УДК 681.3:007.52

Р.О. ВДОВИЧЕНКО
Інститут кібернетики ім. В.М. Глушкова НАН України, Київ, Україна,
e-mail: *ruslan.vdovichenko1@gmail.com*.

В.Г. ТУЛЬЧИНСЬКИЙ
Інститут кібернетики ім. В.М. Глушкова НАН України, Київ, Україна,
e-mail: *dep145@gmail.com*.

ПІДВИЩЕННЯ ЩІЛЬНОСТІ ЗБЕРЕЖЕННЯ СЕМАНТИКИ В РОЗРІДЖЕНО-РОЗПОДІЛЕНІЙ ПАМ'ЯТІ

Анотація. Запропоновано інтеграцію методу стискувальних вимірювань у реалізацію розріджено-розподіленої пам'яті для підвищення ємності збереження бінарних розріджених розподілених представлень семантичних даних, зокрема на графічних процесорах.

Ключові слова: розріджено-розподілена пам'ять, стискувальні вимірювання, асоціативна пам'ять, збереження семантики, нейронні мережі, графічний процесор.

ВСТУП

Нинішній стрибок у застосуванні методів і систем штучного інтелекту спричинений можливістю їхньої високопродуктивної реалізації з використанням графічних процесорів (Graphics Processing Unit, GPU) завдяки локальності операцій з даними і однорідності алгоритмів (fine-grained parallelism). У цьому контексті перспективною є запропонована П. Канервою ще у 1986 р. модель розріджено-розподіленої пам'яті (Sparse Distributed Memory, SDM), яка має багато характеристик та функцій природної пам'яті: асоціативність, здатність до узагальнення, запам'ятовування послідовностей і навіть здатність помилятися. Хоча П. Канерва, розробляючи SDM у межах феноменологічного підходу до моделювання, використав засоби небіологічного характеру (бінарні вектори, лічильники, адреси), подальші дослідження показали близькість будови біологічних механізмів пам'яті до конструкції SDM. Зокрема, у термінах SDM можна добре описати функціонування мозочка ссавців, модель якого незалежно розробили Д. Марр у 1969 р. та Дж. Альбус у 1975 р., та роботу пам'яті імунної системи (Д. Сміт, С. Форрест та А. Перелсон, 1996 р.).

Розглянемо застосування SDM для зберігання векторів ознак у контексті опису об'єктів через їхні атрибути, реалізації вкладених структур та зв'язків, необхідних на концептуальному рівні для представлення семантики. Особливістю векторів ознак є суттєва різниця у частоті появи 0 та 1, бо у реальному світі майже будь-яка ознака є лише у невеликої підмножини всіх об'єктів. У цьому випадку вектори ознак є розрідженими. Класичні конструкції SDM погано пристосовані для зберігання розріджених векторів, тому метою цієї статті є адаптація SDM до таких даних. Основною ідеєю нашого підходу є обчислення та зберігання щільних векторів замість розріджених з подальшою реконструкцією розріджених векторів ознак за зчитаними з SDM даними з використанням методів стискувальних вимірювань (Compressive Sensing, CS) — теорії, основи якої заклали у 2004–2006 рр. Е. Канде і Т. Тао. Іншими словами, у цій роботі запропоновано гібридну модель CS SDM.

Наприкінці 1990-х років науковці взялися досліджувати застосування SDM для представлення структурованих даних, придатних для опису семантичних конструкцій. Ці дослідження цікаві тим, що пов'язують низькорівневу нейромережеву модель штучного інтелекту з концептуальними високорівневими моделями даних. У контексті CS SDM інтерес становлять насамперед бінарні розріджені розподілені представлення (Binary Sparse Distributed Representations, BSDR).

Розглянемо коротко основи SDM, CS і BSDR.

Розріджено-розподілена пам'ять (SDM) Канерви [1–3] перевершує раніше запропоновані нейромережеві моделі пам'яті, а саме: мережі Хопфілда [4] — за ємністю, машини Больцмана [5, 6] — за швидкодією.

Пам'ять SDM складається з N комірок, що зберігають M-вимірні цілочисельні вектори. Отже, першою відмінністю SDM від звичайної пам'яті комп'ютера є те, що одному біту записаних даних під час зберігання відповідає не один біт пам'яті, а ціле число. Пам'ять адресується L-розрядними двійковими адресами, причому $2^L \gg N$. Іншими словами, другою відмінністю від звичайної пам'яті комп'ютера є те, що кількість фізичних комірок пам'яті є набагато меншою ніж кількість можливих адрес. Саме тому пам'ять є розрідженою. Дані, що записуються і зчитуються, є M-розрядними двійковими числами, так само як у звичайній пам'яті комп'ютера.

У класичній конструкції Канерви кожній з N фізичних комірок відповідає L-розрядна двійкова адреса. Зазвичай адреси комірок генеруються у випадковий спосіб, хоча теоретично немає заборони на регулярне розміщення комірок в адресному просторі. Через те, що пам'ять є розрідженою, випадкові адреси комірок потрібно зберігати, як і дані. У цьому полягає третя відмінність SDM від звичайної пам'яті комп'ютера.

Відрізняються від звичайної пам'яті також способи запису і зчитування даних у SDM. Запис здійснюється не в одну комірку, а відразу в декілька активованих комірок. Комірка, що має номер n і адресу $\mathbf{a}_n$, активується для зчитування/запису на адресу $\mathbf{a}$ за умови, що відстань Хеммінга, тобто кількість розбіжних бітів між вхідною адресою та адресою комірки, не перевищує заданого порога d:

$$n \in A(\mathbf{a}) \Leftrightarrow || \mathbf{a}_n - \mathbf{a} ||_0 \le d. \tag{1}$$

Тут і далі, якщо комірку з номером n і значеннями $\mathbf{u}_n = \langle u_1^n u_2^n u_3^n \dots u_M^n \rangle$, $u_i^n \in \mathrm{Z}$, активовано адресою $\mathbf{a}$, то $n \in A(\mathbf{a})$; $|| \mathbf{x} ||_P = \left(\sum |x_i|^P \right)^{1/P}$ означає норму ℓ_P для $1 \le P \le \infty$ і квазінорму для $0 < P < 1$. Це поняття узагальнюється на $P = \infty$: $|| \mathbf{x} ||_\infty = \max (|x_i|)$ і $P = 0$: $|| \mathbf{x} ||_0 = \sum |x_i|^0$, тобто $|| \mathbf{x} ||_0$ — кількість ненульових коефіцієнтів.

Під час запису в активовану комірку нові дані не затирають попереднього вмісту цієї комірки, а додаються до нього за таким правилом:

$$u_i^n (t+1) = \begin{cases} u_i^n (t) + 1, & \text{якщо } v_i = 1, \\ u_i^n (t) - 1, & \text{якщо } v_i = 0. \end{cases} \tag{2}$$

Отже, SDM накопичує всі записані дані в лічильниках, які замінюють окремі біти звичайної пам'яті комп'ютера.

