Міністерство освіти і науки України
Луцький національний технічний університет
Факультет архітектури, будівництва та дизайну
Кафедра прикладної математики та механіки

КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «МАГІСТР»

АНАЛІЗ ТА РОЗРОБКА АЛГОРИТМІВ ДЛЯ ІНТЕРПРЕТОВАНИХ НЕЙРОННИХ МЕРЕЖ ІЗ ВИКОРИСТАННЯМ МЕТОДІВ ТЕОРІЇ ІНФОРМАЦІЇ
ANALYSIS AND DEVELOPMENT OF ALGORITHMS FOR INTERPRETED NEURAL NETWORKS USING INFORMATION THEORY METHODS

спеціальність 113 Прикладна математика
освітня програма Прикладна математика

Виконав: здобувач вищої освіти
Групи ПРМм-21
Вавринюк Володимир Васильович

_____
(підпис)

Керівник:
Д.т.н., професор
Делявський Михайло Володимирович

_____
(підпис)

Кваліфікаційну роботу
допущено до захисту
«___» _____ 20__ р.
PhD, доцент
Гарант освітньої програми:
Самоненко Інга Вікторівна

_____
(підпис)

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *архітектури, будівництва та дизайну*

Кафедра *прикладної математики та механіки*

Ступінь вищої освіти: *магістр*

Галузь знань: *11 Математика і статистика*

Спеціальність *113 Прикладна математика*

Освітня програма *Прикладна математика*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Мікуліч О.А.
«___» _____202__ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Вавринюк Володимир Васильович

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи
Аналіз та розробка алгоритмів для інтерпретованих нейронних мереж із використанням методів теорії інформації / Analysis and development of algorithms for interpreted neural networks using information theory methods
Керівник роботи: *Делявський Михайло Володимирович*
затверджені наказом закладу вищої освіти від «25» січня 2025 р. № 45/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи
«___»_____ 202_ р.
3. Вихідні дані до роботи *наукові публікації з питань теорії інформації та глибокого навчання; набори даних для навчання нейронних мереж MNIST та CIFAR-10; технічна документація бібліотек мови програмування Python; методичні вказівки до виконання кваліфікаційної роботи магістра.*
4. Зміст пояснювальної записки (перелік питань, що потрібно розробити):
Огляд літератури та аналіз проблематики за темою роботи, постановка завдань дослідження. Розробка алгоритмів інформаційно-теоретичного аналізу нейронних мереж. Експериментальна перевірка та аналіз результатів. Висновки
5. Перелік графічного (ілюстративного) матеріалу: *Концептуальна схема передачі та стиснення інформації в глибоких нейронних мережах. Блок-схема загальної архітектури запропонованого підходу. Приклад результату запуску модулю. Інформаційна площина результатів MNIST Інформаційна площина результатів CIFAR-10. Презентація.*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
1 розділ	*Делявський М.В., професор кафедри*		
2 розділ	*Делявський М.В., професор кафедри*		
3 розділ	*Делявський М.В., професор кафедри*		
Висновки	*Делявський М.В., професор кафедри*		

7. Дата видачі завдання «25» січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи магістра	Строк виконання етапів роботи	Примітка
1.	*Обґрунтування теми*	*до 25.01.2025*	
2.	*Огляд літератури із досліджуваної проблеми*	*до 03.06.2025*	
3.	*Перший розділ*	*до 11.09.2025*	
4.	*Другий розділ*	*до 17.10.2025*	
5.	*Третій розділ*	*до 20.11.2025*	
6.	*Висновки та пропозиції*	*до 25.11.2025*	
7.	*Формування списку використаних джерел*	*до 29.11.2025*	
8.	*Формування додатків*	*до 03.12.2025*	
9.	*Оформлення ілюстративного матеріалу*	*до 09.12.2025*	
0.	*Нормоконтроль*	*до 13.12.2025*	
11.	*Інструментальна перевірка на академічний плагіат*	*до 18.12.2025*	
12.	*Представлення кваліфікаційної роботи магістра до захисту*	*до 27.12.2025*	

Здобувач вищої освіти _____ (Вавринюк В.В.)
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ (Делявський М.В.)
(підпис) (прізвище, ініціали)

АНОТАЦІЯ

Вавринюк В. В. Аналіз та розробка алгоритмів для інтерпретованих нейронних мереж із використанням методів теорії інформації. Рукопис.

Кваліфікаційна робота магістра ОП «Прикладна математика» спеціальності 113 «Прикладна математика». – Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота магістра складається з вступу, трьох розділів, висновків і пропозицій, списку використаних джерел, додатків.

У роботі досліджено проблему інтерпретованості моделей глибокого навчання та застосування методів теорії інформації для аналізу їхньої внутрішньої динаміки. Обґрунтовано використання принципу information bottleneck для відстеження еволюції стиснення та передачі інформації між шарами. Розроблено та програмно реалізовано алгоритм оцінки взаємної інформації з використанням функції активації tanh для дискретизації, що забезпечує лінійну обчислювальну складність та чисельну стабільність. Створено програмний модуль мовою Python, який візуалізує траєкторії навчання на «інформаційній площині». Експериментальна перевірка проведена на наборах даних MNIST та CIFAR-10.

Ключові слова: прикладна математика, нейронні мережі, глибоке навчання, теорія інформації, information bottleneck, взаємна інформація, інтерпретованість, python.

ABSTRACT

Vavryniuk V. V. Analysis and development of algorithms for interpreted neural networks using information theory methods. Manuscript.

Master's thesis in the educational program «Applied Mathematics», specialty 113 «Applied Mathematics». – Lutsk National Technical University. Lutsk, 2025.

The master's thesis consists of an introduction, three chapters, conclusions and recommendations, a list of references, and appendices.

The thesis investigates the problem of interpretability of deep learning models and the application of information theory methods to analyze their internal dynamics. The use of the information bottleneck principle to track the evolution of compression and information transfer between layers is justified. An algorithm for evaluating mutual information using the tanh activation function for discretization has been developed and implemented in software, providing linear computational complexity and numerical stability. A software module in Python has been created that visualizes learning trajectories on the «information plane». Experimental verification was performed on the MNIST and CIFAR-10 datasets.

Keywords: applied mathematics, neural networks, deep learning, information theory, information bottleneck, mutual information, interpretability, Python.

ЗМІСТ

ВСТУП

В останні десятиліття нейронні мережі (НМ) продемонстрували виняткову ефективність у розв'язанні складних завдань у таких галузях, як комп'ютерний зір, обробка природної мови та розпізнавання образів. Однак, попри їхню високу точність, більшість сучасних архітектур НМ функціонують як «чорні скриньки», тобто логіка, яка лежить в основі їхніх рішень, залишається непрозорою та складною для людського розуміння.

Проблема браку інтерпретованості стає критичним бар'єром для впровадження систем штучного інтелекту у сферах, де ціна помилки є надзвичайно високою, а вимоги до надійності, безпеки та підзвітності є першочерговими. У медицині, фінансах, юриспруденції та управлінні критичною інфраструктурою важливо не лише отримати результат, а й розуміти, чому система дійшла саме такого висновку.

Метою магістерської роботи є підвищення рівня інтерпретованості моделей глибоких нейронних мереж шляхом розробки та аналізу алгоритмів, що базуються на методах теорії інформації.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести аналіз сучасних підходів до інтерпретованості нейронних мереж та існуючих методів застосування теорії інформації в глибокому навчанні;

– розробити алгоритм для кількісної оцінки інформаційних потоків між шарами нейронної мережі з метою ідентифікації ключових нейронів та зв'язків, що найбільше впливають на кінцевий результат;

– запропонувати метод візуалізації процесу прийняття рішень нейронною мережею, заснований на інформаційно-теоретичних метриках;

– здійснити програмну реалізацію розроблених алгоритмів у вигляді модуля, сумісного з популярними фреймворками глибокого навчання;

– провести експериментальну перевірку ефективності запропонованих алгоритмів на стандартних наборах даних MNIST та CIFAR-10.

Об'єкт дослідження – процес обробки інформації та прийняття рішень у глибоких нейронних мережах.

Предмет дослідження – алгоритми та методи аналізу, візуалізації та підвищення інтерпретованості нейронних мереж, що використовують математичний апарат теорії інформації.

Практична значущість роботи полягає в тому, що розроблені алгоритми та програмний модуль можуть бути використані для:

– створення більш прозорих та надійних систем штучного інтелекту в критичних галузях, таких як медична діагностика, фінансовий скоринг та безпілотний транспорт;

– налагодження та вдосконалення архітектур нейронних мереж шляхом виявлення надлишкових або малоінформативних компонентів;

– побудови довірчих систем, де кінцевий користувач може отримати зрозуміле пояснення рішення, прийнятого моделлю;

– використання в освітньому процесі як інструменту для поглибленого вивчення принципів роботи глибоких нейронних мереж.

Апробація результатів дослідження. Ключові аспекти розробленої архітектури програмного модуля та методів оптимізації оцінки взаємної інформації були представлені на IX Міжнародній студентській науковій конференції «Пріоритетні напрямки та вектори розвитку світової науки» м. Суми, 5 грудня 2025 р. [1].

У процесі підготовки магістерської кваліфікаційної роботи застосовувалися технології штучного інтелекту як допоміжний інструментарій. Зокрема, для стилістичної правки та структурування тексту використано ChatGPT-4o, а для технічної підтримки при написанні коду та візуалізації даних Google Colab AI. Автор несе повну відповідальність за зміст роботи: усі наукові положення та висновки є результатом самостійного дослідження, а згенеровані матеріали пройшли ретельну верифікацію на предмет точності та відсутності плагіату.

РОЗДІЛ 1
ОГЛЯД ЛІТЕРАТУРИ ТА АНАЛІЗ ПРОБЛЕМАТИКИ ЗА ТЕМОЮ РОБОТИ, ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1. Поняття інтерпретованості в штучних нейронних мережах

Інтерпретованість у контексті штучного інтелекту та машинного навчання – це здатність людини розуміти та пояснювати логіку прийняття рішень моделлю. В той час як точність моделі показує, наскільки добре вона виконує своє завдання, наприклад, класифікацію або регресію, інтерпретованість відповідає на питання «чому?» та «як?» модель дійшла певного висновку. Для глибоких нейронних мереж, які складаються з мільйонів параметрів, організованих у складні, багатошарові архітектури, це питання є особливо гострим. Їхня внутрішня робота часто нагадує «чорну скриньку». Ми можемо спостерігати вхідні дані та кінцевий результат, але проміжні кроки та причинно-наслідкові зв'язки залишаються прихованими та неінтуїтивними [2].

Важливість інтерпретованості зумовлена кількома взаємопов'язаними факторами. Перш за все це питання надійності та довіри, особливо у критично важливих сферах, таких як-от діагностика захворювань в медицині чи кредитний скоринг у фінансах, де необхідно бути впевненим, що модель спирається на релевантні та обґрунтовані ознаки, а не на випадкові кореляції в даних. Саме розуміння того, чому модель робить помилки, дозволяє цілеспрямовано її налагоджувати та вдосконалювати, покращуючи архітектуру, навчальні дані та/або процес тренування.

Ще одним фактором є відповідність нормативним вимогам, що набуває все більшого значення. Законодавства багатьох країн, зокрема «Загальний регламент про захист даних» в ЄС, передбачають забезпечення «права на пояснення» щодо рішень, ухвалених автоматизованими системами.

Нарешті, аналіз навчених моделей має також наукову цінність, оскільки може допомогти виявити нові, раніше невідомі закономірності в даних, сприяючи тим самим науковим відкриттям [2].

Таким чином, інтерпретованість перестає бути бажаною характеристикою і стає необхідною умовою для широкого та відповідального впровадження систем ШІ.

1.2. Основні підходи до досягнення інтерпретованості

Усі методи інтерпретації можна умовно поділити на дві великі категорії: пост-хок та вбудовані.

Пост-хок методи застосовуються до вже навченої моделі, не втручаючись у її внутрішню структуру, вони аналізують поведінку «чорної скриньки» ззовні. Розглянемо декілька найпопулярніших методів цієї групи.

