

Івано-Франківський національний технічний університет нафти і газу
Міністерство освіти і науки України

Кваліфікаційна наукова
праця на правах рукопису

Яковин Сергій Васильович

УДК 004.4:004.04:004.9

ДИСЕРТАЦІЯ
МЕТОД ЕМУЛЯЦІЇ ЦИФРОВИХ ЛОГІЧНИХ КОМПОНЕНТ
ДИСКРЕТНИМ ПЕРЦЕПТРОНОМ

123 – Комп’ютерна інженерія
12 – Інформаційні технології

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело  Яковин Сергій Васильович

Науковий керівник
доктор технічних наук, професор
Мельничук Степан Іванович

Івано-Франківськ–2026

АНОТАЦІЯ

Яковин Сергій Васильович. Метод емуляції цифрових логічних компонент дискретним перцептроном. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 123 – Комп’ютерна інженерія. – Івано-Франківський національний технічний університет нафти і газу, Івано-Франківськ, 2026.

На сучасному етапі розвитку комп’ютерних систем спостерігається їх інтенсивна інтеграція зі штучними нейронними мережами. Удосконалення обчислювальних процесів у нейронних мережах має важливе прикладне значення для кіберфізичних систем. Зокрема, ефективне опрацювання дискретних сигналів дозволяє оптимально вирішувати задачі оцінювання динамічних навантажень у складних технічних системах. Іntenсивний розвиток останніх зумовлює необхідність пошуку можливих шляхів зменшення їх обчислювальної і, як наслідок, апаратної складності.

У дисертаційній роботі вирішена науково-практична задача, що пов’язана із розробкою нових структурних рішень компонентів нейронних мереж. А саме структурних, алгоритмічних та схемних рішень перцептрону, що реалізує опрацювання сигналів синапсів на основі статистичного оцінювання їх дискретних станів. Важливим аспектом отриманих результатів є перспективи реалізації спеціалізованих штучних нейронних мереж на платформах з обмеженими обчислювальними ресурсами, таких як мікроконтролери, програмовані логічні інтегральні схеми тощо.

У **вступі** обґрунтовано актуальність задач дослідження перцептронних структур як базових компонентів нейронних мереж, зокрема в частині їх імплементації в дискретному базисі, з метою зниження обчислювальних затрат, відображено зв’язок дисертаційної роботи з науковими програмами, планами, темами; сформульовано мету та поставлено основні завдання для обраного напрямку дослідження; визначено об’єкт та предмет дослідження;

вказано наукову новизну та практичне значення отриманих результатів, зазначено особисті внески здобувача, які викладені в публікаціях за темою роботи, подано відомості про кількість публікацій автора та ступінь апробації дисертаційної роботи.

У **першому розділі** дисертаційної роботи проаналізовано сучасний стан розвитку та впровадження перцептронних структур у комп'ютерних системах за яким показано, що класичні перцептрони реалізуються через операції згортки чи кореляції, які передбачають формування зваженої суми вхідних сигналів із подальшою активацією за допомогою порогової функції, внаслідок чого мають обмежену виразну здатність та не можуть ефективно розпізнавати лінійно-нероздільні шаблони. Проведено класифікацію та порівняння типових функцій кореляції в контексті імплементації у перцептронах за якою встановлено, що їх використання значно ускладнює алгоритмічну реалізацію, зокрема через потребу в операціях центрування, нормування тощо.

Розглянуто сучасні підходи до побудови активаційних функцій, виокремлено класи сигмоїдних, кусочно-лінійних та спеціалізованих функцій, а також вказано на їх вплив на обчислювальну складність і здатність нейронної мережі до навчання. Встановлено, що існує чіткий зв'язок між кількістю вхідних параметрів булевої функції та мінімально необхідною кількістю вагових коефіцієнтів у перцептронній структурі, що дозволяє оцінювати її компонентну складність, а також реалізувати їх точну емуляцію з мінімальними ресурсами.

Запропоновано вдосконалення реалізації класичного перцептрону при якому замість зваженого сумування використовується оцінка інформаційної ентропії, що дозволяє емулювати повний набір логічних функцій в межах однієї перцептронної структури. Означено перспективи подальшого розвитку перцептронних структур, зокрема з використанням різних інформаційних мір, що є інваріантні до інверсії бінарних сигналів і менш чутливі до шумів, проте потребують ефективних алгоритмів реалізації агрегуючої функції.

У **другому розділі** дисертаційної роботи вперше запропоновано

реалізацію дискретного перцептрону на основі імовірнісних оцінок зміщених синаптичних сигналів, що дозволяє перейти до дискретного базису, а також уникнути операцій множення та спростити апаратну реалізацію. Розроблена структура визначає рівень активації перцептрону за кількістю унікальних станів зміщених синаптичних сигналів, що реалізує ентропійну оцінку за спрощеною формулою Хартлі без використання логарифмічної функції.

Показано доцільність переходу до дискретного базису в реалізації перцептронів, що забезпечує зменшення обчислювальної складності та дозволяє ефективно працювати з обмеженим числом станів сигналів, а також встановлено, що використання інформаційної ентропії дозволяє зменшити чутливість моделі до абсолютних значень вхідних сигналів. Досліджено можливості емуляції логічних функцій AND, OR, XOR, NAND, NOR, XNOR на основі запропонованого перцептрону, що підтверджує здатність до опрацювання лінійно-нероздільних множин, а також універсальність підходу до побудови логічних компонентів у дискретному просторі.

Запропоновано метод обмеження множини рішень для дискретного перцептрона шляхом аналізу матриці кількості унікальних станів для 2-ох бінарних входів при фіксованому порозі активації, що зменшує простір пошуку до 35% від повного. Досліджено вплив вибору функції активації як операції порівнювання ($>$, $<$, $=$, $\neq$), що у поєднанні з відповідними коефіцієнтами зміщення дозволяє адаптувати перцептрон до різних типів логічних функцій та їх типових комбінацій.

У **третьому розділі** дисертаційної роботи здійснено розробку моделі дискретного перцептрона на основі багатошарової класичної нейронної мережі, що дозволяє застосувати типові методи машинного навчання нейромереж для визначення коефіцієнтів зміщення для запропонованого дискретного перцептрона. Запропоновано використання гладких апроксимацій індикаторної та порогової функцій (зокрема на основі гіперболічного секансу та логістичної функції), що дозволяє адаптувати дискретний перцептрон до методів градієнтного навчання.

Встановлено, що заміна негладких функцій на відповідні диференційовані апроксимації забезпечує безперервність градієнта та відкриває можливість застосування оптимізаційних алгоритмів у структурах з дискретною природою сигналів. Розроблено модифіковану порогову функцію, яка коректно працює як з додатними, так і з від'ємними порогами, що дає змогу реалізовувати як нижню, так і верхню межу активації на основі кількості унікальних значень зміщених вхідних сигналів. Встановлено можливість використання дискретного перцептрона як поліфункціонального логічного елементу, здатного реалізовувати широкий спектр логічних операцій, зокрема за рахунок вибору відповідної активаційної функції та коефіцієнтів зміщення.

Продемонстровано, що класична структура перцептрона залишається придатною для модифікації на основі зміщених сигналів, однак в такому випадку зникає потреба у множенні, що істотно знижує обчислювальну складність Input-компоненту. Розроблено каскадно-умовний та умовно-логічний алгоритми компоненту визначення кількості унікальних станів і на основі порівняльного аналізу встановлено, що реалізації на основі логічних функцій, замість каскадних умов, мають нижчу часову складність, дозволяють покращити ефективність схемної/програмної реалізації та масштабованість на більшу кількість входів. Отримано логічні вирази для визначення кількості унікальних станів для 3-ох та 4-ох входових перцептронів, що дозволяє мінімізувати кількість умовних конструкцій при алгоритмічній та апаратній реалізаціях дискретних компонентів такого типу.

У **четвертому розділі** дисертаційної роботи розроблено цифровий компонент shift2b як паралельний двобітовий суматор, що виконує апаратне зміщення синаптичних сигналів на основі операції додавання, зменшуючи вимоги до апаратної платформи і дозволяє отримати затримку, яка не перевищує 6 нс для всіх логічних шляхів.

Розроблено цифровий компонент cmp2b, який реалізує порівняння двох двобітових значень, має просту структуру, затримка операції не перевищує 6 нс, що узгоджується з компонентом shift2b, забезпечуючи можливість роботи

на частотах до 166 МГц. Розроблено цифровий компонент *cnts*, що реалізує підрахунок кількості унікальних значень серед зміщених синаптичних сигналів, структура якого ґрунтується на трьох компараторах *cmp2b* та базовій логіці, а результат формується як двобітний код, затримка операції не перевищує 9.6 нс, що дозволяє працювати на частотах до 105 МГц.

Розроблено апаратну структуру дискретного перцептрону на два бінарні входи, що включає окремі цифрові компоненти опрацювання вхідних сигналів (*shift2b*), порівняння проміжних результатів (*cmp2b*) та агрегування інформації (*cnts*). Компоненти реалізовано з урахуванням мінімізації апаратних витрат і забезпечення високої швидкодії. Побудовано інтегрований модуль дискретного перцептрону, який реалізує логіку класифікації за ентропійною ознакою, що забезпечує емуляцію базових логічних функцій: OR, AND, XOR, NOR, NAND, XNOR. Вихідні сигнали у всіх випадках відповідали таблицям істинності, що підтверджує універсальність розробленої схеми та її придатність для реалізації цифрових логічних схем.

Розроблено модульну апаратну структуру, що підтримує імплементацію конвеєризації для підвищення тактової частоти системи. Завдяки чітко визначеним межах між блоками можливе масштабування функціоналу без зміни логічних зв'язків, що забезпечує адаптивність для складніших конфігурацій на ПЛІС.

Ключові слова: перцептрон; бінарні сигнали; імовірнісні оцінки; логічні функції; дискретні сигнали; комп'ютерні компоненти; програмовані логічні інтегральні схеми (ПЛІС); нейронні мережі; машинне навчання.

ABSTRACT

Yakovyn Serhiy Vasyliovych. Method for Emulating Digital Logic Components Using a Discrete Perceptron. — A qualification scientific work submitted as a manuscript.

Dissertation for the degree of Doctor of Philosophy in specialty 123 – Computer Engineering. – Ivano-Frankivsk National Technical University of Oil and Gas, Ivano-Frankivsk, 2026.

At the current stage of computer systems development, there is intensive integration of such systems with artificial neural networks. Improving computational processes in neural networks has significant practical value for cyber-physical systems. In particular, efficient processing of discrete signals enables optimal solutions to problems of dynamic load assessment in complex technical systems. The rapid advancement of such systems necessitates finding ways to reduce their computational, and consequently, hardware complexity.

This dissertation addresses a scientific and practical problem related to the development of new structural solutions for neural network components—namely, structural, algorithmic, and schematic solutions for a perceptron that processes synaptic signals based on statistical evaluation of their discrete states. A key aspect of the results is the potential for implementing specialized artificial neural networks on platforms with limited computational resources, such as microcontrollers, field-programmable gate arrays (FPGAs), etc.

The **introduction** justifies the relevance of studying perceptron structures as fundamental components of neural networks, particularly in their implementation in a discrete basis to reduce computational costs. It highlights the connection of the dissertation with scientific programs, projects, and themes; formulates the objective and main research tasks; defines the object and subject of study; outlines the scientific novelty and practical significance of the results; and indicates the author's personal contributions, provides information on the number of publications and the extent of dissemination/validation of the research.

The **first** chapter analyzes the current state of development and implementation of perceptron structures in computer systems. It is shown that classical perceptrons are realized through convolution or correlation operations, which form a weighted sum of input signals followed by threshold activation. As a result, they have limited expressive power and cannot effectively recognize linearly inseparable patterns. A classification and comparison of typical correlation functions in the context of perceptron implementation reveals that their use complicates algorithmic realization, especially due to operations such as centering and normalization.

Contemporary approaches to activation function construction are reviewed, including sigmoid, piecewise-linear, and specialized functions, emphasizing their impact on computational complexity and neural network trainability. A clear relationship is established between the number of Boolean function inputs and the minimum number of required weights in a perceptron structure, enabling component complexity estimation and accurate emulation with minimal resources.

An improvement to the classical perceptron is proposed, where information entropy evaluation replaces weighted summation. This enables emulation of the full set of logical functions within a single perceptron structure. Prospects are outlined for further development of perceptron structures using various information measures that are invariant to binary signal inversion and less noise-sensitive, though they require efficient aggregation function algorithms.

The **second** chapter presents, for the first time, the implementation of a discrete perceptron based on probabilistic estimates of shifted synaptic signals, transitioning to a discrete basis and eliminating multiplication operations, thus simplifying hardware implementation. The proposed structure determines the perceptron's activation level by the number of unique shifted synaptic signal states, realizing an entropy estimate using Hartley's simplified formula without logarithmic operations.

The feasibility of using a discrete basis in perceptron implementation is demonstrated, reducing computational complexity and enabling efficient processing

of limited signal states. It is shown that entropy use reduces model sensitivity to absolute signal values. Emulation of logical functions (AND, OR, XOR, NAND, NOR, XNOR) based on the proposed perceptron confirms the capability of processing linearly inseparable sets as well as the universality of the approach for constructing logical components in discrete space.

A method for reducing the decision space for the discrete perceptron is proposed via analysis of unique state matrices for two binary inputs under a fixed activation threshold, reducing the search space to 35% of the full solution space. The impact of activation function choice as a comparison operation ($>$, $<$, $=$, $\neq$) is studied, showing that the choice of activation function combined with appropriate bias coefficients allow adapting the perceptron to different logical function types and combinations.

The **third** chapter develops a model of the discrete perceptron based on a multilayer classical neural network, enabling the application of standard neural network machine learning methods to determine bias coefficients. Smooth approximations of indicator and threshold functions (based on hyperbolic secant and logistic functions) are proposed, making the discrete perceptron compatible with gradient-based learning.

Replacing non-smooth functions with differentiable approximations ensures gradient continuity and enables optimization in discrete signal structures. A modified threshold function is developed that works with both positive and negative thresholds, allowing upper and lower activation bounds based on the number of unique shifted input signal values. It is demonstrated that the discrete perceptron can serve as a multifunctional logic element capable of realizing a broad range of logical operations by selecting appropriate activation functions and bias coefficients.

It is shown that the classical perceptron structure remains adaptable for use with shifted signals, but no longer requires multiplication, greatly reducing the computational complexity of the input component. Cascade-conditional and logic-conditional algorithms for determining the number of unique states are developed. Comparative analysis reveals that logic-based implementations have lower time

complexity, improve hardware/software efficiency, and scale better with input size. Logical expressions for 3- and 4-input perceptrons are derived, minimizing conditional constructs in algorithmic and hardware implementations.

The **fourth** chapter presents the development of a digital component, `shift2b`, a parallel 2-bit adder performing hardware-level synaptic signal shifting via addition, reducing hardware platform requirements and achieving a delay not exceeding 6 ns for all logic paths.

A digital component `cmp2b` for comparing two 2-bit values is developed, with a simple structure and a delay not exceeding 6 ns, aligning with `shift2b` for operation at up to 166 MHz. Another component, `cnts`, counts the number of unique values among shifted synaptic signals using three `cmp2b` comparators and base logic, producing a 2-bit result code with a delay of no more than 9.6 ns —supporting operation up to 105 MHz.

A hardware structure for a two-input discrete perceptron is developed, incorporating the `shift2b`, `cmp2b`, and `cnts` components for input signal processing, intermediate comparison, and information aggregation. These components are optimized for minimal hardware costs and high speed. An integrated module is built to implement entropy-based classification logic, emulating basic logic functions: OR, AND, XOR, NOR, NAND, XNOR. In all cases, output signals matched the truth tables, confirming the design’s universality and suitability for digital logic circuit implementation.

A modular hardware structure supporting pipelining is developed to increase system clock speed. Clearly defined block boundaries enable functional scalability without altering logical connections, ensuring adaptability for more complex FPGA configurations.

Keywords: perceptron; binary signals; probabilistic estimates; logic functions; discrete signals; computer components; field-programmable gate arrays (FPGA); neural networks; machine learning.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

Наукові праці, в яких опубліковано основні результати дисертації:

1. Мельничук С. І., Яковин С. В. Ідентифікація об'єктів (сигналів) за оцінками інформаційної ентропії їх двомірних монохромних зображень. Науковий вісник Національного гірничого університету. №3 (147) 2015. – с.131-137. ISSN: 22232362 20712227 (**Видання, що індексується в базі Scopus**). *Здобувач здійснив комп'ютерне моделювання методу ідентифікації об'єктів за умов втрати інформації внаслідок спотворення зображень, а також моделювання впливу інверсії; С. Мельничук запропонував структурні, функціональні та алгоритмічні рішення на основі опрацювання даних, що представляють зображення .*

2. Спосіб реалізації перцептрона на основі імовірнісних характеристик зміщених синаптичних сигналів: пат.126753 Україна: G06N 20/00. / С. І. Мельничук, С. В. Яковин. – № а2019 01301; заявл. 08.02.2019; опубл. 25.01.23, Бюл. №4. 3 с. <https://sis.nipo.gov.ua/uk/search/detail/1719578/> (**Патент**). *Здобувач розробив структурну реалізацію дискретного перцептрону на основі імовірнісних характеристик зміщених синаптичних сигналів, а також переходу до дискретного базису; С. Мельничук запропонував концепцію організації перцептронних структур орієнтованих на опрацювання імовірнісних оцінок, зокрема інформаційної ентропії, синаптичних сигналів, а також приймав участь у розробці структури дискретного перцептрону.*

3. Яковин С. В., Мельничук С. І. Імплементация булевых функций в перцептронных структурах. Вісник Вінницького політехнічного інституту. 2024. Т.176, №5. С.48–55. (**Фахове видання України**). *Здобувач провів дослідження реалізації булевих функцій вищих порядків, на основі якого запропонував використання спрощеної функції оцінювання інформаційної ентропії та ймовірнісних індикаторів сигналів для підвищення розпізнавальної здатності перцептрону; С. Мельничук допоміг у валідації результатів та їх формальному описі.*

4. Yakovyn S. V., Melnychuk S. I. Discrete perceptron based on probabilistic estimates of shifted synaptic signals. *Naukovyi Visnyk Natsionalnoho Hirnychoho Universytetu*. 2025. No.2. P. 189–196. **(Видання, що індексується в базі Scopus)**. Здобувач провів дослідження дискретного перцептрону на основі ймовірнісних оцінок зміщених синаптичних сигналів, в якому замінено операції множення операціями додавання з подальшим оцінюванням кількості унікальних станів; С. Мельничук забезпечив методичний супровід, а також приймав участь у дослідженні характеристик функції помилки.

5. Яковин С. В., Мельничук С. І. Обмеження множини можливих рішень для дискретного перцептрона зі зміщеними синаптичними сигналами при емуляції бінарних логічних функцій. *Вісник Хмельницького національного університету*, №3.1, 2025 (351), – 295–301с. **(Фахове видання України)**. Здобувач провів дослідження обмеження множини можливих рішень для дискретного перцептрона з відхиленими синаптичними сигналами, а також моделювання в числовому експерименті логічних функцій для двох інформаційних входів; С. Мельничук здійснив теоретичний супровід, алгоритмічну підтримку, а також оцінив адекватність результатів моделювання.

6. Яковин С. В., Мельничук С. І. Апроксимація індикаторних функцій у контексті ймовірнісного дискретного перцептрона. *Вісник Хмельницького національного університету*, №3.2, 2025 (353), – 357–364с. **(Фахове видання України)**. Здобувач дослідив особливості апроксимації індикаторних функцій у ймовірнісному дискретному перцептроні, при якому використано статистичні оцінки дискретних синаптичних сигналів та гладкі диференційовані апроксимації активаційних функцій; С. Мельничук забезпечив методичний супровід, допоміг у валідації результатів та їх формальному описі.

7. Яковин С. В., Мельничук С. І., Мануляк І.З. Реалізація двовходового дискретного перцептрону зі зміщеними синаптичними сигналами на ПЛІС засобами AlteraHDL. *Вісник Вінницького політехнічного інституту*, №4, 2025,

– 186–194с. (*Фахове видання України*). Здобувач розробив алгоритмічні та схемні рішення дискретного перцептронну та його компонентів на основі каскадно-умовного та умовно-логічного підходів; С. Мельничук приймав участь у розробці структурних рішень, допоміг із формальним описом результатів; І. Мануляк приймала участь у розробці алгоритмічних та схемних рішень для платформи ПЛІС.

8. Melnychuk S., Kuz M., Yakovyn S. Emulation of logical functions NOT, AND, OR, and XOR with a perceptron implemented using an information entropy function. Proceedings of 14th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET), Lviv-Slavske, 20–24 February 2018. (*Тези конференції, що індексується в базі Scopus*). Здобувач презентував аспекти емуляції логічних функцій на основі перцептронну, що реалізує оцінювання інформаційної ентропії зважених синаптичних сигналів; С. Мельничук приймав участь у розробці аналітичного опису, структурних та алгоритмічних рішень; М. Кузь допоміг у валідації результатів та їх формальному описі.

9. Yakovyn S. V., Melnychuk S. I. Issues with looking semantically equivalent programming statements in java and ruby. Proceedings III International Scientific and Practical Conference Applied scientific and technical research, 3-5 April 2019. – Ivano-Frankivsk - "Ukraine Technical Sciences Academy" – 2019. – р.47. (*Тези конференції*). Здобувач презентував відмінності у використанні виразів мов програмування високого рівня, що виглядають семантично ідентичними в контексті імплементації нейронних мереж; С. Мельничук допоміг у валідації результатів та їх формальному описі.

10. Яковин С. В., Мельничук С. І. Особливості реалізації та функціональні обмеження перцептронних структур. Проблемно-наукова міжгалузева конференція "Інформаційні проблеми комп'ютерних систем, юриспруденції, енергетики, моделювання та управління". (ICSM-2023) – Надвірна, 20-21 листопада, 2023. С. 184–187. (*Тези конференції*). Здобувач презентував результати огляду особливостей реалізації та функціональних

обмежень структури перцептронів як базових компонентів нейронних мереж; С. Мельничук забезпечив методичний супровід та допоміг у валідації результатів та їх формальному описі.

11. Яковин С. В., Мельничук С. І., Мануляк І. З. Алгоритмічні реалізації дискретного перцептрону зі зміщеними синаптичними сигналами. Проблемно-наукова міжнародна конференція "Інформаційні проблеми комп'ютерних систем, енергетики, моделювання та управління". (ICSM-2025) – Надвірна, 5-6 червня, 2025. С. 110–113. **(Тези конференції)**. Здобувач презентував розроблені алгоритми, що описують функціонал дискретного перцептрону зі зміщеними синаптичними сигналами; С. Мельничук допоміг у валідації результатів та їх формальному описі; І. Мануляк приймала участь у розробці алгоритмічних та апаратних рішень для платформи ПЛІС.

12. Яковин С. В., Мельничук С. І. Порівняльне оцінювання обчислювальної складності реалізації кореляційного та дискретного перцептронів. Інформаційні технології в освіті, техніці та промисловості: Матеріали науково-практ. конф., м. Івано-Франківськ, 9 жовт. 2025 р. 2025. С. 317–319. **(Тези конференції)**. Здобувач презентував отримані результати порівняльного оцінювання обчислювальної складності типового та запропонованого перцептронів; С. Мельничук означив критерії оцінювання характерні для досліджуваних структур, а також допоміг у валідації результатів та їх формальному описі.

ЗМІСТ

	Стор.
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	17
ВСТУП.....	18
 РОЗДІЛ 1. МЕТОДИ ТА ЗАСОБИ РЕАЛІЗАЦІЇ КОМПОНЕНТІВ КОМП'ЮТЕРНИХ СИСТЕМ НА ОСНОВІ ПЕРЦЕПТРОННИХ СТРУКТУР	 25
1.1. Сучасний стан розробки та впровадження штучних нейронних компонентів комп'ютерних систем	25
1.2. Особливості імплементації логічних функцій у перцептронних структурах.	33
1.3. Перспективи вдосконалення та розвитку структури перцептрону, постановка задачі досліджень	43
ВИСНОВКИ ДО ПЕРШОГО РОЗДІЛУ	50
 РОЗДІЛ 2. ТЕОРЕТИЧНІ ОСНОВИ РЕАЛІЗАЦІЇ ДИСКРЕТНОГО ПЕРЦЕПТРОНА ЗІ ЗМІЩЕНИМИ СИНАПТИЧНИМИ СИГНАЛАМИ .	51
2.1. Дискретний перцептрон на основі імовірнісних оцінок зміщених синаптичних сигналів.....	51
2.2. Обмеження множини рішень для дискретного перцептрона при емуляції логічних функцій.....	61
ВИСНОВКИ ДО ДРУГОГО РОЗДІЛУ	71
 РОЗДІЛ 3. МОДЕЛІ ТА АЛГОРИТМИ РЕАЛІЗАЦІЇ КОМПОНЕНТІВ ДИСКРЕТНОГО ПЕРЦЕПТРОНУ	 72
3.1. Моделі функцій агрегації дискретного перцептрона на основі багатошарової класичної нейронної мережі	72
3.2. Апроксимація індикаторних функцій активації у контексті ймовірнісного дискретного перцептрона.....	79

3.3. Алгоритмічні рішення компонентів дискретного перцептрону зі зміщеними синаптичними сигналами	93
ВИСНОВКИ ДО ТРЕТЬОГО РОЗДІЛУ	102
 РОЗДІЛ 4. СХЕМНІ РІШЕННЯ ЦИФРОВИХ ЛОГІЧНИХ КОМПОНЕНТІВ НА ОСНОВІ ДИСКРЕТНОГО ПЕРЦЕПТРОНУ	103
4.1. Розробка схемних рішень компонентів бінарного двохвходового дискретного перцептрону	104
4.2. Реалізація бінарного двохвходового дискретного перцептрону для емуляції цифрових логічних компонентів	113
ВИСНОВКИ ДО ЧЕТВЕРТОГО РОЗДІЛУ	117
ЗАГАЛЬНІ ВИСНОВКИ ДО РОБОТИ	118
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	121
ДОДАТКИ	133

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

- ASIC – інтегральна схема для специфічного застосування
- Cnt – функція визначення кількості унікальних станів (значень)
- ELU – експоненційна лінійна функція активації (Exponential Linear Unit)
- GE – міра логічної площі спеціалізованих інтегральних схем
- GPU – графічний процесор (Graphic Processing Unit)
- N-M – нейроморфний (Neuromorphic) процесор
- ReLU – випрямлена лінійна функція активації (Rectified Linear Unit)
- SoftMax – функція м'якого максимуму
- TPU – тензорний процесор (Tensor Processing Unit)
- БФ – булева функція
- МФХ – модифікована функція Хевісайда
- ОРП – однорівневий перцептрон
- ПЛІС – програмована логічна інтегральна схема
- ФА – функція активації
- ФХ – функція Хевісайда
- ІНМ – штучна нейронна мережа

ВСТУП

Актуальність теми. Розвиток комп'ютерних систем значною мірою опирається на структурні рішення типових цифрових компонентів, однак на поточному етапі спостерігається їх інтеграція зі штучними нейронними мережами (ШНМ). Така ситуація зумовлена суттєвим розширенням можливостей цифрових компонентів ШНМ, а також потребами в інтелектуальному опрацюванні великих обсягів даних. Традиційно ШНМ будують на класичних перцептронних структурах, які покладаються на операції зваженого сумування та подальшу активацію, вимагають значних обчислювальних ресурсів (множення, нормування, центрування), що суттєво ускладнює їх імплементацію на апаратних платформах із жорсткими обмеженнями. В результаті виникає неузгодженість між зростаючими вимогами до автономності, енергоефективності та швидкодії та реальними можливостями апаратно-програмних засобів, що використовуються у промисловості, транспорті, комунікаціях тощо.

Комп'ютерні системи ґрунтуються на використанні дискретних та бінарних сигналів, що у задачах моніторингу, діагностування та керування вимагає спеціалізованих методів їх опрацювання, здатних забезпечити стійкість до шумів, інваріантність до інверсії та низьку чутливість до абсолютних значень вхідних параметрів. В такій ситуації використання інформаційно-статистичних мір як агрегуючих функцій у перцептронних структурах розширює можливості щодо подолання обмеженої виразної здатності класичного перцептрона. Крім того, перехід до дискретного базису та вилучення операцій множення дозволяють знизити апаратну та обчислювальну складність, що є критичним для побудови ефективних компонентів ШНМ у ресурсно-обмежених середовищах. Слід зазначити, що задача універсального відтворення логічних функцій у межах однієї перцептронної структури набуває особливої ваги в контексті конструювання цифрових логічних компонентів із можливістю подальшого навчання. Сучасні підходи до логічного синтезу часто не враховують потенціал нейромережових

елементів, тоді як моделі дискретного перцептрона на основі кількості унікальних станів зміщених синаптичних сигналів здатні об'єднати переваги цифрової логіки та механізмів машинного навчання ШНМ. Такий підхід може забезпечити гнучкішу архітектуру для адаптивних цифрових систем і скорочує простір перебору параметрів завдяки формальним методам обмеження множини рішень, що важливо для автоматизованого проєктування. Актуальність дослідження також зумовлена необхідністю створення зручних підходів до застосування градієнтних методів оптимізації у дискретних структурах із дискретною природою сигналів. В результаті, дискретний перцептрон із ймовірнісно-ентропійною агрегуючою функцією може стати перспективним базовим елементом для гібридних нейро-цифрових систем.

Дисертаційна робота присвячена вдосконаленню існуючих та розробці нових методів цифрового опрацювання дискретних сигналів у сфері штучних нейронних мереж, орієнтованих на зменшення обчислювальної та апаратної складності, підвищення швидкодії, а також універсальності нейрокомпонентів емуляції логічних функцій в комп'ютерних системах. Розв'язання цієї задачі дозволяє розширити теорію перцептронів у дискретному базисі, а також створювати ресурсоефективні цифрові компоненти для кіберфізичних систем, інтегрованих платформ і засобів цифрової логіки тощо.

Зв'язок роботи з науковими програмами, планами, темами.

Результати досліджень, що викладені у дисертаційній роботі, виконувались здобувачем відповідно до плану навчання в аспірантурі на кафедрі комп'ютерних систем і мереж (КСМ) Івано–Франківського національного технічного університету нафти і газу (ІФНТУНГ) згідно з тематикою виконання держбюджетної науково-дослідної роботи кафедри КСМ ІФНТУНГ "Наукові засади створення компонентів спеціалізованих комп'ютерних систем на основі методів та технологій цифрового опрацювання інформаційно-вимірювальних та комунікаційних сигналів для промислових і нафтогазових об'єктів", №0121U113385, період з 2021 до 2024 н.р., а також держбюджетної науково-дослідної роботи кафедри КСМ

ІФНТУНГ "Наукові засади створення цифрових компонентів спеціалізованих мереж та інформаційно-вимірювальних систем промислових об'єктів", №0124U004100, яка виконується в період з 2024 до 2027 н.р.

Здобувач залучався до виконання окремих підрозділів, пов'язаних з удосконаленням методів та засобів цифрового опрацювання дискретних сигналів в компонентах комп'ютерних систем, зокрема на основі ймовірнісного підходу.

Мета і завдання дослідження. Метою роботи є вирішення актуального науково-прикладного завдання у галузі інформаційних технологій – розробка методу реалізації дискретного перцептрону на основі ймовірнісних оцінок зміщених синаптичних сигналів, який забезпечує опрацювання лінійно-нероздільних множин, зниження алгоритмічної і апаратної складності та придатний до використання в комп'ютерних системах з обмеженими обчислювальними ресурсами.

Для досягнення зазначеної мети необхідно розв'язати такі **задачі**:

- проаналізувати сучасний стан розвитку перцептронних структур у комп'ютерних системах, оцінити їх обмеження щодо виразної здатності та обчислювальної складності;
- розглянути особливості імплементації логічних функцій у перцептронних структурах, а також перспективи їх вдосконалення та розвитку, зокрема в частині використання інформаційних мір;
- розробити дискретний перцептрон на основі ймовірнісних оцінок зміщених синаптичних сигналів, а також означити критерії його активації;
- дослідити обмеження множини можливих значень коефіцієнтів зміщення для дискретного перцептрона при емуляції логічних функцій, а також вплив різних порогових та індикаторних функцій активації на реалізацію логічних операцій;
- здійснити розробку моделі функцій агрегації дискретного перцептрона на основі багатоваріантної ШНМ для застосування градієнтного методу

машинного навчання, а також запропонувати гладкі апроксимації негладких функцій активації (індикаторної, порогової);

- розробити алгоритмічні рішення компонентів дискретного перцептронну зі зміщеними синаптичними сигналами на основі каскадно-умовного та умовно-логічного підходів;

- розробити схемні рішення бінарного двохвходового дискретного перцептронну для емуляції цифрових логічних компонентів та його структурних компонентів, провести їх функціональне та інструментальне моделювання для платформи ПЛІС.

Об'єкт дослідження: процес опрацювання дискретних сигналів перцептронними структурами штучних нейронних мереж.

Предмет дослідження: методи та засоби емуляції логічних функцій в перцептронах за імовірнісними оцінками зміщених бінарних сигналів.

Методи дослідження: основні наукові результати і висновки зроблено на основі використання теорії сигналів і систем, теорії ймовірностей та математичної статистики, методів статистичного аналізу, цифрового опрацювання сигналів, а також за результатами моделювання в обчислювальному експерименті.

Наукова новизна отриманих результатів:

- вдосконалено структурну реалізацію класичного перцептронну шляхом заміни операцій зваженого сумування вхідних синаптичних сигналів на операції оцінювання їх інформаційної ентропії, що дозволило емулювати повний набір логічних функцій в межах однієї перцептронної структури;

- вперше запропоновано реалізацію дискретного перцептронну на основі імовірнісних оцінок зміщених синаптичних сигналів, у якому, на відміну від відомих перцептронних структур, операція множення замінена на операцію додавання (зміщення), а функція агрегації визначає кількість унікальних значень зміщених синаптичних сигналів, що дозволило перейти до дискретного базису, забезпечити опрацювання лінійно-нероздільних множин та спростити алгоритмічну і апаратну реалізацію;

- вперше розроблено модель емуляції дискретного перцептрона на основі багат шарової класичної нейронної мережі, що дозволяє застосувати типові методи машинного навчання для визначення коефіцієнтів зміщення запропонованого перцептрона;

- набула подальшого розвитку теорія та методи побудови перцептронів, які ґрунтуються на використанні імовірнісного оцінювання синаптичних сигналів у дискретному базисі і дозволяють створювати ресурсоефективні цифрові компоненти кіберфізичних систем.

Практичне значення отриманих результатів:

- розроблено каскадно-умовний та умовно-логічний алгоритми компоненту визначення кількості унікальних станів і на основі порівняльного аналізу встановлено, що реалізації на основі логічних функцій, замість каскадних умов, мають нижчу часову складність, дозволяють покращити ефективність схемної/програмної реалізації та масштабованість на більшу кількість входів;

- отримано логічні вирази для визначення кількості унікальних станів для 3-ох та 4-ох входових перцептронів, що дозволяє мінімізувати кількість умовних конструкцій при алгоритмічній та апаратній реалізаціях дискретних компонентів такого типу;

- розроблено схемні рішення цифрових компонентів перцептрона, зокрема паралельний двобітовий суматор, що виконує апаратне зміщення синаптичних сигналів та компаратор двох двобітових значень, затримка операції яких не перевищує 6 нс, що забезпечує можливість роботи на частотах до 166 МГц в асинхронному режимі;

- розроблено схемні рішення цифрового компоненту агрегації перцептрону, що реалізує підрахунок кількості унікальних значень серед зміщених синаптичних сигналів, затримка операції не перевищує 9.6 нс, що дозволяє працювати на частотах до 105 МГц;

- розроблено схемні рішення інтегрованого модуля дискретного перцептрону, який реалізує логіку класифікації за ентропійною ознакою і

емулює базові логічні функції: OR, AND, XOR, NOR, NAND, XNOR, що підтверджує універсальність запропонованих рішень та їх придатність для реалізації універсальності цифрових нейрокомпонентів логічних схем.

Особистий внесок здобувача. Основні положення, теоретико-розрахункові, аналітичні, експериментальні дані та результати дисертаційної роботи одержані автором самотійно. В опублікованих у співавторстві роботах здобувачу належить:

- комп'ютерне моделювання методу ідентифікації об'єктів за умов втрати інформації внаслідок спотворення зображень, а також моделювання впливу інверсії [48];

- розроблення структурної реалізації дискретного перцептрону на основі ймовірнісних характеристик зміщених синаптичних сигналів, а також переходу до дискретного базису [52];

- теоретичне дослідження реалізації булевих функцій вищих порядків, на основі якого запропоновано використання спрощеної функції інформаційної ентропії та ймовірнісних індикаторів сигналів для підвищення розпізнавальної здатності перцептрону [35];

- теоретичне дослідження дискретного перцептрону на основі ймовірнісних оцінок зміщених синаптичних сигналів, в якому замінено операції множення операціями додавання з подальшим оцінюванням кількості унікальних станів [51];

- теоретичне дослідження обмеження множини можливих рішень для дискретного перцептрона з відхиленими синаптичними сигналами, а також моделювання в числовому експерименті логічних функцій для двох інформаційних входів [64];

- дослідження апроксимації індикаторних функцій у ймовірнісному дискретному перцептроні, при якому використано статистичні оцінки дискретних синаптичних сигналів та гладкі диференційовані апроксимації активаційних функцій [78];

- розроблення алгоритмічних та схемних рішень дискретного перцептрону та його компонентів на основі каскадно-умовний та умовно-логічний підходів [110].

Апробація результатів дисертації. Основні положення дисертаційної роботи доповідалися та обговорювалися на 3 міжнародних та 2 всеукраїнських науково-технічних конференціях:

- 14th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET), Lviv-Slavske, 20–24 February 2018;

- III International Scientific and Practical Conference Applied scientific and technical research, 3-5 April 2019. – Ivano-Frankivsk - "Ukraine Technical Sciences Academy";

- проблемно-наукова міжгалузева конференція "Інформаційні проблеми комп'ютерних систем, юриспруденції, енергетики, моделювання та управління". (ICSM-2023) – Надвірна, 20-21 листопада, 2023;

- проблемно-наукова міжнародна конференція "Інформаційні проблеми комп'ютерних систем, енергетики, моделювання та управління". (ICSM-2025) – Надвірна, 5-6 червня, 2025.

- науково-практична конференція "Інформаційні технології в освіті, техніці та промисловості", Івано-Франківськ, 9 жовтня 2025.

Публікації. Основні результати досліджень викладено у 12 публікаціях, з яких 6 статей, в т. ч. 6 у фахових наукових виданнях України в т. ч. 2, що реферуються наукометричними базами: SCOPUS, 1 патент України на винахід, а також 5 тез доповідей у матеріалах конференцій, в т.ч. міжнародних та 1, що реферується наукометричними базами: SCOPUS.

