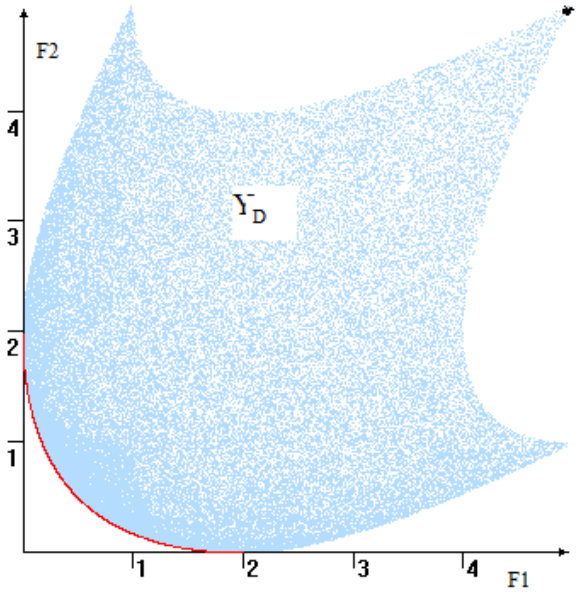


Рисунок 2.6 – Зображення правила для неперервної області розв’язків задачі багатокритеріальної оптимізації

Нехай є два критерії вибору: q_1 та q_2 – обидва вимагають максимальних значень. Визначаємо множину Парето графічним способом. Нехай альтернативи з конкретними значеннями критеріїв зображені точками на площині (рисунок 2.7, а). Для кожного з варіантів визначимо альтернативи, які гірші одразу за

двома критеріями. Для цього через точку, що аналізується, проводимо горизонтальну та вертикальну лінії та видаляємо всі альтернативи, які належать області, що ліворуч та нижче даної точки (рисунок 2.7, б). Як видно множину Парето утворюють точки 3, 6, 8.

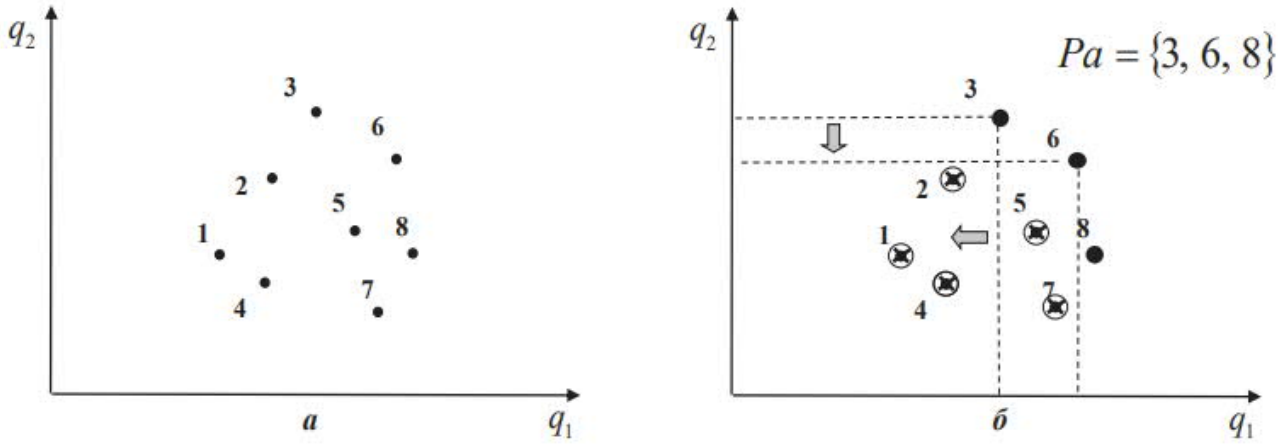


Рисунок 2.7 – Визначення множини Парето за двома критеріями max

Змінимо напрямок критеріїв. Нехай q_1 вимагає максимальних значень, q_2 – мінімальних. Альтернативи з конкретними значеннями критеріїв зображені точками на площині (рисунок 2.8, а). Продемонструємо графічно порядок визначення множини Парето *Pa*. Для кожної точки, що аналізується, видаляємо неефективні альтернативи. Вони знаходяться ліворуч та вище кожної альтернативи (рисунок 2.8, б). У результаті множину Парето утворюють лише точки 7 і 8.

На рисунках 2.7, 2.8 множини Парето мають більш ніж одну альтернативу. Коли множина Парето включає декілька альтернатив, то остаточний вибір ґрунтується на пораді експерта або застосовується додатковий критерій.

В даному підрозділі розглянуто метод багатокритеріальної оптимізації шляхом побудови множини Парето. В наступному підрозділі відійдемо від класичних методів оптимізації, а перейдемо до класу еволюційних алгоритмів.

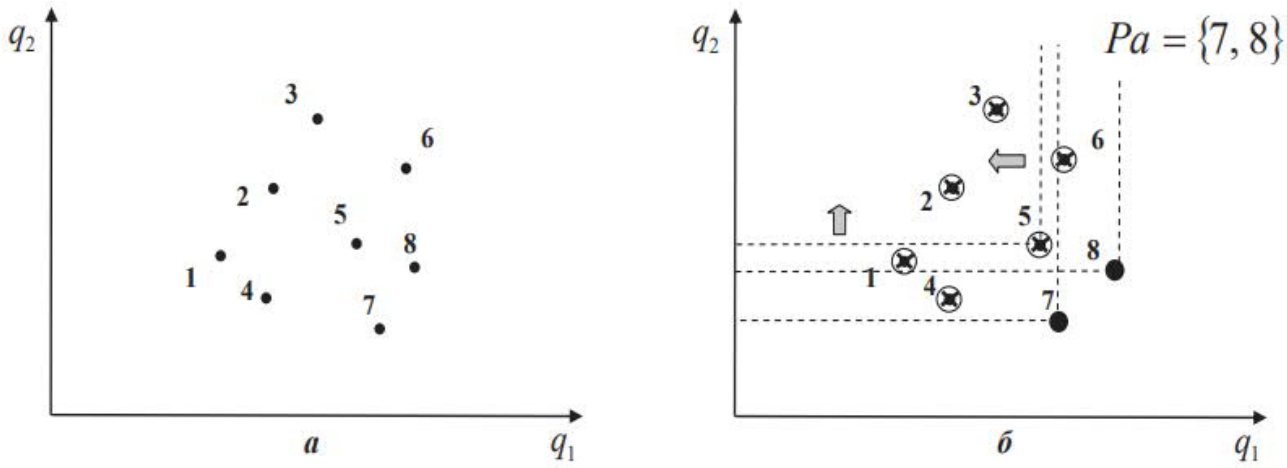


Рисунок 2.8 – Визначення множини Парето за $q_1(\max)$ та $q_2(\min)$

2.3 Алгоритми оптимізації з використанням генетичних алгоритмів

До класу еволюційних алгоритмів належать генетичні алгоритми. Вони мають ряд характеристик, які роблять їх кращими, ніж класичні методи оптимізації [53, 54]:

- пошук ефективних рішень з використанням генетичних алгоритмів не вимагає специфічних знань про задачу і параметри, що входять до неї;
- генетичні алгоритми замість детермінованих використовують стохастичні оператори. Останні показали себе досить стійкими в умовах зашумленості зовнішнього середовища;
- генетичні алгоритми одночасно враховують велику кількість індивідів популяції. Ця властивість паралелізму робить їх менш чутливими до локальних оптимумів і впливу шумів.

Генетичний алгоритм (genetic algorithm) – це алгоритм пошуку, що використовується для вирішення задач оптимізації і моделювання шляхом послідовного підбору, комбінування і варіації параметрів з використанням механізмів, що нагадують біологічну еволюцію.

Класичний генетичний алгоритм складається з наступних кроків:

1) ініціалізація, або вибір вихідної популяції хромосом;

2) оцінка пристосованості хромосом в популяції;

3) перевірка умови зупинки алгоритму;

4) селекція хромосом;

5) застосування генетичних операторів;

6) формування нової популяції;

7) вибір «найкращої» хромосоми.

Блок-схема основного генетичного алгоритму зображена на рисунку 2.9.

Ініціалізація – формування вихідної популяції. Цей крок полягає у випадковому виборі заданої кількості хромосом (особин), які подаються двійковими послідовностями фіксованої довжини.

Оцінка пристосованості хромосом в популяції складається в розрахунку функції пристосованості для кожної хромосоми цієї популяції. Чим більше значення цієї функції, тим вища «якість» хромосоми. Форма функції пристосованості залежить від характеру розв'язуваної задачі. Функція пристосованості завжди приймає невід'ємні значення і, крім того, що для рішення оптимізаційної задачі потрібно максимізувати цю функцію.

Перевірка умови зупинки алгоритму. Генетичний алгоритм зупиняється залежно від його конкретного застосування. В оптимізаційних задачах – після досягнення оптимального значення або, коли його виконання не призводить до покращення вже досягнутого значення. Також алгоритм може бути зупинений після закінчення певного часу виконання або після виконання заданої кількості ітерацій. Якщо умова зупинки виконана, то проводиться перехід до завершального етапу вибору «найкращої» хромосоми. В іншому випадку, на наступному кроці виконується селекція.

Селекція хромосом полягає у виборі за розрахованими значеннями функції пристосованості тих хромосом, які будуть брати участь в створенні нащадків для наступної популяції, тобто для чергового покоління. Такий вибір проводиться відповідно до принципу природного відбору.

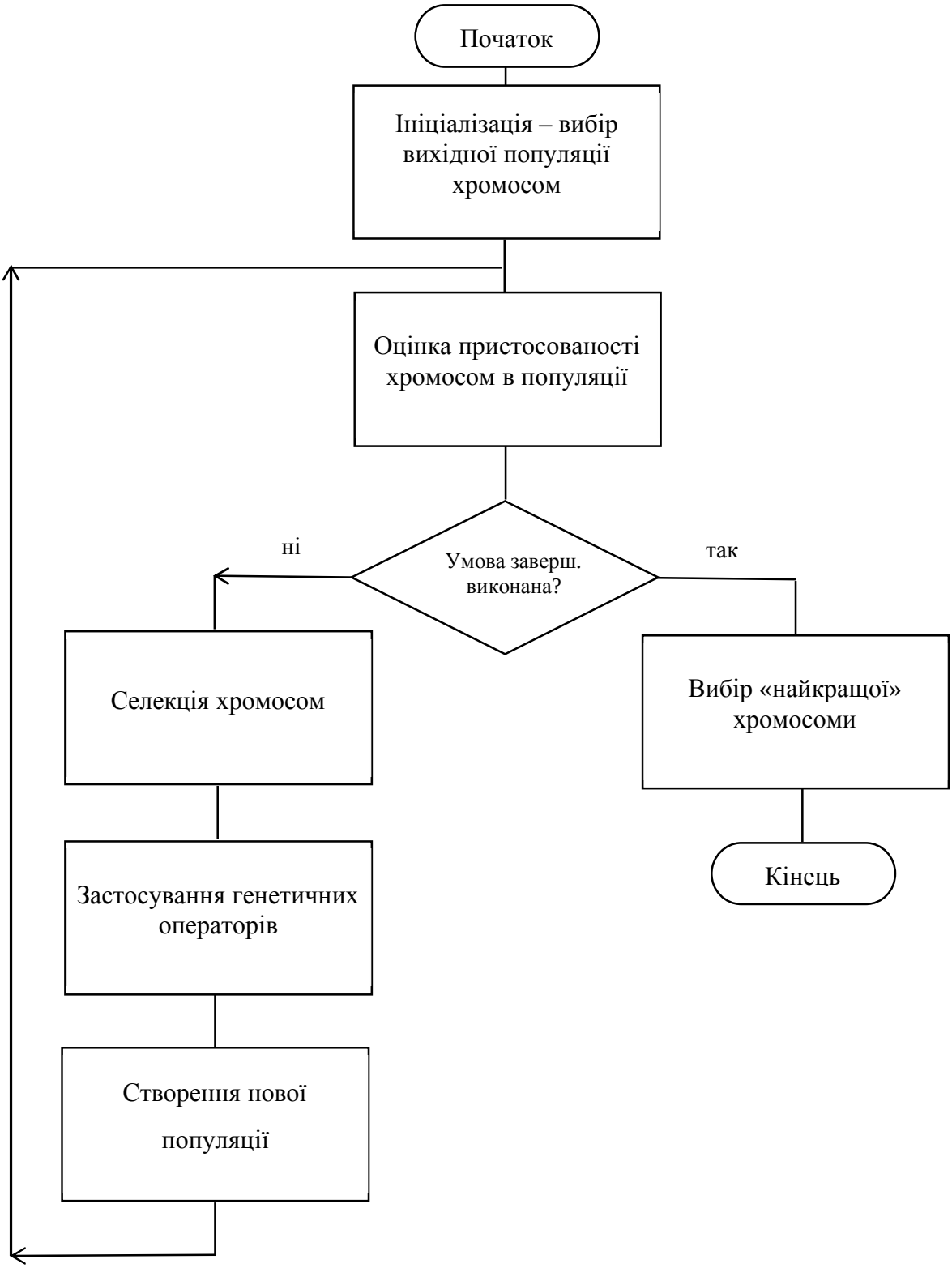


Рисунок 2.9 – Блок-схема основного генетичного алгоритму

За принципом природного відбору найбільші шанси на участь в створенні нових особин мають, хромосоми з найбільшими значеннями функції пристосованості. Існують різні методи селекції. Найбільш популярним вважається метод рулетки (назва за аналогією з відомою азартною грою). Кожній хромосомі зіставлений сектор колеса рулетки, величина якого встановлюється пропорційно значенню функції пристосування даної хромосоми. Чим більше значення функції пристосованості, тим більший сектор на колесі рулетки. Все колесо рулетки відповідає сумі значень функції пристосованості всіх хромосом даної популяції. Кожній хромосомі ch_i для $i = \overline{1, N}$ (де N – численність популяції) відповідає сектор колеса $v(ch_i)$, що виражається у відсотках за формулою:

$$v(ch_i) = p_s(ch_i) \cdot 100\%,$$

$$p_s(ch_i) = \frac{F(ch_i)}{\sum_{j=1}^{N} F(ch_j)},$$

де $F(ch_i)$ – значення функції пристосованості хромосоми ch_i;

$p_s(ch_i)$ – ймовірність селекції хромосоми ch_i.