LIME – Local Interpretable Model-agnostic Explanations. Даний метод пояснює окреме передбачення, апроксимуючи поведінку складної моделі f навколо цього передбачення простішою, інтерпретованою моделлю g, наприклад, лінійною регресією. Пояснення $\xi(x)$ для екземпляра x знаходиться шляхом оптимізації (1):

$$\xi(x) = argmin_{g \in G} L(f,g,\pi_x) + \Omega(g) \quad (1)$$

де G – клас інтерпретованих моделей, $L(f,g,\pi_x)$ – функція втрат, що вимірює, наскільки невірно модель g апроксимує f у локальній області π_x, а $\Omega(g)$ – штраф за складність моделі g [3].

Метод SHAP – SHapley Additive exPlanations, базується на теорії ігор, зокрема, на значеннях Шеплі, для справедливого розподілу «внеску» ϕ_i кожної вхідної ознаки у кінцевий результат. Пояснювальна модель має адитивну структуру (2):

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z_j' \quad (2)$$

де $g(z')$ – пояснення для спрощеного входу z', M – кількість ознак, а ϕ_i – значення SHAP для ознаки i. SHAP надає як локальні, так і глобальні пояснення [4].

Grad-CAM – Gradient-weighted Class Activation Mapping. Цей метод, що використовується переважно для згорткових нейронних мереж, аналізує

градієнти, що надходять до останнього згорткового шару, щоб створити теплову карту. Спочатку обчислюються ваги a_k^c для кожного каналу k карти активацій A^k відносно класу c (3):

$$a_k^c = \frac{1}{Z}\sum_i \quad \sum_j \quad \frac{\partial y^c}{\partial A_{ij}^k} \tag{3}$$

де y^c – вихід моделі для класу c, а Z – кількість пікселів у карті активацій. Теплова карта $L_{Grad-CAM}^c$ є лінійною комбінацією карт активацій з подальшим застосуванням функції ReLU (4):

$$L_{Grad-CAM}^c = ReLU(\sum_k \quad a_k^c A^k) \tag{4}$$

Це показує, які області вхідного зображення були найважливішими для класифікації [5].

Вбудовані методи передбачають створення моделей, які є прозорими за своєю суттю. В них інтерпретованість закладається в саму архітектуру або процес навчання, прикладами є прозорі моделі, механізми уваги та навчання з обмеженнями.

До прозорих моделей належать лінійні моделі, дерева рішень та k-найближчих сусідів. Наприклад, лінійна модель робить передбачення $\hat{y}$ на основі лінійної комбінації вхідних ознак x_i з вагами w_i (5):

$$\hat{y} = w_0 + \sum_{i=1}^{n} \quad w_i x_i \tag{5}$$

Їхня перевага в простоті, але вони часто поступаються в точності складним моделям [6].

Механізми уваги вбудовуються в архітектуру НМ і дозволяють моделі динамічно зважувати важливість різних частин вхідних даних під час генерації виходу. Розрахунок уваги в найпоширенішому варіанті можна описати формулою (6):

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{6}$$

де Q (query), K (key), та V (value) – це векторні представлення вхідних даних, а d_k – розмірність векторів ключів.

Ваги уваги можна візуалізувати, щоб зрозуміти, на що «дивиться» модель.

Навчання з обмеженнями – це введення в функцію втрат L додаткових членів, регуляризаторів R(θ), які стимулюють модель до більш інтерпретованої поведінки, наприклад, шляхом заохочення розрідженості активацій. Загальна функція втрат має вигляд (7):

$$L_{total}(\theta) = L_{task}(y, \hat{y}) + \lambda R(\theta) \quad (7)$$

де λ – гіперпараметр, що контролює силу регуляризації [7].

Вибір між цими підходами залежить від конкретного завдання, вимог до точності та рівня необхідної деталізації пояснень.

1.3. Методи теорії інформації в аналізі моделей машинного навчання

Теорія інформації, розроблена Клодом Шенноном у середині XX століття, надає математичний апарат для кількісного вимірювання інформації, невизначеності, та зв'язку між різними змінними. Цей інструментарій виявився надзвичайно корисним для аналізу процесів, що відбуваються всередині нейронних мереж.

Застосування теорії інформації дозволяє поглянути на процес навчання НМ як на процес передачі та стиснення інформації. Мережа отримує на вхід дані з високою ентропією і поступово, шар за шаром, перетворює їх, намагаючись зберегти якомога більше релевантної інформації про цільову змінну, наприклад, клас об'єкта, і водночас відкинути зайву, нерелевантну інформацію, таку як шум, варіації, що не впливають на результат (рис 1.1).

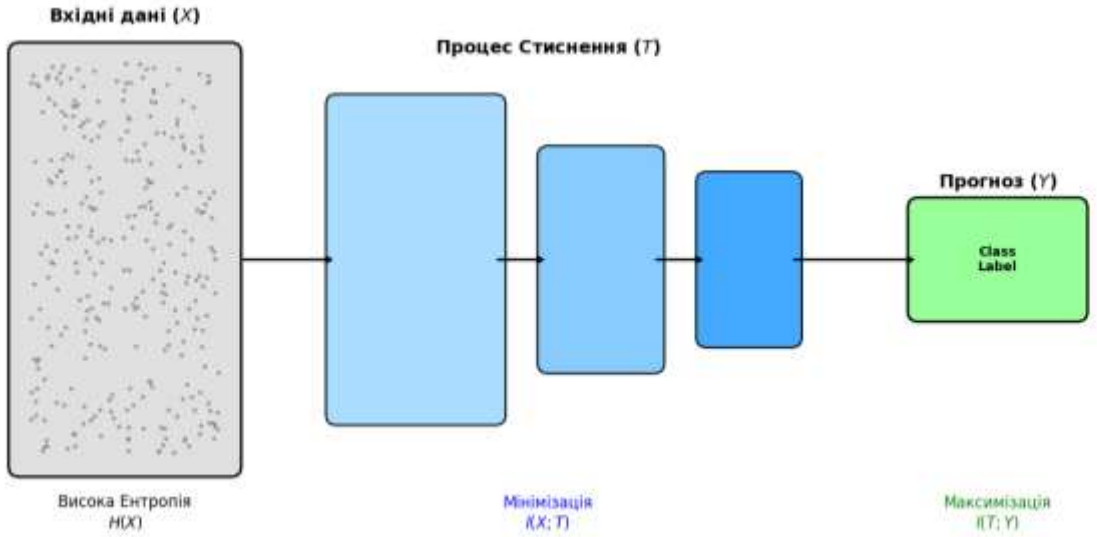

Рисунок 1.1 – концептуальна схема передачі та стиснення інформації в глибоких нейронних мережах

Такий погляд дозволяє аналізувати:

– інформаційні потоки, як інформація про вхідні дані та вихідні мітки поширюється через шари мережі;

– значущість компонентів, які нейрони, канали чи зв'язки передають найбільше інформації і є ключовими для функціонування моделі;

– процес узагальнення, як мережа досягає здатності до узагальнення, стискаючи вхідні дані та створюючи їх компактне, інформативне представлення [8].

1.4. Інформаційні міри: ентропія, взаємна інформація, інформаційне ущільнення

Ключовими поняттями теорії інформації, що використовуються для аналізу НМ, є: ентропія H(X), взаємна інформація I(X; Y) та принцип інформаційного ущільнення.

Ентропія H(X) – це міра невизначеності або «несподіваності», пов'язаної з випадковою змінною X, чим більш рівномірним є розподіл ймовірностей значень змінної, тим вища її ентропія.

Якщо результат експерименту є повністю детермінованим, невизначеність відсутня, і ентропія дорівнює нулю. Максимальна ж ентропія досягається тоді, коли всі можливі стани системи є рівноймовірними, що відповідає повній невизначеності результату. Клод Шеннон, засновник теорії інформації, запропонував математичний спосіб розрахунку цієї міри. Для дискретної випадкової змінної X, що може приймати значення $\{x_1, x_2, ..., x_n\}$ з імовірностями $P(x_i)$, ентропія Шеннона визначається як (8):

$$H(X) = - \Sigma_i P(x_i) \log_2(P(x_i)) \quad (8)$$

В контексті НМ ентропія може характеризувати різноманітність активацій нейронів.

Взаємна інформація I(X; Y) є мірою статистичної залежності між двома випадковими змінними X та Y. Вона показує, скільки інформації про одну змінну ми отримуємо, спостерігаючи за іншою.

Математично вона виражається через ентропію (9):

$$I(X; Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) \quad (9)$$

де H(X|Y) – це умовна ентропія, що вимірює невизначеність X за умови, що Y відома. Наприклад, можна виміряти взаємну інформацію між вхідними даними та активаціями прихованого шару, або між активаціями прихованого шару та вихідними мітками [10].

Принцип інформаційного ущільнення (information bottleneck, IB). Цей принцип постулює, що ідеальна модель машинного навчання повинна діяти як «інформаційне вузьке місце». Вона має максимально стискати інформацію про вхідні дані X, зберігаючи при цьому якомога більше інформації про цільову змінну Y.

Це можна сформулювати як задачу оптимізації з функціоналом:

$$L = I(X; T) - \beta * I(T; Y) \rightarrow \min \quad (10)$$

де T – це внутрішнє представлення даних, наприклад, активації шару, а β – параметр, що регулює баланс між стисненням – мінімізація I(X;T), та інформативністю – максимізація I(T;Y).

Формально, це означає мінімізацію взаємної інформації I(X; T) між входом X та внутрішнім представленням T, при одночасній максимізації I(T; Y) [11].

Ці метрики дозволяють кількісно оцінювати та оптимізувати інформаційні характеристики нейронних мереж, що відкриває шлях до побудови більш інтерпретованих моделей.

1.5. Існуючі роботи, що поєднують теорію інформації з інтерпретованим ШІ

Ідея застосування теорії інформації до нейронних мереж не є новою, але останніми роками вона переживає ренесанс, особливо в контексті аналізу динаміки навчання. Початкові впливові дослідження візуалізували процес навчання на «інформаційній площині», висунувши гіпотезу про те, що він складається з двох фаз: швидкої фази припасування та значно довшої фази стиснення, яку пов'язували з узагальненням. Сучасні дослідження продовжують розвивати цей напрям. Наприклад, робота «An Information Theoretic Interpretation to Deep Neural Networks» [11] пропонує розглядати глибоку мережу як каскад каналів передачі інформації. Автори аналізують, як кожен шар трансформує вхідні дані, та пропонують метрики для кількісної оцінки внеску кожного компонента мережі у фінальне рішення. Такий підхід дозволяє не просто оцінити модель, а й зрозуміти функціональну роль її окремих частин, що є кроком до більш глибокої інтерпретації.

Окрім такого цілісного погляду, інформаційно-теоретичні методи також застосовуються для більш гранулярного аналізу. Вони дозволяють декомпозувати функцію мережі, оцінюючи інформаційний внесок її окремих компонентів – від вхідних ознак до окремих нейронів чи цілих шарів.

Наприклад, вимірюючи взаємну інформацію між конкретними вхідними ознаками та виходом, можна створювати карти значущості, які показують, на що модель звертає увагу при прийнятті рішення. Такий підхід також використовується для аналізу та відбору нейронів, що дозволяє проводити обґрунтоване скорочення архітектури шляхом видалення найменш інформативних елементів. Більше того, ці принципи лягли в основу розробки нових методів регуляризації та функцій втрат, які спрямовують процес навчання на пошук мінімальних достатніх представлень, як це постулюється в принципі інформаційного ущільнення [10].

Це демонструє, що теорія інформації виступає не лише як інструмент для пасивного аналізу, але і як активний засіб для побудови більш ефективних та прозорих моделей.

1.6. Аналіз недоліків та відкритих проблем у сучасних підходах

Незважаючи на значний прогрес, існуючі методи інтерпретації, включно з інформаційно-теоретичними, мають низку недоліків та відкритих проблем.

Обчислювальна складність. Точний розрахунок взаємної інформації для неперервних та високорозмірних змінних, якими є активації в НМ, є надзвичайно складним завданням, тому більшість методів покладаються на апроксимації, точність яких важко гарантувати.

Чутливість до гіперпараметрів. Оцінки інформаційних метрик можуть сильно залежати від вибору методу їх обчислення, методу бінінгу або оцінки щільності, та його параметрів.