Зчитування з SDM теж здійснюється не з однієї комірки, а одразу з групи активованих комірок. Спочатку для кожного розряду обчислюється сума значень відповідних лічильників усіх активованих комірок, а потім відповідний біт результату зчитування $\mathbf{v} = \langle v_1 v_2 v_3 \dots v_M \rangle$ встановлюється в 1, якщо ця сума перевищує поріг T_i, або в 0 в іншому випадку:

$$v_i = \begin{cases} 1, & \text{якщо } \sum\limits_{n \in \mathrm{A}(\mathbf{a})} u_i^n > T_i, \\ 0, & \text{якщо } \sum\limits_{n \in \mathrm{A}(\mathbf{a})} u_i^n \le T_i. \end{cases} \tag{3}$$

У тому разі, коли ймовірність значень 0 та 1 є приблизно однаковою, використовують $T_i = 0$.

Кількома роками пізніше Л. Джекел запропонував гіперплощинну конструкцію SDM [7, 8]. У його моделі замість L-розрядної бінарної адреси кожна

фізична комірка пам'яті (номер n) асоціюється з короткою активаційною маскою (довжиною $K \ll L$), кожен елемент якої складається з номера і значення одного біта адреси: $\{\langle b_1^n, v_1^n\rangle, \langle b_2^n, v_2^n\rangle, \ldots, \langle b_K^n, v_K^n\rangle\}$, $b_i^n \in \{1 \ldots L\}$, $b_i^n \neq b_j^n$, $v_i^n \in \{0,1\}$. Умовою активації в конструкції Джекела є відповідність усіх значень бітів в адресі та в масці:

$$n \in A(\mathbf{a}) \Leftrightarrow a_{b_i^n} = v_i^n \ \ \forall i \in \{1 \ldots K\}. \tag{4}$$

Перевагами конструкції Джекела є менші витрати пам'яті та більша продуктивність завдяки меншій кількості порівнянь. Конструкція Джекела краще відповідає моделі мозочка Марра–Альбуса [9], проте імунна пам'ять, згідно з моделлю Сміта–Форреста–Перелсона [10], більше схожа на початкову конструкцію Канерви. Іншими словами, обидва підходи задіяні у живих організмах. Незалежно від конструкції, SDM можна розглядати як прямоточну неповнозв'язну нейроподібну мережу, що має 2^L основних входів, один прихований шар з N L-розрядних блоків і M вихідних вузлів.

Стискувальні вимірювання. У межах методу CS запропоновано новий погляд на бажані умови відновлення сигналу за непрямими вимірюваннями. В основу CS покладено таку лінійну модель.

Нехай сигнал $y \in \mathrm{R}^N$ — це вектор виміряних дійсних значень, а $\mathbf{\Phi}$ — матриця моделі вимірювача. Потрібно знайти значення невідомого вектора (функції) $\mathbf{f}_0$ за умовою

$$\mathbf{y} = \mathbf{\Phi}\mathbf{f}_0. \tag{5}$$

Система рівнянь (5) є недовизначеною. Припустимо, що у $\mathbf{f}_0$ є розріджене представлення у вигляді вектора дійсних чисел $\mathbf{x}_0 \in \mathrm{R}^N$ в деякому відомому домені $\mathbf{\Psi}$: $\mathbf{f}_0 = \mathbf{\Psi}\mathbf{x}_0$ (наприклад, його дискретне перетворення Фур'є складається всього з кількох гармонік). У цьому випадку рівняння (5) можна записати у такому вигляді:

$$\mathbf{y} = \mathbf{\Lambda}\mathbf{x}_0, \ \mathbf{\Lambda} = \mathbf{\Phi}\mathbf{\Psi}. \tag{6}$$

Припустимо, не зменшуючи загальності міркувань, що N стовпців матриці $\mathbf{\Lambda} \in \mathrm{R}^{n \times N}$ мають одиничну евклідову норму $||\,\boldsymbol{\lambda}_j\,||_2 = 1$. Цю матрицю називають словником, а її стовпці $\boldsymbol{\lambda}_j$ — зразками у словнику.

Вектор $\mathbf{x}$ називається s-розрідженим, якщо кількість його ненульових компонент $||\,\mathbf{x}\,||_0 \le s$. Виходячи з (6), назвемо $\mathbf{x}_0$ представленням сигналу $\mathbf{y}$ у словнику $\mathbf{\Lambda}$.

Розрідженість $\mathbf{x}_0$ можна використовувати, щоб розв'язувати задачу (6) як оптимізаційну: знайти найрозрідженіший розв'язок, тобто підібрати

$$\widetilde{\mathbf{f}} = \mathbf{\Psi} \cdot \arg\min_{\mathbf{x}} ||\,\mathbf{x}\,||_1 \ \text{ за умовою } \ \mathbf{y} = \mathbf{\Lambda}\mathbf{x}. \tag{7}$$

Тут тильда позначає оцінку. Можливість використовувати $||\,\mathbf{x}\,||_1$ замість $||\,\mathbf{x}\,||_0$ є одним з важливих факторів ефективності CS [11], інакше задача була б NP-повною [12].

Більш практична задача відновлення s-стиснених сигналів є набагато складнішою ніж розглянута задача відновлення s-розріджених сигналів. Щоб її розв'язати, треба послабити умови (6), припустивши, що гранична похибка $\varepsilon \ge 0$. У цьому випадку стиснений сигнал можна шукати як зашумлений розріджений сигнал:

$$\widetilde{\mathbf{f}} = \mathbf{\Psi} \cdot \arg\min_{\mathbf{x}} ||\,\mathbf{x}\,||_1 \ \text{ за умовою } \ ||\,\mathbf{\Lambda}\mathbf{x} - \mathbf{y}\,||_2 \le \varepsilon. \tag{8}$$

Хоча здебільшого застосовують формулювання (8), його елементи можна поєднати і в інші способи. Наприклад, можна шукати мінімальну похибку для заданого рівня розрідження s:

$$\widetilde{\mathbf{f}} = \mathbf{\Psi} \cdot \arg\min_{\mathbf{x}} ||\,\mathbf{\Lambda}\mathbf{x} - \mathbf{y}\,||_2 \ \text{ за умовою } \ ||\,\mathbf{x}\,||_0 \le s. \tag{9}$$

До того ж, можна за допомогою параметра регуляризації $\alpha \geq 0$ балансувати похибку та розрідження:

$$\widetilde{\mathbf{f}} = \mathbf{\Psi} \cdot \arg\min_{\mathbf{x}} (|| \mathbf{\Lambda x} - \mathbf{y} ||_2 + \alpha || \mathbf{x} ||_0). \quad (10)$$

Для дослідження задач вигляду (8)–(10) у CS розроблено цілу низку теоретичних підходів і алгоритмів. Зупинимося на двох — найзагальнішій достатній умові та найпростішому методі розв'язування таких задач.

