Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Центр ННЦЗФН
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Еволюційна інкрементна нейронна мережа в задачах опрацювання потоків даних
(тема)

Виконав:
студент 2 курсу, групи ДСзм-21-1
Ляшенко Д.Л.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Науки про дані
(повна назва спеціалізації)

Керівник доц. Дейнеко А.О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В.О. Філатов
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет ННЦЗФН
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Науки про дані
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Ляшенко Дарині Леонідівні
(прізвище, ім'я, по батькові)

1. Тема роботи Еволюційна інкрементна нейронна мережа в задачах опрацювання потоків даних

затверджена наказом університету від __ _____ 20 23 р. № _____

2. Термін подання студентом роботи до екзаменаційної комісії 23 травня 20 23 р.

3. Вихідні дані до роботи Науково-технічні публікації, дані Інтернет-джерел та відомих наукових проектів щодо розробки систем е волюційних інкрементих нейронних мереж в задачах опрацювання потоків даних.

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної галузі та формалізація задачі
2) Алгоритм L STM нейроних мереж
3) Рекурентна нейронна мережа
4) Нейронні мережі за допомогою бібліотеки Tensorflow
5) Імітаційне моделювання і перевірка теоретичних досліджень

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) _____

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз предметної галузі	13.04.2023	Виконано
2	Дослідження існуючих методів візуалізації	14.04.2023	Виконано
3	Розробка еволюційної інкрементої нейронної мережі	15.04.2023	Виконано
4	Створення імітаційної моделі	17.04.2023	Виконано
5	Тестування і аналіз отриманих результатів	23.04.2023	Виконано
6	Оформлення пояснювальної записки	25.04.2023	Виконано
7	Попередній захист	14.05.2023	Виконано
8	Захист перед ЕК	23.05.2023	

Дата видачі завдання __ _____ 20 23 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Дейнеко А.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 113 с., 22 рис., 4 табл., 1 дод., 25 джерел.

ВІЗУАЛІЗАЦІЯ, ІНКРЕМЕНТНА НЕЙРОННА МЕРЕЖА, ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ, НЕЙРОННА МЕРЕЖА, НЕЧІТКА СИСТЕМА, ПОСЛІДОВНЕ НАВЧАННЯ, ШТУЧНИЙ ІНТЕЛЕКТ.

Об'єктом дослідження є розробка програмного коду та desktop-додатку візуалізації даних з використанням алгоритмів побудови еволюційної інкрементної нейронної мережі.

Предметом дослідження є математичні методи візуалізації даних та скорингу.

Метою кваліфікаційної магістерської роботі є розглянути методи побудови інкрементної нейронної мережі в задачах опрацювання потоків даних, графіків та діаграм для візуалізації багатовимірних даних.

Методи дослідження базуються на теорії обчислювального інтелекту, а саме на методах теорії штучних нейронних мереж для побудови архітектури еволюційної інкрементої нейронної мережі, що складаються з нечіткого шару виводу і дозволяють проводити візуалізацію в послідовному режимі. Імітаційне моделювання застосовується для перевірки якості візуалізації.

В кваліфікаційній роботі розглядається задача побудови еволюційної інкрементої мейронної мережі та рекурентних нейронних мереж.

ABSTRACT

Explanatory note: 113 p., 22 fig., 4 tabl., 1 ann., 25 sources.

ARTIFICIAL INTELLIGENCE, FUZZY SYSTEM, INCREMENTAL NEURAL NETWORK, INTELLIGENT DATA ANALYSIS, NEURAL NETWORK, SEQUENTIAL, LEARNING VISUALIZATION,

The object of the research is the development of software code and a desktop application for data visualization using algorithms for building an evolutionary incremental neural network.

The subject of research is mathematical methods of data visualization and scoring.

The purpose of the qualifying master's thesis is to consider the methods of building an incremental neural network in the tasks of processing data flows, graphs and diagrams for the visualization of multidimensional data.

The research methods are based on the theory of computational intelligence, namely on the methods of the theory of artificial neural networks for building the architecture of the evolutionary incremental neural network, consisting of a fuzzy output layer and allowing visualization in a sequential mode. Simulation modeling is used to check the quality of visualization.

The qualification work considers the task of building an evolutionary incremental meiron network and recurrent neural networks.

ЗМІСТ

ВСТУП

У світі постійно виникають завдання, пов'язані з обробкою великих обсягів даних та високим навантаженням. Важливим прикладом можуть бути Web-додатки, такі як соціальні мережі, торгові платформи, великі портали новин і т.д. Всі вони мають свою специфіку, але їх загальною особливістю є робота з величезними обсягами даних та великою кількістю користувачів, які можуть читати або змінювати ці дані. Крім того, обсяги даних у такому випадку постійно зростають, так само як і аудиторія, що призводить до питання про те, як масштабувати додаток (і рішення, що використовуються для роботи з даними). Зазвичай розглядають масштабованість у двох напрямках: вертикальну (нарощування обчислювальної потужності одного сервера) та горизонтальну (нарощування числа серверів). Вертикальна масштабованість, як відомо, має свою межу, а фінансові витрати нелінійно залежить від продуктивності сервера. Таким чином, прийнятним варіантом виявляється лише горизонтальне масштабування, тобто збільшення числа серверів при зростаючих навантаженнях та обсягах даних.

На жаль, горизонтальне масштабування системи керування даними є досить складним завданням. Традиційні SQL-орієнтовані СУБД спочатку створювалися для роботи на одній машині і тому погано пристосовані для роботи в кластері або хмарі, що призвело до розробки нових розподілених систем та підходів, що вирішують проблему масштабування. Незважаючи на те, що всі вони спрямовані на вирішення проблеми масштабованості, кожне рішення має ті чи інші переваги та недоліки і може ефективно справлятися таким чином лише з певним класом завдань. Думка про те, що універсальних систем керування даними не існує, підкреслює загальну тенденцію до переходу від повсюдного використання «універсальних» SQL-орієнтованих СУБД до вибору системи, виходячи з розв'язуваного

завдання.

До великих даних (big data) зазвичай відносять такі типи:

– традиційні корпоративні дані, включаючи інформацію про клієнтів із систем CRM, транзакційні дані ERP, транзакції в інтернет-магазинах та дані головної бухгалтерської книги;

– машинні дані/дані датчиків, включаючи записи дзвінків (CDR), інтернет-журнали, інтелектуальні лічильники, виробничі датчики, журнали обліку експлуатації обладнання («цифровий вихлоп») та дані торгових систем;

– дані соціальних мереж, включаючи потоки зворотного зв'язку з клієнтами, такі сайти мікроблогінгу як Твіттер, а також платформи соціальних медіа типу Facebook.

Щоб отримати максимум з великого обсягу даних, підприємства повинні розвивати свої IT-інфраструктури для обробки нових обсягів високошвидкісних даних з різних джерел та їх інтеграції до існуючих корпоративних даних для аналізу.

Фільтрування та аналіз великих даних разом із традиційними даними допомагає підприємствам краще і глибше зрозуміти свій бізнес, що сприяє зростанню продуктивності, підвищує конкурентоспроможність, відкриває нову дорогу інноваціям і зрештою суттєво збільшує прибуток.

У фінансовій сфері часто розглядається проблема прогнозування короткочасної тенденції часового ряду без розрахунку безпосередньо прогнозних значень.

Це пов’язано з тим, що фінансовим часовим рядам часто притаманний нестійкий характер коливань. Такі ряди можуть бути близькими до випадкових. В цьому випадку будують моделі, які б дозволяли розрахувати прогнози знаків приростів значень часового ряду на одну точку вперед з необхідною максимальною точністю. Особливістю таких моделей і методів є можливість знаходження достатньо стійкої

залежності поточного спостереження від попередніх. На підставі цієї залежності розраховується прогноз знаку приросту на наступну точку, тобто з періодом 1. Моделі такого типу зазвичай застосовуються для визначення напрямку руху часових рядів валютних пар і можуть використовуватися для визначення точок зміни тенденцій. Для прогнозування знаків приростів застосовують специфічні моделі та методи. Слід зазначити, що перед реалізацією таких моделей необхідно дослідити вхідний фінансовий часовий ряд на абсолютну випадковість.

Це може бути реалізовано на основі фрактального аналізу або з використанням критеріїв поворотних точок, розподілу довжини фази тощо.

Основними етапами при побудові прогнозу в цьому випадку є:

– прогнозна ретроспекція (тобто встановлення об’єкту прогнозування, що складається з передпрогнозного аналізу об’єкту та оцінки його параметрів);

– вибір методів прогнозування та побудова моделі прогнозування і її формалізація;

– побудова проспекції (розрахунок прогнозу на визначений період);

– оцінка прогнозу і його верифікація.

Об'єкт дослідження – розробка програмного коду та desktop-додатку візуалізації даних з використанням алгоритмів побудови еволюційної інкрементної нейронної мережі.

Предмет дослідження – математичні методи візуалізації даних та скорингу.

Мета роботи – розглянути методи побудови інкрементної нейронної мережі в задачах опрацювання потоків даних, графіків та діаграм для візуалізації багатовимірних даних.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ФОРМАЛІЗАЦІЯ ЗАДАЧІ

1.1 LSTM нейронні мережі та прогнозування часових рядів

Регресійним аналізом називається розділ математичної статистики, що поєднує методи дослідження регресійної залежності між величинами за статистичними даними. В результаті роботи регресійного аналізу можна визначити загальний вид рівняння регресії, виконати розрахунок оцінок невідомих параметрів, що входять до рівняння регресії, та виконати перевірку статистичних гіпотез про отриману регресію.

Завдання регресійного аналізу вирішується після виконання кількох кроків.

Аналіз тренду. Тренд - це зміна, що визначає загальний напрямок розвитку, основну тенденцію низки. Часто виділяють лінійний, логарифмічний, поліноміальний та експоненційний тренди.

Не існує «автоматичного» способу виявлення тренду в часовому ряді. Однак якщо тренд є монотонним (стійко зростає або стійко зменшується), аналізувати такий ряд зазвичай неважко. Якщо тимчасові ряди містять значну помилку, то першим кроком виділення тренда є згладжування.

Згладжування. Згладжування завжди включає певний спосіб локального усереднення даних, у якому несистематичні компоненти взаємно погашають одне одного. Найзагальніший метод згладжування - ковзне середнє, в якому кожен член ряду замінюється простим або виваженим середнім n сусідніх членів, де n - ширина "вікна". Замість середнього можна використовувати медіану значень, що потрапили у вікно. Основна перевага медіанного згладжування, у порівнянні зі згладжуванням ковзним середнім, полягає в тому, що результати стають більш стійкими до викидів (всередині вікна). Таким чином, якщо даних є

викиди (пов'язані, наприклад, з помилками вимірювань), то згладжування медіаною зазвичай призводить до більш гладким або, принаймні, більш «надійним» кривим, порівняно зі ковзним середнім з тим же самим вікном.

Припасування функції. Багато монотонних часових рядів можна добре наблизити лінійною функцією. Якщо є явна монотонна нелінійна компонента, то дані спочатку слід перетворити, щоб усунути нелінійність. Зазвичай для цього використовують логарифмічне, експонентне або (менш часто) поліноміальне перетворення даних.

Аналіз сезонності. Сезонні коливання – зміни значення ряду, що повторюються, в певні проміжки часу.

Періодична та сезонна залежність (сезонність) є іншим загальним типом компонент тимчасового ряду. Періодична залежність може бути формально визначена як кореляційна залежність порядку k між кожним і елементом ряду і (ik) елементом. Її можна виміряти за допомогою автокореляції (тобто кореляції між самими членами ряду); k зазвичай називають лагом (іноді використовують еквівалентні терміни: зсув, запізнення). Якщо помилка виміру не надто велика, то сезонність можна визначити візуально, розглядаючи поведінку членів низки через кожні k часових одиниць.

Автокореляційна корелограма. Сезонні складові часового ряду можуть бути знайдені за допомогою корелограми. Коррелограмма показує чисельно і графічно автокорреляційну функцію (АКФ), тобто коефіцієнти автокореляції (та його стандартні помилки) для послідовності лагів з певного діапазону.

Часткові автокореляції. Інший корисний метод дослідження періодичності полягає у дослідженні часткової автокореляційної функції. У ЧАКФ усувається залежність між проміжними спостереженнями (спостереженнями всередині лага). Іншими словами, автокореляція на даному лазі аналогічна до звичайної автокореляції, за

винятком того, що при обчисленні з неї видаляється вплив автокореляцій з меншими лагами. На лазі 1 (коли немає проміжних елементів усередині лага), автокореляція дорівнює, очевидно, звичайної автокореляції. Насправді, така автокореляція дає «чистішу» картину періодичних залежностей.

Приклад автокореляційної корелограми показано на рисунку 1.1.

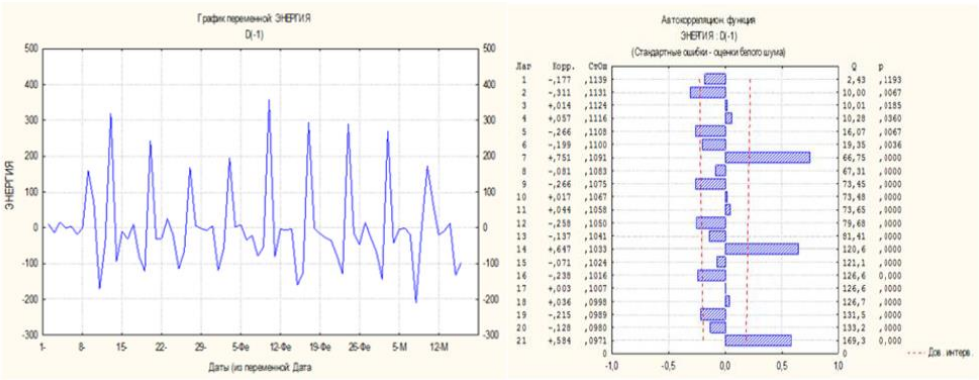

Рисунок 1.1 – Автокореляційна корелограма перетвореного стаціонарного часового ряду

Приклад автокореляційної корелограми показаний на рисунку. 1.2.

