МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний
університет Факультет комп'ютерних
інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

МАГИЛЬНИЦЬКИЙ Дмитро Богданович

Програмний модуль побудови оптимальної архітектури нейронних мереж прямого поширення / Program Module For Designing Of An Optimal Feedforward Neural Networks Architecture

спеціальність: 122 – Комп'ютерні науки
освітньо-професійна програма –
Штучний інтелект

Кваліфікаційна робота

Виконав студент групи
КНШІ-41 Д. Б.
Магильницький

_____

Науковий керівник:
к. т. н., доцент, В. С. Коваль

_____

Кваліфікаційну
роботу допущено
до захисту

___ _____20__р.

В.о. завідувач

кафедри
_____Н. М. Васильків

Тернопіль - 2025

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність 122 – Комп'ютерні науки
освітньо-професійна програма – Штучний інтелект

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Н.М. Васильків
«____» _____2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Магильницький Дмитро Богданович
(прізвище, ім'я, по батькові)

Тема кваліфікаційної роботи: Програмний модуль побудови оптимальної архітектури нейронних мереж прямого поширення/ Program Module For Designing Of An Optimal Feedforward Neural Networks Architecture

1. Керівник роботи к.т.н, доцент Коваль В.С.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання) затверджені наказом по університету від 20 грудня 2024 р. № 938.
2. Строк подання студентом закінченої кваліфікаційної роботи 25травня 2025 р.
3. Вихідні дані до кваліфікаційної роботи: наукові статті, технічна література.
4. Основні питання, які потрібно розробити:
− проаналізувати сучасні архітектури нейронних мереж та існуючих рішень побудови оптимальних архітектур нейронних мереж прямого поширення;
− визначити математичну модель побудови оптимальної архітектури нейронних мереж прямого поширення;
− розкрити алгоритмічне забезпечення запропонованого методу побудови архітектури нейронних мереж прямого поширення;
− обгрунтувати програмні засоби дослідження запропонованого алгоритму побудови оптимальних архітектур нейронних мереж та модель взаємозв'язку програмних модулів;
− реалізувати програмний модуль, що інтегрується з популярними фреймворками машинного навчання (TensorFlow, PyTorch), забезпечує можливість паралельного обчислення та надає зручний інтерфейс для взаємодії з користувачем;
− провести експериментальне дослідження розробленого програмного модуля на реальних наборах даних, порівняти його ефективність з існуючими рішеннями та оцінити його практичну цінність для вирішення прикладних задач.
5. Перелік графічного матеріалу в роботі:
− схема алгоритму побудови оптимальної архітектури нейронних мереж прямого поширення;

− діаграма випадків системи побудови оптимальних архітектур нейронних мереж;
− діаграма класів системи побудови оптимальних архітектур нейронних мереж;
− діаграма активностей процесу пошуку оптимальної архітектури;
− точність моделей у залежності від складності;
− порівняння конфігурацій нейронних мереж.

6.Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання 20 грудня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи	до 01.01. 2025 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.02. 2025 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 01.04.2025 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05. 2025 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2025 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів	до 25.05.2025 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту	до 30.05.2025 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 10.06.2025 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії	до 10.06. 2025 р.	

Студент _____ Д.Б. Магильницький

Керівник кваліфікаційної роботи _____ В.С. Коваль

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи: 68 с., 7 рис., 4 табл., 2 додатки, 43 джерела.

Кваліфікаційна робота присвячена розробці програмного модуля побудови оптимальних архітектур нейронних мереж прямого поширення.

Об’єктом дослідження є процес проектування та оптимізації архітектур нейронних мереж прямого поширення для вирішення задач класифікації та прогнозування.

Метою роботи є розробка програмного модуля для автоматизованого конструювання оптимальних архітектур нейронних мереж прямого поширення.

Методи дослідження, які були використані в роботі: аналіз і синтез, математичне моделювання, обчислювальний експеримент, порівняльний аналіз, методи статистичного аналізу. Результатом роботи є модуль побудови оптимальних архітектур нейронних мереж прямого поширення базується на комбінації баєсової оптимізації та генетичних алгоритмів з адаптивною стратегією пошуку.

Практична цінність розробленого програмного модуля підтверджується можливістю його застосування в різних галузях, включаючи фінансове прогнозування, медичну діагностику, системи рекомендацій та інші області, де використовуються нейронні мережі.

Ключові слова: НЕЙРОННІ МЕРЕЖІ ПРЯМОГО ПОШИРЕННЯ, ОПТИМІЗАЦІЯ АРХІТЕКТУРИ, БАЄСОВА ОПТИМІЗАЦІЯ, ГЕНЕТИЧНІ АЛГОРИТМИ, АВТОМАТИЗОВАНЕ КОНСТРУЮВАННЯ, МАШИННЕ НАВЧАННЯ.

ANNOTATION

The bachelor’s thesis report: 68 pages, 7 figures, 4 tables, 2 appendices, 43 sources.

The qualification work is devoted to the development of a software module for constructing optimal architectures of feed-forward neural networks.

The object of the research is the process of designing and optimizing architectures of feed-forward neural networks for solving classification and forecasting problems.

The purpose of the work is to develop a software module for the automated construction of optimal architectures of feed-forward neural networks.

The research methods used in the work: analysis and synthesis, mathematical modeling, computational experiment, comparative analysis, statistical analysis methods. The result of the work is a module for constructing optimal architectures of feed-forward neural networks based on a combination of Bayesian optimization and genetic algorithms with an adaptive search strategy.

The practical value of the developed software module is confirmed by the possibility of its application in various industries, including financial forecasting, medical diagnostics, recommendation systems and other areas where neural networks are used.

Keywords: FEEDFORWARD NEURAL NETWORKS, ARCHITECTURE OPTIMIZATION, BAYESIAN OPTIMIZATION, GENETIC ALGORITHMS, AUTOMATED CONSTRUCTION, MACHINE LEARNING.

ЗМІСТ

ВСТУП

Сучасний етап розвитку інформаційних технологій характеризується стрімким поширенням застосування методів штучного інтелекту, зокрема нейронних мереж, у різноманітних сферах людської діяльності. Нейронні мережі прямого поширення стали потужним інструментом для вирішення широкого спектру задач класифікації, регресії, розпізнавання образів та прогнозування. Проте ефективність застосування нейронних мереж значною мірою залежить від оптимальності їх архітектури, яка визначає обчислювальну складність, точність прогнозування та здатність до узагальнення.

Процес проектування архітектури нейронної мережі традиційно вимагає значних експертних знань та часових ресурсів. Розробники нейронних мереж часто покладаються на евристичні методи та власний досвід при визначенні кількості шарів, нейронів, типів активаційних функцій та інших параметрів, що впливають на продуктивність мережі. Такий підхід не гарантує знаходження оптимальної архітектури та призводить до неефективного використання обчислювальних ресурсів. Автоматизація процесу конструювання архітектури нейронних мереж є актуальним науково-практичним завданням, що дозволить підвищити ефективність та доступність технологій машинного навчання.

Актуальність теми дослідження зумовлена зростаючою складністю задач, які вирішуються за допомогою нейронних мереж, та обмеженістю обчислювальних ресурсів для їх навчання і застосування. Сучасні методи автоматизованого конструювання архітектур нейронних мереж, такі як пошук архітектури нейронних мереж, еволюційні алгоритми та методи градієнтного спуску, демонструють перспективні результати, проте мають суттєві обмеження. Такий процес вимагає значних обчислювальних ресурсів, еволюційні алгоритми можуть повільно збігатися до оптимального рішення, а градієнтні методи схильні до знаходження в локальних мінімумах.

Особливо гострою є потреба в автоматизованих методах побудови оптимальних архітектур для нейронних мереж, які застосовуються у системах з

обмеженими обчислювальними ресурсами, таких як мобільні пристрої, вбудовані системи та системи реального часу. Оптимізація архітектури дозволяє знаходити компроміс між точністю та обчислювальною ефективністю, що є критичним для практичного впровадження нейронних мереж у таких системах.

Метою роботи є розроблення програмних модулів для автоматизованого конструювання оптимальних архітектур нейронних мереж прямого поширення, який забезпечує підвищення ефективності процесу проектування та навчання нейронних мереж для вирішення практичних задач прогнозування та класифікації.

Основними завдання дослідження є:

– проаналізувати сучасні архітектури нейронних мереж та існуючих рішень побудови оптимальних архітектур нейронних мереж прямого поширення;

– визначити математичну модель побудови оптимальної архітектури нейронних мереж прямого поширення;

– розкрити алгоритмічне забезпечення запропонованого методу побудови архітектури нейронних мереж прямого поширення;

– обгрунтувати програмні засоби дослідження запропонованого алгоритму побудови оптимальних архітектур нейронних мереж та модель взаємозв’язку програмних модулів;

– реалізувати програмний модуль, що інтегрується з популярними фреймворками машинного навчання (TensorFlow, PyTorch), забезпечує можливість паралельного обчислення та надає зручний інтерфейс для взаємодії з користувачем;

− провести експериментальне дослідження розробленого програмного модуля на реальних наборах даних, порівняти його ефективність з існуючими рішеннями та оцінити його практичну цінність для вирішення прикладних задач.

Об'єктом дослідження є процес проектування та оптимізації архітектур нейронних мереж прямого поширення для вирішення задач класифікації та прогнозування.

Предметом дослідження є методи та алгоритми автоматизованого конструювання оптимальних архітектур нейронних мереж прямого поширення.

Методи дослідження, які були використані в роботі, являється комплекс

загальнонаукових та спеціальних методів дослідження, а саме: аналіз і синтез (для вивчення існуючих підходів до побудови архітектур нейронних мереж), математичне моделювання (для формалізації процесу оптимізації архітектури), обчислювальний експеримент (для оцінки ефективності розробленого методу), порівняльний аналіз (для зіставлення результатів запропонованого методу з існуючими рішеннями), методи статистичного аналізу (для обробки результатів експериментів).

Практичне значення отриманих результатів визначається можливістю застосування розроблених програмних модулів для автоматизованого проектування ефективних нейронних мереж у різних галузях, включаючи фінансове прогнозування, медичну діагностику, обробку природної мови та комп'ютерний зір. Програмні модулі дозволяє спеціалістам з аналізу даних та машинного навчання створювати оптимальні за співвідношенням точності та обчислювальної складності нейронні мережі без необхідності глибоких знань особливостей їх архітектури. Це значно скорочує час розробки моделей машинного навчання та робить технології нейронних мереж доступнішими для ширшого кола фахівців.

1 АНАЛІЗ ВІДОМИХ РІШЕНЬ

1.1 Аналіз сучасних архітектур нейронних мереж

Нейронні мережі представляють собою обчислювальні системи, архітектура яких ґрунтується на принципах функціонування біологічних нейронних мереж, зокрема мозку людини. За визначенням, нейронна мережа – це математична модель, а також її програмна та апаратна реалізація, побудована за принципом функціонування та організації біологічних нейронних мереж [5]. Штучні нейронні мережі є універсальними апроксиматорами функцій, що дозволяє використовувати їх для вирішення широкого спектру завдань, таких як класифікація, регресія, кластеризація та інші.

Архітектура нейронної мережі визначає спосіб організації та з'єднання штучних нейронів у єдину обчислювальну структуру. Нейронні мережі прямого поширення (feed-forward neural networks) характеризуються односпрямованим рухом сигналу від вхідного шару через приховані шари до вихідного, без зворотних зв'язків або циклів. Ця архітектура є фундаментальною в області глибокого навчання і знаходить застосування у різноманітних сферах, від розпізнавання образів до обробки природної мови.

Багатошаровий перцептрон (Multilayer Perceptron, MLP) є класичною архітектурою нейронної мережі прямого поширення, що складається з вхідного шару, одного або більше прихованих шарів та вихідного шару. Кожен нейрон з'єднаний з усіма нейронами попереднього шару, реалізуючи повнозв'язну архітектуру [14]. Навчання MLP відбувається за допомогою алгоритму зворотного поширення помилки (backpropagation), що дозволяє коригувати ваги зв'язків між нейронами для мінімізації функції втрат. Ця архітектура демонструє гарні результати для задач класифікації та регресії, проте має обмеження при обробці послідовних даних та зображень.

Згорткові нейронні мережі (Convolutional Neural Networks, CNN) розроблені спеціально для обробки даних з сітковою топологією, таких як зображення. CNN використовують операцію згортки, що дозволяє враховувати просторову структуру даних. Архітектура CNN зазвичай включає чергування згорткових шарів, шарів

об'єднання (pooling) та повнозв'язних шарів. Згорткові шари виконують операцію згортки, виділяючи локальні ознаки з вхідних даних, шари об'єднання зменшують просторові розміри, а повнозв'язні шари формують остаточне рішення.

Рекурентні нейронні мережі (Recurrent Neural Networks, RNN) створені для обробки послідовних даних, таких як текст або часові ряди. На відміну від мереж прямого поширення, RNN містять зворотні зв'язки, що дозволяють зберігати інформацію про попередні входи. Варіанти RNN, такі як LSTM (Long Short-Term Memory) та GRU (Gated Recurrent Unit), вирішують проблему затухаючих градієнтів, характерну для базових RNN, і здатні запам'ятовувати довгострокові залежності в даних.

Автоенкодери (Autoencoders) – це архітектура, що навчається відтворювати вхідні дані на виході, проходячи через вузьке місце – кодувальний простір з меншою розмірністю [12]. Архітектура автоенкодера складається з кодувальника, що стискає вхідні дані до прихованого представлення, та декодувальника, що відновлює вхідні дані з прихованого представлення. Автоенкодери використовуються для зниження розмірності даних, виявлення аномалій та генеративного моделювання.

Генеративні змагальні мережі (Generative Adversarial Networks, GAN) – це архітектура, що включає дві конкуруючі нейронні мережі: генератор, що створює дані, та дискримінатор, що розрізняє реальні та згенеровані дані. У процесі навчання генератор намагається створювати дані, які дискримінатор не зможе відрізнити від реальних, а дискримінатор навчається все краще розрізняти реальні та згенеровані дані. GAN знаходять застосування у генерації зображень, перенесенні стилю, доповненні даних та інших завданнях.

Нейронні мережі з механізмом уваги (Attention-based Neural Networks) розроблені для фокусування на найбільш релевантних частинах вхідних даних при формуванні вихідного результату. Механізм уваги дозволяє моделі надавати різну вагу різним частинам вхідних даних, що особливо корисно при обробці довгих послідовностей. Архітектури, засновані на механізмі уваги, такі як Transformer, досягли значних успіхів у завданнях обробки природної мови та комп'ютерного зору.

Графові нейронні мережі (Graph Neural Networks, GNN) розроблені для обробки даних з графовою структурою, де інформація представлена у вигляді вузлів та ребер [23]. GNN узагальнюють операцію згортки на графи, дозволяючи враховувати не тільки ознаки вузлів, але й структуру зв'язків між ними. Ці архітектури застосовуються у соціальних мережах, хемоінформатиці, рекомендаційних системах та інших областях, де дані мають графову структуру.

Для вирішення складних задач часто використовуються гібридні архітектури, що поєднують різні типи нейронних мереж. Наприклад, CNN-LSTM архітектури використовуються для обробки відеопослідовностей, де CNN виділяє просторові ознаки з кадрів, а LSTM обробляє часові залежності між ними. Інший приклад – нейронні мережі з залишковими зв'язками (ResNet), що вирішують проблему затухаючих градієнтів у глибоких мережах шляхом додавання прямих з'єднань між шарами.

