УДК 004.627 http://doi.org/10.15407/pp2023.01.048

В.О. Лесик, А.Ю. Дорошенко

МОДУЛЬ СТИСНЕННЯ ЗОБРАЖЕНЬ НА ОСНОВІ НЕЙРОМЕРЕЖЕВИХ АВТОКОДУВАЛЬНИКІВ

В роботі запропоновано новий підхід до стиснення даних у вигляді нейромережевого модуля на базі структури автокодувальників, що має найбільш оптимальний час навчання, рівень стиснення та отримує достатньо чітку реконструкцію зображення. Розроблено основні механізми для побудови структури нейромереж кодера та декодера, що застосовуються у якості модуля. Основні дані для реконструкції були обрані із відкритого набору даних Fashion-MNIST, що дозволяє спрощено тестувати нейромережеві структури, процес їх навчання та отримання результатів. Проаналізовано підходи до відтворення зображень за допомогою нейромережевих шарів згортки та оберненої згортки. Проведено аналіз впливу на якість результуючої реконструкції зображення структури вихідного модуля, що застосовується для стиснення вхідного зображення. Знайдено нетипову поведінку під час збільшення шарів у структурі автокодувальника, що не призводять до збільшення якості відтворення зображень. Виділено основну необхідність зміни структурних частин автокодувальника та його застосування у комбінації із іншими технологіями для отримання кращого результату відтворення та нівелювання спотворень.

Ключові слова: автокодувальник, стиснення даних, генеративні нейронні мережі, відновлення зображення, TensorFlow.

Вступ

Цифрова інформація вкоренилася в усіх аспектах нашого життя та суспільства, а зростання кількості створюваної інформації здається нестримним та прогнозується експоненційне збільшення темпів створення та збереження даних у всіх інформаційних системах. Щодня на Землі ми створюємо 500 мільйонів твітів, 294 мільярди електронних листів, 4 мільйони гігабайт даних Facebook, 65 мільярдів повідомлень WhatsApp і 720 000 годин нового вмісту, що додається щодня на YouTube[1]. Стрімке розповсюдження даних потребує значних ресурсів як для збереження, так і для передачі, що потребує значних витрат заощаджень та ресурсів. Основним способом спрощення даної проблеми є застосування алгоритмів стиснення даних.

Основними перевагами стиснення є скорочення обладнання для зберігання, часу передачі даних і пропускної здатності зв'язку, що в свою чергу приводить до значної економії коштів. Стиснуті файли потребують значно меншої ємності, ніж файли без стиснення, що дозволяє значно знизити витрати на зберігання. Стиснутий файл також потребує менше часу для передачі, споживаючи меншу пропускну здатність мережі. Це теж може зменшити витрати, а також збільшити продуктивність. Крім того, оскільки для передачі стиснених файлів через Інтернет потрібно менше часу, такі організації менше інвестують у дороге оновлення пропускної здатності.

У даній статті основна увага приділяється застосуванню нових підходів до стиснення даних для спрощення передачі медіа даних у соціальних мережах різного типу. Алгоритми стиснення потребують більших витрат часу та обчислювальних ресурсів, які власне компенсують виграш у розмірі. Але сучасні прогами доступу до соціальних мереж не потребують великих обчислювальних витрат, тому вільні потужності можна застосувати для оптимізації завантаження контенту для користувача та забезпечення компенсації часу у випадку необхідності економії трафіку або зменшення часу очікування у разі виникнення збоїв передачі даних та інших

непередбачуваних ситуацій.

Основною метою даної роботи є розробка та аналіз нейромережевого модуля стиснення даних та прогнозування можливих застосувань розроблених систем. Нейронні мережі здатні знаходити приховані зв'язки у даних, а це ефективно використовується з метою стиснення.

1. Класичні методи стиснення даних

Основним поняттям теорії інформації є ентропія, яку можна неформально розглядати як міру кількості інформації в повідомленні. Під вартістю кодування розуміється середня довжина кодового слова (у бітах), що доводиться на один символ початкового тексту. Тоді надлишок кодування складає різницю між вартістю кодування і ентропією початкового повідомлення в перерахунку на один символ. Кодування має справу з потоком символів у певному алфавіті, причому частоти появи символів різні. Метою кодування є перетворення потоку символів у потік бітів мінімальної довжини. Це досягається за рахунок зменшення надлишку вихідного потоку шляхом обліку частоти появи символів на вході: довжина коду має бути пропорційна інформації, що міститься у вхідному потоці.