Відомо, що Е. Канде, Дж. Ромберг і Т. Тао ввели поняття умови обмеженої ізометрії (Restricted Isometry Property, RIP) та довели, що RIP гарантує існування стійких алгоритмів для ідентифікації розріджених сигналів на основі стискання їхніх околів [13]. Неформальне визначення умови *s*-RIP полягає в тому, що будь-яка множина з *s* зразків словника містить приблизно ортогональні вектори. Усі колонки матриці не можуть бути взаємно ортогональними, тому що їхня кількість більша ніж кількість рядків.

Зазначимо, що безпосередньо перевірити умову RIP для заданої матриці важко. Щоб оминути етап безпосередньої перевірки RIP, широко використовують випадкову матрицю $\mathbf{\Lambda}$, оскільки випадкові матриці є некогерентними для будь-якого фіксованого базису. Відповідно до леми Джонсона–Лінденштрауса [14], випадкова матриця, породжена з певним розподілом ймовірності, зберігає норму будь-якого *s*-розрідженого вхідного сигналу всередині невеликого околу. Є багато конструктивних розподілів ймовірності, зокрема нормальний та розподіл Бернуллі, що задовольняють умови леми Джонсона–Лінденштрауса.

Бінарні розріджені розподілені представлення. Великий інтерес у межах феноменологічного підходу до дослідження будови пам'яті становить представлення в нейромережевих моделях даних, що мають внутрішню структуру (ієрархічну, типу «ключ–значення», символьні послідовності тощо). Відповідний напрям досліджень зазвичай пов'язують зі сформованою на рубежі 2000-х концепцією векторно-символьної архітектури (Vector Symbolic Architecture, VSA) [15]. Найважливіші результати VSA для дійсних векторів отримані Дж. Фодором і З. Пилишиним [16], П. Смоленським [17], Т. Плейтом [18], а для бінарних векторів — П. Канервою [19, 20], Г. Сйодіном [21], Р. Ґайлером [22], Д. Рачковським [23, 24] та їхніми колегами. Термін VSA запропоновано у 2003 р. Р. Ґайлером [25]. Особливий інтерес до розрідженого представлення спричинений його відповідністю характеру активності нейронів [26]. Методи перетворення структур у бінарні розріджені кодвектори на сьогодні добре розвинені і для числових векторів з дійсними компонентами, і для ієрархічних даних [27]. У контексті CS SDM інтерес становить насамперед BSDR.

Основними операціями VSA є зв'язування, $\otimes$ (binding), та відв'язування, $\oslash$ (unbinding). Зв'язування векторів, наприклад, пари «ключ–значення», створює новий вектор: $\mathbf{c} = \mathbf{a} \otimes \mathbf{b}$. Відв'язування дає змогу отримати один із зв'язаних векторів за результатом та іншим ($\mathbf{b} = \mathbf{c} \oslash \mathbf{a}$, $\mathbf{a} = \mathbf{c} \oslash \mathbf{b}$), або за результатом та номером позиції у зв'язку ($\mathbf{a} = {}_1\oslash \mathbf{c}$, $\mathbf{b} = {}_2\oslash \mathbf{c}$). Не всі варіанти VSA підтримують обидва типи відв'язування. У деяких VSA зв'язування доповнює інша операція композиції, а саме злиття, + (merging). Це дає змогу простіше описувати об'єкти з атрибутами: $\mathbf{o} = \mathbf{c} + \mathbf{a}_1 \otimes \mathbf{v}_1 + \mathbf{a}_2 \otimes \mathbf{v}_2 + \ldots$. Для аналізу композицій зазвичай використовують операцію випробування (probing). Це перевірка того, що вектор у зв'язаному стані входить до цієї композиції. Виконують випробування шляхом відв'язування (наприклад, $\mathbf{b} = \mathbf{o} \oslash \mathbf{v}_1$) та аналізу результату на «розумність». Важливим елементом VSA є асоціативна пам'ять, що «очищає» результат відв'язування від шуму для точного відновлення початково зв'язаного вектора.

Історично першим сформулював модель типу BSDR, мабуть, Г. Сйодін у вигляді Sparchunk Code [21], хоча пізніше створена модель APNN/CDT Д. Рачковського [23–24] спирається на розробки групи науковців під керівництвом Е.М. Куссуля [28, 29], отримані значно раніше. Обидві моделі мають спільні

елементи: зв'язування через диз'юнкцію з наступною заміною на нулі частини одиниць (нормалізація, thinning) задля збереження постійного рівня розрідженості. У моделі Г. Сйодіна під час зв'язування компоненти циклічно зсуваються на різні кроки (відповідно до номера позиції у зв'язку), потім виконується нормалізація їхньої диз'юнкції, причому відбір бітів для підсумкового кодвектора визначається фіксованою (випадково сформованою) функцією. У моделі Рачковського розглянуто кілька варіантів зв'язування, зокрема, на основі циклічного зсуву (як у моделі Сйодіна) та на основі псевдовипадкових перестановок окремих бітів (з ініціалізацією номером позиції). Головною ідеєю моделі Рачковського є контекстно-залежна нормалізація (Context-Dependent Thinning, CDT), тобто застосування маски, яка накопичується як диз'юнкція статистично достатньої кількості перемішувань кодвектора (за фіксованим випадковим законом). Відв'язування та випробування в обох BSDR ґрунтується на зчитуванні з очищувальної пам'яті (у разі потреби — після зворотного перетворення, наприклад, циклічного зсуву у протилежному напрямку). У моделі APNN/CDT кодвектор зберігає подібність до інших кодвекторів, складених зі схожого набору компонент, що підвищує стійкість до помилок. Зазначимо, що мотивацією, що спонукала Сйодіна до розроблення Sparchunk Code, було якраз застосування SDM як очищувальної пам'яті.

Набагато пізніше було запропоновано BSDR на основі сегментованих випадкових кодвекторів [30], де в кожному сегменті гарантується наявність єдиної ознаки, а зв'язування полягає у циклічному зсуві ознак одного кодвектора на величину, яка визначається позицією ознаки у відповідному сегменті парного кодвектора. Іноді ефективність цієї моделі є більшою, а іноді — меншою [15].

Для подальшого викладу слід зауважити, що всі зазначені моделі BSDR оперують розрідженими векторами з постійною (фіксованою) кількістю одиниць. Іншими словами, очищувальна пам'ять має зберігати і ефективно відновлювати за спотвореною копією саме ці вектори.

ОПИС І ВИПРОБУВАННЯ МОДЕЛІ CS-SDM

Конструкція CS SDM. Розглянемо таку конструкцію розріджено-розподіленої пам'яті для роботи з розрідженими бінарними векторами. Пам'ять CS SDM складається з трьох блоків: кодера, власне блока пам'яті (SDM) та декодера. Кодер перетворює вхідні дані у зручну форму для зберігання у блоці пам'яті, декодер перетворює зчитані дані знову у розріджену форму. Подібні архітектури було запропоновано і раніше, наприклад, у [31]. Новизна нашого підходу полягає в інтеграції блока пам'яті на основі SDM та декодера на основі CS.