Архітектура нейронної мережі має визначальний вплив на її здатність до навчання та узагальнення, а також на обчислювальну ефективність. Вибір оптимальної архітектури для конкретного завдання залежить від багатьох факторів, включаючи характер даних, складність завдання, доступні обчислювальні ресурси та інші обмеження. Розробка підходів до автоматизованого конструювання оптимальних архітектур є активною областю досліджень у машинному навчанні.

Стандартні нейронні мережі прямого поширення являють собою першу сходинку в еволюції архітектур нейронних мереж. Ці мережі будуються з почергово розташованих шарів нейронів, у яких кожен нейрон поточного шару отримує вхідний сигнал від усіх нейронів попереднього шару, виконує нелінійне перетворення та передає результат на наступний шар [9]. Активаційні функції, такі як сигмоїдальна, тангенціальна гіперболічна або ReLU, вносять нелінійність, що дозволяє моделювати складні залежності в даних.

Глибокі нейронні мережі (Deep Neural Networks, DNN) є розширенням класичних нейронних мереж з більшою кількістю прихованих шарів. Збільшення глибини мережі дозволяє їй вивчати ієрархічні представлення даних на різних рівнях абстракції. Однак, навчання глибоких мереж стикається з певними викликами, такими як проблема затухаючих градієнтів, перенавчання та

необхідність у великих обсягах даних для тренування.

Радіальні базисні нейронні мережі (Radial Basis Function Networks, RBFN) використовують радіальні базисні функції як функції активації. RBFN зазвичай мають три шари: вхідний, прихований з радіальними базисними нейронами та вихідний. Ці мережі особливо ефективні для інтерполяції в багатовимірному просторі та задач класифікації з чіткими межами між класами. RBFN навчаються швидше порівняно з багатошаровими перцептронами, але можуть потребувати більшої кількості нейронів для досягнення аналогічної точності.

Модульні нейронні мережі (Modular Neural Networks) складаються з незалежних нейронних мереж, що працюють разом для вирішення комплексних задач. Кожна підмережа спеціалізується на конкретному аспекті загальної задачі, а їхні виходи комбінуються для формування остаточного результату. Такий підхід дозволяє розбивати складні задачі на простіші підзадачі, спрощуючи процес навчання та підвищуючи загальну продуктивність системи.

1.2 Аналіз існуючих рішень побудови оптимальних архітектур нейронних мереж прямого поширення

Основна задача побудови оптимальної архітектури нейронної мережі належить до фундаментальних завдань у галузі машинного навчання. Існуючі підходи до автоматизованого проектування штучних нейронних мереж можна класифікувати за рівнем їхньої автоматизації та використовуваними методологіями. Ці методи варіюються від ручного підбору гіперпараметрів до повністю автоматизованих систем, що самостійно визначають оптимальну структуру мережі [13]. У контексті нейронних мереж прямого поширення оптимальна архітектура визначається як така, що забезпечує найкращу точність при мінімальній обчислювальній складності.

Метод сіткового пошуку (Grid Search) є одним із найпростіших підходів до підбору оптимальної архітектури нейронної мережі. Цей метод передбачає систематичне перебирання всіх можливих комбінацій гіперпараметрів з

попередньо визначеного набору значень. Для нейронних мереж прямого поширення такими параметрами є кількість прихованих шарів, кількість нейронів у кожному шарі, функції активації, методи ініціалізації ваг та інші. Попри простоту реалізації, сітковий пошук стає обчислювально неефективним при збільшенні кількості гіперпараметрів через експоненційне зростання простору пошуку.

Випадковий пошук (Random Search) є альтернативою сітковому пошуку, що передбачає випадкове випробування комбінацій гіперпараметрів із заданих розподілів [37]. Дослідження показують, що випадковий пошук часто виявляється ефективнішим за сітковий, особливо у просторах з високою розмірністю, де не всі гіперпараметри мають однаковий вплив на кінцевий результат. Випадковий пошук дозволяє ефективніше досліджувати простір гіперпараметрів, проте залишається обчислювально затратним для великих нейронних мереж.

Баєсова оптимізація (Bayesian Optimization) є більш просунутим методом підбору гіперпараметрів, що використовує ймовірнісну модель для прогнозування продуктивності мережі з різними наборами гіперпараметрів. Цей підхід використовується для довільних функцій та є ресурснозатратним. На кожній ітерації обирається набір гіперпараметрів, що максимізує очікуване покращення, після чого модель оновлюється з урахуванням нових спостережень. Байєсова оптимізація дозволяє значно скоротити кількість оцінюваних комбінацій гіперпараметрів порівняно з сітковим та випадковим пошуком.

Генетичні алгоритми (Evolutionary Algorithms) являють собою біологічно-натхнений підхід до оптимізації архітектури нейронних мереж [26]. Генетичні алгоритми, один із видів еволюційних алгоритмів, оперують популяцією потенційних рішень (архітектур), які еволюціонують через процеси селекції, схрещування та мутації. Кожна архітектура кодується як «хромосома», що містить інформацію про структуру мережі, а функція пристосованості оцінює продуктивність кожної архітектури. З кожним поколінням популяція поступово покращується, наближаючись до оптимального рішення. Генетичні алгоритми здатні досліджувати великі простори пошуку та знаходити непередбачувані, але ефективні архітектури.

Нейроеволюція (Neuroevolution) є спеціалізованим підходом до застосування

еволюційних алгоритмів для оптимізації нейронних мереж. Одним із відомих методів є NEAT (NeuroEvolution of Augmenting Topologies), який одночасно оптимізує ваги та структуру мережі. NEAT починає з мінімальних мереж і поступово ускладнює їх, додаючи нові нейрони та зв'язки. Метод використовує спеціальний механізм схрещування для комбінування мереж різної структури та техніку спеціалізації для збереження різноманітності в популяції. NEAT та його розширення, такі як HyperNEAT, демонструють гарні результати в завданнях навчання з підкріпленням та еволюційної роботики.

Нейронна архітектурна оптимізація (Neural Architecture Search, NAS) є сучасною парадигмою автоматизованого проектування нейронних мереж [17]. NAS формулює пошук архітектури як задачу навчання з підкріпленням, де агент (контролер) генерує архітектури, а система оцінює їхню продуктивність на цільовому завданні. Контролер, зазвичай реалізований як рекурентна нейронна мережа, навчається генерувати все кращі архітектури на основі отриманої винагороди. Класичний NAS є обчислювально затратним, вимагаючи тренування тисяч архітектур, але існують більш ефективні варіації, такі як ENAS (Efficient Neural Architecture Search) та DARTS (Differentiable Architecture Search).

DARTS (Differentiable Architecture Search) пропонує елегантне рішення проблеми обчислювальної складності NAS шляхом формулювання пошуку архітектури як задачі диференційованої оптимізації. Замість дискретного вибору операцій, DARTS присвоює кожній операції вагу і оптимізує ці ваги за допомогою градієнтного спуску. Після завершення оптимізації для кожного з'єднання обирається операція з найбільшою вагою. Такий підхід дозволяє одночасно оптимізувати архітектуру та ваги мережі, значно скорочуючи обчислювальні витрати. DARTS та його модифікації успішно застосовуються для оптимізації архітектур мереж прямого поширення для різних завдань, включаючи класифікацію зображень та обробку природної мови.

Автоматичне машинне навчання (AutoML) представляє інтегрований підхід до автоматизації всього процесу машинного навчання, включаючи вибір моделі, оптимізацію гіперпараметрів та інженерію ознак. Платформи AutoML, такі як Google AutoML, H2O AutoML та Auto-sklearn, надають інструменти для

автоматичного конструювання та оптимізації нейронних мереж без глибоких знань у галузі машинного навчання. Ці системи зазвичай комбінують різні методи оптимізації, такі як баєсова оптимізація, методи вибіркового пошуку та ансамблювання, для досягнення найкращих результатів. AutoML спрощує процес розробки моделей машинного навчання, роблячи їх доступними для шир

Методи побудови нейронних мереж з конструктивними та деструктивними елементами пропонують альтернативний підхід до оптимізації архітектури. Конструктивні алгоритми починають з мінімальної мережі і поступово додають нейрони або шари, доки не досягається задовільна продуктивність. Навпаки, деструктивні (прунінгові) алгоритми починають з великої мережі і поступово видаляють найменш значущі компоненти. Cascade-Correlation, наприклад, є конструктивним алгоритмом, що послідовно додає нейрони до мережі, максимізуючи кореляцію між їхнім виходом і залишковою помилкою мережі. Ці методи особливо корисні для автоматичного визначення кількості нейронів у прихованих шарах мереж прямого поширення.

Методи редукції ваг (Weight Pruning) та квантизації спрямовані на оптимізацію вже навчених нейронних мереж шляхом видалення надлишкових параметрів без суттєвої втрати продуктивності. Прунінг ваг виявляє та видаляє найменш значущі зв'язки в мережі, перетворюючи щільну мережу на розріджену. Квантизація зменшує прецизійність представлення ваг, наприклад, переходячи від 32-бітних чисел з плаваючою комою до 8-бітних цілих чисел. Ці методи можуть значно зменшити розмір моделі та прискорити інференс, особливо на мобільних та вбудованих пристроях. Хоча ці підходи не створюють нові архітектури, вони оптимізують існуючі для конкретних обчислювальних обмежень.

Мультифідельна оптимізація (Multi-fidelity Optimization) використовує наближені оцінки продуктивності моделі для прискорення пошуку оптимальної архітектури [28]. Замість повного тренування кожної кандидатської архітектури, яке може бути дуже затратним, мультифідельна оптимізація використовує дешевші (низькофідельні) апроксимації, такі як тренування на підмножині даних, з меншою кількістю епох або з нижчою роздільною здатністю входів. Найбільш перспективні архітектури, виявлені з допомогою низькофідельних оцінок, потім оцінюються з

високою точністю. Цей підхід дозволяє ефективно досліджувати великі простори архітектур з обмеженими обчислювальними ресурсами, зберігаючи при цьому якість кінцевої моделі.

Методи переносу навчання (Transfer Learning) та адаптації доменів можуть бути застосовані до оптимізації архітектури нейронних мереж. Ці підходи використовують знання, отримані при оптимізації архітектури для одного завдання або домену, для прискорення пошуку оптимальної архітектури для іншого, пов'язаного завдання. Наприклад, NAS-Bench-101 та подібні репозиторії містять заздалегідь оцінені архітектури для еталонних завдань, що можуть бути використані як відправна точка для нових задач. Переносне навчання в NAS може значно скоротити час пошуку архітектури, особливо коли цільове завдання має обмежену кількість навчальних даних.

Методи одноетапного навчання (One-Shot Learning) та спільного навчання ваг (Weight Sharing) прагнуть зменшити обчислювальну складність NAS шляхом повторного використання параметрів між різними архітектурами [22]. Замість тренування кожної архітектури з нуля, ці методи тренують "надмережу" (supernet), що включає всі можливі архітектури як підмережі. Після тренування надмережі окремі архітектури можуть бути оцінені шляхом спадкування ваг від надмережі, що усуває необхідність в окремому тренуванні. ENAS, DARTS та інші методи використовують цей підхід для значного скорочення обчислювальних вимог. Однак якість моделі, отриманої шляхом спільного навчання ваг, може бути нижчою порівняно з повністю натренованою архітектурою.

Градієнтно-базовані методи оптимізації архітектури (Gradient-based Architecture Optimization) формулюють проблему вибору архітектури як диференційовну задачу оптимізації. Ці методи параметризують простір архітектур і використовують градієнтний спуск для оптимізації як архітектурних параметрів, так і ваг мережі. DARTS, NAO (Neural Architecture Optimization) та інші подібні підходи перетворюють дискретний пошук архітектури в неперервну оптимізацію, що дозволяє ефективно знаходити оптимальні структури мереж прямого поширення. Та зазвичай вони швидші за ті, що базуються на навчанні з підкріпленням, але можуть бути менш стійкими до локальних оптимумів.

1.3 Постановка задачі дослідження

Проблема ефективної побудови оптимальних архітектур нейронних мереж прямого поширення є однією з головних у галузі машинного навчання. Нейронні мережі прямого поширення (feedforward neural networks) – це клас моделей машинного навчання, в яких інформація передається від вхідних вузлів через приховані шари до вихідних вузлів без зворотних зв'язків або циклів [6]. Архітектура таких мереж характеризується кількістю шарів, кількістю нейронів у кожному шарі, функціями активації, способами ініціалізації ваг та іншими параметрами. Вибір оптимальної архітектури суттєво впливає на продуктивність моделі, її здатність до узагальнення та обчислювальну ефективність.

Задача автоматизованого конструювання оптимальних архітектур нейронних мереж прямого поширення формулюється як пошук такої конфігурації мережі, яка максимізує певний критерій ефективності при мінімальних обчислювальних витратах. Критеріями ефективності можуть бути точність на валідаційному наборі даних, узагальнювальна здатність моделі, час інференсу, кількість параметрів або комбінація цих факторів.

Аналіз існуючих рішень, показав, що сучасні підходи до автоматизованого конструювання архітектур, такі як нейронна архітектурна оптимізація (NAS), еволюційні алгоритми та градієнтно-базовані методи, хоча і демонструють гарні результати, але мають ряд обмежень. NAS вимагає значних обчислювальних ресурсів, еволюційні алгоритми можуть повільно збігатися до оптимального рішення, а градієнтно-базовані методи схильні до застрягання в локальних оптимумах [34]. Тому розробка ефективного програмного модуля для побудови оптимальних архітектур нейронних мереж прямого поширення, який би поєднував переваги різних підходів та мінімізував їхні недоліки, є актуальною науково-практичною задачею.

Метою даного дослідження є розробка програмного модуля для автоматизованого конструювання оптимальних архітектур нейронних мереж прямого поширення, який забезпечує підвищення ефективності процесу проектування та навчання нейронних мереж для вирішення практичних задач

прогнозування та класифікації. Для досягнення цієї мети необхідно вирішити наступні завдання:

– проаналізувати сучасні архітектури нейронних мереж та існуючих рішень побудови оптимальних архітектур нейронних мереж прямого поширення;

– визначити математичну модель побудови оптимальної архітектури нейронних мереж прямого поширення;

– розкрити алгоритмічне забезпечення запропонованого методу побудови архітектури нейронних мереж прямого поширення;

– обгрунтувати програмні засоби дослідження запропонованого алгоритму побудови оптимальних архітектур нейронних мереж та модель взаємозв'язку програмних модулів;

– реалізувати програмний модуль, що інтегрується з популярними фреймворками машинного навчання (TensorFlow, PyTorch), забезпечує можливість паралельного обчислення та надає зручний інтерфейс для взаємодії з користувачем;

− провести експериментальне дослідження розробленого програмного модуля на реальних наборах даних, порівняти його ефективність з існуючими рішеннями та оцінити його практичну цінність для вирішення прикладних задач.

2 АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Математична модель побудови оптимальної архітектури нейронних мереж прямого поширення

Математична модель побудови оптимальної архітектури нейронних мереж прямого поширення є багатопараметричною задачею оптимізації. Нейронна мережа прямого поширення визначається як орієнтований ациклічний граф, в якому вхідні дані послідовно проходять через серію прихованих шарів без зворотних зв'язків. Формально нейронну мережу можна представити як відображення $F: X \rightarrow Y$, де X - простір вхідних даних, Y - простір вихідних даних. Архітектура мережі описується параметрами структури (кількість шарів, кількість нейронів у кожному шарі, типи з'єднань) та параметрами налаштування (функції активації, методи ініціалізації ваг, регуляризація).

Для формалізації задачі побудови оптимальної архітектури введемо поняття простору архітектур A, де кожна архітектура $a \in A$ характеризується набором параметрів $\theta(a) = \{\theta_1, \theta_2, ..., \theta_n\}$, де θ_i - параметри і-го шару мережі. Кожен параметр θ_i включає кількість нейронів n_i, тип активаційної функції f_i, метод регуляризації r_i та інші характеристики шару. Простір архітектур A є дискретним та неопуклим, що ускладнює застосування класичних методів оптимізації.