Структура та обсяг дисертації. Дисертація складається зі вступу, чотирьох розділів, висновків та додатків. Матеріали викладені на 148 сторінках; робота містить 29 таблиць, 34 рисунки, список використаних джерел зі 110 найменуваннями на 12 сторінках та 4 додатки на 16 сторінках.

РОЗДІЛ 1

МЕТОДИ ТА ЗАСОБИ РЕАЛІЗАЦІЇ КОМПОНЕНТІВ КОМП'ЮТЕРНИХ СИСТЕМ НА ОСНОВІ ПЕРЦЕПТРОННИХ СТРУКТУР

1.1 Сучасний стан розробки та впровадження штучних нейронних компонентів комп'ютерних систем

На сучасному етапі розвитку інформаційних технологій активно впроваджуються спеціалізовані рішення, спрямовані на автоматизацію процесів та заміну людського фактору у виконанні задач низької інтелектуальної складності. Це не тільки підвищує продуктивність та ефективність виробничих та адміністративних процесів, але й дозволяє зменшити помилки, зазвичай пов'язані з людським фактором.

Одним із провідних напрямів у цій області є імплементація нейронних мереж, які можуть бути реалізовані як на апаратно-програмному рівні, так і на чисто програмному, забезпечуючи гнучкість та адаптивність рішень.

Застосування нейронних мереж у промисловості та бізнесі демонструє ряд переваг, зокрема у виробничому секторі вони використовуються для прогнозування поломок обладнання, оптимізації логістичних процесів та підвищення якості продукції; у сфері обслуговування – ефективна заміна операторів кол-центрів, менеджерів з продажів через мережу Інтернет, тощо.

Слід звернути увагу на той факт, що будь-яка нейромережа структурно будується на основі перцептронів, які організовуються окремими шарами, причому зв'язки між ними та вагові значення цих зв'язків встановлюються в ході навчання, див. рис. 1.1 [1, 2, 3, 4, 5]. В загальному випадку розглядають три типи шарів: вхідний, прихований та вихідний. Вхідний шар отримує первинні дані, приховані шари виконують опрацювання даних, а вихідний шар формує реакцію (результат). Структурно перцептрони у цих шарах з'єднані з перцептронами наступного шару через зв'язки. Причому вагові значення

з'єднань є ключовими у процесі навчання нейронної мережі. В ході навчання, яке часто відбувається на основі різних методів, зокрема методом зворотного поширення помилки, мережа постійно коригує ваги зв'язків, щоб мінімізувати розходження між фактичним виходом мережі та очікуваним результатом.

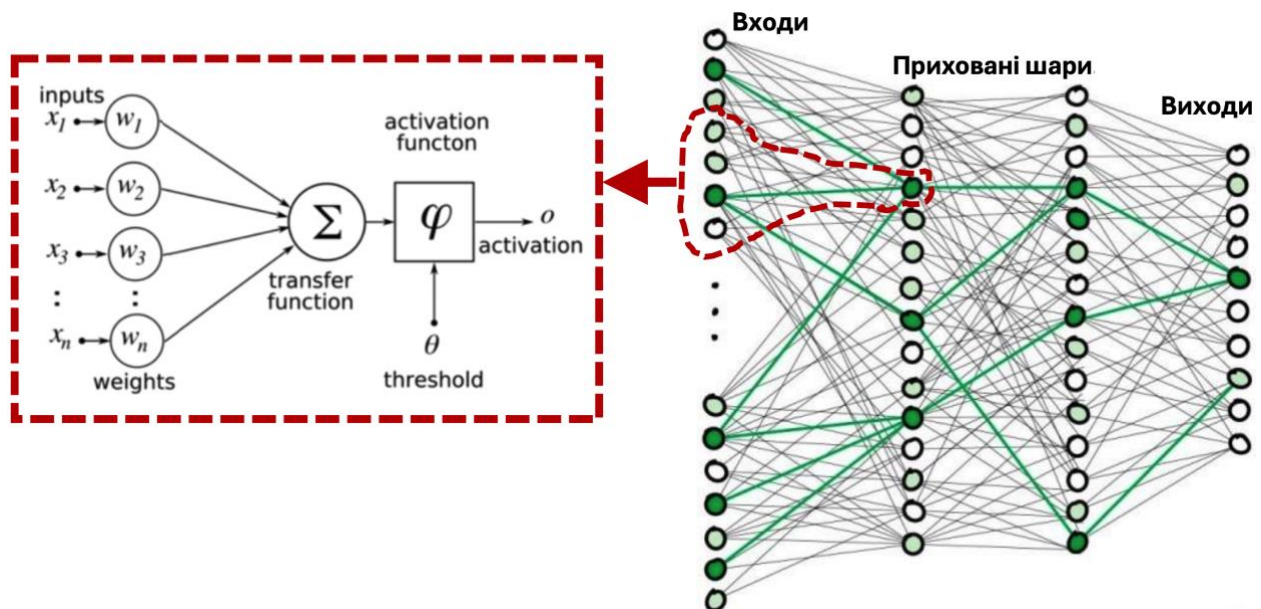


Рисунок 1.1 – Структура типового перцептрона

Таким чином, ефективність нейронних мереж значною мірою залежить від їх архітектури, включаючи кількість шарів, кількість перцептронів у кожному шарі, та спосіб, яким ці шари взаємодіють. Крім того, не слід забувати, що фундаментом будь-якої мережі є перцептрон, робота якого полягає у взятті вхідних сигналів, формуванні їх зваженої суми та виробленні, на основі цієї суми, вихідного сигналу шляхом використання функції активації (ФА), зокрема порогової функції.

Загалом, структура перцептронів у нейронній мережі є ключовим фактором, який визначає її здатність ефективно вирішувати різноманітні задачі, адаптуватися до змінних умов та навчатися на основі досвіду. Таким чином, є постійна потреба у вдосконаленні наявних та створенні нових перцептронних структур, які б дозволили розширити функціонал нейромереж, зокрема на основі застосування ймовірнісних оцінок вхідних сигналів.

Реалізація перцептронів ґрунтується на використанні типової архітектури: синапси (вхідні сигнали) – агрегація (найчастіше зважена сума вхідних сигналів) – реакція (формування вихідного сигналу на основі ФА). Основна мета моделі однорівневого перцептрона полягає в аналізі лінійно роздільних об'єктів з бінарними результатами. Тобто, однорівневі перцептрони можуть розпізнавати лише лінійно роздільні шаблони. Для задачі класифікації з деякою ФА кроку один вузол матиме одну лінію, яка розділяє точки даних, що формують шаблони.

В загальному випадку, типова структура одношарового перцептрону реалізується на основі суми послаблених і/або посиленних вхідних сигналів ($x_1, x_2, \dots, x_n$), що реалізується на основі операції множення на відповідні вагові коефіцієнти ($w_1, w_2, \dots, w_n$) з подальшим використанням активаційної функції φ для внесення нелінійності, рис 1.1.

Слід зазначити, що функція активації зазвичай передбачає наявність деякого порогового значення θ перевищення якого і є причиною активації перцептрона. Слід розуміти, що без використання функції активації побудова багатошарових мереж неможлива, так як така мережа може бути зведена до одношарового перцептрона.

У загальному випадку одношаровий перцептрон є частковим випадком функції $f: R^n \rightarrow R$. Оскільки перцептрон описується не тільки входами, але й ваговими коефіцієнтами, то $y = f(x, w)$. Проте, зазвичай, використовують більш спеціалізовану форму представлення перцептрону [6, 7, 8, 9, 10]:

$$y = \varphi \left(\sum_{i=1}^n w_i x_i + b \right) = \varphi(w^T x + b) \quad (1.1)$$

де φ – функція активації.

Фактично можна сказати, що перцептрон реалізує одну з типових операцій теорії цифрового опрацювання сигналів, а саме кореляції (correlation) або згортки (convolution). Єдиною відмінністю чи доповненням є

використання функції активації, яка визначає рівень активності перцептрону. В такій ситуації варто розглянути різні варіанти реалізації кореляційних функцій, а не тільки типову, яка ґрунтується на операції множення вектору значень вхідних синаптичних сигналів на вектор вагових коефіцієнтів, а саме [11, 12, 13, 14, 15, 16]:

$$K_{xw}(j) = \frac{1}{n} \cdot \sum_{i=1+j}^{n+j} x_i \cdot w_{i+j} \quad (1.2)$$

Реалізація коваріаційної функції (1.2) фактично реалізує операцію перетворення та агрегації класичного перцептрона за винятком усереднення $1/n$.

$$R_{xw}(j) = \frac{1}{n} \cdot \sum_{i=1+j}^{n+j} x_i \circ w_{i+j} \quad (1.3)$$

Кореляційна функція (1.3) порівняно із (1.2) містить суттєве ускладнення, а саме операцію центрування (вилучення постійної складової) векторів вхідних сигналів та вагових коефіцієнтів, що призводить до додаткових обчислювальних затрат.

$$\rho_{xw}(j) = \frac{R_{xw}(j)}{D_x D_w} \quad (1.4)$$

Нормована кореляційна функція (1.4) додатково ускладнює формулу (1.3) необхідністю розрахунку дисперсії та операцією ділення.

$$H_{xw}(j) = \frac{1}{n} \cdot \sum_{i=1+j}^{n+j} \text{sign}(x_i) \cdot \text{sign}(w_{i+j}) \quad (1.5)$$

Знакова функція (1.5) виглядає спрощеною по відношенню до (1.2), оскільки фактично маніпулює бітовою ознакою знаку, однак формування векторів бітових ознак ґрунтується на центрованих векторах вхідних сигналів та вагових коефіцієнтів, що як і у випадку з (1.3) призводить до зростання обчислювальних затрат.

$$B_{xw}(j) = \frac{1}{n} \cdot \sum_{i=1+j}^n x_i \cdot \text{sign}(w_{i+j}) \quad (1.6)$$

Релейна функція (1.6) реалізована як поєднання (1.3) та (1.5), що фактично представляє вектор вагових коефіцієнтів у знаковому вигляді, що зумовлює спрощення (1.3), однак потребує використання громіздкої операції центрування і є складнішою для обчислень у порівнянні із (1.2).

$$C_{xw}(j) = \frac{1}{n} \cdot \sum_{i=1+j}^{n+j} (x_i - w_{i+j})^2 \quad (1.7)$$

Структурна функція (1.7) ґрунтується на операції віднімання, що порівняно із (1.2) є суттєвим спрощення, однак піднесення отриманого результату до квадрату, що фактично нівелює знак результату операції, призводить до суттєвих обчислювальних ускладнень.

$$G_{xw}(j) = \frac{1}{n} \cdot \sum_{i=1+j}^{n+j} |x_i - w_{i+j}| \quad (1.8)$$

Модульна кореляційна функція (1.8) у порівнянні із (1.7) фактично заміняє піднесення до квадрату знаходженням модулю, що дозволяє отримати сумірний результат, однак знижує обчислювальні затрати.

$$F_{xw}(j) = \frac{1}{n} \cdot \sum_{i=1+j}^{n+j} Z_{ij}, \quad Z_{ij} = \begin{cases} x_i, & x_i \leq w_{i+j} \\ w_{i+j}, & x_i > w_{i+j} \end{cases} \quad (1.9)$$

Кореляційна функція еквівалентності (1.9) типово реалізується на операціях порівнювання, зокрема відомі реалізації більше з двох та менше з двох. Такий підхід характеризується низькою обчислювальною складністю, однак її кореляційні властивості є найслабшими серед згаданих функцій кореляції.

Спільним недоліком більшості розглянутих варіантів реалізації функцій кореляції є зростання обчислювальної складності внаслідок необхідності виконання додаткових математичних операцій, зокрема таких як центрування, нормування, піднесення до квадрату, обчислення модуля чи виконання багатократних логічних порівнянь. Тобто їх використання приводить до додаткового ускладнення апаратної та алгоритмічної реалізації перцептрона і, як наслідок, збільшує затрати часу та ресурсів на опрацювання вхідних сигналів.

Таким чином, в основі одношарового перцептрону лежить функція кореляції/згортки вхідного сигналу з ваговими коефіцієнтами, яка доповнена функцією активації. Причому, інші структури, які ґрунтуються на відмінних способах опрацювання вхідних сигналів практично відсутні. Крім того, вхідний сигнал найчастіше представляється амплітудними значеннями (контрольованих параметрів), що не дозволяє оцінити інші інформативні складові таких сигналів.

Як уже зазначалось, функція активації є важливим компонентом у структурі перцептрона, оскільки на її основі реалізується механізм навчання, а також вноситься нелінійність у процес опрацювання вхідних сигналів. Часто вибір функції активації залежить від досвіду та інтуїції розробника, що надає цьому вибору суб'єктивного характеру. Невідповідний вибір функції може вплинути на продуктивність та ефективність роботи перцептрона, крім того, функції активації разом із їхніми першими похідними є ключовими чинниками

обчислювальної складності мережі, тобто обчислювальних витрат на оптимізацію параметрів в ході машинного навчання [17, 18, 19, 20]. Отже, слід розглянути функціональні можливості та обмеження найбільш вживаних функцій активації ще й тому, що вони впливають на здатність мережі апроксимувати цільові функції. Підсумовуючи дослідження проведене у [17, 18, 19, 20] можна виділити наступні класи активаційних функцій: сигмоїдні функції (S-подібні функції), кусочно-лінійні функції та інші функції.

Розглянемо їх детальніше. До сигмоїдних функцій відносять логістичну (сигмоїдну) функцію, арктангенс, гіперболічний тангенс та softsign.

Логістична (або сигмоїдна) функція $f(z) = \frac{1}{1 + e^{-z}}$ може розглядатися як гладка версія бінарної активації. Її перевага над бінарною функцією активації полягає в тому, що вона нескінченно диференційована на всій області визначення.

Арктангенс $f(z) = \arctan(z)$.

Гіперболічний тангенс $f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$.

Softsign $f(z) = \frac{1}{1 + |z|}$. Її перевагою над іншими сигмоїдними та багатьма несигмоїдними функціями є низька обчислювальна складність.

До кусочно-лінійних функцій відносять лінійну, ReLU та Leaky ReLU функції. Їхньою перевагою є низька обчислювальна складність та те, що вони однорідні для невід'ємних помножників: $f(az) = af(z)$, $a \in [0, \infty)$, $z \in R$.

Лінійна $f(z) = z$ – має суттєвий недолік, що не дозволяє її використання для апроксимації нелінійних цільових функцій.

ReLU – $f(z) = \max\{0, z\}$.

Leaky ReLU – $f(z) = \max\{0, z\} + \min\{0, az\}$, де $a \in [0; \infty)$.

Інші функції розглянуті в [17, 18, 19, 20] це Softplus та адаптивні (наприклад адаптивна поліноміальна активаційна функція [21]).

Softplus (гладкий варіант ReLU) – $f(z) = \ln(1 + e^z)$. Вона є диференційована у 0, проте в силу обчислювальної складності перевага віддається ReLU.

Проте поза увагою залишилися функція Гауса $f(z) = e^{-(z-c)^2}$ та інші радіальні базисні функції, такі як мультікватрична, інверсна мультікватрична та їх комбінації.

Як можна побачити, серед розглянутих функцій активації є функції неперервної змінної, що фактично дозволяє розглядати процес машинного навчання як перцептрона так і нейромережі, як пошук деяких екстремумів багатомірної поверхні. Практично досягнення прийнятної адекватності перцептрона чи нейромережі потребує значних обчислювальних ресурсів, оскільки це напряду пов'язано з точністю визначення вагових коефіцієнтів яка фактично необмежена. Однак для вирішення ряду задач може бути достатнім використання дискретних множин як сигналів так і вагових коефіцієнтів, що суттєво зменшує кількість можливих варіантів параметрів перцептрону. Проблемним питанням такого підходу може стати відсутність проміжних значень, тобто простір рішень перестає бути неперервним, що негативним чином буде впливати на процес машинного навчання, що зокрема ґрунтується на методах градієнтного спуску.

Однією з головних переваг використання одношарових перцептронів є їх простота та ефективність. Вони легко реалізуються, навчаються та розуміються. У них є чітке геометричне тлумачення як гіперплощина, яка розділяє два класи даних. Однією з головних недоліків використання одношарових перцептронів є їх обмежена виразність та здатність до узагальнення. Вони не можуть навчитися класифікувати лінійно нероздільні шаблони, такі як XOR, кола або спіралі. Вони також схильні до перенавчання та шуму, оскільки намагаються провести пряму лінію крізь дані.

1.2 Особливості імплементації логічних функцій у перцептронних структурах

Реалізація перцептронних структур ґрунтується на використанні типової архітектури: синапси (вхідні сигнали) – агрегація (найчастіше зважена сума вхідних сигналів) – реакція (формування вихідного сигналу на основі ФА). Основна мета моделі однорівневого перцептрона (ОРП) полягає в аналізі лінійно роздільних об'єктів з бінарними результатами. Тобто, однорівневі перцептрони можуть розпізнавати лише лінійно роздільні шаблони.

Для задачі класифікації з деякою ФА один вузол матиме одну лінію, яка розділяє точки даних, що формують шаблони. Як результат, одношарові нейронні мережі виявляються обмеженими в класифікаційних задачах, де необхідно геометрично розділити набори точок. У випадку двох входів такий роздільник представляє собою пряму лінію, у випадку трьох входів - це площина в тривимірному просторі, а при чотирьох і більше входах розділення відбувається за допомогою "гіперплощини" в просторі більш високої розмірності, яку складно візуалізувати. З врахуванням того, що одношарові перцептрони обмежені лінійною роздільністю, важливо встановити, чи можна розділити дані за допомогою ліній. Однак, немає простого способу визначити це, особливо коли кількість змінних велика. Зокрема, нейрон з n бінарними входами може мати 2^n різних комбінацій вхідних станів, а оскільки кожен вхідний набір може призвести до одного з двох виходів, існує 2^{2^n} можливих функцій з n змінними. Як можна побачити з табл.1.1, імовірність випадково вибрати лінійно роздільну функцію є порівняно низькою навіть при помірній кількості змінних.

Таким чином, однорівневий перцептрон не має можливості розділити лінійно-нероздільні множини, що підтверджено багатьма дослідженнями у цій сфері. Так, Ху (2007) розглянув обмеження адитивної моделі, використовуваної в ОРП, коли вхідні змінні не є незалежними, і запропонував перцептрон на основі нечіткого інтегралу, який демонструє кращу

продуктивність порівняно з традиційними ОРП у задачах класифікації патернів [22].

Таблиця 1.1 – Залежність кількості лінійно роздільних функцій від кількості вхідних параметрів

n	Загальна кількість функцій (2^{2^n})	Кількість лінійно роздільних функцій
1	4	4
2	16	14
3	256	104
4	65 536	1 882
5	4294967296	94 572
6	$1,844 \cdot 10^{19}$	15 028 134

Жонг (2001) представив інтегровану мережу перцептронів для розпізнавання багатокласових лінійно роздільних шаблонів і підтвердив, що ОРП не можуть розпізнавати нелінійно роздільні шаблони [23]. Ху (2010) зауважив, що припущення щодо традиційних ФА про те, що немає взаємозалежності між атрибутами, не завжди відповідає дійсності, тим самим підсилюючи лінійне обмеження ОРП [24]. Додаткові докази подані працях Сінгха та ін. (2014), які представили мемристивний перцептрон для класифікації лінійно роздільних шаблонів, визнаючи, що ОРП непридатні для нелінійно роздільних шаблонів [25].

Крім того, обмеження ОРП до лінійної роздільності вивчаються різними авторами, які пропонують поліпшені або альтернативні моделі для класифікації нелінійно роздільних шаблонів або покращення здатності класифікації ОРП шляхом розгляду поліпшених особливостей або функцій активації, а також використання різних алгоритмів, таких як генетичні алгоритми, для оптимізації процесу навчання та продуктивності ОРП, зокрема:

- Ху і Ценг (2007) запропонували використання нечітких інтегралів для поліпшення ОРП [26];

- Рамчун та ін. (2016) та Рамчун та ін. (2019) досліджували вплив вибору архітектури та типів функцій активації на збіжність та продуктивність багатошарових перцептронів, вказуючи на обмеження однорівневих архітектур [27, 28].

- Мельничук, Яковин, Кузь (2018) досліджували використання функції інформаційної ентропії для емуляції логічних функцій за допомогою лише одного ОРП, продемонстрували можливість емулювати повний набір логічних функцій (NOT, AND, OR, і XOR) та вказали на те, що функціональні можливості перцептрона можуть бути розширені за допомогою ймовірнісних індикаторів сигналів, що генеруються сенсорами елементів S перцептрона [29]. Основні поліпшення щодо класифікації нелінійно роздільних шаблонів, можна розділити на такі групи:

1. Використання функції інформаційної ентропії: автори підходять до використання інформаційної ентропії як способу збільшення розпізнавальної здатності перцептрона. Функція ентропії оцінює розподіл ймовірностей виходів, що дозволяє перцептрону краще адаптуватися до різних шаблонів вхідних даних.

2. Розширення функціональних можливостей за допомогою індикаторів сигналу: додаткове використання ймовірнісних індикаторів сигналів може допомогти в розширенні можливостей перцептрона за рахунок додавання додаткової інформації про стан сенсорів.

3. Емуляція комплексних логічних функцій: автори також показали, що за допомогою запропонованого підходу можна реалізувати складні логічні функції, такі як XOR, які традиційно вважаються складними для класичних однорівневих перцептронів через їх нелінійну роздільність.

Такі покращення можуть бути важливими для вдосконалення наявних чи розробки нових перцептронних структур, здатних виконувати більш складні задачі класифікації, ніж ті, для яких вони традиційно використовувалися, і відкривають нові можливості для застосування перцептронів у сучасних системах штучного інтелекту.

Таким чином, результати проведеного аналізу показують, що хоча ОРП є ефективні для аналізу лінійно роздільних об'єктів з бінарними результатами, вони дійсно нездатні розпізнавати лінійно нероздільні шаблони, що потребує використання більш складних мереж, як-от багат шарові перцептрони, для таких завдань, або застосування модифікованих перцептронних структур, тобто тих, у яких задіяні степеневі функції замість функцій активації від суми зважених входів або радіальних чи інших типів функцій активацій. Проте поза увагою, за винятком [29], залишаються інші функції, зокрема статистичні, які можна застосувати для вирішення задач класифікації [30].

Окремим аспектом функціонування засобів класифікації є спрощена задача, що зводиться до опрацювання бінарних векторів (матриць), яка виникає при класифікації зображень, таких як розпізнавання рукописного тексту, об'єктів, явищ тощо [31].

Для кращого розуміння сучасного контексту досліджень, цей розділ включатиме дані про існуючі моделі розпізнавання, їх розміри та точність. Зокрема, нейронні мережі були застосовані до розпізнавання 2D-шаблонів на основі їх форми, де різні архітектури, включаючи багаторівневі перцептрони (БРП), мережі самоорганізації Кохонена та гібридні структури, призвели до різної точності розпізнавання та класифікації [32]. Зокрема, структуровані нейронні мережі виявилися ефективними для класифікації складних шаблонів та потенційно небезпечних ситуацій у системах спостереження, перевершуючи класичні статистичні процедури класифікації [33]. Щодо статистичного розпізнавання шаблонів, застосування функціональних мереж із статистичними підходами продемонструвало обнадійливі результати в прикладних застосунках, пропонуючи надійні та ефективні техніки в задачах пошуку даних, web-пошуку а також вилучення мультимедійних даних з контенту [34]. Для оцінки ефективності одношарових перцептронів у задачах опрацювання бінарних сигналів доцільно застосувати такі критерії:

- максимальний відсоток булевих функцій (БФ), які обрана структура зможе відтворити;

- мінімальна кількість параметрів (ваг), необхідна для моделювання;
- обчислювальна складність розрахунку реакції.

На цьому етапі з аналізу виключена оцінка обчислювальної складності процедури навчання перцептрону, а увагу зосереджено на здатності перцептронних структур відтворити БФ та на обчислювальній складності їхньої реакції [35].

Для спрощення аналізу використаємо булеві функції (1.10), оскільки вони мають невелику вхідну розмірність, проте дозволяють проаналізувати поведінку різних типів перцептронів на здатність апроксимувати всі можливі функції для вибраної кількості вхідних параметрів. Слід зауважити, що у випадку n вхідних змінних, кількість можливих комбінацій складатиме 2^n , а кількість можливих функцій: 2^{2^n} .

$$y = B_l(\bar{x}) \quad (1.10)$$

де $\bar{x} = (x_1, x_2, \dots, x_n)$ – вектор вхідних значень БФ,

l – порядковий номер булевої функції, $1 \leq l \leq 2^{2^n}$.

У цьому контексті важливим аспектом аналізу є визначення мінімальної кількості параметрів, або "ваг", необхідних для перцептронної структури щоб відтворити будь-яку з можливих БФ з означеного вище набору. Це вимагає оптимізації структури перцептрона, щоб була можливість забезпечити точне моделювання з мінімальними вимогами до обчислювальних ресурсів, що є критичним для практичного використання у великих масштабах, наприклад, у домені машинного навчання чи складних системах опрацювання даних.

Моделювання реалізовано за допомогою функції (1.11), що приймає як вхідний булевий вектор так і параметри, які зафіксовані для кожної окремої логічної функції B_l .

$$y = F(\bar{x}, \bar{p}_l) \quad (1.11)$$

де $\bar{p}_l = (p_1, p_2, \dots, p_m)$ – вектор параметрів "ваг" перцептрону.

Для оцінювання достатньої (мінімально необхідної) кількості параметрів необхідно скористатись такими функціями:

$$E(\bar{x}) = \sum_{i=0}^{n-1} 2^i x_i = \bar{v}\bar{x} = q \quad (1.12)$$

де $\bar{v} = (2^0, 2^1, \dots, 2^{n-1})$

$$G(q, \bar{p}_l) = F(\bar{x}, \bar{p}_l) \quad (1.13)$$

$$G(q, \bar{p}_l) = \sum_{j=0}^{2^n-1} p_{j+1} \theta(q - j) \quad (1.14)$$

де $p_j = y_j - y_{j-1}$,

$y_0 = 0$, $\bar{y} = (y_1, y_2, \dots, y_{2^n})$ – усі послідовні значення булевої функції B_l ,

θ – функція Хевісайда.

$$F(\bar{x}, \bar{p}_l) = \sum_{j=0}^{2^n-1} (y_{j+1} - y_j) \theta(\bar{v}\bar{x} - j) \quad (1.15)$$

Як можна побачити з виразу (1.15), для опису будь-якої булевої функції із n вхідних змінних достатньо 2^n параметрів (ваг). Проте це достатня, але не необхідна умова, тому варто розглянути які можливі спрощення можна застосувати в залежності від кількості вхідних змінних, щоб зменшити кількість параметрів "ваг".

Схематично вираз (1.15) можна представити як трирівневу нейромережу, що складається із 1-го вхідного, 1-го вихідного та 2^n проміжних (прихованих) нейронів, рис.1.2.

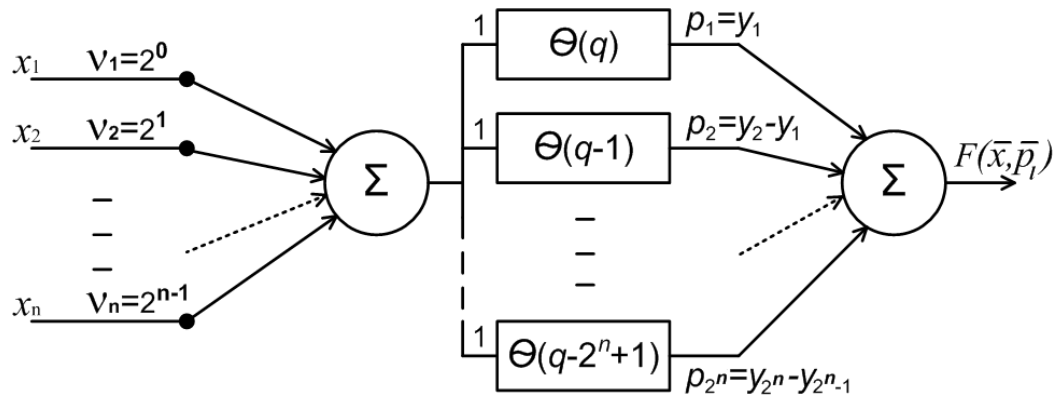


Рисунок 1.2 – Трирівнева мережа класичних перцептронів

Він потребує $n + 3 \cdot 2^n$ ваг, проте тільки 2^n із них залежать від БФ, що емулюється. Таку мережу, описану в [36], можна перетворити у 2-рівневу як продемонстровано на рис.1.3, причому θ_0 – варіант функції Хевісайда, що при аргументі 0 приймає нульове значення.

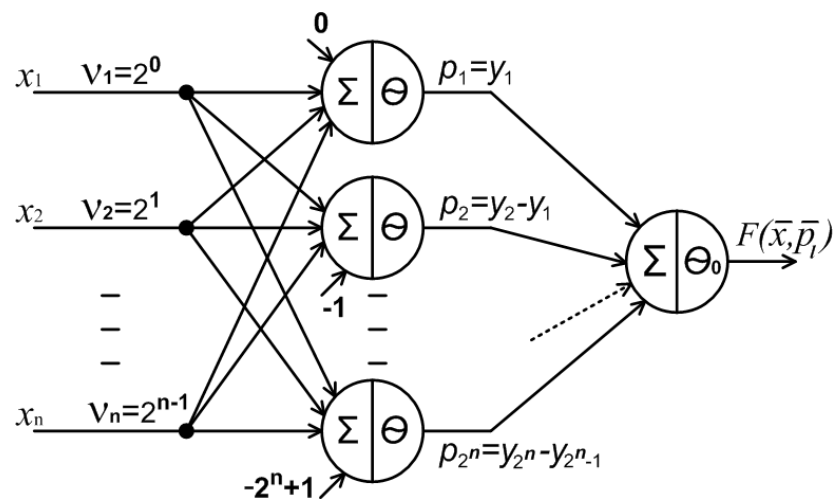


Рисунок 1.3 – Дворівнева мережа класичних перцептронів

Подана дворівнева нейромережа із 2^n вхідних нейронів та 1 вихідного характеризується меншою кількістю шарів та нейронів потребує більшої кількості вагових коефіцієнтів $(n + 2) \cdot 2^n$. Причому, як і в попередньому випадку тільки 2^n із них залежать від БФ, що емулюється [35]. Таким чином, на відміну від нейромережі, запропонованої в [36], нейромережа представлена на рис.1.2:

- потребує менше вагових коефіцієнтів в цілому;
- вимагає менших обчислень, оскільки тільки 2^n коефіцієнтів необхідно обраховувати, при цьому за простими формулами;
- дозволяє реалізувати точну, а не наближену емуляцію булевих функцій.

На першому етапі розглянемо булеві функції порядку 1, див табл.1.2:

Таблиця 1.2 – Булеві функції 1-го порядку

x_1	Y	B_1	B_2	B_3	B_4
0	y_1	0	1	0	1
1	y_2	0	0	1	1

Для булевої функції першого порядку за (1.15) отримаємо:

$$F(\bar{x}, \bar{p}_1) = y_1 \theta(x) + (y_2 - y_1) \theta(x - 1) \quad (1.16)$$

На другому етапі розрахуємо булеві функції першого порядку, табл.1.3:

Таблиця 1.3 – Розрахунок параметрів функцій 1-го порядку

x_1	Y	B_1	B_2	B_3	B_4
0	y_1	0	1	0	1
1	y_2	0	0	1	1
	p_1	0	1	0	1
	p_2	0	-1	1	0
	$F(\bar{x}, \bar{p})$	0	$\theta(x) - \theta(x - 1)$	$\theta(x - 1)$	$\theta(x)$

Як можна побачити, для опису будь-якої функції достатньо 2-ох параметрів, проте, підібравши функцію певним чином (1.17), можна обійтися 1-им параметром:

$$F(\bar{x}, \bar{p}) = \theta(x + 2 - p) - \theta(x - p) \quad (1.17)$$

Результати моделювання булевих функцій першого порядку 1-им параметром див. табл.1.4.

Таблиця 1.4 – Результати моделювання булевих функцій

x_1	Y	B_1	B_2	B_3	B_4
0	y_1	0	1	0	1
1	y_2	0	0	1	1
	p_1	0	1	3	2
	$F(\bar{x}, \bar{p})$	$\theta(x+2) - \theta(x)$	$\theta(x+1) - \theta(x-1)$	$\theta(x-1) - \theta(x-3)$	$\theta(x) - \theta(x-2)$

Таким чином, одного параметру необхідно та достатньо для моделювання логічних функцій першого порядку. За аналогією, для функцій другого порядку отримаємо:

$$F(\bar{x}, \bar{p}_l) = \sum_{j=0}^3 (y_{j+1} - y_j) \theta(\bar{v}\bar{x} - j) \quad (1.18)$$

В результаті, для представлення всіх можливих булевих функцій n -го порядку необхідно та достатньо 2^n біт (доводиться через перетворення всіх значень БФ у число). Таким чином, для задач великої розмірності ($n \geq 30$) виникає необхідність шукати компроміс між повнотою опису та об'ємом необхідної пам'яті.

Крім того, в ході дослідження отримано аналітичні міри (1.17) та (1.18), що дозволяють оцінити компонентну складність реалізації перцептронної структури, що моделює логічні функції першого та другого порядку. Встановлено, що орієнтація на лінійне розділення даних зумовлює схильність таких структур до перенавчання та їх вразливість до шумів у даних, що зумовлено намаганням адаптуватись до флуктуацій (неідеальностей) даних за допомогою лінійного розділення.

Однак, вдосконалення одношарових перцептронних структур можливе, зокрема шляхом застосування степеневих, логарифмічних та інших

нестандартних ФА, включно зі статистичними функціями, які дозволяють розширити межі їх класифікаційних можливостей. Тобто, актуальною залишається є потреба у вдосконаленні наявних та створенні нових перцептронних структур, які б дозволили розширити функціонал нейромереж, зокрема на основі застосування ймовірнісних оцінок вхідних сигналів, що може дозволити реалізувати алгоритми та методи, які спростять імплементацію складних булевих функцій на платформі нейронних мереж, включаючи розробку спеціалізованих структур та алгоритмів машинного навчання, які краще підходять для роботи з функціями вищих порядків.

Також, застосування згаданих методів може знизити ризик перенавчання, забезпечуючи краще балансування між засвоєнням тренувальних даних та здатністю до узагальнення на нових даних, що в кінцевому випадку розширює перспективи для застосування одношарових перцептронних структур в сучасних задачах машинного навчання, які потребують більшої виразності та гнучкості.

1.3 Перспективи вдосконалення та розвитку структури перцептрону, постановка задачі досліджень

В ході розвитку теорії нейронних мереж, дослідниками було запропоновано використання різних функцій активації, зокрема: лінійного порогу (гістерезису), клас сигмоїдних функцій (логістична, гіперболічний тангенс, тощо) [18, 37, 38]. ФА типу одиничного скачка і лінійного порогу на практиці використовують не часто, як правило, для навчальних прикладів. Перспективним варіантом вдосконалення структури перцептрона стало застосування сигмоїдних функції до його входів, зокрема SOFTMAX [39, 40]. Різні реалізації функції SOFTMAX передбачають застосування статистичних оцінок для нормалізації вхідних даних.

Інтенсивний розвиток та впровадження цифрових систем опрацювання даних зумовлює необхідність проведення досліджень щодо розширення функціональних можливостей нейронних мереж. Однією із задач, яку складно вирішити за допомогою одного елементарного перцептрону (на основі вищезгаданих структурних рішень), є емуляція типових логічних функцій: NOT, AND, OR, XOR [39].

Як зазначалося в п.1.2, одним із перспективних напрямків розширення функціональних можливостей перцептрону може стати використання ймовірнісного підходу, зокрема для оцінювання інформаційної ентропії вхідних сигналів сенсорів.

В ході розвитку теорії інформації запропоновано та досліджено властивості різних аналітичних виразів для розрахунку оцінок інформаційної ентропії, які локалізовано для опрацювання сигналів з різними розподілами ймовірностей появи станів у ансамблі [13, 41, 42, 43, 44, 45, 46, 47].

У такому випадку, функція суматора, що традиційно використовується, на першому етапі опрацювання зважених сигналів сенсорів може бути замінена. Фактично авторами [29] пропонується перейти від опрацювання амплітудних

величин до ймовірнісних на основі відомих інформаційних мір [13, 41, 42, 43, 44]:

К. Шеннон (C. Shannon)

$$H = -k \sum_{i=1}^n p_i \cdot \log_2 p_i \quad (1.19)$$

де k – коефіцієнт, який враховує основу логарифма;
 p_i – ймовірність появи i -го стану.

Р. Хартлі (R. Hartley)

$$H = \log_2 S^n = n \cdot \log_2 S \quad (1.20)$$

де S – кількість незалежних рівноймовірних станів;
 n – розмір вибірки.

К. Крамп (C. Kramp)

$$H = \log_2 \sqrt{2\pi e \sigma^2} \quad \text{або} \quad H = \hat{E}[\log_2 3\sigma] \quad (1.21)$$

де $\hat{E}[\]$ – функція округлення до більшого цілого.

Н. Колмогоров (ε -ентропія)

$$H_\varepsilon \leq \frac{T}{\Delta t} + \log \frac{C}{\varepsilon} \quad (1.22)$$

де Δt – крок дискретизації, що забезпечує прийнятну точність квантування ε ;

C – діапазон квантування;

T – інтервал часу спостереження.

Дж. Лонго (G. Longo)

$$H(u, p) = -k \sum_{i=1}^n [u_i p_i \cdot \log_2 p_i] \quad (1.23)$$

де $u_i \geq 0$ – коефіцієнти корисності;
 k – стала величина,
 p_i – ймовірність появи i -го стану.

В. Боюн (δ -ентропія)

$$h_\delta = \frac{|f'_{\text{сер}}(t)|}{|f'_{\text{мах}}(t)|} \quad (1.24)$$

де $f'_{\text{сер}}(t)$, $f'_{\text{мах}}(t)$ – середнє і максимальне значення похідних зміни кількості станів відповідно.

Я. Николайчук

$$H = n \cdot \hat{E} \left[\frac{1}{2} \log_2 \frac{1}{m} \sum_{j=1}^m (D_x^2 - R_{xx}^2(j)) \right] \quad (1.25)$$

де $x_i^\circ = x_i - M_x$ – центровані значення станів;
 m - кількість станів на інтервалі кореляції;
 M_x – математичне сподівання $\{x\}$;
 $D_x = \frac{1}{n} \sum_{i=1}^n (x_i - M_x)^2$ – дисперсія $\{x\}$;
 $R_{xx}(j) = \frac{1}{n} \sum_{i=1+j}^{n+j} x_i^\circ \cdot x_{i+j}^\circ$ – функція автокореляції.

Б.Олівер (B. Oliver)

$$H = \lim_{n \rightarrow \infty} \frac{\log N}{n} \quad (1.26)$$

де $N = n! / \prod_j S_j$ або $\log N = \log n! - \sum_j S_j$.

В.Таллер (W. Tuller)

$$H = k \cdot n \log_2 S_{\text{ave}}, \quad H \leq k 2BT \left(1 + \frac{S}{N} \right) \quad (1.27)$$

де S_{ave} – середнє значення станів;
 BT – інформаційна база;
 N – значення рівня шуму;
 $1/S$ – інтервал кореляції між відліками.