У кодері M-розрядні розріджені вектори даних перетворюються на m-розрядні щільні цілочисельні вектори $\mathbf{v} = \mathbf{\Lambda x}$. У наших експериментах використано довжину щільних векторів, $m = ks$, далі $k \in \{8, 12\}$.

Словник $\mathbf{\Lambda} \in \{\pm 1\}^{m \times M}$ заповнюється рівномірно розподіленими псевдовипадковими числами зі значеннями +1 або −1 і зберігається у звичайній пам'яті. Ця матриця $\mathbf{\Lambda}$ відповідає умовам нормування зразків і водночас є зручною для отримання цілих результатів невеликої амплітуди. Нормування відповідно переноситься на етап зчитування. Якщо врахувати нормування, очевидним є те, що у цьому разі забезпечується умова RIP: скалярний добуток будь-яких двох зразків $\mathbf{\Lambda}$ має розподіл Бернуллі з математичним очікуванням 0, його ділення на кількість доданків дає результат, близький до нуля. Інакше кажучи, зразки є приблизно ортогональними.

Як блок пам'яті використано SDM конструкції Джекела з невеликими змінами:

— оскільки вхідні дані є вже не бінарними, а цілочисельними векторами, то правило зміни лічильників у активованих комірках (2) спрощується до простого підсумовування:

$$u_i^n(t+1) = u_i^n(t) + v_i; \tag{11}$$

— для використання у разі зчитування в кожній комірці передбачено додатковий 0-й розряд, до якого під час кожного запису додається 1:

$$u_0^n(t+1) = u_0^n(t) + 1, \tag{12}$$

тобто пам'ять фактично зберігає вектори довжиною $m+1$;

— вихідні дані — теж уже не бінарні, а дійсні числа, тобто (3) змінюється на

$$v_i = \sum_{n \in A(\mathbf{a})} \frac{u_i^n}{u_0^n} \text{ для } i = \{1 \dots m\}. \tag{13}$$

Виникає важливе запитання: які адреси подавати на блок пам'яті? Якщо адреси також перетворювати з розріджених на щільні за допомогою множення $\mathbf{\Lambda x}$ та порівняння з 0, то SDM працює в оптимальному режимі, забезпечуючи мінімальну помилку зчитування. Однак водночас пам'ять втрачає узагальнювальну здатність — виправляти невеликі помилки, бо зміна навіть в одному біті $\mathbf{x}$ призводить до значної зміни $\mathbf{\Lambda x}$ та активації інших комірок. Тому в нашій конструкції адреси залишаються розрідженими. За такої адресації маска перевіряє лише наявність ознаки ($v_i^n = 1$), бо перевірка на 0 для розріджених векторів не є ефективною. Це також заощаджує фізичну пам'ять, оскільки перевірочні значення можна не зберігати.

Слід зазначити, що у разі практичної реалізації CS SDM може знадобитися більша розрядність чисел у лічильниках блока пам'яті, ніж у конструкціях Канерви чи Джекела для аналогічної кількості записів. Але кількість лічильників є меншою і становить m, а не M на фізичну комірку.

Насамкінець результат, прочитаний з блока пам'яті, надсилається у декодер, який у межах підходу CS шукає s-розріджений розв'язок недовизначеної системи лінійних рівнянь. Наш декодер розв'язує цю задачу у постановці (8) з використанням алгоритмів лінійного програмування [32] та «жадібного» алгоритму типу Orthogonal Matching Pursuit [33] — CoSaMP [34], що спеціально пристосований для задач CS. Лінійне програмування загалом дає кращі розв'язки, але потребує набагато більше часу для обчислень, тому його для великих тестів не використовували. (На невеликих тестах ми порівняли відновлення векторів із застосуванням CoSaMP і лінійним програмуванням.)

Моделі для порівняння. Як зразки для порівняння обрано конструкції SDM Канерви та Джекела з мінімальними модифікаціями для роботи з розрідженими векторами. У нашій тестовій конструкції Джекела, як і у CS SDM, маска перевіряє лише наявність властивості ($v_i^n = 1$).

У конструкції Канерви адреси фізичних комірок були згенеровані випадково, але так, щоб кожна адреса мала рівно s одиниць.

Реалізація. Три описані конструкції SDM реалізовано на NVIDIA CUDA за винятком декодера CS SDM, для якого використано готову реалізацію алгоритму CoSaMP [34, 35]. Також випробувано процедуру LinProg із бібліотеки SciPy на Python (для порівняння) [36, 37]. Початкові тексти нашої програми є відкритими [38, 39].

У конструкціях Л. Джекела та CS SDM застосовано довжину маски $K = 4$, підібрану з міркувань середньої кількості активованих фізичних комірок. У конструкції П. Канерви поріг активації підібрано так, щоб для активації теж був потрібний збіг 4 ознак-одиниць: відстань $d = 2s - 4$.

Тестові дані. Для експериментального дослідження було згенеровано набори даних, що складаються з випадкових розріджених бінарних векторів розрядності $M = L = 600$ з різною кількістю одиниць $s = \{12, 16, 20\}$.

Обчислювальні експерименти. Експерименти проведено на графічному прискорювачі GeForce RTX 2080 Ti (архітектура Turing, 11 ГБ пам'яті GDDR6, 4352 CUDA-ядер). Кількість фізичних комірок вибрано так, щоб повністю заповнити пам'ять цього прискорювача. Для CS SDM вона визначалася довжиною векторів, що записуються так: $m = ks$. Тому, залежно від числа s та коефіцієнта k, кількість

фізичних комірок змінювалася від 30 млн до 100 млн. Для двох інших конструкцій у пам'яті вмістилося по 15 млн комірок розрядності $M = 600$.

Процедура експериментального дослідження полягала у записі одних і тих самих даних в усі вказані варіанти SDM. Результати експериментального порівняння ймовірності помилок зчитування у конструкціях для різних модулів векторів s наведено на рис. 1–3.

Роботу моделей перевірено у режимі асоціативної пам'яті: запис та зчитування даних за адресою, що збігається зі значенням. У половині тестів в адресу внесено шум: одну з одиниць змінювали на нуль. Саме доповнення частково відомих множин ознак є головною вимогою до очищувальної пам'яті щодо BSDR.