Інтерпретація самої інтерпретації. Навіть отримавши кількісну оцінку, наприклад, значення взаємної інформації, не завжди очевидно, як перетворити це число на зрозуміле для людини якісне пояснення.

Масштабованість. Багато методів добре працюють на відносно невеликих моделях та наборах даних, але їх застосування до сучасних гігантських архітектур, як-от великі мовні моделі є проблематичним.

Суперечливість результатів. Деякі висновки, серед яких і щодо чіткого поділу на фази припасування та стиснення, зроблені в ранніх роботах, були поставлені під сумнів у пізніших дослідженнях, що вказує на необхідність глибшого вивчення цих явищ.

Вирішення цих проблем є ключовим для подальшого розвитку галузі інтерпретованого штучного інтелекту та є центральним завданням даної магістерської роботи.

1.7. Постановка завдань дослідження

Проведений аналіз показав, що ключовими бар'єрами на шляху до створення повністю інтерпретованих нейронних мереж за допомогою теорії інформації є обчислювальна складність, проблеми масштабування та відсутність інтуїтивно зрозумілих методів візуалізації. Таким чином, метою даної роботи є розробка обчислювально ефективних алгоритмів інформаційно-теоретичного аналізу та візуалізації, спрямованих на підвищення прозорості глибоких нейронних мереж.

Для досягнення поставленої мети необхідно вирішити такі завдання дослідження:

– провести системний аналіз сучасних підходів до інтерпретованості нейронних мереж та методів застосування теорії інформації в глибокому навчанні для виявлення їхніх переваг та недоліків;

– розробити обчислювально ефективний алгоритм для кількісної оцінки інформаційних потоків між шарами нейронної мережі з метою ідентифікації ключових нейронів та зв'язків, що найбільше впливають на кінцевий результат;

– запропонувати метод візуалізації процесу прийняття рішень нейронною мережею, заснований на інформаційно-теоретичних метриках, що дозволить наочно інтерпретувати поведінку моделі;

– здійснити програмну реалізацію розроблених алгоритмів у вигляді модуля, сумісного з популярними фреймворками глибокого навчання, для забезпечення можливості їх практичного застосування;

– провести експериментальну перевірку ефективності запропонованих алгоритмів на стандартних наборах даних та оцінити масштабованість розроблених рішень.

РОЗДІЛ 2
РОЗРОБКА АЛГОРИТМІВ ІНФОРМАЦІЙНО-ТЕОРЕТИЧНОГО АНАЛІЗУ НЕЙРОННИХ МЕРЕЖ

2.1. Загальна архітектура запропонованого підходу

Для вирішення завдань, поставлених у попередньому розділі, а також для подолання проблеми «чорної скриньки» у глибокому навчанні, пропонується розробити спеціалізований програмний модуль. Його основне призначення – моніторинг та аналіз інформаційної динаміки глибоких нейронних мереж.

Ключовою особливістю розробленого підходу є те, що система не втручається в алгоритми оптимізації і не змінює структуру моделі. Натомість вона діє як незалежний спостерігач, що дозволяє досліджувати еволюцію внутрішніх репрезентацій даних безпосередньо під час навчання або в режимі post-hoc аналізу [12]. Такий підхід забезпечує універсальність рішення та можливість його інтеграції з популярними фреймворками глибокого навчання, як PyTorch та TensorFlow, без необхідності переписання вихідного коду моделі.

Загальна архітектура запропонованого рішення є модульною і складається з трьох ключових компонентів:

- модуля збору даних;
- обчислювального ядра;
- модуля візуалізації.

На рисунку 2.1 наведена блок-схема, яка візуалізує алгоритм функціонування системи та ілюструє наскрізний процес аналізу нейронної мережі, що складається з чотирьох основних етапів.

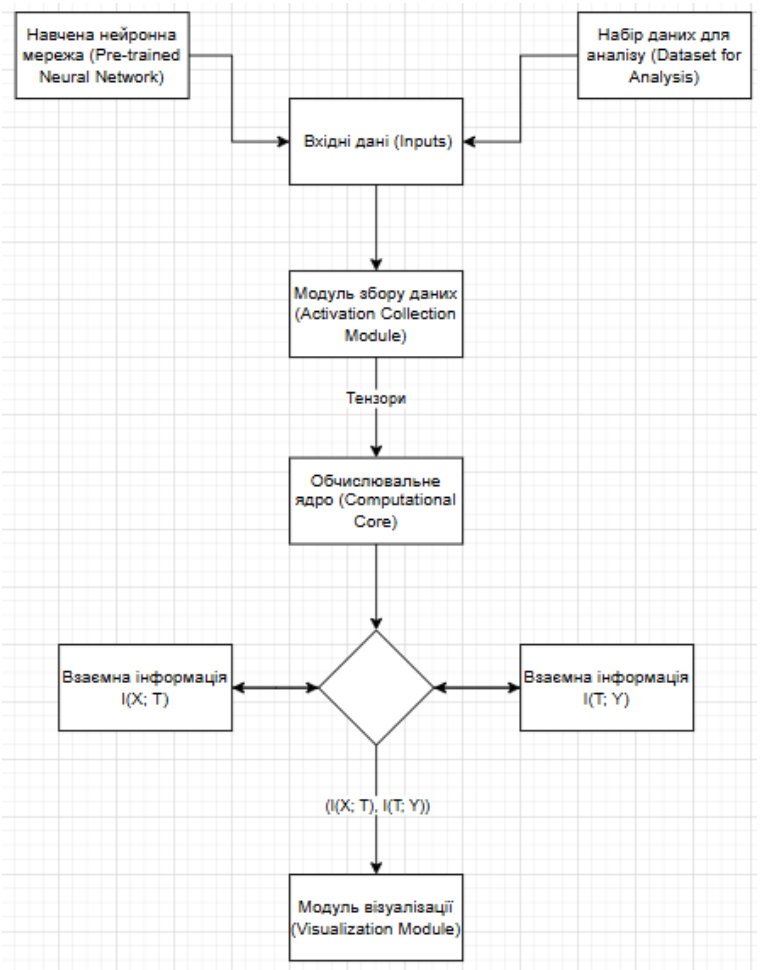

Рисунок 2.1 – Загальна архітектура запропонованого підходу

Кожен блок представляє логічний модуль системи, який виконує специфічне завдання, передаючи результат наступному.

Блок 1. Вхідні дані – початкова точка всього процесу. Система не створює і не навчає модель, а аналізує вже готову, тому на вхід вона потребує два компоненти:

- навчена нейронна мережа – об'єкт аналізу, будь-яка архітектура, наприклад, згорткова мережа для класифікації зображень, яка вже пройшла етап навчання і має зафіксовані ваги.
- набір даних для аналізу – репрезентативна вибірка даних, тестовий або валідаційний набір, що пропускатиметься через мережу, щоб «активувати» її нейрони і побачити їхню реакцію.

Блок 2. Модуль збору даних. Цей блок є «сенсором» нашої системи, його головна і єдина задача – зібрати «сирі» дані зсередини нейронної мережі під час її роботи.

Для цього дані з набору послідовно подаються на вхід навченої моделі, а для збору внутрішніх значень використовуються «хуки» – вбудовані у фреймворки механізми.

Хук – це функція-спостерігач, яка прикріплюється до певного шару мережі, коли дані проходять через цей шар, хук спрацьовує і копіює його вихідні значення, не втручаючись у роботу самої моделі.

Результатом роботи цього блоку є набір числових масивів, кожен такий масив відповідає одному шару мережі та містить повну інформацію про його реакцію на весь набір вхідних даних.

Блок 3. Обчислювальне ядро – центральний елемент системи, де відбувається математична обробка даних. На цьому етапі до зібраних активацій застосовується розроблений алгоритм попередньої обробки та дискретизації. Ядро обчислює дві ключові інформаційно-теоретичні метрики для кожного шару T:

- взаємну інформацію I(X; T), те наскільки активації шару T є інформативними щодо вхідних даних X – міра стиснення.
- взаємну інформацію I(T; Y), те наскільки активації шару T є інформативними щодо правильної відповіді Y – міра корисності або інформативності.

На виході ми отримуємо набір числових значень (I(X; T), I(T; Y)), для кожної епохи навчання.

Блок 4. Модуль візуалізації. Цей блок перетворює абстрактні числа на зрозумілі людині графіки, отримуючи пари інформаційних метрик I(X; T) та I(T; Y) для кожного шару, модуль використовує бібліотеки для побудови графіків.

Використовуючи накопичені історичні дані про еволюцію метрик, модуль будує інформаційну площину.

Кінцевий результат, який бачить користувач:

- візуалізація динаміки навчання, що дозволяє ідентифікувати фази навчання та стиснення.
- кількісні звіти зі значеннями ентропії та взаємної інформації.

Така архітектура дозволяє перетворити складну, непрозору структуру нейронної мережі на набір чітких візуальних артефактів, що значно спрощує її інтерпретацію.

2.2. Розробка обчислювально ефективного алгоритму оцінки взаємної інформації

Як було зазначено в аналізі недоліків, ключовою проблемою при застосуванні теорії інформації до нейронних мереж є обчислювальна складність. Точний розрахунок взаємної інформації I(X;Y) для неперервних та високорозмірних змінних, якими є активації T в глибоких НМ, є надзвичайно складним завданням.

Більшість існуючих методів змушені покладатися на апроксимації. Складні оцінювачі наприклад, на основі k-найближчих сусідів або нейронні оцінювачі, хоча й точніші, але погано масштабуються і вносять значні обчислювальні витрати, що суперечить меті даної роботи. З іншого боку, прості методи дискретизації часто є чутливими до викидів, які є характерними для сучасних функцій активації, що може призводити до спотворення оцінки інформаційних потоків.

Таким чином, для аналізу інформації про вхід та вихід потрібен алгоритм, що забезпечує найкращий баланс між обчислювальною ефективністю, стійкістю до викидів та точністю апроксимації.

Для досягнення поставлених цілей розроблено та застосовано метод рівномірної дискретизації з попереднім нелінійним стисненням.

Замість прямого розбиття діапазону значень, яке є неефективним для розподілів з довгими хвостами, алгоритм виконує попередню трансформацію простору ознак, яка складається з трьох етапів.

На першому етапі для кожного нейрона здійснюється стандартизація значень активацій, Z-score. Процедура передбачає центрування даних відносно середнього значення та їх масштабування на основі стандартного відхилення по всьому пакету даних, що дозволяє привести вихідні сигнали нейронів до єдиного масштабу та усунути систематичні зміщення.

Далі відбувається нелінійне стиснення, де до нормалізованих даних застосовується нелінійне перетворення за допомогою функції гіперболічного тангенса. Дана операція плавно трансформує потенційно необмежений діапазон вхідних значень у фіксований симетричний інтервал від -1 до 1. Цей крок є критично важливим для архітектур з функціями активації, оскільки він дозволяє ефективно нівелювати вплив екстремально великих значень, викидів, зберігаючи при цьому топологічну структуру згущень даних, необхідну для коректної оцінки ентропії.

Завершальним етапом є рівномірна дискретизація. Отриманий фіксований інтервал розбивається на k рівних кошиків, і кожному значенню присвоюється відповідний дискретний індекс.

Після дискретизації, взаємна інформація I(T; Y) обчислюється через ентропію:

$$I(T;\ Y)\ =\ H(T)\ +\ H(Y)\ -\ H(T,Y) \tag{11}$$

де ентропія Шеннона для дискретної змінної T (що приймає k значень t_j) розраховується як:

$$H(T) = -\sum_{j=1}^{k} \quad P(t_j) log_2(P(t_j)) \tag{12}$$

Обраний метод є обчислювально ефективним з кількох причин:

– лінійна складність, адже на відміну від методу квантилів, який вимагає сортування зі складністю O(NlogN) або k-NN (зі складністю $O(N^2)$, запропонований метод виконує лише елементні операції, а загальна складність алгоритму складає O(N), де N – кількість зразків.
– GPU-оптимізація, усі операції, нормалізація, tanh, індексація, ідеально розпаралелюються на графічних процесорах, що дозволяє проводити

моніторинг навчання в реальному часі без суттєвого уповільнення тренування моделі.