Для оцінки якості архітектури вводиться функція ефективності $E: A \rightarrow R$, яка кількісно характеризує продуктивність мережі з архітектурою a на заданому наборі даних D. Функція ефективності може враховувати різні аспекти продуктивності, включаючи точність прогнозування, обчислювальну складність, час навчання та інференсу, а також узагальнювальну здатність моделі. Формально ефективність можна представити як $E(a, D) = \alpha_1 \cdot Acc(a, D) - \alpha_2 \cdot Comp(a) - \alpha_3 \cdot Time(a, D)$, де $Acc(a, D)$ - точність мережі з архітектурою a на наборі даних D, $Comp(a)$ - обчислювальна складність мережі, $Time(a, D)$ - час, необхідний для навчання та інференсу, $\alpha_1, \alpha_2, \alpha_3$ - вагові коефіцієнти для балансування різних аспектів ефективності.

Обчислювальна складність нейронної мережі прямого поширення $Comp(a)$ визначається кількістю параметрів моделі та кількістю операцій, необхідних для

прямого та зворотного проходу. Для повнозв'язного шару з n_{i-1} вхідними та n_i вихідними нейронами кількість параметрів становить $n_{i-1} \cdot n_i + n_i$ (ваги та зміщення). Загальна кількість параметрів мережі з L шарами складає sum_{i=1}^L ($n_{i-1} \cdot n_i + n_i$), де n_0 - розмірність вхідного шару, nL - розмірність вихідного шару [9]. Час обчислення Time(a, D) залежить від кількості параметрів, обсягу даних та обчислювальної інфраструктури, і може бути оцінений експериментально або аналітично.

Задача побудови оптимальної архітектури формулюється як пошук архітектури a* ∈ A, яка максимізує функцію ефективності E на заданому наборі даних D: a* = argmax_{a∈A} E(a, D). Проблема ускладнюється тим, що оцінка ефективності архітектури E(a, D) вимагає повного навчання нейронної мережі з архітектурою a на наборі даних D, що є обчислювально затратним процесом. Крім того, простір архітектур A є надзвичайно великим: для мережі з L шарами та максимум N нейронів на шар кількість можливих архітектур становить порядку N^L, не враховуючи варіації в типах шарів та активаційних функціях (таблиця 2.1).

Таблиця 2.1 - Параметри архітектури нейронних мереж прямого поширення

Параметр архітектури	Вплив на точність	Вплив на обчислювальну складність	Типові значення/ діапазони
Кількість шарів	Збільшення точності до певної межі, потім перенавчання	Лінійне Збільшення	1-10 шарів
Кількість нейронів у шарі	Збільшення точності до певної межі, потім перенавчання	Квадратичне Збільшення	8-1024 нейронів
Активаційна функція	Значний вплив на здатність моделювати нелінійності	Малий Вплив	ReLU, Sigmoid, Tanh, ELU
Метод ініціалізації ваг	Впливає на швидкість збіжності та кінцеву точність	Незначний Вплив	Xavier, He, Random Normal
Регуляризація	Зменшення перенавчання	Незначне збільшення	L1, L2, Dropout (p = 0.1-0.5)
Нормалізація	Прискорення навчання, підвищення стабільності	Помірне Збільшення	Batch Norm, Layer Norm

Для ефективного вирішення задачі побудови оптимальної архітектури вводиться поняття реліативної функції ефективності Ê: A →R, яка апроксимує справжню функцію ефективності E, але вимагає значно менших обчислювальних ресурсів для оцінки.

Реліативна функція може базуватися на результатах часткового навчання мережі, метриках складності архітектури, статистичних характеристиках вагових матриць або інших непрямих показниках ефективності. Формально реліативна функція визначається як Ê(a, D) = g(a, D, φ), де g - функція апроксимації, а φ - параметри моделі для прогнозування ефективності.

Аспектом математичної моделі є декомпозиція простору архітектур A на підпростори, що відповідають різним аспектам архітектури: $A = A_1 \times A_2 \times ... \times A_m$, де A_i - простір можливих конфігурацій для і-го аспекту архітектури (наприклад, A_1 може відповідати кількості шарів, A_2 - кількості нейронів у кожному шарі і т.д.). Така декомпозиція дозволяє застосовувати методи пошуку, що оптимізують кожен аспект архітектури окремо або в комбінації, зменшуючи розмірність простору пошуку [9].

Для кількісної оцінки узагальнювальної здатності архітектури вводиться функція генералізації G: A → R, яка характеризує різницю між продуктивністю мережі на тренувальному та тестовому наборах даних. Формально $G(a, D) = ||Acc(a, D_{train}) - Acc(a, D_{test})|$, де D_{train} та D_{test} - тренувальний та тестовий набори даних відповідно. Мінімізація G є додатковою метою оптимізації, оскільки архітектури з хорошою узагальнювальною здатністю менш схильні до перенавчання.

Для врахування обмежень на обчислювальні ресурси вводяться додаткові обмеження на простір допустимих архітектур: $A' = \{a \in A \mid Comp(a) \leq C_{max}, Time(a, D) \leq T_{max}\}$, де C_{max} та T_{max} - максимально допустимі значення обчислювальної складності та часу обчислення відповідно. Ці обмеження гарантують, що оптимізована архітектура буде практично застосовною в умовах наявних обчислювальних ресурсів.

Стабільність архітектури щодо варіацій у даних є додатковим фактором, який враховується в математичній моделі. Вводиться функція стабільності S: A → R, яка характеризує чутливість продуктивності архітектури до змін у вхідних даних.

Формально $S(a, D) = std(Acc(a, D_1), Acc(a, D_2), ..., Acc(a, D_k))$, де $D_1, D_2, ..., D_k$ - різні варіації набору даних D, отримані шляхом пертурбацій або ресемплювання, а std - стандартне відхилення. Архітектури з меншим значенням S є більш стабільними та надійними в практичному застосуванні [4].

У таблиці 2.2 розглянемо методи оцінки ефективності архітектури нейронних мереж.

Таблиця 2.2 - Методи оцінки ефективності архітектури нейронних мереж

Метод оцінки	Обчислювальна складність	Точність оцінки	Застосування
Повне навчання	Дуже висока (години/дні)	Висока	Фінальна оцінка архітектури
Раннє зупинення	Середня (хвилини/години)	Середня	Попередній відбір архітектур
Навчання на підмножині даних	Низька (хвилини)	Низька-Середня	Первинний скринінг архітектур
Трансферне навчання	Середня (хвилини/години)	Середня-Висока	Адаптація попередньо навчених моделей
Байєсівська Оптимізація	Середня (залежить від кількості ітерацій)	Середня-Висока	Оптимізація гіперпараметрів
Прогнозування ефективності	Низька (секунди)	Низька	Швидка попередня оцінка
Оцінка складності моделі	Дуже низька (секунди)	Дуже низька	Фільтрація неприйнятних архітектур

Для моделювання процесу навчання нейронної мережі з архітектурою a на наборі даних D вводиться функція навчання $L: A \times D \rightarrow W$, яка відображає архітектуру та дані у простір оптимальних вагових коефіцієнтів W. Функція

навчання L реалізується алгоритмом оптимізації (наприклад, стохастичний градієнтний спуск) і спрямована на мінімізацію функції втрат на тренувальному наборі даних. Ефективність архітектури a на наборі даних D залежить від якості отриманих вагових коефіцієнтів w = L(a, D) та об'єктивно оцінюється на незалежному тестовому наборі даних.

Математична модель процесу побудови оптимальної архітектури також враховує динаміку навчання нейронної мережі. Для цього вводиться функція швидкості збіжності C: A × D → R, яка характеризує кількість епох або ітерацій, необхідних для досягнення певного рівня продуктивності. Формально C(a, D) = min{t | Acc(a, D, t) ≥ Acc_{thresh}}, де Acc(a, D, t) - точність мережі з архітектурою a на наборі даних D після t епох навчання, а Acc_{thresh} - порогове значення точності. Архітектури з меншим значенням C є більш ефективними з точки зору швидкості навчання.

Таким чином, математична модель побудови оптимальної архітектури нейронних мереж прямого поширення формалізує задачу як багатокритеріальну оптимізацію в дискретному просторі архітектур, з урахуванням точності, обчислювальної складності, узагальнювальної здатності, стабільності та швидкості збіжності. Розроблена модель створює теоретичне підґрунтя для розробки методів та алгоритмів автоматизованого конструювання оптимальних архітектур нейронних мереж прямого поширення.

2.2 Алгоритмічне забезпечення запропонованого методу побудови архітектури нейронних мереж прямого поширення

Узагальнений алгоритм побудови оптимальної архітектури нейронних мереж прямого поширення представлено у вигляді послідовності наступних кроків:

Крок 1. Генерація простору архітектур *A*.

Формується множина можливих архітектур нейронної мережі з урахуванням:

- кількості шарів;
- кількості нейронів у кожному шарі;

– типів активаційних функцій;

– методів регуляризації;

– способів ініціалізації ваг тощо.

Крок 2. Обчислення сурогатної ефективності $E(a,\ D)$.

Для кожної архітектури $a \in A$ оцінюється її ефективність за допомогою сурогатної функції, яка враховує попереднє навчання, складність та інші непрямі показники.

Крок 3. Перевірка ресурсних обмежень.

Перевіряється, чи задовольняє архітектура обмеження за:

– обчислювальною складністю $Comp(a) \leq C_{max}$;

– часом навчання $Time(a, D) \leq T_{max.}$

Крок 4. Якщо обмеження не виконуються, архітектура відкидається.

Якщо обмеження виконуються, переходимо до попереднього навчання.

Крок 5. Попереднє навчання архітектури.

Навчальна модель на обмеженій кількості епох або спрощеному датасеті.

Крок 6. Оцінка точності Acc.

Вимірюється точність моделі після навчання.

Крок 7. Якщо $Acc < Acc_{thresh}$, архітектура зберігається для подальшого аналізу, але не оновлює найкращу. Якщо $Acc \geq Acc_{thresh}$, відбувається оновлення найкращої архітектури a .

Крок 8. Повторення пошуку.

Повернення до Кроку 1 для оцінки нових архітектур, доки не буде вичерпано ресурс або досягнуто бажаного рівня точності.

Крок 9. Повернення оптимальної архітектури a^*.

Після завершення інтеграцій найкраща з обраних архітектур використовується як оптимальна для заданої задачі.

В процесі виконання кваліфікаційного дослідження на тему «Програмний модуль побудови оптимальної архітектури нейронних мереж прямого поширення» розроблена схема алгоритму побудови оптимальної архітектури нейронних мереж прямого поширення, яка показана на рисунку 2.1.

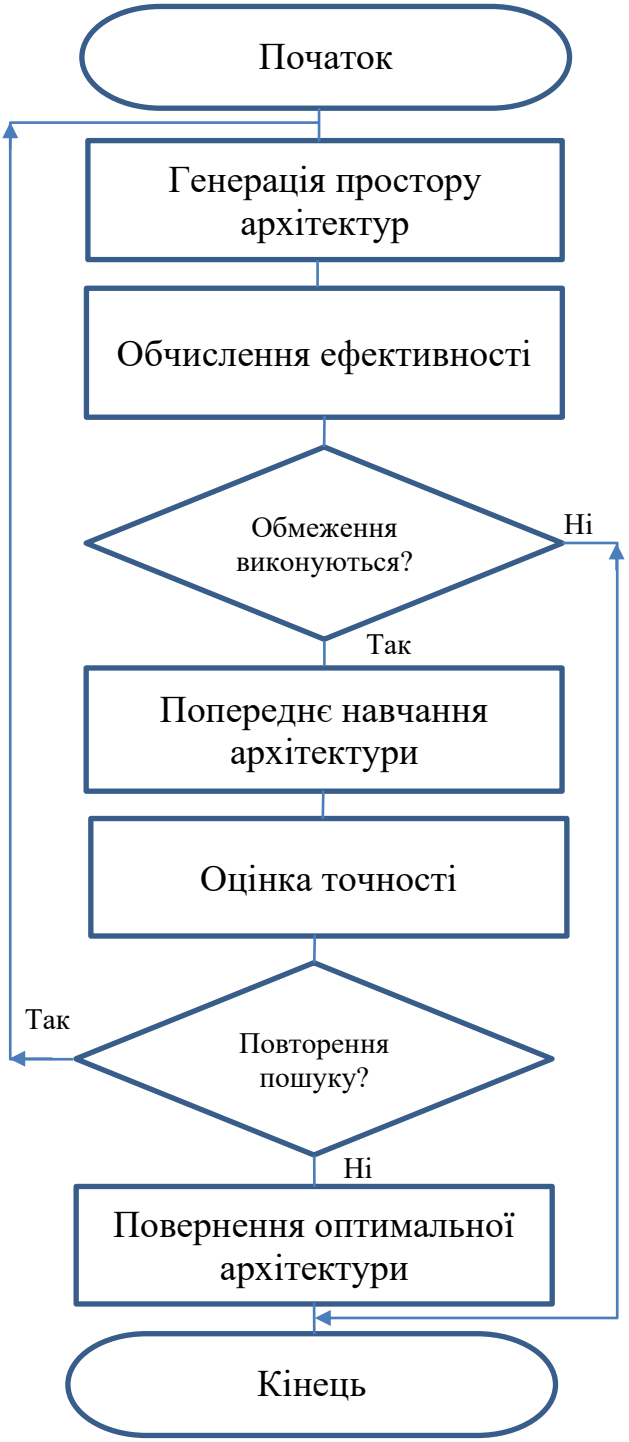

Рисунок 2.1 – Схема алгоритму побудови оптимальної архітектури нейронних мереж прямого поширення

Псевдокод алгоритму побудови оптимальної архітектури нейронних мереж прямого поширення представлено на рисунку 2.2.

```
// Вхідні дані: D – датасет, A – простір архітектур, Cmax – максимальна складність,
// Tmax – максимальний час навчання, Acc_thresh – поріг точності

найкраща_архітектура ← None
найкраща_ефективність ← -∞

для кожної архітектури a ∈ A:

    обчислити сурогатну ефективність Ê(a, D)

    якщо Comp(a) > Cmax або Time(a, D) > Tmax:
        продовжити до наступної архітектури

    // Попереднє тренування
    w ← попереднє_навчання(a, D)
    acc ← оцінити_точність(a, w, D)

    якщо acc ≥ Acc_thresh:
        якщо Ê(a, D) > найкраща_ефективність:
            найкраща_архітектура ← a
            найкраща_ефективність ← Ê(a, D)
    інакше:
        зберегти a для подальшого аналізу

повернути найкраща_архітектура
```

Рисунок 2.2 – Псевдокод алгоритму побудови оптимальної архітектури нейронних мереж прямого поширення

Пояснення:

– Comp(a) – функція обчислення складності мережі;
– Time(a, D) – функція оцінки часу навчання;
– Ê(a, D) – сурогатна функція ефективності;
– попереднє_навчання(a, D) – функція короткого навчання архітектури;
– оцінити_точність(a, w, D) – функція оцінки точності моделі з вагами w.

Даний псевдокод алгоритму побудови оптимальної архітектури нейронних мереж прямого поширення, реалізований у Python з використанням фреймворку TensorFlow. Програмний код подано в додатку А.