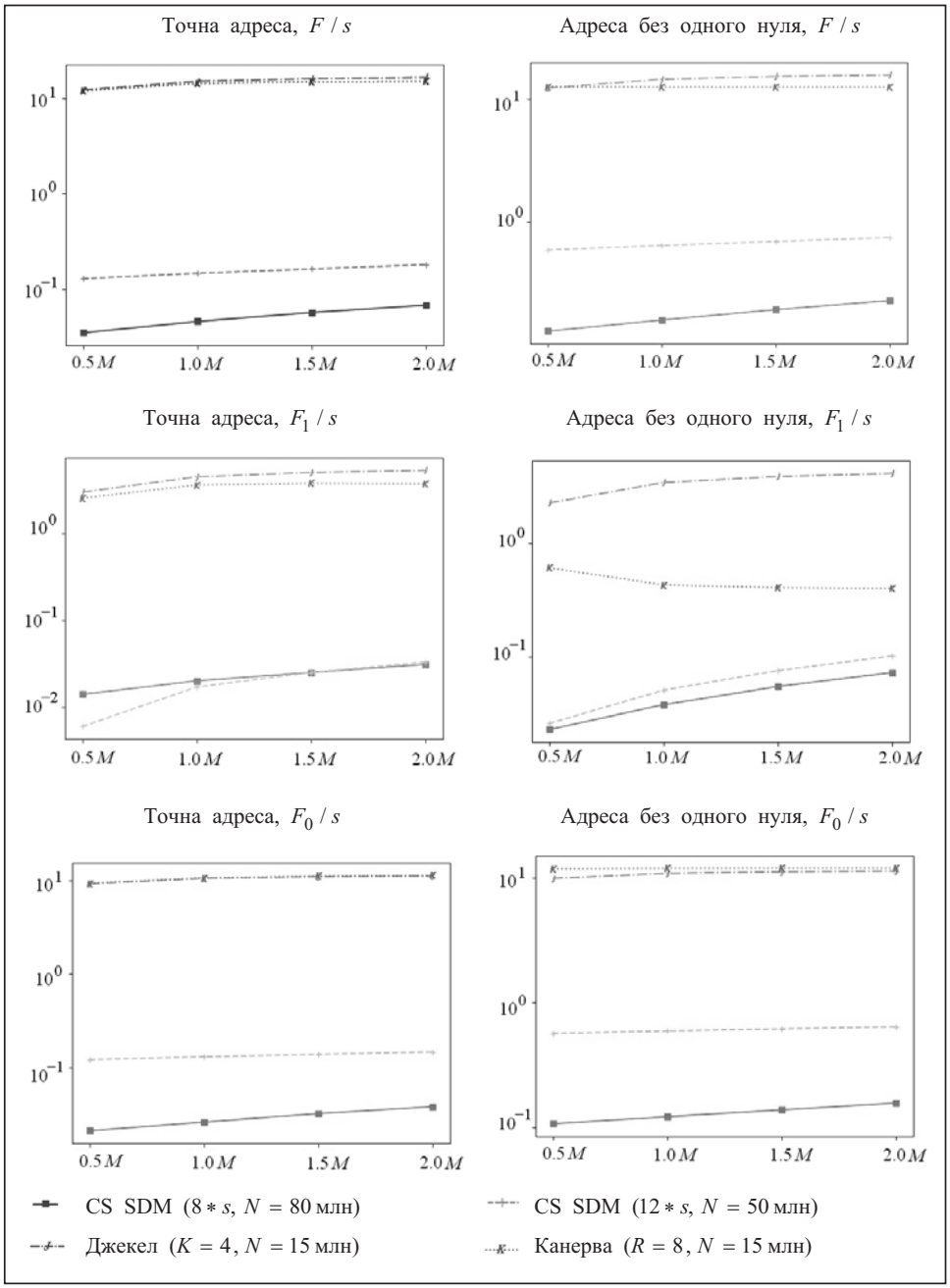

Рис. 1. Середня відносна похибка під час зчитування залежно від кількості записаних векторів з кількістю ознак (одиниць) $s = 12 : F$ — відстань Хеммінга, F_1 — кількість помилкових одиниць, F_0 — кількість помилкових нулів

Рисунки 1–3 свідчать про абсолютну перевагу конструкції CS SDM над конструкціями Канерви і Джекела у разі великої щільності даних як за середньою відстанню Хеммінга між записаним та прочитаним векторами, так і за середньою кількістю помилково негативних та помилково позитивних ознак. Також для коротких векторів ($s = 12$) менший коефіцієнт довжини фізичної комірки CS SDM ($k = 8$) має перевагу, але для довших векторів краще працюють довші комірки. Про аналогічну абсолютну перевагу CS SDM свідчить оцінка відсоткової частки правильно зчитаних векторів (рис. 4).

З рис. 4 видно, що CS SDM стійко забезпечує безпомилкове відновлення майже на 100 %. Натомість конструкціям Канерви і Джекела не вистачає пам'яті і відновлення записаних даних здійснюється переважно з помилками.