Якщо розподіл частот символів відомий, то можна побудувати оптимальне кодування. Завдання ускладнюється, якщо цей розподіл заздалегідь невідомий. Методи стиснення використовують закономірності у надлишкових даних, такі як: розподіл символів та знаків, повторення, наявність шаблонності у даних, позиційна надмірність. Прикладом основних методів стиснення є кодування Хаффмана, кодування довжини серії, програмоване стиснення, [2].

Кодування Хаффмана. Найпопулярнішим методом стиснення є переклад фрагментів вхідних даних фіксованого розміру в символи змінної довжини. Стандартна процедура кодування Хаффмана передбачає спосіб призначення кодів вхідним символам так аби кожна довжина коду в бітах приблизно дорівнювала логарифму від імовірності появи символу в повідомленні.

Кодування довжини серії. Послідовність ідентичних символів може бути закодована як поле підрахунку плюс ідентифікатор символу, що повторюється. Цей підхід ефективний для графічних зображень, він практично не має значення в тексті та має помірну цінність у файлах.

Проблема з кодуванням довжини серії для послідовностей символів, змішаних з іншими даними, полягає в розрізненні полів підрахунку від звичайних символів, які можуть мати той самий шаблон.

Програмоване стиснення. Програмування, як правило, виконується програмістом додатків або системою керування даними. У відформатованих файлах даних використовуються кілька методів. Невикористані пробіли або нульові пробіли усуваються, зробивши зміну довжиною та використовуючи структуру індексу з показниками на кожну позицію поля. Передбачувані значення полів компактно закодовані за допомогою кодової таблиці. Наприклад, коли вказуються назви складів як цілі коди, а не як алфавітні англійські назви. Кожне поле має власну спеціалізовану кодову таблицю, яка стосується позиційної надлишковості.

За результатами застосування даних підходів на різного роду типу даних отримується ступінь стиснення 1,8 для тексту, від 2 до 6 із файлами Cobol, 1 із даними у вигляді масиву із числами з плаваючою комою, 2,1 із форматованими науковими даними, 2,6 із даними системного журналу, 2,3 із програмним кодом та 1,5 із кодом програмних об'єктів[2].

Розглянуті методи характеризуються як стиснення без втрат, за використання якого закодована інформація може бути повністю відновлена зі стиснутих даних. Стиснення без втрат використовується, коли важливо, щоб відновленні дані були ідентичні оригіналу, тобто у загальному випадку. Альтернативою класичних методів стиснення є методи стиснення із втратами, що використовуються тоді, коли втрати допускати доцільно та ефективно. Для людини надлишковість даних часто пов'язана з якістю інформації, оскільки надлишковість, зазвичай,

покращує сприйняття інформації, як-от чіткість та якість зображення. Однак, коли мова йде про зберігання та передачу інформації засобами комп'ютерної техніки, то надлишковість відіграє негативну роль, оскільки вона призводить до зростання вартості зберігання та передачі інформації. Особливо ефективне використання стиснення даних із втратами у контексті передачі медіа файлів, адже людський зір може не сприймати досить малу втрату якості зображень. Деякі графічні файлові формати, зокрема, PNG та GIF використовують тільки стиснення без втрат, тоді як формати TIFF та MNG можуть використовувати стиснення як із втратами, так і без них[3].

2. Застосування нейромереж

Альтернативою алгоритмам стиснення може слугувати нейронна мережа, бо вона має досить схожі методи виконання: нейронна мережа знаходить необхідний нам розподіл шансів появи інформації за наданими їй даними. Тобто за допомогою нейронної мережі можна розробити модель, що буде знаходити зображення однорідних даних у компактнішому вигляді. Основна властивість нейронних мереж – це здатність знаходити непередбачувані зв'язки між вхідними навчальними даними, що вкрай важливо для побудови альтернативних методів стиснення даних[4].

Як правило, нейронні мережі в ролі алгоритму стиснення необхідно вважати як метод із втратами, бо такі мережі не можуть надавати чіткого представлення через ряд технічних обмежень. Результуючі знайдені репрезентації даних знаходяться у результаті мінімізації похибки серед усіх наданих навчальних даних, тобто чітку репрезентацію для класичних структур знайти не можливо.