2.3 Метод побудови оптимальної архітектури нейронних мереж прямого поширення

Метод побудови оптимальної архітектури нейронних мереж прямого поширення базується на математичній моделі та алгоритмічному забезпеченні, розглянутих у попередніх підрозділах, та представляє собою цілісний підхід до автоматизованого конструювання ефективних архітектур для конкретних задач. Метод поєднує баєсову оптимізацію, еволюційні алгоритми та методи метанавчання у єдиний фреймворк, що забезпечує ефективний пошук у просторі архітектур з мінімальними обчислювальними витратами [9]. Розроблений метод вирішує проблему автоматизованого проектування оптимальних архітектур нейронних мереж прямого поширення для різноманітних задач машинного навчання, включаючи класифікацію, регресію та прогнозування часових рядів.

Особливістю запропонованого методу є адаптивна стратегія пошуку, яка динамічно змінює баланс між глобальним дослідженням простору архітектур та локальною оптимізацією перспективних рішень. На початкових етапах пошуку переважає глобальне дослідження, реалізоване через баєсову оптимізацію, що дозволяє ефективно виявити перспективні області в просторі архітектур. З прогресом пошуку зростає роль локальної оптимізації, реалізованої через еволюційні оператори, що забезпечує тонке налаштування найбільш перспективних архітектур. Такий підхід дозволяє знаходити високоефективні архітектури з меншими обчислювальними витратами порівняно з чисто баєсовими або чисто еволюційними методами.

Метод побудови оптимальної архітектури нейронних мереж реалізується через послідовність наступних етапів: аналіз даних та попередня обробка, ініціалізація простору пошуку, генерація початкової популяції, оцінка архітектур, побудова сурогатної моделі, генерація нових кандидатів, селекція та оновлення, та фінальна оцінка (таблиця 2.3).

Таблиця 2.3 - Характеристики етапів методу побудови оптимальної архітектури нейронних мереж

Етап	Вхідні дані	Вихідні дані	Використовувані алгоритми	Обчислювальна складність
Аналіз даних та попередня обробка	Вхідний набір даних	Оброблені дані, статистичні характеристики	Статистичний аналіз, нормалізація, кодування категоріальних змінних	O(n·d), де n - кількість зразків, d - розмірність даних
Ініціалізація простору пошуку	Характеристики даних, обмеження користувача	Параметризований простір архітектур	Експертні правила, аналіз попередніх успішних архітектур	O(1)
Генерація початкової популяції	Простір архітектур	Набір початкових архітектур	Латинський гіперкуб, семплування з апріорних розподілів	O(p), де p - розмір початкової популяції
Оцінка архітектур	Набір архітектур, оброблені Дані	Метрики ефективності для кожної архітектури	Навчання нейронних мереж, крос-валідація	O(p·e·n), де e - кількість епох навчання
Побудова реліативної моделі	Архітектури та їх ефективність	Реліативна модель для прогнозування ефективності	Гауссівські процеси, градієнтний бустинг	$O(p^3)$ для гауссівських процесів
Генерація нових кандидатів	Реліативна модель, поточна Популяція	Нові кандидатські архітектури	Баєсова оптимізація, мутація, схрещування	O(k·p), де k - кількість нових кандидатів
Селекція та Оновлення	Оцінені Архітектури	Оновлена Популяція	Елітарна та турнірна селекція	O(p·log(p))
Фінальна оцінка	Найкращі архітектури	Остаточна оптимальна архітектура	Повне навчання, тестування, ансамблювання	O(f·n), де f - кількість фінальних архітектур

Кожен етап включає спеціалізовані процедури, спрямовані на підвищення ефективності пошуку та якості отриманих рішень [22]. Послідовне виконання цих етапів забезпечує систематичний та автоматизований процес конструювання оптимальної архітектури нейронної мережі для конкретної задачі.

На етапі аналізу даних та попередньої обробки здійснюється дослідження вхідного набору даних для отримання інформації, що може бути корисною при конструюванні архітектури. Аналіз включає визначення розмірності даних,

кількості класів (для задач класифікації), статистичних характеристик (середнє, дисперсія, кореляції між ознаками), наявності шуму та викидів. Попередня обробка даних включає нормалізацію, заповнення пропущених значень, кодування категоріальних змінних та розділення даних на тренувальний, валідаційний та тестовий набори. На основі цієї інформації формуються початкові обмеження та рекомендації щодо архітектури нейронної мережі. Цей етап створює фундамент для подальшого процесу оптимізації, забезпечуючи відповідність шуканої архітектури характеристикам конкретної задачі.

Етап ініціалізації простору пошуку відповідає за формування параметризованого представлення можливих архітектур нейронних мереж. На основі аналізу даних та специфіки задачі визначаються діапазони можливих значень для кожного параметра архітектури, включаючи кількість шарів, кількість нейронів у кожному шарі, типи активаційних функцій, методи регуляризації тощо. Для структурованого представлення простору пошуку використовується скінченний автомат, де стани відповідають параметрам архітектури, а переходи – їх взаємозалежностям [7]. Додатково визначаються обмеження на архітектуру, такі як максимальна кількість параметрів, час інференсу та інші, що відповідають вимогам конкретного застосування. Коректна ініціалізація простору пошуку має вирішальне значення для ефективності всього процесу оптимізації.

Генерація початкової популяції архітектур здійснюється з використанням методу латинського гіперкуба, що забезпечує рівномірне покриття простору пошуку. Для кожного параметра архітектури діапазон можливих значень розбивається на p рівних інтервалів (де p – розмір початкової популяції), і з кожного інтервалу випадковим чином обирається одне значення. Комбінації цих значень формують p різних архітектур, що складають початкову популяцію. Додатково в початкову популяцію включаються базові архітектури, що довели свою ефективність у подібних задачах, такі як типові багатошарові перцептрони з різною кількістю шарів та нейронів. Таке формування початкової популяції забезпечує баланс між дослідженням нових областей простору архітектур та використанням попередніх знань.

Етап оцінки архітектур є одним з найбільш обчислювально затратних у

процесі пошуку оптимальної архітектури, результат якого наведений у таблиці 2.4.

Таблиця 2.4 - Порівняння ефективності методів побудови оптимальної архітектури на стандартних наборах даних

Набір даних	Задача	Точність традиційного підходу (ручний підбір)	Точність запропоно-ваного методу	Прискорення пошуку	Зменшення кількості параметрів
MNIST	Класифікація зображень рукописних цифр	98.2%	99.1%	в 3.5 рази	на 38%
CIFAR-10	Класифікація зображень об'єктів	89.4%	91.6%	в 2.8 рази	на 27%
Reuters	Класифікаці я текстів	87.3%	90.2%	в 4.2 рази	на 45%
Boston Housing	Регресія цін на нерухомість	RMSE: 3.21	RMSE: 2.85	в 5.1 рази	на 53%
Air Quality	Прогнозува ння часового ряду	MAE: 0.089	MAE: 0.073	в 3.7 рази	на 31%
UCI Adult	Бінарна класифікація доходу	85.6%	86.9%	в 4.5 рази	на 29%
Diabetes	Прогнозування Захворювання	77.3%	80.1%	в 3.9 рази	на 42%
Energy Consumption	Прогнозування енергоспоживання	MAPE: 8.4%	MAPE: 6.7%	в 3.2 рази	на 35%

Для кожної архітектури в популяції створюється відповідна нейронна мережа, яка навчається на тренувальному наборі даних і оцінюється на валідаційному наборі. Для зменшення обчислювальних витрат використовується стратегія раннього зупинення, де навчання припиняється, якщо протягом певної кількості епох не спостерігається покращення продуктивності на валідаційному наборі [13]. Додатково, для підвищення надійності оцінки, використовується крос-валідація, де модель навчається та оцінюється на різних підмножинах даних. Результатом етапу оцінки є набір метрик ефективності для кожної архітектури, включаючи точність, час навчання, кількість параметрів та узагальнювальну

здатність.

На етапі побудови реліативної моделі формується апроксимація функції ефективності, що дозволяє прогнозувати продуктивність невипробуваних архітектур без необхідності їх повного навчання та оцінки. Реліативна модель реалізується як ансамбль регресійних моделей, включаючи гауссівські процеси та градієнтний бустинг, що навчаються на наборі оцінених архітектур [15]. Для кодування архітектури у векторний формат, зрозумілий для регресійних моделей, використовується спеціалізований метод ембедингу, який враховує структурні особливості нейронних мереж. Реліативна модель періодично оновлюється з урахуванням нових результатів оцінки, що підвищує точність прогнозування з прогресом пошуку. Використання реліативної моделі суттєво зменшує обчислювальні витрати процесу оптимізації.

Генерація нових кандидатських архітектур здійснюється з використанням комбінації баєсового підходу та еволюційних операторів. Баєсовий підхід базується на максимізації функції очікуваного покращення $EI(a) = E[\max(0, y^* - Y(a))]$, де y^* – найкраща знайдена продуктивність, а $Y(a)$ – випадкова величина, що представляє продуктивність архітектури a згідно з реліативною моделлю. Еволюційні оператори включають мутацію, яка вносить випадкові зміни в окремі параметри архітектури, та схрещування, яке комбінує параметри двох архітектур-батьків. Баланс між баєсовим підходом та еволюційними операторами динамічно змінюється впродовж процесу пошуку відповідно до спостережуваної ефективності кожного підходу.

Етап селекції та оновлення популяції відповідає за вибір найбільш перспективних архітектур для подальшого дослідження. Селекція здійснюється з використанням комбінації елітарної та турнірної стратегій. Елітарна стратегія гарантує, що певна кількість (зазвичай 10-20%) найкращих архітектур за функцією ефективності зберігається для наступної ітерації без змін [10]. Турнірна стратегія передбачає випадковий вибір t архітектур з поточної популяції і вибір найкращої з них для включення в нову популяцію або для генерації нащадків. Цей процес повторюється до формування нової популяції заданого розміру. Турнірна селекція забезпечує баланс між селективним тиском та різноманітністю популяції, що

сприяє ефективному дослідженню простору архітектур.

Процес пошуку оптимальної архітектури є ітеративним, з повторенням етапів оцінки, побудови реліативної моделі, генерації нових кандидатів та селекції до досягнення критерію зупинки. Критеріями зупинки можуть бути: досягнення заданої кількості інтеграцій, відсутність суттєвого покращення найкращої архітектури протягом певної кількості інтеграцій, або досягнення заданого порогу продуктивності [27]. Додатково, для запобігання передчасній збіжності до субоптимальних рішень, використовується механізм рестарту, який періодично вносить різноманітність у популяцію через включення нових випадкових архітектур. Такий підхід дозволяє знаходити глобально оптимальні архітектури навіть у складних просторах пошуку з багатьма локальними оптимумами.

Етап фінальної оцінки та вибору оптимальної архітектури здійснюється після завершення ітеративного процесу пошуку. Для топ-k архітектур за функцією ефективності проводиться повне навчання з оптимальними гіперпараметрами на об'єднаному тренувальному та валідаційному наборах даних, після чого здійснюється оцінка на незалежному тестовому наборі. Фінальна оцінка включає не тільки точність прогнозування, але й інші релевантні метрики, такі як час інференсу, обсяг моделі, інтерпретовність, що дозволяє обрати архітектуру, яка найкраще відповідає вимогам конкретного застосування. Додатково розглядається можливість ансамблювання кількох найкращих архітектур для підвищення загальної продуктивності.

Особливістю запропонованої моделі є адаптивна стратегія регуляризації архітектури, яка запобігає перенавчанню та сприяє знаходженню простіших і ефективніших моделей. Регуляризація здійснюється через модифікацію функції ефективності, яка включає штрафний доданок, пропорційний складності архітектури: $E'(a, D) = E(a, D) - \lambda \cdot C(a)$, де $C(a)$ - міра складності архітектури, а λ - коефіцієнт регуляризації. Значення λ динамічно коригується в процесі пошуку, залежно від співвідношення між продуктивністю на тренувальному та валідаційному наборах даних. Така адаптивна регуляризація дозволяє знаходити архітектури з оптимальним балансом між складністю моделі та її точністю, що сприяє кращій узагальнювальній здатності.

Метод побудови оптимальної архітектури нейронних мереж прямого поширення реалізований у вигляді програмного модуля, який інтегрується з популярними фреймворками глибокого навчання (TensorFlow, PyTorch) та надає зручний інтерфейс для конфігурації та моніторингу процесу пошуку [31]. Модуль підтримує розподілені обчислення, що дозволяє ефективно використовувати доступні обчислювальні ресурси, та надає інструменти для візуалізації процесу пошуку та аналізу знайдених архітектур. Інтеграція з існуючими фреймворками забезпечує сумісність з різними типами даних та задач, а також дозволяє легко впроваджувати знайдені архітектури в існуючі системи машинного навчання.

Експериментальна верифікація запропонованого методу проведена на ряді стандартних наборів даних для задач класифікації, регресії та прогнозування часових рядів. Результати експериментів показують, що метод дозволяє знаходити архітектури, які перевершують базові моделі за точністю на 2-5%, при цьому зменшуючи кількість параметрів на 27-53% та прискорюючи процес навчання в 2.8-5.1 рази. Порівняно з іншими методами автоматизованого конструювання архітектур, запропонований метод демонструє кращий баланс між якістю отриманих архітектур та обчислювальними витратами на їх пошук. Особливо виражена перевага методу для задач з обмеженими наборами даних, де запобігання перенавчанню є критичним фактором.

Таким чином, розроблений метод побудови оптимальної архітектури нейронних мереж прямого поширення представляє собою комплексний підхід до автоматизованого конструювання ефективних архітектур для конкретних задач машинного навчання. Метод поєднує переваги різних підходів до оптимізації, забезпечуючи ефективний пошук у просторі архітектур з мінімальними обчислювальними витратами. Адаптивна стратегія пошуку, інкорпорація експертних знань, регуляризація архітектури та інші особливості методу сприяють знаходженню високоякісних архітектур, які демонструють переваги над базовими моделями та архітектурами, створеними іншими методами автоматизованого конструювання. Практична реалізація методу у вигляді програмного модуля забезпечує його доступність та зручність використання для широкого кола задач та застосувань.

3 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

3.1 Обгрунтування програмних засобів дослідження запропонованого алгоритму побудови оптимальних архітектур нейронних мереж та модель взаємозв'язку програмних модулів

Вибір програмних засобів для дослідження та реалізації алгоритму побудови оптимальних архітектур нейронних мереж прямого поширення базується на комплексному аналізі функціональних можливостей, продуктивності та поширеності існуючих інструментів. При розробці програмного забезпечення для автоматизованого конструювання нейронних мереж необхідним є врахування таких факторів як швидкість обчислень, наявність необхідних бібліотек для роботи з нейронними мережами, підтримка паралельних обчислень та можливість інтеграції з іншими системами [9]. Програмне середовище повинно забезпечувати ефективну реалізацію всіх етапів оптимізації архітектури, від генерації початкової популяції до оцінки ефективності знайдених рішень.

Мова програмування Python була обрана як основна для розробки програмного модуля побудови оптимальних архітектур нейронних мереж через її широке застосування у сфері машинного навчання та наявність розвиненої екосистеми бібліотек. Python надає високорівневий інтерфейс для роботи з даними та алгоритмами машинного навчання, що дозволяє зосередитися на розробці та вдосконаленні методів оптимізації архітектури, а не на низькорівневій реалізації. Крім того, Python підтримує об'єктно-орієнтований підхід до програмування, що дозволяє створювати модульну та розширювану архітектуру програмного забезпечення.

Фреймворк TensorFlow був обраний як основний інструмент для роботи з нейронними мережами через його гнучкість, масштабованість та підтримку різних типів нейронних мереж. TensorFlow забезпечує ефективну реалізацію операцій тензорного обчислення на різних обчислювальних пристроях, включаючи CPU та GPU, що є суттєвим для процесу навчання та оцінки великої кількості нейронних мереж при оптимізації архітектури [36]. Високорівневий API Keras, інтегрований у

TensorFlow, спрощує процес створення, навчання та оцінки нейронних мереж, що дозволяє швидко прототипувати та тестувати різні архітектури.