Селекція хромосоми може бути представлена як результат повороту колеса рулетки, хромосома, яка «виграла» відноситься до сектору цього колеса. Чим більший сектор, тим більша ймовірність «перемоги» хромосоми. Тому ймовірність вибору даної хромосоми виявляється пропорційною значенню її функції пристосованості. Якщо все коло колеса рулетки представити у вигляді цифрового інтервалу [0, 100], то вибір хромосоми можна ототожнити з вибором числа з інтервалу [a, b], де a і b позначають відповідно початок і закінчення фрагмента кола, що відповідає цьому сектору колеса; очевидно, що $0 \le a < b \le 100$. В цьому випадку вибір за допомогою колеса рулетки зводиться до вибору числа з інтервалу [0, 100], яке відповідає конкретній точці на колі колеса.

В результаті процесу селекції створюється батьківська популяція (батьківський пул) чисельністю N, що дорівнює чисельності поточної популяції.

Застосування генетичних операторів до хромосом, відібраних за допомогою селекції, призводить до формування нової популяції нащадків від створеної на попередньому кроці батьківської популяції.

У класичному генетичному алгоритмі застосовуються два основних генетичних оператори: схрещування і мутації. Схрещування проводиться практично завжди з досить великою ймовірністю $(0{,}5 \le p_c \le 1)$, мутація – рідко (імовірність $0 \le p_m \le 0{,}1$).

Оператор схрещування. На першому етапі схрещування об'єднуються пари хромосом з батьківського пулу. Це проводиться випадковим способом відповідно до ймовірності схрещування p_c. Далі для кожної пари відібраних батьків розігрується позиція гена (локус) у хромосомі, що визначає так звану точку схрещування. Якщо хромосома кожного з батьків складається з L генів, то очевидно, що точка схрещування l_k є натуральне число, яке менше L. Тому фіксація точки схрещування зводиться до випадкового вибору числа з інтервалу [1, L–1]. В результаті схрещування пари батьківських хромосом виходить така пара нащадків:

1) нащадок, хромосома якого на позиціях від 1 до l_k складається з генів першого з батьків, а на позиціях від l_k + 1 до L – з генів другого з батьків;

2) нащадок, хромосома якого на позиціях від 1 до l_k складається з генів другого з батьків, а на позиціях від l_k + 1 до L – з генів першого з батьків.

Мутація хромосом може виконуватися на популяції батьків перед схрещуванням або на популяції нащадків, утворених в результаті схрещування. Оператор мутації з імовірністю p_m змінює значення гена в хромосомі на протилежне (тобто з 0 на 1 чи навпаки). Наприклад, якщо в хромосомі [100110101010] мутації піддається ген на позиції 6, то його значення, рівне 0, змінюється на 1, що призводить до утворення хромосоми [100111101010].

Формування нової популяції. Хромосоми, отримані в результаті застосування генетичних операторів до хромосом тимчасової батьківської

популяції, включаються до складу нової популяції. Вона стає поточною популяцією для даної ітерації генетичного алгоритму. На кожній черговій ітерації розраховуються значення функції пристосованості для всіх хромосом цієї популяції, після чого перевіряється умова зупинки алгоритму і або фіксується результат у вигляді хромосоми з найбільшим значенням функції пристосованості, або здійснюється перехід до наступного кроку генетичного алгоритму, тобто до селекції. У класичному генетичному алгоритмі вся попередня популяція хромосом заміщається новою популяцією нащадків, що має ту ж чисельність.

Вибір «найкращої» хромосоми. Якщо умова зупинки алгоритму виконана, то виводиться розв'язок задачі. Кращим розв'язком є хромосома з найбільшим значенням функції пристосованості.

Генетичні алгоритми успадкували властивості природного еволюційного процесу, що складаються в генетичні зміни популяцій організмів з плином часу.

Головним фактором еволюції є природний відбір (природна селекція), який призводить до того, що серед особин однієї і тієї ж популяції виживають і залишають потомство лише найбільш пристосовані. У генетичних алгоритмах також як в навколишньому середовищі є етап селекції, на якому з поточної популяції вибираються і включаються в батьківську популяцію особини, які мають найбільші значення функції пристосованості. На етапі еволюції застосовуються генетичні оператори схрещування і мутації, які виконують рекомбінацію генів в хромосомах.

Операція схрещування – це обмін фрагментами ланцюжків між двома батьківськими хромосомами. Пари батьків для схрещування вибираються з батьківського пулу випадковим чином так, щоб ймовірність вибору конкретної хромосоми для схрещування дорівнювала ймовірності p_c. Наприклад, якщо в якості батьків випадковим чином вибираються дві хромосоми з батьківської популяції чисельністю N, то $p_c = 2/N$, якщо вибирається $2s$ хромосом $(s \le N / 2)$, які утворюють s пар батьків, то $p_c = 2s/N$. Якщо всі хромосоми поточної

популяції об'єднані в пари до схрещування, то p_c = 1. Після операції схрещування батьки в батьківській популяції заміщуються їхніми нащадками.

Операція мутації змінює значення генів в хромосомах із заданою вірогідністю p_m способом, представленим при описі відповідного оператора. Це призводить до інвертування значень відібраних генів з 0 на 1 і назад. Значення p_m, як правило, дуже мале, тому мутації піддається лише невелика кількість генів. Схрещування – це ключовий оператор генетичних алгоритмів, що визначає їх можливості і ефективність. Мутація грає більш обмежену роль. Вона вводить в популяцію деяку різноманітність і попереджає втрати, які могли б статися внаслідок виключення якого-небудь значимого гена в результаті схрещування.

Рисунок 2.10 представляє просту реалізацію генетичного алгоритму на мові програмування C++.

```
# include <iostream.h>
# include <algorithm.h>
# include <numeric.h>
using namespace std;
int main()
{
    //початковий масив (популяція) з 1000 елементів (осіб).
    const int N = 1000;
    int a[N];
    //заповнимо елементи нулями
    fill(a, a+N, 0);
    for (;;)
    {
        //Мутація кожного елемента.
        //Випадково збільшуємо або зменшуємо значення елементу на один;
        for (int i = 0; i < N; ++i)
            if (rand()%2 == 1)
                a[i] += 1;
            else
                a[i] -= 1;
        //відсортуванням по зростанню вибираємо більші за значенням...
        sort(a, a+N);
        //... і тоді більші за значенням виявляться в другій частині масиву.
        //скопіюємо більші в першу половину, коли вони залишили нащадків, а перші померли:
        copy(a+N/2, a+N,    a /*куди*/);
        //тепер поглянемо на середнє значення елементу популяції. Як бачимо, середнє значення все більше і більше.
        cout << accumulate(a, a+N, 0) / N << endl;
    }
}
```

Рисунок 2.10 – Реалізація генетичного алгоритму на C++

Код програми (рисунок 2.10) реалізує програму пошуку в одномірному просторі, без схрещення. Згідно програми більші за значенням елементи, представлені цілими числами, найбільш життєздатні.

Генетичні алгоритми і нейронні мережі пов’язані між собою. Існує об'єднання генетичних алгоритмів і нейронних мереж під абревіатурою COGANN (Combinations of Genetic Algorithms and Neural Networks). Це об'єднання є допоміжним або рівноправним. При допоміжному об'єднанні методи застосовуються послідовно один за іншим, причому один з них служить для підготовки даних, що використовуються при реалізації другого методу. При рівноправному об'єднанні обидва методи застосовуються одночасно.

В таблиці 2.1 представлена класифікація об'єднань генетичних алгоритмів і нейронних мереж [56].

Генетичні алгоритми для підтримки нейронних мереж. Підхід, заснований на використанні генетичного алгоритму для забезпечення роботи нейронної мережі, схематично представлений на рисунку 2.11. Можна виділити три області проблем [39]:

- застосування генетичного алгоритму для підбору параметрів або перетворення простору параметрів, використовуваних нейронною мережею для класифікації;
- застосування генетичного алгоритму для підбору правила навчання або параметрів, які керують навчанням нейронної мережі;
- застосування генетичного алгоритму для аналізу нейронної мережі.

Таблиця 2.1 – Об’єднання генетичних алгоритмів (ГА) та нейронних мереж

Вид об’єднання	Характеристика об’єднання	Приклади використання	Література
1	2	3	4
	ГА та нейронні мережі незалежно застосовуються для вирішення однієї і тієї ж задачі	Однонаправлені нейронні мережі, мережі Кохонена з самоорганізацією і ГА в задачах класифікації	[57]

Продовження таблиці 2.1

1	2	3	4
Допоміжне	Нейронні мережі для забезпечення ГА	Формування вихідної популяції для ГА	[58]
	ГА для забезпечення нейронних мереж	Аналіз нейронних мереж	[59]
		Підбір параметрів або перетворення простору параметрів	[60]
		Підбір параметрів або правила навчання (еволюція правил навчання)	[61]
Рівноправне	ГА для навчання нейронних мереж	Еволюційне навчання мережі (еволюція ваг зв'язків)	[62]
	ГА для вибору топології нейронної мережі	Еволюційний підбір топології мережі (еволюція мережевої архітектури)	[62]
	Системи, що об'єднують адаптивні стратегії ГА і нейронних мереж	Нейронні мережі для вирішення оптимізаційних задач із застосуванням ГА для підбору ваг мережі	[63]
		Реалізація ГА за допомогою нейронної мережі	[64]
		Застосування нейронної мережі для реалізації оператора схрещування в ГА	[65]

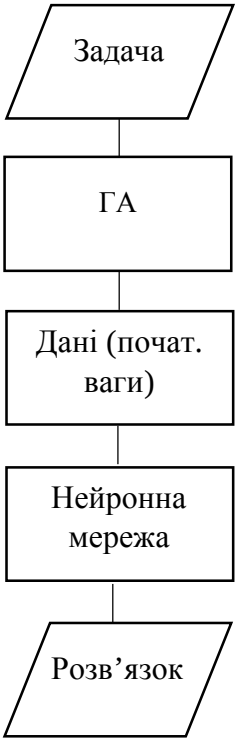


Рисунок 2.11 – Допоміжне об'єднання генетичного алгоритму з нейронною мережею

Генетичний алгоритм використовується при підготовці даних для нейронної мережі, що відіграє роль класифікатора. Ця підготовка може виконуватися шляхом перетворення простору параметрів або виділенням деякого підпростору, що містить необхідні параметри.

Розглянемо рівноправне об'єднання генетичного алгоритму і нейронних мереж. Почнемо з застосування генетичних алгоритмів для навчання нейронних мереж. Як правило, задача полягає в оптимізації ваг нейронної мережі, що має апріорі задану топологію. Ваги кодуються у вигляді двійкових послідовностей (хромосом). Кожна особина популяції характеризується повною множиною ваг нейронної мережі. Оцінка пристосованості особин визначається функцією пристосованості, що задається у вигляді суми квадратів похибок, тобто різниць між очікуваними (еталонними) і фактично отриманими значеннями на виході мережі для різних вхідних даних.

Генетичні алгоритми для вибору топології нейронних мереж. В якості найбільш очевидного способу об'єднання генетичного алгоритму з нейронною

мережею інтуїтивно сприймається спроба закодувати в генотипі топологію об'єкта (в даному випадку - нейронної мережі) із зазначенням кількості нейронів і зв'язків між ними при подальшому визначенні ваг мережі за допомогою будь-якого відомого методу [39].

Проектування оптимальної топології нейронної мережі може бути представлено у вигляді пошуку такої архітектури, яка забезпечує найкраще (щодо обраного критерію) вирішення конкретної задачі. Такий підхід передбачає перебір простору архітектур, складений з усіх можливих варіантів, і вибір точки цього простору, яка є найкращою щодо заданого критерію оптимальності [49].

Як тільки певний вид еволюції вводиться в штучну нейронну мережу, відразу виникає потреба у відповідній йому схемі хромосомного представлення даних, тобто повинен бути створений спосіб генетичного кодування особин популяції. Розробка способу кодування вважається першим етапом такого еволюційного підходу, поряд з яким типовий процес еволюції включає наступні кроки [47]:

- декодування;
- навчання;
- оцінювання пристосованості;
- репродукція;
- формування нового покоління.

Наведена на рисунку 2.9 блок-схема є актуальною, вона відображає і класичний генетичний алгоритм, і так звані еволюційні програми, які засновані на генетичному підході і узагальнюють його принципи. Отже, цій універсальній блок-схемі відповідають різні еволюційні алгоритми, і в кожному з них в першу чергу повинна бути згенерована вихідна популяція хромосом.