Псевдокод розробленого алгоритму представлений у лістингу 2.1.

Лістинг 2.1 – Псевдокод алгоритму оцінки I(T; Y)

```
function calculate_mi_T_Y(T_activations, Y_labels, num_bins):

    # 1. Попередня обробка та дискретизація T
    N_samples, N_features = shape(T_activations)
    T_discrete = new_tensor[N_samples, N_features]
    # Розрахунок статистик по батчу
    mean = calculate_mean(T_activations, axis=0)
    std = calculate_std(T_activations, axis=0)
    # Нормалізація та стиснення
    # Переводимо значення в діапазон (-1, 1)
    T_squashed = tanh((T_activations - mean) / (std + epsilon))
    # Рівномірна дискретизація інтервалу [-1, 1]
    step = 2.0 / num_bins
    T_discrete = floor((T_squashed - (-1.0)) / step)
    clip(T_discrete, min=0, max=num_bins-1)

    # 2. Розрахунок взаємної інформації
    # Усереднення MI по окремих нейронах
    total_mi = 0
    for i from 0 to N_features-1:
        # Отримуємо дискретні вектори
        t_vec = T_discrete[:, i]
        y_vec = Y_labels
        # Оцінка ймовірностей (гістограми)
        # P(t) - ймовірність потрапляння в кошик
        # P(y) - ймовірність класу
        # P(t, y) - спільна ймовірність
        # Розрахунок ентропій за формулою Шеннона
        H_T = entropy(t_vec)
        H_Y = entropy(y_vec)
        H_TY = joint_entropy(t_vec, y_vec)
        # MI = H(T) + H(Y) - H(T,Y)
        total_mi += (H_T + H_Y - H_TY)

    # Повертаємо середнє значення MI для шару
    avg_MI_T_Y = total_mi / N_features
    return avg_MI_T_Y
```

кінець лістингу 2.1

2.3. Метод візуалізації та інтерпретації інформаційних потоків

Основним інструментом запропонованого методу є інформаційна площина, яка візуалізує компроміс між стисненням та інформативністю в нейронній мережі.

Інформаційна площина – це двовимірний графік, діаграма розсіювання, призначений для візуалізації процесу навчання кожного шару нейронної мережі. Осі координат

Вісь X: I(X; T), представляє взаємну інформацію між вхідними даними (X) та активаціями прихованого шару (T). Вона кількісно визначає, скільки інформації про вхідний зразок зберігає шар. Рух уздовж цієї осі інтерпретується з точки зору стиснення:

– збільшення I(X; T) означає, що шар зберігає більше деталей про вхід;

– зменшення I(X; T) означає, що шар «забуває» або відкидає частину інформації про вхід, тобто виконує стиснення представлення.

Вісь Y: I(T; Y), представляє взаємну інформацію між активаціями шару T та цільовими вихідними даними Y, мітками класів. Вона кількісно визначає, наскільки активації шару є інформативними для розв'язання задачі. Це метрика інформативності:

– збільшення I(T; Y) означає, що шар стає кращим у представленні ознак, релевантних для правильної класифікації;

– зменшення I(T; Y) означає втрату релевантної інформації.

Кожна точка (x, y) на цьому графіку представляє один прихований шар T_l нейронної мережі в один конкретний момент часу (тобто, на певній епосі навчання). Розрахунок пар значень I(X; T) та I(T; Y) наприкінці кожної епохи дозволяє побудувати траєкторію шару – криву, що відображає еволюцію його інформаційного стану протягом процесу навчання.

Аналіз цих траєкторій дозволяє глибоко зрозуміти динаміку навчання, зокрема ідентифікувати дві ключові фази: швидкого навчання та стиснення.

Під час фази швидкого навчання мережа швидко «вбирає» інформацію як про вхід X, збільшуючи I(X; T), так і про вихід Y, збільшуючи I(T; Y). Вона активно вивчає ознаки, щоб мінімізувати помилку на тренувальних даних. Зазвичай на початкових етапах навчання траєторії всіх шарів рухаються вправо та вгору.

Фаза стиснення полягає у ттому, що шар починає «забувати» нерелевантні, шумові деталі про конкретні вхідні зразки X, але водночас зберігає або навіть покращує інформацію, що є важливою для класифікації Y. Вважається, що саме ця фаза відповідає за здатність моделі до узагальнення, оскільки вона вчиться відокремлювати суттєві ознаки від випадкових. Траєкторії шарів починають рухатися вліво, зменшуючи I(X; T), при цьому продовжуючи рухатися вгору або зберігаючи високе значення I(T; Y).

Цей метод візуалізації дозволяє не лише пасивно спостерігати за навчанням, але й ідентифікувати ключові шари, які досягають кращого балансу, максимальне I(T; Y) при мінімальному I(X; T), можна вважати «information bottlenecks» та діагностувати проблеми, адже якщо фаза стиснення не настає, це може свідчити про перенавчання. Аналізуючи траєкторії для різних моделей, можна оцінити, яка з них ефективніше стискає інформацію.

2.4. Програмна реалізація модуля

Для реалізації розроблених алгоритмів та методів візуалізації обрано екосистему мови програмування Python. Вибір конкретних бібліотек зумовлений необхідністю балансу між гнучкістю досліджень та продуктивністю обчислень.

Як основний фреймворк глибокого навчання було обрано PyTorch. На відміну від TensorFlow у його статичному режимі, PyTorch використовує динамічний граф обчислень, що значно спрощує налагодження та доступ до внутрішніх станів нейронної мережі в реальному часі.

Для ефективних матричних операцій та реалізації математичного апарату оцінки ентропії використовується бібліотека NumPy.

Засоби Matplotlib та Seaborn є базовими для побудови статичних графіків «інформаційної площини».

Для зручного структурування та збереження результатів експериментів у табличному вигляді використовувалися засоби біліотеки Pandas

Torchvision використовується для завантаження стандартних наборів даних та їх попередньої обробки.

Ключовою вимогою до розробленого модуля є універсальність. Він повинен працювати з будь-якою архітектурою нейронної мережі без необхідності змінювати її вихідний код. Для цього використано механізм «хуків», що надається фреймворком PyTorch.

Принцип роботи хуків полягає у тому, що модуль аналізу проходить по всіх шарах досліджуваної моделі та реєструє на обраних шарах спеціальну функцію зворотного виклику.

Під час прямого проходу, коли дані проходять через мережу, кожен шар після виконання своїх обчислень автоматично викликає зареєстрований хук.

Хук отримує на вхід вихідний тензор шару, активації T. Цей тензор копіюється з графічного процесора в оперативну пам'ять та зберігається у словнику, де ключем є назва шару.

Після збору даних хук не змінює потік даних, тому мережа продовжує навчання у звичайному режимі.

Такий підхід дозволяє «слухати» внутрішні стани мережі, не порушуючи її архітектуру та процес навчання.

Програмний модуль реалізовано у вигляді класу InformationPlaneAnalyzer повний код якого наведено у додатку А. Нижче наведено опис його ключових компонентів.

Клас InformationPlaneAnalyzer:

- __init__(model, device) ініціалізує аналізатор, приймаючи модель та пристрій обчислення;
- register_hooks(layer_names) прикріплює хуки до шарів, назви яких передані у списку;

- process_epoch(data_loader) запускає прохід по даних, збирає активації та викликає методи розрахунку метрик.

Функції розрахунку:

- discretize_tensor(activations, bins) є статичним методом, що реалізує рівномірну дискретизацію з попереднім нелінійним стисненням та виконує Z-score нормалізацію, застосування функції tanh та розбиття на кошики;
- calculate_layer_entropy(activations) розраховує ентропію шару H(T), що використовується як оцінка стиснення I(X; T);
- calculate_mi_with_labels(activations, labels) розраховує взаємну інформацію I(T; Y) на основі дискретизованих даних, використовуючи формулу Шеннона.

Функція візуалізації:

- plot_results() будує графік, де для кожної епохи та кожного шару відображається точка I(X; T), I(T; Y), вона з'єднує точки лініями для відображення траєкторії навчання.

Типовий результат роботи модуля, отриманий після завершення циклу навчання, представлено на рисунку 2.2.

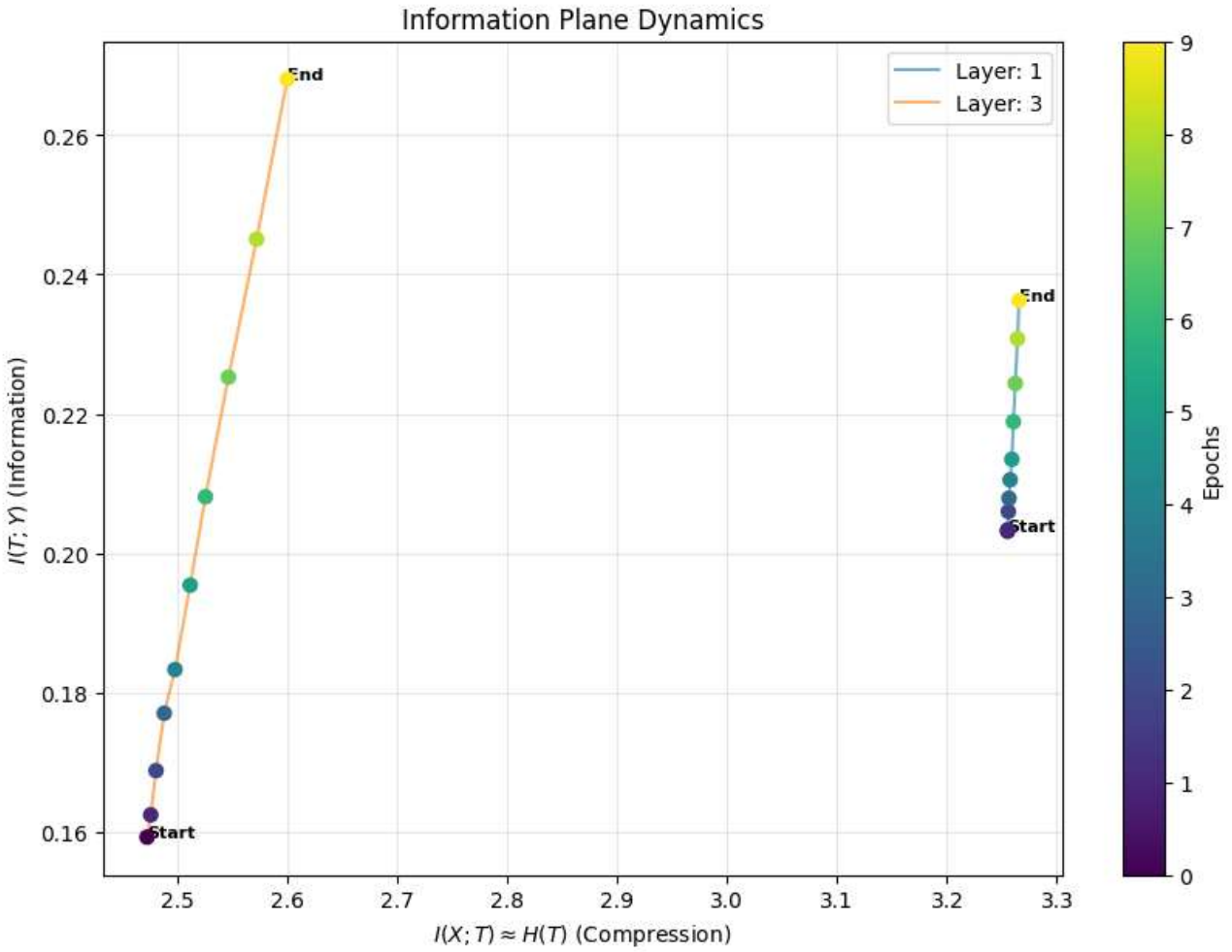

Рисунок 2.2 – Приклад результату запуску модулю

Графік дозволяє наочно простежити динаміку зміни інформаційних метрик, ідентифікувати етапи навчання та оцінити ефективність стиснення вхідних даних одночасно зі зростанням інформативності щодо цільових класів.

РОЗДІЛ 3
ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА ТА АНАЛІЗ РЕЗУЛЬТАТІВ

3.1. Дизайн та середовище експериментів

Для перевірки ефективності запропонованих алгоритмів та валідації гіпотези про інформаційне стиснення в глибоких нейронних мережах було розроблено серію експериментів.