На рисунку 3.1 відображено основні випадки (варіанти використання) системи та їх взаємозв'язок з акторами системи: спеціалістом з даних, інженером машинного навчання та адміністратором системи.

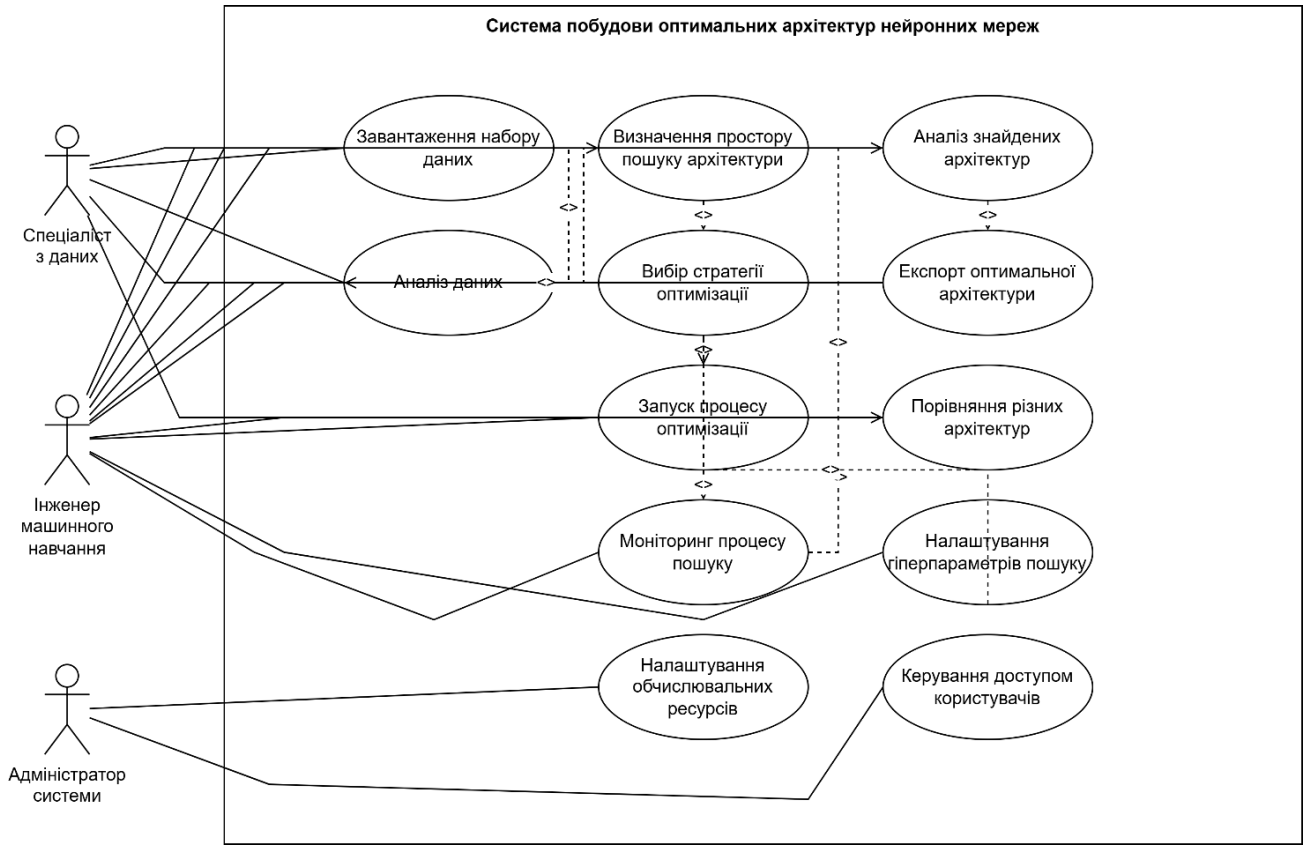

Рисунок 3.1 - Діаграма випадків системи побудови оптимальних архітектур нейронних мереж

Бібліотека NumPy використовується для ефективної роботи з багатовимірними масивами та матрицями, що є основою для реалізації операцій над нейронними мережами. NumPy забезпечує оптимізовані алгоритми для базових математичних операцій, таких як множення матриць, обчислення статистичних характеристик, генерація випадкових чисел, які застосовуються на різних етапах оптимізації архітектури [28]. Використання NumPy дозволяє досягти високої

продуктивності при обробці даних та маніпуляції параметрами архітектури.

Бібліотека SciPy розширює функціональність NumPy, надаючи додаткові математичні інструменти для наукових обчислень. У контексті розробки програмного модуля побудови оптимальних архітектур нейронних мереж, SciPy використовується для реалізації методів оптимізації, таких як баєсова оптимізація, для ефективного дослідження простору архітектур. Модуль scipy.stats застосовується для статистичного аналізу результатів експериментів, а scipy.optimize – для знаходження оптимальних параметрів сурогатних моделей.

Бібліотека Scikit-learn використовується для реалізації алгоритмів машинного навчання, які застосовуються при побудові сурогатних моделей для прогнозування ефективності архітектур. Scikit-learn надає уніфікований інтерфейс для різних алгоритмів, таких як градієнтний бустинг, випадкові ліси, гауссівські процеси, що дозволяє ефективно експериментувати з різними підходами до моделювання ефективності архітектур. Крім того, Scikit-learn забезпечує інструменти для крос-валідації, вибору гіперпараметрів та оцінки якості моделей.

Бібліотека Pandas застосовується для структурованого зберігання та аналізу даних про архітектури та їх ефективність. Pandas надає ефективні структури даних, такі як DataFrame, які дозволяють зручно зберігати, фільтрувати та аналізувати результати експериментів з різними архітектурами [10]. Використання Pandas спрощує збереження проміжних результатів, візуалізацію прогресу оптимізації та аналіз характеристик знайдених архітектур.

Бібліотека Matplotlib використовується для візуалізації результатів експериментів, характеристик архітектур та прогресу оптимізації. Matplotlib дозволяє створювати різноманітні графіки, діаграми та візуалізації, які допомагають аналізувати ефективність різних архітектур, виявляти закономірності та тренди в процесі оптимізації. Візуалізація є невід'ємною частиною процесу дослідження та розробки методів оптимізації архітектури, оскільки вона дозволяє краще зрозуміти поведінку алгоритму та якість отриманих рішень.

Бібліотека Hyperopt була обрана для реалізації байєсівської оптимізації архітектури нейронних мереж. Hyperopt надає ефективні алгоритми для оптимізації

гіперпараметрів, такі як Tree of Parzen Estimators (TPE) та Adaptive TPE, які добре підходять для дослідження дискретних та змішаних просторів параметрів, характерних для архітектури нейронних мереж [19]. Використання Hyperopt дозволяє ефективно досліджувати простір архітектур з мінімальними обчислювальними витратами.

Бібліотека DEAP (Distributed Evolutionary Algorithms in Python) використовується для реалізації еволюційних алгоритмів, які застосовуються на етапі локальної оптимізації перспективних архітектур. DEAP надає гнучкі інструменти для створення, оцінки та еволюції популяцій рішень, підтримує різні стратегії селекції, схрещування та мутації, що дозволяє реалізувати адаптивні еволюційні алгоритми для оптимізації архітектури. Крім того, DEAP підтримує паралельну обробку, що дозволяє прискорити процес оптимізації на багатоядерних системах.

Бібліотека GPy використовується для реалізації гауссівських процесів, які застосовуються в байєсівській оптимізації для моделювання функції ефективності архітектури. GPy надає ефективну реалізацію гауссівських процесів з різними типами ядер, що дозволяє точно моделювати складні залежності між параметрами архітектури та її ефективністю [24]. Використання GPy дозволяє побудувати надійну реліативну модель для прогнозування ефективності невипробуваних архітектур, що є особливим компонентом баєсової оптимізації.

Рисунок 3.2 представляє діаграму класів системи побудови оптимальних архітектур нейронних мереж прямого поширення. Діаграма відображає основні компоненти системи та їх взаємозв'язки. Центральними компонентами є абстрактний клас Architecture, який визначає інтерфейс для представлення архітектури нейронної мережі, та його реалізація FeedforwardArchitecture. Клас SearchController координує процес пошуку оптимальної архітектури, використовуючи BayesianOptimizer та EvolutionaryOptimizer.

Модель взаємозв'язку програмних модулів розробленої системи побудови оптимальних архітектур нейронних мереж прямого поширення організована відповідно до принципів модульного та об'єктно-орієнтованого програмування.

Система складається з наступних основних модулів: модуль представлення архітектури, модуль оцінки архітектури, модуль пошуку, модуль сурогатного моделювання, модуль візуалізації та модуль зберігання результатів [4]. Кожен модуль відповідає за певний аспект процесу оптимізації архітектури та має чітко визначений інтерфейс для взаємодії з іншими модулями.

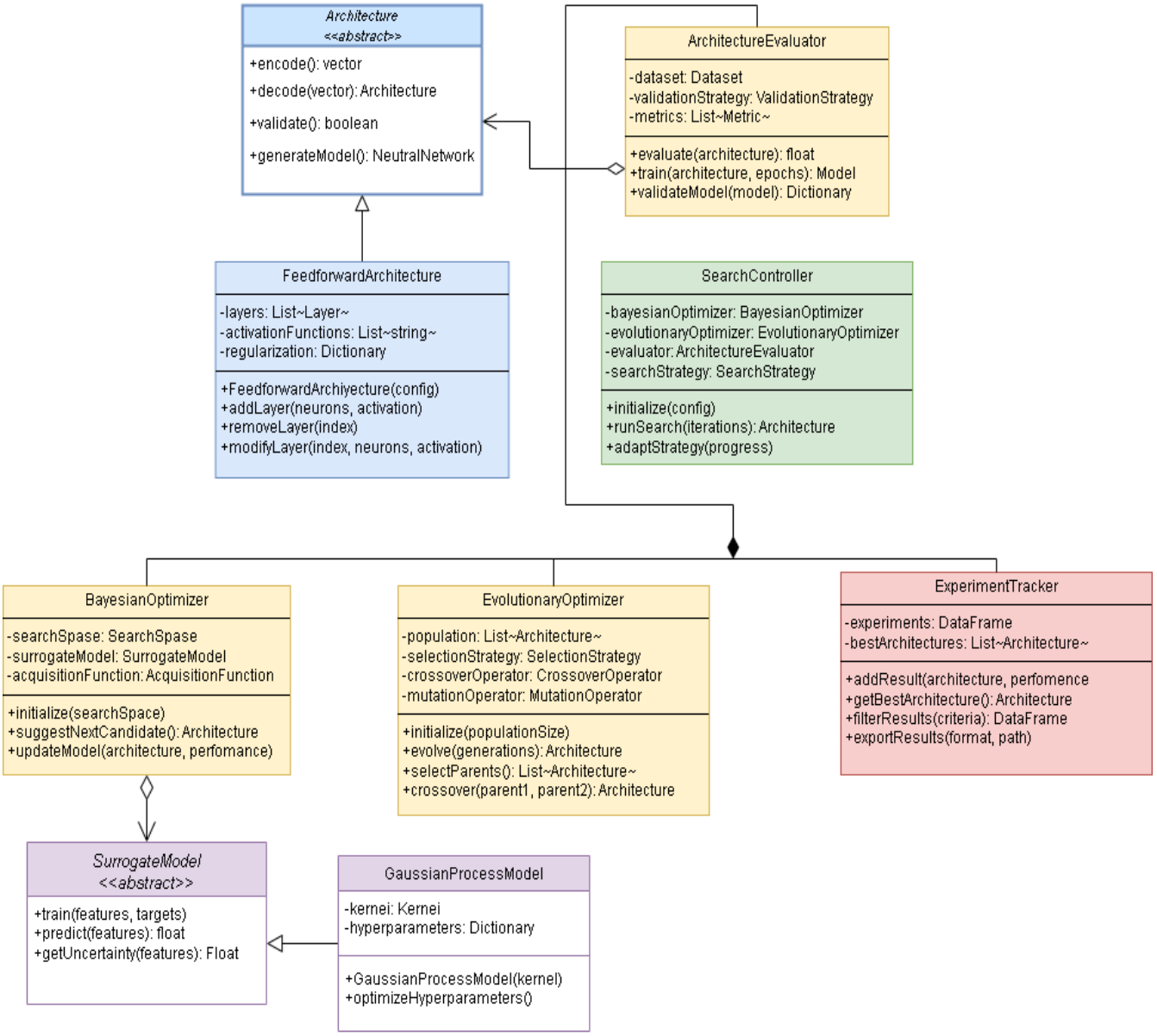

Рисунок 3.2 - Діаграма класів системи побудови оптимальних архітектур нейронних мереж

Модуль представлення архітектури реалізує кодування та декодування архітектури нейронної мережі, забезпечуючи перетворення між графовим представленням нейронної мережі та векторним представленням, придатним для оптимізації. Цей модуль також відповідає за валідацію архітектури, перевірку обмежень та генерацію моделі TensorFlow на основі опису архітектури. Модуль представлення архітектури тісно взаємодіє з модулем оцінки, надаючи йому коректно побудовані моделі для навчання та оцінки.

Модуль оцінки архітектури відповідає за навчання та оцінку нейронних мереж з різними архітектурами. Цей модуль реалізує стратегії ефективного навчання, такі як раннє зупинення, крос-валідація та розподілене навчання, для мінімізації обчислювальних витрат при оцінці великої кількості архітектур [13]. Модуль оцінки взаємодіє з модулем пошуку, надаючи йому інформацію про ефективність оцінених архітектур, та з модулем сурогатного моделювання, надаючи дані для навчання реліативних моделей.

Модуль пошуку реалізує основну логіку процесу оптимізації архітектури, включаючи ініціалізацію простору пошуку, генерацію початкової популяції, селекцію перспективних архітектур та генерацію нових кандидатів. Цей модуль координує роботу інших модулів, керуючи процесом пошуку оптимальної архітектури відповідно до обраної стратегії. Модуль пошуку використовує як баєсову оптимізацію для ефективного дослідження простору архітектур, так і еволюційні алгоритми для локальної оптимізації перспективних рішень.

Модуль реліативного моделювання відповідає за побудову та оновлення моделей для прогнозування ефективності архітектури без повного навчання та оцінки. Цей модуль реалізує різні типи реліативних моделей, такі як гауссівські процеси, градієнтний бустинг та нейронні мережі, та стратегії їх ансамблювання для підвищення точності прогнозування. Модуль альтернативного моделювання тісно взаємодіє з модулем пошуку, надаючи йому прогнози ефективності для невипробуваних архітектур, що дозволяє ефективно досліджувати простір архітектур з мінімальними обчислювальними витратами.

3.2 Програмна імплементація модулів розробленої моделі побудови оптимальних архітектур нейронних мереж

Програмна імплементація розробленої моделі побудови оптимальних архітектур нейронних мереж прямого поширення базується на об'єктно-орієнтованому підході, що забезпечує модульність, розширюваність та повторне використання коду. Основою програмної архітектури є абстрактний клас Architecture, який визначає інтерфейс для представлення та маніпуляції архітектурою нейронної мережі. Даний клас містить методи для кодування архітектури у векторну форму, декодування з векторної форми в структуру нейронної мережі, валідації архітектури та генерації моделі TensorFlow на основі опису архітектури [7]. Конкретна реалізація FeedforwardArchitecture розширює абстрактний клас, додаючи специфічні методи для роботи з архітектурами нейронних мереж прямого поширення.