Еволюційний підхід до навчання нейронних мереж складається з двох основних етапів. Перший з них – це вибір відповідної схеми представлення ваг зв'язків. Він полягає в прийнятті рішення – чи можна кодувати ці ваги двійковими послідовностями чи потрібна якась інша форма. На другому етапі вже здійснюється сам процес еволюції, що базується на генетичному алгоритмі.

Після вибору схеми хромосомного представлення генетичний алгоритм застосовується до популяції особин (хромосом, що містять закодовану множину ваг нейронної мережі) з реалізацією типового циклу еволюції, що складається з чотирьох кроків.

1) Декодування кожної особини (хромосоми) поточного покоління для відновлення множини ваг і конструювання відповідної цій множині нейронної мережі з апріорно заданою архітектурою і правилом навчання.

2) Розрахунок загальної середньоквадратичної похибки між фактичними і заданими значеннями на всіх виходах мережі при подачі на її входи навчальних образів. Ця похибка визначає пристосованість особини (сконструйованої мережі); в залежності від виду мережі функція пристосованості може бути задана і іншим чином.

3) Репродукція особин з ймовірністю, що відповідає їх пристосованості, або відповідно до їх рангу (в залежності від способу селекції - наприклад, за методом рулетки або рангового методу).

4) Застосування генетичних операторів - таких як схрещування, мутація і / або інверсія для отримання нового покоління.

Блок-схема, що ілюструє еволюцію ваг, представлена на рисунку 2.12.

На рисунку 2.12 передбачалося, що архітектура мережі задається апріорно і не змінюється в процесі еволюції ваг. Як вибрати архітектуру мережі? Відомо, що архітектура робить вирішальний вплив на весь процес обробки інформації нейронною мережею. На жаль, найчастіше вона підбирається експертами методом проб і помилок. В таких умовах спосіб оптимального (або майже оптимального) проектування архітектури нейронної мережі для конкретної задачі виявився б дуже корисним. Один з можливих підходів полягає в еволюційному формуванні архітектури із застосуванням генетичного алгоритму.

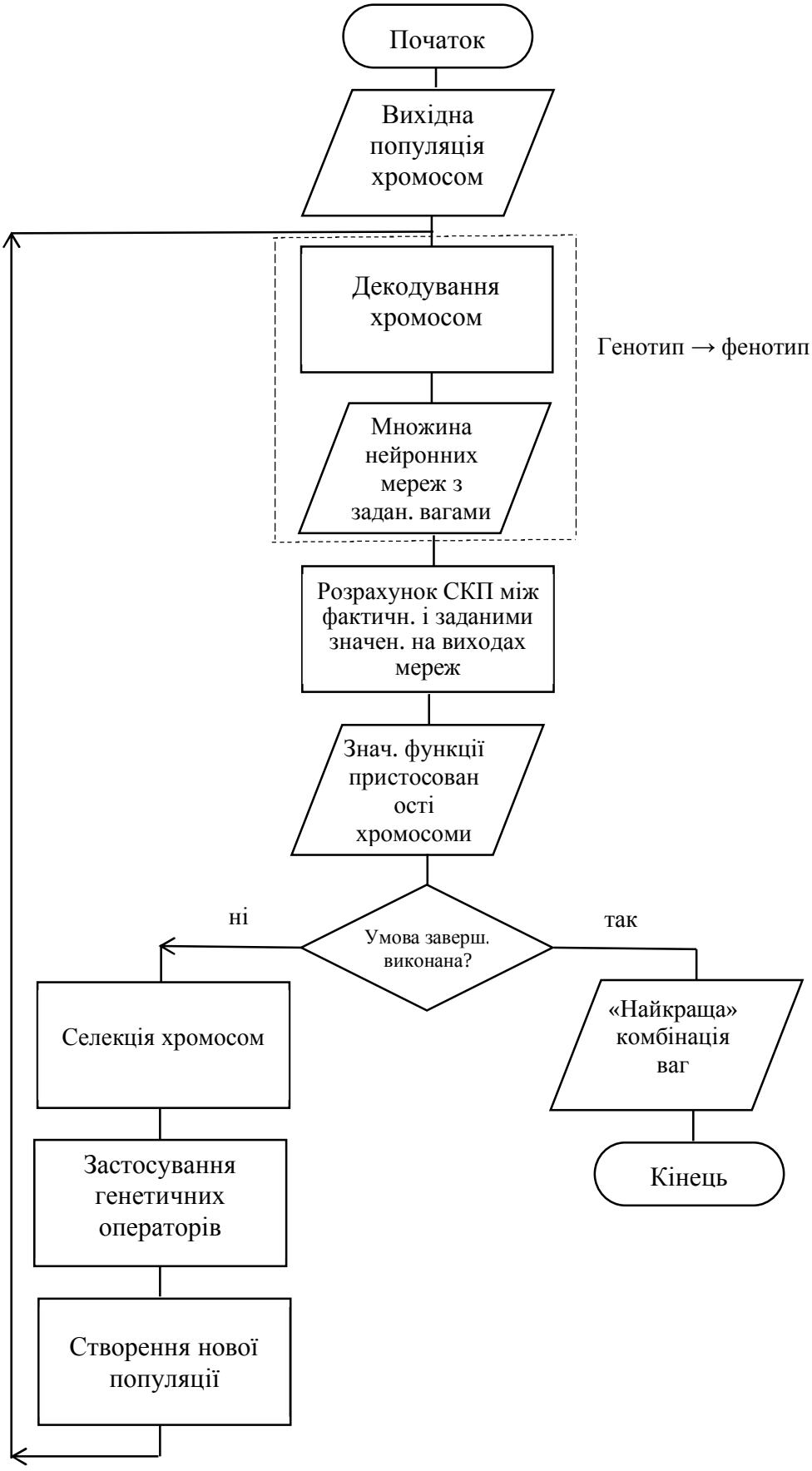


Рисунок 2.12 – Блок-схема генетичного алгоритму пошуку найкращого набору ваг нейронної мережі (випадок еволюції ваг)

Другий етап еволюційного проектування архітектури нейронної мережі полягає (відповідно до типового циклу еволюції) з наступних кроків:

1) Декодування кожної особини поточної популяції для опису архітектури нейронної мережі.

2) Навчання кожної нейронної мережі з архітектурою, отриманою на першому кроці, за допомогою заздалегідь заданого правила (деякі його параметри можуть адаптивно уточнюватися в процесі навчання). Навчання повинно починатися при різних випадково вибраних початкових значеннях ваг і (при необхідності) параметрів правила навчання.

3) Оцінювання пристосованості кожної особини (закодованої архітектури) за досягнутими результатами навчання, тобто за найменшою цілою середньоквадратичною похибкою навчання або на основі тестування, якщо найбільший інтерес викликає здатність до узагальнення, найменша тривалість навчання або спрощення архітектури (наприклад, мінімізація кількості нейронів і зв'язків між ними).

4) Репродукція особин з ймовірністю, що відповідає їх пристосованості або рангу в залежності від використовуваного методу селекції.

5) Формування нового покоління в результаті застосування таких генетичних операторів, як схрещування, мутація і / або інверсія.

Блок-схема, що ілюструє еволюцію архітектур, представлена на рисунку 2.13.

Еволюція правил навчання. Відомо, що для різних архітектур і задач навчання потрібні різні алгоритми навчання. Пошук оптимального (або майже оптимального) правила навчання, як правило, відбувається з урахуванням експертних знань і часто – методом проб і помилок.

Схема хромосомного представлення у разі еволюції правил навчання повинна відображати динамічні характеристики. Статичні параметри (такі як архітектура чи значення ваг мережі) кодувати значно простіше. Спроба створення універсальної схеми представлення, яка дозволила б описувати довільні види динамічних характеристик нейронної мережі, наперед приречена

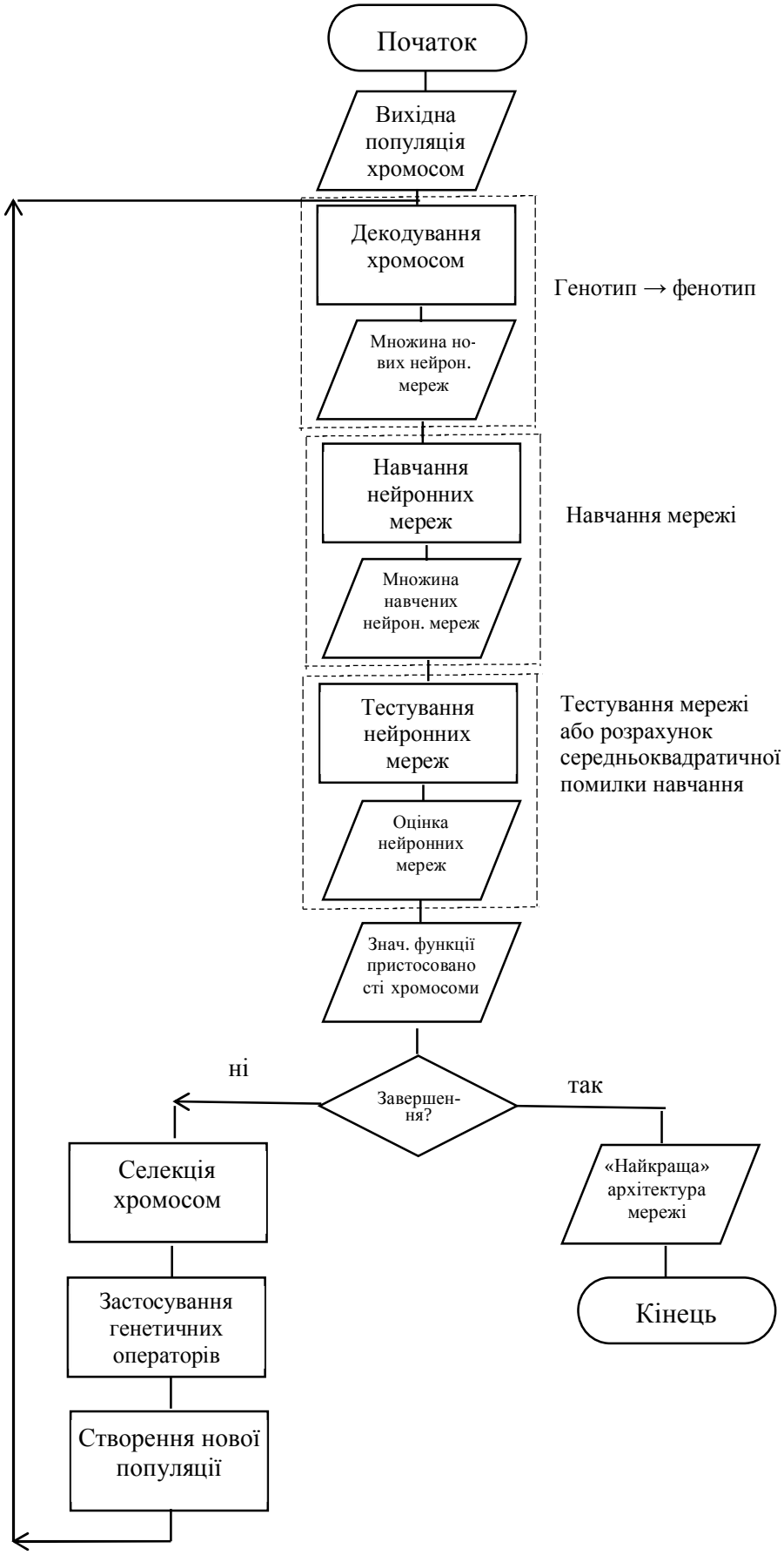


Рисунок 2.13 – Блок-схема генетичного алгоритму для пошуку найкращої архітектури нейронної мережі (випадок еволюції архітектур)

на невдачу, оскільки передбачає невиправдано великий обсяг обчислень, необхідних для перегляду всього простору правил навчання. З цієї причини на тип динамічних характеристик зазвичай накладаються певні обмеження, що дозволяє вибрати загальну структуру правила навчання. Найчастіше встановлюється, що для всіх зв'язків нейронної мережі має застосовуватися одне і те ж правило навчання, яке може бути задано функцією виду [47]

$$\Delta w(t) = \sum_{k=1}^{n} \sum_{i_1,\ldots,i_k=1}^{n} \left[\theta_{i_1,i_2,\ldots,i_k} \prod_{j=1}^{k} \mathrm{x}_{i_j}(t-1) \right] \quad (4.18)$$

де t – час, Δw – приріст ваги, x_{i_j} – так звані локальні змінні, $\theta_{i_1,i_2,\ldots,i_k}$ – дійсні коефіцієнти.

Головна мета еволюції правил навчання полягає в підборі відповідних значень коефіцієнтів $\theta_{i_1,i_2,\ldots,i_k}$.

З урахуванням великої кількості компонентів рівняння (4.18), що може зробити еволюцію занадто повільною і практично неефективною, часто вводяться додаткові обмеження, засновані на евристичних посилках [6].

Представимо типовий цикл еволюції правил навчання.

1. Декодування кожної особини поточної популяції для опису правила навчання, яке буде використовуватися в якості алгоритму навчання нейронних мереж.

2. Формування множини нейронних мереж з випадково згенерованими архітектурами і початковими значеннями ваг, а також оцінювання цих мереж з урахуванням їх навчання за правилом, отриманим на кроці 1, в категоріях точності навчання або тестування, тривалості навчання, складності архітектури і т.п.

3. Розрахунок значення пристосованості кожної особини (закодованого правила навчання) на основі отриманої на кроці 2 оцінки кожної нейронної мережі, що представляє собою своєрідний вид зваженого усереднення.

4. Репродукція особин з ймовірністю, що відповідає їх пристосованості або рангу в залежності від використовуваного методу селекції.