Поняття інформаційної ентропії (1.19) введено К.Шенноном (C. Shannon), як міру кількості інформації, що пов’язує можливі стани дискретного джерела інформації та імовірності їх появи незалежно від виду розподілу [42, 43, 46, 47]. Для випадку рівномірного розподілу ймовірностей Р.Хартлі (R. Hartley) запропонував оцінювати ентропію як функцію логарифма кількості можливих станів (1.20). В загальному випадкові процеси, породжені стохастичними об’єктами, підлягають Гаусовому закону розподілу ймовірностей можливих станів. В такому випадку більшість результатів спостережень випадкової величини знаходиться в околі 3σ її M_x , що дозволяє звести формулу (1.20) до виду (1.21) запропоновану К.Крампом (C. Krampe). Для розширення інформаційної міри К.Шеннона, Дж.Лонго введено поняття “якісних” параметрів (1.23), які характеризують корисність інформації, тоді як (1.19) використовує тільки імовірнісні характеристики. При такому підході кожному з можливих станів ставиться у відповідність деякий коефіцієнт $u_i \geq 0$, причому розподіл u_i не залежить від розподілу ймовірностей, а визначається із семантичних міркувань. Для джерел з незалежними і нерівноймовірними станами Б.Олівер запропонував оцінку загальної кількості можливих комбінацій станів (1.26), яку на основі наближення Стірлінга можна звести до виду (1.19).

Слід зазначити, що подані інформаційні міри є найбільш вживаними, однак дослідженню ентропії та особливостей її використання присвячено значна кількість робіт, зокрема таких науковців, як Г. Шульц (G. Shults), Д. Мідлтон (D. Middleton) та інші [13, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50].

Таким чином, вибір тієї чи іншої інформаційної міри, при реалізації перцептрону, доцільно здійснювати на основі виду розподілу ймовірностей станів вхідних сигналів, які формуються сенсорами S-елементами.

Слід зазначити, що у випадку вирішення задачі емуляції логічних функцій, доцільно використати інформаційну міру Р. Хартлі (R. Hartley). Таке рішення ґрунтується на тому, що згадана формула призначена для опрацювання подій (в нашому випадку можливих станів сигналів сенсорів S-елементів) з рівномірним розподілом ймовірностей появи. Крім того запропонована Р. Хартлі формула характеризується найбільшою простою, оскільки не потребує оцінювання ймовірності появи кожного з можливих станів. Спрощена структура реалізації цифрового перцептрона на основі оцінювання інформаційної ентропії з пороговою передавальною функцією активації подана на рис.1.4а. Як можна побачити, зміни в структурі, стосуються задавання величини порогу ФА. Традиційно, при реалізації перцептронів, значення порогу активації приймають рівним 0. Крім того використано функції розрахунку інформаційної ентропії за формулою (1.20), яка замінює суматор, див. рис.1.1.

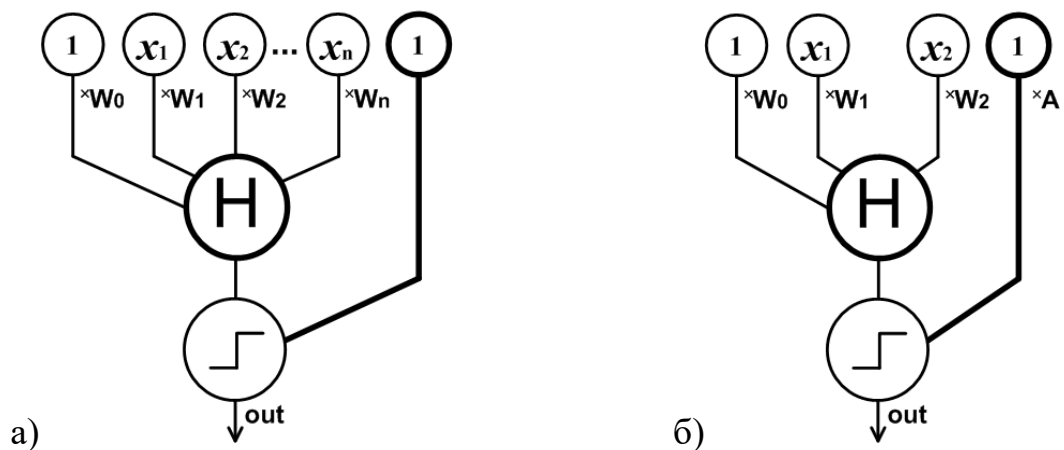


Рисунок 1.4 – Структурна схема елементарного перцептрону на основі оцінювання інформаційної ентропії а) загальна, б) на два бінарні входи для емуляції логічних функцій NOT, AND, OR та XOR

Слід зауважити, що кількість входів x , для вирішення задачі емуляції логічних функцій зводиться до двох $\{x_1, x_2\}$. Відповідно змінюється кількість вагових коефіцієнтів. Структурна схема перцептрону емуляції набору логічних функцій подана на рис.1.46 (відповідні компоненти виділені потовщеними лініями). В ході навчання перцептрону, реалізованого на основі запропонованого підходу, встановлено відповідні значення вагових коефіцієнтів w , величини зміщення b та величини порогового значення A для функції Хевісайда для кожної логічної функції, які забезпечують вирішення поставленої задачі. Результати емуляції згаданих логічних функцій, програмно реалізованої перцептронної структури (згідно структурної схеми рис.1.46), подано в таблицях 1.5 – 1.8.

Таблиця 1.5 – Вхідні сигнали, параметри та результати функціонування ентропійного перцептрона при емуляції логічної функції NOT

x_1	x_2	W_1	W_2	b	H	A	$!X_1$
0	-	1	-	1	1	0	1
1	-	1	-	1	0	0	0

Таблиця 1.6 – Вхідні сигнали, параметри та результати функціонування ентропійного перцептрона при емуляції логічної функції AND

x_1	x_2	W_1	W_2	b	H	A	$X_1 \& X_2$
0	0	1	2	0	0	1	0
0	1	1	2	0	1	1	0
1	0	1	2	0	1	1	0
1	1	1	2	0	1,58	1	1

За результатами досліджень встановлено, що запропонований підхід дозволяє вирішити задачу емуляції логічних функцій NOT, AND, OR та XOR ресурсами одного перцептрона, реалізованого на основі інформаційної ентропії, а також модифікації способу використання функції Хевісайда шляхом задавання порогового значення активації.

Таблиця 1.7 – Вхідні сигнали, параметри та результати функціонування ентропійного перцептрона при емуляції логічної функції OR

x_1	x_2	W_1	W_2	b	H	A	$X_1 X_2$
0	0	1	1	0	0	0	0
0	1	1	1	0	1	0	1
1	0	1	1	0	1	0	1
1	1	1	1	0	1	0	1

Таблиця 1.8 – Вхідні сигнали, параметри та результати функціонування ентропійного перцептрона при емуляції логічної функції XOR

x_1	x_2	W_1	W_2	b	H	A	$X_1 \oplus X_2$
0	0	1	1	2	1	1	0
0	1	1	1	2	1,58	1	1
1	0	1	1	2	1,58	1	1
1	1	1	1	2	1	1	0

Таким чином показано, що використання оцінок ентропії дозволяє емулювати повний набір логічних функцій в межах одного перцептрона. Таким чином, на основі запропонованого підходу вдалося розширити функціональні можливості елементарних перцептронних структур. Крім того, використання імовірнісного підходу при реалізації перцептронних структур дозволяє отримати ряд переваг. Зокрема інваріантність до інверсного представлення вхідних сигналів, а також знижує чутливість до апертури сигналів сенсорів.

Однак слід зазначити, що хоч використання оцінок інформаційної ентропії значень вхідних сигналів і вирішує задачу емуляції логічної функції XOR ресурсами одного перцептроном, але такий підхід зумовлює потребу у суттєвих обчислювальних ресурсах, зокрема для розрахунку функції логарифма, яка типово реалізується на основі алгоритмів розкладу в ряд Тейлора. Крім того, залишається невирішеним питання реалізації функції агрегації, яка у випадку оцінювання ентропії реалізує визначення кількості унікальних станів вхідних синаптичних сигналів.

ВИСНОВКИ ДО ПЕРШОГО РОЗДІЛУ

1. Проаналізовано сучасний стан розвитку та впровадження перцептронних структур у комп'ютерних системах за яким показано, що класичні перцептрони реалізуються через операції згортки чи кореляції, які передбачають формування зваженої суми вхідних сигналів із подальшою активацією за допомогою порогової функції, внаслідок чого мають обмежену виразну здатність та не можуть ефективно розпізнавати лінійно нероздільні шаблони.
2. Проведено класифікацію та порівняння типових функцій кореляції (коваріаційна, центрована, нормована, знакова, релейна тощо) в контексті імплементації у перцептронах за якою встановлено, що їх використання значно ускладнює алгоритмічну реалізацію, зокрема через потребу в операціях центрування, нормування тощо.
3. Розглянуто сучасні підходи до побудови активаційних функцій, виокремлено класи сигмоїдних, кусочно-лінійних та спеціалізованих функцій, а також вказано на їх вплив на обчислювальну складність і здатність нейронної мережі до навчання.
4. Запропоновано вдосконалення реалізації класичного перцептрону при якому замість зваженого сумування використовується оцінка інформаційної ентропії, що дозволяє емулювати повний набір логічних функцій в межах однієї перцептронної структури.
5. Означено перспективи подальшого розвитку перцептронних структур, зокрема з використанням різних інформаційних мір, що є інваріантні до інверсії бінарних сигналів і менш чутливі до шумів, проте потребують ефективних алгоритмів реалізації агрегуючої функції.

РОЗДІЛ 2

ТЕОРЕТИЧНІ ОСНОВИ РЕАЛІЗАЦІЇ ДИСКРЕТНОГО ПЕРЦЕПТРОНА ЗІ ЗМІЩЕНИМИ СИНАПТИЧНИМИ СИГНАЛАМИ

2.1 Дискретний перцептрон на основі імовірнісних оцінок зміщених синаптичних сигналів

На основі проведеного аналізу, встановлено, що одним з перспективних напрямків розвитку перцептронних структур може стати перехід до дискретного базису а також застосування різних імовірнісних характеристик (таких як математичне сподівання, дисперсія, мода, медіана, середнє квадратичне відхилення, центральні моменти різних порядків, кількість та розподіл ймовірностей наявних станів а також інформаційну ентропію наявних станів), як окремо так і сумісно, що розраховують для множини попередньо зміщених сигналів синапсів. Такий підхід може покращити достовірність розпізнавання а відмова від операції множення та дискретний базис дозволить відчутно зменшити обчислювальні потреби.

Для стаціонарних процесів, значення імовірнісної характеристики, що обчислюється за відповідним фрагментом вхідного сигналу прямує до деякої усталеної величини. При чому, зі збільшенням кількості спостережень флуктуації таких імовірнісних характеристик в околі усталеної величини зменшуються. Використання операції зміщення вхідних сигналів забезпечує можливість коригування кількості можливих дискретних станів сигналу, зокрема на основі цілочисельної операції додавання відповідних коефіцієнтів. Наслідком такого опрацювання буде зміна величини обраної імовірнісної характеристики, що можна використати при реалізації функції активації дискретної перцепторної структури.

В результаті запропоновано новий спосіб реалізації нейронних елементів шляхом використання значення однієї або декількох сумісно імовірнісних характеристик обчислених за зміщеними синаптичними

сигналами, що дозволяє спростити схемну, алгоритмічну і програмну реалізацію а також покращити достовірність розпізнавання. В якості таких характеристик можна використати: кількість наявних станів, розподіл ймовірностей наявних станів, інформаційну ентропію наявних станів, дисперсію, середнє квадратичне відхилення та центральні моменти різних порядків тощо. Тобто реалізація такого дискретного перцептрону стає можливою завдяки тому, що при опрацюванні використовують значення однієї або декількох сумісно імовірнісних характеристик обчислених за зміщеними синаптичними сигналами. Значення імовірнісної характеристики стаціонарних процесів, що обчислюються за зміщеними синаптичними сигналами прямує до постійної величини, яка залежить від характеристик таких сигналів. В ході опрацювання сигналів відбувається зміна кількості їх можливих станів (значень, що не повторюються) шляхом додавання відповідних коефіцієнтів зміщення і, як наслідок, змінюється значення відповідної імовірнісної характеристики, що використано для визначення рівня збудження нейрону [51].

В результаті запропоновано спосіб, у якому зміна стану активації (збудження) нейрону ґрунтується на оцінюванні значення імовірнісних характеристик обчислених за зміщеними синаптичними сигналами, що потребує менших обчислювальних затрат, розширює його функціональні можливості, покращує достовірність, кількісні та якісні характеристики. Тобто, при опрацюванні вхідних сигналів, здійснюють зміну кількості їх можливих станів (значень, що не повторюються) шляхом додавання відповідних коефіцієнтів зміщення, а рівень активації (збудження) нейрону визначають за значенням однієї або декількох сумісно імовірнісних характеристик обчислених за зміщеними синаптичними сигналами, що використовується у способі структурної організації цифрового перцептрону для дискретних синаптичних сигналів [52]. Опрацювання таких сигналів передбачає: маніпуляції з кількістю можливих станів шляхом використання операції зміщення та додаткового синаптичного входу з попередньо заданим

фіксованим дискретним станом, розрахунок однієї чи одночасно кількох імовірнісних оцінок для отриманої множини станів сигналів; порівняння із заданим пороговим значенням для зміни стану активації, зокрема на основі функції Хевісайда (ФХ), що аналітично представляється так:

$$y = \sigma(f(\bar{x} + \bar{w})) \quad (2.1)$$

де $\bar{x} = (x_1, x_2, \dots, x_n)$ – вектор вхідних дискретних сигналів,
 $\bar{w} = (w_1, w_2, \dots, w_n)$ – вектор зміщень вхідних сигналів,
 f – функція ймовірнісної оцінки вхідних дискретних сигналів,
 σ – активаційна функція

Як приклад, доцільно розглянути дискретний перцептрон з двома бінарними синапсами, який нескладно масштабувати на довільну кількість таких синаптичних входів (позначено пунктиром), рис.2.1.

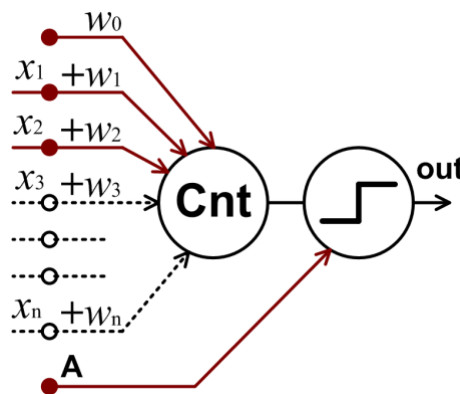


Рисунок 2.1 – Структура дискретного перцептрону для 2-ох бінарних входів з можливістю розширення до n дискретних входів, Cnt – функція визначення кількості унікальних станів

Як можна побачити, запропонована структура фактично реалізує оцінювання інформаційної ентропії зміщених сигналів синапсів за спрощеною формулою С.Хартлі (без функції $\log$), яка представляє ентропію, як логарифм кількості наявних станів джерела інформації.

В загальному випадку вектори вхідних синаптичних сигналів $\bar{X} = \{x_1, x_2, \dots, x_n\}$ та коефіцієнтів зміщення $\bar{W} = \{w_0, w_1, \dots, w_n\}$ зводяться до часткового а саме двох входів $\bar{X} = \{x_1, x_2\}$ та $\bar{W} = \{w_0, w_1, w_2\}$ відповідно. Доцільно зазначити, що w_0 не впливає на вектор $\bar{X}$ і є додатковим інструментом маніпуляції кількістю станів сигналів синапсів. В ході навчання, формування коефіцієнтів $\bar{W}$ здійснюється таким чином, щоб забезпечити найбільшу чи найменшу (залежно від вирішуваної задачі) кількість станів для розрахунку $f(\bar{W}, \bar{X})$ і порівняння результату з пороговим значенням функції активації $f(A)$ перцептрону. Таким чином, вектор зміщених сигналів синапсів $\{w_0, w_1 + x_1, w_2 + x_2\}$ отримується в результаті додавання відповідних елементів векторів $\bar{X}$ та $\bar{W}$. Далі, для отриманого вектору зміщених сигналів, розраховують обрану імовірнісну характеристику, зокрема визначають кількість унікальних станів сигналу $Q = \text{Cnt}(\bar{W}, \bar{X})$. Після чого, результат порівняння величини Q зі пороговим значенням $f(Q, A)$ визначає рівень активації нейрону. Результати емуляції типових логічних функцій AND, OR, XOR а також NAND та NOR з відповідними коефіцієнтами зміщення $\bar{W}$ та значенням порогу активації A запропонованої перцептронної структури подано в таблицях 2.1-2.5.

Таблиця 2.1 – Емуляція логічної функції **AND** з коефіцієнтами зміщення $\bar{W} = \{0, 0, 1\}$ та порогом активації перцептрону $A = 3$

x_1	0	0	1	1
x_2	0	1	0	1
w_0	0			
$x_1 + w_1$	0	0	1	1
$x_2 + w_2$	1	2	1	2
$Q = \text{Cnt}(\bar{W}, \bar{X})$	2	2	2	3
A	3			
$x_1 \text{ AND } x_2$	0	0	0	1

Таблиця 2.2 – Емуляція логічної функції **OR** з коефіцієнтами зміщення $\bar{W} = \{0, 0, 0\}$ та порогом активації перцептрону $A = 2$

x_1	0	0	1	1
x_2	0	1	0	1
w_0	0			
$x_1 + w_1$	0	0	1	1
$x_2 + w_2$	0	1	0	1
$Q = \text{Cnt}(\bar{W}, \bar{X})$	1	2	2	2
A	2			
$x_1 \text{ OR } x_2$	0	1	1	1

Таблиця 2.3 – Емуляція логічної функції **XOR** з коефіцієнтами зміщення $\bar{W} = \{1, 2, 2\}$ та порогом активації перцептрону $A = 3$

x_1	0	0	1	1
x_2	0	1	0	1
w_0	1			
$x_1 + w_1$	2	2	3	3
$x_2 + w_2$	2	3	2	3
$Q = \text{Cnt}(\bar{W}, \bar{X})$	2	3	3	2
A	3			
$x_1 \text{ XOR } x_2$	0	1	1	0

Таблиця 2.4 – Емуляція логічної функції **NAND** з коефіцієнтами зміщення $\bar{W} = \{1, 0, 0\}$ та порогом активації перцептрону $A = 2$

x_1	0	0	1	1
x_2	0	1	0	1
w_0	1			
$x_1 + w_1$	0	0	1	1
$x_2 + w_2$	0	1	0	1
$Q = \text{Cnt}(\bar{W}, \bar{X})$	2	2	2	1
A	2			
$x_1 \text{ NAND } x_2$	1	1	1	0

Як можна побачити, описаний спосіб реалізації дискретного перцептрону дозволяє емулювати типові логічні функції, забезпечує

спрощення алгоритмічної, програмної та апаратної реалізації, що відкриває можливості використання в малопродуктивних обчислювальних системах та цифрових компонентах, а також розширює функціональні можливості компонентів штучних нейронних мереж.

Таблиця 2.5 – Емуляція логічної функції **NOR** з коефіцієнтами зміщення $\overline{W} = \{2, 1, 0\}$ та порогом активації перцептрону $A = 3$

x_1	0	0	1	1
x_2	0	1	0	1
w_0	2			
$x_1 + w_1$	1	1	2	2
$x_2 + w_2$	0	1	0	1
$Q = \text{Cnt}(\overline{W}, \overline{X})$	3	2	2	2
A	3			
$x_1 \text{ NOR } x_2$	1	0	0	0

Одним із найпростіших способів навчання вищеприписаного перцептрона є пошук в дискретному просторі можливих станів варіантів величин зміщення [53, 54], оскільки дискретне представлення фактично суттєво зменшує поле пошуку.

Так, для прикладу, реалізація логічних функцій та їх комбінацій на основі вище запропонованої структури перцептрону передбачає необхідність перебрати лише 256 можливих комбінацій.

Найбільш характерні із результатів пошуку для запропонованого дискретного перцептрону на два інформаційні входи подано на рис.2.2-2.7.

Тут вісь Z представляє абсолютну похибку – відхилення від значення емуляваної логічної функції при заданих параметрах дискретного перцептрону а осі X та Y відображають значення коефіцієнтів зміщення w_1 та w_2 , коефіцієнт w_0 та A – зафіксовані $w_0=1$, а $A=3$ для логічних функцій AND та XOR, $A=2$ для логічної функції OR.

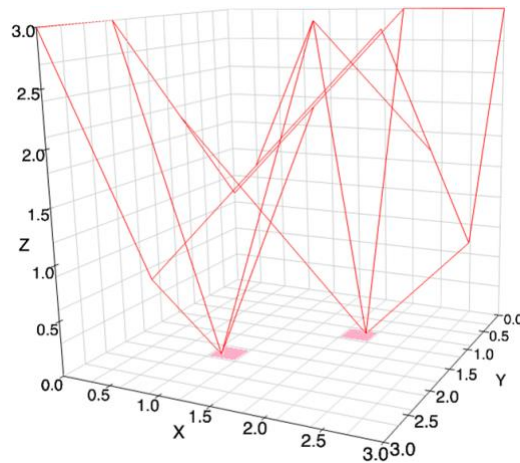


Рисунок 2.2 – Функція помилки для емульованої логічної функції AND при фіксованих w_0 та порогу активації A ФХ

Застосування методів машинного навчання, які традиційно базуються на пошуку екстремумів в просторі можливих рішень, зокрема таких як: метод градієнтного спуску (Gradient Descent), метод зворотного поширення помилки (Backpropagation), стохастичний градієнтний спуск (Stochastic Gradient Descent) тощо, зазнають суттєвих труднощів у випадках, коли вихідна функція, як можна побачити на рисунках 2.2-2.4, не є гладкою і до того ж має розриви.

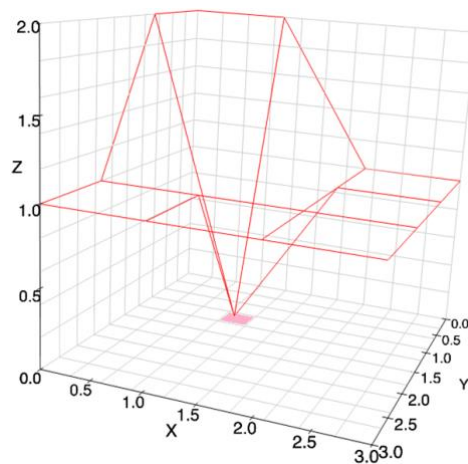


Рисунок 2.3 – Функція помилки для емульованої логічної функції OR при фіксованих w_0 та порогу активації A ФХ

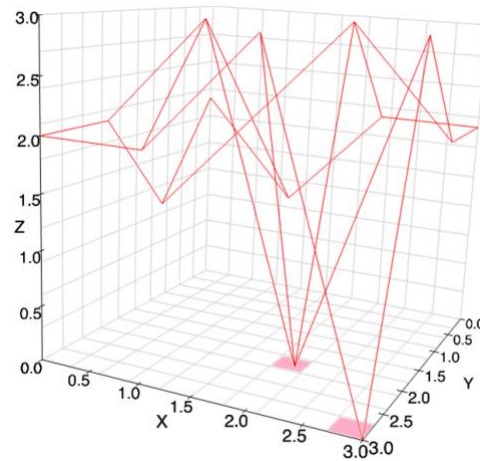


Рисунок 2.4 – Функція помилки для емульованої логічної функції XOR при фіксованих w_0 та порогу активації A ФХ

Аналогічні дослідження функції помилки, рис.2.5-2.7, також здійснено для модифікованої функції Хевісайда (МФХ), яка при $A \geq 0$ збігається з класичною пороговою функцією, а для випадку коли $A < 0$ реалізується інвертована порогова функція.

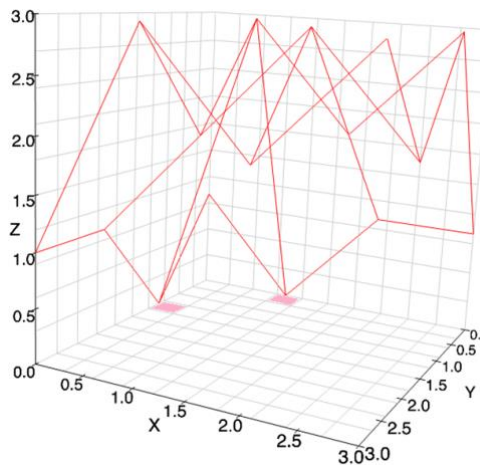


Рисунок 2.5 – Функція помилки для емульованої логічної функції AND при фіксованих w_0 та порогу активації A МФХ

Гладкість функції є важливою властивістю, яка забезпечує безперервність її похідних, що дозволяє використовувати градієнтні методи для оптимізації. У разі наявності розривів градієнти не визначені або стають

нескінченними, що робить градієнтні методи неефективними або взагалі непридатними для такого типу функцій [55, 56, 57].

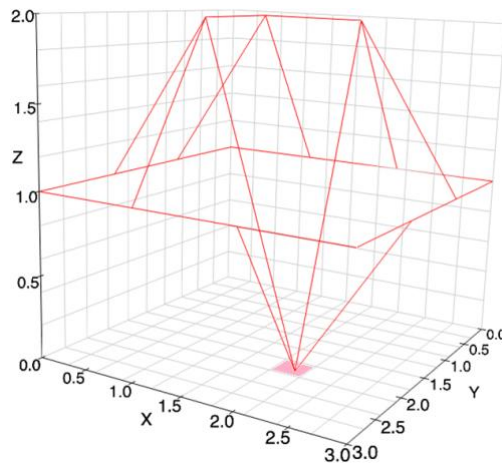


Рисунок 2.6 – Функція помилки для емульованої логічної функції OR при фіксованих w_0 та порогу активації A МФХ

Однією з основних проблем при оптимізації негладких функцій є відсутність надійних інструментів для оцінки напрямку та величини зміни функції. Градієнтні методи, такі як стохастичний градієнтний спуск (SGD), покладаються на обчислення похідних для визначення напрямку руху до мінімуму або максимуму функції.

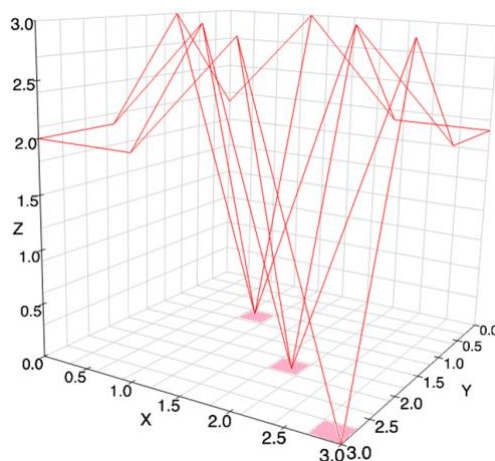


Рисунок 2.7 – Функція помилки для емульованої логічної функції XOR при фіксованих w_0 та порогу активації A МФХ

У випадку розривів такі методи машинного навчання стають ненадійними, оскільки наявність точок розриву призводить до різких стрибків значень похідних, що, в свою чергу, спричиняє нестабільність алгоритму [58].

Ще однією складністю є проблеми з конвергенцією. У негладких функціях розриви можуть спричиняти затримки або повну зупинку процесу конвергенції, оскільки алгоритми не можуть правильно визначити напрямок і величину кроку. Це особливо критично в задачах, де важливо швидко та точно знайти оптимальні параметри, зокрема при навчанні штучних нейронних мереж. Розриви можуть також призводити до того, що алгоритми "застрягатимуть" у локальних екстремумах, не досягаючи глобального мінімуму або максимуму [59].

2.2 Обмеження множини рішень для дискретного перцептрона при емуляції логічних функцій

До ключових проблеми, що виникають при машинному навчанні дискретних перцептронів з типовою структурою (див. рис.2.8а) мається на увазі перцептронів із пороговою чи східчастою функцією активації, що формує "дискретний" вихід, слід віднести [14, 60, 61, 62]:

- непридатність до вирішення нелінійно роздільних задач, тобто перцептрон здатний коректно навчитися лише у випадку, коли вибірка навчальних даних є лінійно віддільною, зокрема такі логічні функції як AND, OR, NAND, NOR та інші, що є лінійно роздільними, нескладно відтворюються за допомогою простих перцептронів, однак, якщо дана задача не є лінійно роздільною, наприклад логічна функція XOR, то простий дискретний перцептрон не може знайти розв'язок у принципі;

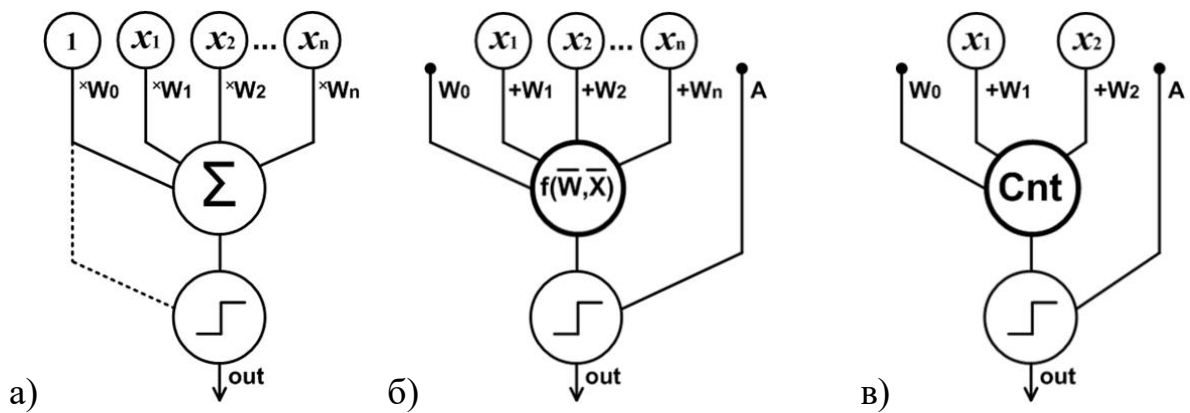


Рисунок 2.8 – Структура перцептрону на основі: а) суматора (типова) на n входів; б) ймовірнісних оцінок зміщених сигналів синапсів на n входів; в) функції Cnt - підрахунку кількості різних станів зміщених синаптичних сигналів на 2 бінарні входи

- відсутність похідної та "сигналу" для градієнтних методів, тобто для порогової активації похідна не визначена в точці порогу, отже звичні методи (зокрема градієнтного спуску) не застосовуються безпосередньо оскільки

неможливо обчислити градієнтну похибку в традиційному вигляді. В такій ситуації, дискретний перцептрон можна навчати за допомогою правил перцептронної (Hebbian-like) корекції - коригування ваг лише за наявності помилки. Недоліком такого підходу є те, що процес машинного навчання може довго коливатися навколо коректного рішення, зокрема для випадків коли вибірка майже лінійно роздільна або містить шум;

- надмірна чутливість до зашумленості вхідних даних, оскільки вихід перцептронну дискретний і змінюється стрибкоподібно то навіть незначні коливання на вході в околі порогу можуть змінити результат на протилежний. В такій ситуації використовують методи згладжування вхідних сигналів чи використовують інші функції активації;

- низька роздільна/виразна здатність, оскільки дискретний перцептрон є лінійним класифікатором то він не може описати нелінійні розділові поверхні без доповнення його додатковими шарами чи особливими методами трансформації ознак;

- проблема зупинки навчання, якщо набори вхідних даних не є ідеально лінійно роздільними чи містять протиріччя, алгоритм навчання може бути не збіжним. Це може проявлятися в ситуації безкінечного навчання (постійне коригування ваг у різні боки після кожної помилки) без гарантії зменшення помилок до нуля, або в осциляціях навколо часткового рішення;

- потреба в обережному виборі гіперпараметрів, оскільки швидкість машинного навчання (learning rate) та ініціалізація вагових коефіцієнтів суттєво впливають на часову складність навчання то некоректно обраний крок оновлення може спричиняти зайві стрибки ваг і тривале чи навіть нескінченне навчання.

Таким чином, основні проблеми можна звести до того, що: дискретна активація ускладнює використання градієнтних методів, модель від природи має низьку роздільну/виразну здатність, а зашумлені або нелінійно роздільні дані роблять таке машинне навчання проблематичним або неможливим.

В такій ситуації доцільно розглянути запропоновану реалізацію перцептрону (див. рис.2.8б), що ґрунтується на використанні імовірнісних характеристик, зокрема оцінюванні міри хаотичності, а саме інформаційної ентропії вхідних дискретних сигналів синапсів [29].

Фактично тут запропоновано перехід до дискретного базису, хоча структура перцептрону залишається незмінною, однак опрацювання зважених (перемножених з відповідними ваговими коефіцієнтами) синаптичних сигналів, які формують множину з обмеженою кількістю дискретних станів, передбачає оцінювання інформаційної ентропії такої множини з подальшим залученням порогової функції активації. Такий підхід є нечутливим до абсолютних значень сигналів синапсів а враховує тільки імовірності їх появи.

До проблемних моментів згаданої структури слід віднести те, що формула ентропії містить функцію розрахунку логарифму і це призводить до додаткових обчислювальних затрат, а також відсутність досліджень щодо особливостей навчання таких перцептронних структур.

Таким чином, перцептрон, як базова ланка класичних нейронних мереж, можна розглядати у "статистико-механічних" категоріях, зокрема: розподіл вагових коефіцієнтів, динаміку навчання та вплив випадковості в процесі тренування. В матеріалах [63] розглянуто методи, які дозволяють оцінювати зміни в параметрах та структуру просторів станів, що можуть бути адаптовані для аналізу перцептрону: як складність чи хаотичність даних відображається на здатності нейронної мережі навчатися. Знання щодо критичних режимів або фазових переходів (key transitions) у складних системах дають можливість, коли алгоритм машинного навчання перцептрону може "застрягати" а коли він здатний до ефективної узагальнюючої здатності. Тобто, якщо можна виявити певні "точки фазового переходу", зокрема, межу між простими та складними для навчання наборами даних, це може розширити можливості для покращення алгоритмів тренування.

Доцільно зазначити, що аналіз дискретних перцептронів і досі лежить в основі розуміння лінійної роздільності та обчислювальної спроможності

нейронних мереж а класичне правило навчання перцептрону має сформульовані докази збіжності лише для лінійно віддільних випадків.

Реалізація перцептрону, що використовує імовірнісні оцінки (див. рис.2.8б), зокрема інформаційну ентропію вхідних дискретних сигналів синапсів [29], нескладно спростити замінивши формулу розрахунку ентропії К.Шеннона: $H = -k \sum_{i=1}^n p_i \cdot \log p_i$ яка для випадку рівномірного розподілу вхідних сигналів (даних) може бути зведена до формули С.Хартлі: $H = \log_2 S^n = n \cdot \log_2 S$. Тут доцільно зазначити, що кількість входів незмінна для всіх випадків, тому коефіцієнтом n можна знехтувати.

Крім того, відмова від використання функції логарифму дозволяє суттєво зменшити обчислювальні затрати без порушення адекватності результату. В результаті запропоновано структуру перцептрону, у якій зміна стану активації (збудження) ґрунтується на оцінюванні кількості унікальних станів (значень, що не повторюються) шляхом додавання відповідних дискретних коефіцієнтів зміщення до сигналів синапсів [51, 52].

В такій ситуації доцільно провести дослідження множини можливих рішень, зокрема на прикладі перцептронної структури (див. рис. 2.8в), що здатна емулювати вичерпний перелік логічних функцій для 2-ох бінарних вхідних сигналів [64]. При виборі коефіцієнтів зміщення (що можна вважати деяким аналогом ваг) перцептрону одне значення, для наочності, необхідно зафіксувати як константу, зокрема додатковий вхід $w_0=0$. Такий вибір пояснюється тим, що зміщення всіх коефіцієнтів на однакову величину не змінює кількості унікальних станів. Так для логічних функції OR значення коефіцієнтів зміщення (w_0, w_1, w_2) для 2-ох входового перцептрону можуть бути [0,0,0] або [1,1,1] або [2,2,2].

Одним із ефективних прикладів знаходження допустимих значень коефіцієнтів зміщення, в цьому випадку, може бути використання матриці кількості унікальних станів при фіксованому значенні додаткового входу w_0 ,

яка формується таким чином: для кожної комбінації $(0, w_1, w_2)$ знаходиться кількість унікальних станів. Наприклад: $(0,0,0)$ дає 1 стан, $(0,1,1)$ - 2 стани.

Варіант реалізації такої матриці для 2-ох входового дискретного перцептрону, при зафіксованому значенні додаткового входу $w_0=0$ подано в табл.2.6.

Таблиця 2.6 – Кількість унікальних станів зміщених синаптичних сигналів $sx_1=(x_1+w_1)$ та $sx_2=(x_2+w_2)$ на відрізку $[-4,4]$ для 2-ох входового дискретного перцептрону при зафіксованому значенні $w_0=0$

$sx_1 \backslash sx_2$	-4	-3	-2	-1	0	1	2	3	4
-4	2	3	3	3	2	3	3	3	3
-3	3	2	3	3	2	3	3	3	3
-2	3	3	2	3	2	3	3	3	3
-1	3	3	3	2	2	3	3	3	3
0	2	2	2	2	1	2	2	2	2
1	3	3	3	3	2	2	3	3	3
2	3	3	3	3	2	3	2	3	3
3	3	3	3	3	2	3	3	2	3
4	3	3	3	3	2	3	3	3	2

Як можна побачити отримана матриця є симетричною відносно головної діагоналі, що дозволяє скоротити, як мінімум наполовину, множину пошуку можливих коефіцієнтів зміщення, які необхідно опрацювати, при реалізації перцептроном відповідної логічної функції.

Знаходження прийнятних значень коефіцієнтів зміщення w_1 та w_2 здійснюється шляхом порівняння можливих значень з таблиці істинності обраної логічної функції, для всіх комбінацій входів x_1 та x_2 , із відповідно розташованими значеннями фрагменту ідентичної розмірності (виділено

потовщенням) у таблиці кількості унікальних станів синаптичних сигналів 2-ох входового дискретного перцептрону.

Тобто, в ході пошуку значень коефіцієнтів зміщення w_1 та w_2 , що задовольняють умовам емуляції відповідної логічної функції відповідний фрагмент таблиці істинності почергово накладається на відповідні фрагменти матриці кількості унікальних станів з метою виявлення характерних відношень між відповідними значеннями таблиці істинності та аналогічно розташованими значеннями матриці.

Іншими словами необхідно виявити функціональну відповідність між значеннями таблиці істинності логічної функції та значеннями матриці кількості унікальних станів. Результати знаходження коефіцієнтів зміщення для емуляції логічних функцій OR та NOR подано в таблиці 2.7.