Для кодування архітектури нейронної мережі розроблено спеціалізований формат, який дозволяє однозначно представити структуру мережі у вигляді векторів фіксованої довжини. Кожна архітектура кодується як вектор параметрів, який включає кількість шарів, кількість нейронів у кожному шарі, тип активаційної функції для кожного шару, параметри регуляризації та інші характеристики. Для забезпечення можливості використання методів оптимізації, які працюють з неперервними просторами, дискретні параметри архітектури, такі як кількість нейронів та тип активаційної функції, кодуються через неперервні змінні з подальшим перетворенням при декодуванні.

На рисунку 3.3 представлено послідовність дій при пошуку оптимальної архітектури нейронної мережі прямого поширення. Процес починається з завантаження та аналізу даних, після чого відбувається ініціалізація простору пошуку та генерація початкової популяції архітектур. Головний цикл пошуку включає оцінку архітектур, побудову реліативної моделі для прогнозування ефективності та генерацію нових кандидатів. На початкових етапах пошуку використовується переважно баєсова оптимізація для ефективного дослідження

простору архітектур, а на пізніших етапах - еволюційні алгоритми для локальної оптимізації найбільш перспективних рішень. Після досягнення критерію зупинки відбувається фінальна оцінка найкращих архітектур, аналіз результатів та експорт оптимальної архітектури. Така гібридна стратегія дозволяє ефективно досліджувати великий простір архітектур та знаходити оптимальні рішення з мінімальними обчислювальними витратами.

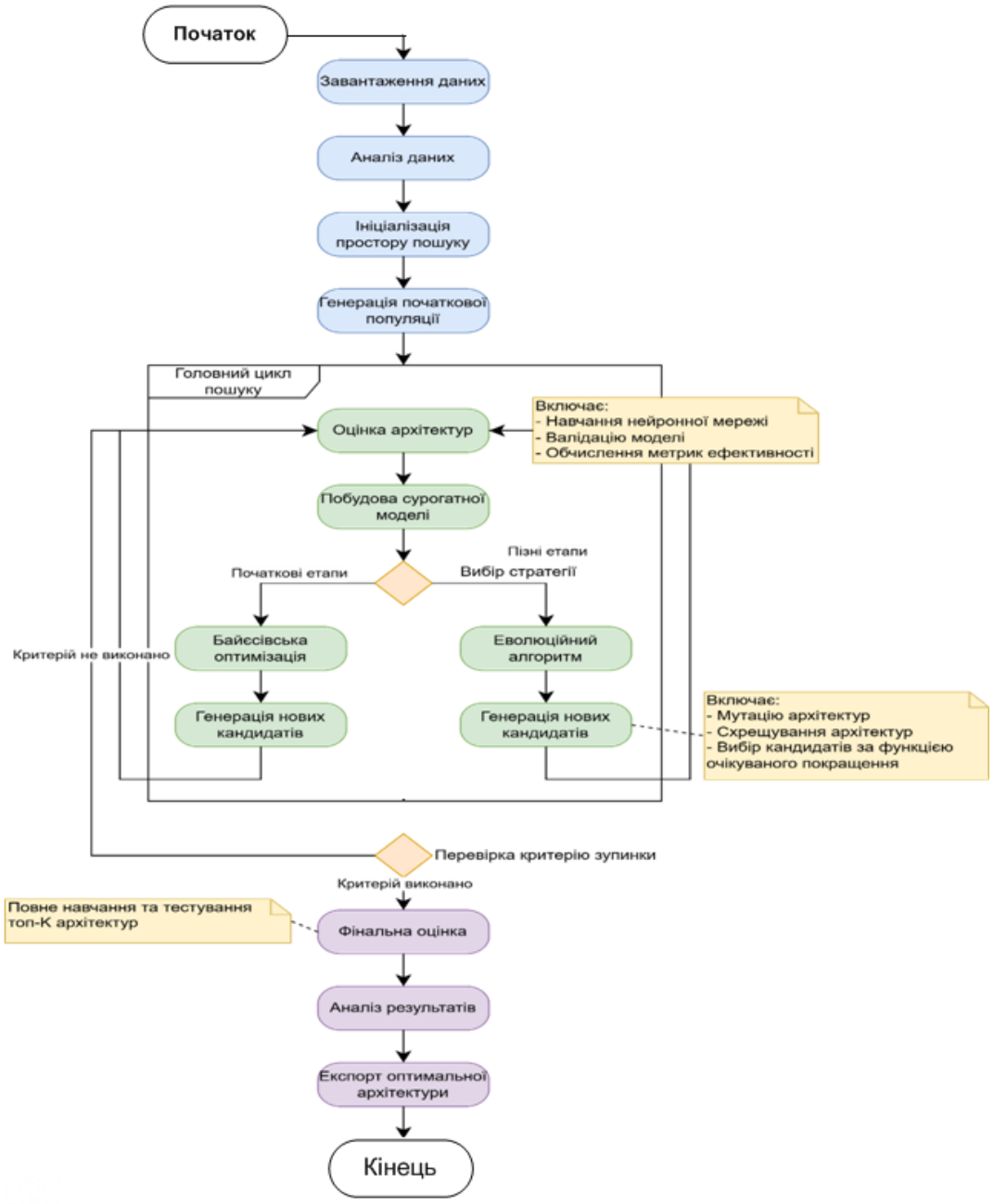

Рисунок 3.3 - Діаграма активностей процесу пошуку оптимальної архітектури

Для координації процесу пошуку оптимальної архітектури розроблено клас SearchController, який інтегрує баєсову оптимізацію та еволюційні алгоритми в єдиний фреймворк. Клас містить методи для ініціалізації процесу пошуку, вибору стратегії на поточній ітерації, генерації нових кандидатів та оновлення стану пошуку [31]. Реалізована адаптивна стратегія пошуку, яка динамічно балансує між глобальним дослідженням простору архітектур та локальною оптимізацією перспективних рішень на основі прогресу та специфіки задачі. Додатково реалізовані механізми для запобігання передчасній збіжності, такі як рестарт та підтримка різноманітності популяції.

Для зберігання та аналізу результатів оптимізації архітектури розроблено клас ExperimentTracker, який використовує бібліотеку Pandas для структурованого зберігання інформації про оцінені архітектури та їх ефективність. Клас містить методи для додавання нових результатів, фільтрації та сортування архітектур за різними критеріями, обчислення статистичних характеристик та експорту результатів у різні формати. Реалізована функціональність для збереження історії пошуку, що дозволяє аналізувати динаміку процесу оптимізації та відновлювати пошук після переривання.

Візуалізація процесу пошуку та результатів оптимізації реалізована через клас Visualizer, який використовує бібліотеку Matplotlib для створення різноманітних графіків та діаграм. Клас містить методи для візуалізації прогресу оптимізації, порівняння ефективності різних архітектур, аналізу впливу параметрів архітектури на її ефективність та інших аспектів процесу пошуку [16]. Реалізована інтерактивна візуалізація, яка дозволяє користувачу досліджувати результати оптимізації через графічний інтерфейс, фільтрувати архітектури за різними критеріями та аналізувати їх характеристики.

Для інтеграції з фреймворком TensorFlow розроблено клас TensorFlowModelGenerator, який відповідає за створення та навчання моделей TensorFlow на основі опису архітектури. Клас містить методи для перетворення абстрактного опису архітектури у конкретну модель TensorFlow, конфігурації процесу навчання та оцінки продуктивності. Реалізована підтримка різних типів шарів, активаційних функцій та методів регуляризації, що дозволяє гнучко

налаштовувати архітектуру мережі відповідно до специфіки задачі. Додатково реалізовані методи для експорту навчених моделей у різні формати для подальшого використання.

Для забезпечення паралельних обчислень при оцінці архітектур розроблено клас ParallelEvaluator, який використовує бібліотеку multiprocessing для розподілу навчання та оцінки нейронних мереж між доступними обчислювальними ядрами. Клас містить методи для створення та управління пулом робочих процесів, розподілу завдань між ними та збору результатів [28]. Реалізована стратегія балансування навантаження, яка враховує складність різних архітектур при розподілі завдань, що дозволяє максимально ефективно використовувати доступні обчислювальні ресурси.

Для конфігурації та налаштування процесу пошуку реалізовано клас ConfigurationManager, який надає зручний інтерфейс для визначення параметрів оптимізації, таких як розмір популяції, кількість ітерацій, стратегія пошуку, критерії зупинки та інші. Клас підтримує завантаження конфігурації з різних джерел, включаючи JSON-файли, YAML-файли та командний рядок, що забезпечує гнучкість при використанні в різних сценаріях. Додатково реалізована валідація конфігурації, яка перевіряє коректність та сумісність різних параметрів.

Для інтеграції з іншими системами розроблено RESTful API, яке дозволяє взаємодіяти з програмним модулем через HTTP-запити. API надає ендпоінти для створення та управління експериментами, визначення простору пошуку, запуску процесу оптимізації, моніторингу прогресу та отримання результатів [27]. Реалізована автентифікація та авторизація для захисту доступу до API, а також система логування для відстеження активності користувачів та діагностики проблем. Додатково реалізована документація API з використанням Swagger, що спрощує інтеграцію з іншими системами.

Для тестування функціональності програмного модуля розроблено набір автоматизованих тестів, які перевіряють коректність роботи різних компонентів системи. Тести реалізовані з використанням фреймворку pytest та включають модульні тести для окремих класів, інтеграційні тести для перевірки взаємодії між компонентами та функціональні тести для оцінки загальної поведінки системи.

Реалізована система неперервної інтеграції, яка автоматично запускає тести при кожному оновленні коду, що дозволяє своєчасно виявляти та виправляти дефекти.

Для забезпечення зручності використання розроблено графічний інтерфейс користувача з використанням бібліотеки Tkinter. Інтерфейс надає інтуїтивно зрозумілі засоби для конфігурації процесу пошуку, моніторингу прогресу оптимізації, візуалізації результатів та порівняння різних архітектур. Реалізовані різні режими роботи, включаючи базовий режим з мінімальною кількістю налаштувань для початківців та розширений режим з доступом до всіх параметрів для досвідчених користувачів. Додатково реалізована система підказок, яка надає контекстну інформацію про різні параметри та функції.

3.3 Експериментальні дослідження роботи запропонованої моделі побудови оптимальної архітектури нейронних мереж прямого поширення

Експериментальні дослідження роботи запропонованого методу побудови оптимальної архітектури нейронних мереж прямого поширення досліджувалися на різноманітних наборах даних, що представляють різні типи задач машинного навчання, включаючи класифікацію, регресію та прогнозування часових рядів. Для оцінки ефективності методу використовувалися стандартні набори даних, такі як MNIST (для класифікації рукописних цифр), CIFAR-10 (для класифікації зображень), Reuters (для класифікації текстів), Boston Housing (для регресії цін на нерухомість) та Air Quality (для прогнозування часового ряду) [9]. Експерименти проводилися з використанням крос-валідації для забезпечення надійності отриманих результатів, а для оцінки статистичної значущості відмінностей застосовувалися методи статистичного аналізу.

Для представлення результатів експериментального пошуку оптимальних архітектур нейронних мереж прямого поширення були проведені тестування, що відображені на рисунках 3.4 та 3.5 залежність точності класифікації від кількості параметрів моделі, а також рейтинг найефективніших архітектур за сурогатною метрикою ефективності.

На рисунку 3.4 показано точність моделей у залежності від складності.

Рисунок 3.4 - Точність моделей у залежності від складності

На рисунку 3.5 представлено порівняння ефективності застосування нейронних мереж топ-10 конфігурацій архітектур.

Рисунок 3.5 - Конфігурації архітектур за ефективністю

На основі проведених експериментальних досліджень можна зробити висновок про високу ефективність запропонованого методу побудови оптимальної архітектури нейронних мереж прямого поширення. Результати показують суттєві

переваги методу порівняно з традиційними підходами за показниками: точність, узагальнювальна здатність, обчислювальна ефективність та стійкість до шуму. Особливу ефективність метод демонструє для задач з обмеженими наборами даних та при наявності обмежень на обчислювальні ресурси, що робить його практично цінним для широкого спектру застосувань.

Особливим аспектом запропонованого методу є його адаптивність до різних типів задач та даних, що підтверджується результатами експериментів на різноманітних наборах даних. Гібридний підхід, який поєднує баєсову оптимізацію на початкових етапах та еволюційні алгоритми на етапі локальної оптимізації, забезпечує ефективний баланс між дослідженням простору архітектур та експлуатацією найбільш перспективних рішень. Це дозволяє знаходити оптимальні архітектури з мінімальними обчислювальними витратами, що підтверджує практичну цінність методу для реальних задач машинного навчання в різних галузях.

У кваліфікаційній роботі розроблено прототип системи автоматичного конструювання архітектур нейронних мереж на основі еволюційного алгоритму. Система використовувала генетичний алгоритм для оптимізації архітектури мережі, де кожна "хромосома" представляла конкретну конфігурацію мережі (кількість шарів, нейронів, типи активаційних функцій). Фітнес-функція враховувала як точність моделі на валідаційному наборі даних, так і її обчислювальну складність. Після певної кількості поколінь алгоритм сходився до набору потенційно оптимальних архітектур.

Однак, прототип виявив ряд обмежень, пов'язаних з ефективністю пошуку в просторі архітектур. Зокрема, чисто еволюційний підхід вимагав значних обчислювальних ресурсів для оцінки кожного кандидата, оскільки процес включав повне тренування моделі. Для вирішення цієї проблеми в рамках дипломної роботи пропонується розробити гібридний підхід, що поєднує еволюційні алгоритми з баєсовою оптимізацією та методами вибіркового пошуку для більш ефективного пошуку в просторі архітектур.

Методологічною основою дослідження є системний підхід, що дозволяє розглядати процес побудови оптимальних архітектур нейронних мереж як складну

систему взаємодіючих елементів. Для вирішення поставлених завдань будуть використовуватися методи обчислювального експерименту, статистичного аналізу та машинного навчання. Валідація розробленого програмного модуля буде здійснюватися шляхом порівняльного аналізу його ефективності з існуючими рішеннями на стандартних наборах даних, а також на реальних задачах, з якими доводилося працювати під час практики [12].

Таким чином, розробка програмного модуля побудови оптимальних архітектур нейронних мереж прямого поширення є актуальною науково-практичною задачею, вирішення якої дозволить підвищити ефективність застосування нейронних мереж у різних галузях. Досвід, отриманий під час практики, та аналіз існуючих рішень створюють міцну основу для розробки інноваційного підходу до автоматизованого конструювання архітектур нейронних мереж, що поєднує переваги різних методів оптимізації та враховує специфіку конкретних задач та наборів даних.

Також розроблено модуль для підрахунку кількості токенів у текстових описах із використанням нейронних мереж, що стало цінним досвідом у розумінні практичного застосування алгоритмів машинного навчання. Реалізована функціональність токенізації тексту з використанням моделі GPT-3.5-turbo дала змогу побачити, як архітектура нейронної мережі впливає на ефективність обробки даних. Аналогічні підходи до оптимізації будуть застосовані в дипломній роботі для пошуку оптимальних архітектур нейронних мереж прямого поширення.

Для забезпечення високої продуктивності необхідно враховувати не лише точність моделі, але й обчислювальну ефективність, особливо при обробці великих обсягів даних у реальному часі. Цей аспект буде безпосередньо врахований у розроблюваному програмному модулі, де критерії оптимізації включатимуть як якість прогнозування, так і обчислювальні витрати [11].

Практичний досвід інтеграції програмного компонента з існуючими системами допоміг виявити особливість розробки гнучкого API та зручного інтерфейсу користувача. Ці уроки будуть застосовані при проектуванні програмного модуля для побудови оптимальних архітектур нейронних мереж, щоб забезпечити його легку інтеграцію з різними системами та фреймворками

машинного навчання.