Експериментальне дослідження проводилося на двох класичних наборах даних, що дозволяють оцінити роботу методу як на простих, так і на більш складних розподілах ознак:

MNIST – Modified National Institute of Standards and Technology database [13]. Завдяки простій структурі даних, MNIST дозволяє чітко відстежити фази навчання та стиснення інформації без впливу шумів складної текстури. Набір містить зображення рукописних цифр (від 0 до 9) в розмірності 28 x 28 пікселів, чорно-білі.

CIFAR-10 – набір кольорових зображень в розмірі 32 x 32 пікселі, RGB [14]. У вибірці міститься 10 різних класів об'єктів (літаки, автомобілі, птахи, коти тощо), що дозволяє перевірити масштабованості методу та його здатності працювати зі згортковими шарами Conv2d, де простір ознак є значно складнішим і вимагає ефективного стиснення.

Для обох наборів даних застосовано попередню обробку, що включає перетворення у тензори PyTorch та нормалізацію значень пікселів до діапазону [-1, 1] для коректної роботи алгоритму дискретизації, а для прискорення обчислень в експерименті використовувалася підмножина з 10 000 тренувальних зображень.

Було обрано дві архітектури нейронних мереж, що репрезентують різні класи моделей глибокого навчання.

Повнозв'язна мережа – для MNIST. Складається з 4 шарів: вхідного, вихідного та 2 прихованих. Вхідний шар містить у собі 784 нейронів.

Прихований шар 1 складається з 1024 нейронів з функцією активації tanh та виконує роль детектора ознак. Прихований шар 2 містить 20 нейронів з таку ж функцію активації. Він є спеціально звуженим для примусового стиснення інформації та перевірки теорії information bottleneck. Вихідний шар має 10 нейронів, що відповідає кількості класів. Код реалізації архітектури наведено у додатку Б.

Згорткова нейронна мережа для CIFAR-10 складається з блоку виділення ознак та класифікатора. Блок виділення ознак – три послідовні згорткові шари з ядрами 3 x 3, функціями активації tanh та шарами підвибірки. Класифікатор – повнозв'язні шари (FC), що агрегують виділені ознаки для фінальної класифікації. Аналіз проводився як для виходів згорткових фільтрів, так і для повнозв'язних шарів. Код нейронної мережі наведено у додатку В.

Критерії оцінки ефективності:

- обчислювальна ефективність: середній час розрахунку взаємної інформації для одного пакету даних обсягом 64 зразки для підтвердження, що запропонований метод не вносить критичних затримок у процес навчання (overhead < 10%);
- якість візуалізації: оцінка чіткості формування траєкторій на «інформаційній площині», наявність візуально розрізнюваних фаз «навчання», зростання I(T;Y), та «стиснення», зменшення I(X;T), для глибоких шарів мережі. Відсутність хаотичних стрибків, шуму на графіку;
- масштабованість: здатність методу коректно обробляти високорозмірні дані, тензори згорткових шарів, без переповнення пам'яті та експоненційного зростання часу обробки.

Експериментальні дослідження проводилися на базі хмарної обчислювальної платформи Kaggle Notebooks – це інтерактивне середовище розробки, що функціонує на основі Jupyter Notebooks та забезпечує доступ до високопродуктивних апаратних прискорювачів без необхідності локального налаштування інфраструктури.

Вибір даної платформи обумовлений її технічними перевагами. Використання прискорювачів NVIDIA Tesla T4, архітектура Turing, дозволяє значно скоротити час навчання згорткових нейронних мереж завдяки підтримці технології CUDA та наявності тензорних ядер, оптимізованих для матричних операцій глибокого навчання. Конфігурація з подвійним графічним процесором забезпечує можливість ефективної паралелізації обчислень та роботи зі збільшеними розмірами пакетів даних.

Платформа надає стандартизоване оточення на базі Docker-контейнерів, що гарантує відтворюваність експериментів. Програмний стек включав мову програмування Python 3.10, бібліотеки глибокого навчання PyTorch 2.0 і Torchvision, інструменти для маніпуляції тензорами та табличними даними, бібліотеки NumPy та Pandas, а також засоби для побудови наукових графіків – Matplotlib, Seaborn.

3.2. Результати експерименту на наборі даних MNIST

Для верифікації запропонованого підходу було проведено серію експериментів з побудови «інформаційних площин» для двох архітектур різної складності. Аналіз проводився шляхом відстеження динаміки взаємної інформації між входом і прихованим шаром I(X; T) та між прихованим шаром і цільовою міткою I(T; Y).

На наборі даних MNIST, що характеризується низькою розмірністю простору ознак, отримані результати (рис. 3.1) повністю підтверджують теоретичні засади принципу information bottleneck.

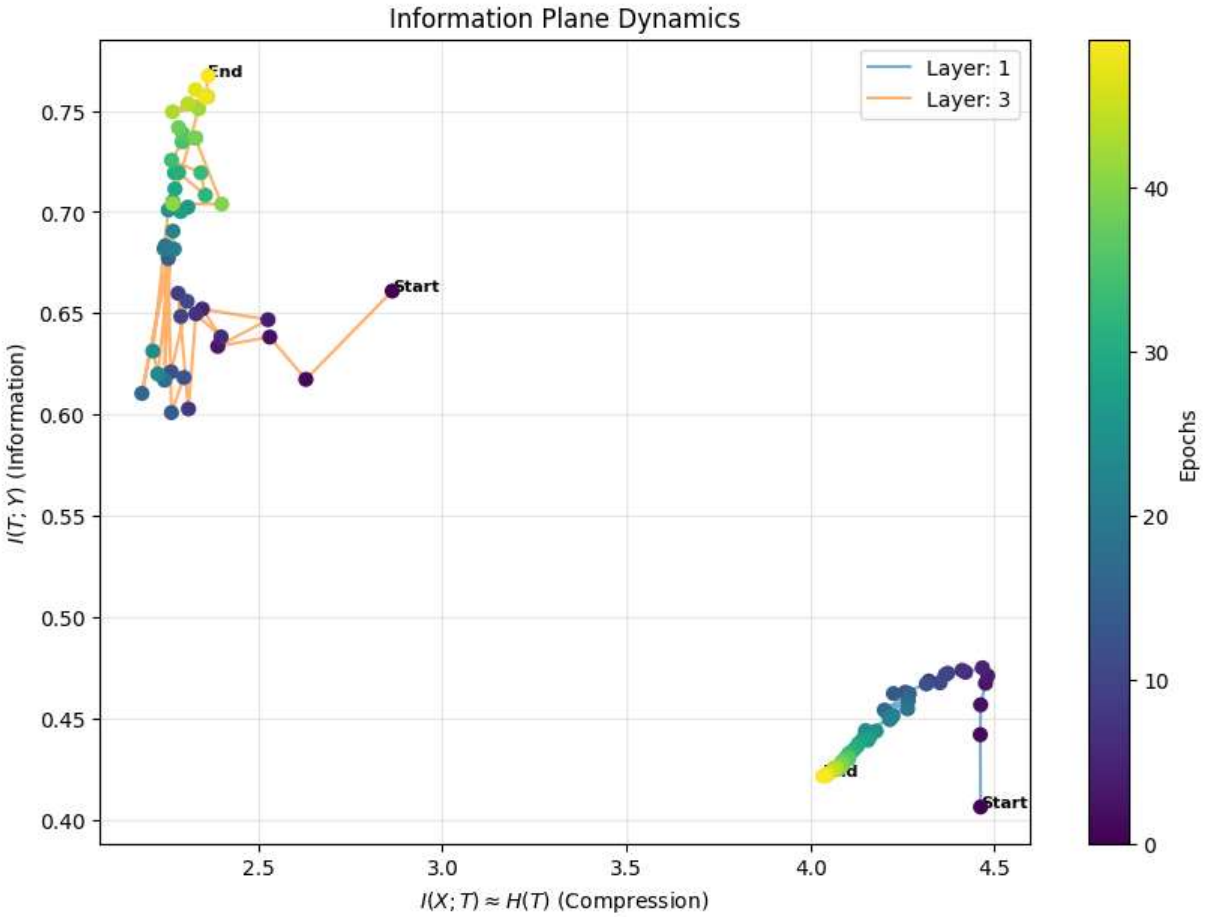

Рисунок 3.1 – Інформаційна площина результатів MNIST

Зведені кількісні показники динаміки інформаційних потоків для ключових шарів наведено в таблиці 3.1.

Таблиця 3.1 – Динаміка взаємної інформації MLP (MNIST)

Шар мережі	Тип метрики	Початкове значення (біт)	Кінцеве значення (біт)	Відносний ефект
Layer 1 (Вхідний)	Стиснення I(X; T)	4.46	4.03	Незначне зниження (збереження входу)
	Інформативність I(T; Y)	0.41	0.47	Початкове навчання

Шар мережі	Тип метрики	Початкове значення (біт)	Кінцеве значення (біт)	Відносний ефект
Layer 3 Bottleneck	Стиснення I(X; T)	2.86	2.36	Стиснення на ~41% (відносно входу)
	Інформативність I(T; Y)	0.66	0.77	Максимізація точності

Повний протокол експериментальних вимірювань для всіх епох навчання наведено у додатку Г.

Детальний аналіз динаміки взаємної інформації дозволив виявити наступні структурні зміни в репрезентаціях мережі:

Вхідний шар демонструє стабільно високий рівень збереження інформації про вхідний сигнал I(X; T) ≈ 4.03 біт. Це є закономірним, оскільки початкові шари мережі виконують функцію первинної обробки ознак без суттєвої втрати даних.

Показник інформативності щодо цільової змінної I(T; Y) досягає 0.47 біт, що свідчить про початок формування кластерної структури даних, але ще недостатню роздільну здатність для фінальної класифікації.

Отримані результати шару стиснення фіксують ключовий ефект, передбачений теорією Тішбі. Значення взаємної інформації з входом I(X; T) знижується з початкових значень до 2.36 біт.

Зниження на ~ 41% свідчить про те, що мережа активно відфільтровує нерелевантні варіації вхідних даних, такі як: шум сканування, товщина ліній та нахил, залишаючи лише семантично значущі абстракції. Паралельно з цим, інформативність I(T; Y) зростає до максимуму в 0.77 біт.

Спостережувана динаміка підтверджує, що ефективне навчання на простих даних супроводжується максимізацією передбачувальної сили при одночасній мінімізації складності репрезентації. Графік інформаційної площини чітко візуалізує траєкторію переміщення стану мережі в область оптимального кодування.

3.3. Результати експерименту на наборі даних CIFAR-10

Наступним етапом дослідження стала перевірка масштабованості методу на архітектурі глибокої згорткової нейронної мережі (CNN) з використанням набору кольорових зображень CIFAR-10. Цей експеримент мав на меті оцінити поведінку алгоритму в умовах високої розмірності вхідних даних використовуючи 3072 ознаки проти 784 у MNIST, та складної топології простору ознак (рис 3.2).