Таблиця 2.7 – Доступні рішення емуляції логічного компоненту OR та NOR на основі 2-ох входового дискретного перцептрону для $sx_1=(x_1+w_1)$ та $sx_2=(x_2+w_2)$ на відрізку $[-4,4]$ при зафіксованому значенні $w_0=0$

			$sx_1 \backslash sx_2$	-4	-3	-2	-1	0	1	2	3	4
			-4	2	3	3	3	2	3	3	3	3
			-3	3	2	3	3	2	3	3	3	3
			-2	3	3	2	3	2	3	3	3	3
			-1	3	3	3	2	2	3	3	3	3
			0	2	2	2	2	1	2	2	2	2
			1	3	3	3	3	2	2	3	3	3
			2	3	3	3	3	2	3	2	3	3
			3	3	3	3	3	2	3	3	2	3
			4	3	3	3	3	2	3	3	3	2

OR	0	1
0	0	1
1	1	1

NOR	0	1
0	1	0
1	0	0

З аналогічним підходом не складно знайти значення коефіцієнтів зміщення бінарних сигналів перцептронів для логічних функцій AND, NAND, XOR, а

також XNOR. Результати знаходження коефіцієнтів зміщення для емуляції згаданих логічних функцій подано в таблицях 2.8 та 2.9.

Таблиця 2.8 – Доступні рішення емуляції логічного компоненту AND та NAND на основі 2-ох входового дискретного перцептрону для $sx_1=(x_1+w_1)$ та $sx_2=(x_2+w_2)$ на відрізку $[-4,4]$ при зафіксованому значенні $w_0=0$

			$sx_1 \backslash sx_2$	-4	-3	-2	-1	0	1	2	3	4
			-4	2	3	3	3	2	3	3	3	3
			-3	3	2	3	3	2	3	3	3	3
			-2	3	3	2	3	2	3	3	3	3
			-1	3	3	3	2	2	3	3	3	3
			0	2	2	2	2	1	2	2	2	2
			1	3	3	3	3	2	2	3	3	3
			2	3	3	3	3	2	3	2	3	3
			3	3	3	3	3	2	3	3	2	3
			4	3	3	3	3	2	3	3	3	2

AND	0	1
0	0	0
1	0	1

NAND	0	1
0	1	1
1	1	0

Таблиця 2.9 – Доступні рішення емуляції логічного компоненту XOR та XNOR на основі 2-ох входового дискретного перцептрону для $sx_1=(x_1+w_1)$ та $sx_2=(x_2+w_2)$ на відрізку $[-4,4]$ при зафіксованому значенні $w_0=0$

			$sx_1 \backslash sx_2$	-4	-3	-2	-1	0	1	2	3	4
			-4	2	3	3	3	2	3	3	3	3
			-3	3	2	3	3	2	3	3	3	3
			-2	3	3	2	3	2	3	3	3	3
			-1	3	3	3	2	2	3	3	3	3
			0	2	2	2	2	1	2	2	2	2
			1	3	3	3	3	2	2	3	3	3
			2	3	3	3	3	2	3	2	3	3
			3	3	3	3	3	2	3	3	2	3
			4	3	3	3	3	2	3	3	3	2

XOR	0	1
0	0	1
1	1	0

XNOR	0	1
0	1	0
1	0	1

Як можна побачити, реалізація логічних функцій XOR та XNOR може бути представлена тими ж коефіцієнтами зміщень, але використано різні функції активації ($=$) та ($\neq$) відповідно. Така ситуація пояснюється тим, що значення функцій XOR та XNOR симетричні відносно головної та побічної діагоналей (див. таблиці істинності цих функцій).

Доцільно зазначити, що при реалізація логічних функцій OR, AND, NOR та NAND є можливість маніпуляції як коефіцієнтами зміщення w_1 та w_2 так і функціями активації оскільки вони симетричні тільки відносно головної діагоналі.

Фактично, вибір функції активації, зокрема для розглянутих випадків це: $<$, $>$, $\leq$, $\geq$, $=$, $\neq$ безпосередньо впливає на оцінювання відношень між елементами матриці кількості унікальних станів, що необхідно враховувати при виборі необхідних значень коефіцієнтів зміщень.

Таблиця 2.10 – Узагальнена множина доступних рішень емуляції логічних компонентів OR, AND, XOR, NOR, NAND, XNOR на основі 2-ох входового дискретного перцептрону для $sx_1=(x_1+w_1)$ та $sx_2=(x_2+w_2)$ на відрізку $[-4,4]$ при зафіксованому значенні $w_0=0$

$sx_1 \backslash sx_2$	-4	-3	-2	-1	0	1	2	3	4
-4	2	3	3	3	2	3	3	3	3
-3	3	2	3	3	2	3	3	3	3
-2	3	3	2	3	2	3	3	3	3
-1	3	3	3	2	2	3	3	3	3
0	2	2	2	2	1	2	2	2	2
1	3	3	3	3	2	2	3	3	3
2	3	3	3	3	2	3	2	3	3
3	3	3	3	3	2	3	3	2	3
4	3	3	3	3	2	3	3	3	2

Оскільки функції XOR та XNOR мають ідентичні рішення в матриці кількості унікальних станів (див. табл.2.9), то в такому випадку доцільно використовувати від'ємні значення порогу функції активації як ознаку інверсії результату виходу перцептрону.

Таким чином, використання операцій порівнювання $>$, $\geq$, $<$, $\leq$ в якості функцій активації результату оцінювання кількості унікальних значень зміщених синаптичних сигналів дискретного перцептрону дозволяє розділяти дискретні функції, значення яких симетричні відносно головної діагоналі можливих коефіцієнтів зміщення, див. табл. 2.7–2.9.

Однак, використання операцій рівності ($=$) чи нерівності ($\neq$) розширює функціонал дискретного перцептрону і забезпечує ефективне розділення дискретних функцій значення яких симетричні відносно обох діагоналей коефіцієнтів зміщення, що нескладно демонструється на прикладі функції XNOR, див. табл.2.11.

Таблиця 2.11 – Емуляція логічної функції **XNOR** з коефіцієнтами зміщення $\bar{W} = \{1, 2, 2\}$ та порогом активації перцептрону $A = 2$

x_1	0	0	1	1
x_2	0	1	0	1
w_0	1			
$x_1 + w_1$	2	2	3	3
$x_2 + w_2$	2	3	2	3
$Q = \text{Cnt}(\bar{W}, \bar{X})$	2	3	3	2
A	2			
$x_1 \text{ XNOR } x_2$	1	0	0	1

Якщо узагальнити прийнятні варіанти коефіцієнтів зміщення перцептрону для усіх вищерозглянутих логічних функцій OR, AND, XOR, NOR, NAND, XNOR, результат подано в табл.2.10, то можна побачити, що множина розв'язків, значень w_1 та w_2 при зафіксованому $w_0=0$, групується в околі головної діагоналі.

Таким чином встановлено, що для запропонованої структури 2-ох входового дискретного перцептрону прийнятні розв'язки слід шукати вздовж головної діагоналі матриці кількості унікальних станів околі, який не перевищує 35% вибірки для розглянутих значень коефіцієнтів w_1 та w_2 . З огляду на отримані результати можна припустити, що тренд розташування можливих розв'язків буде зберігатися і для інших логічних функцій, зокрема на більшу кількість входів.

На основі отриманих результатів, а також для перевірки їх коректності засобами мови програмування Python розроблено програму знаходження коефіцієнтів зміщення та порогу активації дискретного перцептрону для емуляції логічних функцій, див. додаток А. В результаті чисельного моделювання з використанням вищезгаданої програми підтверджено коректність запропонованих рішень, а також визначено коефіцієнти зміщення (w_0 , w_1 , w_2 та поріг A), які забезпечують емуляцію відповідних логічних функцій на два входи для всіх можливих комбінацій (функція активації – індикаторна, функції агрегації – $\text{Cnt}()$), див. додаток Б.

ВИСНОВКИ ДО ДРУГОГО РОЗДІЛУ

1. Вперше запропоновано реалізацію дискретного перцептрону на основі імовірнісних оцінок зміщених синаптичних сигналів, що дозволяє перейти до дискретного базису, забезпечити опрацювання лінійно-нероздільних множин, уникнути використання операцій множення та спростити апаратну реалізацію. Розроблена структура визначає рівень активації перцептрону за кількістю унікальних станів зміщених синаптичних сигналів, що реалізує ентропійну оцінку за спрощеною формулою Хартлі без використання логарифмічної функції.
2. Показано доцільність переходу до дискретного базису в реалізації перцептронів, що забезпечує зменшення обчислювальної складності та дозволяє ефективно працювати з обмеженим числом станів сигналів, а також встановлено, що використання інформаційної ентропії дозволяє зменшити чутливість моделі до абсолютних значень вхідних сигналів.
3. Досліджено можливість опрацювання лінійно-нероздільних множин на основі емуляції логічних функцій AND, OR, XOR, NAND, NOR, XNOR запропонованим дискретним перцептроном, що підтверджує розширення функціоналу порівняно з реалізацією перцептрону на основі кореляції.
4. Запропоновано метод обмеження множини рішень для дискретного перцептрона шляхом аналізу матриці кількості унікальних станів для 2-ох бінарних входів при фіксованому порозі активації, що зменшує простір пошуку до 35% від повного.
5. Досліджено вплив вибору функції активації як операції порівнювання ($>$, $<$, $=$, $\neq$), що у поєднанні з відповідними коефіцієнтами зміщення дозволяє адаптувати перцептрон до різних типів логічних функцій та їх типових комбінацій.

РОЗДІЛ 3

МОДЕЛІ ТА АЛГОРИТМИ РЕАЛІЗАЦІЇ КОМПОНЕНТІВ ДИСКРЕТНОГО ПЕРЦЕПТРОНУ

3.1 Моделі функцій агрегації дискретного перцептрона на основі багат шарової класичної нейронної мережі

Означена в п.2.1 та досліджена в п.2.2 проблема машинного навчання дискретного перцептрона зі зміщеними синаптичними сигналами окреслює пошук можливих рішень значень коефіцієнтів зміщень в n -мірному просторі станів (в залежності від кількості вхідних сигналів). Іншими варіантами вирішення означеної проблеми можуть бути, наприклад, методи евристичного пошуку, генетичні алгоритми або методи, засновані на імітації відпалу, які не потребують обчислення похідних і можуть бути більш стійкими до розривів [65]. Однак, ці методи часто мають свої власні недоліки, такі як більша обчислювальна складність та менша ефективність, що робить задачу оптимізації негладких функцій все ще відкритою та актуальною для дослідження у сфері машинного навчання [66].

Використання дискретного базису суттєво обмежує поле пошуку при переборі і у випадку невеликої кількості вхідних параметрів може успішно використовуватися як варіант швидкого навчання перцептронних структур. Однак, при збільшенні кількості входів такого перцептрону кількість кроків перебору зростає експоненційно, що потребує застосування інших підходів до машинного навчання [67].

Таким чином, доцільно розглянути використання еквівалентних нейромережових структур, що реалізуються на основі класичних перцептронів [39, 68, 69]. Тоді, в загальному випадку вираз (2.1) маніпулює: вектором вхідних дискретних сигналів $\bar{x} = (x_1, x_2, \dots, x_n)$ та вектором зміщень $\bar{w} = (w_1, w_2, \dots, w_n)$ з подальшим використанням функції f оцінювання інформаційної ентропії (або іншої ймовірнісної характеристики) зміщених

сигналів результат якої є аргументом активаційної функції σ (наприклад порогова функція Хевісайда θ) [14, 70, 71, 72].

В такому випадку доцільно зазначити, що інформаційна ентропія, як параметр, не залежить від значень вхідних сигналів, хоча в залежності від їх розподілу може бути розрахована по-різному, зокрема з використанням формул Шеннона (1.19), Хартлі (1.20), Крампа (1.21), Колмогорова (1.22), Лонго (1.23), Боюна (1.24), Николайчука (1.25), Олівера (1.26) а також Таллера (1.27). Проаналізувавши згадані інформаційні міри слід відмітити, що тільки формула Шеннона (1.19) винятково оперує ймовірностями появи наявних значень (станів) вхідного сигналу. Фактично усі інші формули є актуальні тільки для певних видів розподілу оскільки послуговуються значеннями вхідного сигналу і дозволяють отримати верхню оцінку міри ентропії. Однак верхня оцінка такої міри не відображає поточного значення ентропії, що в окремих ситуаціях може бути некоректно [73].

Спільним недоліком вищенаведених формул (1.19-1.27) є те, що при їх розрахунку використовується функція логарифму, що потребує найбільше обчислювальних затрат, яка традиційно розраховується на основі розкладу в ряд Тейлора. Крім того, кількість членів ряду, що необхідно розрахувати для забезпечення прийнятної точності стрімко зростає у випадках, коли ймовірності появи станів вхідного сигналу розташовані в околі нульових значень. В результаті такого непропорційного зростання обчислювальних затрат стає складно забезпечити ефективну апаратну чи програмну реалізацію алгоритмів розрахунку інформаційної ентропії.

Одним з перспективних варіантів спрощення обчислювальної складності може бути використання формули Хартлі (1.20), що дозволяє оцінити інформаційну ентропію вхідних сигналів розподілених рівномірно, без операції логарифмування. Таким чином, функція оцінки інформаційної ентропії для (2.1) замінюється на функцію визначення кількості унікальних станів вхідних сигналів $Q(\bar{x} + \bar{w})$, зокрема: $Q((1,2,1) + (1,0,1)) = Q((2,2,2)) = 1$,

оскільки є тільки одне унікальне значення серед результуючих значень сигналу, що рівне 2.

Для побудови штучної нейронної мережі еквівалентної до (2.1) необхідно земулювати функцію виду [51]:

$$y = C(x_1, x_2) = \begin{cases} 0, x_1 \neq x_2 \\ 1, x_1 = x_2 \end{cases} \quad (3.1)$$

Подану функцію можна виразити через функцію Хевісайда θ , а саме:

$$y = C(x_1, x_2) = \theta(x_1 - x_2) + \theta(x_2 - x_1) - 1 \quad (3.2)$$

Базуючись на виразі (3.2) можна представити штучну нейронну мережу, рис.3.1, що буде обчислювати функцію $C(x_1, x_2)$.

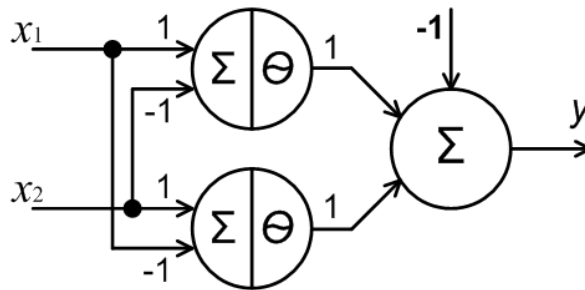


Рисунок 3.1 – Приклад нейронної структури для порівняння двох сигналів на рівність

Розширивши (3.2) на вектор $\bar{x} = (x_1, x_2, \dots, x_n)$ у якому визначається кількість елементів рівних елементу (i) отримаємо:

$$\begin{aligned} C(\bar{x}, i) &= \sum_{j=1}^n [\theta(x_i - x_j) + \theta(x_j - x_i) - 1] = \\ &= \sum_{j=1}^n \theta(x_i - x_j) + \sum_{j=1}^n \theta(x_j - x_i) - n \end{aligned} \quad (3.3)$$

Для підрахунку кількості неповторюваних значень вхідних сигналів вводимо метрику унікальності обернену до кількості елементів рівних елементу i :

$$U(\bar{x}, i) = \frac{1}{C(\bar{x}, i)} \quad (3.4)$$

Підставивши вираз (3.3) у (3.4) та провівши необхідні перетворення, отримаємо:

$$Q(\bar{x}) = \sum_{i=1}^n U(\bar{x}, i) = \sum_{i=1}^n \frac{1}{C(\bar{x}, i)} = \sum_{i=1}^n \frac{1}{\sum_{j=1}^n \theta(x_i - x_j) + \sum_{j=1}^n \theta(x_j - x_i) - n} \quad (3.5)$$

де $Q(\bar{x})$ – кількість неповторюваних елементів вектору $\bar{x}$.

Базуючись на (3.5), нескладно реалізувати нейронну мережу, здатну визначати кількість унікальних значень вхідних синаптичних сигналів [51], варіант реалізації подано на рис.3.2, 3.3.

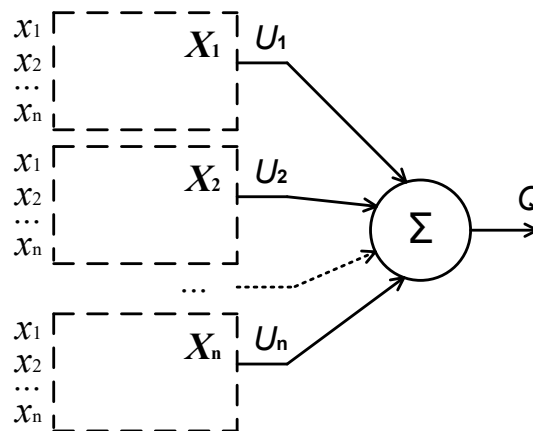


Рисунок 3.2 – Суматор, що реалізує підрахунок кількості унікальних сигналів

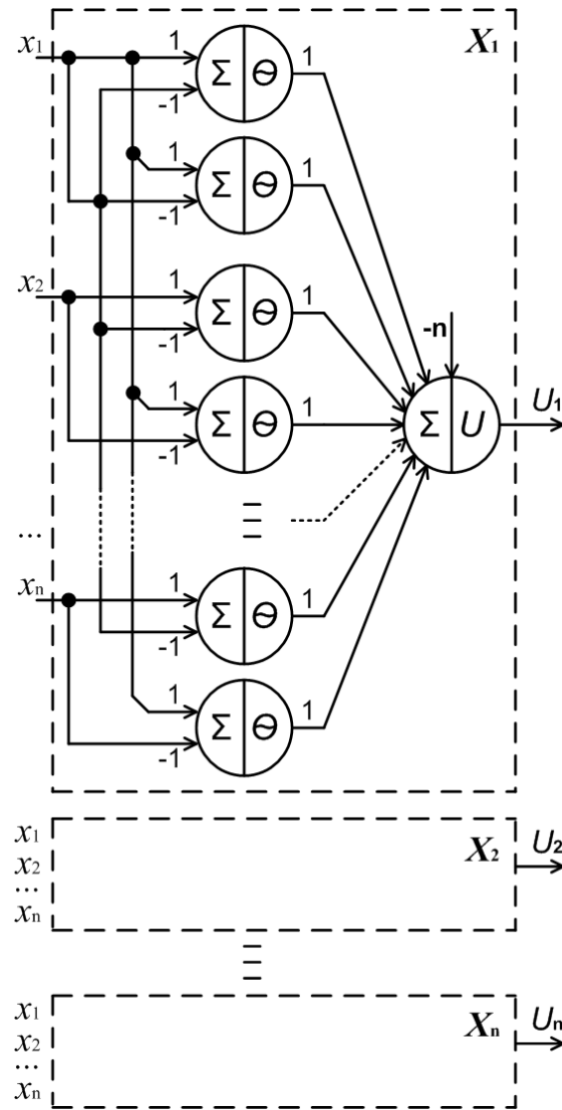


Рисунок 3.3 – Набір компонентів, що визначають унікальність кожного дискретного сигналу

Інший підхід полягає у використанні функції підрахунку кількості унікальних значень зміщених вхідних синаптичних сигналів тільки в одному напрямку [51]:

$$\begin{aligned}
 C_{>}(\bar{x}, i) &= \sum_{j=1}^n [\theta(x_i - x_j) + \theta(x_j - x_i) - 1] = \\
 &= \sum_{j=1}^n \theta(x_i - x_j) + \sum_{j=1}^n \theta(x_j - x_i) - n + i - 1
 \end{aligned} \tag{3.6}$$

Як результат проведених перетворень, замість виразу (3.3) отримаємо вираз (3.6).

Для порівняння $C_{>}(\bar{x}, i)$ та $C(\bar{x}, i)$ використаємо як приклад $\bar{x} = \overline{(2, 2, 2)}$, тоді отримаємо (див. табл. 3.1):

Таблиця 3.1 – Порівняння функцій $C(\bar{x}, i)$ та $C_{>}(\bar{x}, i)$

i	$C(\bar{x}, i)$	$C_{>}(\bar{x}, i)$
1	3	3
2	3	2
3	3	1

Як можна побачити $C(\bar{x}, i)$ порівнює i -ий елемент з усіма елементами вектору, а $C_{>}(\bar{x}, i)$ – тільки із i -им та наступними.

В такому випадку можна записати:

$$\begin{aligned}
 Q(\bar{x}) &= \sum_{i=1}^n \theta(1 - C_{>}(\bar{x}, i)) = \\
 &= \sum_{i=1}^n \theta \left(1 - \left[\sum_{j=1}^n \theta(x_i - x_j) + \sum_{j=1}^n \theta(x_j - x_i) - n + i - 1 \right] \right) = \\
 &= \sum_{i=1}^n \theta \left(n + 2 - i - \sum_{j=1}^n \theta(x_i - x_j) + \sum_{j=1}^n \theta(x_j - x_i) \right)
 \end{aligned} \tag{3.7}$$

Використовуючи кількість унікальних станів $Q(\bar{x})$ як заміну функції оцінки інформаційної ентропії в (2.1) отримаємо вираз:

$$y = \sigma(Q(\bar{x} + \bar{w})) \tag{3.8}$$

який, на основі запропонованої моделі (3.7), можна представити у вигляді штучної нейронної мережі, що реалізується на основі типових (кореляційних) перцептронів, структуру подано на рис.3.4.

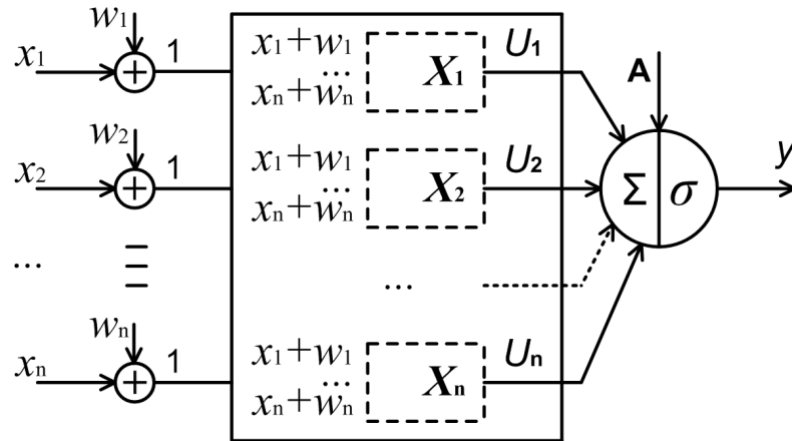


Рисунок 3.4 – Штучна нейронна мережа, яка емулює дискретний перцептрон зі зміщеними синаптичними сигналами

Доцільно зазначити, що в поданій на рис.3.4 структурній схемі на вхід компоненту визначення унікальності дискретного сигналу подаються уже зміщені синаптичні сигнали.

В результаті отримане еквівалентне представлення ймовірнісного дискретного перцептрону на основі класичної нейронної мережі є суттєво складнішим структурно та більш громіздкими щодо обчислювальних затрат, однак у перспективі це може спростити вирішення задач пов'язаних з машинним навчанням штучних нейронних компонентів та нейронних мереж, які реалізовані на основі дискретних перцептронних структур.

3.2 Апроксимація індикаторних функцій активації у контексті ймовірнісного дискретного перцептрона

Перцептрон, як окремий компонент, традиційно реалізується на основі функцій кореляції та згортки, результат розрахунку яких в подальшому опрацьовуються активаційними функціями. Використання статистичного оцінювання вхідних сигналів дозволяє розширити функціонал таких компонентів за межі лінійного розділення. У проведених дослідженнях основну увагу приділено розробці нових методів апроксимації активаційних функцій, що забезпечують гладкість і диференційованість, необхідні для ефективного машинного навчання. Застосування таких методів дозволяє знизити обчислювальну складність та підвищити точність роботи як перцептрону так і нейронної мережі, що відкриває нові перспективи щодо їх використання при вирішенні прикладних задач.

У загальному випадку, перцептрон, який реалізує опрацювання дискретних вхідних синаптичних сигналів на основі ймовірнісного оцінювання [51, 52], що представлено в (2.1) доцільно моделювати гладкими функціями.

В ході дослідження перцептронних структур вираз (3.1) подавався через порогову функцію Хевісайда, що є незручним рішенням для застосування градієнтного навчання оскільки згадана функція недиференційована. Тому замість неї зазвичай використовують гладкі апроксимації, зокрема логістичну сигмоїду $\theta_a(x) = \frac{1}{1 + e^{-kx}}$, арктангенс $\theta_a(x) = \frac{1}{2} + \frac{1}{\pi} \arctan(kx)$, функцію помилок Гауса $\theta_a(x) = \frac{1}{2}(1 + \operatorname{erf}(kx))$ тощо [74, 75]. Однак, підстановка цих апроксимацій у вираз (3.2) показує, що для випадку $x_1 = x_2$ всі вони приймають значення рівне 0 замість 1, що унеможлиблює їх застосування у такому контексті без додаткових модифікацій.

Інший підхід, описаний у [76, 77, 78], полягає у використанні індикаторної функції $\mathbf{1}_0(x)$, у цьому випадку:

$$y = C(x_1, x_2) = \mathbf{1}_0(x_1 - x_2) \quad (3.9)$$

Опираючись на вираз (3.9) структура перцептрону, що обчислює $C(x_1, x_2)$ на два синаптичні входи, матиме вигляд, рис.3.5.

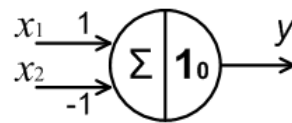


Рисунок 3.5 – Перцептрон для порівняння двох сигналів на рівність на основі індикаторної функції

В такому випадку, замість виразів (3.3) та (3.5), відповідно отримаємо вирази (3.10) та (3.11):

$$C(\bar{x}, i) = \sum_{j=1}^n \mathbf{1}_0(x_j - x_i) \quad (3.10)$$

$$Q(\bar{x}) = \sum_{i=1}^n \frac{1}{C(\bar{x}, i)} = \sum_{i=1}^n \frac{1}{\sum_{j=1}^n \mathbf{1}_0(x_j - x_i)} \quad (3.11)$$

Базуючись на виразі (3.11), можна отримати нейромережу, здатну визначати кількість унікальних елементів, структуру якої подано на рис. 3.6.

Аналогічно, для виразів (3.6) та (3.7), отримаємо вирази (3.12) та (3.13):

$$C_{>}(\bar{x}, i) = \sum_{j=i}^n \mathbf{1}_0(x_j - x_i) \quad (3.12)$$

$$Q(\bar{x}) = \sum_{i=1}^n \theta(1 - C_{>}(\bar{x}, i)) = \sum_{i=1}^n \theta \left(1 - \sum_{j=1}^n \mathbf{1}_0(x_j - x_i) \right) \quad (3.13)$$

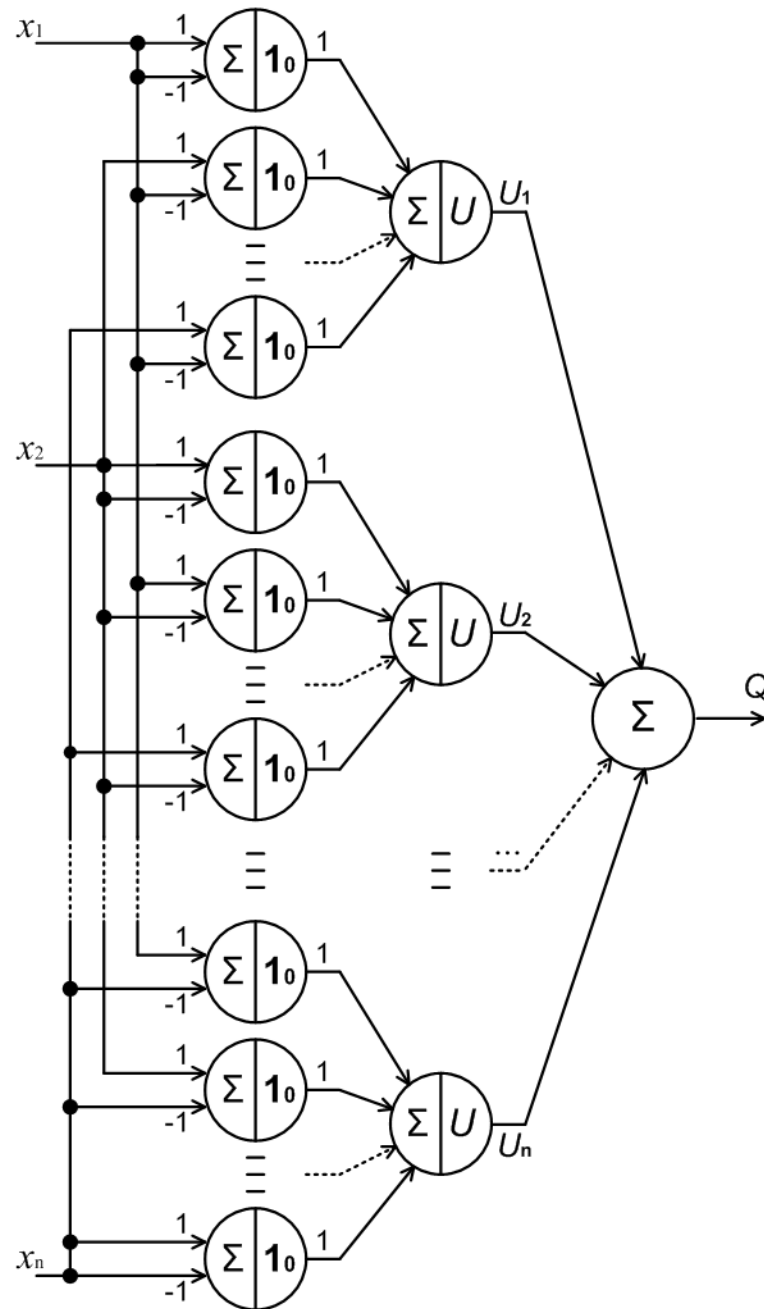


Рисунок 3.6 – Структура нейронної мережі, що визначає кількість унікальних станів

Однак, у виразі (3.13) все ще використовується функція Хевісайда, що незручна для емуляції операції порівнювання, провівши її заміну на індикаторну функцію, отримаємо:

$$Q(\bar{x}) = \sum_{i=1}^n \mathbf{1}_0(C_>(\bar{x}, i)) = \sum_{i=1}^n \mathbf{1}_0\left(\left[\sum_{j=1}^n \mathbf{1}_0(x_j - x_i)\right] - 1\right) \quad (3.14)$$

Доцільно зазначити, що індикаторна функція $\mathbf{1}_0(x)$ також непридатна для градієнтного методу машинного навчання, тому її необхідно замінювати гладкою диференційованою апроксимацією. Як приклади таких апроксимацій можна використати функцію Гауса $f(x) = e^{-kx^2}$, функцію Лоренца $f(x) = \frac{1}{1 + kx^2}$, гіперболічний секанс $f(x) = \text{sech}(kx) = \frac{2}{e^{kx} + e^{-kx}}$, функцію Лапласа $f(x) = e^{-k|x|}$ тощо [79, 80, 81, 82], результати дослідження апроксимацій цих функцій див. рис.3.7, а програмна реалізація подана у додатку В.

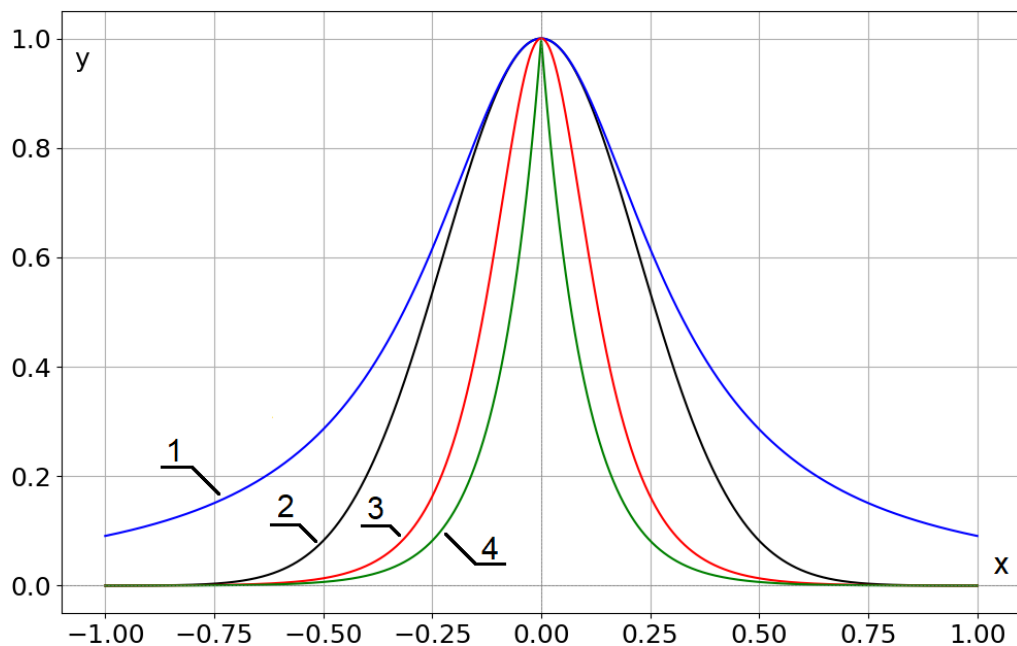


Рисунок 3.7 – Функції, які апроксимують індикаторну функцію $\mathbf{1}_0(x)$:

1 - Лоренца, 2 – Гауса 3 - гіперболічний секанс, 4 - Лапласа

Аналіз графіків цих функцій при $k=10$ в околі $x=0$, як показано на рис.3.7, показує, що форму індикаторної функції найточніше відтворює гіперболічний секанс і функція Лапласа. Оскільки остання містить модуль i , відповідно,

немає похідної в точці 0, доцільно обрати саме гіперболічний секанс як основну апроксимацію у подальших розрахунках [78].

Використавши $\text{sech}(kx)$ у виразах (3.11) та (3.14), отримаємо апроксимації $Q(\bar{x})$ придатні для градієнтного навчання, а саме:

$$Q(\bar{x}) \approx \sum_{i=1}^n \frac{1}{\sum_{j=1}^n \text{sech}(k(x_j - x_i))} \quad (3.15)$$

$$Q(\bar{x}) \approx \sum_{i=1}^n \text{sech} \left(k \left(\left[\sum_{j=1}^n \text{sech}(x_j - x_i) \right] - 1 \right) \right) \quad (3.16)$$

Таким чином, отримано гладку апроксимацію дискретної характеристики $Q(\bar{x})$, яка підраховує кількість унікальних станів на входах, що придатна для градієнтного спуску. На наступному етапі необхідно реалізувати перетворення $Q(\bar{x})$ у вихідний сигнал (реакцію) перцептрона за допомогою функції активації (ФА).

Перед вибором активаційних функцій слід розглянути поведінку функції $Q(\bar{x})$ [76, 77, 78, 83], що визначає кількість унікальних станів. Така функція є дискретною та приймає цілі значення на проміжку $[1; n]$, де n – кількість входів перцептрону. На відміну від лінійної суми, яка зростає із зростанням компонент (та корелюється із "силою сигналу"), $Q(\bar{x})$ може поводитися нелінійно. Тобто однакова зміна вагових коефіцієнтів W може як збільшити так і зменшити кількість унікальних станів або залишити кількість цих станів без змін. Зокрема, якщо кілька вхідних сигналів (координат) x_i стали рівними після операції зміщення (додавання w_i) то загальна кількість унікальних станів зменшилася, і навпаки.

Функції активації в перцептронах відіграють важливу роль, зокрема додають нелінійність до моделі, а також обмежують вихідні значення нейрона в певному діапазоні, що допомагає у стабілізації процесу машинного

навчання. Основними ФА, які використовуються в перцептронах є сходи́нкова (ступі́нчата) та сигмої́дна, а також часто використовують гіперболі́чний тангенс, ReLU, LeakyReLU, ELU та SoftMax, основні аспекти використання яких подано в таблиці 3.2.

Таблиця 3.2 – Типові функції активації, що використовують у перцептронних структурах

Функція активації	Застосування
Сходи́нкова (Threshold): $f(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$	Використовується в класичному перцептроні для задач, де потрібно приймати рішення між двома класами так, як вона забезпечує чітке розділення класів, проте вона не дозволяє оцінити "ймовірність" або ступінь впевненості у рішенні та не підходить для задач з багатьма класами або нелінійними залежностями
Сигмої́дна (Sigmoid): $\sigma(x) = \frac{1}{1 + e^x}$	Особливістю є область визначення (0;1), що добре підходить для задач класифікації (наприклад, бінарної класифікації), проте у неї є й недоліки: проблема зникнення градієнта через те, що похідна стає дуже малою для великих значень x та вихідне значення не центроване відносно 0, що може уповільнити навчання
Гіперболі́чний тангенс (Tanh): $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	Основною особливістю є область визначення (-1;1), що надає їй перевагу над сигмоїдною функцією, оскільки вихідне значення відцентроване відносно 0, фактично вилучено можливе зміщення, проте вона також може мати проблему зникнення градієнта

Крім того, як ФА використовуються різні нелінійні модифікації лінійної функції (Linear Unit), подано в табл. 3.3, тобто варіанти активаційних функцій, які зберігають або адаптують її лінійну або частково лінійну властивість для подолання обмежень базового перцептрону.

Модифікації лінійної функції (LU) забезпечують кращу виразну здатність перцептронів, зменшують проблему зникання градієнта, сприяють розрідженню активацій і, завдяки простоті (особливо у ReLU та її варіаціях), дозволяють ефективну реалізацію на апаратному рівні.

Таблиця 3.3 – Варіанти функції активації Linear Unit, що використовують у перцептронних структурах

Функція активації	Застосування
ReLU (Rectified Linear Unit): $f(x) = \max(0, x)$	Особливість цієї функції полягає в тому, що вона швидко обчислюється і добре працює для глибоких мереж та усуває проблему зникнення градієнта в позитивній області. Проте її використання призводить до проблеми "мертвих нейронів" (вихідне значення завжди дорівнює 0) для випадку $x \leq 0$
LeakyReLU: $f(x) = \begin{cases} x, & x > 0 \\ ax, & x \leq 0 \end{cases}$	У порівнянні з RELU, вона вирішує проблему "мертвих" нейронів, допускаючи невеликий градієнт для випадків $x \leq 0$
ELU (Exponential Linear Unit): $f(x) = \begin{cases} x, & x > 0 \\ a(e^x - 1), & x \leq 0 \end{cases}$	Основною особливістю цієї функції є плавний перехід для негативних значень і порівнянні із ReLU

До спільних недоліків слід віднести: потенційну проблему мертвих нейронів (особливо у ReLU), потребу в налаштуванні або навчанні додаткових параметрів (як у PReLU), обмежену гладкість (у більшості варіантів, крім ELU та Softplus), а також зменшення узагальнюючої здатності при обмеженому діапазоні активацій (наприклад, у Clipped ReLU) [84].