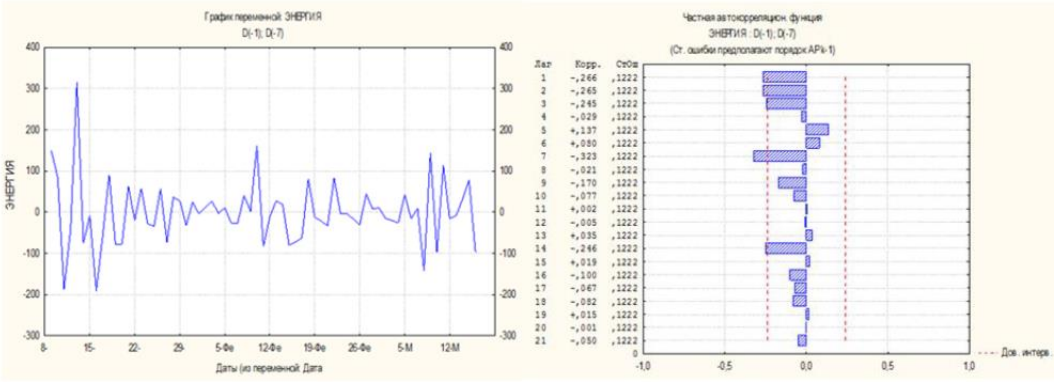

Рисунок 1.2 – Автокореляційна корелограма перетвореного стаціонарного часового ряду

Видалення періодичної залежності. Періодична складова даного лага k може бути видалена взяттям різниці відповідного порядку. Це означає, що з кожного і-го елемента ряду віднімається (ik)-й елемент. Є два аргументи на користь таких перетворень.

При моделюванні реальних об'єктів часто виявляється, що порушена передумова класичного регресійного аналізу, за якою $V(\varepsilon)=\sigma^2 I$.

Таке порушення зазвичай проявляється у наступному. Може виявитися, що обуренняємає неоднорідні дисперсії в різних дослідах і його коваріаційна матриця (а значить, і матриця вектора у при невипадковій матриці плану) буде хоч і діагональної, але з різними елементами по діагоналі. І тут кажуть, що спостереження неоднорідні чи є неоднорідність.

Неоднорідність може виникнути природно, коли вона обумовлена природою явища, що вивчається. Характерним прикладом такого роду є об'єкти, про які з фізичних міркувань відомо, що дисперсія $\sigma^2(y)$ відгуку (у) залежить від його математичного очікування Е(у). Зрозуміло, що при змінах факторів у різних дослідах змінюватиметься і $\sigma^2(y)$. Такі властивості має розподіл Пуассона, а також біноміальний розподіл. Їхні дисперсії відповідно рівні:

$$E(y)\left(1-E(y)\right)/N, \tag{1.1}$$

де N – обсяг вибірки.

Часто природа досліджуваного явища підказує, що дисперсія відгуку зростає зі зростанням певного чинника. Подібні випадки трапляються в економіці, біології, хімічній кінетиці.

Є підстави вважати, що випадкове обурення іноді обумовлено головним чином помилкою виміру відгуку. Тоді цілком імовірно, що більшим значенням будуть відповідати й великі обурення, отже дисперсіїε-неоднорідні.

Перерахуємо деякі випадки, коли неоднорідність, що виникла, введена «ззовні». Така неоднорідність виникає через лінеаризації деяких моделей. Так, при дослідженні кінетики реакцій рівняння для швидкостей часто-густо лінеаризуються за допомогою звернення. При цьому неоднорідність, що вводиться, може виявитися дуже серйозною. Подібне явище спостерігається і коли переходять від поширених у хімічному експерименті нелінійних моделей з адитивною помилкою ε_0:

$$y = \alpha e^{\beta x} + \varepsilon_0 = \alpha e^{\beta x}\left(1 + \varepsilon_0 / E\left(y\right)\right). \tag{1.2}$$

До лінійних моделей виду:

$$\ln y = \ln \alpha + \beta x + \ln\left(1 + \varepsilon_0 / E\left(y\right)\right), \tag{1.3}$$

де $E(y) = \alpha e^{\beta x}$.

Вплив лінеаризованої моделі:

$$\varepsilon = \ln\left(1 + \varepsilon_0 / E\left(y\right)\right). \tag{1.4}$$

Отже, її дисперсія змінюється залежно від зміни $E(y)$ у планах.

Неоднорідність може виникнути також через неправильний вибір структури регресійної моделі. Якщо фактор x_i входить до неї лінійно, а в істинній моделі присутній і член x_i^2. У випадковому обуренні моделі з'явиться добавка, яка залежить від x_i^2. Тому дисперсія $\sigma^2(\varepsilon)$ буде змінюватися від досвіду до досвіду під впливом x_i^2.

Неоднорідність виникає й за порушення передумови регресійного аналізу, що стосується невипадкового характеру матриці плану, як у ході

експерименту задані рівні чинників встановлюються помилками. У цьому випадку для нелінійної за факторами моделі у всіх дослідах дисперсія у збільшується на деяку складову, що залежить не тільки від коефіцієнтів моделі та моментів помилок факторів, а й від заданих рівнів факторів. Тому й тут $\sigma^2(y)$ змінюється від дослідження до дослідження.

Експонентний відеоімпульс описується наступною математичною моделлю:

$$s(t) = \exp(-\alpha t) \cdot \sigma(t), \tag{1.5}$$

Даний імпульс у частотній області представлений своїм енергетичним спектром виду:

$$W_s(\omega) = \frac{1}{\left(\alpha^2 + \omega^2\right)}, \tag{1.6}$$

та нормою:

$$\|s\| = \sqrt{\frac{1}{\pi}\int_0^{\infty} \frac{d\omega}{\left(\alpha^2 + \omega^2\right)}} = \frac{0{,}7071}{\sqrt{\alpha}}, \tag{1.7}$$

Ефективна тривалість експоненційного імпульсу дорівнює:

$$\tau_{\text{и}} = 2{,}303/\alpha\,. \tag{1.8}$$

Спектр сигналу, що розглядається, необмежений, тому слід попередньо піддати сигнал низькочастотної фільтрації, пропустивши його

через фільтр нижніх частот (ФНЧ). Значення верхньої частоти ωу смуги пропускання фільтра слід вибирати в залежності від того, як часто беруться відліки сигналу на виході ФНЧ. Припустимо, що за час τ та вимірюються k=10 відліків, тоді інтервал дискретизації становитиме $t_0 = \tau_и/(k-1) = \tau_и/9 = 0{,}2558/\alpha =$ 0,2558/2000 = 0,0001279 наведено на рисунку 1.3.

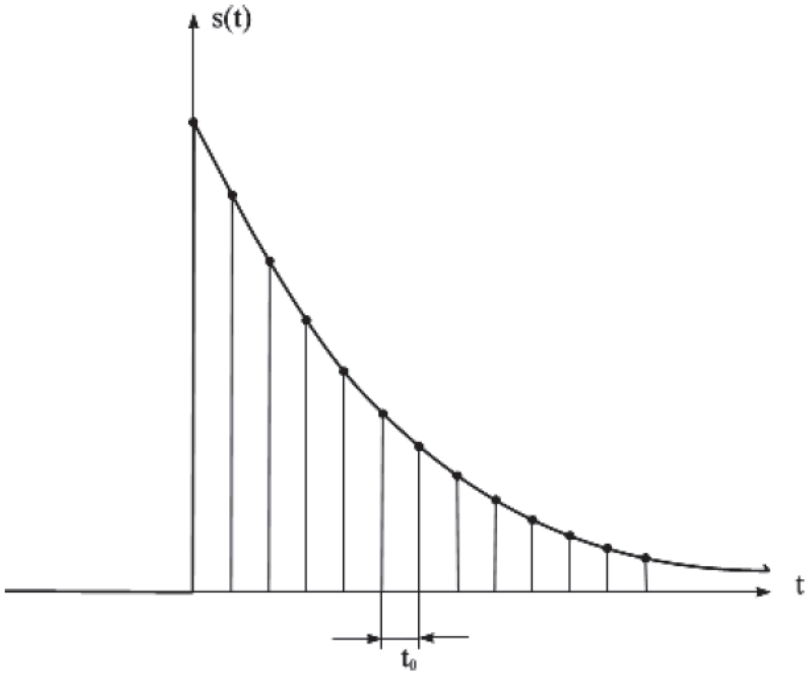

Рисунок 1.3 – Дискретизація експоненційного імпульсу

У загальному вигляді регресійну модель, що спирається на фактори, що не залежать від ряду, можна віднести до причинно - слідчої групи моделей, але існують варіації, що використовують тільки значення ряду з минулого - це модель авторегресії AR(p) порядку p

$$AR(p) = c + \sum_{i=1}^{p} bi \cdot X_{t-i} + \varepsilon_t \quad , \qquad (1.9)$$

де c – константа;

bi – параметри моделі;

εi – помилки.

Передбачається, що помилки незалежні та нормально розподілені.

Завдання методу полягає у визначенні параметрів bi. Зазвичай це робиться за допомогою системи рівнянь Юла-Уокера, але можна скористатися і більш простим методом найменших квадратів.

Дана модель краще адаптується до зміни показників та сезонності ряду, ніж модель ковзного середнього, але як основа для прогнозу так само рідко використовується на практиці.

У випадках, коли є одна незалежна та одна залежна змінні, природним заходом залежності (у рамках лінійного підходу) є вибірковий (парний) коефіцієнт кореляції між ними.

Використання множинної регресії дозволяє узагальнити це поняття у разі, коли є кілька незалежних змінних. У цьому випадку необхідне коригування, так як високе значення коефіцієнта кореляції між залежною і будь-якою незалежною змінною може означати високий ступінь лінійної залежності, але може означати і те, що третя змінна, значно впливає на дві перші і, що саме вона служить основний причиною їхньої високої кореляції. Тому необхідно знайти чисту кореляцію між двома змінними, виключивши вплив інших факторів шляхом розрахунку коефіцієнта кореляції.

Коефіцієнти кореляції для рівняння регресії з двома незалежними змінними розраховуються як:

$$r_{yx_1(x_2)} = \frac{r_{yx_1} - r_{yx_2} \cdot r_{x_1x_2}}{\sqrt{(1 - r_{yx_2}^2) \cdot (1 - r_{x_1x_2}^2)}}, \quad (1.10)$$

$$r_{yx_2(x_1)} = \frac{r_{yx_2} - r_{yx_1} \cdot r_{x_1x_2}}{\sqrt{(1 - r_{yx_1}^2) \cdot (1 - r_{x_1x_2}^2)}}, \quad (1.11)$$

$$r_{x_1x_2(y)} = \frac{r_{x_1x_2} - r_{yx_1} \cdot r_{yx_2}}{\sqrt{(1 - r_{yx_1}^2) \cdot (1 - r_{yx_2}^2)}}, \quad (1.12)$$

де $r_{yx_1(x_2)}$– коефіцієнт кореляції між у та х1 при виключеному впливі х2;

$r_{yx_2(x_1)}$ – коефіцієнт кореляції між у та х2 при виключеному впливі х1;

$r_{x_1x_2(y)}$ – коефіцієнт кореляції між х1 і х2, що виключає вплив у.

1.2 Векторизація слів

Векторизація слів (world embeding) – це клас підходів для представлення слів і документів з використанням векторного представлення. Це поліпшення порівняно з традиційними схемами кодування, де для представлення кожного слова використовували великі розріджені вектори або оцінку кожного слова у векторі для представлення цілого словникового запасу. Ці уявлення були мізерними, тому що словники були великими, і дане слово або документ представляли б великим вектором, що складається в основному з нульових значень.

Натомість у world embeding слова представлені щільними векторами, де вектор представляє проєкцію слова в безперервний векторний простір.

Представлення слова у векторному просторі отримується з тексту і ґрунтується на словах, які оточують слово, коли воно використовується.

Два популярних приклади методів вкладання слів у текст включають:

– Word2Vec.

– GloVe.

На додаток до цих раніше розроблених методів, векторизацію слів можна вивчити як частину моделі глибокого навчання.

1.3 Рекурентні нейронні мережі

Рекурентні нейронні мережі (англ. Recurrent neural network; RNN) – вид нейронних мереж, де зв'язки між елементами утворюють спрямовану послідовність. Завдяки цьому з'являється можливість обробляти серії подій у часі або послідовні просторові ланцюжки (рисунок 1.4).

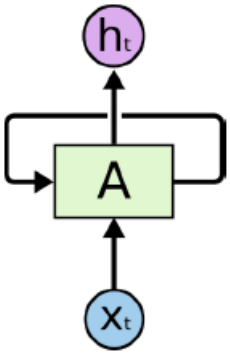

Рисунок 1.4 – Рекурентна мережа

На відміну від багатошарових перцептронів рекурентні мережі можуть використовувати свою внутрішню пам'ять для обробки послідовностей довільної довжини. Тому мережі RNN застосовні у таких завданнях, де щось цілісне розбите на сегменти, наприклад, розпізнавання рукописного тексту чи розпізнавання мови. Було запропоновано багато різних архітектурних рішень для рекурентних мереж від найпростіших до складних. Останнім часом найбільшого поширення набули мережі з довготривалою та короткочасною пам'яттю (LSTM) та керований рекурентний блок (GRU).

У діаграмі вище ділянка нейронної мережі A отримує деякі дані X на вхід і подає на вихід деяке значення H. Циклічний зв'язок дозволяє передавати інформацію від поточного етапу мережі до наступного.

Існує багато різновидів, рішень та конструктивних елементів рекурентних нейронних мереж. Проблема рекурентної мережі полягає в

тому, що якщо враховувати кожен крок часу, то стає необхідним для кожного кроку часу створювати свій шар нейронів, що викликає серйозні обчислювальні складності. Крім того, багатошарові реалізації виявляються обчислювально нестійкими, тому що в них зазвичай зникають або зашкалюють ваги. Якщо обмежити розрахунок фіксованим тимчасовим вікном, то отримані моделі не відображатимуть довгострокових трендів. Різні підходи намагаються вдосконалити модель історичної пам'яті та механізм запам'ятовування та забування.

Рекурентні нейронні мережі не так сильно відрізняються від звичайних нейронних мереж. Їх можна уявити, як безліч копій однієї і тієї ж мережі, причому кожна копія передає повідомлення наступної копії (рисунок 1.5).