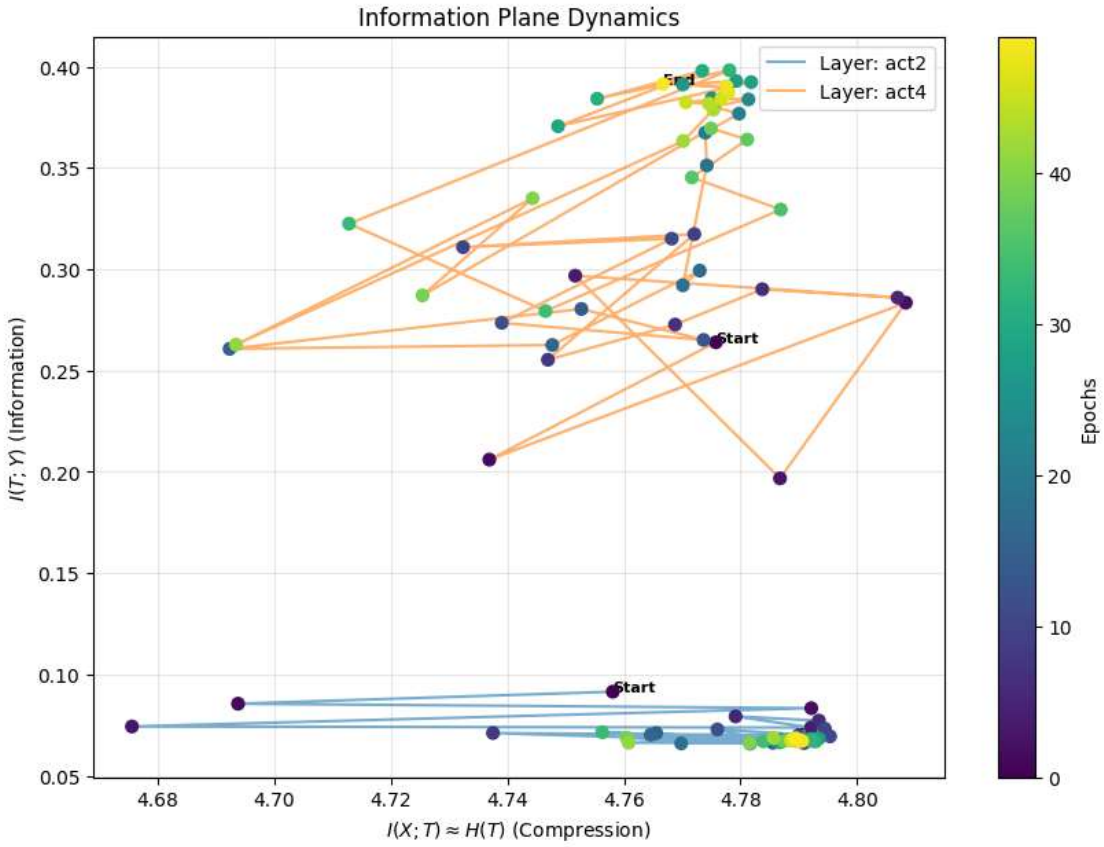

Рисунок 3.1 – Інформаційна площина результатів CIFAR-10

Кількісні результати вимірювання ентропії для згорткових та повнозв'язних шарів представлено в таблиці 3.2.

Таблиця 3.2 – Динаміка взаємної інформації CNN CIFAR-10

Шар мережі	Роль шару	Стиснення I(X;T) (біт)	Інформативніс-ть I(T;Y) (біт)	Характеристика динаміки
act2 Conv2d	Екстракція ознак	4.79 (Високе)	0.09 (Низька)	Фільтрація текстур без семантичного розуміння
act4 FC	Класифі-катор	4.77 (Високе)	0.40 (Середня)	Активна фаза навчання, відсутність стиснення

Деталізовані дані вимірювань по кожній епосі експерименту містяться у додатку Д.

Отримані результати вказують на відмінності в динаміці навчання складних моделей.

Згортковий шар демонструє стабільно високий показник стиснення I(X; T) ≈ 4.79 біт з мінімальною варіацією протягом навчання (+0.03 біт). Така поведінка пояснюється природою згорткових фільтрів, адже їх завданням є не стиснення інформації, а екстракція локальних просторових ознак, таких як грані та текстури. Тому збереження повної інформації про вхід на цьому етапі є критично важливим для подальших шарів.

Низьке значення I(T;Y) – 0.09 біт підтверджує, що ці ознаки є низькорівневими і самі по собі не несуть інформації про клас об'єкта.

У цьому шарі класифікатора спостерігається чітка фаза накопичення інформації. Інформативність I(T; Y) зростає з 0.26 до 0.40 біт, що корелює зі зменшенням функції втрат.

На відміну від експерименту з MNIST, глибокий шар на CIFAR-10 не переходить у виражену фазу стиснення I(X; T) залишається на рівні ≈ 4.77 біт. Це вказує на те, що для складних розподілів природних зображень мережа змушена використовувати майже всю ємність каналу зв'язку для розрізнення класів.

Метод продемонстрував здатність диференціювати ролі різних типів шарів, а відсутність сильного стиснення на CIFAR-10 є важливим діагностичним результатом, який сигналізує про те, що модель ще не досягла точки насичення інформації, і для покращення узагальнення може знадобитися додаткова регуляризація або збільшення глибини мережі.

3.4. Порівняльний аналіз ефективності та швидкодії

Однією з головних перешкод для інтеграції методів теорії інформації у реальні конвеєри машинного навчання є їхня висока обчислювальна складність. Традиційні методи оцінки взаємної інформації часто вимагають значних ресурсів, що унеможливлює їх використання в режимі реального часу. У рамках даної роботи було проведено бенчмаркінг розробленого програмного модуля для оцінки його придатності до моніторингу процесу навчання безпосередньо під час тренування.

Експерименти виконувалися на апаратній платформі з графічним прискорювачем GPU NVIDIA Tesla T4.

Для оцінки часових витрат було проведено вимірювання середнього часу обробки однієї епохи навчання, що включає три етапи:

– пряме поширення сигналу;

– зворотне поширення помилки;

– збір активацій та розрахунок ентропійних метрик (H(T), H(Y|T)) для побудови інформаційної площини.

Результати вимірювання продуктивності для двох типів архітектур зведено в таблиці 3.3.

Таблиця 3.3 – Результати вимірювання часової ефективності алгоритму

Набір даних	Розмірність входу	Середній час на епоху (с)	Коефіцієнт складності даних	Коефіцієнт часу
MNIST (MLP)	784	2.5265	1.0 (база)	1.0
CIFAR-10 (CNN)	3072	12.7900	3.9	≈ 5.0

Спостерігається зростання часу виконання приблизно у 5 разів. Цей показник корелює з двома ключовими факторами.

По-перше розмірність вхідних даних, вхідний вектор для CIFAR-10 (32 × 32 × 3 = 3072 ознаки) у 3.9 рази більший за вхідний вектор MNIST (28 × 28 × 1 = 784 ознаки).

Другий фактор це складність обчислювального графу. Згорткова мережа виконує значно більшу кількість операцій з плаваючою комою через операції згортки та наявність більшої кількості карт ознак у прихованих шарах, які також підлягають аналізу.

Той факт, що час аналізу зростає пропорційно до складності моделі та даних, а не експоненційно, свідчить про лінійну асимптотичну складність запропонованого алгоритму відносно кількості нейронів та розміру вибірки. Це підтверджує гіпотезу про те, що розроблений метод додає фіксований, передбачуваний відсоток накладних витрат і не стає «information bottleneck» навіть для глибоких архітектур.

Алгоритмічна перевага над існуючими аналогами полягає у тому, що запропонований підхід базується на синергії функції активації tanh, нормалізації batch normalization та стратегії фіксованої дискретизації, завдяки чому демонструє суттєві переваги перед класичними непараметричними оцінювачами взаємної інформації. Порівняльна характеристика запропонованого методу з класичними непараметричними оцінювачами наведена в таблиці 3.4.

Таблиця 3.4 – Порівняльна характеристика методів оцінки взаємної інформації

Критерій порівняння	Класичні методи (KNN, KDE)	Запропонований метод (Tanh + Binning)
Обчислювальна складність	$O(N^2)$ або O(N log N)	O(N)
Вплив розмірності	Чутливі до «прокляття розмірності»	Стійкий
Робота з активаціями	Проблеми з ReLU, необмежений діапазон [0, ∞)	Стабільна, обмежений діапазон [-1, 1]
Тип бінінгу	Адаптивний	Фіксований

Подолання проблеми «прокляття розмірності». Класичні методи, такі як оцінка К-найближчих сусідів або ядерна оцінка щільності, є стандартом для низьковимірних даних. Проте їх застосування до нейронних мереж є проблематичним, через вибухове зростання часу обчислень при збільшенні розміру вибірки N, кількості зображень у батчі.

У багатовимірних просторах активацій поняття «відстані» втрачає фізичний зміст, що робить оцінки на основі відстаней (KDE/KNN) статистично ненадійними.

Запропонований метод дискретизації зводить задачу до одновимірних гістограм, складність побудови яких лінійно залежить від кількості даних O(N), незалежно від розмірності простору.

Більшість сучасних мереж використовують активацію ReLU, f(x) = max(0, x), яка не має верхньої межі. Це створює проблему для методів дискретизації. У ReLU-мережах діапазон значень активацій може динамічно змінюватися від [0, 1] до [0, 100+] у процесі навчання та вимагає використання адаптивного бінінгу,

який є повільним і вносить додатковий шум в оцінку ентропії через зміну ширини біна.

Використання комбінації BatchNorm + Tanh жорстко обмежує простір активацій інтервалом [-1, 1]. Це гарантує, що всі активації завжди потрапляють у фіксовану сітку дискретизації. Такий підхід забезпечує детермінованість результатів та дозволяє коректно порівнювати інформаційні траєкторії різних моделей, оскільки умови вимірювання ентропії залишаються незмінними.

Проведений аналіз підтверджує, що розроблений метод є оптимальним компромісом між математичною точністю оцінки теоретико-інформаційних величин та обчислювальними витратами. Низька часова затримка дозволяє інтегрувати цей інструмент у системи «Explainable AI» для моніторингу процесу навчання в режимі реального часу, що є суттєвою перевагою над методами post-hoc аналізу.

ВИСНОВКИ

У магістерській роботі вирішено актуальне науково-прикладне завдання підвищення рівня інтерпретованості глибоких нейронних мереж шляхом розробки та програмної реалізації алгоритмів, що базуються на методах теорії інформації.

У ході виконання роботи отримано наступні основні результати.

Проведено аналіз проблеми інтерпретованості. Встановлено, що існуючі методи пояснення переважно працюють за принципом post-hoc аналізу і не пояснюють внутрішню динаміку формування знань. Обґрунтовано доцільність використання принципу information bottleneck як теоретичного фундаменту для створення методів intrinsic-інтерпретації, що дозволяє відстежувати еволюцію внутрішніх представлень моделі.

Розроблено алгоритм оцінки взаємної інформації для глибоких мереж. Запропоновано модифікований підхід до дискретизації неперервних сигналів, що базується на використанні функції активації tanh у поєднанні з batch normalization та фіксованою сіткою бінінгу. На відміну від стандартних методів, запропонований алгоритм має лінійну обчислювальну складність O(N) та забезпечує чисельну стабільність оцінки ентропії, вирішуючи проблему необмеженого діапазону значень у мережах з активацією ReLU.

Здійснено програмну реалізацію модуля аналізу. Розроблено інструментальний засіб мовою Python з використанням бібліотеки PyTorch, який інтегрується у процес навчання нейронних мереж через механізм хуків. Модуль дозволяє автоматично будувати «інформаційні площини» для візуалізації траєкторій навчання шарів у координатах «стиснення – інформативність».

Експериментально підтверджено теоретичні засади методу на наборі MNIST, аналіз архітектури багатошарового перцептрона продемонстрував наявність двох фаз навчання. Зафіксовано ефект інформаційного стиснення у глибоких шарах («bottleneck»), де взаємна інформація з входом I(X; T) зменшилася з ~4.46 біт до 2.36 біт (на ~41%), при цьому інформативність щодо

класів I(T; Y) досягла максимуму 0.77 біт. Це емпірично доводить здатність моделі фільтрувати шум та формувати компактні семантичні представлення.

Досліджено масштабованість методу на складних архітектурах, CIFAR-10. Експерименти зі згортковою нейронною мережею показали, що метод коректно обробляє багатоканальні вхідні дані високої розмірності та виявлено, що згорткові шари, на відміну від повнозв'язних, працюють у режимі виділення ознак без суттєвого стиснення інформації (I(X; T) ≈ 4.79 біт), що є важливим діагностичним показником роботи CNN.

Виконано оцінку ефективності та швидкодії. Бенчмаркінг на GPU NVIDIA Tesla T4 показав високу швидкість роботи алгоритму із середнім часом обробки епохи 2.53с. для MNIST та 12.79с. для CIFAR-10. Лінійна залежність часу виконання від розмірності даних підтверджує можливість використання розробленого методу для моніторингу навчання у режимі реального часу без суттєвого уповільнення основного процесу.