З метою ефективною маніпуляції вихідними сигналами, перцептрон повинен мати можливість адекватно змінювати коефіцієнти зміщення W . Однак результат маленької зміни кожного w_i не завжди плавно змінює Q .

Часто при невеликих зміщеннях вихід перцептрону залишається незмінним, доки певні компоненти не співпадуть/розійдуться. Тобто функція агрегації змінюється стрибкоподібно, при чому в багатьох точках залишається сталою, що ускладнює або й унеможлиблює використання методів градієнтного навчання.

Таким чином, якщо використовувати для машинного навчання підходи, що ґрунтуються на методах зворотного поширення помилки, то часто виникатиме ситуація з постійними нульовими "градієнтами", оскільки майже всюди похідна за дискретним зсувом буде нуль, а в точках перестановок буде розрив.

Отже, запропонована структура в [51, 52] дискретного перцептрону, на поточному етапі досліджень, є перспективною для дискретної, зокрема бінарної, логіки чи комбінаторних задач, а не для класичного безперервного машинного навчання. Якщо ж машинне навчання реалізувати не градієнтними методами (перебором, генетичними алгоритмами чи іншими еволюційними/комбінаторними методами), то подібні активації можуть бути актуальні і для інших прикладних задач.

В ході проведених досліджень перцептронних структур [78] розглядалися такі ФА (3.17, 3.19, 3.21, 3.23).

$$f(x, A) = \begin{cases} 0, & x < A \\ 1, & x \geq A \end{cases} \quad (3.17)$$

Фактично це загальновідома функція Хевісайда, що ділить множину значень на дві підмножини, які дають значення 0 та 1, вона стрибкоподібно змінюється на пороговому значенні A і може бути записана так:

$$f(x, A) = \theta(x - A) \quad (3.18)$$

Порогова функція у вихідному шарі нейронної мережі для задач двокласової класифікації така:

$$f(x, A) = \begin{cases} 0, & x < A \wedge A \geq 0 \vee x \geq -A \wedge A < 0 \\ 1, & x \geq A \wedge A \geq 0 \vee x < -A \wedge A < 0 \end{cases} \quad (3.19)$$

Ця функція є модифікованою пороговою функцією, що враховує знак параметру A . Таким чином, послуговуючись рівністю (3.18) її можна представити так:

$$f(x, A) = \begin{cases} \theta(-x - A), & x < A \\ \theta(x - A), & x \geq A \end{cases} \quad (3.20)$$

В обох режимах ($A \geq 0$ та $A < 0$) така функція залишається бінарною і при $A \geq 0$ збігається з класичною пороговою функцією (3.18). Для випадку $A < 0$ отримуємо інвертовану порогову функцію.

Для типових реалізацій перцептронів така функція активації не використовується, оскільки не надає жодних переваг у порівнянні із класичною пороговою функцією. Однак, у разі застосування порогової функції Хевісайда $\theta(x - A)$ до величини x , яка відповідає кількості унікальних значень вхідних дискретних сигналів, виникає природне обмеження на діапазон A , оскільки мова йде про мінімально допустиму кількість різних станів у векторі сигналу (синаптичних входах). Слід зауважити, що таке обмеження може бути надто звуженим для задач, у яких важливо, визначати не тільки "чи вистачає унікальних значень?", але й "чи не є їх забагато?". У подібних випадках корисно мати можливість задати від'ємне A , щоб інтерпретувати від'ємні пороги як "верхню межу" кількості унікальних значень або реалізувати інверсну логіку активації.

Таким чином, модифікована порогова функція, яка коректно реагує і на додатний, і на від'ємний поріг A , істотно розширює сферу її застосувань. Додавання підтримки від'ємних значень A дає змогу: керувати поведінкою перцептрона у випадках, коли слід вимикати активацію при надмірно великій кількості унікальних значень та забезпечувати більшу гнучкість у побудові логічних умов. Зокрема, поріг $A > 0$ відповідає умові "активуватися, якщо унікальних елементів не менше A ", тоді як поріг $A < 0$ задає протилежний сенс – "активуватися, якщо унікальних елементів не більше $-A$ ". Таким чином, одна й та ж функція активації може універсально задовольняти обидва типи порогових перевірок.

З погляду практичної реалізації, таке узагальнення не змінює основних дискретних властивостей функції Хевісайда, проте надає додаткові можливості для логічної інтерпретації виходу. Зокрема, в контексті нетрадиційних перцептронних моделей, що працюють із визначенням унікальних значень, від'ємні A перетворюють бінарне рішення "більше/менше за поріг" на більш універсальну умову "виходу кількості унікальних ознак за межі діапазону". Такий підхід суттєво спрощує побудову комбінаторних чи логічних правил без упровадження додаткових елементів у модель.

$$f(x, A) = \begin{cases} 0, & x \neq A \\ 1, & x = A \end{cases} \quad (3.21)$$

Функція (3.21) це індикаторна функція рівності (3.22), яка перевіряє чи вхід x точно рівний заданому параметру A :

$$f(x, A) = \mathbf{1}_0(x - A) \quad (3.22)$$

Подана функція не використовується як активаційна у класичних перцептронах через вузький діапазон активації (єдине значення).

$$f(x, A) = \begin{cases} 0, & x \neq A \wedge A \geq 0 \vee x = -A \wedge A < 0 \\ 1, & x = A \wedge A \geq 0 \vee x \neq -A \wedge A < 0 \end{cases} \quad (3.23)$$

Ця функція також є модифікованою індикаторною функцією, що поводитьсь по різному залежно від знаку параметру A . В результаті рівність (3.23) можна представити так:

$$f(x, A) = \begin{cases} \mathbf{1}_0(x - A), & x \geq 0 \\ 1 - \mathbf{1}_0(x + A), & x < A \end{cases} \quad (3.24)$$

Як і попередня, подана модифікована індикаторна функція теж не використовується як активаційна у класичних перцептронах через її занадто широкий діапазон активації (виключене тільки єдине значення).

Основним обмеженням розглянутих функцій є їх негладкість, що вимагає спеціальних методів машинного навчання (зокрема, методи нескінченно малих збурень, евристики типу прямопрохідного оцінювача або еволюційні алгоритми), оскільки стандартний підхід, що ґрунтується на зворотному поширенні помилки не оновлює ваги при нульових похідних.

Однак, для випадку запропонованого дискретного перцептрону, такі функції активації мають суттєву перевагу у простоті та інтерпретованості, що фактично дозволяє розглядати перцептрон як окремий поліфункціональний логічний компонент. Мається на увазі, що емулюючи будь-які логічні функції запропонований дискретний перцептрон, що використовує одну з вище описаних (3.17, 3.19, 3.21, 3.23) функцій активації, структурно залишається незмінним.

Така особливість дозволяє розглянути питання створення програмованих логічних інтегральних схем (ПЛІС) на базі ідентичних компонентів, які в залежності від потреби будуть реалізовувати будь-яку функцію бінарної логіки.

З поміж двох сценаріїв використання дискретного перцептрона: 1 - як програмованого логічного елементу, де негладкі активації зручні та цілком виправдані; 2 - як частини нейронної мережі, що навчається градієнтно - другий знову зумовлює появу проблеми відсутності похідної [79, 80, 85]. Тобто, якщо вийти за межі чисто логічної інтерпретації і перейти до оптимізації ваг методом зворотного поширення помилки, негладкі функції (3.18, 3.20, 3.22, 3.24) стають непридатні. В такій ситуації доцільно зосередитись на пороговій функції Хевісайда (3.18) та індикаторній функції (3.22) і дослідити, як стандартно її замінити гладкими неперервними апроксимаціями, зокрема такими як логістична сигмоїда, арктангенс тощо.

Оскільки функція (3.17) – це функція Хевісайда (3.18), то відомо багато варіантів її емуляції за допомогою неперервних функцій, зокрема логістичної функції (3.25), арктангенса (3.26) [74, 75, 86]:

$$\theta(x) \approx \sigma_k(x) = \frac{1}{1 + e^{-kx}} = \frac{1}{2} \left(1 + \tanh\left(\frac{kx}{2}\right) \right) \quad (3.25)$$

$$\theta(x) \approx \frac{1}{2} + \frac{1}{\pi} \arctan(kx) \quad (3.26)$$

де k – коефіцієнт крутизни нахилу, збільшення якого дозволяє отримати кращу апроксимацію, проте за рахунок збільшення значень похідних утруднюємо застосування методу градієнтного спуску для пошуку оптимумів.

Для емуляції модифікованої функції Хевісайда неперервною необхідно позбутися умови. Для цього скористаємося функцією знаку (3.27) та тим, що $A=0$ не є цікавою у нашому випадку (так як кількість унікальних станів завжди більша 0), підставивши її у (3.20), отримаємо (3.28):

$$\text{sign}(x) = \begin{cases} -1, & x < 0 \\ 0 & x = 0 \\ 1, & x > 0 \end{cases} \quad (3.27)$$

$$f(x, A) = \theta(\text{sign}(A)x - A) \quad (3.28)$$

Скориставшись тим, що $\text{sign}(x) \approx \tanh(kx)$, отримаємо:

$$f(x, A) \approx \frac{1}{2} \left(1 + \tanh \left(\frac{k(x \cdot \tanh(kA) - A)}{2} \right) \right) \quad (3.29)$$

Індикаторну функцію (3.22), як було продемонстровано вище, доцільно емулювати за допомогою гіперболічного секанса, в такому випадку:

$$f(x, A) = \mathbf{1}_0(x - A) \approx \text{sech}(k(x - A)) \quad (3.30)$$

Для емуляції модифікованої індикаторної функції неперервною необхідно позбутися умови. Скориставшись знаковою функцією (3.27) та виразом (3.24) для модифікованої індикаторної функції, отримаємо:

$$f(x, A) = \begin{cases} \mathbf{1}_0(x - \text{sign}(A)A), & A \geq 0 \\ 1 - \mathbf{1}_0(x + \text{sign}(A)A), & A < 0 \end{cases} \quad (3.31)$$

Оскільки $A=0$ не є інформативною у нашому випадку, так як кількість унікальних станів завжди більша 0, тоді отримаємо:

$$f(x, A) \approx \frac{1}{2} + \text{sign}(A) \left(\mathbf{1}_0(x - \text{sign}(A)A) - \frac{1}{2} \right) \quad (3.32)$$

Скориставшись тим, що $\text{sign}(x) \approx \tanh(kx)$, отримаємо:

$$f(x, A) = \frac{1}{2} + \tanh(kA) \left(\text{sech}(k(x - \tanh(kA)A)) - \frac{1}{2} \right) \quad (3.33)$$

В результаті, отримані функції (3.25), (3.29), (3.30) та (3.33) дозволяють скористатися методом градієнтного спуску за рахунок емуляції дискретних функцій (3.17), (3.19), (3.21) та (3.23) неперервними.

Таким чином, використання гладких диференційованих апроксимацій замість класичних активаційних функцій дозволяє подолати проблеми, пов'язані з недиференційованістю, і забезпечити ефективне машинне навчання такої неромережі.

Запропоновані рішення мають перспективи для практичної імплементації, оскільки дозволяють уніфікувати, а також суттєво знизити схемну, алгоритмічну та програмну складність цифрових компонентів комп'ютерних систем.

3.3 Алгоритмічні рішення компонентів дискретного перцептрону зі зміщеними синаптичними сигналами

Конструктивно структура перцептрону, як елементарної складової нейронної мережі, традиційно може бути представлена трьома умовно-окремим компонентами: компонент перетворення входних сигналів (Input Transformation), компонент агрегування (Aggregation) та компонент формування результату (Output Transformation чи Activation), що реалізує активаційну функцію [87], рис. 3.8.

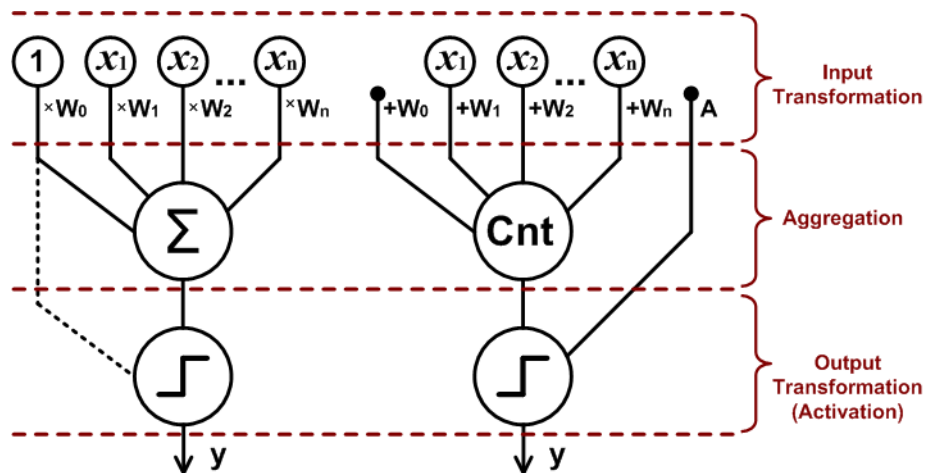


Рисунок 3.8 – Основні функціональні компоненти традиційного та дискретного перцептронів

Компонент Input Transformation реалізує перетворення таких входних сигналів x_i , як: бінарні $x_i \in \{0,1\}$, зокрема для логічних схем; дійсні $x_i \in \mathcal{R}$ (загальний випадок); нормалізовані $x_i \in [-1,1]$ або $[0,1]$; стохастичні або ймовірнісні, зокрема для ймовірнісного перцептрону, де x_i представляють параметри розподілу. В свою чергу вагові коефіцієнти w_i можуть бути: фіксовані, навчені (одним з методів: методом градієнтного спуску, правилом перцептрону, тощо); дискретні $x_i \in \{-1,0,1\}$ в логічних або апаратних реалізаціях та стохастичні або квантизовані ваги (у випадку low-precision або hardware-орієнтованих моделей) [20, 29, 51, 52, 87, 88].

Компонент Aggregation реалізує опрацювання перетворених вхідних сигналів на основі різних математичних інструментів, зокрема: сумування із зсувом на величину b та без нього $z = \sum x_i w_i + b$ (типова реалізація), множинна агрегація $z = \frac{1}{n} \sum x_i w_i$; мін/макс-агрегація, зокрема $z = \max_i (x_i w_i)$ (використовується в деяких fuzzy logic системах); стохастичне підсумовування – обчислення у випадковому порядку з додаванням шуму, а також різного роду функції статистичного оцінювання, зокрема інформаційної ентропії, кількості можливих станів [51, 52], тощо.

Компонент Output Transformation чи Activation – функція оцінювання результату агрегації для прийняття рішення, щодо формування вихідного сигналу перцептрону і може бути реалізована як:

- дискретна (класична): порогова (Heaviside), зворотна порогова;
- негладка, зокрема знакова;
- гладка апроксимація, використовується для градієнтного навчання, зокрема: сигмоїда, гіперболічний тангенс, ReLU, Softsign, GELU, Swish, тощо;
- дискретно-апроксимована, зокрема: piecewise-linear, таблично-реалізована, тощо [20].

Структурно реалізація зі зміщеними синаптичними сигналами (рис. 3.8, схема справа) фактично не відрізняється від типової (рис. 3.8, схема зліва) і включає ті ж три основні компоненти. Обидві моделі включають три послідовні етапи: вхідне перетворення, агрегацію сигналів та вихідну активацію. Однак слід зауважити, що компонент Input Transformation не потребує використання операції множення на відміну від типової структури, що суттєво зменшує обчислювальні затрати необхідні для використання цього компоненту, блок схема подана на рис. 3.9а.

Алгоритмічна реалізація компонент Aggregation для дискретного перцептрону зі зміщеними сигналами синапсів складніша за типову реалізацію, оскільки передбачає визначення кількості унікальних станів

вхідних сигналів після зміщення (Cnt()) [52]. Для перцептронів на два входи задача визначення кількості унікальних станів тривіальна і реалізується на основі єдиної операції порівняння, див рис. 3.9б. Однак, при збільшенні кількості вхідних сигналів такий підхід пропорційно ускладнюється, що потребує дослідження можливостей реалізації функції (Cnt()) [89].

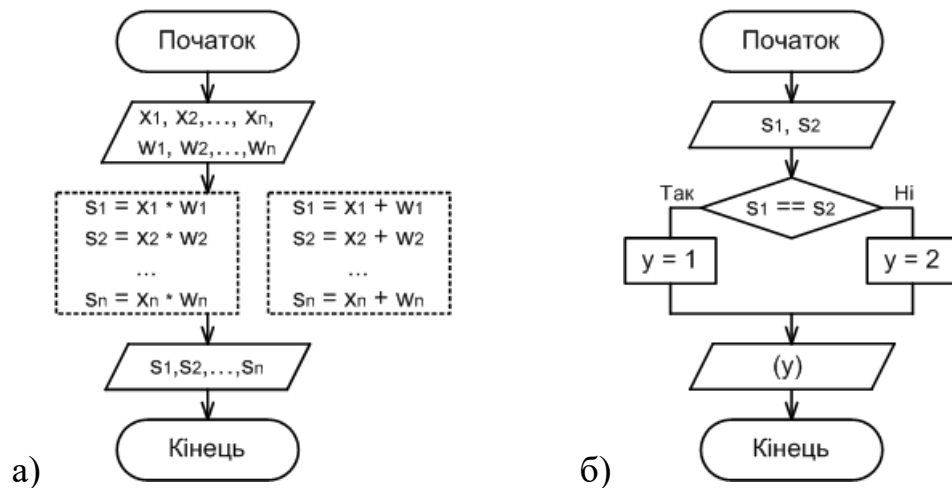


Рисунок 3.9 – Алгоритми реалізації функцій а) Input Transformation, б) Aggregation (Cnt()) дискретного перцептронів на 2-ва входи

Умовні конструкції характеризуються більшою тривалістю порівняно із логічними. Тобто, для визначення кількості унікальних станів слід розглянути можливість заміни надлишкових операцій порівнювання логічними виразами, які нескладно синтезувати на основі таблиць істинності [90]. Мається на увазі, що необхідні операції попарного порівнювання зміщених вхідних сигналів проводяться один раз, зі збереженням біту результату (як окремого бітового компоненту) з подальшим опрацюванням цих бітів у відповідних логічних виразах. Зокрема, таблиця істинності на три інформаційні входи подана в таблиці 3.4. В поданій таблиці пропущені комбінації результатів порівнювання c_{12} , c_{13} , c_{23} відповідних пар вхідних сигналів $(x_1=x_2)$, $(x_1=x_3)$ та $(x_2=x_3)$ представляють не можливі стани і не потребують подальшого опрацювання. y_1 , y_0 – старший та молодший біти результату відповідно, Y – десяткове значення результату.

Таблиця 3.4 – таблиця істинності для бітів результату трьох входового дискретного перцептронну

$c_{12} (x_1=x_2)$	$c_{13} (x_1=x_3)$	$c_{23} (x_2=x_3)$	y_1	y_0	Y
0	0	0	1	1	3
0	0	1	1	0	2
0	1	0	1	0	2
1	0	0	1	0	2
1	1	1	0	1	1

На основі таблиці 3.4 логічні вирази для старшого y_1 та молодшого бітів y_0 результату агрегації визначають:

$$y_1 = \overline{c_{12} \wedge c_{13} \wedge c_{23}} \quad (3.34)$$

$$y_0 = (c_{12} \wedge c_{13} \wedge c_{23}) \vee \overline{(c_{12} \vee c_{13} \vee c_{23})} \quad (3.35)$$

Аналогічно сформовано таблицю істинності для випадку 4-ох входового дискретного перцептронну, див. табл. 3.5.

Таблиця 3.5 – таблиця істинності для бітів результату чотирьох входового дискретного перцептронну

c_{12}	c_{13}	c_{14}	c_{23}	c_{24}	c_{34}	y_2	y_1	y_0	Y
0	0	0	0	0	0	1	0	0	4
0	0	0	0	0	1	0	1	1	3
0	0	0	0	1	0	0	1	1	3
0	0	0	1	0	0	0	1	1	3
0	0	0	1	1	1	0	1	0	2
0	0	1	0	0	0	0	1	1	3
0	0	1	1	0	0	0	1	0	2
0	1	0	0	0	0	0	1	1	3
0	1	0	0	1	0	0	1	0	2
0	1	1	0	0	1	0	1	0	2
1	0	0	0	0	0	0	1	1	3
1	0	0	0	0	1	0	1	0	2
1	0	1	0	1	0	0	1	0	2
1	1	0	1	0	0	0	1	0	2
1	1	1	1	1	1	0	0	1	1

Як і в попередньому випадку, пропущені комбінації результатів порівнювання c_{12} , c_{13} , c_{14} , c_{23} , c_{24} , c_{34} відповідних вхідних сигналів $(x_1=x_2)$, $(x_1=x_3)$, $(x_1=x_4)$, $(x_2=x_3)$, $(x_2=x_4)$, $(x_3=x_4)$ представляють не можливі стани і не потребують опрацювання. На основі таблиці 3.5 логічні вирази для бітів результату агрегації від старшого y_2 до молодшого y_0 матимуть вигляд:

$$y_2 = \overline{c_{12} \vee c_{13} \vee c_{14} \vee c_{23} \vee c_{24} \vee c_{34}} \quad (3.36)$$

$$y_1 = (c_{12} \vee c_{13} \vee c_{14} \vee c_{23} \vee c_{24} \vee c_{34}) \wedge \overline{(c_{12} \wedge c_{13} \wedge c_{14} \wedge c_{23} \wedge c_{24} \wedge c_{34})} \quad (3.37)$$

$$y_0 = (c_{14} \wedge c_{24} \wedge c_{34}) \vee (\overline{c_{12}} \wedge \overline{c_{13}} \wedge \overline{c_{23}} \wedge c_{34}) \vee (\overline{c_{12}} \wedge \overline{c_{13}} \wedge c_{23} \wedge \overline{c_{24}}) \vee (\overline{c_{12}} \wedge \overline{c_{13}} \wedge c_{14} \wedge \overline{c_{23}}) \vee (\overline{c_{12}} \wedge c_{13} \wedge \overline{c_{14}} \wedge \overline{c_{24}}) \vee (c_{12} \wedge \overline{c_{13}} \wedge \overline{c_{14}} \wedge \overline{c_{34}}) \quad (3.38)$$

Для порівняння ефективності різних підходів розроблено та проведено дослідження блок-схем алгоритмів реалізації функції агрегації для дискретного перцептрону на три входи, що ґрунтуються на використанні винятково умовних конструкцій, рис. 3.10а, та комбінації умовних та логічних конструкцій, рис. 3.10б.

Алгоритм, блок-схема на рис. 3.10а, здійснює каскадне покрокове порівняння зміщених сигналів, а саме: $S1 == S2$, $S1 == S3$, $S2 == S3$ із вкладеними умовами. Кількість умов: до 3 умовних операторів, залежно від гілки виконання.

Складність коду:

- простий до реалізації, але з великою кількістю вкладеності (що ускладнює читання);
- повна перевірка структури умов вимагає уважного стеження за вкладеними if-else.

Алгоритм, блок-схема на рис. 3.10б, здійснює умовно-логічну агрегацію зміщених синаптичних сигналів (кількість умов: три базові порівняння та 2 складні булеві умови), а саме: спочатку формуються логічні змінні C12, C13, C23, а потім перевіряються їхні комбінації:

- $y = 1$ якщо всі C12, C13, C23 істинні (всі значення однакові);
- $y = 3$ якщо всі C12, C13, C23 хибні (всі значення різні);
- інакше $y = 2$ (є дублікати, але не всі однакові).

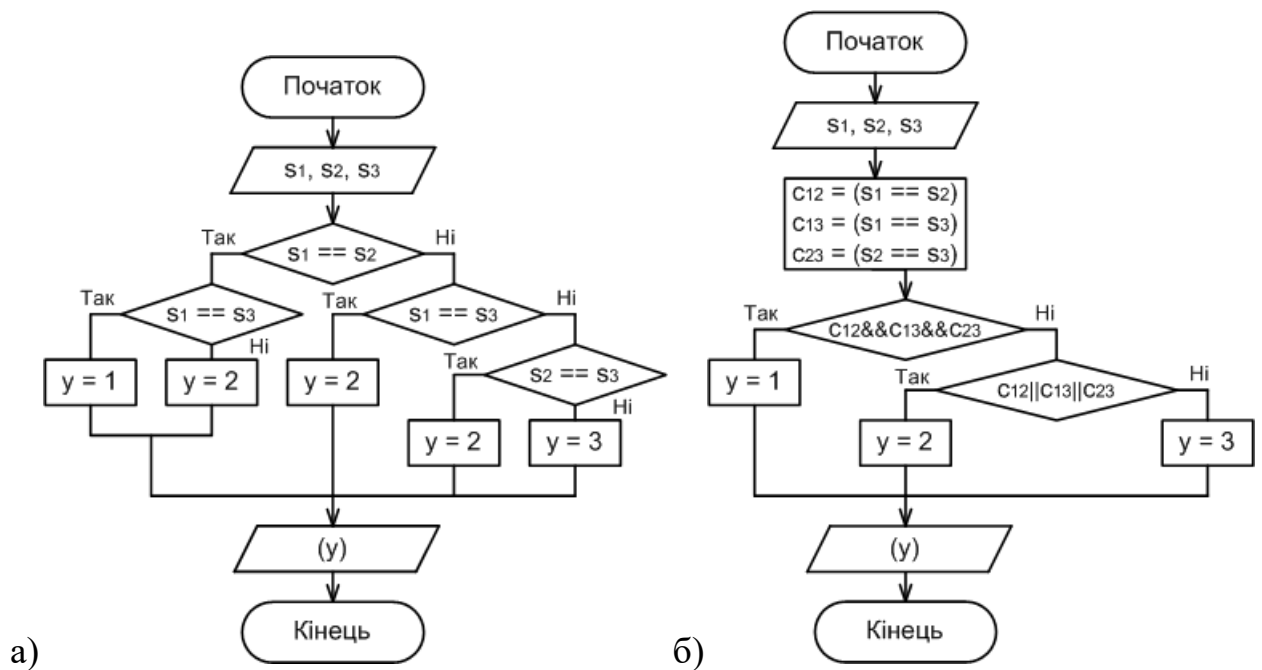


Рисунок 3.10 – Алгоритми реалізації функції Aggregation (Cnt()) перцептрону на 3-ри входи з використанням: а) умовних конструкцій, б) логічних операцій

Складність коду:

- менше вкладень, компактніша реалізація;
- зрозуміліша логіка програмування;
- підвищена читабельність і спрощене тестування.

Узагальнене порівняння представлених на рис. 3.10 алгоритмів за функціональними критеріями подано в таблиці 3.6.

Таким чином, з погляду програмної реалізації, блок-схема подана на рис. 3.10б, що ґрунтується на умовно логічній агрегації є більш ефективною та

елегантною, порівняно із блок-схемою поданою на рис. 3.10а, що ґрунтується на умовній агрегації, а саме:

- простіша логіка, яку легше реалізувати, налагоджувати й тестувати;
- є модульною та краще масштабується;
- надає чітку відповідність між логічними умовами та результатом.

Таблиця 3.6 – Порівняння алгоритмів компоненту агрегації за функціональними критеріями

Критерій	Блок-схема рис. 3.10а	Блок-схема рис. 3.10б
Кількість умов	До 3 вкладених if	3 базові + 2 булеві
Структура умов	Глибоко вкладена	Плоска, з булевими виразами
Реалізація мовою програмування	Більше if-else	Більше логічних виразів
Легкість модифікації	Нижча	Вища (зручно змінювати логіку)
Тестування	Ускладнене через вкладення	Простіше через незалежні умови
Читабельність	Нижча	Вища

З огляду на отримані результати, можна стверджувати, що використання винятково умовних конструкцій для програмної реалізації функції агрегації дискретного перцептрону є менш ефективним, порівняно із використанням умовно логічної агрегації, оскільки призводить до створення каскадних порівнянь глибокого вкладення, що в свою чергу створює значні відмінності у тривалості агрегації вхідних сигналів для різних комбінацій.

Для коректного та релевантного порівняння складності реалізації розглянутих (див. рис. 3.8) перцептронних структур, призначених для емуляції довільних двохвходових логічних функцій, доцільно оцінити витрати на їх обчислювальну складність у логічних елементах за підходом ASIC

(Application-specific integrated circuit), тобто в одиницях NAND2-еквівалентів (GE – Gateway Element) – універсальній мірі логічної площі інтегральних схем спеціального призначення (ІССП). Послугуючись типовими оцінками [91, 92, 93] основних обчислювальних блоків (таблиця 3.7) отримаємо, що для паралельної реалізації кореляційного перцептрона необхідно по 2 типові компоненти: FP-mul, FP-add та int $\rightarrow$ double, а також компоненти sign перевірка знаку операції $s < 0$ та MUX2, загалом $\approx 83k$ GE.

Таблиця 3.7 – Типові оцінки основних обчислювальних блоків у GE

Компонент	Кількість
INV (інвертор)	1
AND2, OR2, XOR2, XNOR2 (базові 2-вх. логічні компоненти)	2, 3, 4, 5
MUX2 (мультиплексор 2:1 – 1 біт)	4
HA (1-бітний напівсуматор)	6
FA (1-бітний суматор)	15
Adder_n – RCA (n-бітний суматор – ripple-carry)	$\approx 15 \cdot n$
Eq_n (порівняння на рівність)	$\approx 7n-2$
FP-add – double (IEEE-754 додавання – вирівн., нормал., округл.)	$\approx 2\,300$
FP-mul – double (IEEE-754 множення мантис + експоненти)	$\approx 39\,000$
int32 $\rightarrow$ double (перетворення типу – нормалізація/пакування)	≈ 300
FP sign<0 (перевірка знаку double – спрощено)	≈ 10

Для дискретного перцептрону потрібно: 2 - Adder32 (960 GE), 3 - Eq32 (666 GE), логіка (10 GE), 4 MUX2 (16 GE), загалом 1652 GE, що у 50 разів менше порівняно з кореляційним перцептроном.

Таким чином, використання одного кореляційного перцептрону еквівалентно використанню 50 дискретних перцептронів.

Крім того, доцільно розглянути ефективність запропонованої структури дискретного перцептрону в контексті емуляції бінарних логічних функцій

одним перцептроном з різною кількістю входів, результати досліджень подано в табл.3.8.

Таблиця 3.8 – Кількість логічних функцій, що емулюють перцептрони, для різної кількості бінарних входів

n	Загальна кількість функцій (2^{2^n})	Кількість функцій, що емулює перцептрон	
		кореляційний	дискретний
1	4	4	4
2	16	14	16
3	256	104	194
4	65536	1882	4518

Доцільно зазначити, що результати подані в табл.3.8 отримані шляхом пошуку в межах повної множини можливих комбінацій параметрів дискретного перцептрону і тільки для випадку чотирьох бінарних входів було застосовано екстраполяційні методи, що зумовлено значними обсягами обчислень при реалізації прямого пошуку. Як можна побачити, див. табл.3.8 запропонована структура дозволяє реалізувати усі логічні функції для 2-ох бінарних функцій, що недоступно для кореляційного перцептрону, а також 76%, порівняно із 41% для кореляційного перцептрону, логічних функцій для 3-ох бінарних входів.

Отже, з урахуванням більшої виразності дискретного перцептрону, таке рішення є перспективним з погляду імплементації в комп'ютерних системах та компонентах з обмеженими обчислювальними ресурсами.

ВИСНОВКИ ДО ТРЕТЬОГО РОЗДІЛУ

1. Розроблено модель дискретного перцептрона на основі багат шарової класичної нейронної мережі, що дозволяє застосувати типові методи машинного навчання нейромереж для визначення коефіцієнтів зміщення для запропонованого дискретного перцептрона.
2. Запропоновано використання гладких апроксимацій індикаторної та порогової функцій (зокрема на основі гіперболічного секансу та логістичної функції), що дозволяє адаптувати дискретний перцептрон до градієнтних методів машинного навчання.
3. Розроблено модифіковану порогову функцію, яка коректно працює як з додатними, так і з від'ємними порогами, що дає змогу реалізовувати як нижню, так і верхню межу активації на основі кількості унікальних значень зміщених вхідних сигналів.
4. Встановлено можливість використання дискретного перцептрона як поліфункціонального логічного елементу, здатного реалізовувати широкий спектр логічних операцій, зокрема за рахунок вибору відповідної активаційної функції та коефіцієнтів зміщення.
5. Розроблено каскадно-умовний та умовно-логічний алгоритми компоненту визначення кількості унікальних станів і встановлено, що реалізації на основі логічних функцій мають нижчу часову складність, дозволяють покращити ефективність схемної/програмної реалізації та масштабованість на більшу кількість входів.
6. Отримано логічні вирази для визначення кількості унікальних станів для 3-ох та 4-ох входових перцептронів, що дозволяє мінімізувати кількість умовних конструкцій при алгоритмічній та апаратній реалізаціях дискретних компонентів такого типу.

РОЗДІЛ 4

СХЕМНІ РІШЕННЯ ЦИФРОВИХ ЛОГІЧНИХ КОМПОНЕНТІВ НА ОСНОВІ ДИСКРЕТНОГО ПЕРЦЕПТРОНУ

Апаратна реалізація штучних нейронних мереж, що ґрунтуються на перцептронних структурах, передбачає використання спеціалізованих цифрових компонентів, що оптимізовані для виконання багатопотокових обчислень, зокрема графічні процесори (GPU), тензорні процесори (TPU) тощо, а також нейроморфні (Neuromorphic) процесори (N-M), які імітують роботу відповідних біологічних структур, такий підхід сприяє зменшенню енергоспоживання та підвищенню продуктивності. Сучасні технології, зокрема програмовані логічні інтегральні схеми (ПЛІС – FPGA / CPLD) та спеціалізовані інтегральні схеми (ASIC), дають змогу створювати високопродуктивні апаратні рішення завдяки можливості реконфігурації внутрішніх компонентів. Це зокрема, дозволяє вносити зміни в структуру мережі без потреби повторного проєктування друкованої плати (PCB). Порівняння основних платформ, на яких імплементують штучні нейронні мережі із типовими характеристиками подано в табл.4.1 [95-106, 110].

Таблиця 4.1 – Порівняння основних платформ штучних нейронних мереж

Критерій	MCU	CPU	GPU	FPGA	ASIC	N-M
Продуктивність	KOps	GOPS-TOPS	T/P FLOPS	залеж.конвеєр	T/P FLOPS	асинхрона
Затримка	~1-10 мс	>1 мс	10-100мкс	<1 мкс	10-100мкс	<100 нс
Енергоспоживання	10 ⁻³ Вт	10 ¹ Вт	10 ² Вт	10 ¹ Вт	10 ² Вт	10 ⁻³ Вт
Гнучкість	висока	висока	середня	максимум	фіксована	специфічна

Інтеграція дискретних перцептронних структур у спеціалізовані мікросхеми, зокрема Altera MAX7000 CPLD [107], дозволяє розширити функціонал та збільшити швидкість прийняття рішень нейронною мережею, реалізувати паралельне виконання операцій, а також створює нові можливості для розробки компактних цифрових систем штучного інтелекту, зокрема для платформ з обмеженими обчислювальними ресурсами.

4.1 Розробка схемних рішень компонентів бінарного двохвходового дискретного перцептрону

У запропонованій раніше структурі дискретного перцептрону, яку подано на рис. 3.8, компонент Input Transformation, що забезпечує зміщення вхідних бінарних синаптичних сигналів x_1 , x_2 ($\bar{X}$) шляхом додавання відповідних коефіцієнтів w_0 , w_1 , w_2 ($\bar{W}$), нескладно реалізувати як паралельний суматор засобами мови AlteraHDL [98, 108, 109], див. рис.4.1.

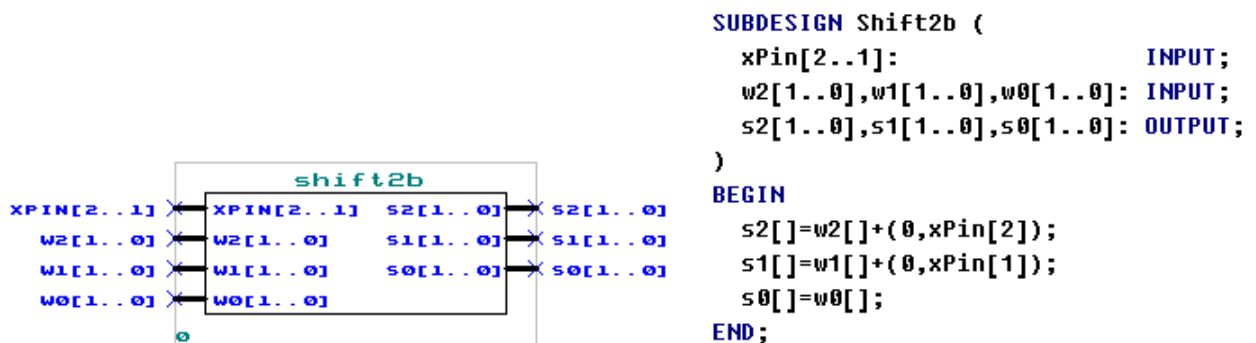


Рисунок 4.1 – Позначення та реалізація на AHDL компоненту shift2b
зміщення синаптичних сигналів на два бінарні входи

Структуру та опис апаратного компоненту shift2b, який реалізує компонент Input Transformation для двох бінарних синаптичних сигналів подано на рис.4.2, символ у схемному редакторі (ліва частина), показує зовнішній інтерфейс модуля:

- вектори-входи xPin[2..1] містять бітові значення вхідних сигналів x_2 , x_1 ;
- вектори-входи коефіцієнтів зсуву w2[1..0], w1[1..0], w0[1..0] задають наперед обрані двобітні зміщення w_2 , w_1 , w_0 ;
- вектори-виходи s2[1..0], s1[1..0], s0[1..0] формують зміщені синаптичні сигнали, які надалі подаються на компонент Aggregation.

Фрагмент коду AHDL (права частина рис.4.2), що описує логіку модуля, для кожного каналу формується двобітний вектор:

$$s2[] = w2[] + (0, xPin[2]); \quad - \text{MSB} = 0, \text{LSB} = x_2$$

$s1[] = w1[] + (0, xPin[1]);$ – MSB = 0, LSB = x_1

$s0[] = w0[];$ – постійний коефіцієнт зміщення

Операція "+" синтезується у паралельний двобітний суматор, тому зсув (додавання ваги) виконується апаратно за один такт. Однак відсутність операції додавання для ваги $w0[]$ створює апаратну несиметричність між виходами $s2[]$, $s1[]$ та виходом $s0[]$, що приводить до різних часових затримок формування згаданих вихідних сигналів. Цей факт необхідно враховувати при використанні у асинхронних системах. Таким чином, shift2b компактно реалізує зміщення вхідних бітів на основі простого додавання, мінімізуючи апаратні ресурси ПЛІС і забезпечуючи готові, вирівняні за розрядністю вектори для подальшого ентропійного агрегування [110].

Комплексну перевірку компоненту shift2b у двох площинах – функціональній та часовій подано на рис. 4.2 та 4.3.