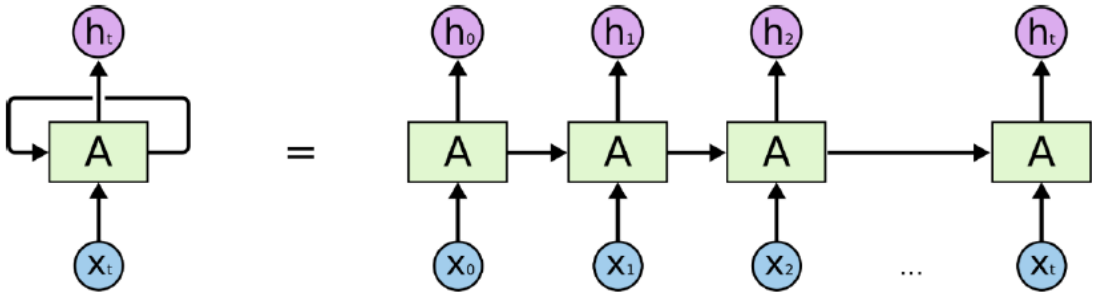

Рисунок 1.5 – Передача копій

Така «ланцюгова» сутність показує, що рекурентні нейронні мережі за своєю природою тісно пов'язані з послідовностями і списками.

1.4 Повністю рекурентна мережа

Це базова архітектура, розроблена в 1980-х. Мережа будується з вузлів, кожен з яких з'єднаний з усіма іншими вузлами. У кожного нейрона

поріг активації змінюється з часом і є речовим числом. Кожне з'єднання має змінну дійсну вагу. Вузли поділяються на вхідні, вихідні та приховані.

Для навчання з учителем з дискретним часом, кожен (дискретний) крок часу на вхідні вузли подають дані, а інші вузли завершують свою активацію, і вихідні сигнали готують для передання нейроном наступного рівня. Якщо, наприклад, мережа відповідає за розпізнавання мови, у результаті на вихідні вузли надходять уже мітки (розпізнані слова).

У навчанні з підкріпленням (reinforcement learning) немає вчителя, що забезпечує цільові сигнали для мережі, натомість інколи використовується функція придатності[en] або функція оцінки (reward function), за якою проводять оцінювання якості роботи мережі, при цьому значення на виході впливає на поведінку мережі на вході. Зокрема, якщо мережа реалізує гру, на виході вимірюється кількість пунктів виграшу або оцінки позиції. Кожен ланцюжок обчислює помилку як сумарну девіацію за вихідними сигналами мережі. Якщо є набір зразків навчання, помилка обчислюється з урахуванням помилок кожного окремого зразка.

1.5 Проблема довгострокових залежностей

Одна з ідей, яка робить РНЗ настільки привабливими, полягає в тому, що вони могли б використовувати отриману в минулому інформацію для поточних завдань. Наприклад, вони могли б використати попередні кадри відео для розуміння наступних. Іноді нам достатньо нещодавньої інформації для виконання поточного завдання. Наприклад, представимо модель мови, яка намагається передбачити наступне слово, ґрунтуючись на попередніх. Якщо ми намагаємося передбачити останнє слово у реченні «Хмари на небі», нам не потрібен більше ніякий контекст – досить очевидно, що наприкінці пропозиції йдеться про небо. У таких випадках, де невеликий проміжок між необхідною інформацією та місцем, де вона

потрібна, РНЗ можуть навчитися використовувати інформацію, отриману раніше (рисунок 1.6).

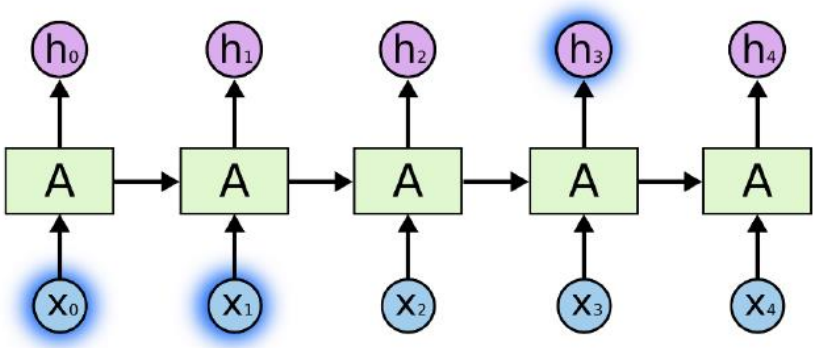

Рисунок 1.6 – Довгострокові залежності

Але також трапляються випадки, коли нам потрібен ширший контекст. Припустимо, потрібно передбачити останнє слово у тексті «Я виріс у Франції… Я вільно розмовляю французькою». Нещодавня інформація підказує, що наступне слово, ймовірно, назва мови, але якщо ми хочемо уточнити, якої саме, нам потрібен попередній контекст аж до інформації про Францію. Зовсім нерідко проміжок між необхідною інформацією та місцем, де вона потрібна, стає дуже великим. На жаль, у міру зростання проміжку РНС стають нездатними навчитися з'єднувати інформацію (рисунок 1.7).

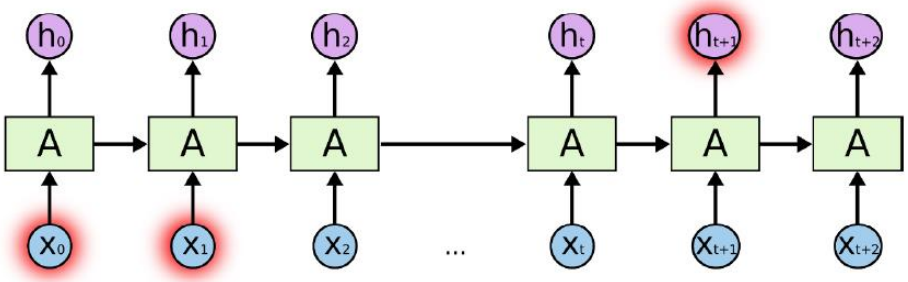

Рисунок 1.7 – Проблема поєднання

Теоретично РНС здатні обробляти такі довгострокові залежності. Людина може ретельно підібрати їх параметри, щоб вирішувати іграшкові проблеми такої форми. Проте, практично, РНС не здатні вивчити таке.

1.6 Різновиди LSTM мереж

Мережі довготривалої пам'яті (Long Short Term Memory) – зазвичай просто називають «LSTM» – особливий вид РНС, здатних до навчання довгострокових залежностей. Вони працюють неймовірно добре на великому розмаїтті проблем і зараз широко застосовуються. LSTM спеціально спроектовано таким чином, щоб уникнути проблеми довгострокових залежностей. Запам'ятовувати інформацію на тривалий період часу – це практично їхня поведінка за замовчуванням, а не щось таке, що вони тільки намагаються зробити.

У мережах LSTM вдалося обійти проблему зникнення або зашкалювання градієнтів у процесі навчання методом зворотного поширення помилки. Мережа LSTM зазвичай управляється за допомогою рекурентних вентилів, які називаються вентилі (gates) «забування». Помилки поширюються назад у часі через потенційно необмежену кількість віртуальних шарів. Таким чином відбувається навчання в LSTM, при цьому зберігаючи пам'ять про тисячі і навіть мільйони часових інтервалів у минулому. Топології мереж типу LSTM можуть розроблятися відповідно до специфіки завдання. У мережі LSTM навіть великі затримки між значними подіями можуть враховуватися, і тим самим високочастотні та низькочастотні компоненти можуть поєднуватися.

Усі рекурентні нейронні мережі мають форму ланцюга повторювальних модулів (repeating module) нейронної мережі (рисунок 1.8). У стандартній РНС ці модулі, що повторюють,

матимуть дуже просту структуру, наприклад, всього один шар гіперболічного тангенсу (tanhtanh).

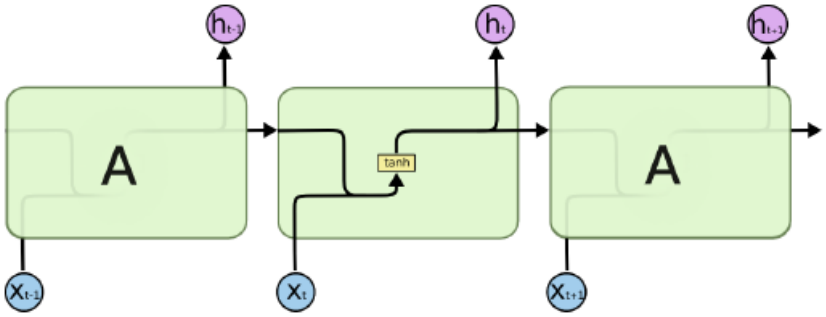

Рисунок 1.8 – Ланцюгова структура

LSTM теж мають таку ланцюгову структуру, але модуль, що повторює, має іншу будову. Замість одного нейронного шару їх чотири, причому вони взаємодіють особливим чином (рисунок 1.9).

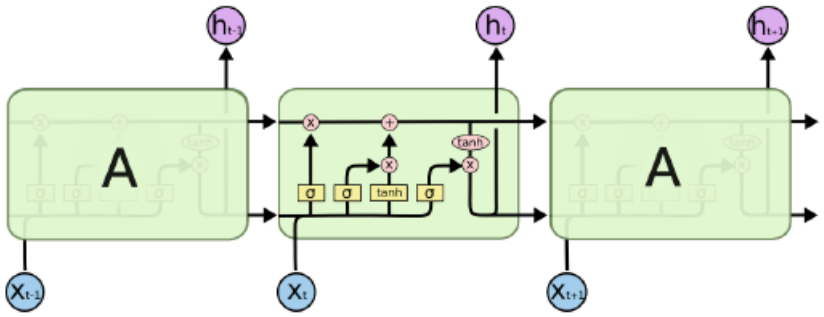

Рисунок 1.9 – LSTM-модель

У діаграмі вище, кожна лінія передає цілий вектор від виходу одного вузла до входів інших. Рожеві кола представляють поточкові оператори, такі як складання векторів, тоді як жовті У діаграмі вище кожна лінія передає цілий вектор від виходу одного вузла до входів інших. Рожеві кола представляють поточкові оператори, такі як складання векторів, тоді як жовті прямокутники – це навчені шари нейронної мережі. Лінії, що

зливаються, позначають конкатенацію, в той час як лінії, що гілкуються, позначають, що їх вміст копіюється, і копії відправляються в різні місця.

1.7 Прогнозування часових рядів

Завдання прогнозування часових рядів – складний тип проблеми прогнозуючого моделювання. На відміну від регресійного передбачуваного моделювання, тимчасові ряди також додають складність залежності послідовності від вхідних змінних. Потужний тип нейронної мережі, призначений для обробки послідовностей, називається рекурентними нейронними мережами. Мережа з

Довгою короткою пам'яттю або мережа LSTM – це тип рекурентної нейронної мережі, що використовується в глибокому навчанні, тому що можна успішно навчати дуже великі архітектури.

Завдання, яке ми розглянемо – проблема прогнозування пасажирських авіаперевезень. Завдання полягає в тому знаючи рік і місяць передбачити кількість пасажирів міжнародних авіакомпаній. Набір даних доступний безкоштовно можна завантажити з адреси https://datamarket.com/data/set/22u3/international-airline-passengers-monthly-totals-in-thousands-jan-49-dec-60#!ds=22u3&display=line з ім'ям файлу «international-airlines-passengers.csv». Дані варіюються від січня 1949 року до грудня 1960 року або 12 років із 144 спостереженнями. Ми можемо завантажити цей набір даних за допомогою бібліотеки Pandas. Нам не цікава дата з огляду на те, що кожне спостереження поділяється одним і тим же інтервалом в один місяць. Тому, коли завантажуємо набір даних, ми можемо виключити перший стовпець. Після завантаження ми можемо легко збудувати весь набір даних.

З часом можна побачити висхідний тренд у наборі даних та деяку періодичність для набору даних, який, ймовірно, відповідає періоду відпустки у північній півкулі (рисунок 1.10).

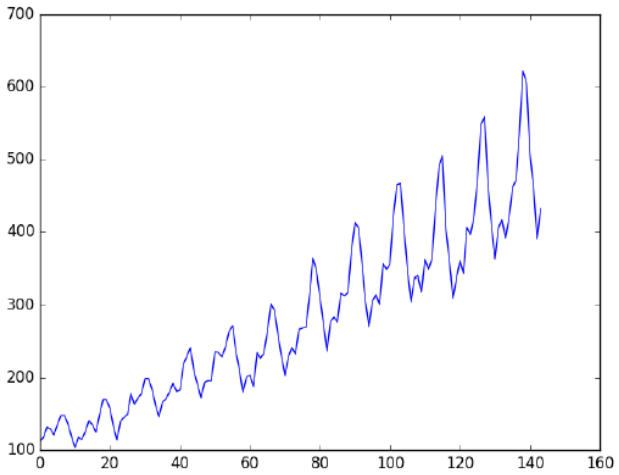

Рисунок 1.10 – Побудова діаграми даних

Ми можемо сформулювати це завдання як завдання регресії. Тобто з огляду на кількість пасажирів (у тисячах одиниць) цього місяця прогнозувати кількість пасажирів наступного місяця. Ми можемо написати просту функцію, щоб перетворити наш єдиний стовпець даних на двостовпцевий набір даних: перша колонка, що містить кількість пасажирів і другий стовпець, який міститиме кількість пасажирів наступного місяця.

LSTM чутливі до шкали вхідних даних, особливо коли використовуються сигмоїдні (за замовчуванням) або функції активації tanh. Тому необхідно провести масштабування даних до діапазону від 0 до 1, який також називається нормалізацією. Ми можемо легко нормалізувати набір даних, використовуючи клас попередньої обробки MinMaxScaler із бібліотеки scikit-learn.

scaler = MinMaxScaler(feature_range=(0, 1))

dataset = scaler.fit_transform(dataset) Після того, як ми моделюємо наші дані та оцінимо якість нашої моделі на навчальному наборі даних, нам потрібно дізнатися наскільки точний прогноз на даних, які мережа не бачила. Для звичайної задачі класифікації чи регресії ми робитимемо це з використанням перехресної перевірки. При використанні часових рядів важлива послідовність значень. Простим методом, який ми можемо використовувати, є поділ впорядкованого набору даних на навчальний та тестовий набор даних. Нижче наведений код поділяє дані на навчальні набори даних з 67% спостережень, які ми можемо використовувати для навчання нашої моделі, залишаючи 33% для тестування моделі. train_size = int(len(dataset) * 0.67) test_size = len(dataset) – train_size train, test = dataset[0:train_size,:], dataset[train_size:len(dataset),:] Тепер визначимо функцію створення нового набору даних, як описано вище. Функція приймає два аргументи: набір даних, який являє собою масив NumPy, який ми хочемо перетворити на набір даних, і look_back, який є числом попередніх кроків часу для використання в якості вхідних змінних для прогнозування наступного періоду часу – в цьому випадку за замовчуванням – 1. Це значення за промовчанням створить набір даних, де X – кількість пасажирів у заданий час (t), а Y – кількість пасажирів наступного часу (t + 1), у лістингу 1.1.