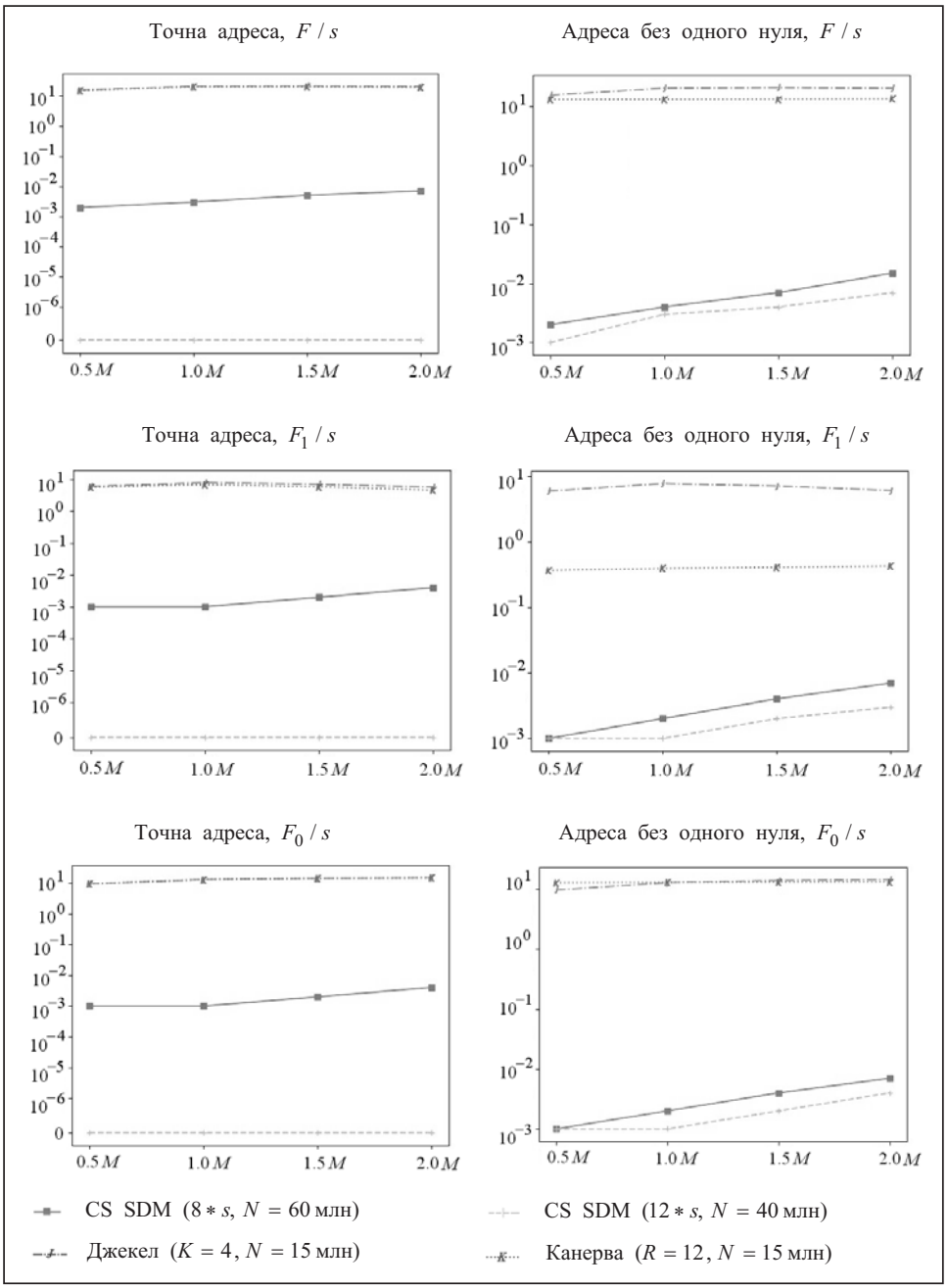

Рис. 2. Середня відносна похибка під час зчитування залежно від кількості записаних векторів з кількістю ознак (одиниць) $s = 16$: F — відстань Хеммінга, F_1 — кількість помилкових 1, F_0 — кількість помилкових 0

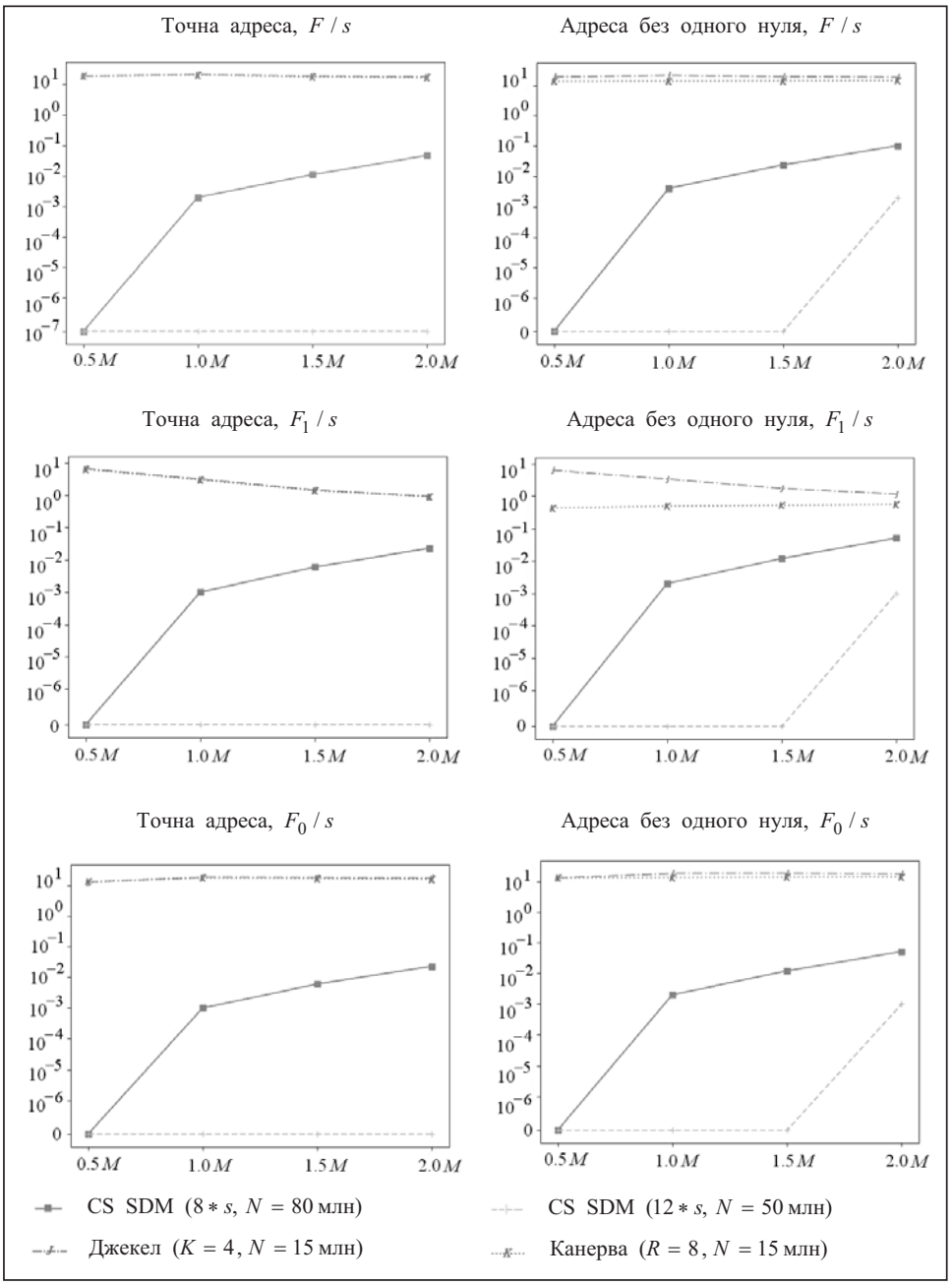

Рис. 3. Середня відносна похибка під час зчитування залежно від кількості записаних векторів з кількістю ознак (одиниць) $s = 20$: F — відстань Хеммінга, F_1 — кількість помилкових 1, F_0 — кількість помилкових 0

В експериментах, результати яких відображено на рис. 1–4, для розв'язування задачі CS використано швидкий «жадібний» алгоритм CoSaMP. З табл. 1–2 видно, наскільки результати можна було б теоретично покращити за допомогою більш якісного алгоритму ℓ_1-мінімізації — лінійного програмування. Через значну тривалість розрахунків ці експерименти проведено для суттєво меншої кількості записаних векторів (100 тис), оскільки процедура LinProg є набагато повільнішою. До того ж, зменшено довжину фізичних комірок ($k = \{6, 8\}$), щоб різниця була більш відчутною.