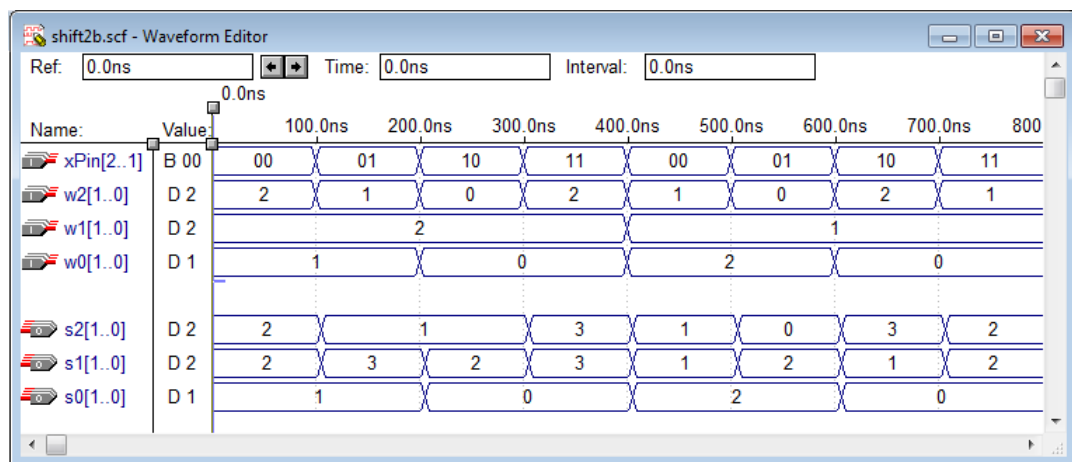


Рисунок 4.2 – Результати функціонального моделювання компоненту shift2b зміщення синаптичних сигналів для двох бінарних входів

Осцилограма функціоналу в середовищі Waveform Editor, показує реакцію виходів $s2[1..0]$, $s1[1..0]$, $s0[1..0]$ на послідовний перебір комбінацій вхідних бітів $xPin[2..1]$ ("00", "01", "10", "11") за фіксованого зміщення $w2$, $w1$, $w0$. На кожному такті можна побачити, що [110]:

- $s2$ дорівнює двобітній сумі $w_2 + x_2$;

- s1 дорівнює $w_1 + x_1$;
- s0 незмінно повторює w_0 .

Коректність логіки підтверджується тим, що для всіх 16 перевірених комбінацій входів отримані очікувані числові коди (зокрема, при "xPin=01" та "w1=2" на виході формується "s1=3"). Матриця затримок, яку подано на рис. 4.3, середовища Timing Analyzer показує для кожної пари "джерело → призначення" однорідну затримку 6 нс, що означає:

- усі шляхи від будь-якого входу (xPin[], w2[], w1[], w0[]) до будь-якого біта виходу (s2[], s1[], s0[]) мають однакову, невелику критичну затримку;
- модуль може надійно працювати з тактовою частотою близько $1/6(\text{нс}) \approx 166 \text{ МГц}$ без порушення часових обмежень [110].

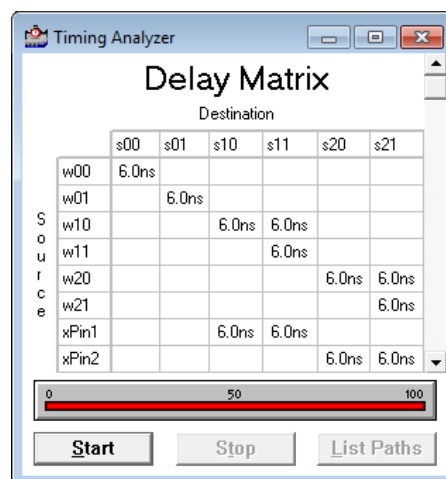


Рисунок 4.3 – Результати інструментального (часового) моделювання компоненту shift2b зміщення синаптичних сигналів на два бінарні входи

Отже, функціональне моделювання підтвердило правильність арифметичного зміщення, а інструментальний аналіз показав рівномірну та прийнятну часову поведінку, що робить shift2b ефективним і масштабованим блоком для подальшого використання у дискретному перцептроні.

На наступному етапі розробки необхідно реалізувати компонент формування результату (Output Transformation чи Activation), що представляє собою компаратор двобітових чисел s1 та s0. Фактично цей компонент буде

також використано на при реалізації алгоритму визначення кількості унікальних значень зміщених синаптичних сигналів, що формує компонент shift2b. Позначення та реалізація засобами мови AlteraHDL [98, 108, 109], подано на рис.4.4.

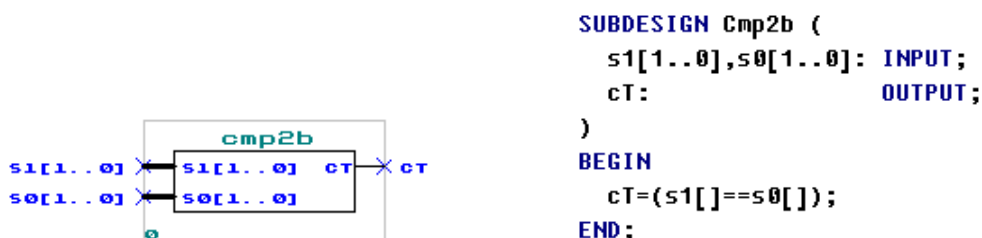


Рисунок 4.4 – Позначення та реалізація на AHDL компоненту cmp2b
(компаратора) перевірки на рівність двох двобітних чисел

Компактна апаратна реалізація компоненту cmp2b, який виконує функцію Output Transformation (Activation) для дискретного перцептрона – двобітовий компаратор, що перевіряє рівність між зміщеними синаптичними сигналами s1[1..0] та s0[1..0] подано на рис.4.4. Символ у схемному редакторі (ліва частина), що показує зовнішній інтерфейс модуля, що складається з двох 2-бітових входів s1[1..0], s0[1..0] і одного однобітового виходу cT, який стає "1", коли обидва вектори тотожно рівні, та "0" інакше. Саме цей біт інтерпретується як результат активації (y). Фрагмент коду AHDL (права частина рис.4.4), мінімалістичний опис логіки модуля [98, 108, 109, 110].

Єдина операція порівняння "==" синтезується інструментом у двобітовий елемент "XNOR + AND", що використовує лише кілька логічних елементів ПЛІС і не вимагає тактування. В наслідок цього компонент cmp2b додає незначну апаратну затримку до критичного шляху й може безпосередньо бути підключений до наступних стадій агрегування чи ансамблевої логіки кількості унікальних значень. В результаті, cmp2b можна ефективно використати в компоненті агрегації для швидкого порівнювання s1[], s0[] - вхідних зміщуваних синаптичних сигналів. Крім того cmp2b, як компонент активації, може завершувати каскад "shift2b → cmp2b", забезпечуючи швидке та

ресурсоефективне формування вихідного сигналу розробленого компоненту дискретного перцептрона [110].

Результати перевірки коректності функціонування компоненту `cmp2b` – функціональне моделювання в середовищі Waveform Editor, а також аналіз часових характеристик у Timing Analyzer подано на рис. 4.5 та 4.6.

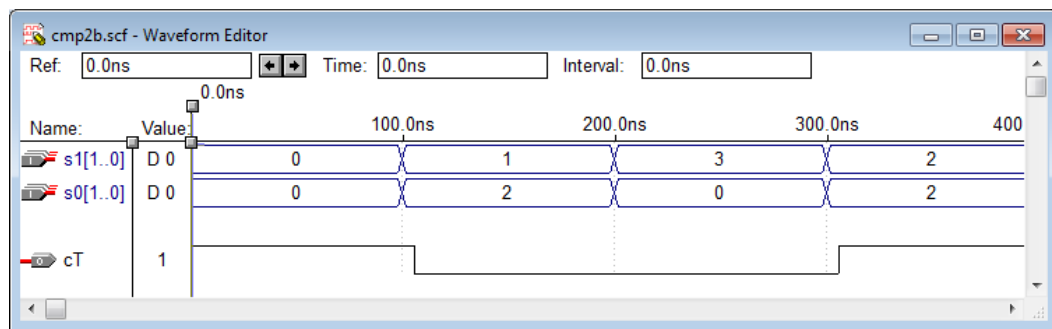


Рисунок 4.5 – Результати функціонального моделювання компоненту `cmp2b` (компаратора) перевірки на рівність двох двобітних чисел

Осцилограма функціоналу в середовищі Waveform Editor показує зміну двобітових вхідних векторів `s1[]` і `s0[]` протягом чотирьох тактових інтервалів (00 → 01 → 11 → 10). Вихід `cT` приймає логічну "1" лише тоді, коли обидва вектори співпадають ("00" на початку та "10" наприкінці), і залишається "0" для різних комбінацій ("01" та "10", "11" та "00" тощо), що підтверджує коректність логічної умови $cT = (s1[] == s0[])$. Матриця затримок – Delay Matrix, яка подана на рис.4.6, показує що для (`s00`, `s01`, `s10`, `s11`) всіх можливих бітів джерела до виходу `cT` зафіксовано однакову критичну затримку у 6 нс. Отже, компаратор не створює суттєвих апаратних затримок й забезпечує стабільну роботу на частотах до ≈ 166 МГц, що відповідає часовим параметрам блоку зміщення вхідних сигналів `shift2b` [110].

Таким чином, результати функціонального та інструментального моделювання підтверджують коректність функціоналу та високу швидкість роботи компоненту порівняння `cmp2b`, фіналізуючи логічну послідовність "зміщення → порівняння" у розробленому перцептроні.

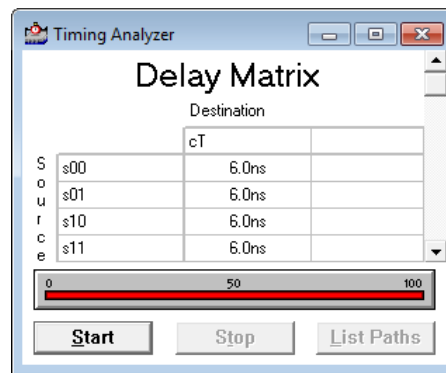


Рисунок 4.6 – Результати інструментального моделювання компоненту компаратора cmp2b перевірки на рівність двобітних чисел

Схемна реалізація цифрового компоненту агрегування (Aggregation) ґрунтується на використанні результатів порівнювання зміщених синаптичних сигналів s_2 , s_1 , s_0 , які є аргументами логічних функцій, що формують біти числа результату c_1 та c_0 . Отже, для формування бітів результату c_1 та c_0 компонент агрегування оперує лише трьома булевими ознаками – результатами попарного порівняння зміщених синаптичних сигналів, а саме: $c_{01} = (s_0 = s_1)$, $c_{02} = (s_0 = s_2)$, $c_{12} = (s_1 = s_2)$. Ці три ознаки повною мірою визначають кількість унікальних значень $U \in \{1, 2, 3\}$ серед $\{s_2, s_1, s_0\}$. Кількість унікальних станів U задається двома бітами c_1 та c_0 за правилом поданим у табл. 4.2 [110].

Таблиця 4.2 – Взаємозв'язок між кількістю унікальних станів серед s_2 , s_1 , s_0 та значеннями бітів c_1 та c_0

U (кількість унікальних станів)	Біти c_1 та c_0	Пояснення
1 (усі рівні)	01	$s_2=s_1=s_0$
2 (один відрізняється)	10	рівні лише двоє
3 (усі різні)	11	жодної рівності

Як можна побачити із табл. 4.2, c_1 приймає значення "1" якщо серед трьох порівнянь є хоча б одна нерівність, а c_0 рівне "1" у двох крайніх випадках – коли усі пари є рівні або коли жодна з пар не є рівна. Тобто, логічні функції для c_1 та c_0 можна записати так:

$$c_1 = \overline{c_{01} \wedge c_{02} \wedge c_{12}}, c_0 = (c_{01} \wedge c_{02} \wedge c_{12}) \vee \overline{c_{01}} \vee \overline{c_{02}} \vee \overline{c_{12}} \quad (4.1)$$

Таким чином, отримані логічні функції дозволяють реалізувати компактний, однотактний вузол агрегації (cnts), який кодує кількість унікальних значень серед $\{s_2, s_1, s_0\}$ як двобітовий результат: $c_1 c_0$. Позначення та логічну схему розробленого компоненту cnts, який визначає кількість унікальних значень серед $\{s_2, s_1, s_0\}$, подано на рис.4.7 [110].

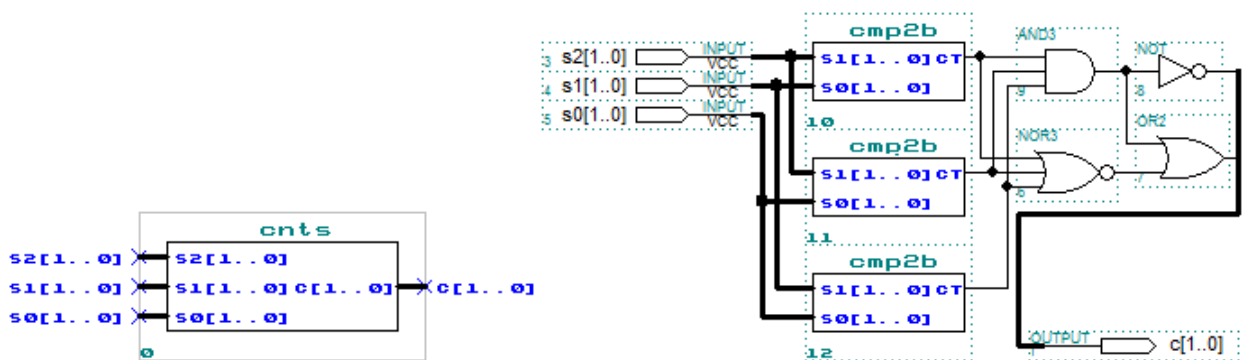


Рисунок 4.7 – Позначення та схемна реалізація компоненту cnts визначення кількості унікальних значень для трьох двобітних чисел

Компонент агрегування (cnts), що об'єднує три двобітові входи $s_2[]$, $s_1[]$, $s_0[]$ і формує результат – двобітовий код кількості унікальних значень $c[]$, подано на рис. 4.7. Ліва частина – символ у схемному редакторі, що показує зовнішній інтерфейс модуля що, складається з трьох 2-бітових шин (зміщені синаптичні сигнали) та однієї 2-бітової шини результату $c[]$, що відображає кількість унікальних станів, причому:

$c[1]$ – старший біт (сигналізує, чи є серед трійки хоча б одна нерівність);

$c[0]$ – молодший біт (розрізняє крайні випадки "усі рівні" та "усі різні").

Структурна схема логіки (права частина рис. 4.7) складається з трьох модулів cmp2b (рис. 4.4) та логічної схеми для синтезу кількості унікальних станів [110]. Модулі cmp2b попарно порівнюють:

cmp2b_0 – s_2 з $s_1 \rightarrow$ вихід c_{12} ;

$\text{cmp2b_1} - s2 \text{ з } s0 \rightarrow \text{вихід } c20;$

$\text{cmp2b_2} - s1 \text{ з } s0 \rightarrow \text{вихід } c10.$

Логічна схема на виході реалізує логічні вирази для кожного вихідного біта: $c_1 = \overline{c_{01} \wedge c_{02} \wedge c_{12}}$ та $c_0 = (c_{01} \wedge c_{02} \wedge c_{12}) \vee \overline{c_{01} \vee c_{02} \vee c_{12}}$:

- тривходовий AND3 обчислює: $c_{01} \wedge c_{02} \wedge c_{12}$;

- NOT інвертує цей сигнал для формування c_1 ;

- тривходовий NOR3 генерує $\overline{c_{01} \vee c_{02} \vee c_{12}}$;

- двовходовий OR2 об'єднує результати AND3 та NOR3, формуючи c_0 .

Завдяки використанню раніше розроблених компараторів та чотирьох базових логічних елементів модуль cnts визначає кількість унікальних значень за один такт, забезпечуючи узгоджену затримку з попередніми компонентами "shift2b $\rightarrow$ cmp2b" і готовий двобітовий код для подальшого вирішення задачі перцептронної класифікації [110].

Перевірку компоненту cnts, який реалізує визначення кількості унікальних значень для трьох вхідних двобітових сигналів подано на рис. 4.8 та 4.9.

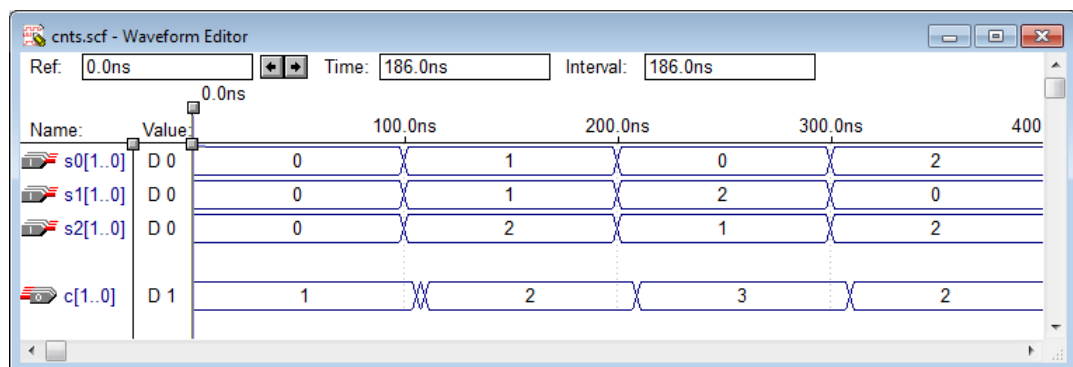


Рисунок 4.8 – Результати функціонального моделювання компоненту cnts визначення кількості унікальних значень для трьох двобітних чисел

Осцилограма функціонального моделювання (ліва частина) де вхідні вектори $s0[]$, $s1[]$, $s2[]$ перебирають комбінації, у яких:

на 0 нс усі вхідні сигнали рівні ("00 00 00") $\Rightarrow c[] = 01$ (код "1 унікальне");

на ≈ 100 нс лише $s2$ відрізняється ("01 01 10") $\Rightarrow c[] = 10$ (код "2 унікальні");

на ≈ 200 нс усі різні ("00 10 01") $\Rightarrow c[] = 11$ (код "3 унікальні");

на ≈ 300 нс лише s1 відрізняється ("10 00 10") $\Rightarrow c[] = 10$, повернення до випадку "2 унікальні". Отже, вихідний код c[] точно відповідає очікуваній кількості унікальних значень у кожному такті. Матриця затримок (права частина рис.4.9) показує [110]:

- для старшого біта c1 (формується інверсією AND3) критична затримка становить 6 нс для всіх маршрутів, що збігається з попередніми блоками shift2b та cmp2b.
- молодший біт c0 (складніша логіка AND3 v NOR3), має дещо більшу затримку ≈ 9.5 нс.

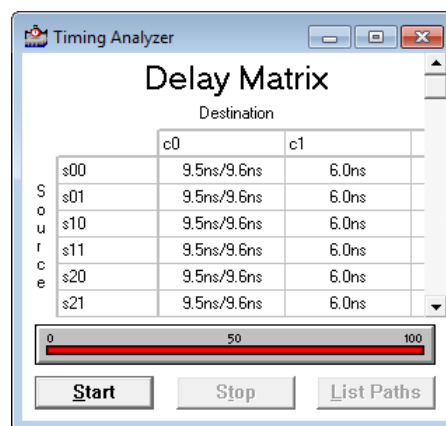


Рисунок 4.9 – Результати інструментального (часового) моделювання цифрового компоненту cnts для трьох двобітних чисел

Отримані результати дозволяють забезпечити роботу модуля cnts на частотах до ≈ 105 МГц (обмежені шляхами до c0), що залишається узгодженим із раніше синтезованими компонентами каскаду.

Таким чином, результати функціонального та часового моделювання підтвердили коректність кодування кількості унікальних значень, а також прийнятні швидкісні характеристики розробленого цифрового компоненту агрегування cnts.

4.2 Реалізація бінарного двохвходового дискретного перцептрону для емуляції цифрових логічних компонентів

На основі раніше реалізованих та досліджених цифрових компонентів shift2b (рис. 4.1), cmp2b (рис. 4.4) та cnts (рис. 4.7) здійснено компонування дискретного перцептрону на два бінарні входи. Схемна реалізація, як інтеграція раніше спроектованих цифрових компонентів у єдиний модуль perceptron, подана на рис.4.10. Схемне позначення модуля (ліва частина) показує мінімалістичний зовнішній інтерфейс: [110]

- $xb[]$ – вхідний вектор початкових бітів $xb2$, $xb1$;
- $w2[]$, $w1[]$, $w0[]$ – двобітні коефіцієнти зміщення;
- $aN[]$ – еталон-поріг (аналог A) для остаточного порівняння;
- y – однобітовий вихід перцептрона.

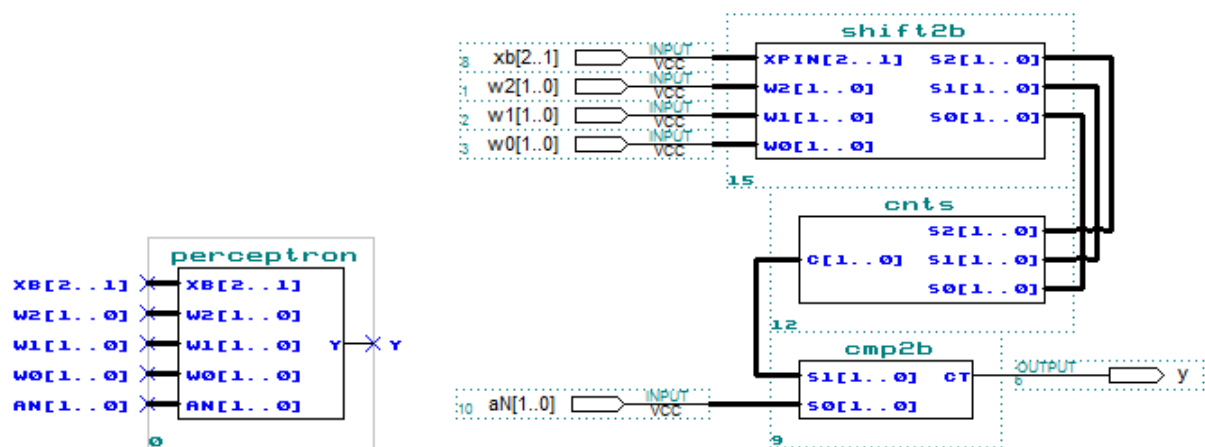


Рисунок 4.10 – Позначення та схемна реалізація дискретного перцептрону на два бінарні входи для емуляції логічних функцій

Структурна схема логіки (права частина рис.4.10) показує, як дані послідовно проходять крізь внутрішні компоненти: [110]

- shift2b реалізує операцію паралельного додавання ($x_i + w_i$), що в результаті формує зміщені сигнали $s2[]$, $s1[]$, $s0[]$;
- cnts реалізує спрощений ентропійний критерій агрегації, а саме визначає кількість унікальних значень серед $s2[]$, $s1[]$, $s0[]$;

- `cmp2b` перевіряє на рівність отриманий код із порогом `aN[]`, а результат порівняння подається на вихід перцептрона (`y`).

Завдяки такій каскадній організації модуль `perceptron` зберігає одnobітну комбінаційну затримку, успадковану від найповільнішого шляху (`shift2b` → `cnts` → `cmp2b`), а також дозволяє гнучко налаштовувати логічну функцію за допомогою зміни коефіцієнтів: `w2[]`, `w1[]`, `w0[]`, а також порогу `aN[]`, без потреби перепрошивки коду чи структурних змін схеми [110].

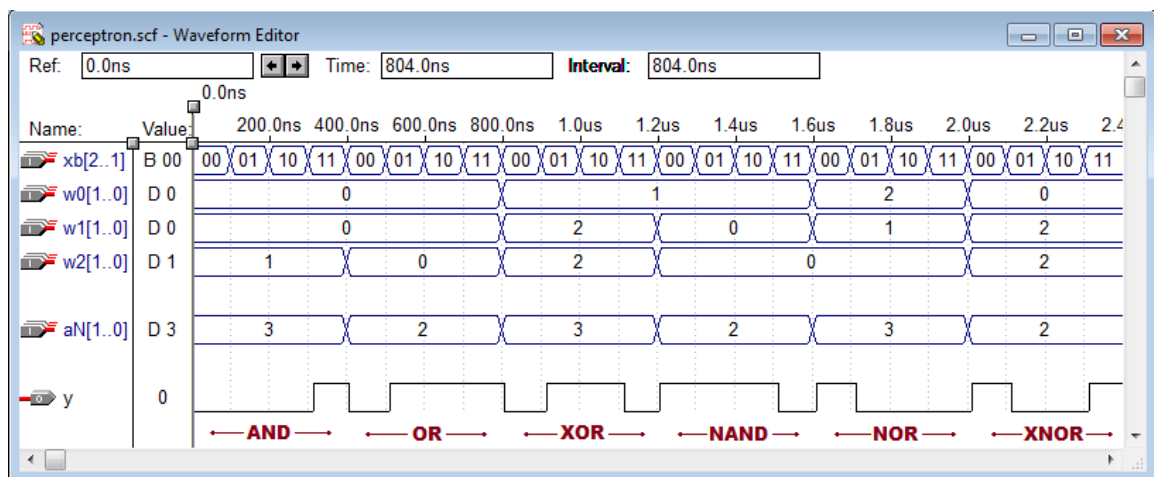


Рисунок 4.11 – Результати функціонального моделювання логічних функцій `or`, `and`, `xor`, а також `nor`, `nand`, `xnor` дискретного перцептрону на два бінарні входи

Результати функціонального та інструментального моделювання, які подано на рис.4.11, фактично підтверджують, що інтегрований цифровий модуль `perceptron` коректно емулює шість базових логічних операцій, а також демонструє його часові обмеження на кристалі Altera MAX7000. Осцилограма функціоналу (ліва частина) показує сигнали на входах `xb[]`, що змінюються по циклу "00, 01, 10, 11" кожні 200 нс. Коефіцієнти зміщення: `w0[]`, `w1[]`, `w2[]` і поріг `aN[]` залишаються постійними на інтервалі моделювання відповідної логічної функції переформовуються через 800нс, задаючи послідовно пари "параметри → модельована функція", відповідно підписані внизу діаграми [110]:

- $w0[] = w1[] = 0, w2[] = 1, aN[] = 3 \rightarrow \text{OR}$ ($y = 0$ тільки при $xb[] = 00$).
- $w0[] = w1[] = w2[] = 0, aN[] = 2 \rightarrow \text{AND}$ ($y = 1$ тільки при $xb[] = 11$).
- $w0[] = 1, w1[] = w2[] = 2, aN[] = 3 \rightarrow \text{XOR}$.
- $w0[] = 1, w1[] = w2[] = 0, aN[] = 2 \rightarrow \text{NOR}$.
- $w0[] = 2, w1[] = 1, w2[] = 0, aN[] = 3 \rightarrow \text{NAND}$.
- $w0[] = 0, w1[] = w2[] = 2, aN[] = 2 \rightarrow \text{XNOR}$.

На кожному інтервалі вихід (y) відповідно реалізує ту таблицю істинності, що очікується для відповідної логічної функції, підтверджуючи функціональну гнучкість налаштування перцептрона, що потребує тільки зміни коефіцієнтів і порогу активації без перепроєктування схеми [110].

Матриця затримок (див. рис. 4.12) відображає часові характеристики різних логічних шляхів у структурі розробленого дискретного перцептрону на два бінарні входи, що дозволяє побачити критичні ділянки схеми, які визначають максимальну затримку сигналу і, відповідно, верхню межу тактової частоти пристрою.

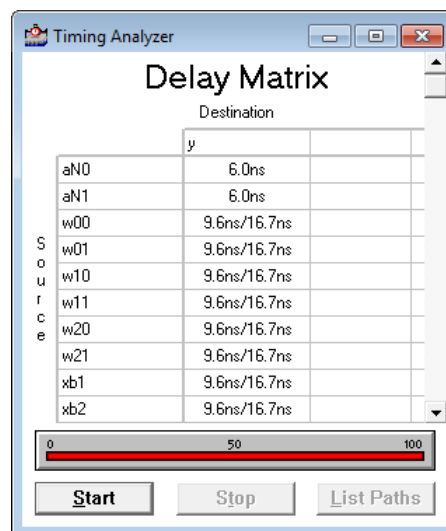


Рисунок 4.12 – Результати інструментального (часового) моделювання дискретного перцептрону на два бінарні входи при емуляції логічних функцій or, and, xor та nor, nand, xnor

Зокрема, короткий шлях (містить мінімальну кількість логічних елементів) – від еталонних бітів порогових значень $aN0/aN1$ до вихідного сигналу – має критичну затримку 6 нс, і реалізує лише просту логічну операцію – або логічне заперечення (NOT), або комбінацію AND з наступним NOT (інвертовану кон'юнкцію).

Шляхи від коефіцієнтів зміщення: $w0[]$, $w1[]$, $w2[]$ та інформаційних входів $xb[]$ є суттєво довшими, тобто затримка стає дещо більшою і становить: $\min$ – 9.6 нс, $\max$ – 16.7 нс.

Найбільшу затримку спричиняє проходження сигналу через шлях ($\text{shift2b} \rightarrow \text{cnts} \rightarrow \text{cmp2b}$), що складає 16.7 нс, який включає послідовне опрацювання бітових зсувів shift2b , підрахунок кількості унікальних cnts , та порівняння з порогом cmp2b (двобітовий компаратор). Відповідно, ця затримка є критичною для всієї системи, оскільки визначає, з якою максимальною тактовою частотою може працювати схема в повністю комбінаційній реалізації (тобто без внутрішніх регістрів або конвеєрних стадій).

З огляду на інверсну залежність між затримкою та частотою:

$$f_{\max} = \frac{1}{t_{\max}} = \frac{1}{16,7 \times 10^{-9}} \approx 60(\text{МГц}) \quad (4.2)$$

Таким чином, найбільша затримка 16.7 нс задає верхню оцінку робочої тактової частоти $\approx 60\text{МГц}$ для повністю комбінаційної реалізації [110].

За потреби вищих швидкостей блок легко конвеєрується, оскільки між компонентами існують чітко визначені шини даних.

Таким чином, функціональне моделювання доводить коректність емуляції будь-яких бінарних логічних функцій через параметризацію ваг та порога, а інструментальний аналіз підтверджує, що результуючий дискретний перцептрон відповідає вимогам швидкодії й може бути безпосередньо інтегрований у більші цифрові системи чи нейромережеві прискорювачі на ПЛІС.

ВИСНОВКИ ДО ЧЕТВЕРТОГО РОЗДІЛУ

1. Розроблено цифровий компонент (cnts), що реалізує визначення кількості унікальних значень серед зміщених за допомогою двобітового суматора (shift2b) синаптичних сигналів, структура якого ґрунтується на трьох компараторах (cmp2b) та базовій логіці, а результат формується як двобітний код, затримка операції не перевищує 9.6 нс, що дозволяє працювати на частотах до 105 МГц.
2. Розроблено апаратну структуру дискретного перцептрону на два бінарні входи, що включає окремі цифрові компоненти опрацювання вхідних сигналів (shift2b), порівняння проміжних результатів (cmp2b) та агрегування інформації (cnts). Компоненти реалізовано з урахуванням мінімізації апаратних витрат і забезпечення високої швидкодії.
3. Побудовано інтегрований модуль дискретного перцептрону, який реалізує логіку класифікації за ентропійною ознакою, що забезпечує емуляцію базових логічних функцій: OR, AND, XOR, NOR, NAND, XNOR. Вихідні сигнали у всіх випадках відповідали таблицям істинності, що підтверджує універсальність розробленої схеми та її придатність для реалізації цифрових логічних схем.
4. Розроблено модульну апаратну структуру, що підтримує імплементацію конвеєризації для підвищення тактової частоти системи. Завдяки чітко визначеним межах між блоками можливе масштабування функціоналу без зміни логічних зв'язків, що забезпечує адаптивність для складніших конфігурацій на ПЛІС.

ЗАГАЛЬНІ ВИСНОВКИ ДО РОБОТИ

На основі проведених теоретичних та експериментальних досліджень при вирішенні актуального науково-прикладного завдання у галузі інформаційних технологій – розробка методу реалізації дискретного перцептронну на основі імовірнісних оцінок зміщених синаптичних сигналів, який дозволяє забезпечити опрацювання лінійно-нероздільних множин, зниження алгоритмічної та апаратної складності, а також придатний до використання в комп'ютерних системах з обмеженими обчислювальними ресурсами:

1. Проаналізовано сучасний стан розвитку та впровадження перцептронних структур у комп'ютерних системах за яким показано, що класичні перцептрони реалізуються через операції згортки чи кореляції, які передбачають формування зваженої суми вхідних сигналів із подальшою активацією за допомогою порогової функції, в наслідок чого мають обмежену виразну здатність та не можуть ефективно розпізнавати лінійно нероздільні шаблони.
2. Запропоновано вдосконалення реалізації класичного перцептронну при якому замість зваженого сумування використовується оцінка інформаційної ентропії, що дозволяє емулювати повний набір логічних функцій в межах однієї перцептронної структури та означено перспективи подальшого розвитку перцептронних структур.
3. Вперше запропоновано реалізацію дискретного перцептронну на основі імовірнісних оцінок зміщених синаптичних сигналів, що дозволяє перейти до дискретного базису, забезпечити опрацювання лінійно-нероздільних множин, уникнути використання операцій множення та спростити апаратну реалізацію. Розроблена структура визначає рівень активації перцептронну за кількістю унікальних станів зміщених синаптичних сигналів, що реалізує ентропійну оцінку за спрощеною формулою Хартлі без використання логарифмічної функції.

4. Досліджено можливість опрацювання лінійно-нероздільних множин на основі емуляції логічних функцій AND, OR, XOR, NAND, NOR, XNOR запропонованим дискретним перцептроном, що підтверджує розширення функціоналу порівняно з реалізацією перцептрону на основі кореляції.
5. Запропоновано метод обмеження множини рішень для дискретного перцептрона шляхом аналізу матриці кількості унікальних станів для 2-ох бінарних входів при фіксованому порозі активації, що зменшує простір пошуку до 35% від повного, а також досліджено вплив вибору функції активації як операції порівнювання, що у поєднанні з відповідними коефіцієнтами зміщення дозволяє адаптувати перцептрон до різних типів логічних функцій та їх типових комбінацій.
6. Розроблено модель дискретного перцептрона на основі багат шарової класичної нейронної мережі, що дозволяє застосувати типові методи машинного навчання нейромереж для визначення коефіцієнтів зміщення для запропонованого дискретного перцептрона.
7. Розроблено модифіковану порогову функцію, яка коректно працює як з додатними, так і з від'ємними порогами, що дає змогу реалізовувати як нижню, так і верхню межу активації на основі кількості унікальних значень зміщених вхідних сигналів.
8. Встановлено можливість використання дискретного перцептрона як поліфункціонального логічного елементу, здатного реалізовувати широкий спектр логічних операцій, зокрема за рахунок вибору відповідної активаційної функції та коефіцієнтів зміщення.
9. Розроблено каскадно-умовний та умовно-логічний алгоритми компоненту визначення кількості унікальних станів і на основі порівняльного аналізу встановлено, що реалізації на основі логічних функцій, замість каскадних умов, мають нижчу часову складність, дозволяють покращити ефективність схемної/програмної реалізації та масштабованість на більшу кількість входів.

10. Отримано логічні вирази для визначення кількості унікальних станів для 3-ох та 4-ох входових перцептронів, що дозволяє мінімізувати кількість умовних конструкцій при алгоритмічній та апаратній реалізаціях дискретних компонентів такого типу.
11. Розроблено апаратну структуру дискретного перцептрону на два бінарні входи, що включає окремі цифрові компоненти опрацювання вхідних сигналів, порівняння проміжних результатів та агрегування інформації, який реалізує логіку класифікації за ентропійною ознакою і забезпечує емуляцію базових логічних функцій: OR, AND, XOR, NOR, NAND, XNOR, затримка операції не перевищує 9.6 нс, що дозволяє працювати на частотах до 105 МГц.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Multilayer perceptrons. Department of Computer Science | Aalto University. URL: <https://users.ics.aalto.fi/ahonkela/dippa/node41.html>.
2. Multilayer perceptron and neural networks / M.-C. POPESCU et al. WSEAS Transactions on Circuits and Systems. 2009. Vol. 8, no. 7. P. 579–588. https://www.researchgate.net/publication/228340819_Multilayer_perceptron_and_neural_networks.
3. The fundamentals of artificial neural networks. Mathematics of Deep Learning. 2023. P. 6–18. URL: <https://doi.org/10.1515/9783111025551-004>.
4. Ghayoumi M. Artificial Neural Networks (ANNs) Fundamentals and Architectures. Deep Learning in Practice. Boca Raton, 2021. P. 51–75. URL: <https://doi.org/10.1201/9781003025818-4>.
5. Russell J. Artificial Intelligence and Deep Learning Essentials: Fundamentals of Deep Learning and Neural Networks. Independently Published, 2018.
6. Murtagh F. Multilayer perceptrons for classification and regression. Neurocomputing. 1991. Vol. 2, no. 5-6. P. 183–197. URL: [https://doi.org/10.1016/0925-2312\(91\)90023-5](https://doi.org/10.1016/0925-2312(91)90023-5).
7. Neural Networks: From the Perceptron to Deep Nets / M. Gabrié et al. Spin Glass Theory and Far Beyond. 2023. P. 477–497. URL: https://doi.org/10.1142/9789811273926_0024.
8. Principles of Deep Learning Theory. Cambridge University Press, 2022.
9. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions / L. Alzubaidi et al. Journal of Big Data. 2021. Vol. 8, no. 1. URL: <https://doi.org/10.1186/s40537-021-00444-8>.
10. State-of-the-art in artificial neural network applications: A survey / O. I. Abiodun et al. Heliyon. 2018. Vol. 4, no. 11. P. e00938. URL: <https://doi.org/10.1016/j.heliyon.2018.e00938>.

11. Modern Measurements: Fundamentals and Applications / A. Ferrero et al. Wiley & Sons, Incorporated, John, 2015. 400 p.
12. Jiang J., Tan L. Digital Signal Processing: Fundamentals and Applications. Elsevier Science & Technology Books, 2018. 920 p.
13. Polyanskiy Y. Information Theory: From Coding to Learning. Cambridge University Press, 2024.
14. Bengio Y., Courville A., Goodfellow I. Deep learning. MIT Press, 2016. 800 p.
15. Venkatesan R., Li B. Convolutional Neural Networks in Visual Computing. Boca Raton; London: Taylor & Francis, CRC Press, 2017: CRC Press, 2017. URL: <https://doi.org/10.4324/9781315154282>.
16. Oppenheim A. V. Discrete-Time Signal Processing. Pearson Education, Limited, 2010. 1137 p.
17. Morozov A. V., Levkivskyi V. L., Plechystyy D. D. ACTIVATION FUNCTIONS IN NEURAL NETWORKS: OVERVIEW AND COMPARISON. Scientific notes of Taurida National V.I. Vernadsky University. Series: Technical Sciences. 2024. Vol. 1, no. 3. P. 144–151. URL: <https://doi.org/10.32782/2663-5941/2024.3.1/22>.
18. Researcher. ACTIVATION FUNCTIONS IN NEURAL NETWORKS: A COMPREHENSIVE OVERVIEW. International Journal of Research In Computer Applications and Information Technology (IJRCAIT). 2024. Vol. 7, no. 2. P. 214–227. URL: <https://doi.org/10.5281/zenodo.13950374>.
19. Lederer J. Activation functions in artificial neural networks: a systematic overview. 2021. 42 p. (Preprint. Ruhr-University Bochum). URL: <https://arxiv.org/pdf/2101.09957.pdf>.
20. Яковин С. В., Мельничук С. І. Особливості реалізації та функціональні обмеження перцептронних структур. Проблемно-наукова міжгалузева конференція “Інформаційні проблеми комп’ютерних систем, юриспруденції, енергетики, моделювання та управління”. 2023. С. 184–187.