Лістинг 1.1

```
def create_dataset(dataset, look_back=1):
dataX, dataY = [], []
for i in range(len(dataset)-look_back-1):
a = dataset[i:(i+look_back), 0]
dataX.append(a)
dataY.append(dataset[i + look_back, 0])
return numpy.array(dataX), numpy.array(dataY)
```

Далі використовуємо цю функцію для підготовки наборів даних для навчання та для тестування нейронної мережі та перетворюємо дані у структуру, що відповідає входу нейронної мережі (лістинг 1.2).

Лістинг 1.2

```
look_back = 1
trainX, trainY = create_dataset(train, look_back)
testX, testY = create_dataset(test, look_back)
trainX = numpy.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))
testX = numpy.reshape(testX, (testX.shape[0], 1, testX.shape[1]))
```

Через те, як було підготовлено набір даних, ми повинні зрушити передбачення так, щоб вони вирівнялися по осі x з вихідним набором даних (рисунок 1.11).

```
trainPredictPlot = numpy.empty_like(dataset)
trainPredictPlot[:, :] = numpy.nan
trainPredictPlot[look_back:len(trainPredict)+look_back, :] = trainPredict
testPredictPlot = numpy.empty_like(dataset)
testPredictPlot[:, :] = numpy.nan
testPredictPlot[len(trainPredict)+(look_back*2)+1:len(dataset)-1, :] = testPredict
plt.plot(scaler.inverse_transform(dataset))
plt.plot(trainPredictPlot)
plt.plot(testPredictPlot)
plt.show()
```

Рисунок 1.11 – Набір даних

Вихідний набір даних відображається синім кольором, прогнози для набору навчальних даних зеленим кольором та прогнози тестового набору даних червоним кольором. Ми, що модель чудово впоралася з прогнозом як навчальному, і на тестовому наборі даних (рисунок 1.12).

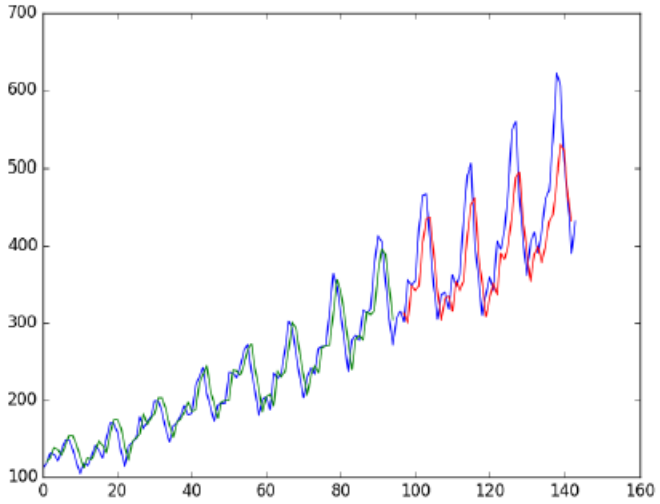

Рисунок 1.12 – Побудова діаграми даних та результату прогнозу

2 НЕЙРОННІ МЕРЕЖІ НА ОСНОВІ БІБЛІОТЕКИ TENSORFLOW

2.1 Основи роботи в TensorFlow

TensorFlow – це бібліотека програмного забезпечення з відкритим кодом, створена Google, яка використовується для впровадження систем машинного навчання та глибокого навчання. Ці два імені містять низку потужних алгоритмів, які поділяють загальне завдання – дозволити комп'ютеру дізнатися, як автоматично визначати складні шаблони та/або приймати найкращі можливі рішення. TensorFlow, в основі своєї, є бібліотекою для програмування потоку даних. Він використовує різні методи оптимізації, щоб зробити обчислення математичних виразів простіше та ефективніше.

Деякі з ключових особливостей TensorFlow:

– ефективно працює з математичними виразами, що включають багатовимірні масиви;

– оптимальна підтримка глибоких нейронних мереж та концепцій машинного навчання;

– використання GPU/CPU, де один і той же код може бути виконаний на обох архітектурах;

– висока масштабованість обчислень на машинах та величезні масиви даних.

Встановити TensorFlow можна за допомогою команди: pip install tensorflow

Завантаження та складання TensorFlow може тривати кілька хвилин.

У TensorFlow обчислення описується з використанням графів потоків даних. Кожен вузол графа являє собою екземпляр математичної

операції (наприклад, додавання, поділ або множення), і кожне ребро є багатовимірним набором даних (тензор), на якому виконуються операції (рисунок 2.1).

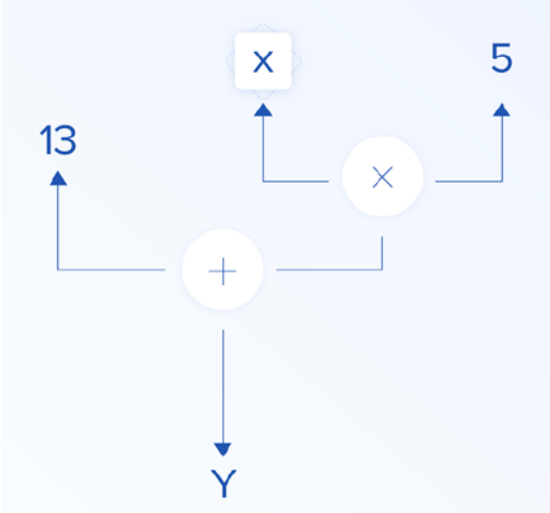

Рисунок 2.1 – Графи потоків даних

У TensorFlow константи створюються за допомогою функції: constant(value, dtype=None, shape=None, name='Const', verify_shape=False), де value постійне значення, яке буде використовуватися при подальших обчисленнях, dtype є параметром, що вказує тип даних (наприклад, float32/64, int8/16), shape є необов'язковим параметром, що вказує розмір масиву даних, name є необов'язковим ім'ям для тензора. Якщо вам потрібні константи з певними значеннями всередині вашої навчальної моделі, тоді об'єкт типу constant може використовуватися як: z = tf.constant(5.2, name=«x», dtype=tf.float32).

Змінні TensorFlow є буферами в пам'яті, що містять тензори, які повинні бути явно ініціалізовані. Просто викликаючи конструктор, ми додаємо змінну обчислювальний граф. Змінні особливо корисні, як тільки ви починаєте з моделей навчання, і вони використовуються для зберігання та оновлення параметрів. Початкове значення, передане як аргумент

конструктора, являє собою тензор або об'єкт, який може бути перетворений або повернутий як тензор. Це означає, що якщо ми хочемо заповнити зміну деякими зумовленими чи випадковими значеннями, які будуть використовуватись згодом у процесі навчання та оновлені при ітераціях, ми можемо визначити її наступним чином: k = tf.Variable(tf.zeros([1]), name=«k»)

Інший спосіб використання змінних у TensorFlow – це обчислення, коли ця змінна не є навчальною і може бути визначена таким чином: k = tf.Variable(tf.add(a, b), trainable=False)

Сеанс TensorFlow інкапсулює керування та стан середовища виконання TensorFlow. Сеанс без параметрів використовуватиме стандартний граф, створений у поточному сеансі, інакше клас сеансу приймає параметр графа, який використовується в цьому сеансі для виконання. Нижче наведено короткий фрагмент коду, де показано, як терміни, визначені вище, можуть використовуватися в TensorFlow для обчислення простої лінійної функції y=a*x+b, у лістингу 2.1.

Лістинг 2.1

```
import tensorflow as tf
x = tf.constant(-2.0, name="x", dtype=tf.float32)
a = tf.constant(5.0, name="a", dtype=tf.float32)
b = tf.constant(13.0, name="b", dtype=tf.float32)
y = tf.Variable(tf.add(tf.multiply(a, x), b))
init = tf.global_variables_initializer()
with tf.Session() as session:
session.run(init)
print(session.run(y))
```

2.2 Визначення обчислювальних графів у TensorFlow

Перевага під час роботи з графами потоків даних у тому, що модель відділена її виконання, тобто. неважливо, на якому обчислювальному пристрої виконується програмний код, на процесорі, графічному процесорі або деякій комбінації. Програма серед TensorFlow може виконуватися на процесорі чи графічному процесорі, а вся складність, пов'язана з кодом виконання прихована від користувача. Граф обчислення - це вбудований процес, який використовує бібліотеку без прямого виклику об'єкта графа. Граф обчислень може бути побудований у процесі використання бібліотеки TensorFlow без необхідності створювати об'єкти Graph.

Об'єкт Graph TensorFlow може бути створений в результаті простого рядка коду c = tf.add(a, b). Це код створить операційний вузол, який приймає два тензори a і b які обчислюють їхню суму c як результат.

Плейсхолдер - це спосіб, що дозволяє розробникам вводити дані до графіка обчислень через заповнювачі, які пов'язані всередині деяких виразів. Сигнатура плейсхолдера: placeholder(dtype, shape=None, name=None), де dtype – тип елементів у тензорах.

Перевага плейсхолдерів полягає в тому, що вони дозволяють розробникам створювати операції у обчислювальному графі взагалі, без необхідності надавати заздалегідь дані, і дані можуть бути додані під час виконання із зовнішніх джерел.

Розглянемо просте завдання множення двох цілих чисел x і y у TensorFlow та використанням плейсхолдера (лістинг 2.2).

Лістинг 2.2

```
import tensorflow as tf
x = tf.placeholder(tf.float32, name="x")
y = tf.placeholder(tf.float32, name="y")
```

Продовження лістингу 2.2

```
z = tf.multiply(x, y, name="z")
with tf.Session() as session:
print(session.run(z, feed_dict={x: 2.1, y: 3.0}))
```

2.3 Візуалізація обчислювального графа за допомогою TensorBoard

TensorBoard – інструмент візуалізації для аналізу графіків потоку даних. Він може бути корисним для кращого розуміння моделей машинного навчання. З TensorBoard ви можете отримати уявлення про різні типи статистики про параметри та подробиці про частини обчислювального графа. Глибока нейронна мережа має велику кількість вузлів. TensorBoard дозволяє розробникам отримати уявлення про кожен вузл і про те, як обчислення виконується під час виконання TensorFlow.

Тепер давайте повернемося до нашого прикладу, де ми визначили лінійну функцію y = a*x + b.

Щоб реєструвати події з сеансу, які пізніше можна використовувати в TensorBoard, TensorFlow надає клас FileWriter. Його можна використовувати для створення файлу подій для зберігання зведень та подій. Конструктор FileWriter приймає шість параметрів і має такий вигляд: __init__(logdir, graph=None, max_queue=10, flush_secs=120, graph_def=None, filename_suffix=None) - де потрібно вказати обов'язковий параметр logdir, інші - значення за промовчанням. Параметр графа буде передано з об'єкта сеансу, створеного у програмі. Повний код прикладу виглядає наступним чином (лістинг 2.3)

Лістинг 2.3

```
import tensorflow as tf
x = tf.constant(-2.0, name="x", dtype=tf.float32)
```

Продовження лістингу 2.3

```
a = tf.constant(5.0, name="a", dtype=tf.float32)
b = tf.constant(13.0, name="b", dtype=tf.float32)
y = tf.Variable(tf.add(tf.multiply(a, x), b))
init = tf.global_variables_initializer()
with tf.Session() as session:
merged = tf.summary.merge_all()
writer = tf.summary.FileWriter("logs", session.graph)
session.run(init)
print(session.run(y))
```

Ми додали два нові рядки, у яких створюється і використовується об'єкт FileWriter для виведення подій у файл, як описано вище.

Після запуску програми у нас з'являється файл у журналах каталогів, і щоб подивитися вміст цього файлу необхідно запустити tensorboard: tensorboard --logdir logs/

Після відкриття http://localhost:6006 та натискання на пункт меню «Графіки» (розташований у верхній частині сторінки) ви зможете побачити графік, як на рисунку 2.2.

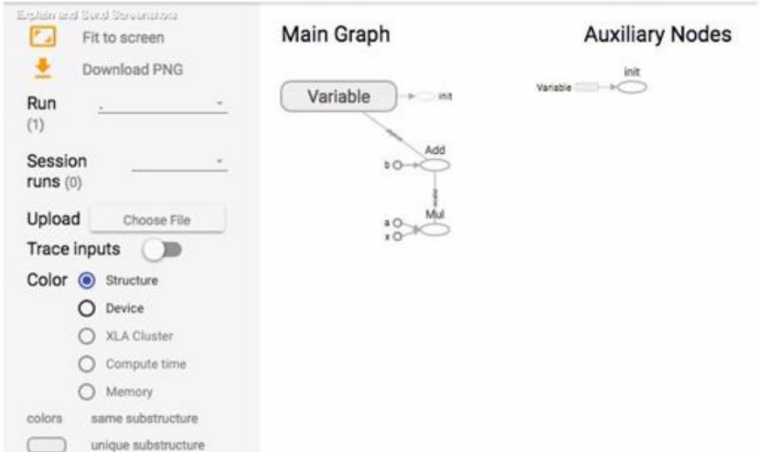

Рисунок 2.2 – Налаштування моделі

TensorBoard маркує константи та зведені вузли конкретними символами, що показані на рисунку 2.3.