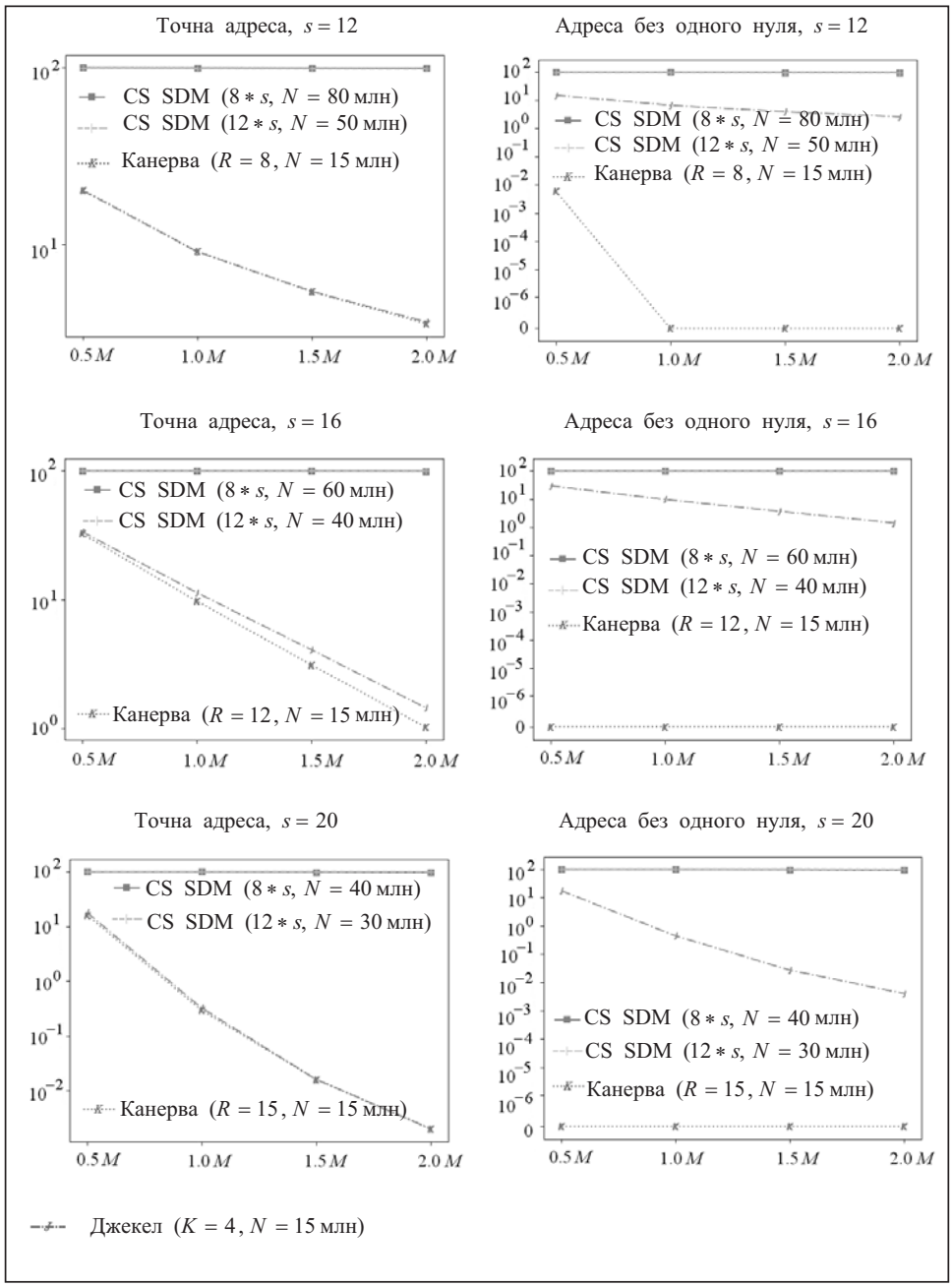

Рис. 4. Відсоткова частка правильно зчитаних векторів

Таблиця 1

Кількість ознак s	Відсоткова частка правильно зчитаних векторів для точних адрес			
	$k = 6$		$k = 8$	
	CoSAMP, %	LinProg, %	CoSAMP, %	LinProg, %
12	71.491	99.834	99.654	99.918
16	87.940	100	99.984	100
20	96.007	100	99.998	100

Таблиця 2

Кількість ознак s	Відсоткова частка правильно зчитаних векторів для адрес з помилкою (один 0 замість 1)			
	$k = 6$		$k = 8$	
	CoSAMP, %	LinProg, %	CoSAMP, %	LinProg, %
12	71.190	99.594	99.966	99.246
16	87.880	100	99.981	100
20	95.951	100	99.998	100

Отже, для коротких векторів, тобто для великого коефіцієнта стиснення m / M, перевага лінійного програмування є значною, але коли внаслідок зростання s і/або k довжина фізичної комірки збільшується та коефіцієнт стиснення зменшується, ця перевага наближається до 0 і використання «жадібного» алгоритму не погіршує результат.

ВИСНОВКИ

Запропоновано нову гібридну модель нейронної пам'яті CS SDM, що ґрунтується на інтеграції класичної моделі розріджено-розподіленої пам'яті (SDM) та сучасних досягнень теорії стискувальних вимірювань (CS). У моделі CS SDM ємність очищувальної пам'яті для бінарного розріджено-розподіленого представлення є суттєво більшою порівняно з більш ранніми конструкціями SDM.

У контексті концептуального моделювання будови пам'яті людини важливо з'ясувати, наскільки запропонована конструкція є близькою або далекою від біологічної реальності. Основні її елементи, а саме кодування за допомогою випадкової матриці (міжнейронних зв'язків) та декодування за допомогою мінімізації (зокрема, «жадібного» алгоритму), здаються достатньо примітивними для того, щоб подібні механізми можна було шукати у живій природі. Це є предметом досліджень фізіологів та нейробіологів. Ще одне питання залишається відкритим — чи можна (та в який спосіб) модифікувати цю модель для того, щоб послабити залежність від однорідності інформації, яка зберігається у пам'яті.

Під час дослідження на платформі NVIDIA CUDA було реалізовано бібліотеку [39] для роботи як з CS SDM, так і з двома класичними конструкціями, яка є відкритою для інших дослідників.

СПИСОК ЛІТЕРАТУРИ

1. Kanerva P. Sparse distributed memory. Cambridge, MA: MIT Press, 1988. 180 p.
2. Flynn M.J., Kanerva P., Bhadkamkar N. Sparse distributed memory: principles and operation. Technical Report CSL-TR-89-400. Research Institute for Advanced Computer Science (RIACS), NASA Ames Research Center. 1989. P. 29–32. URL: http://i.stanford.edu/pub/cstr/reports/csl/tr/89/400/CSL-TR-89-400.pdf.
3. Kanerva P. Sparse distributed memory and related models. In: Associative Neural Memories: Theory and Implementation. Hassoum M.H. (Ed.). New York: Oxford University Press, 1993. P. 50–76.
4. Hopfield J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*. 1982. Vol. 79, N 8. P. 2554–2558.
5. Ackley D.H., Hinton G.E., Sejnowski T.J. A learning algorithm for Boltzmann machines. *Cognitive Science.* 1985. Vol. 9, N 1. P. 147–169.
6. Salakhutdinov R., Mnih A., Hinton G.E. Restricted Boltzmann machines for collaborative filtering. *Proc. 24th International Conference on Machine Learning (ICML'07)* (20–24 June 2007, Corvallis, USA). Corvallis, 2007. P. 791–798.

7. Jaeckel L.A. An alternative design for a sparse distributed memory. Technical Report TR 89.28. Research Institute for Advanced Computer Science (RIACS), NASA Ames Research Center. 1989. P. 13–20. URL: https://ntrs.nasa.gov/api/citations/19920001073/downloads/19920001073.pdf.