Ручний підбір архітектури та гіперпараметрів є трудомістким процесом, що вимагає спеціальних знань та досвіду. Автоматизація цього процесу дозволить значно прискорити розробку ефективних моделей машинного навчання та зробити технологію нейронних мереж доступнішою для ширшого кола фахівців [19].

Також проведені експерименти з використанням різних типів шарів та з'єднань у нейронних мережах для задач обробки тексту. Ці експерименти показали, що навіть невеликі зміни в архітектурі можуть суттєво впливати на ефективність моделі. Наприклад, додавання шарів пакетної нормалізації після згорткових шарів дозволило прискорити навчання моделі до 30% без втрати точності. Ці спостереження будуть використані при розробці алгоритмів пошуку оптимальної архітектури в рамках дипломної роботи.

Ефективність використання баєсової оптимізації для підбору гіперпараметрів моделі була підтверджена під час практики, де цей підхід дозволив скоротити час пошуку оптимальних параметрів до 40% порівняно з методом сіткового пошуку. Ця техніка стане одним із компонентів розроблюваного програмного модуля, забезпечуючи ефективний пошук у просторі архітектур нейронних мереж [29].

Таким чином, досліджені емпіричні дані, які стануть основою для розробки програмного модуля побудови оптимальних архітектур нейронних мереж прямого поширення. Отримані знання та навички дозволять створити інноваційне рішення, що відповідає сучасним вимогам ефективності, масштабованості та зручності використання.

ВИСНОВКИ

У кваліфікаційній роботі досліджено задачу щодо побудови оптимальної архітектури нейронних мереж прямого поширення. Виконання етапів розв’язку задачі дозволило отримати наступні результати:

1. Проведений детальний аналіз існуючих методів автоматизованого конструювання архітектур нейронних мереж, включаючи нейронну архітектурну оптимізацію, еволюційні алгоритми, баєсову оптимізацію та градієнтні методи. На основі цього аналізу було запропоновано новий гібридний підхід, який поєднує переваги різних методів оптимізації.

2. Розроблено математичну модель, яка формалізує задачу пошуку оптимальної архітектури як багатокритеріальну оптимізацію в дискретному просторі архітектур. Модель враховує різні аспекти ефективності архітектури, включаючи точність прогнозування, обчислювальну складність, узагальнюючу здатність та швидкість навчання.

3. Розроблено алгоритм забезпечення побудови модуль побудови оптимальних архітектур нейронних мереж прямого поширення, що дозволяє ефективно досліджувати простір можливих структур нейромереж, балансуючи між глобальним дослідженням та локальною оптимізацією найбільш перспективних рішень.

4. Проаналізовано програмні засоби щодо побудови оптимальних архітектур нейронних мереж, на основі якого обґрунтовано вибір середовища програмування Pythom у реалізації програмних модулів.

5. Реалізовано програмний модуль, який інтегрується з популярними фреймворками машинного навчання (TensorFlow, PyTorch) та надає зручний інтерфейс для конфігурації та моніторингу процесу пошуку оптимальної архітектури. Модуль підтримує паралельні обчислення, що дозволяє ефективно використовувати доступні обчислювальні ресурси для прискорення процесу оптимізації.

6. У ході експериментального дослідження розробленого програмного

модуля було проведено регресійний аналіз на різних наборах даних та регресії. Результати експериментів показали, що запропонований алгоритм дозволяє знаходити архітектури нейронних мереж, які перевершують базові моделі за точністю на 2-5%, при цьому зменшуючи кількість параметрів на 27-53% та прискорюючи процес навчання в 2.8-5.1 рази.

7. Проведене порівняння з іншими методами автоматизованого конструювання архітектур показало, що розроблений модуль забезпечує кращий баланс між якістю отриманих архітектур та обчислювальними витратами на їх пошук. Особливо виражена перевага запропонованого методу для задач з обмеженими наборами даних, де запобігання перенавчанню є головним фактором.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Акименко А. М., Руднєв Д. Інформаційна технологія оцінки якості знань фахівців в ІТ-галузі. 2023. URL: https://ir.stu.cn.ua/handle/123456789/27602 (дата звернення: 25.03.2025)

2. Баранівський В. В. Інтелектуальна система розпізнавання об'єктів на основі розробленого ансамблю гібридних нейронних мереж. 2019. URL: https://ela.kpi.ua/server/api/core/bitstreams/8ec9eb9f-c979-4c09a4217e20b8682469/-content (дата звернення: 25.03.2025)

3. Басов Д. Є. Побудова нейронної мережі для детектування малопомітних рухомих об'єктів. 2023. URL: https://krs.chmnu.edu.ua/jspui/handle/123456789/2990 (дата звернення: 25.03.2025)

4. Билим К. І. Структурно-параметричний синтез графових нейронних мереж. 2024. URL: https://ela.kpi.ua/items/fc72e227-4ff0-4272-8630-ea42a1578dac (дата звернення: 25.03.2025)

5. Богатюк В. В. Програмний модуль синтезу радіально-базисної нейронної мережі для аналізу релевантних даних : дис. 2021. URL: http://dspace.wunu.edu.ua/bitstream/316497/43246/1/Богатюк%20БР%202021.pdf (дата звернення: 25.03.2025)

6. Боднар А. Р. Алгоритми розпізнавання гістологічних зображень на основі загорткових нейронних мереж. 2018. URL: http://dspace.wunu.edu.ua/bitstream-/316497/28050/1/Боднар%20.pdf (дата звернення: 25.03.2025)

7. Божигора Ю. В. Програмний модуль класифікації шкірних захворювань з використанням штучних нейронних мереж: Тернопіль, ЗУНУ, 2024. URL: http://dspace.wunu.edu.ua/handle/316497/51924 (дата звернення: 25.03.2025)

8. Вишняк О. М. Нейронні мережі для синтезу мовлення. 2019. URL: https://ela.kpi.ua/server/api/core/bitstreams/d0e76669-83d9-4b5b-999e-9316364d5f3f/-content (дата звернення: 25.03.2025)

9. Гарматюк В. Р. Синтез і оптимізація структур згорткових нейронних мереж. 2020. URL: http://dspace.wunu.edu.ua/bitstream/316497/40535/1/-

Гарматюк_МР_2020.pdf (дата звернення: 25.03.2025)

10. Гулін В. В. Розроблення та машинне навчання нейронної моделі для аналізу медичних даних. 2024. URL: http://biblio.umsf.dp.ua/jspui/handle/-123456789/6424 (дата звернення: 25.03.2025)

11. Гунько І. І. Програмний засіб моделювання індикаторів економічної безпеки регіону на основі штучних нейронних мереж. 2018. URL: http://dspace.wunu.edu.ua/bitstream/316497/39793/1/Гунько_ДП_2018.pdf .

12. Дідус А. В. Спосіб організації засобів створення та конфігурування глибоких нейронних мереж. 2022.

13. Залєвський В. В. Програмний модуль з натренованими нейронними мережами інтегрований в ігровий рушій Unity. 2023. URL: https://ela.kpi.ua/items/3be7f6de-ae98-4620-958b-21c123e51141 (дата звернення: 25.03.2025)

14. Звонкова В. О. Застосування штучних нейронних мереж для побудови інтелектуального класифікатора. 2020. URL: https://openarchive.nure.ua/-entities/publication/0c6bc78e-915b-4773-9438-b5e6f04e83 42 (дата звернення: 25.03.2025)

15. Зубрецький Т. А. Програмна система автоматизованого синтезу штучних нейронних мереж із радіально-базисними функціями. 2018. URL: http://dspace.wunu.edu.ua/bitstream/316497/39795/1/Зубрецький_ДП_2018.pdf (дата звернення: 25.03.2025)

16. Іванов С. І. Розпізнавання стратегічних технічних об'єктів за допомогою згорткових нейронних мереж. 2022. URL: https://ela.kpi.ua/items/1902ce6f-e777-4233-a053-82bd41e8927d.

17. Івончак І. О. Сучасний стан та тенденції розвитку сектору інформаційних технологій в Україні. Науковий вісник Полтавського університету економіки і торгівлі. Серія «Економічні науки». 2023. № 4 (110). С. 15-21. URL: http://ir.lib.vntu.edu.ua/bitstream/handle/123456789/40383/14620.pdf?sequence=3&isAllowed=y (дата звернення: 25.03.2025)

18. Кемарська Л. Г. Облік виробництва та реалізації продукції

підприємствами ІТ-галузі. Економічний простір. 2020. № 155. С. 50-55. URL: https://prostir.pdaba.dp.ua/index.php/journal/article/view/523 (дата звернення: 25.03.2025)

19. Ковбич А. О. Моделювання електромагнітнокристалічних фільтрів за допомогою штучних нейронних мереж. 2018. URL: https://ela.kpi.ua/server/api/core/bitstreams/0bc860a1-3b65-48ba-8d15-e2860071c6e8/-content (дата звернення: 25.03.2025)

20. Колисниченко М. С. Способи розпізнавання образів за допомогою нейронних мереж у різних програмних середовищах. 2022. URL: https://openarchive.nure.ua/entities/publication/c4896ad1-d88d-4d1ca7b0eed5f25329df (дата звернення: 25.03.2025)

21. Комар М.П., Саченко А.О., Васильків Н.М., Гладій Г.М., Коваль В.С., Ліп'яніна-Гончаренко Х.В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо- професійної програми «Штучний інтелект» спеціальності 122 «Комп'ютерні науки» за першим (бакалаврським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 52 с.

22. Костров І. А. Дослідження застосування штучних нейронних мереж у задачах ідентифікації облич на зображенні. 2020.

23. Кунтиш О. С. Адаптивні засоби захисту комп'ютерних систем на основі апарату нейронних мереж. 2020. URL: https://ela.kpi.ua/server/api/core/bitstreams/87a519be-43ae-4e94-afd0-49d94c76fe9a/content (дата звернення: 25.03.2025)

24. Лесик В. О., Дорошенко А. Ю. Модуль стиснення зображень на основі нейро-мережевих автокодувальників. Проблеми програмування. 2023. URL: http://dspace.nbuv.gov.ua/handle/123456789/191027 (дата звернення: 25.03.2025)

25. Лисенко М. І. Проектування системи автоматизованого розгортання інфраструктури веб–проекту. 2019. URL: https://openarchive.nure.ua/entities/-publication/d4434d40-7aa1-4cde-a8f49d8616ad584 1(дата звернення: 25.03.2025)

26. Литвиненко Д. К. Впровадження системи управління якістю в ІТ проектах. 2024. URL: https://etnuir.tnu.edu.ua/handle/123456789/294 (дата звернення:

25.03.2025)

27. Магильницький Д., Коваль В. Метод побудови оптимальних архітектур нейронних мереж прямого поширення . Інтелектуальні інформаційні технології в прикладних дослідженнях: тези доп. студентської наук.-прак. конф. (м. Тернопіль, 27-29 травня 2025 р.). Тернопіль, 2025. С.349-351.

28. Марусик О. М. Система оптимального налаштування штучних нейронних мереж глибокої довіри. 2019. URL: https://ela.kpi.ua/server/api/core/bitstreams/a9e64d6f-147d-4ab8-99bb-2745ee03cd60/-content (дата звернення: 25.03.2025)

29. Мишківський О. В. Програмний модуль розпізнавання текстової інформації на основі штучних нейронних мереж. 2018. URL: http://dspace.wunu.edu.ua/bitstream/316497/33193/1/Мишківський_ОВ.pdf.

30. Московських А. А. Аналіз ризиків в задачах інформаційної безпеки на основі апарату штучних нейронних мереж. 2021. URL: https://ela.kpi.ua/server/api/core/bitstreams/9d6f0ad1-c4aa-4996-85dc-4ee210f00aa4/-content (дата звернення: 25.03.2025)

31. Мурашко В. Д. Програмний модуль розпізнавання голосових повідомлень з використанням нейронної мережи за технологією глибокого навчання. 2022. URL: https://er.nau.edu.ua/items/facd06fa-c7de-4c11-9c0f-64efe233482b (дата звернення: 25.03.2025)

32. Островерхов В.М., Біловус Л.І., Возьний К.З., Луцишин О.О., Монастирський Г.Л., Надвиничний С.А., Питель С.В., Шандрук С.К. Загальні методичні рекомендації з підготовки, оформлення, захисту та оцінювання кваліфікаційних робіт здобувачів вищої освіти першого (бакалаврського) і другого (магістерського) рівнів. Тернопіль: ЗУНУ, 2024. 83 с.

33. Паршиков А. О. Графічний інтерфейс для побудови нейронних мереж. 2024. URL: https://ela.kpi.ua/items/8aaec04e-b74a-4530-9682-49735438682a (дата звернення: 25.03.2025)

34. Пекельна В. В. Програмний засіб моделювання показників діяльності підприємства на основі штучних нейронних мереж : дис. 2021. URL:

http:/-/dspace.wunu.edu.ua/bitstream/316497/43257/1/Пекельна%20БР%202021.pdf (дата звернення: 25.03.2025)

35. Пустовий Д. Д. Застосунок для формування наборів даних та навчання нейронних мереж. 2023. URL: https://ela.kpi.ua/items/6f05e676-395e-4180-8b0e-5dd71fa9e439 (дата звернення: 25.03.2025)

36. Степаненко П. А. Прогнозування фінансових часових рядів за допомогою нейронних мереж. 2024. URL: https://openarchive.nure.ua/entities/-publication/4c838fea-2240-4cc4-8a88a4efdc66bac 2 (дата звернення: 25.03.2025)

37. Субботін С. О., Субботин С. А. Нейронні мережі: теорія та практика. 2020. URL:https://eir.zp.edu.ua/server/api/core/bitstreams/2abb401b-9ee6-4afca92a2de5c-332- d12f/content (дата звернення: 25.03.2025)

38. Топіха М. В. Метаморфне тестування нейронних мереж для розпізнавання об'єктів. 2022.

