УДК 004.93'1

В.Є. Яшин, магістрант
С.В. Яцук, магістрант
В.П. Полторак, к.т.н., доц.
Національний технічний університет України "КПІ"

ЗАСТОСУВАННЯ БАГАТОПОТОКОВОГО СИМУЛЯТОРА НЕЙРОННИХ МЕРЕЖ ДО ПРОБЛЕМИ РОЗПІЗНАВАННЯ ОБЛИЧ

Зроблено огляд розробленого пакету програмного забезпечення, що дозволяє виконувати моделювання штучних нейронних мереж довільної архітектури, проводити їх навчання та тестування з використанням багатопроцесорних машин, та проведений ряд дослідів з розпізнавання облич людей за допомогою згорточної нейронної мережі.

Постановка проблеми. На даний момент вивчення й використання нейронних мереж знову набирає оберти [1]. Це обумовлено розвитком комп'ютерної техніки й постійним ростом обчислювальних потужностей, доступних дослідникам. Разом з тим, інструменти для емуляції роботи й попереднього розрахунку нейронних мереж розвиваються слабо. Однією з переваг нейронних мереж є їхня швидкість обробки даних. Добре спроектована й навчена мережа працює на порядок швидше аналогічних рішень із використанням класичних алгоритмів. Але, на жаль, процес проектування й навчання нейронної мережі займає незрівнянно більше часу, ніж пряме її використання. Найчастіше для досягнення необхідних результатів потрібно перебрати десяток топологій. А сам процес навчання займає багато часу через необхідність обробки великих обсягів даних. Таким чином, необхідно мати інструмент для проектування нейронних мереж, що дозволив би задіяти максимум ресурсів комп'ютера. Сьогодні навіть на домашніх комп'ютерах стоять багатопроцесорні системи, які при оптимізованому програмуванні дозволяють прискорити роботу додатків у декілька разів. На жаль, жодна з відомих на даний момент систем для проектування нейронних мереж не використовує можливості комп'ютера на повну потужність. Це обумовлено тим, що більшість із таких пакетів написані давно, коли багатопроцесорні системи були мало поширені й не було сенсу оптимізувати код під такі системи. Зараз же, маючи материнську плату з 2-ма процесорами по 4 ядра в кожному, можна прискорити навчання нейронної мережі в 8 разів, що дозволило б істотно скоротити час проектування й тестування.

Аналіз досліджень та публікацій. Зараз існують такі пакети для проектування нейронних мереж:

SNNS – потужна бібліотека, розроблена в Штуттгардському університеті. Більша частина коду написана ще на початку 90-х рр. на чистому C, без використання об'єктно орієнтованого підходу. Це дуже ускладнює подальший розвиток.

Joone – більш сучасний пакет, написаний на Java, що трохи відбивається на швидкості роботи. Перевагою є повністю об'єктна модель. Але сама по собі бібліотека малорозвинена, зроблено натиск на візуалізацію, тому розроблювачі мало часу приділили розвитку ядра.

Matlab Neural Network Toolbox – це пакет розширення MATLAB, що містить засоби для проектування, моделювання, розробки й візуалізації нейронних мереж. Основний недолік – дуже низька швидкість роботи.

Загальним недоліком для всіх існуючих пакетів є однопотоковість обробки, а також подання нейронної мережі виняткова шарами. Шаровt подання спрощує розробку додатка, але виключає можливість проектування нейронних мереж з довільною топологією.

Мета роботи. Таким чином, було ухвалене рішення про створення власного програмного пакета для проектування нейронних мереж з використанням переваги багатопроцесорних систем. Основними вимогами до продукту стали:

– висока швидкість обробки даних;
– багатопоточність;
– об'єктно-орієнтований дизайн;
– модульність програмного пакета;
– мультиплатформенність;
– висока стабільність.