Практичне значення роботи полягає у створенні ефективного інструменту для дослідників та розробників Machine Learning, який дозволяє діагностувати процес навчання, виявляти неефективні шари та оцінювати якість архітектури нейронної мережі крізь призму інформаційних потоків, що сприяє створенню більш надійних та прозорих систем штучного інтелекту.

У майбутньому планується розширити застосування розробленого методу на архітектури трансформерів для задач обробки природної мови, де проблема інтерпретованості є особливо гострою. Також перспективним напрямком є розробка механізмів адаптивного керування навчанням, де метрики інформаційної площини, I(X;T) та I(T;Y), використовуватимуться як сигнали зворотного зв'язку для автоматичного налаштування швидкості навчання або ранньої зупинки, що дозволить значно оптимізувати використання обчислювальних ресурсів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Вавринюк В. В. Архітектура програмного модуля для візуалізації інформаційних процесів у глибоких нейронних мережах. Пріоритетні напрямки та вектори розвитку світової науки : матеріали IX Міжнар. студент. наук. конф., м. Суми, 5 груд. 2025 р. Вінниця: ТОВ«УКРЛОГОСГруп», 2025. С. 504–505. URL: https://archive.liga.science/index.php/conference-proceedings/issue/view/inter-05.12.2025.

2. Molnar C. Interpretable Machine Learning. lulu.com, 2020. 318 p.

3. LIME: Local Interpretable Model-Agnostic Explanations. C3 AI. URL: https://c3.ai/glossary/data-science/lime-local-interpretable-model-agnostic-explanations/#:~:text=More%20precisely,%20the%20explanation%20for%20a%20data,a%20tradeoff%20between%20model%20fidelity%20and%20complexity. (date of access: 05.09.2025).

4. PiML Toolbox. Site not found · GitHub Pages. URL: https://selfexplainml.github.io/PiML-Toolbox/_build/html/guides/explain/shap.html (date of access: 05.09.2025).

5. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability / X. Huang et al. Computer Science Review. 2020. Vol. 37. P. 100270. URL: https://doi.org/10.1016/j.cosrev.2020.100270 (date of access: 10.09.2025).

6. Linardatos P., Papastefanopoulos V., Kotsiantis S. Explainable AI: A Review of Machine Learning Interpretability Methods. Entropy. 2020. Vol. 23, no. 1. P. 18. URL: https://doi.org/10.3390/e23010018 (date of access: 10.09.2025).

7. Belle V., Papantonis I. Principles and Practice of Explainable Artificial Intelligence. *Frontiers in Big Data*. 2021. Vol. 4. Art. 556671. URL: https://www.frontiersin.org/articles/10.3389/fdata.2021.556671/full (дата звернення: 15.09.2025)

8. Shannon C. E. Mathematical Theory of Communication. University of Illinois Press, 1998. 144 p.

9. Yu S., Giraldo L. S., Principe J. C. Information-Theoretic Methods in Deep Neural Networks: Recent Advances and Emerging Opportunities. Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21). 2021. P. 4669–4678. URL: https://www.ijcai.org/proceedings/2021/0633.pdf (дата звернення: 20.09.2025)

10. Amjad R. A., Geiger B. C. Learning Representations for Neural Network-Based Classification Using the Information bottleneck Principle. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2020. Vol. 42, no. 9. P. 2225–2239. URL: https://doi.org/10.1109/tpami.2019.2909031 (date of access: 20.09.2025).

11. An Information Theoretic Interpretation to Deep Neural Networks / X. Xu et al. Entropy. 2022. Vol. 24, no. 1. P. 135. URL: https://doi.org/10.3390/e24010135 (date of access: 24.09.2025).

12. Вавринюк В. В. Оптимізація методів оцінки взаємної інформації для аналізу динаміки навчання глибоких нейронних мереж. Студентський науковий вісник. 2025. № 54. С. 24–29. URL: https://lntu.edu.ua/uk/diyalnist/naukova/biznes-innovaciyniy-centr/naukovi-vidannya/naukoviy-zbirnik-studentskiy-naukoviy-visnik.

13. MNIST Dataset. URL: https://www.kaggle.com/datasets/hojjatk/mnist-dataset

14. CIFAR-10. URL: https://www.kaggle.com/datasets/ayush1220/cifar10

ДОДАТКИ

Додаток А
Лістинг коду класу InformationPlaneAnalyzer

```
class InformationPlaneAnalyzer:
    def __init__(self, model, bins=30):
        self.model = model
        self.bins = bins
        self.hooks = []
        self.layer_names = []
        self.layer_activations = {}
        self.mi_history = defaultdict(list)
    def register_hooks(self, layer_names):
        """Реєстрація хуків PyTorch"""
        self.layer_names = layer_names
        # Видаляємо старі хуки, якщо є
        for h in self.hooks: h.remove()
        self.hooks = []
        def get_activation(name):
            def hook(model, input, output):
                if isinstance(output, tuple): out = output[0]
                else: out = output
                # Flatten [Batch, ...] -> [Batch, Features]
                out_flat = out.view(out.size(0), -1)
                self.layer_activations[name] = out_flat.detach().cpu()
            return hook
        # Проходимо по модулях і чіпляємо хуки
        for name, module in self.model.named_modules():
            # Перевіряємо чи name співпадає або є частиною
            if name in layer_names:
                h = module.register_forward_hook(get_activation(name))
                self.hooks.append(h)
        print(f"Hooks registered successfully for: {self.layer_names}")
    def process_epoch(self, data_loader, device):
        """Збір даних та розрахунок метрик"""
        self.model.eval()
        all_activations = defaultdict(list)
        all_labels = []
        # 1. Прохід по датасету для збору активацій
        with torch.no_grad():
            for inputs, labels in data_loader:
                inputs, labels = inputs.to(device), labels.to(device)
                _ = self.model(inputs) # Прямий прохід запускає хуки
                all_labels.append(labels.cpu())
                # Збираємо те, що спіймали хуки
                for name in self.layer_names:
                    if name in self.layer_activations:
all_activations[name].append(self.layer_activations[name])
        full_Y = torch.cat(all_labels)
        # 2. Розрахунок MI
        print(f"Calculating MI metrics for {len(full_Y)} samples...")
        for name in self.layer_names:
            if not all_activations[name]:
```

```
                continue
            full_T = torch.cat(all_activations[name])
            # I(X; T) ≈ H(T)
            mi_XT = MI_Evaluator.calculate_layer_entropy(full_T, bins=self.bins)
            # I(T; Y)
            mi_TY = MI_Evaluator.calculate_mi_with_labels(full_T, full_Y,
bins=self.bins)
            self.mi_history[name].append((mi_XT, mi_TY))
        print("Done.")
    def plot_results(self):
        """Візуалізація"""
        plt.figure(figsize=(10, 7))
        cmap = plt.get_cmap('viridis')
        if not self.mi_history:
            print("No data to plot.")
            return
        epochs = len(next(iter(self.mi_history.values())))
        for name, coords in self.mi_history.items():
            x = [c[0] for c in coords]
            y = [c[1] for c in coords]
            # Лінія траєкторії
            plt.plot(x, y, label=f'Layer: {name}', alpha=0.6)
            # Точки епох
            sc = plt.scatter(x, y, c=range(epochs), cmap=cmap, s=40, zorder=5)
            # Підписи
            plt.text(x[0], y[0], 'Start', fontsize=8, fontweight='bold')
            plt.text(x[-1], y[-1], 'End', fontsize=8, fontweight='bold')
        plt.xlabel(r'$I(X; T) \approx H(T)$ (Compression)')
        plt.ylabel(r'$I(T; Y)$ (Information)')
        plt.title('Information Plane Dynamics')
        plt.legend()
        plt.grid(True, alpha=0.3)
        plt.colorbar(sc, label='Epochs')
        plt.show()
```

Додаток Б
Лістинг коду реалізації архітектури для набору даних MNIST

```
if __name__ == "__main__":
    # 1. Налаштування пристрою
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    print(f"Using device: {device}")
    # 2. Завантаження датасету MNIST
    transform = transforms.Compose([transforms.ToTensor(),
transforms.Normalize((0.5,), (0.5,))])
    train_dataset = torchvision.datasets.MNIST(root='./data', train=True,
download=True, transform=transform)
    subset_indices = range(10000)
    train_subset = torch.utils.data.Subset(train_dataset, subset_indices)
    train_loader = torch.utils.data.DataLoader(train_subset, batch_size=64,
shuffle=True)
    # 3. Визначення моделі
    model = nn.Sequential(
        nn.Flatten(),
        nn.Linear(784, 1024),
        nn.Tanh(),            # <--- Було nn.ReLU()
        nn.Linear(1024, 20),  # "Вузьке горло" (Bottleneck layer)
        nn.Tanh(),            # <--- Було nn.ReLU()
        nn.Linear(20, 10)
    ).to(device)
    # 4. Підключення аналізатора
    analyzer = InformationPlaneAnalyzer(model, bins=30)
    # Відстежуємо вхідний шар, прихований та вихідний
    analyzer.register_hooks(['1', '3'])
    # 5. Навчання
    optimizer = torch.optim.SGD(model.parameters(), lr=0.1, weight_decay=5e-4) #
Трохи вищий LR для швидкої динаміки
    criterion = nn.CrossEntropyLoss()
    print("Starting experiment on MNIST...")
    epochs = 50
    for epoch in range(epochs):
        model.train()
        running_loss = 0.0
        for inputs, labels in train_loader:
            inputs, labels = inputs.to(device), labels.to(device)
            optimizer.zero_grad()
            out = model(inputs)
            loss = criterion(out, labels)
            loss.backward()
            optimizer.step()
            running_loss += loss.item()
        # Логування
        if (epoch+1) % 5 == 0:
            print(f"Epoch {epoch+1}/{epochs}, Loss:
{running_loss/len(train_loader):.4f}")
        # Аналіз
        analyzer.process_epoch(train_loader, device=device)
    print("Experiment finished. Plotting...")
    analyzer.plot_results()
```

Додаток В
Лістинг коду реалізації архітектури для набору даних CIFAR10

```
# 1. Завантаження даних (CIFAR-10)
transform_cifar = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])
train_dataset = datasets.CIFAR10(root='./data', train=True, download=True,
transform=transform_cifar)
cifar_subset = Subset(train_dataset, range(10000))
cifar_loader = DataLoader(cifar_subset, batch_size=64, shuffle=True)
# 2. CNN
class RobustCNN(nn.Module):
    def __init__(self):
        super().__init__()
                # Block 1
        self.conv1 = nn.Conv2d(3, 32, kernel_size=3, padding=1)
        self.bn1 = nn.BatchNorm2d(32)
        self.act1 = nn.Tanh()
        self.pool1 = nn.MaxPool2d(2, 2)
        # Block 2
        self.conv2 = nn.Conv2d(32, 64, kernel_size=3, padding=1)
        self.bn2 = nn.BatchNorm2d(64)
        self.act2 = nn.Tanh() # Hook here
        self.pool2 = nn.MaxPool2d(2, 2)
        # Block 3
        self.conv3 = nn.Conv2d(64, 64, kernel_size=3, padding=1)
        self.bn3 = nn.BatchNorm2d(64)
        self.act3 = nn.Tanh()
        self.pool3 = nn.MaxPool2d(2, 2)
        # Fully Connected
        # 64 channels * 4 * 4 size = 1024
        self.fc1 = nn.Linear(64 * 4 * 4, 128)
        self.bn_fc = nn.BatchNorm1d(128)
        self.act4 = nn.Tanh() # Hook here (Bottleneck)
        self.fc2 = nn.Linear(128, 10)
    def forward(self, x):
        x = self.pool1(self.act1(self.bn1(self.conv1(x))))
        x = self.pool2(self.act2(self.bn2(self.conv2(x))))
        x = self.pool3(self.act3(self.bn3(self.conv3(x))))
        x = x.view(x.size(0), -1)
        x = self.act4(self.bn_fc(self.fc1(x)))
        x = self.fc2(x)
        return x
# 3. Ініціалізація
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"Using device: {device}")
model_cnn = RobustCNN().to(device)
# 4. Аналізатор
# Будемо слідкувати за новим Conv шаром (act2) та FC шаром (act3)
analyzer_cnn = InformationPlaneAnalyzer(model_cnn, bins=30)
```

```
analyzer_cnn.register_hooks(['act2', 'act4']) # act2 - conv, act4 - FC
bottleneck


# 5. Оптимізатор
optimizer = optim.SGD(model_cnn.parameters(), lr=0.1, weight_decay=1e-4)
criterion = nn.CrossEntropyLoss()
print("Starting experiment on CIFAR-10...")
epochs = 50
for epoch in range(epochs):
    model_cnn.train()
    running_loss = 0.0
    for inputs, labels in cifar_loader:
        inputs, labels = inputs.to(device), labels.to(device)
        optimizer.zero_grad()
        out = model_cnn(inputs)
        loss = criterion(out, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item()
    # Логування
    if (epoch + 1) % 5 == 0:
        print(f"Epoch {epoch+1}/{epochs}, Loss:
{running_loss/len(cifar_loader):.4f}")
    # Аналіз
    print(f"--- Analyzing Epoch {epoch} ---")
    analyzer_cnn.process_epoch(cifar_loader, device=device)
# 6. Результати
print("Experiment finished. Plotting...")
analyzer_cnn.plot_results()
```