Моделі нейронних мереж мають практично незліченну кількість виглядів та використань. Вихідні дані, їхня якість та виконання поставлених задач повністю залежать від структури побудови нейронних мереж та додаткових модулів і підсистем. Тому слід зауважити, що вже є приклади використання нейромереж, які дозволяють отримати чіткі дані без втрат[5].

Основне завдання при застосуванні нейронних мереж до стиснення даних є підбір необхідної структури, що може забезпечити нівелювання надлишкової чи недоречної інформації, яка не буде спотворювати результуючі відтворені дані. Якщо рівень спотворення даних є занадто великим після досягнення мінімальної похибки серед відновлень, то правильним рішенням є ускладнення моделі або повноцінна зміна підходу до побудови моделі.

3. Модель Автокодувальника

Автокодувальник — це певний тип нейронних мереж прямого зв'язку, навчальні дані якого є як вхідними даними, так і цільовими. Вони стискають вхідні дані в код меншої розмірності, а потім реконструюють вихідні дані з цього представлення, проводячи порівняння із оригіналом. Код є компактним «підсумком» або «стисненням» вхідних даних, що також називається представленням латентного простору[5].

Сучасні автокодувальники узагальнили ідею кодера та декодера за межі детермінованих функцій до стохастичних відображень. Традиційно автокодувальники використовувалися для зменшення розмірності або для виділення якісних зв'язків функцій у простішому представленні. Нещодавно теоретичні зв'язки між автокодувальниками та моделями латентних змінних автокодувальників вивели дані моделі на передній план генеративного моделювання. Загалом автокодувальники широко використовуються під час реконструкції, знаходження латентного відображення даних, зниження шуму, кластеризації, знаходження аномалій[6].

Автокодувальники можна розглядати як особливий випадок мереж прямого зв'язку, і їх можна навчити за допомогою всіх тих самих методів, як правило, міні-пакетного градієнтного спаду після градієнтів, обчислених за допомогою зворотного поширення.

Застосування автокодувальників передбачає низку особливостей застосу-

вання: специфічність даних, втрати, неконтрольоване навчання.

Специфічність даних. Автокодувальники здатні стискати лише дані, схожі на ті, на яких їх навчали. Оскільки вони вивчають функції, специфічні для заданих навчальних даних, то відрізняються від стандартного алгоритму стиснення даних, такого як gzip. Тому ми не можемо очікувати, що автокодувальник, навчений рукописним цифрам, стискатиме пейзажні фотографії.

Втрати. Вихід автокодувальника не буде точно таким, як вхід, це буде близьке, але погіршене представлення. Якщо вам потрібне стиснення без втрат, це не той шлях.

Неконтрольоване навчання. Щоб навчити автокодувальника, не треба робити нічого надто складного, достатньо введення на вході необроблених даних. Автокодувальники вважаються технікою неконтрольованого навчання, оскільки їм не потрібні чіткі мітки для навчання. Але якщо бути точнішим, вони самоконтрольовані, оскільки генерують власні мітки з даних навчання.

Автокодувальники мають типову архітектуру: як кодер, так і декодер — це повністю зв'язані нейронні мережі прямого зв'язку, по суті штучні нейронні мережі,. Код (латентне представлення) — це один рівень нейронної мережі із розмірністю за вибором. Кількість вузлів у шарі коду (розмір коду) є гіперпараметром, який встановлюється перед навчанням автокодувальника[7]. Представлення моделі архітектури зображено на рис. 1.

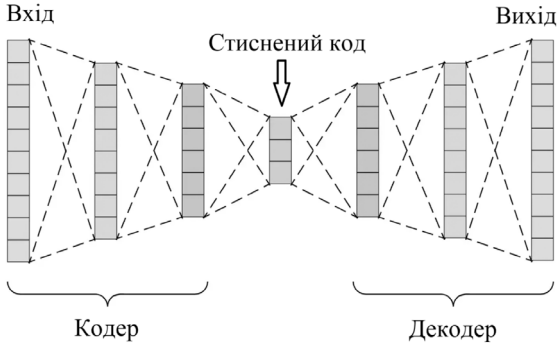

Рис. 1. Модель архітектури автокодувальника

Спочатку вхідні дані проходять через кодер, який є повністю підключеною мережею, для створення коду. Декодер, який має аналогічну структуру мережі та створює вихід лише за допомогою коду. Мета полягає в тому, щоб отримати вихід, ідентичний входу. Архітектура декодера є дзеркальним відображенням кодера. Це не обов'язкова вимога, але така модель є загальноприйнятою. Єдина вимога – розмірність входу та виходу має бути однаковою.