Вибір мови програмування й оточення. Існує безліч мов програмування для реалізації яких завгодно завдань. І важливо правильно вибрати інструмент, адже кожна мова має як переваги, так і недоліки. Відповідно до наданих вимог, були відкинуті інтерпретовані мови, тому що вони істотно повільніші компільованих. Для досягнення стабільності довелося відмовитися від мов з динамічною типізацією. Кінець кінцем список звузився до трьох кандидатів: Java, C# й C++. Java була відкинута через наявність віртуальної машини. C# має проблеми з терпимістю. Тому ми зупинили свій вибір на C++. Це досить стара, популярна, об'єктно-орієнтована мова програмування. C++ дозволяє максимально

використати ресурси комп'ютера, сучасні компілятори генерують настільки чистий код, що він дуже близький до ассемблерної реалізації. Таким чином, C++ дає нам можливість максимально використати апаратні ресурси ПК. Також незаперечним плюсом є наявність величезної кількості бібліотек для C++, що якоюсь мірою спрощує розробку. Природно, не обійшлося без перешкод – стандарт C++ не підтримує багатопоточність, але, на щастя, існують бібліотеки, що дозволяють писати потокобезпечні програми на C++, зокрема бібліотека Boost. Ця бібліотека є дуже потужною підмогою при розробці програм на C++, вона має безліч часто використовуваних компонентів, і новий стандарт розробляється на її базі, що забезпечує гарну сумісність у майбутньому.

При написанні допоміжних утиліт (графічного інтерфейсу, тестів тощо) були задіяні наступні набори бібліотек.

Для написання графічного інтерфейсу для супутніх програм використаний Qt – крос-платформний інструментарій розробки ПЗ. Qt дозволяє запускати написане з його допомогою ПЗ в більшості сучасних операційних систем шляхом простої компіляції програми для кожної ОС без зміни вихідного коду. Містить у собі всі основні класи, які можуть знадобитися при розробці прикладного програмного забезпечення, починаючи від елементів графічного інтерфейсу й закінчуючи класами для роботи з мережею, базами даних і XML. Qt є повністю об’єктно-орієнтованим, легко розширюваним та підтримує техніку компонентного програмування.

Для побудови графіків й аналізу тенденцій використана бібліотека gnuplot – вільна програма для створення дво- та тривимірних графіків. Gnuplot має власну систему команд, може працювати інтерактивно (у режимі командного рядка) та виконувати скрипти, що читають із файлів.

Проблеми багатопоточності. Сучасні операційні системи багатозадачні. Це дозволяє користувачеві запускати одночасно кілька додатків. Операційна система дозволяє запускати велику кількість додатків навіть на одному фізичному процесорі. Така квазібагатозадачність реалізовується шляхом постійного перемикання активного завдання. Кожен додаток у системі на якийсь квант часу захоплює процесор і виконує необхідні дії, потім ОС перемикає активність на інший додаток і так у нескінченному циклі.