5. Формування нового покоління в результаті застосування таких генетичних операторів, як схрещування, мутація і / або інверсія.

Блок-схема, що ілюструє еволюцію правил навчання, представлена на рисунку 2.14.

2.4 Висновки до розділу 2

Результатами другого розділу є:

− проведений аналіз алгоритмів структурного синтезу згорткових нейронних мереж, зокрема: «І/АБО» граф, метод віток і границь, побудова морфологічної матриці;

− проведений аналіз алгоритмів оптимізації з використанням побудови множини Парето;

− проведений аналіз алгоритмів оптимізації з використанням генетичних алгоритмів.

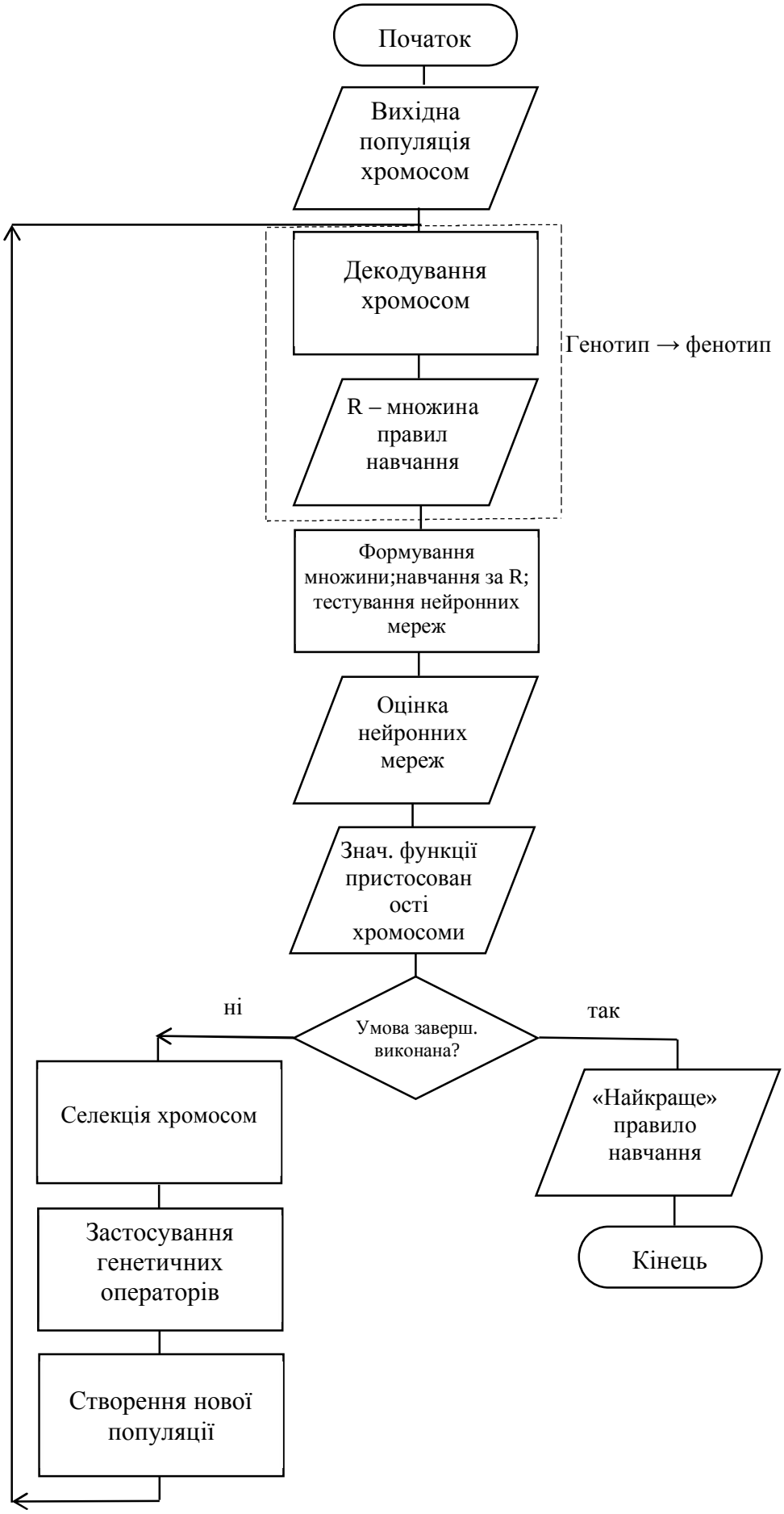


Рисунок 2.14 – Блок-схема ГА для пошуку найкращого правила навчання

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ГЕНЕРАТОРА СИНТЕЗУ І ОПТИМІЗАЦІЇ СТРУКТУР ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ

3.1 Системні вимоги

Для того щоб розроблений програмний модуль функціонував на персональному комп'ютері, останній має відповідати наступним мінімальним системним вимогам:

– оперативна пам'ять обсягом 8 Гб та поколінням пам'яті DDR4 або DDR3 (наприклад, Kingston DDR4-2400 HyperX Fury);

– процесор з тактовою частотою не меншою ніж 2,2 ГГц;

– відеокарта з підтримкою технології CUDA, обсягом пам'яті не меншим ніж 8 Гб та з поколінням пам'яті не нижчим GDDR5 (GeForce GTX 1070 8 GB);

– жорсткий диск розміром 50 Гб.

Для функціонування програмного модуля на комп'ютері користувача має бути встановлене наступне програмне забезпечення:

– операційна система Windows 10 ×64 або Ubuntu ×64;

– середовище Python 3.6

– середовище Anaconda із встановленим Tensorflow ≥1.13 (рисунок 3.1);

– останні версії драйверів для відеокарти.

Технологія CUDA являє собою програмно-апаратну архітектуру паралельних обчислень. CUDA дозволяє істотно збільшити обчислювальну продуктивність завдяки використанню графічних процесорів (GPUs) фірми Nvidia (рисунок 3.2).

CUDA SDK дозволяє включати в текст програм на C виклик підпрограм, що виконуються на графічних процесорах Nvidia. Це розроблено шляхом команд, які записуються на C. Архітектура CUDA дає розроблювачу шанс на свій розсуд організовувати доступ до набору інструкцій графічного прискорювача й керувати його пам'яттю.

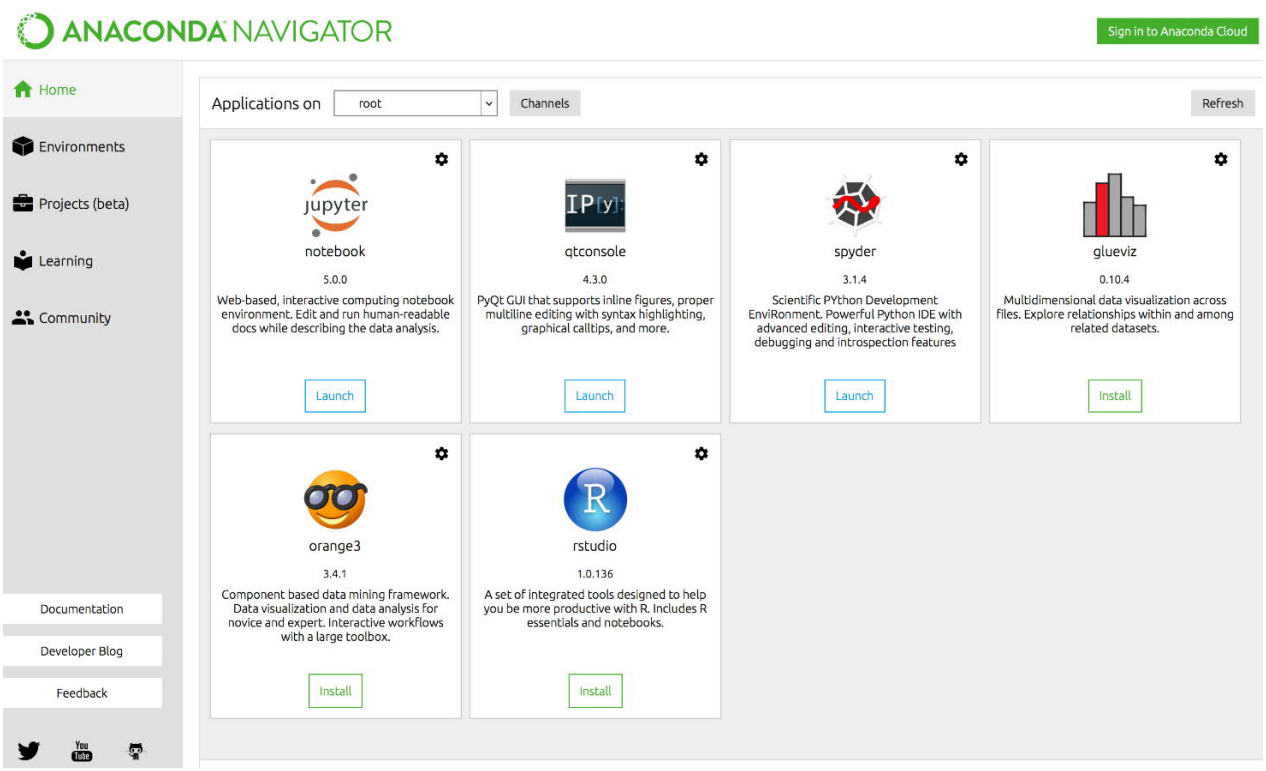


Рисунок 1.1 – Початковий інтерфейс Anaconda

Рисунок 3.2 – Логотип CUDA

Для паралельних обчислень існують спеціальні відеокарти сімейства QUADRO і TESLA.

Архітектура CUDA, завдяки можливостям графічних API, має переваги над традиційним підходом до організації обчислень загального призначення. Відзначають такі переваги [66]:

– інтерфейс програмування додатків CUDA базується на стандартній мові програмування С з деякими обмеженнями;

– спільна між потоками пам'ять розміром в 16 Кб може бути використана під організований користувачем кеш з більш широкою смугою пропускання, ніж при вибірці зі звичайних текстур;

– більш ефективні транзакції між пам'яттю центрального процесора і відеопам'яттю;

– повна апаратна підтримка цілочисельних і побітових операцій;

– підтримка компіляції GPU коду засобами відкритого LLVM.

Більшість сучасних відеокарт підтримують технологію CUDA. У таблиці 3.1 наведено перелік відеокарт, що підтримують CUDA.

Графічні процесори на даний час знайшли своє застосування в багатьох галузях. У галузі медицини для візуалізації графічної інформації у деяких випадках графічні процесори є важливими для забезпечення практичного використання алгоритмів, які вимагають обчислень. Використання графічних процесорів таке широке головним чином тому, що вони можуть різко прискорити паралельні обчислення, бути доступними та енергоефективними.

Розглянемо типи пам’яті DDR, DDR2, DDR3, DDR4. Синхронна динамічна пам’ять з випадковим доступом з подвійною швидкістю передачі даних, офіційно скорочена як DDR SDRAM, являє собою клас інтегрованих схем пам’яті синхронної динамічної оперативної пам'яті (SDRAM) із подвійною швидкістю передачі даних (DDR). DDR SDRAM, який також називається DDR1 SDRAM, був замінений DDR2 SDRAM, DDR3 SDRAM і DDR4 SDRAM, і незабаром буде замінений DDR5 SDRAM.

Таблиця 3.1 – Список підтримуваних відеокарт

Версія специфікації	GPU	Відеокарта
1	2	3
1.0	G80, G92, G92b, G94, G94b	GeForce 8800GTX/Ultra, Tesla C/D/S870, FX4/5600, 360M, GT 420
1.1	G86, G84, G98, G96, G96b, G94, G94b, G92, G92b	GeForce 8400GS/GT, 8600GT/GTS, 8800GT/GTS, 9400GT, 9600 GSO, 9600GT, 9800GTX/GX2, 9800GT, GTS 250, GT 120/30/40, FX 4/570, 3/580, 17/18/3700, 4700x2, 1xxM, 32/370M, 3/5/770M, 16/17/27/28/36/37/3800M, NVS420/50
1.2	GT218, GT216, GT215	GeForce 210, GT 220/40, FX380 LP, 1800M, 370/380M, NVS 2/3100M
1.3	GT200, GT200b	GeForce GTX 260, GTX 275, GTX 280, GTX 285, GTX 295, Tesla C/M1060, S1070, Quadro CX, FX 3/4/5800
2.0	GF100, GF110	GeForce (GF100) GTX 465, GTX 470, GTX 480, Tesla C2050, C2070, S/M2050/70, Quadro Plex 7000, Quadro 4000, 5000, 6000, GeForce (GF110) GTX 560 TI 448, GTX570, GTX580, GTX590
2.1	GF104, GF114, GF116, GF108, GF106	GeForce 610M, GT 430, GT 440, GT 640, GTS 450, GTX 460, GTX 550 Ti, GTX 560, GTX 560 Ti, 500M, Quadro 600, 2000
3.0	GK104, GK106, GK107	GeForce GTX 690, GTX 680, GTX 670, GTX 660 Ti, GTX 660, GTX 650 Ti, GTX 650, GeForce GTX 680MX, 645M, GeForce GT 640M

Продовження таблиці 3.1

1	2	3
3.5	GK110, GK208	GeForce GTX TITAN, GeForce GTX TITAN Black, GeForce GTX 780 Ti, GeForce GTX 780
5.0	GM107, GM108	GeForce GTX 750 Ti, GeForce GTX 750 , GeForce GTX 860M, GeForce GTX 850M, GeForce 840M, GeForce 830M
5.2	GM200, GM204, GM206	GeForce GTX Titan X, GeForce GTX 980 Ti, GeForce GTX 980, GeForce GTX 970, GeForce GTX 960, GeForce GTX 950, GeForce GTX 970M, GeForce GTX 965M

Жоден з його наступників не є сумісним із DDR1 SDRAM, тобто модулі пам'яті DDR2, DDR3, DDR4 та DDR5 не працюватимуть на материнських платах, обладнаних DDR1, і навпаки. Порівняно з SDRAM з однією швидкістю передачі даних (SDR), інтерфейс DDR SDRAM робить можливим більш високі швидкості передачі даних завдяки більш суворому контролю синхронізації електричних даних та тактових сигналів. Для реалізації часто доводиться використовувати такі схеми, як фазові петлі та самокалібрування, щоб досягти необхідної точності синхронізації.