Є три основні гіперпараметри, які можна змінювати для навчання автокодувальника: розмір коду, кількість вузлів на шар та функція втрат.

Розмір коду. Кількість вузлів у середньому шарі, менший розмір приводить до більшого стиснення.

Кількість вузлів на шар. Архітектура автокодувальника, над якою ми працюємо, називається стекованим автокодувальником, оскільки шари складаються один за одним. Зазвичай стекові автокодери виглядають як «сендвіч». Кількість вузлів на шарі зменшується з кожним наступним шаром кодера і збільшується назад у декодері. Крім того, декодер є симетричним до кодера з точки зору структури шару. Як зазначалося вище, це не є необхідним, і ми повністю контролюємо ці параметри.

Функція втрат. Використовуємо середню квадратичну помилку або двійкову крос-ентропію. Якщо вхідні значення знаходяться в діапазоні [0, 1], зазвичай використовується крос-ентропія, за інших умов використовується середня квадратична помилка. Для налаштування складніших структур автокодувальників необхідно підбирати оптимізатор та функцію вибірково відповідно до обраних даних та структури.

4. Масштабування

Масштабування – один із перших основних підходів до вирішення проблеми надлишковості даних під час передачі зображень.

Великі масиви зображень сумарно потребують значного простору на носіях даних та викликають ускладнення під час

передачі. Для вирішення даної проблеми, як правило, застосовується метод масштабування зображення, тобто відсіювання певної кількості пікселів для зменшення розмірності, що, власне приводить до зменшення розміру файлу. Далі на точці прийому файл зображення повертають до його початкового стану за допомогою застосувань алгоритмів масштабування, що використовують корисні шаблони в окремих зонах отриманого зображення. Процес збільшення масштабу називають укрупненням масштабу (*англ. Upscaling*).

Проблема масштабування залишається актуальною і в сьогоденні. Масштабування можна використовувати також як метод підвищення якості розширеного зображення для покриття більших екранів та автоматичної генерації або покращення зображень.

Сучасні фреймворки розробки нейронних мереж мають відповідні структурні компоненти, що відповідають алгоритмам масштабування. Яскравим прикладом застосування масштабування є програмний інтерфейс фреймворку TensorFlow для створення програмованого шару нейронної мережі UpSample2D, який виконує основні функції укрупнення масштабу. Базова структура даного інтерфейсу виконує функцію дублювання пікселів відповідно до наведеної форми, що в результаті продукує масштабоване зображення. Даний інтерфейс включає застосування інтерполяції для точнішого та якісніше знаходження масштабованого зображення. У ролі інтерполяції можна використати інтерполяцію найближчого сусіда, в якій піксель із підвищеною дискретизацією є копією найближчого пікселя з пулу; білінійну інтерполяцію, де обчислюється зважена відстань між кожним найближчим пікселем за допомогою лінійної інтерполяції для обчислення нового пікселя. Можна використать бікубічну інтерполяцію, де також обирається найближчий піксель, але замість лінійної інтерполяції для обчислення нового значення пікселя отримується поліноміальна інтерполяція 3-го порядку, що потребує більше часу, але дає кращі результати.

5. Згортка, обернення згортки

Альтернативним рішенням та ефективним доповненням до моделей із використанням масштабування є використання згорток – функцій перетворення тензорів у новий формат.

Згортка — це проста математична операція, яка є фундаментальною для багатьох поширених операторів обробки зображень. Згортка забезпечує спосіб «перемноження» двох масивів чисел, переважно, різного розміру, але однакової розмірності, для створення третього масиву чисел однакової розмірності. Це можна використовувати в обробці зображень для реалізації операторів, вихідні піксельні значення яких є простими лінійними комбінаціями певних вхідних піксельних значень.

Застосування згортки дозволяє представити більший тензор за допомогою меншого та зберегти усі якісні зв’язки та закономірності даних.

У контексті обробки зображень один із вхідних масивів зазвичай є зображенням сірого рівня та для кольорових зображень – три масиви, що відповідають червоному, зеленому та синьому елементу кольорів. Другий масив, як правило, набагато менший і також є двовимірним (хоча він може мати товщину лише один піксель) і відомий як ядро. Згортка виконується ковзанням ядра по зображенню, зазвичай починаючи з верхнього лівого кута, щоб перемістити ядро через усі позиції, де воно повністю вміщується в межах зображення. Кожна позиція ядра відповідає одному вихідному пікселю, значення якого обчислюється шляхом множення значення ядра та значення пікселя основного зображення для кожного клітинок у ядрі, а потім додавання всіх цих чисел разом[8]. Типове покрокове виконання процесу згортки можна побачити на рис. 2.