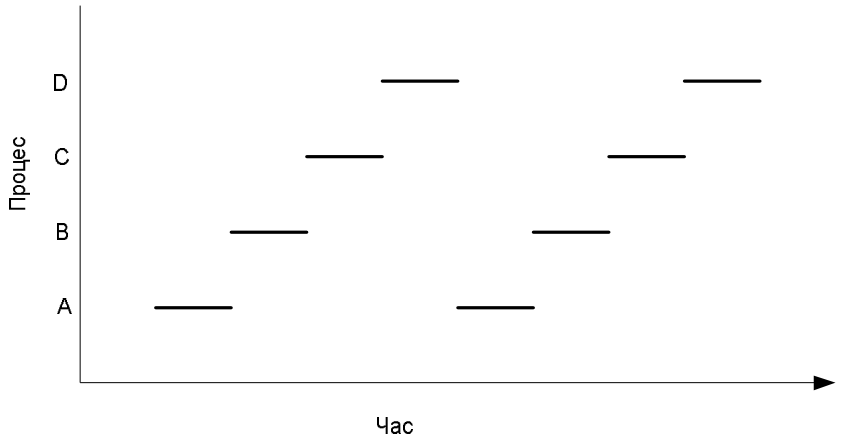

Рис. 1. Багатозадачна робота процесора

Якщо в системі доступно кілька процесорів (ядер), то ОС ділить процеси на доступні ядра й тоді кілька процесів можуть працювати дійсно одночасно. Більше того, можна розпаралелити обчислення одного додатка на кілька фізичних процесорів і тим самим збільшити швидкість виконання складних обчислень [2].

Основною проблемою при роботі багатопоточних додатків є ресурси комп'ютера – пристрої, потоки вводу/виводу, файли, дані. Складність полягає в тому, що якщо кілька потоків використовують один ресурс, то можуть виникнути колізії. Уявимо ситуацію: один потік відкриває файл для запису даних, і в цей же час другий потік видаляє цей файл. У результаті перший потік одержує невалідний покажчик на неіснуючий файл. Інший приклад, коли два потоки намагаються одночасно одержати доступ у якусь область даних. Припустимо, існує загальна змінна, в яку два потоки пишуть дані, така програма буде нестабільною і працювати буде неправильно. Для забезпечення унікальності й атомарності доступу до конкурентних ресурсів операційна система дає кілька інструментів:

Критична область – частина коду, що гарантовано виконається без перемикання поточного процесу.

Семафор – об'єкт, що дозволяє ввійти в задану ділянку коду не більш ніж *n* потокам.

Мьютекси – це найпростіші двійкові семафори, які можуть перебувати в одному із двох станів – відзначеному або невідміченому (відкритий і закритий відповідно). Коли який-небудь потік, що належить будь-якому процесу, стає власником об'єкта mutex, останній переводиться в невідмічений стан. Якщо завдання звільняє мьютекс, його стан стає відзначеним.

У різних системах реалізація таких системних викликів відрізняється, тому ми використали обгортку boost::thread, що уніфікує роботу з потоками й дає зручні інструменти.

Ідея багатопоточної обробки. Через викладені вище труднощі роботи багатопоточних додатків необхідно було мінімізувати взаємне використання одних і тих самих даних різними потоками, тому що неправильно спроектований багатопоточний додаток може працювати повільніше однопотокового через можливість постійної боротьби за унікальні ресурси [3].

Раніше ухвалене рішення про відмову від канонічного пошарового подання нейронної мережі дозволяє створювати мережі довільних топологій. Але для уніфікації роботи було введене поняття "відстані між нейронами" – hop і "довжини зв'язку" – link latency [4]. Розглянемо схему складної частини нейронної мережі з наскрізним зв'язком. Колами представлені нейрони N1, N2 та N3. Цифра біля кола – це hop. Цифра на зв'язку – це дальність зв'язку. Такий підхід дозволяє проектувати абсолютно будь-які топології.

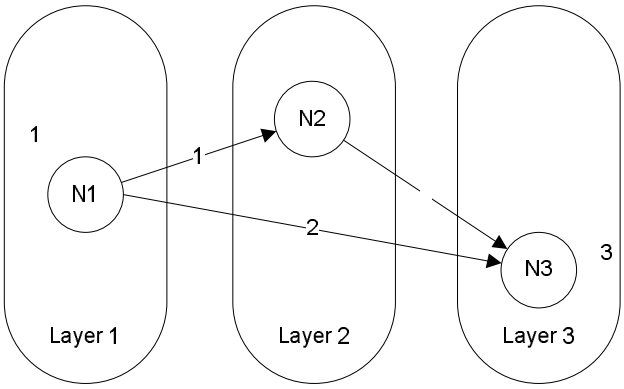

Рис. 2. Представлення складних топологій

Таким чином, ми ділимо мережу довільної топології на умовні шари обробки.