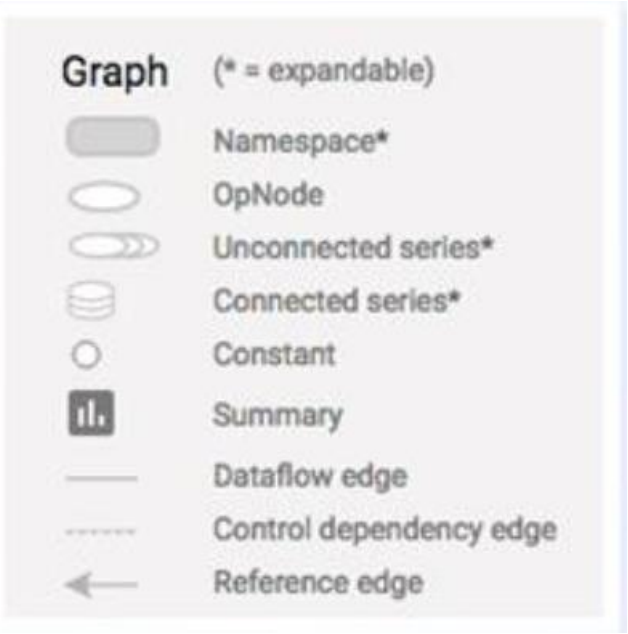

Рисунок 2.3 – Компоненти моделі потоків даних

2.4 Математика з TensorFlow

Тензори – це основні структури даних в TensorFlow, і вони є сполучними грані в графі потоку даних.

Тензор просто ідентифікує багатовимірний масив чи список. Тензорну структуру можна ідентифікувати трьома параметрами: рангом, розміром та типом.

Ранг: Визначає кількість вимірів тензора. Ранг відомий як порядок або n-розмірності тензора, де, наприклад, тензор рангу 1 є тензор вектора або рангу 2, є матрицею.

Розмір: Розмір тензора – це кількість рядків та стовпців.

Тип: Тип даних елементів тензора.

Щоб побудувати тензор TensorFlow, ми можемо побудувати n-мірний масив. Це можна зробити легко, використовуючи бібліотеку NumPy, або шляхом перетворення n-вимірного масиву Python в тензор TensorFlow (рисунок 2.4).

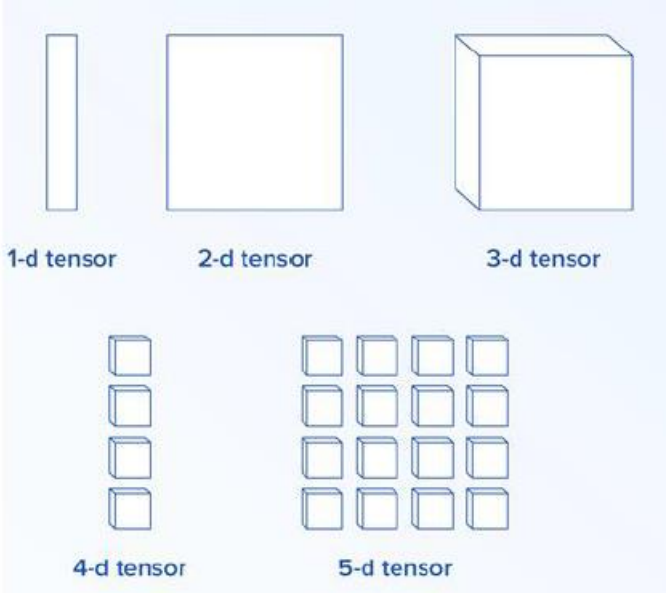

Рисунок 2.4 – Побудова тензора

Щоб побудувати 1-мірний тензор, ми будемо використовувати масив NumPy:

import numpy as np

tensor_1d = np.array([1.45, -1, 0.2, 102.1])

Робота з подібним масивом аналогічна роботі із вбудованим списком Python. Основна відмінність полягає в тому, що масив NumPy також з масивом NumPy можна легко перетворити в тензор TensorFlow використовуючи допоміжну функцію convert_to_tensor, що допомагає розробникам перетворювати об'єкти Python на об'єкти тензора. Ця функція приймає тензорні об'єкти, масиви NumPy та списки Python.

tensor = tf.convert_to_tensor(tensor_1d, dtype=tf.float64)

Тепер, якщо прив'яжемо наш тензор до сеансу TensorFlow, ми зможемо побачити результати нашого перетворення (лістинг 2.4).

Лістинг 2.4

```
import numpy as np
import tensorflow as tf
tensor_1d = np.array([1.45, -1, 0.2, 102.1])
tensor = tf.convert_to_tensor(tensor_1d, dtype=tf.float64)
with tf.Session() as session:
print(session.run(tensor))
print(session.run(tensor[0]))
print(session.run(tensor[1]))
```

Висновок:

```
[1.45 -1. 0.2 102.1]
1.45
-1.0
```

2.5 Тензорні операції

Рекурентні нейронні мережі (RNN) є багатим класом динамічних моделей, які використовуються для генерації послідовностей у таких різних областях, як музика, текст і дані захоплення руху. RNN можуть бути навчені для генерації послідовностей шляхом обробки послідовностей реальних даних по одному кроку за раз і прогнозування того, що буде далі. Припускаючи, що прогнози є ймовірнісними, нові послідовності можуть бути згенеровані з навченої мережі шляхом ітеративної вибірки вихідного розподілу мережі, а потім подачі вибірки як вхідні дані на наступному кроці. Хоча сама мережа є детермінованою,

стохастичність, запроваджена під час відбору вибірок, викликає розподіл за послідовностями. Цей розподіл є умовним, оскільки внутрішній стан мережі

Рекурентні нейронні мережі «нечіткі» у тому плані, що вони не використовують точні шаблони з даних для навчання, щоб передбачити, а скоріше використовують багатовимірну інтерполяцію між прикладами навчання. Рекурентні нейронні мережі, на відміну від шаблонних, складним шляхом синтезують та відновлюють навчальні дані та рідко генерують одні й ті самі речі двічі. Більше того, нечіткі передбачення не страждають від «прокляття розмірності», і тому вони набагато краще підходять для моделювання реальних чи багатовимірних даних, ніж точні збіги (рисунок 2.5).

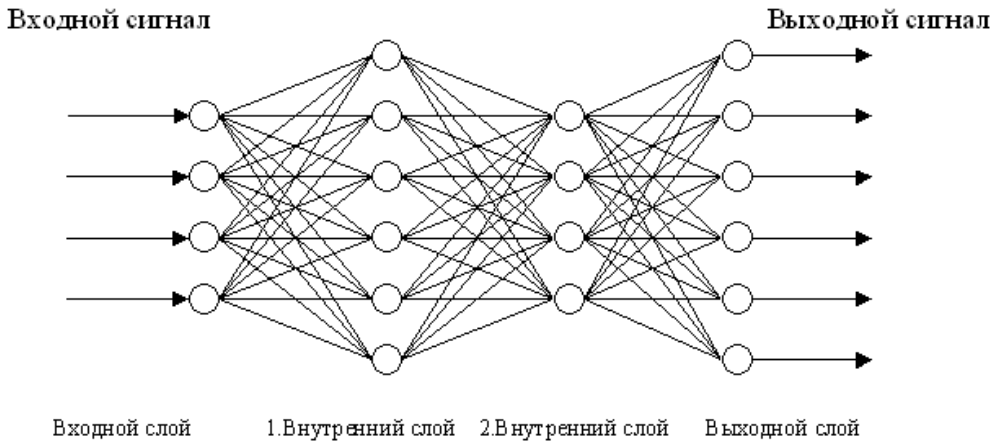

Рисунок 2.5 – Архітектура рекурентної нейронної мережі

Те, що рекурентні нейронні мережі дають хороші результати під час роботи з різними послідовностями зумовлено особливостями їхньої архітектури. Рекурентні нейронні мережі (Recurrent Neural Networks, RNN) – це мережі, що містять зворотні зв'язки та дозволяють зберігати інформацію. Рекурентну мережу можна розглядати як кілька копій однієї і

тієї ж мережі, кожна з яких передає інформацію наступної копії. Якщо її розгорнути, ми побачимо, що RNN нагадують ланцюжок (див. рис. 2.6). Це і говорить про те, що вони тісно пов'язані із послідовностями та списками. Таким чином, RNN – найприродніша архітектура нейронних мереж для роботи з даними такого типу (рисунок 2.6).

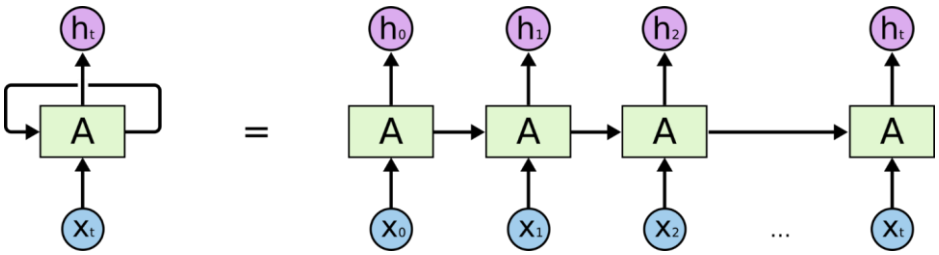

Рисунок 2.6 – Подання рекурентної нейронної мережі у вигляді послідовності

Довгострокова короткочасна пам'ять (LSTM – Long short-term memory) – це архітектура RNN, призначена для покращеного зберігання та доступу до інформації, ніж стандартні RNN, т.к. вона здатна до навчання довгострокових залежностей.

LSTM-модуль – це рекурентний модуль мережі, здатний запам'ятовувати значення як у короткі, і довгі проміжки часу. Ключем до цієї можливості є те, що LSTM-модуль (рисунок 2.7) не використовує функції активації всередині своїх рекурентних компонентів. Таким чином, значення, що зберігається, не розмивається в часі, і градієнт або штраф не зникає при використанні методу зворотного поширення помилки в часі при тренуванні мережі.

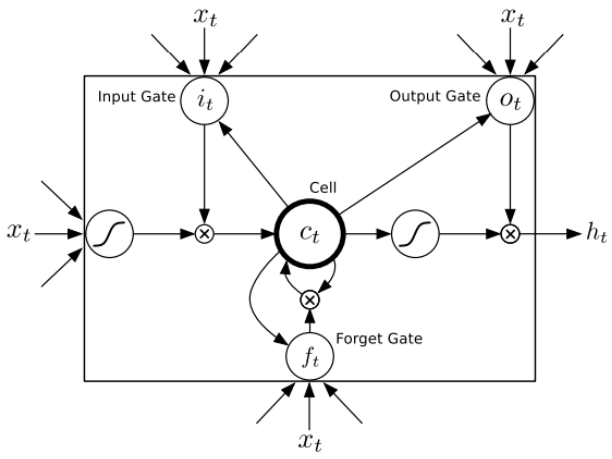

Рисунок 2.7 – Клітка довгої короткострокової пам'яті

Стан осередку схожий на конвеєрну стрічку (рисунок 2.7), яка проходить безпосередньо через весь ланцюжок і бере участь лише в деяких лінійних перетвореннях. По ній інформація може текти без перешкод і не зазнавати жодних змін. Однак LSTM може видаляти інформацію зі стану комірки. Причому операція видалення регулюється спеціальними структурами, які називаються фільтрами (gates) (рисунок 2.8).

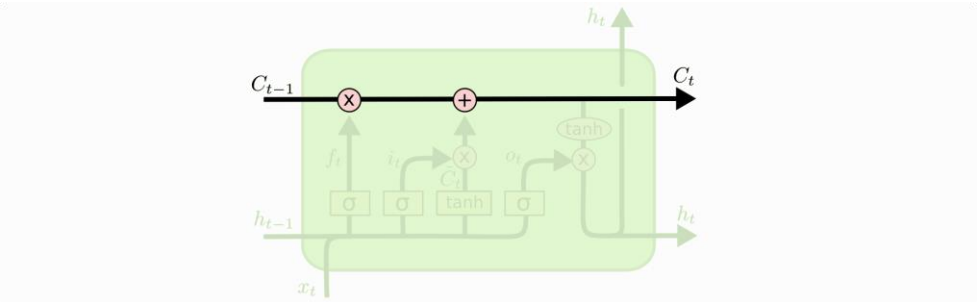

Рисунок 2.8 – Стан осередку – конвеєрна стрічка

Основою фільтрації є деякі умови, що дозволяють пропускати інформацію. Вони складаються із двох об'єктів. Ними є шар сигмоїдальної нейронної мережі та операція крапкового множення. Сигмоїдальний шар

повертає числа від нуля до одиниці, які позначають, яку частку кожного блоку інформації слід пропустити далі через мережу. Нуль у разі означає «не пропускати нічого», одиниця – «пропустити все». У LSTM включає три фільтри, які дозволяють контролювати і захищати стан комірки (рисунок 2.9).

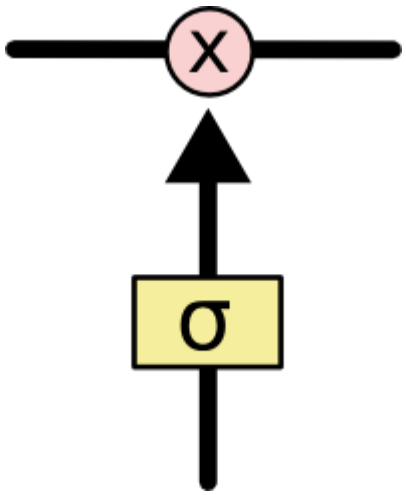

Рисунок 2.9 – Фільтр у LSTM

LSTM зараз дає передові результати в різних завданнях обробки послідовності, включаючи розпізнавання мови і почерку.

На рисунку 2.10 зображено рекурентну нейронну мережу прогнозування. Вхідна векторна послідовність$x(1)$передається через зважені з'єднання в стек з N рекурентно пов'язаних прихованих шарів, щоб спочатку обчислити приховані векторні послідовності$h^n(2)$, а потім вихідну векторну послідовність $y(3)$.