21. Piazza F., Uncini A., Zenobi M. Neural network complexity reduction using adaptive polynomial activation functions. *Icann '93*. London, 1993. P. 452–455. URL: https://doi.org/10.1007/978-1-4471-2063-6_108.
22. Hu Y. Fuzzy integral-based perceptron for two-class pattern classification problems. *Information sciences*. 2007. Vol. 177. P. 1673–1686. URL: <https://doi.org/10.1016/j.ins.2006.09.009>.
23. Zhong Y., Cang W. An Integrated Perceptron Network and Learning Algorithm for Multi-Class Patterns Recognition. *Transactions of Beijing institute of Technology*. 2001. Vol. 4. P. 459–463. URL: <https://journal.bit.edu.cn/zr/en/article/id/200104127>.
24. Hu Y.-C. Pattern classification by multi-layer perceptron using fuzzy integral-based activation function. *Applied Soft Computing*. 2010. Vol. 10, no. 3. P. 813–819. URL: <https://doi.org/10.1016/j.asoc.2009.09.011>.
25. Singh K., Sahu C., Singh J. Linearly separable pattern classification using memristive crossbar circuits. *2014 15th International Symposium on Quality Electronic Design (ISQED)*, Santa Clara, CA, USA, 3–5 March 2014. 2014. URL: <https://doi.org/10.1109/isqed.2014.6783343>.
26. Hu Y.-C., Tseng F.-M. Functional-link net with fuzzy integral for bankruptcy prediction. *Neurocomputing*. 2007. Vol. 70, no. 16-18. P. 2959–2968. URL: <https://doi.org/10.1016/j.neucom.2006.10.111>.
27. Multilayer Perceptron: Architecture Optimization and Training / H. Ramchoun et al. *International Journal of Interactive Multimedia and Artificial Intelligence*. 2016. Vol. 4, no. 1. P. 26. URL: <https://doi.org/10.9781/ijimai.2016.415>.
28. Multilayer Perceptron New Method for Selecting the Architecture Based on the Choice of Different Activation Functions / H. Ramchoun et al. *International Journal of Information Systems in the Service Sector*. 2019. Vol. 11, no. 4. P. 21–34. URL: <https://doi.org/10.4018/ijjiss.2019100102>.
29. Melnychuk S., Kuz M., Yakovyn S. Emulation of logical functions NOT, AND, OR, and XOR with a perceptron implemented using an information

entropy function. Proceedings of 14th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET), Lviv-Slavske, 20–24 February 2018. URL: <https://doi.org/10.1109/tcset.2018.8336337>.

30. Jain A. K., Duin P. W., Jianchang Mao. Statistical pattern recognition: a review. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2000. Vol. 22, no. 1. P. 4–37. URL: <https://doi.org/10.1109/34.824819>.

31. Osowski S., Nghia D. D. Neural networks for classification of 2-D patterns. 5th International Conference on Signal Processing, Beijing, China. URL: <https://doi.org/10.1109/icosp.2000.893399>.

32. El-Sebakhy E. A. Functional networks training algorithm for statistical pattern recognition. Proceedings. ISCC 2004. Ninth International Symposium on Computers And Communications (IEEE Cat. No.04TH8769), Alexandria, Egypt. URL: <https://doi.org/10.1109/iscc.2004.1358387>.

33. Murino V. Structured neural networks for pattern recognition. IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics). 1998. Vol. 28, no. 4. P. 553–561. URL: <https://doi.org/10.1109/3477.704294>.

34. Kittler J. Statistical pattern recognition in image analysis. Journal of Applied Statistics. 1994. Vol. 21, no. 1-2. P. 61–75. URL: <https://doi.org/10.1080/757582968>.

35. Яковин С. В., Мельничук С. І. Імплементация булевих функцій в перцептронних структурах. Вісник Вінницького політехнічного інституту. 2024. Т. 176, № 5. С. 48–55. URL: <https://doi.org/10.31649/1997-9266-2024-176-5-48-55>.

36. Approximation to Boolean functions by neural networks with applications to thinning algorithms / Xiong Shenshu et al. 17th IEEE Instrumentation and Measurement Technology Conference, Baltimore, MD, USA. URL: <https://doi.org/10.1109/imtc.2000.848892>.

37. Liou D.-R., Liou J.-W., Liou C.-Y. Learning behaviors of perceptron. iConcept Press, 2013.

38. Morozov A. V., Levkivskiy V. L., Plechystyy D. D. ACTIVATION FUNCTIONS IN NEURAL NETWORKS: OVERVIEW AND COMPARISON. Scientific notes of Taurida National V.I. Vernadsky University. Series: Technical Sciences. 2024. Vol. 1, no. 3. P. 144–151. URL: <https://doi.org/10.32782/2663-5941/2024.3.1/22>.
39. Bishop C. M., Bishop H. Deep Learning. Cham: Springer International Publishing, 2024. URL: <https://doi.org/10.1007/978-3-031-45468-4>.
40. Bishop C. M. Neural networks for pattern recognition. Oxford University Press, 1996. 504 p.
41. Suh C. Information Theory for Data Science. Boston-Delft: Now Publishers, 2023. URL: <https://doi.org/10.1561/9781638281153>.
42. Information Theory for Complex Systems: An Information Perspective on Complexity in Dynamical Systems and Statistical Mechanics. Springer, 2024.
43. Cover T. M. Elements of Information Theory. Hoboken, USA: John Wiley & Sons, Inc., 2006.
44. Yeung R. W. A First Course in Information Theory. Boston, MA: Springer US, 2002. URL: <https://doi.org/10.1007/978-1-4419-8608-5>.
45. Lindgren K. Information Theory for Complex Systems. Berlin, Heidelberg: Springer Berlin Heidelberg, 2024. URL: <https://doi.org/10.1007/978-3-662-68214-2>.
46. MACKAY D. J. C. Information Theory, Inference & Learning Algorithms. Cambridge, UK: Cambridge University Press, 2003. 640 p.
47. Longo G. Quantitative – Qualitative Measure of Information. Vienna: Springer Vienna, 1972. URL: <https://doi.org/10.1007/978-3-7091-2796-4>.
48. Мельничук С. І., Яковин С. В. Ідентифікація об'єктів (сигналів) за оцінками інформаційної ентропії їх двовірних монохромних зображень. Науковий вісник Національного гірничого університету. 2015. № 3 (147). С. 131–137. ISSN: 22232362 20712227.

49. Manookin M. B., Rieke F. Two Sides of the Same Coin: Efficient and Predictive Neural Coding. *Annual Review of Vision Science*. 2023. Vol. 9, no. 1. URL: <https://doi.org/10.1146/annurev-vision-112122-020941>.
50. MacKay D. J. *Information Theory, Inference, and Learning Algorithms*. 4th ed. Cambridge University Press, 2004. 640 p. URL: <http://www.cambridge.org/0521642981>.
51. Yakovyn S. V., Melnychuk S. I. Discrete perceptron based on probabilistic estimates of shifted synaptic signals. *Naukovyi visnyk natsionalnoho hirnychoho universytetu*. 2025. No. 2. P. 189–196. URL: <https://doi.org/10.33271/nvngu/2025-2/189>.
52. Спосіб реалізації перцептрона на основі імовірнісних характеристик зміщених синаптичних сигналів: пат. 126753 Україна: G06N 20/00. С. І. Мельничук, С. В. Яковин. – № а2019 01301; заявл. 08.02.2019; опубл. 25.01.2023, Бюл.№4. 3 с. <https://sis.nipo.gov.ua/uk/search/detail/1719578/>
53. Simulation of Learning Logical Functions Using Single-Layer Perceptron / M. V. Ahamad et al. *Advances in Intelligent Systems and Computing*. Singapore, 2020. P. 121–133. URL: https://doi.org/10.1007/978-981-15-1518-7_10.
54. Neirotti J. P. Parallel strategy for optimal learning in perceptrons. *Journal of Physics A: Mathematical and Theoretical*. 2010. Vol. 43, no. 12. P. 125101. URL: <https://doi.org/10.1088/1751-8113/43/12/125101>.
55. Dwivedi R., Srivastava V. K. Fundamental optimization methods for machine learning. *Statistical Modeling in Machine Learning*. 2023. P. 227–247. URL: <https://doi.org/10.1016/b978-0-323-91776-6.00005-1>.
56. Chelouah R., Siarry P. *Optimization and Machine Learning: Optimization for Machine Learning and Machine Learning for Optimization*. Wiley & Sons, Incorporated, John, 2022.
57. Bandura A., Skaskiv O. Analog of Hayman's Theorem and its Application to Some System of Linear Partial Differential Equations. *Zurnal*

matematicheskoy fiziki, analiza, geometrii. 2016. Vol. 15, no. 2. P. 170–191. URL: <https://doi.org/10.15407/mag15.02.170>.

58. Surrogate modeling in design optimization of structures with discontinuous responses / C. Boursier Niutta et al. Structural and Multidisciplinary Optimization. 2018. Vol. 57, no. 5. P. 1857–1869. URL: <https://doi.org/10.1007/s00158-018-1958-7>.

59. Jagtap A. D., Kawaguchi K., Karniadakis G. E. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. Journal of Computational Physics. 2020. Vol. 404. P. 109136. URL: <https://doi.org/10.1016/j.jcp.2019.109136>.

60. Haykin S. Neural networks and learning machines. Pearson Education, Limited, 2009. 937 p.

61. Bishop C. M. Pattern recognition and machine learning (information science and statistics). New York: Springer, 2006. 778 p.

62. Howard W. R. Pattern recognition and machine learning. Kybernetes. 2007. Vol. 36, no. 2. P. 275. URL: <https://doi.org/10.1108/03684920710743466>.

63. Rosen-Zvi M., Kanter I. Training a perceptron in a discrete weight space. Physical review E. 2001. Vol. 64, no. 4. URL: <https://doi.org/10.1103/physreve.64.046109>.

64. Яковин С., Мельничук С. Обмеження множини можливих рішень для дискретного перцептрона зі зміщеними синаптичними сигналами при емуляції бінарних логічних функцій. Вісник Хмельницького національного університету. Серія: Технічні науки. 2025. Т. 351, № 3.1. С. 295–301. <https://doi.org/10.31891/2307-5732-2025-351-37>.

65. Hybrid Genetic Algorithm and Simulated Annealing for Function Optimization / T. N. Fatyanosa et al. Journal of Information Technology and Computer Science. 2017. Vol. 1, no. 2. P. 82. URL: <https://doi.org/10.25126/jitecs.20161215>.

66. Maucher M., Schöning U., Kestler H. A. Search heuristics and the influence of non-perfect randomness: examining Genetic Algorithms and

Simulated Annealing. Computational Statistics. 2011. Vol. 26, no. 2. P. 303–319. URL: <https://doi.org/10.1007/s00180-011-0237-5>.

67. Multitasking Associative Networks / E. Agliari et al. Physical Review Letters. 2012. Vol. 109, no. 26. URL: <https://doi.org/10.1103/physrevlett.109.268101>.

68. Aggarwal C. C. Neural Networks and Deep Learning. Cham: Springer International Publishing, 2023. URL: <https://doi.org/10.1007/978-3-031-29642-0>.

69. Schmidhuber J. Deep learning in neural networks: An overview. Neural Networks. 2015. Vol. 61. P. 85–117. URL: <https://doi.org/10.1016/j.neunet.2014.09.003>.

70. Fundamentals of Higher Order Neural Networks for Modeling and Simulation / M. M. Gupta et al. Artificial Higher Order Neural Networks for Modeling and Simulation. 2013. P. 103–133. URL: <https://doi.org/10.4018/978-1-4666-2175-6.ch006>.

71. Stochastic implementation of the activation function for artificial neural networks / I. Yeo et al. 2016 IEEE Biomedical Circuits and Systems Conference (BioCAS), Shanghai, China, 17–19 October 2016. 2016. URL: <https://doi.org/10.1109/biocas.2016.7833826>.

72. Zhang A., Lipton Z. C. Dive into Deep Learning. Cambridge University Press, 2023.

73. Kumar N., Dixit A., Vijay V. Entropy measures and their applications: A comprehensive review. 2025. URL: <https://doi.org/10.48550/arXiv.2503.15660>.

74. Han X., Kloeden P. E. Sigmoidal approximations of Heaviside functions in neural lattice models. Journal of Differential Equations. 2020. Vol. 268, no. 9. P. 5283–5300. URL: <https://doi.org/10.1016/j.jde.2019.11.010>.

75. Wang X., Yang M., E. Kloeden P. Sigmoidal approximations of a delay neural lattice model with Heaviside functions. Communications on Pure & Applied Analysis. 2020. Vol. 19, no. 4. P. 2385–2402. URL: <https://doi.org/10.3934/cpaa.2020104>.

76. Supplier evaluation model based on entropy-TOPSIS and 0-1 programming algorithm / D. Xing et al. International Conference on Computer, Artificial Intelligence, and Control Engineering (CAICE 2023), Hangzhou, China, 17–19 February 2023 / ed. by A. Bhattacharjya, X. Feng. 2023. URL: <https://doi.org/10.1117/12.2680793>.
77. An Entropy evaluation method of hierarchical clustering / Q. Tu et al. 2020 19th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES), Xuzhou, 16–19 October 2020. 2020. URL: <https://doi.org/10.1109/dcabes50732.2020.00066>.
78. Яковин С., Мельничук С. Апроксимація індикаторних функцій у контексті ймовірнісного дискретного перцептрона. Вісник Хмельницького національного університету. Серія: Технічні науки. 2025. Т. 353, № 3.2. С. 357–364. <https://doi.org/10.31891//2307-5732-2025-353-50>
79. A Theoretical and Empirical Comparison of Gradient Approximations in Derivative-Free Optimization / A. S. Berahas et al. Foundations of Computational Mathematics. 2021. URL: <https://doi.org/10.1007/s10208-021-09513-z>.
80. Chaudhuri S., Solar-Lezama A. Smooth interpretation. the 2010 ACM SIGPLAN conference, Toronto, Ontario, Canada, 5–10 June 2010. New York, New York, USA, 2010. URL: <https://doi.org/10.1145/1806596.1806629>.
81. Murphy K. P. Probabilistic Machine Learning: An Introduction. MIT Press, 2022. 848 p.
82. Rasmussen C. E., Williams C. K. I. Gaussian Processes for Machine Learning. The MIT Press, 2005. URL: <https://doi.org/10.7551/mitpress/3206.001.0001>.
83. Ben-David S., Shalev-Shwartz S. Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press, 2014.
84. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning. New York, NY: Springer New York, 2009. URL: <https://doi.org/10.1007/978-0-387-84858-7>.

85. Zhang Y., Chen D., Ye C. Deep Neural Networks. Taylor & Francis Group, 2020.
86. Grabe M. Generalized Gaussian Error Calculus. Berlin, Heidelberg : Springer Berlin Heidelberg, 2010. URL: <https://doi.org/10.1007/978-3-642-03305-6>.
87. Hu Y. Fuzzy integral-based perceptron for two-class pattern classification problems. Information sciences. 2007. Vol. 177. P. 1673–1686. URL: <https://doi.org/10.1016/j.ins.2006.09.009>.
88. SIT: Stochastic input transformation to defend against adversarial attacks on deep neural networks / A. Guesmi et al. IEEE Design & Test. 2022. Vol. 39. P. 63–72. URL: <https://doi.org/10.1109/mdat.2021.3077542>.
89. Яковин С. В., Мельничук С. І., Мануляк І. З. Алгоритмічні реалізації дискретного перцептрону зі зміщеними синаптичними сигналами. Проблемно-наукова міжнародна конференція "Інформаційні проблеми комп'ютерних систем, енергетики, моделювання та управління". (ICSM-2025): Матеріали проблем.-наук. міжнар. конф., м. Надвірна, 5–6 черв. 2025 р. 2025. С. 110–113.
90. Білинський Й. Й., Книш Б. П. Цифрова схемотехніка. Електронно-обчислювальні пристрої: навч. посіб. Вінниця: ВНТУ, 2021. 66 с.
91. Parhami B. Computer arithmetic: Algorithms and hardware designs. 2nd ed. New York : Oxford University Press, 2010. 641 p.
92. Ercegovac M. D., Lang T. Review of Basic Number Representations and Arithmetic Algorithms. Digital Arithmetic. 2004. P. 3–49. URL: <https://doi.org/10.1016/b978-155860798-9/50003-8>.
93. Sinclair I. Gating and Logic Circuits. Electronics Simplified. 2011. P. 185–201. URL: <https://doi.org/10.1016/b978-0-08-097063-9.10010-x>.
94. Яковин С. В., Мельничук С. І. Порівняльне оцінювання обчислювальної складності реалізації кореляційного та дискретного перцептронів. Інформаційні технології в освіті, техніці та промисловості:

Матеріали науково-практ. конф., м. Івано-Франківськ, 9 жовт. 2025 р. 2025. C. 317–319. URL: <https://doi.org/10.5281/zenodo.17295363>.

95. Efficient Processing of Deep Neural Networks / V. Sze et al. Morgan & Claypool Publishers, 2020. 341 p.

96. Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks / Y.-H. Chen et al. IEEE Journal of Solid-State Circuits. 2017. Vol. 52, no. 1. P. 127–138. URL: <https://doi.org/10.1109/jssc.2016.2616357>.

97. Loihi: A Neuromorphic Manycore Processor with On-Chip Learning / M. Davies et al. IEEE Micro. 2018. Vol. 38, no. 1. P. 82–99. URL: <https://doi.org/10.1109/mm.2018.112130359>.

98. In-Datcenter Performance Analysis of a Tensor Processing Unit / N. P. Jouppi et al. ACM SIGARCH Computer Architecture News. 2017. Vol. 45, no. 2. P. 1–12. URL: <https://doi.org/10.1145/3140659.3080246>.

99. Can FPGAs Beat GPUs in Accelerating Next-Generation Deep Neural Networks? / E. Nurvitadhi et al. FPGA '17: The 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey California USA. New York, NY, USA, 2017. URL: <https://doi.org/10.1145/3020078.3021740>.

100. Indiveri G., Liu S.-C. Memory and Information Processing in Neuromorphic Systems. Proceedings of the IEEE. 2015. Vol. 103, no. 8. P. 1379–1397. URL: <https://doi.org/10.1109/jproc.2015.2444094>.

101. Benchmarking Energy and Latency in Tinyml: A Novel Method for Resource-Constrained AI / P. Bartoli et al. 2025. 8 p. (Preprint). URL: <https://doi.org/10.48550/arXiv.2505.15622>.

102. Kachris C. A survey on hardware accelerators for large language models. Applied Sciences. 2025. Vol. 15, no. 2. P. 586.

103. Zhu Y. Analysis and application on enhancing CNN performance via FPGA integration. International Conference on Electronics. Electrical and Information Engineering, Haikou, China, 16–18 August 2024 / ed. by S. Li, B. Hu. 2024. P. 31. URL: <https://doi.org/10.1117/12.3052318>.

104. Breakthrough Low-Latency, High-Energy-Efficiency LLM Inference Performance Using NorthPole / R. Appuswamy et al. 2024 IEEE High Performance Extreme Computing Conference (HPEC), Wakefield, MA, USA, 23–27 September 2024. 2024. P. 1–8. URL: <https://doi.org/10.1109/hpec62836.2024.10938418>.
105. FPGA-based deep learning inference accelerators: Where are we standing? / A. Nechi et al. ACM Transactions on Reconfigurable Technology and Systems. 2023. URL: <https://doi.org/10.1145/3613963>.
106. The neurobench framework for benchmarking neuromorphic computing algorithms and systems / J. Yik et al. Nature Communications. 2025. Vol. 16, no. 1. <https://doi.org/10.1038/s41467-025-56739-4>.
107. MAX 7000 CPLD Overview. <https://www.vemeko.com/max-7000/>.
108. FPGA Implementations of Neural Networks / ed. by A. R. Omondi, J. C. Rajapakse. Springer US, 2006. <https://doi.org/10.1007/0-387-28487-7>.
109. Ananthakrishnan A., Allen M. G. All-Passive hardware implementation of multilayer perceptron classifiers. IEEE Transactions on Neural Networks and Learning Systems. 2020. P. 1–10. URL: <https://doi.org/10.1109/tnnls.2020.3016901>.
110. Яковин С. В., Мельничук С. І., Мануляк І. З. Реалізація двохвходового дискретного перцептрону зі зміщеними синаптичними сигналами на ПЛІС засобами AlteraHDL. Вісник Вінницького політехнічного інституту. 2025. № 4. С. 186–194. <https://doi.org/10.31649/1997-9266-2025-181-4-186-194>.

ДОДАТОК А

Програма знаходження коефіцієнтів зміщення та порогу активації дискретного перцептрону для емуляції логічних функцій

```

import numpy
import itertools
import functools
class Perceptron:
    def __init__(self, inputs, aggregation_function, activation_function,
extra_inputs=0):
        self.inputs = inputs
        self.aggregation_function = aggregation_function
        self.activation_function = activation_function
        self.extra_inputs = extra_inputs
    def weight_ranges():
        # ranges = [range(-weight_range, weight_range + 1)] * (n +
extra_inputs) + [range(-weight_range, weight_range + 1)]
        ranges = [range(0, 2 * upper_limit + 1) for upper_limit in range(0, n)]
        return itertools.product(*ranges)
    def count_distinct_elements(arr):
        return len(set(arr))
    def sum_and_count_unique(inputs, weights, extra_inputs=0):
        return count_distinct_elements(numpy.array(inputs + [0] * extra_inputs) +
numpy.array(weights))
    def weighted_sum(inputs, weights, _extra_inputs=None):
        return sum(x * w for x, w in zip(inputs, weights))
    def step_function(x, threshold=0):
        return int(x >= threshold)
    def modified_step_function(x, threshold=0):
        return int(x >= threshold) if threshold >= 0 else int(x <= -threshold)
    def indicator_function(x, threshold=0):
        return int(x == threshold)
    def modified_indicator_function(x, threshold=0):
        return int(x == threshold) if threshold >= 0 else int(x != -threshold)
    def extended_indicator_function(x, limits=(0, 0)):
        return int(limits[0] <= x <= limits[1])
    def perceptron(inputs, weights, aggregation_function=weighted_sum,
activation_function=step_function, extra_inputs=0):
        bias_threshold = weights[-1]
        return activation_function(aggregation_function(inputs, weights[:-1],
extra_inputs), bias_threshold)
    def in_out_matrices(dim, val):
        input_vectors = [number_to_binary_vector(i, bits=dim) for i in range(2 **
dim)]
        output_len = len(input_vectors)
        return input_vectors, number_to_binary_vector(val, bits=output_len)
    def number_to_binary_vector(n, bits=None):
        if bits is None:
            bits = n.bit_length()
        return [(n >> i) & 1 for i in range(bits)]
    def all_weights(weight_range, n, extra_inputs):
        # ranges = [range(-weight_range, weight_range + 1)] * (n + extra_inputs) +
[range(-weight_range, weight_range + 1)]
        main_ranges = [range(0, 2 * upper_limit) for upper_limit in range(1, n +
1)]
        extra_ranges = [range(0, 2 * n + 1)] * extra_inputs
        bias_range = [range(-weight_range, weight_range + 1)]
        ranges = main_ranges + extra_ranges + bias_range
        return itertools.product(*ranges)

```

```

def find_weights(aggregation_functions, activation_functions,
emulated_functions, n, weight_range, extra_inputs, only_one=False):
    for aggregation_function in aggregation_functions:
        print(f"\nAggregation function: {aggregation_function.__name__}")
        for activation_function in activation_functions:
            print(f"\n Activation function: {activation_function.__name__}")
            count_not_found = 0
            for emulated_function in emulated_functions:
                print(f"    Emulated function: {'.'.join(map(str,
number_to_binary_vector(emulated_function, 2 ** n)))}")
                found = False
                inputs, outputs = in_out_matrices(n, emulated_function)
                for w in all_weights(weight_range, n, extra_inputs):
                    y = [perceptron(inputs[i], w, aggregation_function,
activation_function, extra_inputs) for i in range(0, len(inputs))]
                    if y == outputs:
                        found = True
                        print(f"        Weights: {w}")
                        if only_one:
                            break
                if not found:
                    count_not_found += 1

            print(f"\n {count_not_found} functions not found")

aggregation_functions = [
    # weighted_sum,
    sum_and_count_unique
]
activation_functions = [
    # step_function,
    # modified_step_function,
    # indicator_function,
    modified_indicator_function
]

n = 2
# emulated_functions = range(0, 2 ** 2 ** n)
# AND, OR, XOR, NAND, NOR, NXOR
# emulated_functions = [128, 254, 150, 127, 1, 105]
emulated_functions = [9]

# n = 4
# emulated_functions = [32768, 65534, 27030, 32767, 1, 38505]

extra_inputs = 1
weight_range = 3
only_one = True

# find_weights(aggregation_functions, activation_functions,
emulated_functions, n, weight_range, extra_inputs, only_one)

inputs, outputs = in_out_matrices(n, 9)
weights = itertools.product(range(0, weight_range + 1), repeat=n)
fixed_w = 0

for w in weights:
    for i in inputs:
        print(w, i)
        print(sum_and_count_unique(i, [*w, fixed_w], 1), end=',')
    print()

```

ДОДАТОК Б

Коефіцієнти зміщення (w_0, w_1, w_2 та поріг A) , які забезпечують емуляцію відповідних логічних функцій на два входи для всіх можливих комбінацій (функція активації – індикаторна, функції агрегації – Cnt())

Emulated function: 0,0,0,0

Weights: (-3, -3, -3, -3)
 Weights: (-3, -3, -3, -2)
 Weights: (-3, -3, -3, -1)
 Weights: (-3, -3, -3, 0)
 Weights: (-3, -3, -3, 3)
 Weights: (-3, -3, -2, -3)
 Weights: (-3, -3, -2, -2)
 Weights: (-3, -3, -2, -1)
 Weights: (-3, -3, -2, 0)
 Weights: (-3, -3, -2, 3)
 Weights: (-3, -3, -1, -3)
 Weights: (-3, -3, -1, -2)
 Weights: (-3, -3, -1, -1)
 Weights: (-3, -3, -1, 0)
 Weights: (-3, -3, -1, 1)
 Weights: (-3, -3, 0, -3)
 Weights: (-3, -3, 0, -2)
 Weights: (-3, -3, 0, -1)
 Weights: (-3, -3, 0, 0)
 Weights: (-3, -3, 0, 1)
 Weights: (-3, -3, 1, -3)
 Weights: (-3, -3, 1, -2)
 Weights: (-3, -3, 1, -1)
 Weights: (-3, -3, 1, 0)
 Weights: (-3, -3, 1, 1)
 Weights: (-3, -3, 2, -3)
 Weights: (-3, -3, 2, -2)
 Weights: (-3, -3, 2, -1)
 Weights: (-3, -3, 2, 0)
 Weights: (-3, -3, 2, 1)
 Weights: (-3, -3, 3, -3)
 Weights: (-3, -3, 3, -2)
 Weights: (-3, -3, 3, -1)
 Weights: (-3, -3, 3, 0)
 Weights: (-3, -3, 3, 1)
 Weights: (-3, -2, -3, -3)
 Weights: (-3, -2, -3, -2)
 Weights: (-3, -2, -3, -1)
 Weights: (-3, -2, -3, 0)
 Weights: (-3, -2, -3, 1)
 Weights: (-3, -2, -2, -3)
 Weights: (-3, -2, -2, -2)
 Weights: (-3, -2, -2, -1)
 Weights: (-3, -2, -2, 0)
 Weights: (-3, -2, -1, -3)
 Weights: (-3, -2, -1, -2)
 Weights: (-3, -2, -1, -1)
 Weights: (-3, -2, -1, 0)
 Weights: (-3, -2, -1, 1)
 Weights: (-3, -2, 0, -3)
 Weights: (-3, -2, 0, -2)
 Weights: (-3, -2, 0, -1)
 Weights: (-3, -2, 0, 0)
 Weights: (-3, -2, 0, 1)
 Weights: (-3, -2, 1, -3)
 Weights: (-3, -2, 1, -2)
 Weights: (-3, -2, 1, -1)
 Weights: (-3, -2, 1, 0)
 Weights: (-3, -2, 1, 1)
 Weights: (-3, -2, 2, -3)
 Weights: (-3, -2, 2, -2)
 Weights: (-3, -2, 2, -1)
 Weights: (-3, -2, 2, 0)
 Weights: (-3, -2, 2, 1)
 Weights: (-3, -2, 3, -3)
 Weights: (-3, -2, 3, -2)

Weights: (-3, -2, 3, -1)
 Weights: (-3, -2, 3, 0)
 Weights: (-3, -2, 3, 1)
 Weights: (-3, -1, -3, -3)
 Weights: (-3, -1, -3, -2)
 Weights: (-3, -1, -3, -1)
 Weights: (-3, -1, -3, 0)
 Weights: (-3, -1, -3, 1)
 Weights: (-3, -1, -2, -3)
 Weights: (-3, -1, -2, -2)
 Weights: (-3, -1, -2, -1)
 Weights: (-3, -1, -2, 0)
 Weights: (-3, -1, -2, 1)
 Weights: (-3, -1, -1, -3)
 Weights: (-3, -1, -1, -2)
 Weights: (-3, -1, -1, -1)
 Weights: (-3, -1, -1, 0)
 Weights: (-3, -1, -1, 1)
 Weights: (-3, -1, 0, -3)
 Weights: (-3, -1, 0, -2)
 Weights: (-3, -1, 0, -1)
 Weights: (-3, -1, 0, 0)
 Weights: (-3, -1, 0, 1)
 Weights: (-3, -1, 1, -3)
 Weights: (-3, -1, 1, -2)
 Weights: (-3, -1, 1, -1)
 Weights: (-3, -1, 1, 0)
 Weights: (-3, -1, 1, 1)
 Weights: (-3, -1, 1, 2)
 Weights: (-3, -1, 2, -3)
 Weights: (-3, -1, 2, -2)
 Weights: (-3, -1, 2, -1)
 Weights: (-3, -1, 2, 0)
 Weights: (-3, -1, 2, 1)
 Weights: (-3, -1, 2, 2)
 Weights: (-3, -1, 3, -3)
 Weights: (-3, -1, 3, -2)
 Weights: (-3, -1, 3, -1)
 Weights: (-3, -1, 3, 0)
 Weights: (-3, -1, 3, 1)
 Weights: (-3, -1, 3, 2)
 Weights: (-3, 0, -3, -3)
 Weights: (-3, 0, -3, -2)
 Weights: (-3, 0, -3, -1)
 Weights: (-3, 0, -3, 0)
 Weights: (-3, 0, -2, -3)
 Weights: (-3, 0, -2, -2)
 Weights: (-3, 0, -2, -1)
 Weights: (-3, 0, -2, 0)
 Weights: (-3, 0, -2, 1)
 Weights: (-3, 0, -1, -3)
 Weights: (-3, 0, -1, -2)
 Weights: (-3, 0, -1, -1)
 Weights: (-3, 0, -1, 0)
 Weights: (-3, 0, -1, 1)
 Weights: (-3, 0, -1, 2)
 Weights: (-3, 0, 0, -3)
 Weights: (-3, 0, 0, -2)
 Weights: (-3, 0, 0, -1)
 Weights: (-3, 0, 0, 0)
 Weights: (-3, 0, 0, 1)
 Weights: (-3, 0, 1, -3)
 Weights: (-3, 0, 1, -2)
 Weights: (-3, 0, 1, -1)
 Weights: (-3, 0, 1, 0)
 Weights: (-3, 0, 1, 1)

Weights: (-3, 0, 2, -3)
 Weights: (-3, 0, 2, -2)
 Weights: (-3, 0, 2, -1)
 Weights: (-3, 0, 2, 0)
 Weights: (-3, 0, 2, 1)
 Weights: (-3, 0, 2, 2)
 Weights: (-3, 0, 3, -3)
 Weights: (-3, 0, 3, -2)
 Weights: (-3, 0, 3, -1)
 Weights: (-3, 0, 3, 0)
 Weights: (-3, 0, 3, 1)
 Weights: (-3, 0, 3, 2)
 Weights: (-3, 1, -3, -3)
 Weights: (-3, 1, -3, -2)
 Weights: (-3, 1, -3, -1)
 Weights: (-3, 1, -3, 0)
 Weights: (-3, 1, -3, 1)
 Weights: (-3, 1, -2, -3)
 Weights: (-3, 1, -2, -2)
 Weights: (-3, 1, -2, -1)
 Weights: (-3, 1, -2, 0)
 Weights: (-3, 1, -2, 1)
 Weights: (-3, 1, -1, -3)
 Weights: (-3, 1, -1, -2)
 Weights: (-3, 1, -1, -1)
 Weights: (-3, 1, -1, 0)
 Weights: (-3, 1, -1, 1)
 Weights: (-3, 1, -1, 2)
 Weights: (-3, 1, 0, -3)
 Weights: (-3, 1, 0, -2)
 Weights: (-3, 1, 0, -1)
 Weights: (-3, 1, 0, 0)
 Weights: (-3, 1, 0, 1)
 Weights: (-3, 1, 0, 2)
 Weights: (-3, 1, 1, -3)
 Weights: (-3, 1, 1, -2)
 Weights: (-3, 1, 1, -1)
 Weights: (-3, 1, 1, 0)
 Weights: (-3, 1, 1, 1)
 Weights: (-3, 1, 2, -3)
 Weights: (-3, 1, 2, -2)
 Weights: (-3, 1, 2, -1)
 Weights: (-3, 1, 2, 0)
 Weights: (-3, 1, 2, 1)
 Weights: (-3, 1, 3, -3)
 Weights: (-3, 1, 3, -2)
 Weights: (-3, 1, 3, -1)
 Weights: (-3, 1, 3, 0)
 Weights: (-3, 1, 3, 1)
 Weights: (-3, 1, 3, 2)
 Weights: (-3, 2, -3, -3)
 Weights: (-3, 2, -3, -2)
 Weights: (-3, 2, -3, -1)
 Weights: (-3, 2, -3, 0)
 Weights: (-3, 2, -3, 1)
 Weights: (-3, 2, -2, -3)
 Weights: (-3, 2, -2, -2)
 Weights: (-3, 2, -2, -1)
 Weights: (-3, 2, -2, 0)
 Weights: (-3, 2, -2, 1)
 Weights: (-3, 2, -1, -3)
 Weights: (-3, 2, -1, -2)
 Weights: (-3, 2, -1, -1)
 Weights: (-3, 2, -1, 0)
 Weights: (-3, 2, -1, 1)
 Weights: (-3, 2, -1, 2)
 Weights: (-3, 2, 0, -3)