Розберемо багатопоточну обробку нейронної мережі на прикладі багатошарового персептрона.

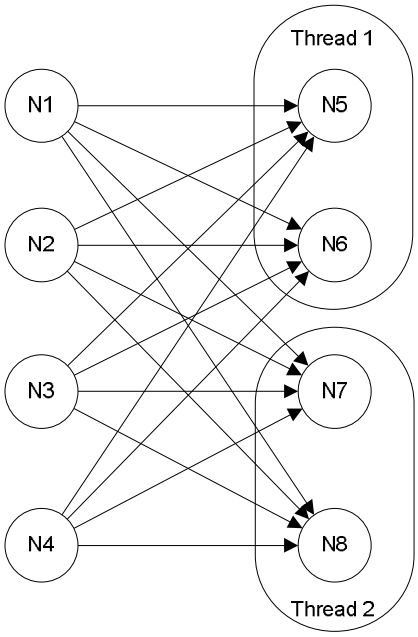

Рис. 3. Частина багатошарового персептрона

Для розрахунку локального рецептивного поля нейрона N5 потрібно прочитати значення виходів нейронів N1–N4, помножити на ваги й просумувати, потім ті ж операції потрібно повторити з наступними нейронами N6–N8. Доступ до першого шару буде винятково на читання, а значить, обробку нейронів N5–N8 можна виконувати паралельно.

Так само можна застосувати даний алгоритм багатопоточної обробки й для мереж більш складної

топології (наприклад, до зготних нейронних мереж).

Очевидно, що алгоритм навчання зворотним поширенням помилки відрізняється від прямого проходу мережі тільки напрямком. Це означає, що даний підхід дозволяє легко виконувати навчання мережі в кілька потоків.

У представленому пакеті кожен умовний шар ділиться на кілька частин (по кількості ядер у системі). Далі кожен потік обробляє свою частину даних. Очевидно, що для обробки наступного умовного шару нам необхідно повністю порахувати результати попереднього шару. Щоб потоки рухалися по мережі рівномірно, після кожного умовного шару, ставиться спеціальний об'єкт boost::barrier, який гарантує синхронізацію потоків. Таким чином, обробка йде хвилями від шару до шару.

Приклад роботи зі згорточною нейронною мережею. Розглянемо структуру згортної нейронної мережі, топологія якої надалі буде застосовуватися для розпізнавання образів.

У 1981 році нейробіологи Торстен Візел і Девід Хабел досліджували зорову кору головного мозку кішки й виявили, що існують так звані прості клітини, які особливо сильно реагують на прямі лінії під різними кутами, і складні клітини, які реагують на рух ліній в одному напрямку.

Пізніше Ян Лекун запропонував використати так звані згортні нейронні мережі як аналог зорової кори головного мозку для розпізнавання зображень.

Ідея згортних нейронних мереж полягає в чергуванні згортних шарів (C-layers), субдискретизуючих шарів (S-layers) і наявності повнозв'зних (F-layers) шарів на виході.

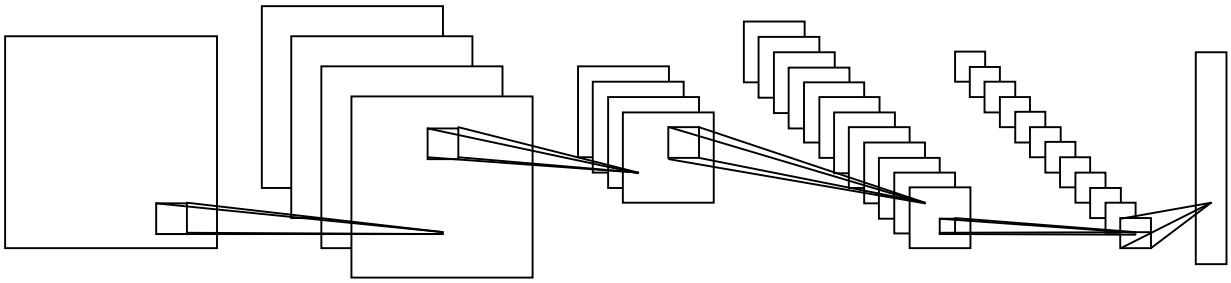

Рис. 4. Структура згортної мережі

Така архітектура містить у собі 3 основні парадигми:

– локальне сприйняття;

– поділювані ваги;

– субдискретизація.