При передачі даних 64 біт за раз DDR SDRAM дає швидкість передачі (в байтах / с) (тактова частота шини пам'яті) × 2 (для подвійної швидкості) × 64 (кількість переданих бітів) / 8 (кількість бітів / байт). Таким чином, при частоті шини 100 МГц DDR SDRAM дає максимальну швидкість передачі 1600 МБ / с.

На рисунку 3.3 приведена різниця між типами пам’яті DDR, DDR2, DDR3, DDR4. Звернемо увагу на різницю між двома останніми. DDR3 був вперше випущений у 2007 році та використовувався у всьому, від LG1313, LGA1151 (лише 6-й / 7-й Gen Core), а також AMD / AM3 / AM3 + та FM1 / 2/2 +. Однак усі сучасні платформи (починаючи з 2017 року) перейшли на режим

DDR4, і більшість платформ Intel відійшли від DDR3 за допомогою процесорів 6-го покоління Skylake.

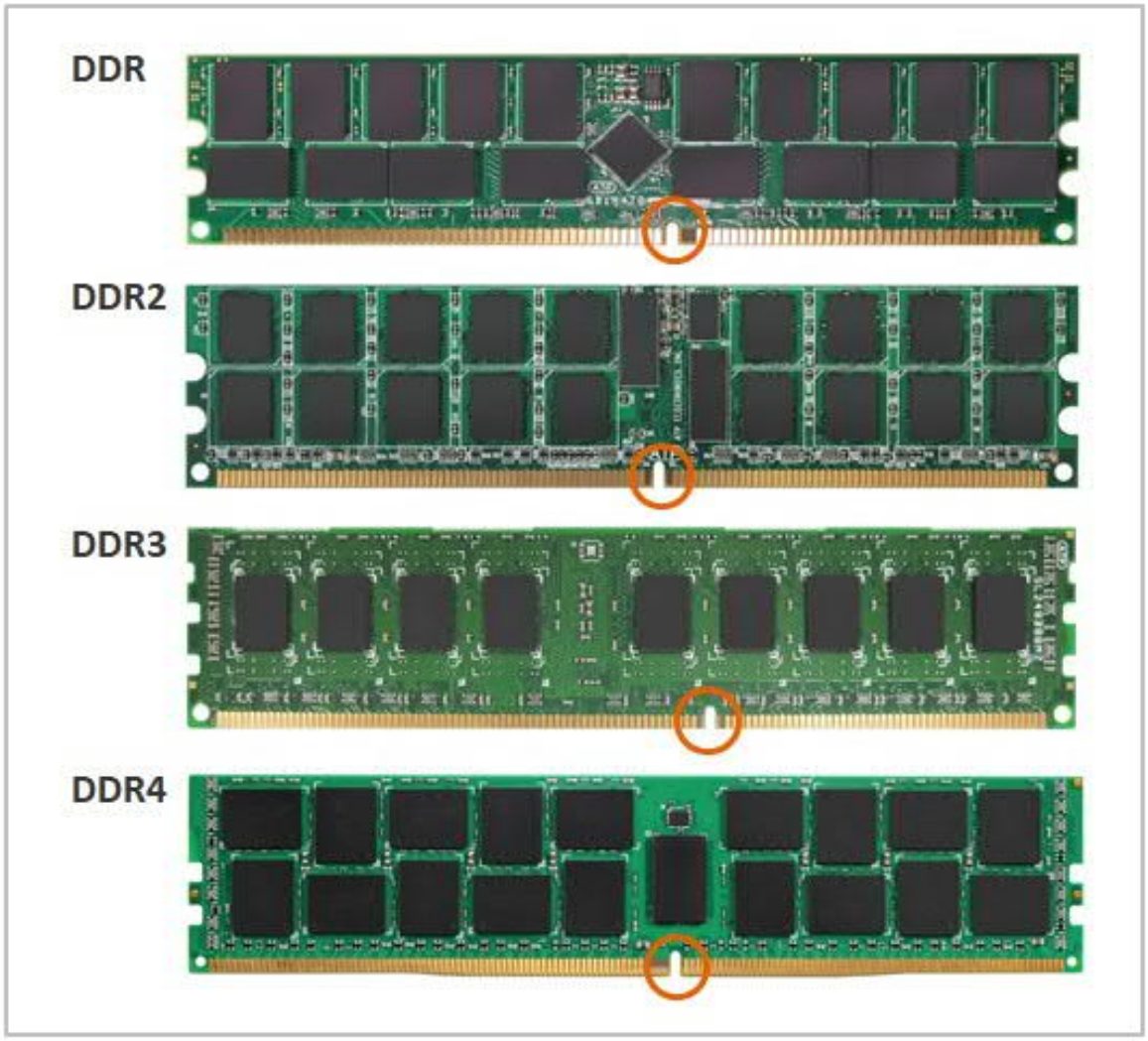


Рисунок 3.3 – Різниця між типами пам'яті

DDR4 працює при меншій напрузі (зазвичай на 1,2 вольта), ніж DDR3. Різниця в напрузі може призвести до економії 15 Вт в порівнянні з DDR3.

Також DDR3 та DDR4 відрізняються швидкістю. Специфікації DDR3 офіційно починаються від 800 MT/c (або мільйони передач в секунду) і закінчуються на DDR3 – 2133. DDR4 запускається на частоті 1600 МГц, з офіційною підтримкою до DDR4 – 3200 – і набори розгону можуть набирати – 4800. Підвищена швидкість означає загальне збільшення пропускної здатності.

3.2 Апаратне та програмне забезпечення

Для проведення першого експерименту використано комп’ютер з такими параметрами (таблиця 3.2).

Таблиця 3.2 – Конфігурація комп’ютера

Процесор	AMD Athlon 64 x2 Dual Core, 2.51ГГц
Оперативна пам’ять	4ГБ, DDR2
Відеокарта	Nvidia GTX 950 Asus, серія Strix, 2Гб, GDDR5
Операційна система	Windows 10 Pro

Дана конфігурація не є достатньою для даного проекту. Тому необхідно збільшити обсяг оперативної пам’яті та кількість ядер.

Оптимальний варіант для реалізації поставленого завдання приведено у таблиці 3.3.

Таблиця 3.3 – Характеристики оптимального варіанту комп’ютера

Процесор	2 ядра + 13Гб пам’яті
Диск	100Гб
Графічний процесор	Nvidia Tesla K80, 12Гб

Для реалізації поставленого завдання можна скористатися створенням віртуальної машини наступними сервісами:

- Google Cloud Platform;
- Amazon Web Services;
- Google Colaboratory.

Такий варіант оптимального варіанту комп’ютера – це створення віртуальної машини у сервісі Google Cloud Platform. Віртуальний комп’ютер у

даному сервісі має інструменти для керування та надає деякі модульні хмарні служби, серед яких є обчислення, зберігання даних, аналіз даних та машинне навчання. Користувач може обрати характеристики віртуальної машини, збільшити або зменшити обсяг оперативної пам’яті та кількості ядер.

Іншим можливим варіантом є створення віртуальної машини на сервісі AWS (Amazon Web Services). AWS надає платформу хмарних обчислень в оренду приватним особам, компаніям та урядам на основі платної чи безкоштовної підписки (протягом перших 12 місяців). Технологія через Інтернет надає повноцінний віртуальний кластер комп'ютерів. Віртуальні комп'ютери AWS мають:

- апаратні пристрої – процесор, відеокарту, локальну та оперативну пам'ять, жорсткий диск або SSD-накопичувач;
- операційну систему (на вибір);
- мережу;
- прикладні програми – веб-сервер, база даних, CRM і т. д.

AWS має віртуальний консольний ввід/вивід (клавіатура, дисплей і миша), тому користувач AWS може підключитися до своєї системи AWS за допомогою браузера. Через браузер користувач заходить у віртуальний комп'ютер, входить в систему, налаштовує та використовує свої віртуальні системи так само, як і для фізичного комп'ютера. Користувач після налаштування може здійснювати інтернет-орієнтовані сервіси та послуги.

На рисунку 3.4 приведена порівняльна таблиця конфігурацій і цін за годину VMs з підтримкою GPU на AWS.

Сервіс Google Colaboratory надає можливість користувачеві запускати Jupyter Notebook («ноутбук», інтерактивне виконання коду на мові Python). Існує можливість запускати не тільки Jupyter Notebook, а й звичайні файли коду Python . Також в цьому сервісі можна використовувати графічний процесор Nvidia Tesla K80. В сервісі Google Colaboratory можна працювати 12 годин. Користувач повинен вкластися в цей час, закінчити виконувані процеси, навчити нейронну мережу, зберегти всі дані.

В таблиці 3.4 приведені технічні характеристики Jupyter Notebook.

Name	GPUs	vCPUs	RAM (GiB)	Network Bandwidth	Price/Hour*	RI Price / Hour**
p2.xlarge	1	4	61	High	$0.900	$0.425
p2.8xlarge	8	32	488	10 Gbps	$7.200	$3.400
p2.16xlarge	16	64	732	20 Gbps	$14.400	$6.800

Рисунок 3.4 – VMs з підтримкою GPU на AWS та їх ціна за годину

Таблиця 3.4 – Технічні характеристики Jupyter Notebook сервісу Google Colaboratory

Процесор	Intel Xeon 2,3ГГц, 1 ядро
Диск	320Гб
Пам’ять	12,6Гб
Графічний процесор	Nvidia Tesla K80, 12Гб
Кожні 12 годин дані із диска, оперативної пам’яті, кешу процесора тощо, які будуть розміщені на виділеній віртуальній машині, будуть стерті	

В сервісі Google Colaboratory встановлені за замовчуванням популярні бібліотеки для машинного навчання, такі як Tensorflow, PyTorch, які користувач використовує при написанні коду.

Для втілення проекту застосовано третій варіант.

3.3 Засоби та платформа реалізації

Згідно даних GitHub (сервіс для хостингу IT-проектів і їх спільного розвитку) за 2019 рік найпопулярнішими мовами програмування, що використовуються для машинного навчання стали 10 мов [67]. Список складений на основі кількості репозиторіїв, автори яких вказують, що в їх додатках використовуються МН-алгоритми.

В дослідженнях вказано, що для розробки програм, які використовують алгоритми машинного навчання, найчастіше застосовуються Python і C ++.

1. Python – найпопулярніша мова програмування серед розробників МН-програм в GitHub. Python має набір попередньо налаштованих інструментів для впровадження МН-моделей і алгоритмів, документовану бібліотеку Scikit-Learn з великої кількістю алгоритмів машинного навчання, бібліотеку ChatterBot, призначену для обробки мови і навчання на наборах даних в форматі діалогів.

2. C ++ зайняв високу позицію завдяки МН-бібліотеці Google TensorFlow, в якій акцент зроблений на нейромережах. C ++ має розподілену високопродуктивну платформу для градієнтного бустінга Microsoft LightGBM (підвищує швидкість і ефективність навчання МН-моделі), бібліотеку Turi Create (спрощує розробку користувача моделей машинного навчання для початківців розробників).

3. JavaScript.

4. Java.

5. C#.

6. Julia.

7. Shell.

8. R.

9. TypeScript.

10. Scala.

Для реалізації поставленого завдання обрано фреймворк Tensorflow версії 1.14 – це відкрита програмна бібліотека для машинного навчання, розроблена компанією Google , де зараз її і застосовують для досліджень та розробки власних продуктів.

В якості віртуальної машини обрано сервіс Google Colaboratory та мову програмування Python 3.6.

3.4 Початкова архітектура та вхідні параметри

Початковою моделлю мережі стала наступна модель, яка складається із таких шарів:

– згортка;

– активація ReLU;

– згортка;

– активація ReLU;

– макс-пулінг 2d;

– dropout;

– згортка;

– активація ReLU;

– згортка;

– активація ReLU;

– dropout;

– flatten;

– повнозв’язний шар;

– повнозв’язний шар.

Вхідними даними для мережі є кольорові цитологічні зображення 64×64 пікселі, що розділені на чотири класи: цито-фібро-кістозна мастопатія, цито-

кістозна мастопатія, цито-непроліферативна-фібромастопія, цито-непроліферативна мастопатія [68-73].

На рисунку 3.5 приведено приклад цитологічного зображення. Вся вибірка цитологічних зображень наведена на рисунку 3.6. Обсяг даної вибірки: 78 зображень. Для навчання глибокої згорткової нейронної мережі цього недостатньо. Цю проблему вирішено отриманням нової вибірки шляхом генерування зображень за допомогою GAN-мережі.

Згенерована вибірка має близько 1900 зображень (рисунок 3.7).