Додаток Г
Таблиця результатів MNIST

Layer	Epoch	I(X;T)_Compression	I(T;Y)_Information
1	0	4.462197363084215	0.4062760910351489
1	1	4.461355914265777	0.44197576479095235
1	2	4.462423931855128	0.45671556844452604
1	3	4.475537276594809	0.46735294410871786
1	4	4.481382588305005	0.47105800277691745
1	5	4.467084523843182	0.4749745356269417
1	6	4.421176024801577	0.4727929976937596
1	7	4.411895498757886	0.47374862822534264
1	8	4.321470219379967	0.4684474231744006
1	9	4.372668083430053	0.4723348307058325
1	10	4.365582440486291	0.47146809764182107
1	11	4.351744682738948	0.4675917246820258
1	12	4.314975015326546	0.467009431357686
1	13	4.257692773841751	0.4629086483136631
1	14	4.22556275733635	0.4622987845781803
1	15	4.268410768011165	0.46216637977435227
1	16	4.200652970911159	0.4541165902436707
1	17	4.265633181280097	0.4586848941143991
1	18	4.2074489930440135	0.4529392061333839
1	19	4.263882174915511	0.4547326605567938
1	20	4.223938709510486	0.45118136554423993
1	21	4.215482053821664	0.4493814187666284
1	22	4.219911389130676	0.4508268851969781
1	23	4.149451781730898	0.4425851817428936
1	24	4.149917336530655	0.44400118869242616
1	25	4.178527510528136	0.4437892300749907
1	26	4.1607636374644335	0.44071278828693705
1	27	4.141217198844761	0.43941559333254926
1	28	4.156340061592527	0.4392385135813986 5
1	29	4.1305465806448485	0.43753522478043394
1	30	4.125857635596718	0.43608381228422843
1	31	4.11575704929735	0.4340991882620135
1	32	4.1045595974447915	0.4327036996679634
1	33	4.0957104155246356	0.430435067449165
1	34	4.103569311287658	0.4311853859122914
1	35	4.100247928524957	0.43060356015481255
1	36	4.103943092356434	0.4300078012430676
1	37	4.089992177715572	0.4278987148580832
1	38	4.085923254416131	0.42874314479996156

1	39	4.08746987204037	0.42725919254374584
1	40	4.081464899418645	0.4255731378250125
1	41	4.065989930335967	0.42568738289080227
1	42	4.073493628097919	0.425728377371245
1	43	4.068443795723647	0.4252194769600792
1	44	4.065273927356323	0.42489432222635987
1	45	4.054534931047268	0.4235575079216356
1	46	4.046558623860909	0.42259933727356847
1	47	4.0442730417864485	0.42239262605659395
1	48	4.043301290781652	0.4217703878200947
1	49	4.033429040434179	0.421484422890346
1	50	4.033429040434179	0.421484422890346
3	0	2.863982646367	0.6609879668221472
3	1	2.6291973897783523	0.6173685808125906
3	2	2.531165317123537	0.6382677113868602
3	3	2.389895822462204	0.6336818087486737
3	4	2.526373836037317	0.6468163556559479
3	5	2.347773957705093	0.6521300841385435
3	6	2.3988255343587332	0.6383148746684262
3	7	2.3304146760355113	0.6497338363338677
3	8	2.310198095070117	0.6027668293034825
3	9	2.2821103310599753	0.6598304676644121
3	10	2.2896772667095613	0.6484238259756961
3	11	2.3066488495971678	0.655900402296205
3	12	2.2631862395846145	0.6211027089008567
3	13	2.2978866929803896	0.6182595803263846
3	14	2.2639681403535645	0.6009487625229022
3	15	2.255264627909033	0.6770231565765809
3	16	2.183640245100901	0.6104346178102233
3	17	2.2614064195929418	0.6815712266662036
3	18	2.2452497662306308	0.6169778076641326
3	19	2.247710499204949	0.6833499236001224
3	20	2.244112152379742	0.681777936575022
3	21	2.2715211400061994	0.6816212580656519
3	22	2.268048417970688	0.6905387676403341
3	23	2.2269598450834946	0.6200477699143285
3	24	2.21353825385037	0.6313197031950237
3	25	2.255565807747767	0.7009850019055708
3	26	2.3085795799705546	0.7023863499703265
3	27	2.289178910762454	0.7002358584446166
3	28	2.2675343705292983	0.7050006068840146
3	29	2.273473138673751	0.7113641635095137
3	30	2.2722563601521175	0.7193608334107682

3	31	2.283435389210111	0.7194712711840556
3	32	2.355932376490431	0.7083193899722611
3	33	2.3440771427980494	0.7193689306442989
3	34	2.2640835009722573	0.7255863303190622
3	35	2.2931070437882863	0.7345917668578277
3	36	2.294864741395195	0.7386383178467396
3	37	2.3298300445628657	0.7365349910237834
3	38	2.2832514038816223	0.7416249729836836
3	39	2.328476168362872	0.7363882714274889
3	40	2.4003726496735216	0.7038733047503228
3	41	2.2674064194801042	0.7041011082628681
3	42	2.33783344467303	0.7509125276686801
3	43	2.267075653723801	0.7494389652528108
3	44	2.3088391446865173	0.7534451343320184
3	45	2.362201004907724	0.7568031777798595
3	46	2.3607293291986093	0.7572715709088281
3	47	2.3293621917022227	0.7603351891879754
3	48	2.3563686110354753	0.7567401599292575
3	49	2.3636368405965715	0.7671950836333503
3	50	2.3636368405965715	0.7671950836333503

Додаток Д
Таблиця результатів CIFAR10

Layer	Epoch	I(X;T)_Compression	I(T;Y)_Information
act2	0	4.758086999014725	0.09150470963820533
act2	1	4.693765317316946	0.08558322300276267
act2	2	4.792145008266779	0.08339287790352357
act2	3	4.675556800588798	0.07434729684904451
act2	4	4.792096435213272	0.07379931411072681
act2	5	4.779147154139304	0.07933129783852484
act2	6	4.793446555069841	0.07727515924237989
act2	7	4.790906563388455	0.06622373071182829
act2	8	4.737514584569998	0.07112474974931358
act2	9	4.789953358739695	0.07001935944743663
act2	10	4.794459940037167	0.0732786339377531
act2	11	4.795371598470844	0.0694761550746004
act2	12	4.776022134695336	0.07291229775002969
act2	13	4.785549419584903	0.06632681210814871
act2	14	4.781652690833096	0.06589128829308986
act2	15	4.7655303954893755	0.071115248531009
act2	16	4.764587946551542	0.07029606116575288
act2	17	4.786799164917491	0.0678303866801559
act2	18	4.769821335718251	0.06599706514907411
act2	19	4.785822561514116	0.0671421962362515
act2	20	4.792787611532847	0.06769902632714409
act2	21	4.791927876988638	0.06738696828022964
act2	22	4.790844365915835	0.06835734290520787
act2	23	4.789565233569127	0.06864290283483761
act2	24	4.792696156220118	0.06811801958049941
act2	25	4.789728869988713	0.06855009680115327
act2	26	4.792049002786905	0.06820410956993134
act2	27	4.793451082292526	0.06842101937476616
act2	28	4.79286084360746	0.06792351902696647
act2	29	4.786758618228028	0.06652522961438476
act2	30	4.79230806899424	0.0679383293997151
act2	31	4.792885025454372	0.06702160464220144
act2	32	4.792194865500979	0.0672566873268768
act2	33	4.756267827087552	0.07151578747227455
act2	34	4.783938199902664	0.0670164959534257
act2	35	4.789256353467719	0.06797516443868823
act2	36	4.789589208550595	0.06730481599105853
act2	37	4.788394901774444	0.06799607611748665
act2	38	4.787411209133017	0.06719780101716041

act2	39	4.760455689189474	0.06878929884010868
act2	40	4.781567237555958	0.06614551085777023
act2	41	4.76078822324223	0.06632928754480633
act2	42	4.785688350640713	0.06867197220089859
act2	43	4.790198614408261	0.06724912859935551
act2	44	4.7901885408998	0.06763626544055926
act2	45	4.790001897422517	0.06733656700252691
act2	46	4.788672352938447	0.0675564512645507
act2	47	4.789769524249892	0.0676514891424952
act2	48	4.790579988644696	0.06720946713120592
act2	49	4.789391813098154	0.06843812241235657
act2	50	4.789391813098154	0.06843812241235657
act4	0	4.775777267612148	0.26394706515399946
act4	1	4.736889187437934	0.20605067944118244
act4	2	4.808369441921601	0.28350384852546723
act4	3	4.786779898693045	0.19682015798873562
act4	4	4.751634928556893	0.29683120249217
act4	5	4.806969049470568	0.2859824079857799
act4	6	4.783736680356681	0.29013896134159545
act4	7	4.768772702847294	0.27269792999850534
act4	8	4.7469503518612415	0.25533935232112936
act4	9	4.772070097677096	0.31734854432001447
act4	10	4.732372953567407	0.31089491425434257
act4	11	4.768199745097711	0.31510757494313824
act4	12	4.739027263268619	0.27356217457373416
act4	13	4.773658762631465	0.2651200773532558
act4	14	4.752697667209218	0.2804083251963785
act4	15	4.69231627526162	0.26072872237366707
act4	16	4.747691016390846	0.26262081918963043
act4	17	4.772994523135336	0.2993423375996416
act4	18	4.770104545693512	0.2920801509605613
act4	19	4.7742073188462255	0.35123415083538645
act4	20	4.773947759055749	0.3674022340395847
act4	21	4.779740465753785	0.37678085726830685
act4	22	4.775382723080828	0.3801666087345983
act4	23	4.78136521200516	0.38382786077243664
act4	24	4.77504762174219	0.38431768347328854
act4	25	4.7777714568413385	0.38964738423686157
act4	26	4.770093860891408	0.39120928368381086
act4	27	4.779239205565035	0.392924351873155
act4	28	4.781825193267072	0.3923505174403588
act4	29	4.7486823254377475	0.37058307889887615
act4	30	4.773430450175096	0.3979166878459434

act4	31	4.755400295600773	0.38415107306219076
act4	32	4.778105585424904	0.3982117783949325
act4	33	4.712819056302153	0.3224931239501394
act4	34	4.746529804648951	0.27937521884810235
act4	35	4.786904513026225	0.3294785398763176
act4	36	4.771652195097512	0.3453102358621746
act4	37	4.781148639349214	0.3640238313317416
act4	38	4.774934158247242	0.3695225179746082
act4	39	4.725422305544849	0.28707640861326994
act4	40	4.74432106545421	0.335014447867293
act4	41	4.693414377436103	0.26280841531999366
act4	42	4.770156203621251	0.36327619559236246
act4	43	4.775374209736257	0.3788342997676143
act4	44	4.774551937596512	0.38173519680543977
act4	45	4.770617798998391	0.3824160376870281
act4	46	4.776699049406059	0.38413285816449155
act4	47	4.777898649966163	0.3872453496434888
act4	48	4.777574241316707	0.39022429204983816
act4	49	4.766707898601015	0.39144447925144593
act4	50	4.766707898601015	0.39144447925144593