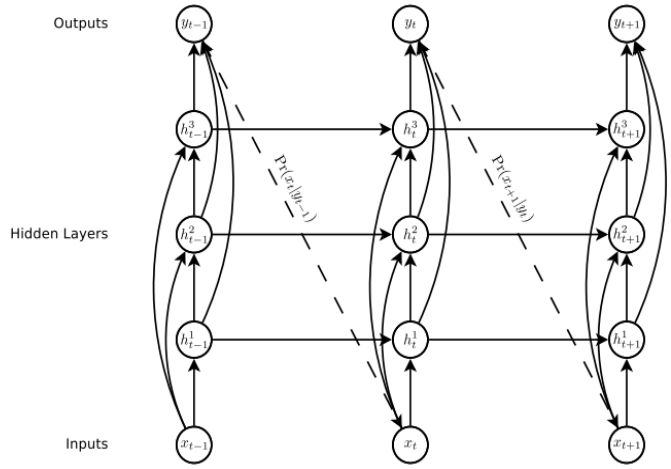

Рисунок 2.10 – Архітектура рекурентної нейронної мережі, де кола є шарами мережі, суцільні лінії – зважені зв'язки, а пунктирні – прогнози

$$x = (x_1, \dots, x_T) (1)$$

$$h^n = (h_1^n, \dots, h_T^n) (2)$$

$$y = (y_1, \dots, y_T) (3)$$

Кожен вихідний векторy_tвикористовується для того, щоб параметризувати прогнозуючий розподіл по всіх можливих входах. Перший елемент кожної вхідної послідовності завжди є нульовим вектором, всі записи якого дорівнюють нулю; отже, мережа генерує прогноз першого реального входу без попередньої інформації. Мережа є «глибокої» як у просторі, і у часі, тому, що у кожен фрагмент інформації, що проходить або вертикально, чи горизонтально через обчислювальний граф, впливатимуть численні послідовні вагові матриці і нелінійності.$Pr(x_{t+1} \vee y_t) x_{t+1} x_1 x_2$

Необхідно звертати увагу на «пропуск з'єднань» від входів до всіх прихованих шарів та від усіх прихованих шарів до виходів. Це значно полегшує процес навчання для глибоких мереж, зменшуючи кількість етапів обробки між дном мережі та верхом та, таким чином, пом'якшуючи проблему зникаючого градієнта. В особливому випадку, коли архітектура

зводиться до звичайного однорівневого RNN (recurrent neural network $N = 1$– Рекурентної нейронної мережі) з прогнозуванням наступного кроку. Активації прихованого шару обчислюються шляхом ітерації наступних рівнянь від до і до : $t = 1Tn = 2N$

$$h_1^t = H(W_{ih^1}x_t + W_{h^1h^1}h_{t-1}^1 + b_n^1), \quad (2.1)$$

$$h_t^n = H(W_{ih^n}x_t + W_{h^{n-1}h^n}h_t^{n-1} + W_{h^nh^n}h_{t-1}^n + b_n^1), \quad (2.2)$$

де W– вагові матриці (наприклад, – вагова матрицяW_{ih^n}, що з'єднує входи 3-nм прихованим шаром; $W_{h^1h^1}$, – рекурентне з'єднання на першому прихованому шарі і т.д.);

b – вектору зміщення (bias vector);

H – Функція прихованого шару.

З урахуванням прихованих послідовностей, вихідні послідовності розраховуються так:

$$\hat{y}_t = b_y + \sum_{n=1}^{N} W_{h^ny}h_t^n, \quad (2.3)$$

$$y_t = Y(\hat{y}_t), \quad (2.4)$$

де Y – функція вихідного шару. Таким чином, вся мережа визначає функцію, параметризовану ваговими матрицями, від вхідних історій до вихідних векторів$x1: ty_t$.

Вихідні вектори використовуються для параметризації прогнозуючого розподілу для наступного входу. Форма має бути підібрана свідомо, спираючись на вхідні дані. Зокрема, знаходження відповідного прогнозуючого розподілу для багатовимірних реальних даних (зазвичай згадується як «моделювання щільності») може бути досить скрутним.

$$y_t Pr(x_{t+1} \vee y_t) Pr(x_{t+1} \vee y_t). \quad (2.5)$$

Імовірність, задана мережею для вхідної послідовності:

$$Pr(x) = \prod_{t=1}^{T} P\, r(x_{t+1} \vee y_t)\,. \quad (2.6)$$

Втрати послідовності, що використовується для того, щоб навчити мережу, є негативним логарифмом від :$L(x)Pr(x)$

$$L(x) = -\sum_{t-1}^{T} log\, Pr(x_{t+1} \vee y_t). \quad (2.7)$$

Приватні похідні втрат за вагами мережі можуть бути ефективно розраховані зі зворотним розповсюдженням помилки у часі, застосованим до графа обчислень, показаного на рисунку і мережа потім може бути навчена за допомогою алгоритму градієнтного спуску.

У більшості RNN функція прихованого шару є поелементним застосуванням сигмоїдальної функції. Для версії LSTM реалізована за допомогою наступної складової функції:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i), \quad (2.8)$$

$$f_t = \sigma\big(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\big), \quad (2.9)$$

$$c_t = f_t c_{t-1} + i_t tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c), \quad (2.10)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o), \quad (2.11)$$

$$h_t = o_t tanh(c_t), \quad (2.12)$$

Де σ – функція логістичної сигмоїди, а й i, f, oc-відповідно, вхідні вектори активації для вхідного фільтра, що пропускає фільтра, тобто. фільтра, який здатний видаляти інформацію зі стану комірки, вихідного фільтра, а також комірки та входу в комірку (input gate, forget gate, output gate, cell and cell input), кожен з яких має той же розмір, що і прихований вектор h.

В оригінальному алгоритмі LSTM використовується спеціально розроблений розрахунок приблизного градієнта, який дає змогу оновлювати ваги після кожного тимчасового кроку. Однак натомість повний градієнт може бути розрахований зі зворотним поширенням помилки у часі, метод, який буде використовуватися далі.

Одна з труднощів при навчанні LSTM з повним градієнтом полягає в тому, що похідні іноді стають надмірно великими, що призводить до чисельних проблем. Щоб запобігти цьому, можна обрізати похідну втрат по відношенню до виходів мережі в шари LSTM (до застосування функцій сигмоїду та гіперболічного тангенсу), щоб вони знаходилися в заздалегідь визначеному діапазоні.

2.6 Матричні операції

Матричні операції є дуже важливими для моделей машинного навчання, таких як лінійна регресія, оскільки вони часто використовуються в них. TensorFlow підтримує всі найпоширеніші операції з матрицями, такі як множення, інверсія, обчислення визначника, рішення лінійних рівнянь та багато іншого. Давайте напишемо деякий код, який виконуватиме базові матричні операції, такі як множення, транспонування, обчислення визначника, множення та інші.

Нижче наведено основні приклади виклику цих операцій (лістинг 2.5).

Лістинг 2.5

```
import tensorflow as tf
import numpy as np
def convert(v, t=tf.float32):
return tf.convert_to_tensor(v, dtype=t)
m1 = convert(np.array(np.random.rand(4, 4), dtype='float32'))
```

TensorFlow підтримує різні види редукції. Редукція – це операція, яка видаляє один або кілька вимірів із тензора, виконуючи певні операції з цих вимірів. Ми представимо кілька з них у наведеному нижче прикладі (лістинг 2.6).

Лістинг 2.6

```
import tensorflow as tf
import numpy as np
def convert(v, t=tf.float32):
return tf.convert_to_tensor(v, dtype=t)
x = convert (
np.array( [ (1, 2, 3), (4, 5, 6), (7, 8, 9) ]), tf.int32)
```

Сегментація – це процес, в якому одним із вимірів є процес відображення розмірів на надані сегментні індекси, а результуючі елементи визначаються рядком індексу.

Сегментація групує елементи під повторними індексами, тому, наприклад, у нашому випадку ми маємо сегментовані ids, [0, 0, 1, 2, 2] застосовувані до тензору tens1, що означає, що перший і другий масиви будуть перетворені після операції сегментації (у нашому випадку підсумовування) та отримаємо новий масив, який виглядає (2, 8, 1, 0) = (2 + 0,5 + 3,3 -2 -5+5). Третій елемент у тензорі tens1

недоторканий, тому що він не згрупований в жодний повторний індекс, а останні два масиви підсумовуються так само, як і у випадку першої групи (рисунок 2.11).

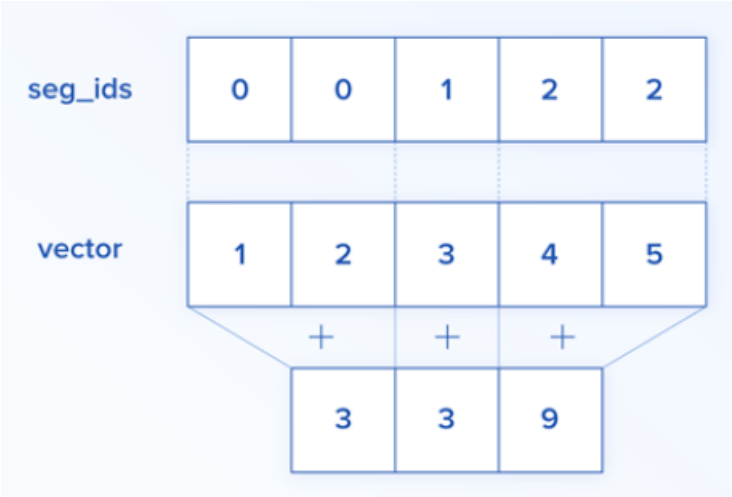

Рисунок 2.11 – Сегментація

TensorFlow містить такі методи, як:

– argmin, яка повертає індекс з мінімальним значенням по осях вхідного тензора;

– argmax, яка повертає індекс з максимальним значенням по осях вхідного тензора;

– setdiff, який обчислює різницю між двома списками чисел чи рядків.

3 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНИХ МОДЕЛЕЙ І РОЗРОБКА АЛГОРИТМІВ ОПРАЦЮВАННЯ ПОТОКІВ ДАНИХ

3.1 Порівняльний аналіз сучасних засобів і методів аналізу та обробки великих даних

Джерела збільшення кількості даних можна поділити на кілька категорій:

– цифрові пристрої;

– взаємодія людини із певними інтерфейсами;

– обробка даних.

Як правило, перший тип пов'язаний із поширенням різних пристроїв з інтегрованими датчиками, збільшенням площі покриття стільникового зв'язку та пристроями, що записують звуки, зображення або відео, а також взаємодії автоматизованих систем. Зокрема такі пристрої, як камери, записують відео, мобільні телефони збирають геопросторові дані машини у виробничих лініях промислових систем, обмінюються важливою інформацією під час обробки їх діяльності. Наприклад:

– медична інформація – машини, що записують ЕЕГ (електроенцефалографія), серцебиття, геном;

– мультимедіа – фотографії та відеоролики, завантажені в Інтернет;

– мобільні пристрої – надають геопросторові дані (місце розташування, гіроскоп), а також метадані телефонні дзвінки, повідомлення, використання Інтернету та дані, зібрані мобільними програмами;

– інші пристрої – системи управління складами (WMS), що забезпечують внутрішнє розташування через Wi-Fi, ідентифікацію продуктів або матеріалів, що збереглися, за допомогою BAR, QR (Quick

Response) коди або чіпів RFID (Radio Frequency Identification). Існує багато інших технологій, таких як навігаційні системи, сейсмічна обробка тощо.

Доцільно розглянути інші джерела, породжені діяльністю в Інтернеті загалом. Уявимо набір серверів, які запускають веб-сторінки, які можна використовувати для роздрібного бізнесу. Сервери можуть збирати записи про всі дії власних користувачів та поведінки власників веб-сайтів, користувачів, транзакцій, програм та серверів. Приклад даних, які можуть бути зібрані:

- бізнес-процеси – зміни до облікового запису, купівлі та звіти про проблеми;
- дані кліків-потоку – зберігати інтереси клієнтів;
- конфігураційні файли – збережені конфігурації серверів та програм;
- журнали баз даних – щоб показати, хто змінював бази даних.

Щоб більше усвідомлювати масштаби збору даних, зручно згадати ще один приклад, пов'язаний з набором астрономічних даних. У 2000 році було розпочато проект з відображення неба під назвою Sloan Digital Sky Survey (SDSS). Протягом перших тижнів проекту велику кількість даних було зібрано телескопом; більше даних, ніж будь-коли було зібрано в історії астрономії. Цей набір складав приблизно 140 терабайт даних. Великий синоптичний телескоп (LSST), проект, який планується запустити до Чилі, має збирати однакову кількість даних кожні п'ять днів.

Наступна категорія стосується обміну інформацією для людей. Наприклад, деякі приклади соціальних мереж можуть бути Facebook, Twitter, LinkedIn. Щосекунди ці системи генерують величезну кількість даних, якими оперують мільйони людей.

Обробка даних полягає в тому, щоб розглядати сирі або вже оброблені дані, щоб отримати певний результат або перейти на інший етап процесу розробки, який бере участь у конкретному проекті.

Великі дані можна визначити як обсяги даних, які доступні різною мірою складності, що створюються з різною швидкістю і різним ступенем невизначеності, які не можуть бути оброблені за допомогою традиційних технологій, методів обробки, алгоритмів або будь-яких комерційних рішень.

Стверджується, що є три атрибути, які виділяються як визначення характеристик великих даних: «великий обсяг даних: замість тисяч чи мільйонів рядків Big Data може зайняти мільярд рядків та мільйони стовпців».

Складність типів і структур даних: великі дані відображають різноманітність нових джерел даних, форматів та структур, у тому числі цифрових слідів, що залишаються в Інтернеті та інших цифрових сховищах для подальшого аналізу.

Швидкість створення нових даних та зростання: великі дані можуть описувати дані з високою швидкістю, швидкою передачею даних та аналізом в режимі реального часу. На рисунку 3.1 зображені основні атрибути Big Data.

Великі дані – це сховище для набору масивів даних, таких великих і складних, які важко обробити за допомогою традиційних інструментів управління базою даних або програм обробки даних.

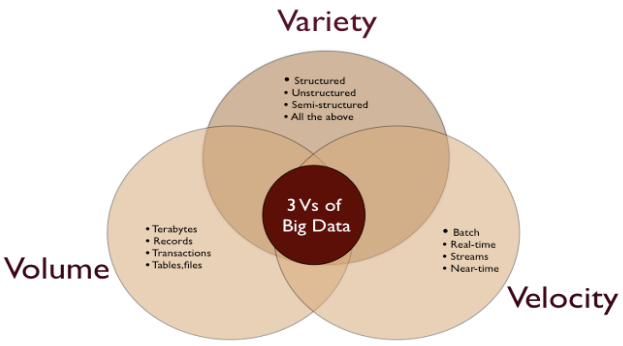

Рисунок 3.1 – Атрибути Big Data

Зазвичай, для опису різних аспектів великих даних використовуються три атрибути:

- об’єм;
- швидкість;
- структурованість.

Ці три атрибути дозволяють визначити природу даних, доступних для аналізу.

Багато можна отримати з текстових даних, даних геолокації або файлів журналів. Наприклад, комунікативні шаблони електронною поштою, споживчі переваги та тенденції, дослідження безпеки. Технології BD пропонують рішення для отримання корисної інформації із цих величезних обсягів даних.