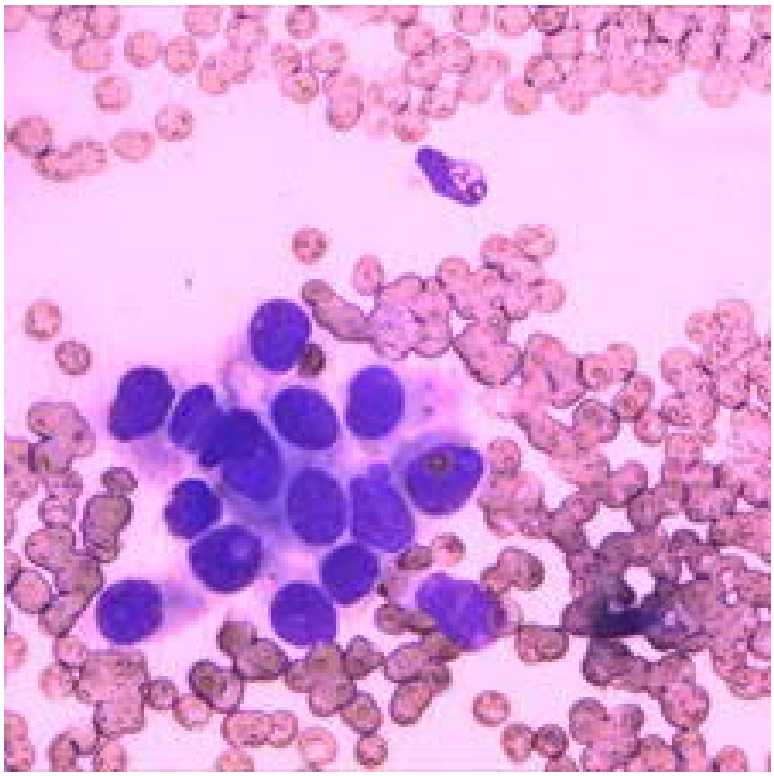


Рисунок 3.5 – Цитологічне зображення

3.5 Опис роботи програмної частини

Програма генерує структури згорткових нейронних мереж за допомогою методу повного перебору, який був описаний у підрозділі 2.1 (рисунок 2.3).

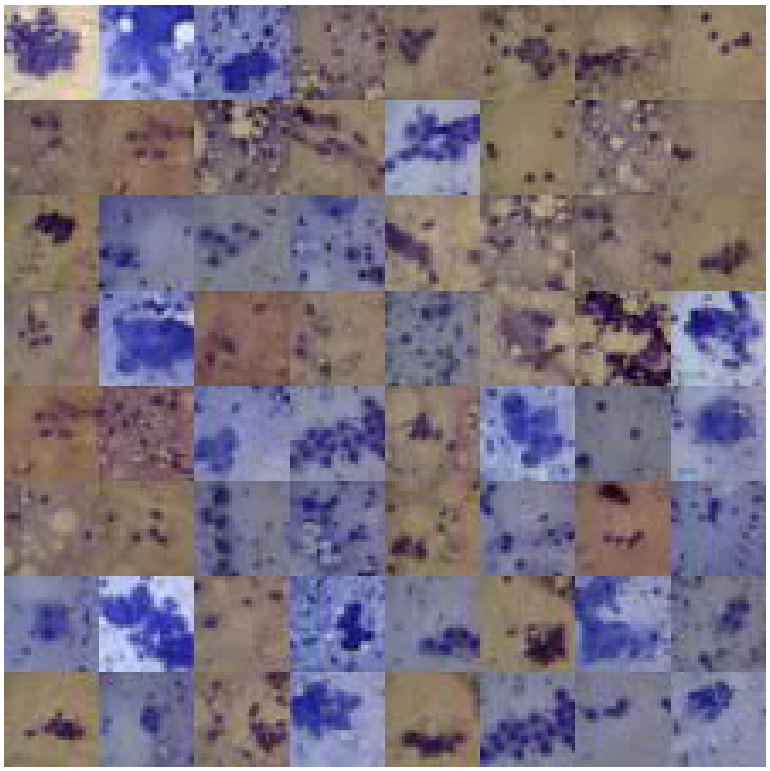


Рисунок 3.6 – Наявна вибірка цитологічних зображень

Користувач задає наступні параметри (рисунок 3.8):

– число повнозв'язних шарів;

– число фільтрів;

– число згорткових шарів.

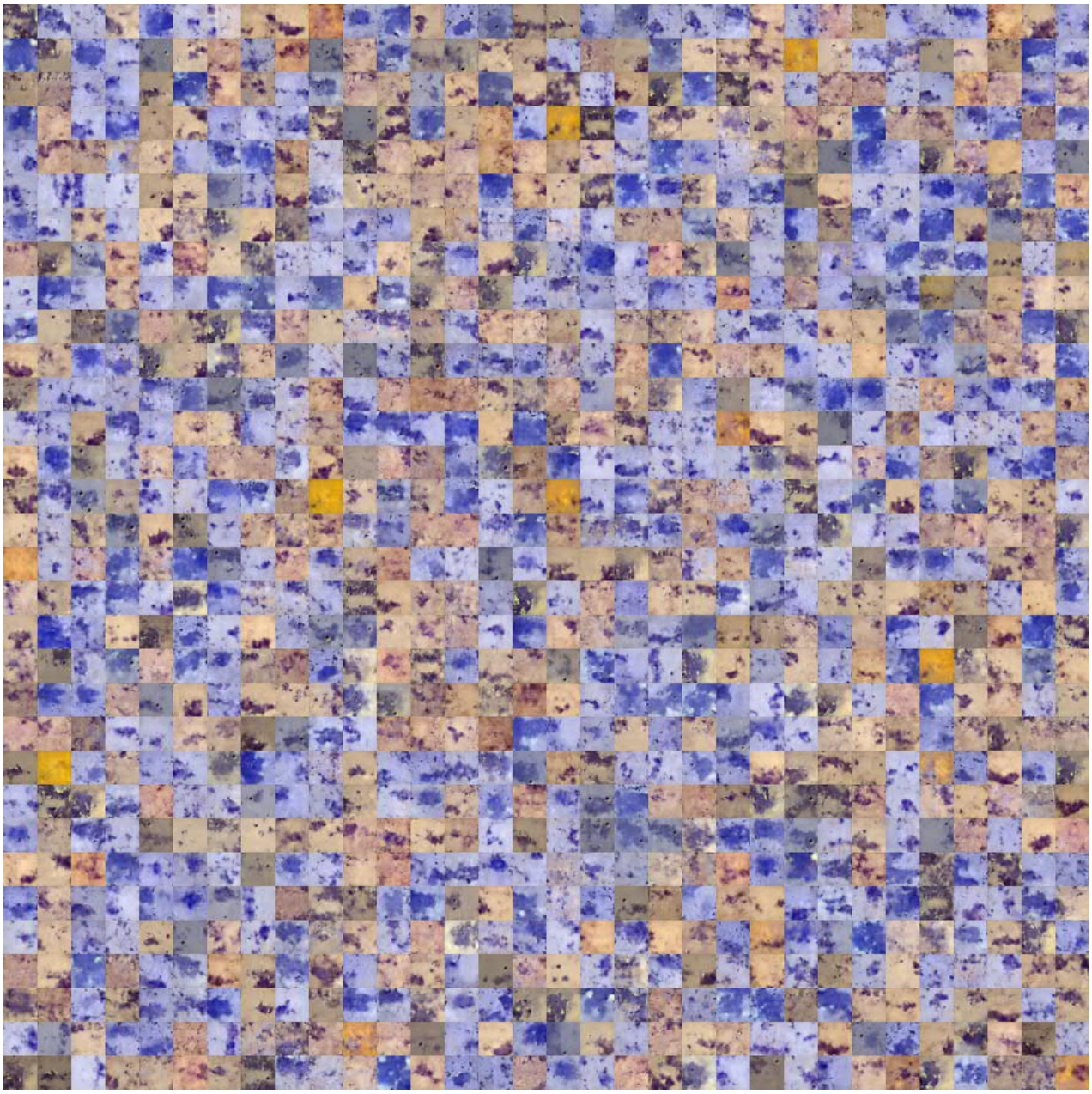


Рисунок 3.7 – Згенерована вибірка цитологічних зображень

```
dense_layers = [2]
filters = [64, 128, 256]
conv_layers = [2, 3]
```

Рисунок 3.8 – Параметри для генерування структури

Програма генерує структури згорткових нейронних мереж для всіх комбінацій, потім, кожна отримана мережа навчається протягом двадцяти епох на даних, що продемонстровані на рисунку 3.7.

Кожна з моделей оцінюється на тестових даних.

Для кожної моделі виводиться точність та ROC-крива. Після завершення навчання всіх моделей, користувачу виводиться комбінація параметрів, що показали найкращі результати.

Алгоритм роботи програми представлений на рисунку 3.9.

3.6 Тестування програмної частини

Початкові параметри були задані згідно з рисунком 3.8. Дані комбінації передбачають шість структур згорткових нейронних мереж ($1 \times 3 \times 2 = 6$).

Після навчання кожна модель отримала таку точність:

– 2-conv-with-64-filters-2-dense-with-64-units – точність 0.9538;

– 2-conv-with-128-filters-2-dense-with-128-units – точність 0.9612;

– 2-conv-with-256-filters-2-dense-with-256-units – точність 0.9530;

– 3-conv-with-64-filters-2-dense-with-64-units – 0.9460;

– 3-conv-with-128-filters-2-dense-with-128-units – 0.9499;

– 3-conv-with-256-filters-2-dense-with-256-units – 0.9643.

На рисунку 3.10 зображена ROC-крива для моделі №1.

Рисунок 3.9 – Схема роботи алгоритму

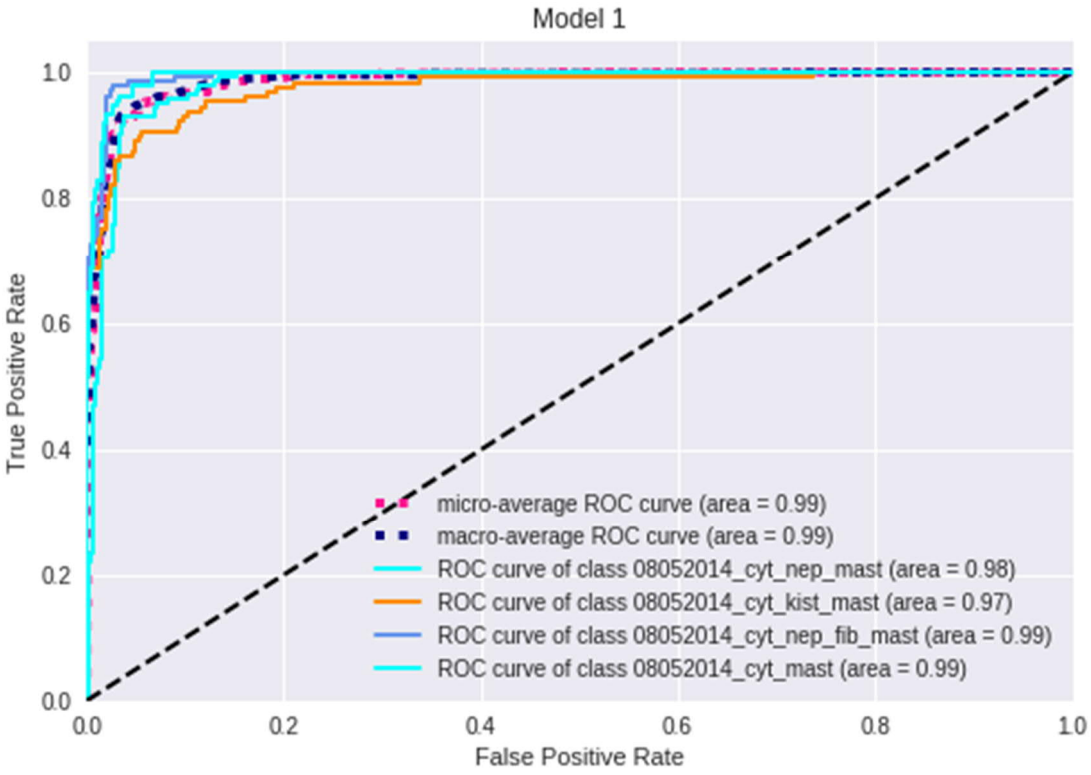


Рисунок 3.10 – ROC-крива для моделі №1

На рисунку 3.11 зображена ROC-крива для моделі №2.