Weights: (-2, 2, -3, -2)	Weights: (-1, -3, -3, 1)	Weights: (-1, -1, -1, -1)
Weights: (-2, 2, -3, -1)	Weights: (-1, -3, -2, -3)	Weights: (-1, -1, -1, 0)
Weights: (-2, 2, -3, 0)	Weights: (-1, -3, -2, -2)	Weights: (-1, -1, -1, 3)
Weights: (-2, 2, -3, 1)	Weights: (-1, -3, -2, -1)	Weights: (-1, -1, 0, -3)
Weights: (-2, 2, -3, 2)	Weights: (-1, -3, -2, 0)	Weights: (-1, -1, 0, -2)
Weights: (-2, 2, -2, -3)	Weights: (-1, -3, -2, 1)	Weights: (-1, -1, 0, -1)
Weights: (-2, 2, -2, -2)	Weights: (-1, -3, -1, -3)	Weights: (-1, -1, 0, 0)
Weights: (-2, 2, -2, -1)	Weights: (-1, -3, -1, -2)	Weights: (-1, -1, 0, 3)
Weights: (-2, 2, -2, 0)	Weights: (-1, -3, -1, -1)	Weights: (-1, -1, 1, -3)
Weights: (-2, 2, -2, 1)	Weights: (-1, -3, -1, 0)	Weights: (-1, -1, 1, -2)
Weights: (-2, 2, -1, -3)	Weights: (-1, -3, -1, 1)	Weights: (-1, -1, 1, -1)
Weights: (-2, 2, -1, -2)	Weights: (-1, -3, 0, -3)	Weights: (-1, -1, 1, 0)
Weights: (-2, 2, -1, -1)	Weights: (-1, -3, 0, -2)	Weights: (-1, -1, 1, 1)
Weights: (-2, 2, -1, 0)	Weights: (-1, -3, 0, -1)	Weights: (-1, -1, 2, -3)
Weights: (-2, 2, -1, 1)	Weights: (-1, -3, 0, 0)	Weights: (-1, -1, 2, -2)
Weights: (-2, 2, 0, -3)	Weights: (-1, -3, 0, 1)	Weights: (-1, -1, 2, -1)
Weights: (-2, 2, 0, -2)	Weights: (-1, -3, 1, -3)	Weights: (-1, -1, 2, 0)
Weights: (-2, 2, 0, -1)	Weights: (-1, -3, 1, -2)	Weights: (-1, -1, 2, 1)
Weights: (-2, 2, 0, 0)	Weights: (-1, -3, 1, -1)	Weights: (-1, -1, 3, -3)
Weights: (-2, 2, 0, 1)	Weights: (-1, -3, 1, 0)	Weights: (-1, -1, 3, -2)
Weights: (-2, 2, 0, 2)	Weights: (-1, -3, 1, 1)	Weights: (-1, -1, 3, -1)
Weights: (-2, 2, 1, -3)	Weights: (-1, -3, 1, 2)	Weights: (-1, -1, 3, 0)
Weights: (-2, 2, 1, -2)	Weights: (-1, -3, 2, -3)	Weights: (-1, -1, 3, 1)
Weights: (-2, 2, 1, -1)	Weights: (-1, -3, 2, -2)	Weights: (-1, 0, -3, -3)
Weights: (-2, 2, 1, 0)	Weights: (-1, -3, 2, -1)	Weights: (-1, 0, -3, -2)
Weights: (-2, 2, 1, 1)	Weights: (-1, -3, 2, 0)	Weights: (-1, 0, -3, -1)
Weights: (-2, 2, 1, 2)	Weights: (-1, -3, 2, 1)	Weights: (-1, 0, -3, 0)
Weights: (-2, 2, 2, -3)	Weights: (-1, -3, 2, 2)	Weights: (-1, 0, -3, 1)
Weights: (-2, 2, 2, -2)	Weights: (-1, -3, 3, -3)	Weights: (-1, 0, -2, -3)
Weights: (-2, 2, 2, -1)	Weights: (-1, -3, 3, -2)	Weights: (-1, 0, -2, -2)
Weights: (-2, 2, 2, 0)	Weights: (-1, -3, 3, -1)	Weights: (-1, 0, -2, -1)
Weights: (-2, 2, 2, 1)	Weights: (-1, -3, 3, 0)	Weights: (-1, 0, -2, 0)
Weights: (-2, 2, 3, -3)	Weights: (-1, -3, 3, 1)	Weights: (-1, 0, -2, 1)
Weights: (-2, 2, 3, -2)	Weights: (-1, -3, 3, 2)	Weights: (-1, 0, -1, -3)
Weights: (-2, 2, 3, -1)	Weights: (-1, -2, -3, -3)	Weights: (-1, 0, -1, -2)
Weights: (-2, 2, 3, 0)	Weights: (-1, -2, -3, -2)	Weights: (-1, 0, -1, -1)
Weights: (-2, 2, 3, 1)	Weights: (-1, -2, -3, -1)	Weights: (-1, 0, -1, 0)
Weights: (-2, 3, -3, -3)	Weights: (-1, -2, -3, 0)	Weights: (-1, 0, -1, 1)
Weights: (-2, 3, -3, -2)	Weights: (-1, -2, -3, 1)	Weights: (-1, 0, 0, -3)
Weights: (-2, 3, -3, -1)	Weights: (-1, -2, -2, -3)	Weights: (-1, 0, 0, -2)
Weights: (-2, 3, -3, 0)	Weights: (-1, -2, -2, -2)	Weights: (-1, 0, 0, -1)
Weights: (-2, 3, -3, 1)	Weights: (-1, -2, -2, -1)	Weights: (-1, 0, 0, 0)
Weights: (-2, 3, -3, 2)	Weights: (-1, -2, -2, 0)	Weights: (-1, 0, 1, -3)
Weights: (-2, 3, -2, -3)	Weights: (-1, -2, -2, 1)	Weights: (-1, 0, 1, -2)
Weights: (-2, 3, -2, -2)	Weights: (-1, -2, -1, -3)	Weights: (-1, 0, 1, -1)
Weights: (-2, 3, -2, -1)	Weights: (-1, -2, -1, -2)	Weights: (-1, 0, 1, 0)
Weights: (-2, 3, -2, 0)	Weights: (-1, -2, -1, -1)	Weights: (-1, 0, 1, 1)
Weights: (-2, 3, -2, 1)	Weights: (-1, -2, -1, 0)	Weights: (-1, 0, 2, -3)
Weights: (-2, 3, -1, -3)	Weights: (-1, -2, 0, -3)	Weights: (-1, 0, 2, -2)
Weights: (-2, 3, -1, -2)	Weights: (-1, -2, 0, -2)	Weights: (-1, 0, 2, -1)
Weights: (-2, 3, -1, -1)	Weights: (-1, -2, 0, -1)	Weights: (-1, 0, 2, 0)
Weights: (-2, 3, -1, 0)	Weights: (-1, -2, 0, 0)	Weights: (-1, 0, 2, 1)
Weights: (-2, 3, -1, 1)	Weights: (-1, -2, 0, 1)	Weights: (-1, 0, 3, -3)
Weights: (-2, 3, 0, -3)	Weights: (-1, -2, 1, -3)	Weights: (-1, 0, 3, -2)
Weights: (-2, 3, 0, -2)	Weights: (-1, -2, 1, -2)	Weights: (-1, 0, 3, -1)
Weights: (-2, 3, 0, -1)	Weights: (-1, -2, 1, -1)	Weights: (-1, 0, 3, 0)
Weights: (-2, 3, 0, 0)	Weights: (-1, -2, 1, 0)	Weights: (-1, 0, 3, 1)
Weights: (-2, 3, 0, 1)	Weights: (-1, -2, 1, 1)	Weights: (-1, 1, -3, -3)
Weights: (-2, 3, 0, 2)	Weights: (-1, -2, 2, -3)	Weights: (-1, 1, -3, -2)
Weights: (-2, 3, 1, -3)	Weights: (-1, -2, 2, -2)	Weights: (-1, 1, -3, -1)
Weights: (-2, 3, 1, -2)	Weights: (-1, -2, 2, -1)	Weights: (-1, 1, -3, 0)
Weights: (-2, 3, 1, -1)	Weights: (-1, -2, 2, 0)	Weights: (-1, 1, -3, 1)
Weights: (-2, 3, 1, 0)	Weights: (-1, -2, 2, 1)	Weights: (-1, 1, -3, 2)
Weights: (-2, 3, 1, 1)	Weights: (-1, -2, 3, -3)	Weights: (-1, 1, -2, -3)
Weights: (-2, 3, 1, 2)	Weights: (-1, -2, 3, -2)	Weights: (-1, 1, -2, -2)
Weights: (-2, 3, 2, -3)	Weights: (-1, -2, 3, -1)	Weights: (-1, 1, -2, -1)
Weights: (-2, 3, 2, -2)	Weights: (-1, -2, 3, 0)	Weights: (-1, 1, -2, 0)
Weights: (-2, 3, 2, -1)	Weights: (-1, -2, 3, 1)	Weights: (-1, 1, -2, 1)
Weights: (-2, 3, 2, 0)	Weights: (-1, -1, -3, -3)	Weights: (-1, 1, -2, 2)
Weights: (-2, 3, 2, 1)	Weights: (-1, -1, -3, -2)	Weights: (-1, 1, -1, -3)
Weights: (-2, 3, 2, 2)	Weights: (-1, -1, -3, -1)	Weights: (-1, 1, -1, -2)
Weights: (-2, 3, 3, -3)	Weights: (-1, -1, -3, 0)	Weights: (-1, 1, -1, -1)
Weights: (-2, 3, 3, -2)	Weights: (-1, -1, -3, 1)	Weights: (-1, 1, -1, 0)
Weights: (-2, 3, 3, -1)	Weights: (-1, -1, -2, -3)	Weights: (-1, 1, -1, 1)
Weights: (-2, 3, 3, 0)	Weights: (-1, -1, -2, -2)	Weights: (-1, 1, 0, -3)
Weights: (-2, 3, 3, 1)	Weights: (-1, -1, -2, -1)	Weights: (-1, 1, 0, -2)
Weights: (-1, -3, -3, -3)	Weights: (-1, -1, -2, 0)	Weights: (-1, 1, 0, -1)
Weights: (-1, -3, -3, -2)	Weights: (-1, -1, -2, 1)	Weights: (-1, 1, 0, 0)
Weights: (-1, -3, -3, -1)	Weights: (-1, -1, -1, -3)	Weights: (-1, 1, 0, 1)
Weights: (-1, -3, -3, 0)	Weights: (-1, -1, -1, -2)	Weights: (-1, 1, 1, -3)

Weights: (0, 1, -3, 0)	Weights: (0, 3, -2, 2)	Weights: (1, -2, -1, 0)
Weights: (0, 1, -3, 1)	Weights: (0, 3, -1, -3)	Weights: (1, -2, -1, 1)
Weights: (0, 1, -2, -3)	Weights: (0, 3, -1, -2)	Weights: (1, -2, 0, -3)
Weights: (0, 1, -2, -2)	Weights: (0, 3, -1, -1)	Weights: (1, -2, 0, -2)
Weights: (0, 1, -2, -1)	Weights: (0, 3, -1, 0)	Weights: (1, -2, 0, -1)
Weights: (0, 1, -2, 0)	Weights: (0, 3, -1, 1)	Weights: (1, -2, 0, 0)
Weights: (0, 1, -2, 1)	Weights: (0, 3, -1, 2)	Weights: (1, -2, 0, 1)
Weights: (0, 1, -1, -3)	Weights: (0, 3, 0, -3)	Weights: (1, -2, 0, 2)
Weights: (0, 1, -1, -2)	Weights: (0, 3, 0, -2)	Weights: (1, -2, 1, -3)
Weights: (0, 1, -1, -1)	Weights: (0, 3, 0, -1)	Weights: (1, -2, 1, -2)
Weights: (0, 1, -1, 0)	Weights: (0, 3, 0, 0)	Weights: (1, -2, 1, -1)
Weights: (0, 1, -1, 1)	Weights: (0, 3, 0, 1)	Weights: (1, -2, 1, 0)
Weights: (0, 1, 0, -3)	Weights: (0, 3, 1, -3)	Weights: (1, -2, 1, 1)
Weights: (0, 1, 0, -2)	Weights: (0, 3, 1, -2)	Weights: (1, -2, 2, -3)
Weights: (0, 1, 0, -1)	Weights: (0, 3, 1, -1)	Weights: (1, -2, 2, -2)
Weights: (0, 1, 0, 0)	Weights: (0, 3, 1, 0)	Weights: (1, -2, 2, -1)
Weights: (0, 1, 0, 1)	Weights: (0, 3, 1, 1)	Weights: (1, -2, 2, 0)
Weights: (0, 1, 1, -3)	Weights: (0, 3, 2, -3)	Weights: (1, -2, 2, 1)
Weights: (0, 1, 1, -2)	Weights: (0, 3, 2, -2)	Weights: (1, -2, 3, -3)
Weights: (0, 1, 1, -1)	Weights: (0, 3, 2, -1)	Weights: (1, -2, 3, -2)
Weights: (0, 1, 1, 0)	Weights: (0, 3, 2, 0)	Weights: (1, -2, 3, -1)
Weights: (0, 1, 2, -3)	Weights: (0, 3, 2, 1)	Weights: (1, -2, 3, 0)
Weights: (0, 1, 2, -2)	Weights: (0, 3, 2, 2)	Weights: (1, -2, 3, 1)
Weights: (0, 1, 2, -1)	Weights: (0, 3, 3, -3)	Weights: (1, -2, 3, 2)
Weights: (0, 1, 2, 0)	Weights: (0, 3, 3, -2)	Weights: (1, -1, -3, -3)
Weights: (0, 1, 2, 1)	Weights: (0, 3, 3, -1)	Weights: (1, -1, -3, -2)
Weights: (0, 1, 3, -3)	Weights: (0, 3, 3, 0)	Weights: (1, -1, -3, -1)
Weights: (0, 1, 3, -2)	Weights: (0, 3, 3, 1)	Weights: (1, -1, -3, 0)
Weights: (0, 1, 3, -1)	Weights: (1, -3, -3, -3)	Weights: (1, -1, -3, 1)
Weights: (0, 1, 3, 0)	Weights: (1, -3, -3, -2)	Weights: (1, -1, -3, 2)
Weights: (0, 1, 3, 1)	Weights: (1, -3, -3, -1)	Weights: (1, -1, -2, -3)
Weights: (0, 2, -3, -3)	Weights: (1, -3, -3, 0)	Weights: (1, -1, -2, -2)
Weights: (0, 2, -3, -2)	Weights: (1, -3, -3, 1)	Weights: (1, -1, -2, -1)
Weights: (0, 2, -3, -1)	Weights: (1, -3, -2, -3)	Weights: (1, -1, -2, 0)
Weights: (0, 2, -3, 0)	Weights: (1, -3, -2, -2)	Weights: (1, -1, -2, 1)
Weights: (0, 2, -3, 1)	Weights: (1, -3, -2, -1)	Weights: (1, -1, -2, 2)
Weights: (0, 2, -3, 2)	Weights: (1, -3, -2, 0)	Weights: (1, -1, -1, -3)
Weights: (0, 2, -2, -3)	Weights: (1, -3, -2, 1)	Weights: (1, -1, -1, -2)
Weights: (0, 2, -2, -2)	Weights: (1, -3, -1, -3)	Weights: (1, -1, -1, -1)
Weights: (0, 2, -2, -1)	Weights: (1, -3, -1, -2)	Weights: (1, -1, -1, 0)
Weights: (0, 2, -2, 0)	Weights: (1, -3, -1, -1)	Weights: (1, -1, -1, 1)
Weights: (0, 2, -2, 1)	Weights: (1, -3, -1, 0)	Weights: (1, -1, 0, -3)
Weights: (0, 2, -2, 2)	Weights: (1, -3, -1, 1)	Weights: (1, -1, 0, -2)
Weights: (0, 2, -1, -3)	Weights: (1, -3, -1, 2)	Weights: (1, -1, 0, -1)
Weights: (0, 2, -1, -2)	Weights: (1, -3, 0, -3)	Weights: (1, -1, 0, 0)
Weights: (0, 2, -1, -1)	Weights: (1, -3, 0, -2)	Weights: (1, -1, 0, 1)
Weights: (0, 2, -1, 0)	Weights: (1, -3, 0, -1)	Weights: (1, -1, 1, -3)
Weights: (0, 2, -1, 1)	Weights: (1, -3, 0, 0)	Weights: (1, -1, 1, -2)
Weights: (0, 2, -1, 2)	Weights: (1, -3, 0, 1)	Weights: (1, -1, 1, -1)
Weights: (0, 2, 0, -3)	Weights: (1, -3, 0, 2)	Weights: (1, -1, 1, 0)
Weights: (0, 2, 0, -2)	Weights: (1, -3, 1, -3)	Weights: (1, -1, 1, 1)
Weights: (0, 2, 0, -1)	Weights: (1, -3, 1, -2)	Weights: (1, -1, 2, -3)
Weights: (0, 2, 0, 0)	Weights: (1, -3, 1, -1)	Weights: (1, -1, 2, -2)
Weights: (0, 2, 0, 1)	Weights: (1, -3, 1, 0)	Weights: (1, -1, 2, -1)
Weights: (0, 2, 1, -3)	Weights: (1, -3, 1, 1)	Weights: (1, -1, 2, 0)
Weights: (0, 2, 1, -2)	Weights: (1, -3, 2, -3)	Weights: (1, -1, 2, 1)
Weights: (0, 2, 1, -1)	Weights: (1, -3, 2, -2)	Weights: (1, -1, 3, -3)
Weights: (0, 2, 1, 0)	Weights: (1, -3, 2, -1)	Weights: (1, -1, 3, -2)
Weights: (0, 2, 1, 1)	Weights: (1, -3, 2, 0)	Weights: (1, -1, 3, -1)
Weights: (0, 2, 2, -3)	Weights: (1, -3, 2, 1)	Weights: (1, -1, 3, 0)
Weights: (0, 2, 2, -2)	Weights: (1, -3, 3, -3)	Weights: (1, -1, 3, 1)
Weights: (0, 2, 2, -1)	Weights: (1, -3, 3, -2)	Weights: (1, -1, 3, 2)
Weights: (0, 2, 2, 0)	Weights: (1, -3, 3, -1)	Weights: (1, 0, -3, -3)
Weights: (0, 2, 2, 1)	Weights: (1, -3, 3, 0)	Weights: (1, 0, -3, -2)
Weights: (0, 2, 3, -3)	Weights: (1, -3, 3, 1)	Weights: (1, 0, -3, -1)
Weights: (0, 2, 3, -2)	Weights: (1, -3, 3, 2)	Weights: (1, 0, -3, 0)
Weights: (0, 2, 3, -1)	Weights: (1, -2, -3, -3)	Weights: (1, 0, -3, 1)
Weights: (0, 2, 3, 0)	Weights: (1, -2, -3, -2)	Weights: (1, 0, -2, -3)
Weights: (0, 2, 3, 1)	Weights: (1, -2, -3, -1)	Weights: (1, 0, -2, -2)
Weights: (0, 3, -3, -3)	Weights: (1, -2, -3, 0)	Weights: (1, 0, -2, -1)
Weights: (0, 3, -3, -2)	Weights: (1, -2, -3, 1)	Weights: (1, 0, -2, 0)
Weights: (0, 3, -3, -1)	Weights: (1, -2, -3, 2)	Weights: (1, 0, -2, 1)
Weights: (0, 3, -3, 0)	Weights: (1, -2, -2, -3)	Weights: (1, 0, -1, -3)
Weights: (0, 3, -3, 1)	Weights: (1, -2, -2, -2)	Weights: (1, 0, -1, -2)
Weights: (0, 3, -3, 2)	Weights: (1, -2, -2, -1)	Weights: (1, 0, -1, -1)
Weights: (0, 3, -2, -3)	Weights: (1, -2, -2, 0)	Weights: (1, 0, -1, 0)
Weights: (0, 3, -2, -2)	Weights: (1, -2, -2, 1)	Weights: (1, 0, 0, -3)
Weights: (0, 3, -2, -1)	Weights: (1, -2, -1, -3)	Weights: (1, 0, 0, -2)
Weights: (0, 3, -2, 0)	Weights: (1, -2, -1, -2)	Weights: (1, 0, 0, -1)
Weights: (0, 3, -2, 1)	Weights: (1, -2, -1, -1)	

Weights: (1, 0, 0, 0)
Weights: (1, 0, 0, 1)
Weights: (1, 0, 1, -3)
Weights: (1, 0, 1, -2)
Weights: (1, 0, 1, -1)
Weights: (1, 0, 1, 0)
Weights: (1, 0, 2, -3)
Weights: (1, 0, 2, -2)
Weights: (1, 0, 2, -1)
Weights: (1, 0, 2, 0)
Weights: (1, 0, 2, 1)
Weights: (1, 0, 3, -3)
Weights: (1, 0, 3, -2)
Weights: (1, 0, 3, -1)
Weights: (1, 0, 3, 0)
Weights: (1, 0, 3, 1)
Weights: (1, 1, -3, -3)
Weights: (1, 1, -3, -2)
Weights: (1, 1, -3, -1)
Weights: (1, 1, -3, 0)
Weights: (1, 1, -3, 1)
Weights: (1, 1, -2, -3)
Weights: (1, 1, -2, -2)
Weights: (1, 1, -2, -1)
Weights: (1, 1, -2, 0)
Weights: (1, 1, -2, 1)
Weights: (1, 1, -1, -3)
Weights: (1, 1, -1, -2)
Weights: (1, 1, -1, -1)
Weights: (1, 1, -1, 0)
Weights: (1, 1, -1, 1)
Weights: (1, 1, 0, -3)
Weights: (1, 1, 0, -2)
Weights: (1, 1, 0, -1)
Weights: (1, 1, 0, 0)
Weights: (1, 1, 0, 1)
Weights: (1, 1, 1, -3)
Weights: (1, 1, 1, -2)
Weights: (1, 1, 1, -1)
Weights: (1, 1, 1, 0)
Weights: (1, 1, 1, 3)
Weights: (1, 1, 2, -3)
Weights: (1, 1, 2, -2)
Weights: (1, 1, 2, -1)
Weights: (1, 1, 2, 0)
Weights: (1, 1, 2, 3)
Weights: (1, 1, 3, -3)
Weights: (1, 1, 3, -2)
Weights: (1, 1, 3, -1)
Weights: (1, 1, 3, 0)
Weights: (1, 1, 3, 1)
Weights: (1, 2, -3, -3)
Weights: (1, 2, -3, -2)
Weights: (1, 2, -3, -1)
Weights: (1, 2, -3, 0)
Weights: (1, 2, -3, 1)
Weights: (1, 2, -2, -3)
Weights: (1, 2, -2, -2)
Weights: (1, 2, -2, -1)
Weights: (1, 2, -2, 0)
Weights: (1, 2, -2, 1)
Weights: (1, 2, -1, -3)
Weights: (1, 2, -1, -2)
Weights: (1, 2, -1, -1)
Weights: (1, 2, -1, 0)
Weights: (1, 2, -1, 1)
Weights: (1, 2, 0, -3)
Weights: (1, 2, 0, -2)
Weights: (1, 2, 0, -1)
Weights: (1, 2, 0, 0)
Weights: (1, 2, 0, 1)
Weights: (1, 2, 1, -3)
Weights: (1, 2, 1, -2)
Weights: (1, 2, 1, -1)
Weights: (1, 2, 1, 0)
Weights: (1, 2, 1, 1)
Weights: (1, 2, 2, -3)
Weights: (1, 2, 2, -2)
Weights: (1, 2, 2, -1)
Weights: (1, 2, 2, 0)

Weights: (3, -1, 1, -2)
 Weights: (3, -1, 1, -1)
 Weights: (3, -1, 1, 0)
 Weights: (3, -1, 1, 1)
 Weights: (3, -1, 1, 2)
 Weights: (3, -1, 2, -3)
 Weights: (3, -1, 2, -2)
 Weights: (3, -1, 2, -1)
 Weights: (3, -1, 2, 0)
 Weights: (3, -1, 2, 1)
 Weights: (3, -1, 2, 2)
 Weights: (3, -1, 3, -3)
 Weights: (3, -1, 3, -2)
 Weights: (3, -1, 3, -1)
 Weights: (3, -1, 3, 0)
 Weights: (3, -1, 3, 1)
 Weights: (3, 0, -3, -3)
 Weights: (3, 0, -3, -2)
 Weights: (3, 0, -3, -1)
 Weights: (3, 0, -3, 0)
 Weights: (3, 0, -3, 1)
 Weights: (3, 0, -3, 2)
 Weights: (3, 0, -2, -3)
 Weights: (3, 0, -2, -2)
 Weights: (3, 0, -2, -1)
 Weights: (3, 0, -2, 0)
 Weights: (3, 0, -2, 1)
 Weights: (3, 0, -2, 2)
 Weights: (3, 0, -1, -3)
 Weights: (3, 0, -1, -2)
 Weights: (3, 0, -1, -1)
 Weights: (3, 0, -1, 0)
 Weights: (3, 0, -1, 1)
 Weights: (3, 0, -1, 2)
 Weights: (3, 0, 0, -3)
 Weights: (3, 0, 0, -2)
 Weights: (3, 0, 0, -1)
 Weights: (3, 0, 0, 0)
 Weights: (3, 0, 0, 1)
 Weights: (3, 0, 1, -3)
 Weights: (3, 0, 1, -2)
 Weights: (3, 0, 1, -1)
 Weights: (3, 0, 1, 0)
 Weights: (3, 0, 1, 1)
 Weights: (3, 0, 2, -3)
 Weights: (3, 0, 2, -2)
 Weights: (3, 0, 2, -1)
 Weights: (3, 0, 2, 0)
 Weights: (3, 0, 2, 1)
 Weights: (3, 0, 2, 2)
 Weights: (3, 0, 3, -3)
 Weights: (3, 0, 3, -2)
 Weights: (3, 0, 3, -1)
 Weights: (3, 0, 3, 0)
 Weights: (3, 0, 3, 1)
 Weights: (3, 1, -3, -3)
 Weights: (3, 1, -3, -2)
 Weights: (3, 1, -3, -1)
 Weights: (3, 1, -3, 0)
 Weights: (3, 1, -3, 1)
 Weights: (3, 1, -3, 2)
 Weights: (3, 1, -2, -3)
 Weights: (3, 1, -2, -2)
 Weights: (3, 1, -2, -1)
 Weights: (3, 1, -2, 0)
 Weights: (3, 1, -2, 1)
 Weights: (3, 1, -2, 2)
 Weights: (3, 1, -1, -3)
 Weights: (3, 1, -1, -2)
 Weights: (3, 1, -1, -1)
 Weights: (3, 1, -1, 0)
 Weights: (3, 1, -1, 1)
 Weights: (3, 1, -1, 2)
 Weights: (3, 1, 0, -3)
 Weights: (3, 1, 0, -2)
 Weights: (3, 1, 0, -1)
 Weights: (3, 1, 0, 0)
 Weights: (3, 1, 0, 1)
 Weights: (3, 1, 0, 2)
 Weights: (3, 1, 1, -3)

Weights: (3, 1, 1, -2)
 Weights: (3, 1, 1, -1)
 Weights: (3, 1, 1, 0)
 Weights: (3, 1, 1, 1)
 Weights: (3, 1, 2, -3)
 Weights: (3, 1, 2, -2)
 Weights: (3, 1, 2, -1)
 Weights: (3, 1, 2, 0)
 Weights: (3, 1, 2, 1)
 Weights: (3, 1, 3, -3)
 Weights: (3, 1, 3, -2)
 Weights: (3, 1, 3, -1)
 Weights: (3, 1, 3, 0)
 Weights: (3, 1, 3, 1)
 Weights: (3, 2, -3, -3)
 Weights: (3, 2, -3, -2)
 Weights: (3, 2, -3, -1)
 Weights: (3, 2, -3, 0)
 Weights: (3, 2, -3, 1)
 Weights: (3, 2, -2, -3)
 Weights: (3, 2, -2, -2)
 Weights: (3, 2, -2, -1)
 Weights: (3, 2, -2, 0)
 Weights: (3, 2, -2, 1)
 Weights: (3, 2, -1, -3)
 Weights: (3, 2, -1, -2)
 Weights: (3, 2, -1, -1)
 Weights: (3, 2, -1, 0)
 Weights: (3, 2, -1, 1)
 Weights: (3, 2, 0, -3)
 Weights: (3, 2, 0, -2)
 Weights: (3, 2, 0, -1)
 Weights: (3, 2, 0, 0)
 Weights: (3, 2, 0, 1)
 Weights: (3, 2, 1, -3)
 Weights: (3, 2, 1, -2)
 Weights: (3, 2, 1, -1)
 Weights: (3, 2, 1, 0)
 Weights: (3, 2, 1, 1)
 Weights: (3, 2, 2, -3)
 Weights: (3, 2, 2, -2)
 Weights: (3, 2, 2, -1)
 Weights: (3, 2, 2, 0)
 Weights: (3, 2, 2, 1)
 Weights: (3, 2, 3, -3)
 Weights: (3, 2, 3, -2)
 Weights: (3, 2, 3, -1)
 Weights: (3, 2, 3, 0)
 Weights: (3, 3, -3, -3)
 Weights: (3, 3, -3, -2)
 Weights: (3, 3, -3, -1)
 Weights: (3, 3, -3, 0)
 Weights: (3, 3, -3, 1)
 Weights: (3, 3, -2, -3)
 Weights: (3, 3, -2, -2)
 Weights: (3, 3, -2, -1)
 Weights: (3, 3, -2, 0)
 Weights: (3, 3, -2, 1)
 Weights: (3, 3, -1, -3)
 Weights: (3, 3, -1, -2)
 Weights: (3, 3, -1, -1)
 Weights: (3, 3, -1, 0)
 Weights: (3, 3, -1, 1)
 Weights: (3, 3, 0, -3)
 Weights: (3, 3, 0, -2)
 Weights: (3, 3, 0, -1)
 Weights: (3, 3, 0, 0)
 Weights: (3, 3, 0, 1)
 Weights: (3, 3, 1, -3)
 Weights: (3, 3, 1, -2)
 Weights: (3, 3, 1, -1)
 Weights: (3, 3, 1, 0)
 Weights: (3, 3, 1, 1)
 Weights: (3, 3, 2, -3)
 Weights: (3, 3, 2, -2)
 Weights: (3, 3, 2, -1)
 Weights: (3, 3, 2, 0)
 Weights: (3, 3, 2, 1)
 Weights: (3, 3, 3, -3)
 Weights: (3, 3, 3, -2)

Weights: (3, 3, 3, -1)
 Weights: (3, 3, 3, 0)
 Weights: (3, 3, 3, 3)
 Emulated function: 1,0,0,0
 Weights: (-3, -3, -3, 1)
 Weights: (-3, -2, -1, 3)
 Weights: (-2, -3, -1, 3)
 Weights: (-2, -2, -2, 1)
 Weights: (-2, -1, 0, 3)
 Weights: (-1, -2, 0, 3)
 Weights: (-1, -1, -1, 1)
 Weights: (-1, 0, 1, 3)
 Weights: (0, -1, 1, 3)
 Weights: (0, 0, 0, 1)
 Weights: (0, 1, 2, 3)
 Weights: (1, 0, 2, 3)
 Weights: (1, 1, 1, 1)
 Weights: (1, 2, 3, 3)
 Weights: (2, 1, 3, 3)
 Weights: (2, 2, 2, 1)
 Weights: (3, 3, 3, 1)
 Emulated function: 0,1,0,0
 Weights: (-3, -2, -2, 1)
 Weights: (-3, -2, 0, 2)
 Weights: (-3, -2, 1, 2)
 Weights: (-3, -2, 2, 2)
 Weights: (-3, -2, 3, 2)
 Weights: (-2, -3, -2, 3)
 Weights: (-2, -1, -3, 2)
 Weights: (-2, -1, -1, 1)
 Weights: (-2, -1, 1, 2)
 Weights: (-2, -1, 2, 2)
 Weights: (-2, -1, 3, 2)
 Weights: (-1, -2, -1, 3)
 Weights: (-1, 0, -3, 2)
 Weights: (-1, 0, -2, 2)
 Weights: (-1, 0, 0, 1)
 Weights: (-1, 0, 2, 2)
 Weights: (-1, 0, 3, 2)
 Weights: (0, -1, 0, 3)
 Weights: (0, 1, -3, 2)
 Weights: (0, 1, -2, 2)
 Weights: (0, 1, -1, 2)
 Weights: (0, 1, 1, 1)
 Weights: (0, 1, 3, 2)
 Weights: (1, 0, 1, 3)
 Weights: (1, 2, -3, 2)
 Weights: (1, 2, -2, 2)
 Weights: (1, 2, -1, 2)
 Weights: (1, 2, 0, 2)
 Weights: (1, 2, 2, 1)
 Weights: (2, 1, 2, 3)
 Weights: (2, 3, -3, 2)
 Weights: (2, 3, -2, 2)
 Weights: (2, 3, -1, 2)
 Weights: (2, 3, 0, 2)
 Weights: (2, 3, 1, 2)
 Weights: (2, 3, 3, 1)
 Weights: (3, 2, 3, 3)
 Emulated function: 1,1,0,0
 Weights: (-3, -1, -1, 2)
 Weights: (-3, -1, 0, 3)
 Weights: (-3, 0, 0, 2)
 Weights: (-3, 0, 1, 3)
 Weights: (-3, 1, 1, 2)
 Weights: (-3, 1, 2, 3)
 Weights: (-3, 2, 2, 2)
 Weights: (-3, 2, 3, 3)
 Weights: (-3, 3, 3, 2)
 Weights: (-2, 0, 0, 2)
 Weights: (-2, 0, 1, 3)
 Weights: (-2, 1, 1, 2)
 Weights: (-2, 1, 2, 3)
 Weights: (-2, 2, 2, 2)
 Weights: (-2, 2, 3, 3)
 Weights: (-2, 3, 3, 2)
 Weights: (-1, -3, -3, 2)
 Weights: (-1, -3, -2, 3)
 Weights: (-1, 1, 1, 2)
 Weights: (-1, 1, 2, 3)

Weights: (0, -3, 3, 3)
 Weights: (0, -2, -3, 3)
 Weights: (0, -2, 2, 3)
 Weights: (0, -2, 3, 3)
 Weights: (0, 2, -3, 3)
 Weights: (0, 2, -2, 3)
 Weights: (0, 2, -1, 3)
 Weights: (0, 3, -3, 3)
 Weights: (0, 3, -2, 3)
 Weights: (0, 3, -1, 3)
 Weights: (0, 3, 2, 3)
 Weights: (1, -3, -1, 3)
 Weights: (1, -3, 0, 3)
 Weights: (1, -3, 3, 3)
 Weights: (1, -2, -3, 3)
 Weights: (1, -2, 0, 3)
 Weights: (1, -2, 3, 3)
 Weights: (1, -1, -3, 3)
 Weights: (1, -1, -2, 3)
 Weights: (1, -1, 3, 3)

Weights: (1, 3, -3, 3)
 Weights: (1, 3, -2, 3)
 Weights: (1, 3, -1, 3)
 Weights: (1, 3, 0, 3)
 Weights: (2, -3, -1, 3)
 Weights: (2, -3, 0, 3)
 Weights: (2, -3, 1, 3)
 Weights: (2, -2, -3, 3)
 Weights: (2, -2, 0, 3)
 Weights: (2, -2, 1, 3)
 Weights: (2, -1, -3, 3)
 Weights: (2, -1, -2, 3)
 Weights: (2, -1, 1, 3)
 Weights: (2, 0, -3, 3)
 Weights: (2, 0, -2, 3)
 Weights: (2, 0, -1, 3)
 Weights: (3, -3, -1, 3)
 Weights: (3, -3, 0, 3)
 Weights: (3, -3, 1, 3)
 Weights: (3, -3, 2, 3)

Weights: (3, -2, -3, 3)
 Weights: (3, -2, 0, 3)
 Weights: (3, -2, 1, 3)
 Weights: (3, -2, 2, 3)
 Weights: (3, -1, -3, 3)
 Weights: (3, -1, -2, 3)
 Weights: (3, -1, 1, 3)
 Weights: (3, -1, 2, 3)
 Weights: (3, 0, -3, 3)
 Weights: (3, 0, -2, 3)
 Weights: (3, 0, -1, 3)
 Weights: (3, 0, 2, 3)
 Weights: (3, 1, -3, 3)
 Weights: (3, 1, -2, 3)
 Weights: (3, 1, -1, 3)
 Weights: (3, 1, 0, 3)

ДОДАТОК В

Програма розрахунку та графічного представлення апроксимацій
індикаторної функції

```

import numpy as np
import matplotlib.pyplot as plt

# Визначаємо функції апроксимації
def gaussian(x, k=10):
    return np.exp(-k * x ** 2)

def lorentzian(x, k=10):
    return 1 / (1 + k * x ** 2)

def hyperbolic_secant(x, k=10):
    return 2 / (np.exp(k * x) + np.exp(-k * x))

def laplacian(x, k=10):
    return np.exp(-k * np.abs(x))

# Визначаємо діапазон x
x = np.linspace(-1, 1, 2000)

# Обчислюємо значення y для кожної функції
y_gaussian = gaussian(x)
y_lorentzian = lorentzian(x)
y_hyperbolic_secant = hyperbolic_secant(x)
y_laplacian = laplacian(x)

# Побудова графіків
plt.figure(figsize=(12, 8))

# Додаємо кожну функцію до графіку
plt.plot(x, y_gaussian, label='Gaussian  $e^{-kx^2}$ ', color='black')
plt.plot(x, y_lorentzian, label='Lorentzian  $1 / (1 + kx^2)$ ', color='blue')
plt.plot(x, y_hyperbolic_secant, label='Hyperbolic Secant  $2 / (e^{kx} + e^{-kx})$ ', color='red')
plt.plot(x, y_laplacian, label='Laplacian  $e^{-k|x|}$ ', color='green')

# Додаткові налаштування графіка
plt.title('Функції, що апроксимують індикаторну функцію  $\mathbf{1}_0(x)$ ')
plt.xlabel('x')
plt.ylabel('y')
plt.axhline(y=0, color='gray', linestyle='--', linewidth=0.5)
plt.axvline(x=0, color='gray', linestyle='--', linewidth=0.5)
plt.grid(True)
plt.legend(loc='lower center', ncol=2, bbox_to_anchor=(0.5, -0.2))
plt.subplots_adjust(bottom=0.15)
plt.show()

```

ДОДАТОК Г

Список публікацій здобувача

1. Мельничук С. І., Яковин С. В. Ідентифікація об'єктів (сигналів) за оцінками інформаційної ентропії їх двомірних монохромних зображень. Науковий вісник Національного гірничого університету. №3 (147) 2015. – с.131-137. ISSN: 2223-2362, 2071-2227 (**Видання індексується в базі Scopus**).
2. Спосіб реалізації перцептрона на основі імовірнісних характеристик зміщених синаптичних сигналів: пат. 126753 Україна: G06N 20/00. / С. І. Мельничук, С. В. Яковин. – № а2019 01301; заявл. 08.02.2019; опубл. 25.01.2023, Бюл. №4. 3 с. <https://sis.nipo.gov.ua/uk/search/detail/1719578/> (**Патент**).
3. Яковин С. В., Мельничук С. І. Імплементация булевых функций в перцептронных структурах. Вісник Вінницького політехнічного інституту. 2024. Т. 176, №5. С.48–55. <https://doi.org/10.31649/1997-9266-2024-176-5-48-55> (**Фахове видання України**).
4. Yakovyn S. V., Melnychuk S. I. Discrete perceptron based on probabilistic estimates of shifted synaptic signals. Naukovyi Visnyk Natsionalnoho Hirnychoho Universytetu. 2025. No.2. P. 189–196. <https://doi.org/10.33271/nvngu/2025-2/189> (**Видання індексується в базі Scopus**).
5. Яковин С. В., Мельничук С. І. Обмеження множини можливих рішень для дискретного перцептрона зі зміщеними синаптичними сигналами при емуляції бінарних логічних функцій. Вісник Хмельницького національного університету, №3.1, 2025 (351), – 295–301с. <https://doi.org/10.31891/2307-5732-2025-351-37> (**Фахове видання України**).
6. Яковин С. В., Мельничук С. І. Апроксимація індикаторних функцій у контексті ймовірнісного дискретного перцептрона. Вісник Хмельницького національного університету, №3.2, 2025 (353), – 357–364с. <https://doi.org/10.31891/2307-5732-2025-353-50> (**Фахове видання України**).
7. Яковин С. В., Мельничук С. І., Мануляк І.З. Реалізація двовходового дискретного перцептрону зі зміщеними синаптичними сигналами на ПЛІС

засобами AlteraHDL. Вісник Вінницького політехнічного інституту, №4, 2025, – 186–194с. <https://doi.org/10.31649/1997-9266-2025-181-4-186-194> (*Фахове видання України*).

8. Melnychuk S., Kuz M., Yakovyn S. Emulation of logical functions NOT, AND, OR, and XOR with a perceptron implemented using an information entropy function. Proceedings of 14th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET), Lviv-Slavske, 20–24 February 2018. <https://doi.org/10.1109/tcset.2018.8336337> (*Тези конференції, що індексується в базі Scopus*).

9. Yakovyn S. V., Melnychuk S. I. Issues with looking semantically equivalent programming statements in java and ruby. Proceedings III International Scientific and Practical Conference Applied scientific and technical research, 3-5 April 2019. – Ivano-Frankivsk - "Ukraine Technical Sciences Academy" – 2019. – p.47. (*Тези конференції*).

10. Яковин С. В., Мельничук С. І. Особливості реалізації та функціональні обмеження перцептронних структур. Проблемно-наукова міжгалузева конференція "Інформаційні проблеми комп'ютерних систем, юриспруденції, енергетики, моделювання та управління". (ICSM-2023) – Надвірна, 20-21 листопада, 2023. С. 184–187. (*Тези конференції*).

11. Яковин С. В., Мельничук С. І., Мануляк І. З. Алгоритмічні реалізації дискретного перцептронного зі зміщеними синаптичними сигналами. Проблемно-наукова міжнародна конференція "Інформаційні проблеми комп'ютерних систем, енергетики, моделювання та управління". (ICSM-2025) – Надвірна, 5-6 червня, 2025. С. 110–113. (*Тези конференції*).

12. Яковин С. В., Мельничук С. І. Порівняльне оцінювання обчислювальної складності реалізації кореляційного та дискретного перцептронів. Інформаційні технології в освіті, техніці та промисловості: Матеріали науково-практ. конф., м. Івано-Франківськ, 9 жовт. 2025 р. 2025. С. 317–319. (*Тези конференції*).