Під локальне сприйняттям мається на увазі, що на вхід одного нейрона подається не все зображення (або виходи попереднього шару), а лише деяка його область. Такий підхід дозволяє зберігати топологію зображення від шару до шару.

Концепція поділюваних ваг припускає, що для великої кількості зв'язків використовується дуже невеликий набір ваг. Тобто, якщо в нас є на вході зображення розмірами 32x32 піксела, то кожний з нейронів наступного шару прийме на вхід тільки невелику ділянку цього зображення розміром, наприклад, 5x5, причому кожний із фрагментів буде оброблений тим самим набором. Важливо розуміти, що самих наборів ваг може бути багато, але кожний з них буде застосований до всього зображення. Такі набори часто називають ядрами (kernels).

Більшість систем розпізнавання зображень будуються на основі двовимірних фільтрів. Фільтр являє собою матрицю коефіцієнтів, звичайно задану вручну. Ця матриця застосовується до зображення за допомогою математичної операції, названої згорткою. Суть цієї операції в тому, що кожен фрагмент зображення множиться на матрицю (ядро) згортки поелементно, результат сумується й записується в аналогічну позицію вихідного зображення. Основна властивість таких фільтрів полягає в тому, що значення їхнього виходу тим більше, чим більше фрагмент зображення схожий на сам фільтр. У такий спосіб зображення, згорнуте з якимсь ядром, дасть нам інше зображення, кожен піксел якого буде означати ступінь подібності фрагмента зображення на фільтр. Іншими словами, це буде карта ознак.

Розглянемо більш докладно процес поширення сигналу в С-шарі.

Кожен фрагмент зображення поелементно множиться на невелику матрицю ваг (ядро), результат підсумовується. Ця сума є пікселом вихідного зображення, що називається картою ознак. Варто сказати, що в ідеалі не різні фрагменти проходять послідовно через ядро, а паралельно все зображення проходить через ідентичні ядра. Крім того, кількість ядер (наборів ваг) визначається розроблювачем і залежить від того, яку кількість ознак необхідно виділити. Ще одна особливість згоротного шару в тому, що він недуже зменшує зображення за рахунок крайових ефектів.

Суть субдискретизації й S-шарів полягає в зменшенні просторової розмірності зображення. Тобто вхідне зображення грубо (усередненням) зменшується в задану кількість разів. Найчастіше в 2 рази, хоча може бути й нерівномірна зміна, наприклад, 2 по вертикалі й 3 по горизонталі. Субдискретизація потрібна для забезпечення інваріантності до масштабу.

Чергування шарів дозволяє встановити карти ознак з карт ознак, що на практиці означає здатність розпізнавання складних ієрархій ознак.

Звичайно після проходження декількох шарів карта ознак вироджується у вектор або навіть скаляр, але таких карт ознак стає сотні. У такому вигляді вони подаються на один-два шари повнозв'язної мережі. Вихідний шар такої мережі може мати різні функції активації. У найпростішому випадку це може бути тангенціальна функція, також успішно використовуються радіальні базисні функції.

Експерименти й результати. Для проведення ряду експериментів з пакетом була обрана топологія згортної нейронної мережі [5]. Мережа складається із семи шарів, у яких в цілому 5161 нейрон та 132418 зв'язків. Як функція активації був обраний тангенс гіперболічний.

Експеримент проводився на персональному комп'ютері з 8-ма процесорами по 2 гігагерца кожний.