8. Jaeckel L.A. A class of designs for a sparse distributed memory. Technical Report TR 89.30. Research Institute for Advanced Computer Science (RIACS), NASA Ames Research Center. 1989. P. 17–25. URL: https://ntrs.nasa.gov/api/citations/19920002426/downloads/19920002426.pdf.

9. Marr D. A theory of cerebellar cortex. *The Journal of Physiology*. 1969. Vol. 202, N 2. P. 437–470.

10. Smith D.J., Forrest S., Perelson A.S. Immunological memory is associative. In: Artificial Immune Systems and Their Applications. Dasgupta D. (Ed.). Berlin; Heidelberg; New York: Springer, 1998. P. 105–112.

11. CandЩs E.J., Wakin M.B. An introduction to compressive sampling. *IEEE Signal Processing Magazine.* 2008. Vol. 25, N 2. P. 21–30.

12. Mallat S., Zhang Z. Matching pursuits with time-frequency dictionaries. *IEEE Trans. Signal Process.* 1993. Vol. 41, N 12. P. 3397–3415.

13. CandЩs E., Romberg J., Tao T. Stable signal recovery from incomplete and inaccurate measurements. *Comm. Pure Appl. Math.* 2006. Vol. 59, N 8. P. 1207–1223.

14. Baraniuk R., Davenport M., DeVore R., Wakin M. A simple proof of the restricted isometry property for random matrices. *Constructive Approximation*. 2008. Vol. 28, N 3. P. 253–263.

15. Schlegel K., Neubert P., Protzel P. A comparison of vector symbolic architectures. *Artif. Intell. Rev*. 2021. https://doi.org/10.1007/s10462-021-10110-3.

16. Fodor J.A., Pylyshyn Z.W. Connectionism and cognitive architecture: A critical analysis. *Cognition*. 1988. Vol. 28, Iss. 1-2. P. 7–31.

17. Smolensky P. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence.* 1990. Vol. 46, Iss. 1-2. P. 159–216.

18. Plate T.A. Holographic reduced representations. *IEEE Transactions on Neural Networks.* 1995. Vol. 6, N 3. P. 41–59.

19. Kanerva P. The spatter code for encoding concepts at many levels. *Proc. International Conference on Artificial Neural Networks (ICANN'94)* (26–29 May 1994, Sorrento, Italy). Sorrento, 1994. Vol. 1. P. 226–229.

20. Kanerva P. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive Computation*. 2009. Vol. 1, N 2. P. 139–159.

21. Sjödin G. The Sparchunk code: A method to build higher-level structures in a sparsely encoded SDM. *Proc. IEEE International Joint Conf. on Neural Networks (IJCNN/WCCI'98)* (4–9 May 1998, Anchorage, USA). Anchorage, 1998. P. 50–58.

22. Gayler R.W. Multiplicative binding, representation operators & analogy. In: Advances in Analogy Research: Integration of Theory and Data from the Cognitive, Computational, and Neural Sciences. Gentner D., Holyoak K. J., Kokinov B.N. (Eds.). Sofia: New Bulgarian University, 1998. P. 1–4.

23. Rachkovskij D.A., Kussul E.M. Binding and normalization of binary sparse distributed representations by context-dependent thinning. *Neural Computation*. 2001. Vol. 13, N 2. P. 411–452.

24. Rachkovskij D.A. Representation and processing of structures with binary sparse distributed codes. *IEEE Trans. on Knowledge and Data Engineering*. 2001. Vol. 13, Iss. 2. P. 261–276.

25. Gayler R. Vector symbolic architectures answer Jackendoff's challenges for cognitive neuroscience. *Proc. ICCS/ASCS International Conference on Cognitive Science* (13–17 July 2003, Sydney, Australia). Sydney, 2003. P. 133–138.

26. Frady E.P., Kleyko D., Sommer F.T. Variable binding for sparse distributed representations: Theory and applications. *IEEE Transactions on Neural Networks and Learning Systems*. 2020. URL: https://arxiv.org/abs/2009.06734.

27. Рачковский Д.А. Кодвекторы: Разреженное бинарное распределенное представление числовых данных. Киев: Интерсервис, 2019. 200 с.

28. Kussul E.M., Rachkovskij D.A., Baidyk T.N. Associative-projective neural networks: architecture, implementation, applications. *Proc. 4th Intern. Conf. "Neural Networks & Their Applications"* (4–8 November 1991, Nimes, France). Nimes, 1991. P. 463–476.

29. Куссуль Э.М. Ассоциативные нейроподобные структуры. Киев: Наук. думка, 1992. 144 с.

30. Laiho M., Poikonen J.H., Kanerva P., Lehtonen E. High-dimensional computing with sparse vectors. *Proc. IEEE Biomedical Circuits and Systems Conference: Engineering for Healthy Minds and Able Bodies (BioCAS-2015)* (22–24 Oct. 2015, Atlanta, USA). Atlanta, 2015. P. 1–4.

31. Ramalho T., Garnelo M. Adaptive posterior learning: few-shot learning with a surprise-based memory module. *Proc. 7th International Conference on Learning Representations (ICLR 2019)* (6–9 May 2019, New Orleans, USA). New Orleans, 2019. URL: https://arxiv.org/abs/ 1902.02527.

32. Dantzig G.B. Linear programming and extensions. Princeton, NJ: Princeton University Press, 1963. 656 p.

33. Mallat S., Zhang Z. Matching pursuits with time-frequency dictionaries. *IEEE Trans. Signal Process.* 1993. Vol. 41, Iss. 12. P. 3397–3415.

34. Needell D., Tropp J.A. CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. *Appl. Comp. Harmonic Anal.* 2008. Vol. 26, N 3. P. 301–321.

35. Відкриті коди бібліотеки CoSaMP. URL: https://github.com/rfmiotto/CoSaMP/blob/master/cosamp.ipynb.

36. Virtanen P., Gommers R., Oliphant T.E. et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods.* 2020. Vol. 17, N 3. P. 261–272.

37. Модуль лінійного програмування LinProg з бібліотеки SciPy. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.linprog.html.

38. Вдовиченко Р.А. Комп'ютерна програма «Гібридна модель нейронної пам'яті CS SDM». Свідоцтво про реєстрацію авторського права на твір № 104882 від 26.05.2021 р. (ідентифікатор в базі УкрПатенту: CR0278260521). ДП «Український інститут інтелектуальної власності». 2021.

39. Відкриті коди бібліотеки CS-SDM. URL: https://github.com/Rolandw0w/phd-sdm-cs.

R. Vdovychenko, V. Tulchinsky

INCREASING THE SEMANTIC STORAGE DENSITY OF SPARSE DISTRIBUTED MEMORY

Abstract. The Compressive Sensing (CS) method integration in the Sparse Distributed Memory (SDM) implementation is proposed for increasing the storage capacity for Binary Sparse Distributed Representations of semantics particularly in Graphics Processing Units (GPU).

Keywords: Sparse Distributed Memory, SDM, Compressive Sensing, CS, associative memory, Binary Sparse Distributed Representations, neural networks, GPU.

Надійшла до редакції 10.12.2021