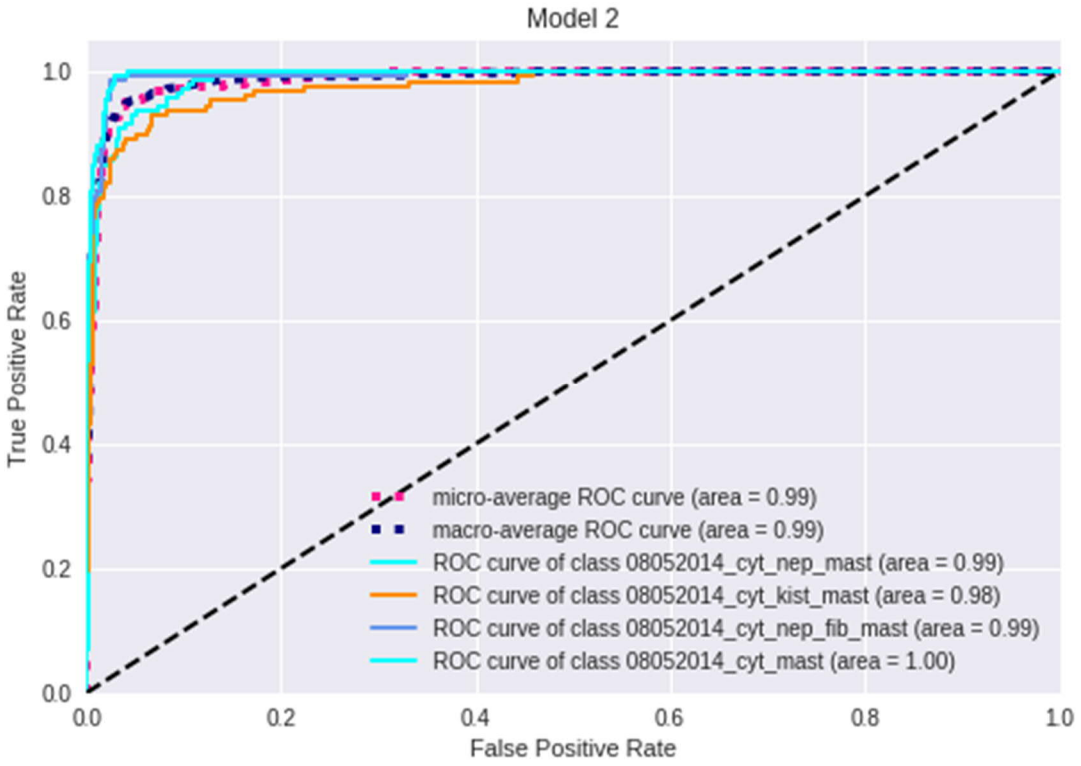


Рисунок 3.11 – ROC-крива для моделі №2

На рисунку 3.12 зображена ROC-крива для моделі №3.

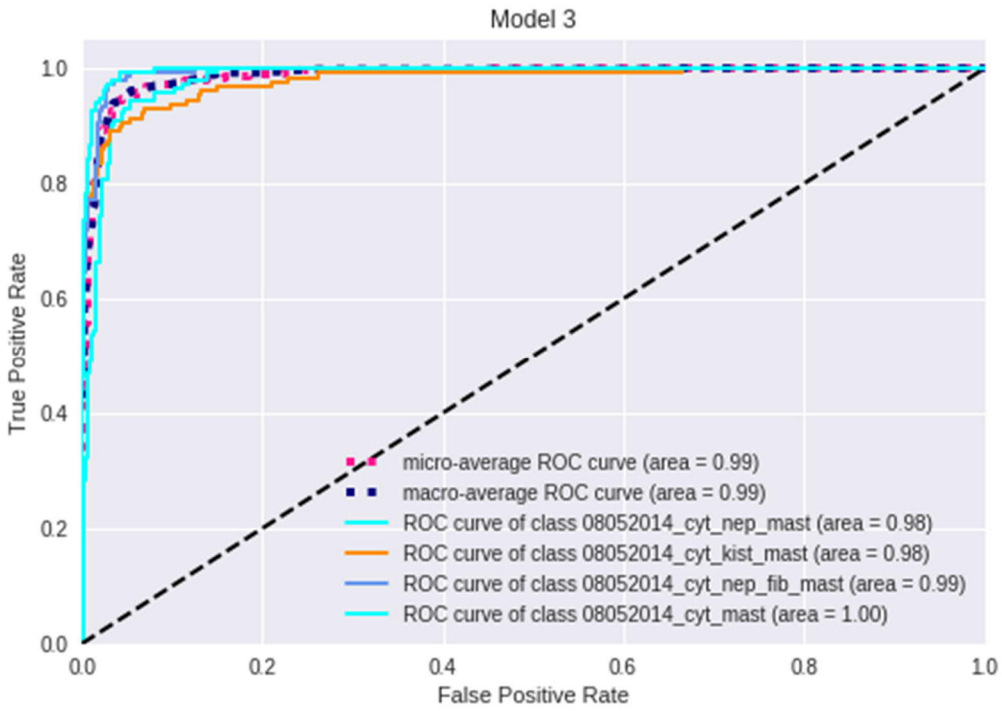


Рисунок 3.12 – ROC-крива для моделі №3

На рисунку 3.13 зображена ROC-крива для моделі №4.

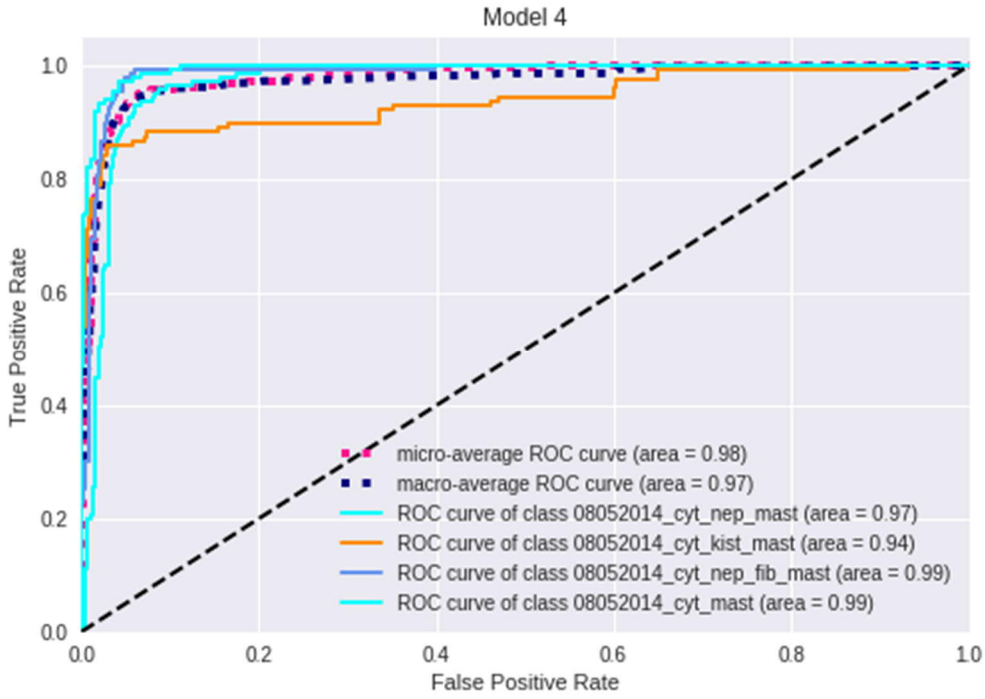


Рисунок 3.13 – ROC-крива для моделі №4

На рисунку 3.14 зображена ROC-крива для моделі №5.

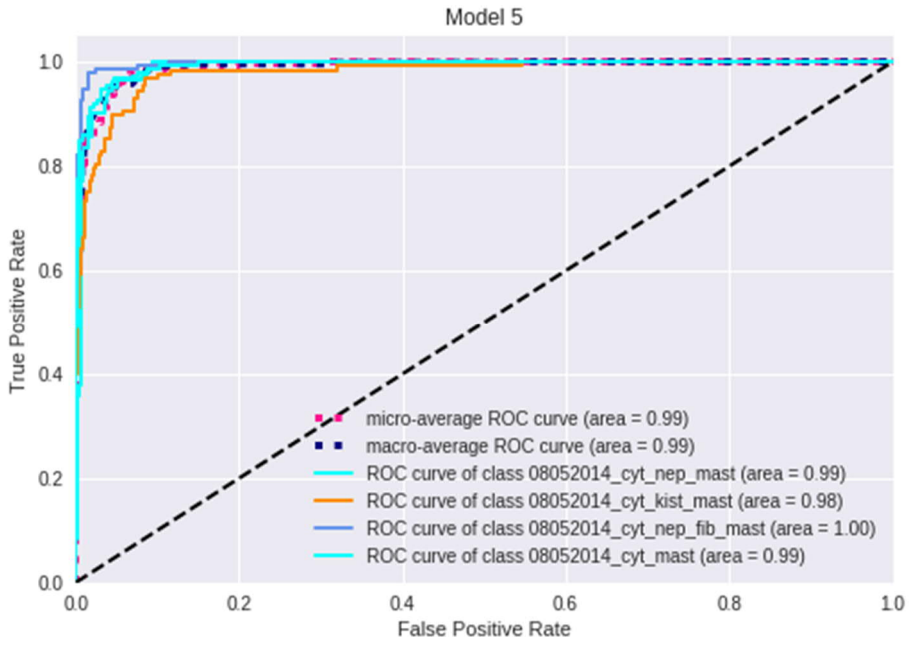


Рисунок 3.14 – ROC-крива для моделі №5

На рисунку 3.15 зображена ROC-крива для моделі №6.

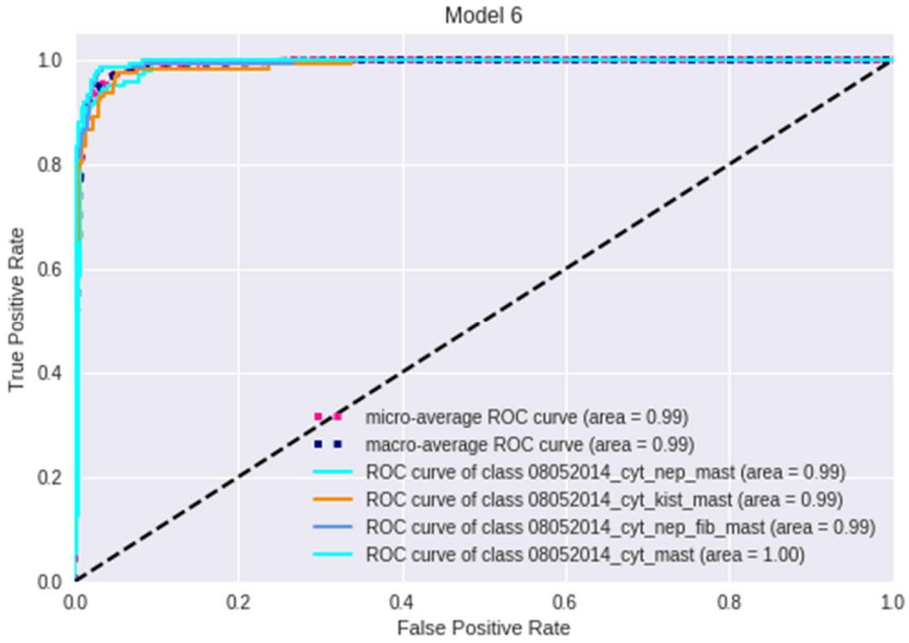


Рисунок 3.15 – ROC-крива для моделі №6

3.7 Висновки до розділу 3

Отже, в даному розділі отримані такі результати:

- Проаналізоване апаратне та програмне забезпечення, засоби та платформа реалізації.
- Розроблена архітектура та вхідні параметри.
- Зроблено опис роботи програмної частини.
- Проведено тестування програмної системи.

ВИСНОВКИ

1. Проаналізовано області застосування нейронних мереж, структури згорткових нейронних мереж та методи навчання нейронних мереж, що показало актуальність задачі синтезу структур ЗНМ.

2. Розроблено алгоритми структурного синтезу згорткових нейронних мереж: «І/АБО» граф, метод віток і границь, побудова морфологічної матриці; алгоритми оптимізації з використанням побудови множини Парето та алгоритми оптимізації з використанням генетичних алгоритмів, що дало змогу їх порівняти та програмно реалізувати.

3. Розроблено генератор структур згорткових нейронних мереж і оптимізовано його структуру на основі генетичних алгоритмів.

4. Програмно реалізовано синтез структур згорткових нейронних мереж, що дало змогу синтезувати множину структур згорткових нейронних мереж з високою точністю класифікації.

5. Проведено тестування розробленого генератора структур згорткових нейронних мереж з точністю класифікації гістологічних зображень близько 99 відсотків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Convolutional Neural Networks for Visual Recognition: веб-сайт. URL: https://cs231n.github.io/convolutional-networks/.

2. Гарматюк В.Р. Кухарук В.Р. Алгоритми оптимізації структур згорткових нейронних мереж: III Наук.-практ. конф. молодих вчених і студентів «Інтелектуальні комп'ютерні системи та мережі». 26 листопада 2020 р. Тернопіль, 2020. С. 49.

3. Кухарук В.Р., Гарматюк В. Р. Алгоритми генерування зображень на основі нейронних мереж: III Наук.-практ. конф. молодих вчених і студентів «Інтелектуальні комп'ютерні системи та мережі». 26 листопада 2020 р. Тернопіль, 2020. С. 50.

4. Методичні рекомендації до виконання дипломної роботи з освітньо-кваліфікаційного рівня «Магістр». Спеціальність «Комп'ютерні системи та мережі» / О.М. Березький, Л.О. Дубчак, Г.М. Мельник /Під ред. О.М. Березького. Тернопіль: ТНЕУ, 2016. 47 с.

5. Методичні вказівки до оформлення курсових проектів, звітів про проходження практики, випускних кваліфікаційних робіт для студентів спеціальності «Комп'ютерна інженерія» / І.В. Гураль, Л.О. Дубчак / Під ред. О.М. Березького. Тернопіль: ТНЕУ, 2019. 33 с.

6. Використання нейронних мереж – перспективна сфера науки і суспільства: веб-сайт. URL: http://oldconf.neasmo.org.ua/node/139.

7. Кальченко Д. Нейронные сети: на пороге будущого / КомпьютерПресс - 2005. N1: веб-сайт. URL: http: //www.compr.ru.