Після проведення ряду запусків були отримані усереднені дані:

Таблиця 1

Результати першого експерименту

Кількість потоків	1	2	3	4	5	6	7	8	9	10	11	12
Час (с)	89,8	67,9	55,7	50,8	46,7	44,4	42,1	40,8	43,7	42,4	42,6	42,4

Для спрощення аналізу побудуємо графік:

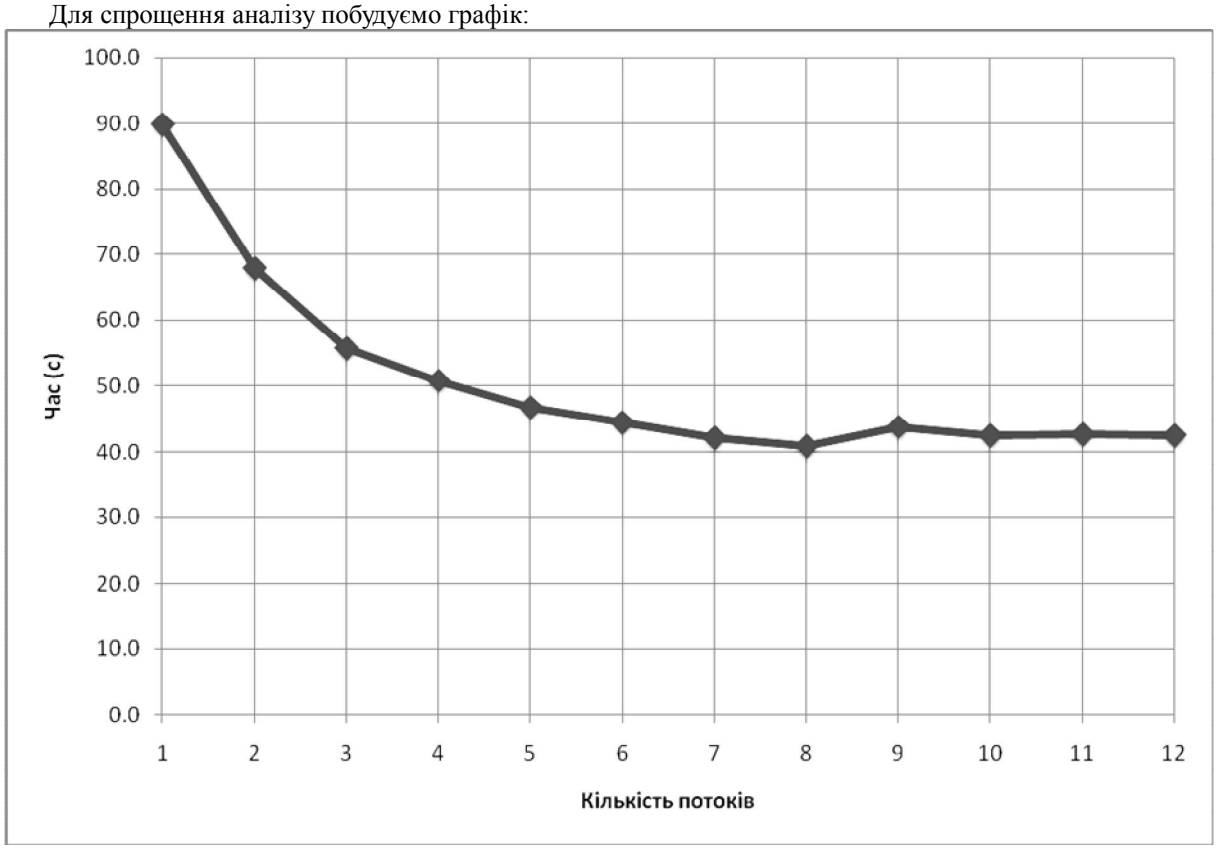

Рис. 5

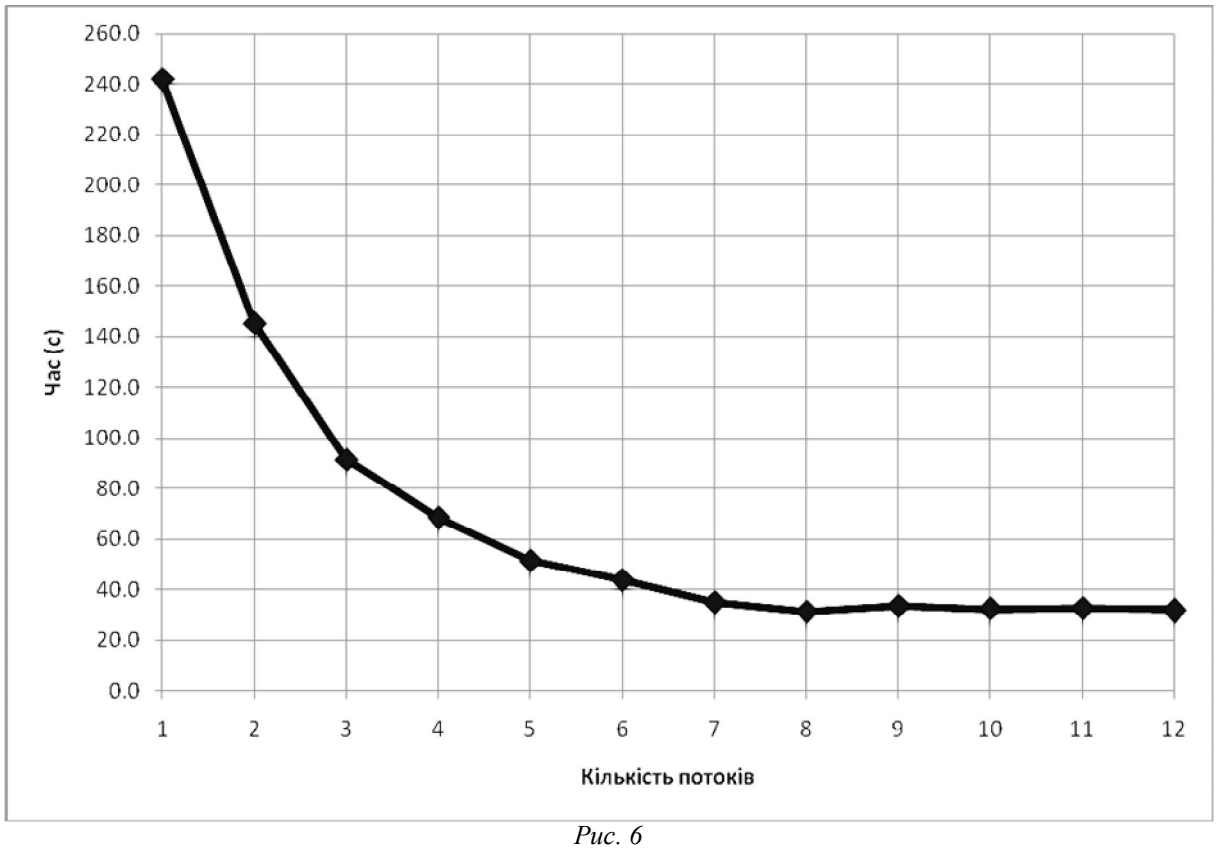

Рис. 6

Як видно (рис. 5), з ростом кількостей задіяних процесорів час обробки падає. Так триває, поки кількість потоків не починає перевищувати кількість доступних процесорів. У такому випадку час обробки знову збільшується, це обумовлено необхідністю перемикання потоків між доступними процесорами. Також варто зазначити, що швидкість обробки від доступних процесорів не лінійна – це результат синхронізації потоків за допомогою boost::barrier. Більший виграш за часом можна одержати при ще більших шарах, адже, як було описано вище, паралельно обробляється кожен шар, а поле його обробки необхідно синхронізувати з потоками обробки.