У контексті автокодувальників, згорткові мережі застосовуються для отримання кращої репрезентації даних зображення та ефективнішої латентної репрезентації. Але для відновлення коду необхідно застосувати функцію, яка повертає вихідні дані зі згортки. Однак визначеної оберненої

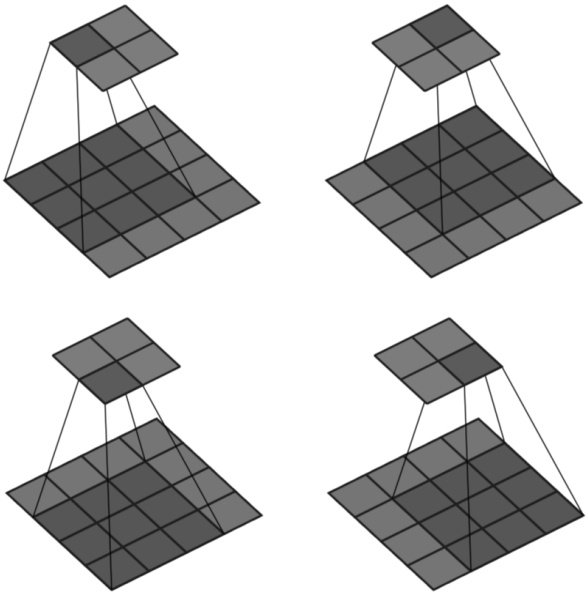

Рис. 2. Проведення згортки із ядром 3 × 3 по масиву 4 × 4 та отримання результуючого масиву 2 × 2.

функції до функції згортки не існує та аналітично її знайти неможливо. Кожне визначення комбінацій вихідного масиву та ядра, що було застосоване для його отримання, має безліч початкових масивів, що відповідають отриманню виходу.

Альтернативою обчислення оберненої згортки є транспонована згортка (*англ. Transposed Convolution*). Їх ще називають дробовими згортками, що працюють за методом зміни місцями прямого і зворотного проходів згортки. Процес оберненої згортки можна описати наступним чином: обирається ядро та перемножуються його значення із першим числом матриці. Далі знайдені числа записуються у лівий край вихідної матриці, що відновлюється. Процес відбувається для наступних чисел так само, але зі зміщенням праворуч відповідно до налаштувань ядра та кроку. Слід зауважити, що всі числа, які записуються одне на одне, будуть додаватися. Але в результаті така репрезентація не буде дорівнювати початковій матриці перед згорткою.

Фреймворк TensorFlow надає гнучкість рішенню даної проблеми. Програмний інтерфейс Convolution2DTranspose уможливлює розширення процесу оберненої згортки за допомогою включення додаткової нейронної мережі, що навчається імітувати процес згортки, який привів до отримання вихідної матриці. Тож, із кожною епохою навчання рівень мережі Convolution2DTranspose буде все більше відповідати оберненій функції згортки.

Застосування згортки та оберненої згортки у процесах відновлення та обробки приводять до значного зростання ефективності рішення задачі масштабування. Застосування даних структур дозволяє з легкістю перевершити класичні підходи із використанням інтерполяції[9] та мають широке застосування для обробки даних.

6. Розробка та аналіз базових моделей нейромереж

Для розробки базових моделей нейронних мереж на основі автокодувальників використовується програмний пакет TensorFlow. За дані для навчання та подальшого очікуваного пересилання обирається набір даних Fashion_MNIST[10] – він містить 70000 фото розміром 28×28×1 десяти різних виробів одягу, які вже є класифікованими. Такий набір даних уже можна вважати наближено нормованим та однорідним, що дає досить вагому підставу для використання нейронних мереж у ролі медіа передачі даних.

Початкова базова модель складається із прихованих шарів різного розміру. Модель декодера зазвичай повністю відповідає моделі декодера, тому реалізація має дане впровадження побудови моделі. Як дослідження буде перевірятися розмір латентного представлення даних, тобто вихід із кінцевого шару кодера. Декодер в свою чергу отримує латентне представлення та відновлює його до оригінального розміру.