Дані надходять користувачам із великою швидкістю. Інтернет та мобільні технології дозволили компаніям налагодити зворотний зв'язок зі своїми клієнтами.

Технології BD з іншого боку сприяють повному збереженню всіх даних для подальшого аналізу. Головний принцип технологій BD полягає в тому, що в кожному біті даних може бути прихована важлива та цінна інформація.

Раніше для зберігання великих обсягів даних використовувалися реляційні СУБД. Технології BD перевищують реляційні СУБД за низкою критеріїв, включаючи потребу у складніших резервних копіях, відновленні та більш швидких алгоритмів пошуку.

Переваги використання технологій BD можуть призвести до втрати конфіденційності даних.

3.2 Вибір методів проектування

Документування архітектури ПЗ спрощує процес комунікації між заінтересованими особами (англ. stakeholders), дозволяє зафіксувати прийняті на ранніх етапах проектування рішення про високорівневий дизайн системи та дозволяє використовувати компоненти цього дизайну та шаблони повторно в інших проектах.

Основною ідеєю програмної архітектури є ідея зниження складності системи шляхом абстракції та розмежування повноважень.

Типове рішення модель-подання-контролер має на увазі виділення трьох окремих ролей. Модель – це об'єкт, який дає деяку інформацію про домен. У моделі немає візуального інтерфейсу, вона містить у собі всі дані та поведінку, не пов'язані з призначеним для користувача інтерфейсом. В об'єктно-орієнтованому контексті найчистішою формою моделі є об'єкт моделі предметної області. Як модель можна розглядати і сценарій транзакції, якщо він не містить ніякої логіки, пов'язаної з призначеним для користувача інтерфейсом. Подібне визначення не надто розширює поняття моделі, проте повністю відповідає розподілу ролей у розглянутому типовому рішенні.

Подання відображає вміст моделі засобами графічного інтерфейсу. Таким чином, якщо модель – це об'єкт користувача, відповідне уявлення може бути кадром з великою кількістю елементів керування або HTML-сторінкою, заповненою інформацією про користувача. Функції подання полягають лише у відображенні інформації на екрані. Усі зміни інформації обробляються третім «учасником» системи – контролером. Контролер отримує вхідні дані від користувача, виконує операції над моделлю і вказує на необхідність відповідного оновлення. У цьому плані графічний інтерфейс можна як сукупність уявлення і контролера (рисунок 3.2).

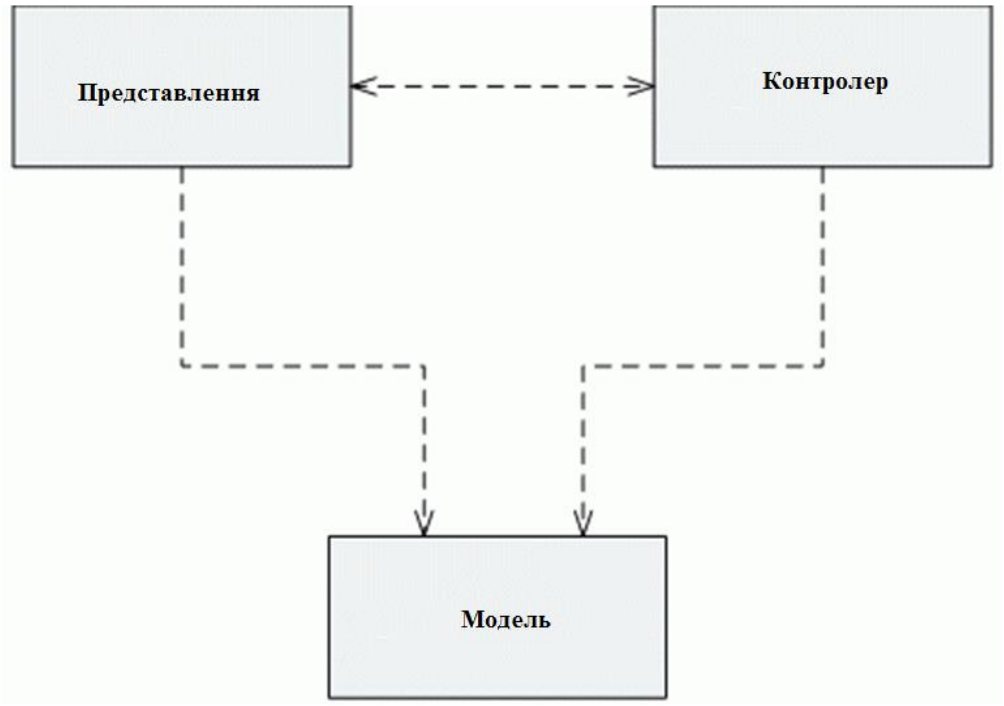

Рисунок 3.2 – Загальний вид рішення «Модель-Подання-Контролер»

Говорячи про типове рішення модель-представлення-контролер, не можна не підкреслити два основних типи поділу: відділення подання від моделі та відділення контролера від подання.

Відділення уявлення від моделі – це один із фундаментальних принципів проектування програмного забезпечення. Наявність такого поділу дуже важлива з низки причин:

– уявлення та модель відносяться до абсолютно різних сфер програмування;

– користувачі хочуть, щоб, залежно від ситуації, та сама інформація могла бути відображена різними способами;

– об'єкти, які не мають візуального інтерфейсу, набагато легше тестувати, ніж об'єкти з інтерфейсом [3].

Ключовим моментом у відділенні уявлення від моделі є усунення залежностей: уявлення залежить від моделі, але модель залежить від уявлення. Це означає, що зміна уявлення вимагає зміни моделі.

Стандартну схему архітектури «Модель-вид-контролер» зображено на рисунку 3.3.

Рисунок 3.3 – Схема архітектури MVC

3.3 Розробка алгоритму системи

Модель авторегресії є ефективним інструментом для розуміння та прогнозування майбутніх значень тимчасового ряду, яка включає регресування змінної за значеннями ряду в минулому. Важливість моделей ARMA полягає в їхній гнучкості, а також у їхній здатності описувати майже всі особливості стаціонарних часових рядів. Авторегресивні частини цих моделей описують, як послідовні спостереження в часі впливають один на одного, тоді як частини ковзних середніх захоплюють деякі можливі потрясіння, що не спостерігаються, дозволяє моделювати різні явища, які можна спостерігати в різних галузях від біології до фінансів.

Проста модель авторегресії (AR Autoregression).

Модель авторегресії показує, що значення вихідного змінна залежить лінійно лише від попередніх значень і від випадкового шуму.

Позначення AR(p) позначає авторегресійну модель порядку p. AR(p) модель математично визначається як:

$$y_n = \sum_{i=1}^{p} a_i y_{n-i} + e_n \,, \tag{3.1}$$

де ai – параметри моделі;

en – білий шум.

Найпростішим процесом AR є AR(0), який не має залежності між компонентами. На значення прогнозу впливає лише шум чи похибка. Для AR з коефіцієнтом a1 тільки попереднє значення часового ряду впливає на значення прогнозу.

Якщо a1 близький до 0, процес буде виглядати як білий шум, але якщо a1 по модулю буде наближатися до 1, то вихідне значення отримуватиме більший внесок від попередніх значень ряду в порівнянні з шумом.

Модель авторегресії - ковзного середнього (ARMA Autoregressive moving average).

Модель ARMA характеризує стохастичний процес за допомогою двох компонентів автомобільної регресії (AR) і ковзного середнього (MA).

Частина AR передбачає регресування змінної на власні попередні значення. Частина MA включає моделювання похибки як лінійної комбінації помилок, які у минулому.

Позначення ARMA (p, q) характеризує модель з p авто регресійними компонентами і q компонентами для ковзного середнього:

$$y_n = \sum_{i=1}^{p} a_i y_{n-i} + \sum_{i=1}^{q} b_i e_{n-i} + e_n \,, \tag{3.2}$$

де ai, bi – параметри моделі;

en – білий шум.

Модель ARMA доречна, коли система є функцією ряду потрясінь, що не спостерігаються, і власної поведінки. Наприклад, ціни на акції можуть бути суттєво змінитися через якусь фундаментальну інформацію, а також продемонструвати ефект повернення до середнього через певні дії учасників ринку.

Метод ковзного середнього було реалізовано для різних значень параметра N кількості попередніх моментів часу, було врахувати при побудові прогнозу (рисунок 3.4). Було перевірено, що при зменшенні довжини вікна N модель показує точніший результат на тестовій вибірці, вказує на властивість останніх даних впливати на прогнозну оцінку в майбутньому.

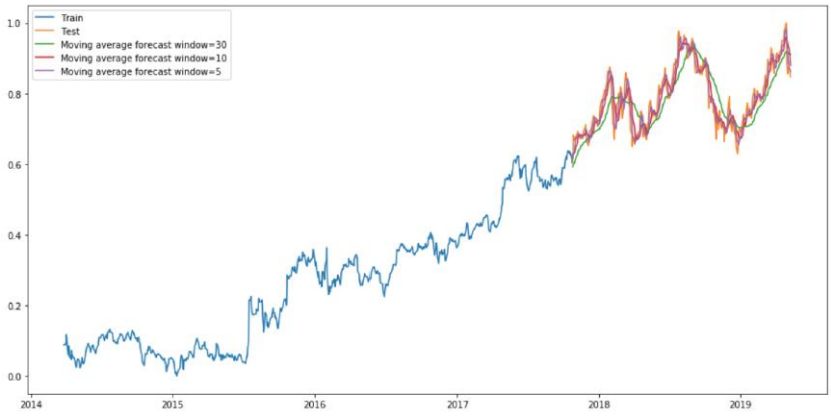

Рисунок 3.4 – Графік прогнозу простого ковзного середнього

Метод виваженого середнього було реалізовано за допомогою методу ewma бібліотеки Pandas (рисунок 3.5). За документацією відповідні вагові коефіцієнти обчислювалися за такою формулою:

$$\alpha = 1 - e^{\frac{log(0.5)}{halflife}}, \tag{3.3}$$

Тобто параметром налаштування моделі став halflife. Це дало наступні результати для різних значень halflife.

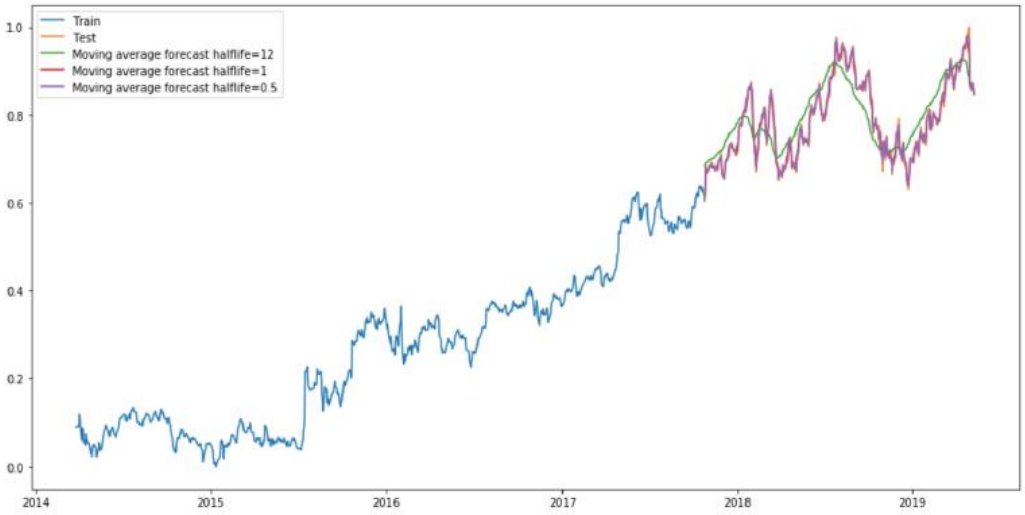

Рисунок 3.5 – Графік прогнозу методом виваженого ковзного середнього

Аналогічний підхід використовується для експоненційного ковзного середнього (рисунок 3.6), де як параметр задається рівень згладжування коефіцієнт α, який є ступенем зменшення зважування від 0 до 1.

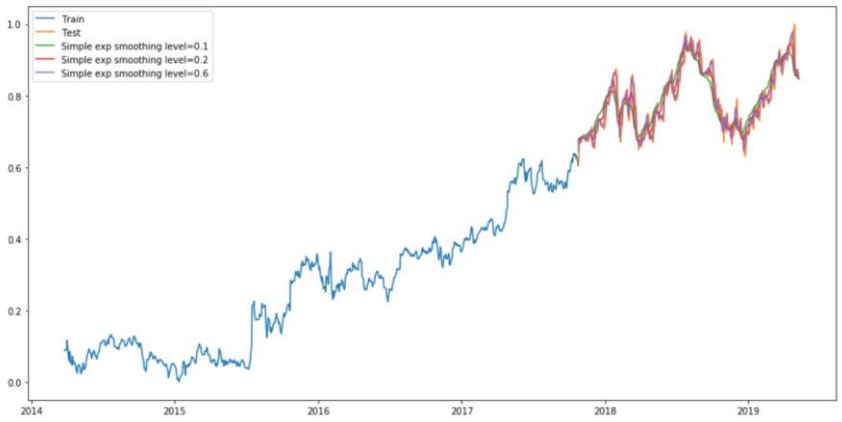

Рисунок 3.6 – Графік прогнозу методом експоненційного ковзного середнього

Чим менший рівень згладжування, тим точніше прогноз, оскільки кожне попереднє значення важить більше.

Метод подвійного експонентного згладжування передбачав завдання додаткового параметра β, який відповідає за згладжування тренду (рисунок 3.7).

Комбінація пари α і β коригували точність та якість прогнозу.