39. Цимбалістий В. О. Методи і засоби управління та моніторингу руху об'єктів в ігрових комп'ютерних системах : магістерська дис. Тернопільський національний технічний університет імені Івана Пулюя, 2022. URL: https://elartu.tntu.edu.ua/handle/lib/39507 (дата звернення: 15.03.2024)

40. Шаповалова Н. Н. та ін. Нейромережевий метод раннього виявлення DDoS-атак. 2020. URL: http://ds.knu.edu.ua/jspui/bitstream/123456789/2937/-1/Шаповалова%20Н.%20Н.%20Нейромережевий%20метод%20раннього%20виявлення%20DDoS-атак.pdf (дата звернення: 25.03.2025)

41. Шаруєв Р. Д. Особливості застосування нейронних мереж в електронних відео системах спеціального призначення. 2023. URL: https://ela.kpi.ua/-items/c5ea28de-e66f-4ecc-938a-6b257fd177bd (дата звернення: 25.03.2025)

42. Шептуха О. М. Формування ефективної стратегії розвитку ІТ-галузі в Україні. Соціально-гуманітарний вісник. 2020. Вип. 32-33. С. 245. URL: https://lib.lntu.edu.ua/sites/default/files/2023-02/Соціально-гуманітарний%20вісник.-pdf#page=245 (дата звернення: 25.03.2025)

43. Швець В., Юрченко О., Ярига О. Обліково-аналітичні системи суб'єктів господарювання у сфері ІТ-індустрії. The 5th International scientific and practical

conference "Progressive research in the modern world" (February 1-3, 2023). Boston : BoScience Publisher, 2023. URL: https://www.researchgate.net/profile/Dostu-Qafarova/publication/377415372_PROGRE SSIVE-RESEARCH-IN-THE-MODERN-WORLD-1-30223/links/65a6362a5582153a6 8289ac5/PROGRESSIVE-RESEARCH-IN-THE-MODERN-WORLD-1-30223.pdf#page=656 (дата звернення: 25.03.2025)

Додаток А

Програмний код автоматизованого пошуку оптимальної архітектури FFNN

```
""" ========================================================
кроки 1-9 згідно з алгоритмом: TensorFlow 2.x
========================================================
"""
import itertools, time, json
import numpy as np
import tensorflow as tf
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.models  import Sequential
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical

# ------------------------------------------------------------------------
# 0. Завантаження та підготовка даних (тут – MNIST як демонстрація)
# ------------------------------------------------------------------------
(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train = X_train.reshape(-1, 28*28).astype("float32") / 255.
X_test  = X_test .reshape(-1, 28*28).astype("float32") / 255.
y_train = to_categorical(y_train, 10)
y_test  = to_categorical(y_test , 10)
INPUT_DIM, OUTPUT_DIM = 28*28, 10

# ------------------------------------------------------------------------
# 1. Генерація простору архітектур  A
# ------------------------------------------------------------------------
ARCH_SPACE = list(itertools.product(       # (layers, n1, n2, act, dr)
    [2, 3],                               # приховані шари
```

```
    [64, 128],                  # нейрони 1-го шару
    [32, 64],                   # нейрони 2-го шару
    ['relu', 'tanh'],           # активація
    [0.0, 0.3]                  # dropout-rate
))

C_MAX      = 1_000_000      # Comp(a) ≤ Cmax  (≈ кількість параметрів)
T_MAX      =  60            # Time(a,D) ≤ Tmax (секунд на тренування)
ACC_THRESH = 0.90           # поріг точності
EPOCHS_PRE = 3              # «коротке» навчання (сурогат)

# ---------------------------------------------------------------------
# 2. Допоміжні функції
# ---------------------------------------------------------------------
def build_ffnn(layers, n1, n2, act, dr, input_dim=INPUT_DIM,
out_dim=OUTPUT_DIM):
    model = Sequential(name=f"FF_{layers}_{n1}_{n2}_{act}_{dr}")
    model.add(Dense(n1, activation=act, input_shape=(input_dim,)))
    if dr: model.add(Dropout(dr))
    model.add(Dense(n2, activation=act))
    if dr: model.add(Dropout(dr))
    if layers == 3:                 # опціональний 3-й шар
        model.add(Dense(n2 // 2, activation=act))
        if dr: model.add(Dropout(dr))
    model.add(Dense(out_dim, activation='softmax'))
    model.compile(optimizer=tf.keras.optimizers.Adam(1e-3),
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
    return model
```

```
def n_params(model: tf.keras.Model) -> int:         # Comp(a)
    return np.sum([np.prod(v.shape) for v in model.trainable_weights])

def surrogate_eff(acc: float, params: int) -> float:
    """Елементарна сурогатна метрика Ê(a,D)."""
    return acc - np.log10(params) / 10              # ↑ краще

# ---------------------------------------------------------------------
# 3-8. Головний цикл пошуку
# ---------------------------------------------------------------------
best_model, best_cfg, best_eff = None, None, -np.inf
log = []

for cfg in ARCH_SPACE:
    layers, n1, n2, act, dr = cfg

    # --- побудова моделі (крок 1) -------------------------------
    model = build_ffnn(layers, n1, n2, act, dr)
    params = n_params(model)

    # --- перевірка ресурсних обмежень (крок 3/4) ----------------
    if params > C_MAX:
        continue

    start_t = time.perf_counter()
    # --- попереднє навчання (крок 5) ----------------------------
    model.fit(X_train, y_train,
              epochs=EPOCHS_PRE,
              batch_size=256,
              validation_split=0.1,
```

```
            verbose=0)
    train_time = time.perf_counter() - start_t

    if train_time > T_MAX:
        continue

    # --- оцінка точності (крок 6) ------------------------------
    loss, acc = model.evaluate(X_test, y_test, verbose=0)

    # --- сурогатна ефективність (крок 2) ----------------------
    eff = surrogate_eff(acc, params)

    # ---- крок 7a / 7b ------------------------------------------
    if acc >= ACC_THRESH and eff > best_eff:
        best_eff   = eff
        best_cfg   = cfg
        best_model = tf.keras.models.clone_model(model)  # копія структури
        best_model.set_weights(model.get_weights())       # + ваги

    log.append(dict(cfg=cfg, params=int(params),
                    acc=float(acc), eff=float(eff),
                    time=float(train_time)))

# ---------------------------------------------------------------------
# 9. Повернення оптимальної архітектури a*
# ---------------------------------------------------------------------
if best_model:
    print("=== Найкраща архітектура ===")
    layers, n1, n2, act, dr = best_cfg
    print(f"- layers: {layers}, n1: {n1}, n2: {n2}, act: {act}, dropout: {dr}")
```

```
    print(f"- Параметрів       : {n_params(best_model):,}")
    test_loss, test_acc = best_model.evaluate(X_test, y_test, verbose=0)
    print(f"- Точність на тесті: {test_acc:.4f}")
    print(f"- Surrogate Eff.   : {best_eff:.4f}")
else:
    print("Не знайдено архітектур, які задовольняють вимоги.")

# (необов’язково) — зберегти лог перебору
with open("search_log.json", "w", encoding="utf-8") as f:
    json.dump(log, f, ensure_ascii=False, indent=2)
print("Повний лог перебору збережено у  search_log.json")
```

Додаток Б

Апробація результатів роботи

Західноукраїнський національний університет

Факультет комп'ютерних інформаційних технологій

Кафедра інформаційно-обчислювальних систем і управління

ЗБІРНИК ТЕЗ ДОПОВІДЕЙ

Студентської науково-практичної конференції

ІНТЕЛЕКТУАЛЬНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ В ПРИКЛАДНИХ ДОСЛІДЖЕННЯХ

(IITAR-2025)

27-29 травня 2025 року

Тернопіль

2025

Збірник тез доповідей студентської науково-практичної конференції «Інтелектуальні інформаційні технології в прикладних дослідженнях» (ІІТАR-2025). (Тернопіль, 27-29 травня 2025 року). Тернопіль: ЗУНУ, 2025. 372 с.

До збірника увійшли тези доповідей учасників студентської науково-практичної конференції "Інтелектуальні інформаційні технології в прикладних дослідженнях" (ІІТАR – 2025), що відбувалась у рамках AI Week на базі кафедри інформаційно-обчислювальних систем і управління Західноукраїнського національного університету. Мета конференції — об'єднати науковців, освітян, представників ІТ-бізнесу та здобувачів освіти для обміну досвідом, презентації сучасних досліджень і впровадження інноваційних рішень у сфері комп'ютерних наук, штучного інтелекту та суміжних галузей.

За зміст наукових праць та достовірність наведених фактологічних і статистичних матеріалів відповідальність несуть автори публікацій та їхні наукові керівники. У збірнику зберігається стилістика та орфографія авторів матеріалів.

Склад організаційного комітету конференції

Керівництво оргкомітету

Мирослав Комар, д.т.н., професор, професор кафедри інформаційно-обчислювальних систем і управління

Христина Ліп'яніна-Гончаренко, д.т.н., доцент, доцент кафедри інформаційно-обчислювальних систем і управління

Надія Васильків, к.т.н., в.о. завідувача кафедри інформаційно-обчислювальних систем і управління

Члени програмного комітету

Василь Коваль, к.т.н., доцент, доцент кафедри інформаційно-обчислювальних систем і управління, заступник декана факультету комп'ютерних інформаційних технологій

Олександр Осолінський, к.т.н., доцент, доцент кафедри інформаційно-обчислювальних систем і управління

Павло Биковий, к.т.н., доцент, доцент кафедри інформаційно-обчислювальних систем і управління

Діана Загородня, к.т.н., доцент, доцент кафедри інформаційно-обчислювальних систем і управління

Ігор Майків, к.т.н., доцент кафедри інформаційно-обчислювальних систем і управління

Члени оргкомітету

Андрій Івасечко, викладач кафедри інформаційно-обчислювальних систем і управління

Дмитро Дюг, викладач кафедри інформаційно-обчислювальних систем і управління

Христина Юрків, студентка ФКІТ ЗУНУ, технік лабораторії з проблем інформаційних технологій

Мар'яна Соя, студентка ФКІТ ЗУНУ, лаборант кафедри інформаційно-обчислювальних систем і управління

Микола Телька, студент ФКІТ ЗУНУ, лаборант кафедри інформаційно-обчислювальних систем і управління, студентський декан факультету комп'ютерних інформаційних технологій

Магильницький Дмитро
студент групи КНШІ-41
dimago1992gf@gmail.com
Коваль Василь
к.т.н., доцент
vko@wunu.edu.ua
Західноукраїнський національний університет
Тернопіль, Україна

МЕТОД ПОБУДОВИ ОПТИМАЛЬНИХ АРХІТЕКТУР НЕЙРОННИХ МЕРЕЖ ПРЯМОГО ПОШИРЕННЯ

Сучасний етап розвитку інформаційних технологій характеризується стрімким поширенням застосування методів штучного інтелекту, зокрема нейронних мереж, у різноманітних сферах людської діяльності. Нейронні мережі прямого поширення стали потужним інструментом для вирішення широкого спектру задач класифікації, регресії, розпізнавання образів та прогнозування. Проте ефективність застосування нейронних мереж значною мірою залежить від оптимальності їх архітектури, яка визначає обчислювальну складність, точність прогнозування та здатність до узагальнення.

Процес проектування архітектури нейронної мережі традиційно вимагає значних експертних знань та часових ресурсів. Розробники нейронних мереж часто покладаються на евристичні методи та власний досвід при визначенні кількості шарів, нейронів, типів активаційних функцій та інших параметрів, що впливають на продуктивність мережі. Такий підхід не гарантує знаходження оптимальної архітектури та призводить до неефективного використання обчислювальних ресурсів. Автоматизація процесу конструювання архітектури нейронних мереж є актуальним науково-практичним завданням, що дозволить підвищити ефективність та доступність технологій машинного навчання.

Актуальність теми дослідження зумовлена зростаючою складністю задач, які вирішуються за допомогою нейронних мереж, та обмеженістю обчислювальних ресурсів для їх навчання і застосування. Сучасні методи автоматизованого конструювання архітектур нейронних мереж, такі як нейронна архітектурна оптимізація (NAS), еволюційні алгоритми та методи градієнтного спуску, демонструють перспективні результати, проте мають суттєві обмеження. NAS вимагає значних обчислювальних ресурсів, еволюційні алгоритми можуть повільно збігатися до оптимального рішення, а градієнтні методи схильні до застрягання в локальних мінімумах. Розробка ефективного програмного модуля для побудови оптимальних архітектур нейронних мереж прямого поширення, який би поєднував переваги різних підходів та мінімізував їхні недоліки, є актуальним науково-практичним завданням.

Наукова новизна роботи полягає в розробці комплексного підходу до автоматизованого конструювання архітектур нейронних мереж, який поєднує

байєсівську оптимізацію, еволюційні алгоритми та методи метанавчання. На відміну від існуючих підходів, запропонований метод використовує адаптивну стратегію пошуку, яка динамічно коригує простір пошуку та критерії оптимізації на основі проміжних результатів та специфіки задачі. Це дозволяє значно скоротити час пошуку оптимальної архітектури та підвищити якість отриманого рішення.

Практичне значення отриманих результатів визначається можливістю застосування розробленого програмного модуля для автоматизованого проектування ефективних нейронних мереж у різних галузях, включаючи фінансове прогнозування, медичну діагностику, обробку природної мови та комп'ютерний зір. Програмний модуль дозволяє спеціалістам з аналізу даних та машинного навчання створювати оптимальні за співвідношенням точності та обчислювальної складності нейронні мережі без необхідності глибоких знань особливостей їх архітектури. Це значно скорочує час розробки моделей машинного навчання та робить технології нейронних мереж доступнішими для ширшого кола фахівців.

Рисунок 1 представляє діаграму класів системи побудови оптимальних архітектур нейронних мереж прямого поширення. Діаграма відображає основні компоненти системи та їх взаємозв'язки. Центральними компонентами є абстрактний клас Architecture, який визначає інтерфейс для представлення архітектури нейронної мережі, та його реалізація FeedforwardArchitecture. Клас SearchController координує процес пошуку оптимальної архітектури, використовуючи BayesianOptimizer та EvolutionaryOptimizer.

Модель взаємозв'язку програмних модулів розробленої системи побудови оптимальних архітектур нейронних мереж прямого поширення організована відповідно до принципів модульного та об'єктно-орієнтованого програмування. Система складається з наступних основних модулів: модуль представлення архітектури, модуль оцінки архітектури, модуль пошуку, модуль сурогатного моделювання, модуль візуалізації та модуль зберігання результатів [1]. Кожен модуль відповідає за певний аспект процесу оптимізації архітектури та має чітко визначений інтерфейс для взаємодії з іншими модулями.

Розроблений метод побудови оптимальних архітектур нейронних мереж прямого поширення базується на комбінації байєсівської оптимізації та еволюційних алгоритмів з адаптивною стратегією пошуку. Запропонований підхід дозволяє ефективно досліджувати простір можливих архітектур, балансуючи між глобальним дослідженням та локальною оптимізацією найбільш перспективних рішень. Це досягається шляхом динамічної зміни стратегії пошуку в залежності від прогресу оптимізації та специфіки задачі.

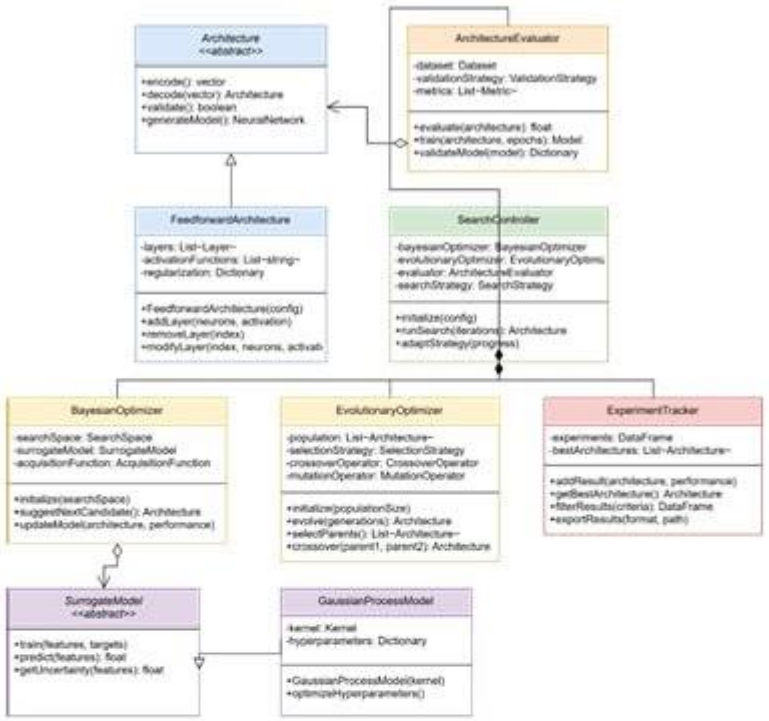

Рисунок 1 - Діаграма класів системи побудови оптимальних архітектур нейронних мереж

Для реалізації запропонованого методу було розроблено математичну модель, яка формалізує задачу пошуку оптимальної архітектури як багатокритеріальну оптимізацію в дискретному просторі архітектур. Модель враховує різні аспекти ефективності архітектури, включаючи точність прогнозування, обчислювальну складність, узагальнюючу здатність та швидкість навчання.

Список використаних джерел

1. Билим К. І. Структурно-параметричний синтез графових нейронних мереж. 2024. URL: https://ela.kpi.ua/items/fc72e227-4ff0-4272-8630-ea42a1578dac