Базова модель має структуру, що складається із моделі кодера та декодера. Спочатку на вхід кодера подається тензор розмірністю (n, 28, 28), де n – це розмір навчальної вибірки. Далі здійснюється вирівнювання тензору через спеціальний шар, отримується одновимірне представлення шару нейронів без функції активації; Наступний крок: будується головна структура кодувальника – будується N послідовно з'єднаних шарів, де N – кількість структурних шарів кодера; останній шар

є шаром латентного представлення, кількість нейронів якого ініціалізується під час створення моделі; кожний шар, перед латентним має вдвічі більше нейронів, ще один перед ним в 4 рази, і так далі до шару, що має в 2^N разів більше нейронів. Наступним етапом є побудова декодера: отримується тензор з латентного шару та пропускається через M шарів, останній з яких має розмірність 28×28, тобто очікуваного вихідного тензору; шар, що стоїть перед останнім, має вдвічі менше нейронів, той, що стоїть перед цим – в 4 рази, і так далі до того, що має в 2^M разів менше нейронів, ніж останній шар. Вихід останнього шару подається на особливий шар зміни розмірності, що повертає необхідну розмірність файлу зображення у вигляді двовимірного масиву, який має тип даних «ndarray». Графічне представлення побудови кодера та декодера представлено на рисунку 3 (назви, зображені на рисунку, відповідають програмним назвам виклику конструкторів класів із програмного пакету «tf.keras.layers»). Слід зауважити, що кількість шарів декодеру дещо обмежується розмірністю вихідних даних.

Під час навчання моделі було використано оптимізатор Adam[11] та середньоквадратична похибка. Кожен шар до кодувальника має функцію активації ReLU та кожен шар декодувальника має сигмоїдальну функцію активації.

Наступний крок: здійснюється аналіз ряду моделей зі зміненими структурними параметрами. Кожна модель отримує своє маркування, що має вигляд *LD{K} E{N}D{M}* , де K – кількість нейронів латентного шару, N – кількість шарів кодера, M – кількість шарів декодера. Для кожної моделі здійснюється аналіз оптимальної кількості епох до досягнення найкращого результату. Оцінюється отримана помилка на тестових та навчальних даних. Результуюча модель використовується для отримання закодованого набору тестових даних, що далі аналізуються на фактор стиснення. Як порівняння здійснюється розрахунок відношення розміру збережених зображень у форматі PNG до розміру закодованого зображення за допомогою кодувальника. Дані кодувальника зберігаються як серіалізований масив даних із розширенням “numpy”. Для збереження зображень використовується програмний пакет cv2. Результуючий коефіцієнт стиснення є середнім значенням усіх відношень розміру PNG та numpy файлів. Дані, отримані в результаті аналізу, записані у таблиці 1. Найкращі результати виділено.

За результатами досліджень варіацій базових структур автокодувальників було знайдено ряд переваг та недоліків моделей нейромереж. Із одного боку, моделі, що мали найменший шар латентного представлення даних, хоч і мають наймен-

Таблиця 1

Результати навчання базових моделей із різною структурою

Код моделей нейромереж	Кількість епох	Помилка навчання	Помилка тесту	Коеф. стиснення
LD16E1D1	10	0.0172	0.0172	9.15
LD16E2D2	50	0.0106	0.0111	9.15
LD16E3D3	100	0.0100	0.0106	9.15
LD16E4D4	120	0.0112	0.0124	9.15
LD32E1D1	10	0.0124	0.126	4.57
LD32E2D2	55	0.0082	0.0086	4.57
LD32E3D3	70	0.0072	0.0077	4.57
LD32E4D4	95	0.0086	0.0093	4.57
LD64E1D1	15	0.0087	0.0088	2.28
LD64E2D2	60	0.0059	0.0063	2.28
LD64E3D3	100	0.0058	0.0064	2.28
LD64E4D4	210	0.0075	0.0084	2.28