Тепер ускладнимо обробку. Візьмемо детальніше топологію згортної нейронної мережі й проведемо навчання алгоритмом зворотного поширення помилки. Отримано такі усереднені результати.

Таблиця 2

Результати другого експерименту

Кількість потоків	1	2	3	4	5	6	7	8	9	10	11	12
Час (с)	241,8	145,1	91,4	68,6	51,4	43,7	35,0	31,5	33,6	32,4	32,7	32,1

Графік витрат часу обробки при використанні різної кількості ядер наведений на рис. 6.

Наведений графік схожий на перший експеримент. Але через складність обчислень для кожного нейрона отриманий значно більший виграш від багатопоточної обробки. При використанні 8-ми ядер швидкість обробки зросла практично у 8 разів.

Висновки. У ході розробки пакета PANN були створені унікальні алгоритми паралельної обробки нейронних мереж з довільною топологією. У процесі створення даного пакета розроблювачі перебороли безліч складностей проектування й розробки концепту. Довелося піти від канонічного пошарового подання топології, що дозволило проектувати найбільш складні й немислимі топології. У результаті маємо потужний, гнучкий і швидкий пакет для проектування, тестування й використання нейронних мереж з довільною топологією.

ЛІТЕРАТУРА:

1. Neural Networks: A Comprehensive Foundation (2nd Edition) / Simon Haykin. – 842 p.,Prentice Hall; 2 edition (July 16, 1998).
2. Алгоритмы: построение и анализ / *Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест,*

Клиффорд Штайн. – 296 с.
3. Gradient-Based Learning Applied to Document Recognition / *Yann Lecun, Léon Bottou, Yoshua Bengio, Patrick Haffne* // Proc. IEEE. – 1998.

ЯШИН Володимир Євгенович – магістрант кафедри автоматики та управління в технічних системах Національного технічного університету України “КПІ”.

Наукові інтереси:
– штучний інтелект;
– нейронні мережі.

ЯЦУК Сергій Володимирович — магістрант кафедри автоматики та управління в технічних системах Національного технічного університету України “КПІ”.

Наукові інтереси:
– нейронні мережі;
– алгоритмізація.

ПОЛТОРАК Вадим Петрович – кандидат технічних наук, доцент кафедри автоматики та управління в технічних системах Національного технічного університету України “КПІ”.

Наукові інтереси:
– безпека інформаційних та телекомунікаційних систем;
– нейронні мережі.

Подано 12.01.2010

Полторак В.П., Яшин В.Є., Яцук С.В. Застосування багато потокового симулятора нейронних мереж до проблеми розпізнавання облич
Полторак В.П., Яшин В.Е., Яцук С.В. Применение многопоточного симулятора нейронных сетей к проблеме распознавания лиц.
Poltorak V.P., Yashin V.Y., Yatsuk S.V. Application of multithreaded simulator of neural networks to face recognition

УДК 004.93'1

Применение многопоточного симулятора нейронных сетей к проблеме распознавания лиц / В.П. Полторак, В.Е Яшин, С.В. Яцук

Рассмотрены разработанный пакет программного обеспечения, который позволяет выполнять моделирование искусственных нейронных сетей произвольной архитектуры, проводить их обучение и тестирование с использованием многопроцессорных систем. Так же был проведен ряд опытов по распознаванию человеческих лиц с помощью сверточной нейронной сети.

УДК 004.93'1

Application of multithreaded simulator of neural networks to face recognition / V.P. Poltorak, V.Y. Yashin, S.V. Yatsuk

In this work we give an overview of developed software package for modeling of artificial neural networks of arbitrary architecture, conduct their training and testing using multiprocessor machines. Also a series of experiments on recognition of people's faces with convolution neural networks is given.