8. Проценко М.І., Васюра А.С. Нейронні мережі в медицині: веб-сайт. URL: http://ir.lib.vntu.edu.ua/bitstream/handle/123456789/29316/8908.pdf?sequence=3.

9. Кислова О. М., Бондаренко К. Б. Можливості застосування штучних нейронних мереж в аналізі соціологічної інформації // Вісник Харківського національного університету імені В.Н. Каразіна. 2010. № 891. С. 78-82.

10. Хлапонін Ю. І., Жиров Г. Б., Нікітчін О. М. Застосування нейронних мереж в статистичній системі аналізу і моніторингу телекомунікаційних мереж // Технологический аудит и резервы производства. № 5/2(31). 2016. С. 35-41.

11. Нейронні мережі: їх застосування, робота: веб-сайт. URL: https://www.poznavayka.org/uk/nauka-i-tehnika-2/neyronni-merezhi-yih-zastosuvannya-robota/.

12. Глибинне навчання: веб-сайт. URL: https://uk.wikipedia.org/wiki/Глибинне_навчання.

13. Данилов В. Я., Данілов В. Я., Краснощок І. О. Застосування згорткових нейронних мереж для розпізнавання зображень вибірки CIFAR-10. Системные технологии. 2017. № 1(108). С. 167-171.

14. A Tutorial on Deep Learning Part 2: Autoencoders, Convolutional Neural Networks and Recurrent Neural Networks. 2015.

15. Patrice Y. Simard, Dave Steinkraus, John C. Platt Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis.

16. Лавренюк М.С., Новіков О.М. Огляд методів машинного навчання для класифікації великих обсягів супутникових даних // Системні дослідження та інформаційні технології, 2018, № 1. С. 52-71.

17. LeCun Y. Deep learning / Y. LeCun, B. Yoshua, H. Geoffrey // Nature. Vol. 521, N 7553. 2015. P. 436–444.

18. Dong C. et al. Image super-resolution using deep convolutional networks [Text] //IEEE transactions on pattern analysis and machine intelligence. – 2016. – Т. 38. – №. 2. – С. 295-307.

19. Romano Y., Isidoro J., Milanfar P. RAISR: rapid and accurate image super resolution [Text] // IEEE Transactions on Computational Imaging. – 2017. – Т. 3. – №1. – С. 110-125.

20. K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in International Conference for Learning Representations, 2015.

21. J. Kim, J. K. Lee, and K. M. Lee, “Accurate image super-resolution using very deep convolutional networks,” in IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1646–1654.

22. Wenzhe Shi, Jose Caballero, Ferenc Huszar, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, Zehan Wang «Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network», 2016, pp. 1874–1883. https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Shi_Real-Time_Single_Image_CVPR_2016_paper.pdf.

23. Wu H., Gu X. Max-Pooling Dropout for Regularization of Convolutional Neural Networks. Neural Information Processing. 2015. No.9489.

24. Amari S.I. Statistical theory of learning curves under entropic loss criterion / S.I. Amari, N. Murata // Neural Computation. — Vol. 5, N 1. — 1993. — P. 140–153.

25. Щербина Ю. М. Методи навчання штучної нейронної мережі / Ю. М. Щербина, О. В. Годич // Вісник Національного університету "Львівська політехніка". – 2001. – № 438 : Інформаційні системи та мережі. – С. 160–170.

26. Schmidhuber J. Deep Learning in Neural Networks: an Overview // Neural Networks. 2015. Vol. 1. P. 85–117.

27. Machine learning explained: Understanding supervised, unsupervised, and reinforcement learning [Електронний ресурс] – Режим доступу до ресурсу: http://bigdata-madesimple.com/machine-learning-explained-understanding-supervisedunsupervised-and-reinforcement-learning/.

28. Hebb D.O. The Organization of Behavior. New York: Wiley. 1949. 335 p.

29. Созыкин А.В. Обзор методов обучения глубоких нейронных сетей // Вестник ЮУрГУ. Серия «Вычислительная математика и информатика». 2017. Т. 6, № 3. С. 28–59.

30. Novikoff A.B. On Convergence Proofs on Perceptrons. Symposium on the Mathematical Theory of Automata. 1962. Vol. 12. P. 615–622.

31. Rosenblatt F. The Perceptron: a Probabilistic Model for Information Storage and Organization in the Brain // Psychological Review. 1958. P. 65–386.

32. Widrow B., Hoff M. Associative Storage and Retrieval of Digital Information in Networks of Adaptive Neurons // Biological Prototypes and Synthetic Systems. 1962. Vol. 1. 160 p.

33. Narendra K.S., Thathatchar M.A.L. Learning Automata – a Survey // IEEE Transactions on Systems, Man, and Cybernetics. 1974. Vol. 4. P. 323–334.

34. Rosenblatt F. Principles of Neurodynamics; Perceptrons and the Theory of Brain Mechanisms. 1962. Washington: Spartan Books. 616 p.

35. Grossberg S. Some Networks That Can Learn, Remember, and Reproduce any Number of Complicated Space-Time Patterns // International Journal of Mathematics and Mechanics. 1969. Vol. 19. P. 53–91.

36. Kohonen T. Correlation Matrix Memories // IEEE Transactions on Computers. 1972. Vol. 100, No. 4. P. 353–359.

37. Von der Malsburg C. Self-Organization of Orientation Sensitive Cells in the Striate Cortex // Kybernetik. 1973. Vol. 14, No. 2. P. 85–100.

38. Willshaw D.J., von der Malsburg C. How Patterned Neural Connections Can Be Set Up by Self-Organization // Proceedings of the Royal Society London B. 1976. Vol. 194. P. 431–445.

39. Ivakhnenko A.G. Heuristic Self-Organization in Problems of Engineering Cybernetics. Automatica. 1970. vol. 6, no. 2. pp. 207–219.

40. Грицик В.В. Моделювання та синтез складних зображень симетричної структури / Грицик В.В., Березька К.М., Березький О. М. – Львів: УАД – ДНДІІІ, 2005. – 140 с.

41. Rumelhart D.E., Hinton G.E., Williams R.J. Learning Internal Representations by Error Propagation // Parallel Distributed Processing. 1986. Vol. 1. P. 318–362.

42. Галушкин А. И. Синтез многослойных систем распознавания образов. М.: «Энергия», 1974.

43. Werbos P. J., Beyond regression: New tools for prediction and analysis in the behavioral sciences. Ph.D. thesis, Harvard University, Cambridge, MA, 1974.

44. Методи, алгоритми і програмні засоби опрацювання біомедичних зображень / Березький О. М., Батько Ю.М., Березька К.М. і ін. Тернопіль: Економічна думка, ТНЕУ, 2017. 330 с.

45. Коршунов Ю. М. Математические основы кибернетики: учеб. пособие для вузов. – 3-изд., перераб. и доп. – М.: Энергоатомиздат, 1987. – 496 с.

46. Mei S., Montanari A., Nguyen P.-M. A mean field view of the landscape of two-layers neural networks. Proceedings of the National Academy of Sciences, pp. E7665– E7671, 2018.

47. Нильсон Н. Искусственный интеллект. Методы поиска решений. М.: Мир, 1973. 272 с.

48. Неміш В.М., Березька К.М. Алгоритми синтезу структур нейронних мереж: III Наук.-практ. конф. молодих вчених і студентів «Інтелектуальні комп’ютерні системи та мережі». 26 листопада 2020 р. Тернопіль, 2020. С. 48.

49. Гліненко Л. К., Смердов А. А. Технологія інженерного проектування: структурний синтез технічних та біотехнічних систем: Навч. посібник. Львів: Видавництво Національного університету «Львівська політехніка», 2004. 388 с.

50. Вітлінський В.В. Економіко-математичні методи та моделі: оптимізація: навч. посібник [Електронний ресурс] / Вітлінський В. В., Терещенко Т. О., Савіна С. С. К. : КНЕУ, 2016. 303 с.

51. Теслюк В.М. Математичне моделювання в САПР: Ч.1. Конспект лекцій з курсу “Математичне моделювання в САПР” для студентів базового напрямку “Комп’ютерні науки”. – Львів: Видавництво Національного університету “Львівська політехніка”, 2009. – 64 с.

52. Березький О.М., Лящинський П.М., Лящинський П.М., Сухович А.Р., Долинюк Т.М. Синтез біомедичних зображень на основі генеративно-змагальних мереж. Український журнал інформаційних технологій. 2019, т. 1, № 1. С. 35-40.

53. Синєглазов В., Чумаченко О. Глибокі нейронні мережі для вирішення завдань розпізнавання і класифікації зображення [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: http://itcm.comp-sc.if.ua/2017/Sineglazov.pdf/

54. Синєглазов В.М. Вдосконалення гібридного генетичного алгоритму для синтезу глибоких нейронних мереж / В. М. Синєглазов, О. І. Чумаченко, Д. Коваль // IV Міжнародна науково-практична конференція «Обчислювальний інтелект» (Київ, 16-18 травня, 2017). – С. 142 – 143.

55. Скобцов Ю. А. Основи еволюційних обчислень: навч. посібник. Донецьк: ДонНТУ, 2008. 326 с.

56. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы: Пер. с польск. И. Д. Рудинского. М.: Горячая линия – Телеком, 2004. 452 с.

57. Schizas C. N., Pattichis C. S., Middleton L. T. Neural networks, genetic algorithms and k-means algorithm: In search of data classification, in: Proceedings of International Workshop on Combinations of Genetic Algorithms and Neural Networks, COGANN-92, 1992.

58. Kadaba N. Nygard K. E., Improving the performance of genetic algorithms in automated discovery of parameters, in: Porter B. W, Mooney R. J. (ed.), Proceedings of the Seventh International Conference of Machine Learning, San Mateo, CA: Morgan Kauffmann, 1990, pp. 140-148.

59. Suzuki K., Kakazu Y. An approach to the analysis of the basins of the associative memory model using genetic algorithms, in: Belew R. K., Booker L. B (ed.), Fourth International Conference on Genetic Algorithms, San Mateo, CA: Morgan Kaufmann, 1991, pp. 539-546.

60. Guo Z. Uhrig R. E. Use of genetic algorithms to select inputs for neural networks in: Proceedings of International Workshop on Combinations of Genetic Algorithms and Neural Networks, COGANN-92, 1992, pp. 223-224.

61. Schaffer J. D., Caruana R. A, Eshelma'n L. J. Using genetic search to exploit the emergent behavior of neural networks, in: Forrest S. (ed.) Emergent Computation, Amsterdam: North Holland, 1990, pp. 244-248.

62. Whitley D. Applying genetic algorithms to neural network learning, Proceedings of the Seventh Conference of the Society of Artificial Intelligence and Simulation of Behavior, Sussex, England, Pitman Publishing, 1989, pp. 137-144.

63. Ackley D. H. A connectionist machine for genetic hillclimbing, Boston, MA: KluwerAcademic Publishers, 1987.

64. Gonzalez-Seco J. A genetic algorithm as the learning procedure for neural networks, IEEE International Joint Conference on Neural Networks, Baltimore, MD, IEEE, 1992, pp. 835-840.

65. Shonkwiler R., Miller K. R. Genetic algorithm, neural network synergy for nonlinearly constrained optimization problems, Proceedings of International Workshop on Combinations of Genetic Algorithms and Neural Networks, COGANN-92, 1992, pp. 248-257.

66. Gpu кластер: веб-сайт. URL: https://uk.wikipedia.org/wiki/Gpu_кластер.

67. 10 лучших языков программирования для машинного обучения – GitHub: веб-сайт. URL: http://www.dialog-e.ru/market-news/674/.

68. Березький О. М. Інформаційно-аналітична система для дослідження і діагностування пухлинних (ракових) клітин людини на основі аналізу їх зображень / О. М. Березький, Т.В. Дацко, Ю.М Батько // Каталог матеріалів Міжнародного Форуму «Регіони знань: Україна в європейському просторі освіти – науки - інновацій для ревіталізації та процвітання територій», 26-27 березня 2010, м. Тернопіль. – Тернопіль, 2010. – С. 37.

69. Березький О. М. Нечітка база знань інтелектуальної системи діагностування видів раку молочної залози / О. М. Березький, Г. М. Мельник, К. М. Березька // Вісник Хмельницького національного університету. Технічні науки. – 2013. – №6. – С.284-291.

70. Березький О. М. Інтелектуальна система для діагностування різних форм раку молочної залози на основі аналізу гістологічних і цитологічних зображень / О. М. Березький, Г.М.Мельник, Ю.М. Батько, Т. В. Дацко // Науковий вісник НЛТУ України: зб. наук.-техн. праць. – Львів: РВВ НЛТУ України. – 2013. – Вип. 23.13. – С. 357-367.

71. Березький О. М. Інформаційно-аналітична система дослідження та діагностування пухлинних клітин на основі аналізу їх зображень / О. М. Березький, Ю.М. Батько, Г.М.Мельник // Вісник Хмельницького національного університету. Технічні науки. – 2008. – №4. – С.33-41.

72. Березький О. М. Синтез альтернативних рішень при структурному проектуванні систем автоматизованої мікроскопії / О. М. Березький, Ю. М. Батько, Г. М. Мельник // Науковий вісник НЛТУ України: зб. наук.-техн. праць. – Львів: РВВ НЛТУ України. – 2009. – Вип. 19.5. – С. 258-268.

73. Berezsky O. Vision-based medical expert system / Berezsky O., Berezska K., BatkoYu., Melnyk G. // 6th International Scientific and Technical Conference «Computer Sciences and Information Technologies» (CSIT'2011), Lviv, November 16-19, 2011. – Lviv, 2011. – P. 288-289.