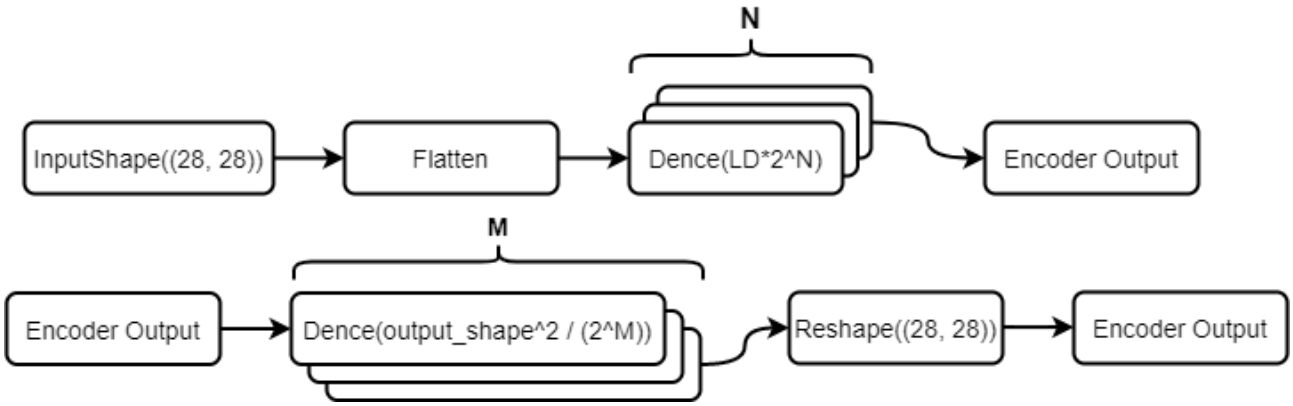

Рис. 3. Структура базової моделі автокодувальника.

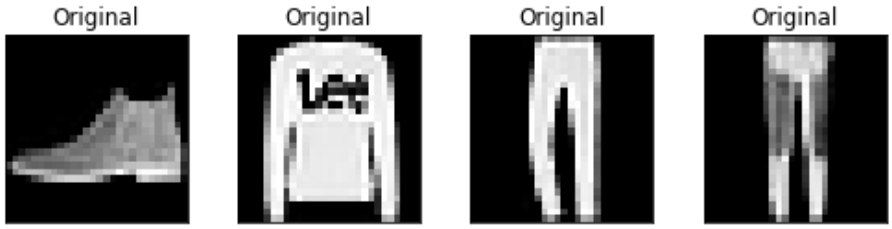

Рис. 4. Оригінальне зображення

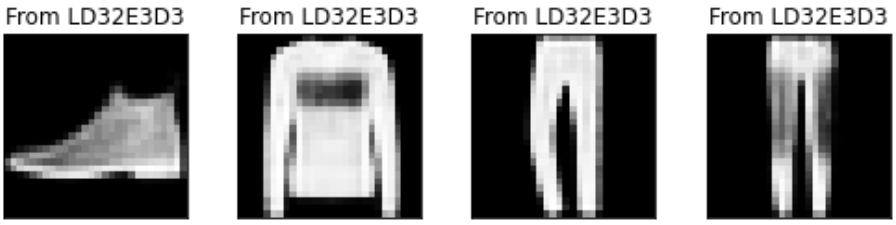

Рис. 5. Найкраще відтворення від моделі LD32E3D3

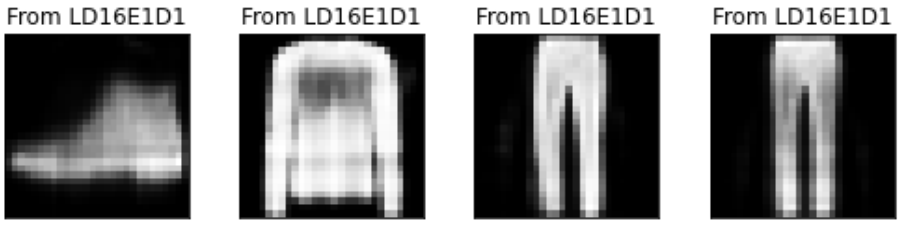

Рис. 6. Найгірше відтворення від моделі LD16E1D1

шу кількість епох, необхідних для досягнення максимальної можливої здатності до реконструкції зображень, однак мають найбільшу похибку. Найменшу похибку, як було очікувано, мають автокодувальники із найбільшою кількість нейронів у латентному шарі. Але це компенсується зменшенням коефіцієнту стиснення. Найкращою моделлю було обрано модель LD32E3D3.

Із основних спостережень за навчанням моделей можна зробити висновки, що збільшення кількості нейронів та розширення структури не завжди приводить до збільшення точності відтворення інформації. Чим більше у базових моделей шарів, тим більше вони підпадають під вплив перенавчання. Також збільшення кількості шарів на певному етапі не приводить до збільшення точності

Загальну якість відтворення зображень можна порівняти на рисунках 4-6.

Загалом відтворення із точки зору людського розпізнавання сприймаєть-

ся досить добре Але навіть у найкращій моделі помічаються розмитість та деякі спотворення. Навіть найкраща модель не може відтворити напис чи логотип, який є частиною зображення. У найгіршому випадку модель може навіть не впоратись із відтворенням єдиного елементу одягу, накладаючи на нього суміш інших.

Моделі не можуть нескінченно поліпшувати якість реконструкцій, навіть якщо додати більшу кількість шарів чи нейронів. Основний елемент, що може впливати на якість навчання системи – це збільшення кількості нейронів латентного шару, що приводить до зменшення рівня компресії даних, тому необхідно шукати нову структуру автокодувальника.

Висновки

Використання структури нейронних мереж на основі автокодувальників дозволяє досить ефективно відновлювати стандартизовані дані та має значну перевагу над іншими способами збереження даних, як, наприклад, PNG. Через можливі нечіткості отримані реконструкції можна використовувати у контексті надання попереднього вигляду зображення для ознайомлення. Людський зір може толерувати дані спотворення, але результуюча реконструкція може втрачати елементи написів та деякі деталі, що в окремих випадках можуть бути вкрай вагомими елементами.

Нейронна мережа має спотворення через знаходження узагальненого методу реконструкції, тому для отримання відтворень зображень необхідно ускладнювати структуру нейронних мереж та застосовувати комбіновані підходи до вирішення задачі стиснення. Постійне нарощування структури та кількості нейронів не приводить до покращення результату, а лише наближає до оптимального методу створення репрезентацій.

Як додаткові методи покращення якості відтворення зображень слід застосовувати комбінації підходів отримання чітких логічних зв'язків, додаткових методів реконструкції, таких як обернена згортка. Загальна якість кінцевого відтворення може бути легко поліпшена, але тільки за рахунок зменшення коефіцієнту стиснення, що не завжди є оптимальним рішенням, бо отримана якість не є суттєвою. Найбільш оптимальним вирішенням є розширення системи реконструкції зображень за допомогою введення метаданих, генералізацій, кластеризацій та застосування варіативності у структурі автокодувальників.

References

1. VOPSON, Melvin M. The world's data explained: How much we're producing and where it's all stored. In: *World Economic Forum, May*. 2021.
2. WELCH, Terry A. A technique for high-performance data compression. *Computer*, 1984, 17.06: 8-19.
3. WIGGINS, Richard H., et al. Image file formats: past, present, and future. *Radiographics*, 2001, 21.3: 789-798.
4. MAZIASHVILI, A. R. Feasibility of using neural networks for compression of video data. *Information and control systems in railway transport,,* 2016, 6: 30-35.
5. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. *Deep learning*. MIT press, 2016.
6. BANK, Dor; KOENIGSTEIN, Noam; GIRYES, Raja. Autoencoders. *arXiv preprint arXiv:2003.05991*, 2020.
7. DERTAT, Arden. Applied deep learning - part 3: Autoencoders. *Towards data science*, 2017.
8. DUMOULIN, Vincent; VISIN, Francesco. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.
9. DONG, Chao, et al. Learning a deep convolutional network for image super-resolution. In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part IV 13*. Springer International Publishing, 2014. p. 184-199.
10. XIAO, Han; RASUL, Kashif; VOLLGRAF, Roland. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.
11. KINGMA, Diederik P.; BA, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Одеоржано: 18.02.2023

Про авторів:
Лесик Валентин Олександрович,
магістрант Національного Технічного Університету України «КПІ імені Ігоря Сікорського». https://orcid.org/0000-0002-8307-5707

Дорошенко Анатолій Юхимович,
доктор фізико-математичних наук,
професор, завідувач відділу теорії комп'ютерних обчислень,
професор кафедри інформаційних систем та технологій Національного Технічного Університету України
«КПІ імені Ігоря Сікорського».
Кількість наукових публікацій в українських виданнях – понад 200.
Кількість наукових публікацій в зарубіжних виданнях – понад 80.
Індекс Гірша – 6. http://orcid.org/0000-0002-8435-1451

Місце роботи авторів:
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»,
проспект Перемоги 37 та
Інститут програмних систем НАН України, 03187, м. Київ-187, проспект Академіка Глушкова, 40.
Тел.: (044) 526 3559
E-mail:
artemissterio@gmail.com,
doroshenkoanatoliy2@gmail.com