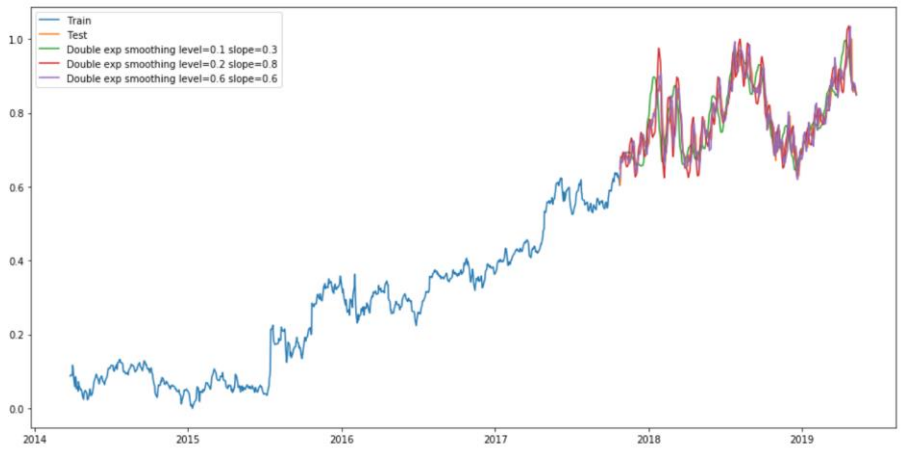

Рисунок 3.7 – Графік прогнозу методом подвійного експоненційного ковзного середнього

Потрійне експоненційне ковзне середнє або метод Голта-Вінтерса використовується для даних, в яких є як тренд, так і сезонність, і має додатковий коефіцієнт згладжування сезонної компоненти.

У роботі розглядається саме адитивна модель тренду та сезонної компоненти часового ряду. Функція ExponentialSmoothing у модулі tsa.holtwinters дозволяє задавати модель лише кількістю сезонних періодів, що є у даних.

Отриманий результат відображено у рисунку 3.8.

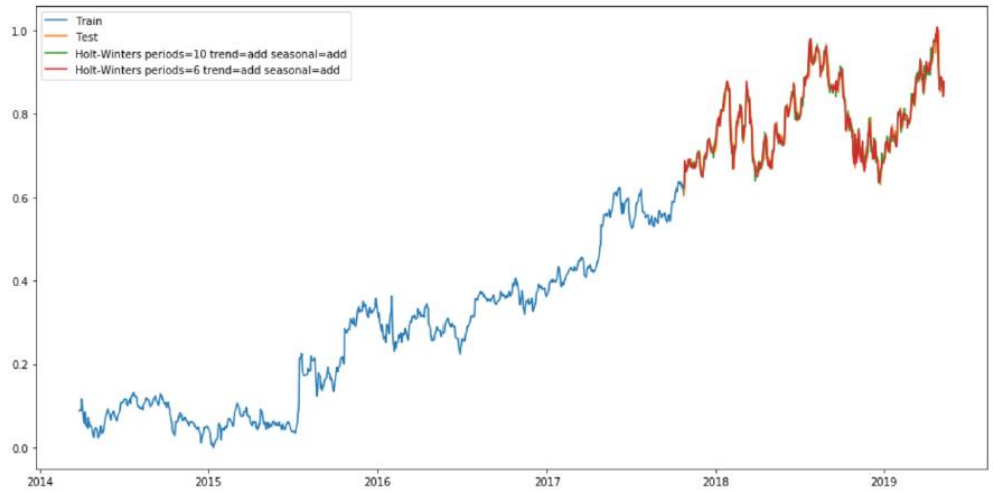

Рисунок 3.8 – Графік прогнозу методом Голта-Вінтерса

3.4 Схема логічної структури

Для експериментів було обрано рекурентну нейронну мережу з LSTMкомірками. Для прогнозування даних, які проводяться на якомусь часовому відрізку, видається важливим, щоб нейромережа з обраною архітектурою могла під час навчання використовувати дані про минулі параметри та показники. Цією особливістю володіють, як показано вище, рекурентні нейронні мережі.

Мінусом класичної рекурентної архітектури є те, що мережа не здатна «запам'ятати» дані про попередні стани надовго. Так обраний набір даних охоплює період у кілька років, за який було зіграно тисячі матчів, це може негативно впливати на якість прогнозування, тому в архітектуру було додано LSTM-осередки, які вирішують цю проблему.

На останньому шарі до виходів рекурентного шару застосовується softmax функція.

Як функцію втрат було обрано перехресну ентропію, метод оптимізації, що використовується – Adam.

Вхідними значеннями для нейронної мережі є вектори, з компонентами, що містять закодовані значення заздалегідь визначених ознак.

Гіпер-параметрами моделі є:

– розмір батча – кількість рядків даних, що подаються на вхід моделі за одну ітерацію. Цей параметр визначає кількість вхідних даних (підмножини навчальних даних), за якими буде розраховуватися градієнт, необхідний для зворотного поширення помилки;

– число епох. Одна епоха – це повне проходження всіх навчальних даних через нейронну мережу один раз. Нестача епох призводить до недонавчання (слабка узагальнювальна здатність моделі як на тренувальних, так і на тестових даних), надлишок – до перенавчання (модель занадто сильно «підлаштувалася» під навчальні дані, водночас демонструючи погану ефективність на тестових). На питання про єдино правильну кількість епох відповісти заздалегідь неможливо, цей параметр підбирається експериментально;

– довжина послідовності. Оскільки вхідні дані є вектором, у компонентах якого містяться ознаки, то довжина цього вектора також є параметром, який можна налаштувати;

– число LSTM-осередків. Кількість Long Short-Term Memory блоків, розташованих у прихованому шарі рекурентної мережі.

4 ТЕХНОЛОГІЯ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ РОЗРОБЛЕНИХ МОДЕЛЕЙ ТА АЛГОРИТМІВ

4.1 Вибір операційної системи

Як основне системне програмне забезпечення використовується операційна система Windows 10. Наведемо коротко огляд характеристик.

Корпорація Microsoft представляє три випуски операційної системи Windows 10, які відповідають практично будь-яким запитам користувачів персональних комп'ютерів, які застосовуються на роботі або в домашніх умовах. Операційна система Windows 10, призначена для корпоративних користувачів, забезпечує високий рівень масштабованості та надійності. Windows, будучи найкращою платформою для роботи з цифровими мультимедійними матеріалами, є найбільш вдалим вибором для користувачів домашніх комп'ютерів та любителів комп'ютерних ігор. 64-розрядна операційна система Windows здатна задовольнити найвибагливіших користувачів, які мають спеціальну технічну підготовку.

Опишемо 5 аргументів на користь використання 64-розрядної версії Windows 10 як керуюча операційна система для впровадження рішень:

– 64-розрядна версія Windows fessional – це потужна платформа, що забезпечує можливість високопродуктивних обчислень. Вбудовані 64-розрядні програми дозволяють обробляти більше даних за один такт, працюючи набагато швидше та ефективніше;

– підтримка великого обсягу пам'яті 64-розрядна версія Windows підтримує до 128 ГБ оперативної пам'яті та 16 ТБ віртуальної пам'яті, що дозволяє програмам швидше працювати з більшими обсягами даних. У віртуальну пам'ять можуть бути завантажені значно більші обсяги даних додатків, що забезпечує 64-розрядному процесору можливість швидкого доступу до цих додатків;

– гнучкість застосування Завдяки середовищу емуляції Windows on Windows 64 (WOW64) x86 64-розрядна версія Windows забезпечує потужну платформу для інтеграції 64-розрядних програм з наявними 32-розрядними, що дозволяє користувачам перейти на 64-розрядні обчислювальні середовища без втрати інвестицій -розрядне програмне забезпечення та з можливістю використовувати накопичений досвід роботи з Windows;

– багатопроцесорна обробка та багатоядерні процесори 64-розрядна версія Windows призначена для підтримки до двох одноядерних або багатоядерних 64-розрядних процесорів, що забезпечують максимальну продуктивність та масштабованість;

– колишня модель програмування Розробники з навичками роботи в 32-розрядному середовищі зможуть швидко і легко освоїтися в 64-розрядному середовищі розробки Windows, яке практично нічим не відрізняється від 32-розрядного.

4.2 Вибір середовища розробки

Єдиною істотною перевагою мов програмування, що компілюються, є створюваний ними високошвидкісний код. Коли швидкість виконання програми не є критичною величиною, найбільш правильним вибором буде інтерпретована мова, як більш простий і гнучкий інструмент програмування.

У зв'язку з цим, певний інтерес представляє розгляд порівняно нової мови програмування Python, яка була створена її автором Гвідо ван Россумом на початку 90-х років.

Відмінні характеристики мови:

– дуже низький поріг входження, вже після одного дня вивчення можна почати писати прості програми;

– мінімалістична мова, з невеликою кількістю конструкцій;

– короткий код;

– чудово підходить для створення програм-обгорток, підтримується імпорт Сі-бібліотек;

– існує велика кількість реалізацій: CPython (основна реалізація); Jython (реалізація для JVM); IronPython (CLR); PyPy;

– дуже хороша підтримка математичних обчислень (бібліотеки NumPy, SciPy);

– використовується для обробки природних мов (NLTK);

– велика кількість розвинених web-фреймворків (Django, TurboGear, CherryPy, Flask).

Причини використання Python.

Якість програмного забезпечення. Для багатьох основна перевага мови Python полягає в читабельності, ясності та вищій якості, що відрізняють її від інших інструментів у світі мов програмування. Програмний код мовою Python читається легше, а отже, багаторазове його використання та обслуговування виконується набагато простіше, ніж використання програмного коду на інших мовах сценаріїв. Однаковість оформлення програмного коду мовою Python полегшує його розуміння навіть для тих, хто не брав участі в його створенні. Крім того, Python підтримує найсучасніші механізми багаторазового використання програмного коду, яким є об'єктно-орієнтоване програмування (ООП).

Висока швидкість розробки. Порівняно з компілювальними або строго типізованими мовами, такими як C, C++ і Java, Python у багато разів підвищує продуктивність праці розробника. Обсяг програмного коду мовою Python зазвичай становить третину або навіть п'яту частину еквівалентного програмного коду мовою C++ або Java. Це означає менший

обсяг введення з клавіатури, меншу кількість часу на налагодження і менший обсяг трудовитрат на супровід. Крім того, програми мовою Python запускаються відразу ж, минаючи тривалі етапи компіляції та зв'язування, необхідні в деяких інших мовах програмування, що ще більше збільшує продуктивність праці програміста.

Переносимість програм. Велика частина програм мовою Python виконується без змін на всіх основних платформах. Перенесення програмного коду з операційної системи Linux у Windows зазвичай полягає в простому копіюванні файлів програм з однієї машини на іншу. Більше того, Python надає масу можливостей зі створення переносних графічних інтерфейсів, програм доступу до баз даних, веб-додатків і багатьох інших типів програм. Навіть інтерфейси операційних систем, включно зі способом запуску програм і обробкою каталогів, у мові Python реалізовано переносним способом.

4.3 Вибір технології для розробки

Оперативна аналітична обробка та інтелектуальний аналіз даних – дві складові процесу підтримки прийняття рішень. Але сьогодні більшість систем OLAP загострює увагу лише на забезпеченні доступу до багатовимірних даних, а більшість засобів ІАД, які працюють у сфері закономірностей, мають справу з одновимірними перспективами даних. Ці два види аналізу мають бути тісно об'єднані, тобто системи OLAP повинні фокусуватися не лише на доступі, а й пошуку закономірностей. Вчений K. Parsaye вводить складений термін «OLAP Data Mining» (багатомірний інтелектуальний аналіз) і пропонує кілька варіантів інтеграції двох технологій (рисунок 4.1).

«Cubing then mining». Можливість виконання інтелектуального аналізу має забезпечуватися над будь-яким результатом запиту до

багатовимірного концептуального подання, тобто над будь-яким фрагментом будь-якої проекції гіперкубу показників.

«Mining then cubing». Подібно до даних, вилучених зі сховища, результати інтелектуального аналізу повинні подаватися в гіперкубічній формі для подальшого багатовимірного аналізу.

«Cubing while mining». Цей гнучкий спосіб інтеграції дозволяє автоматично активізувати однотипні механізми інтелектуальної обробки над результатом кожного кроку багатовимірного аналізу (переходу між рівнями узагальнення, отримання нового фрагмента гіперкуба і т. д.).

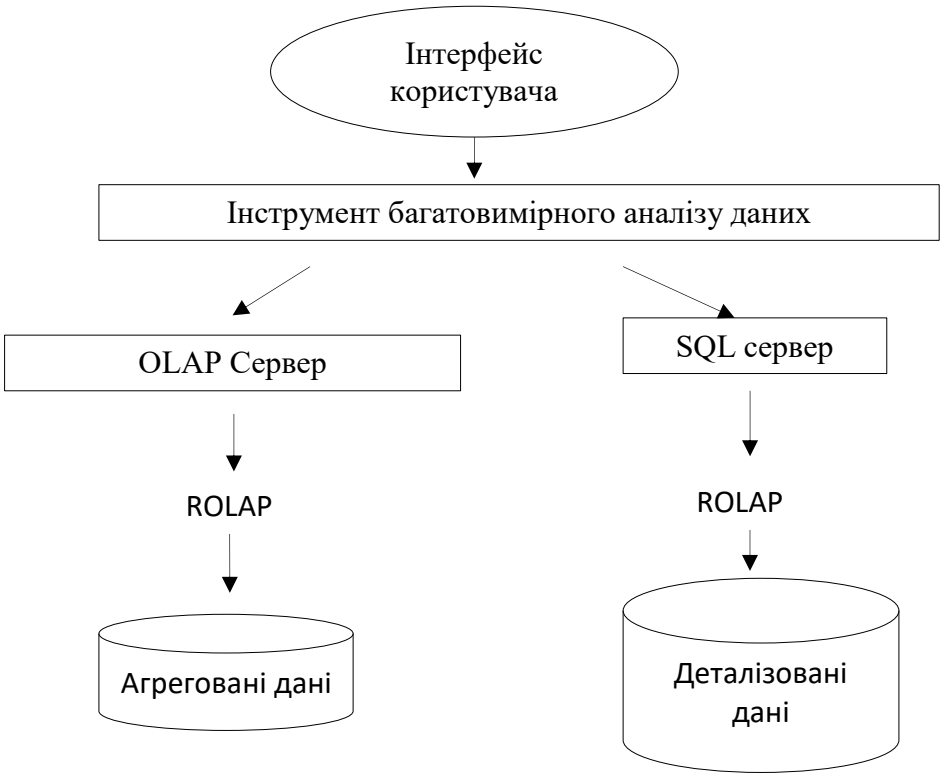

Рисунок 4.1 – Архітектура системи багатовимірного інтелектуального аналізу даних

Розглянемо основні функціональні можливості засобів оперативної аналітичної обробки даних:

– F1. Багатовимірне концептуальне подання даних;

– F2. Інтуїтивне маніпулювання даними. Інструмент повинен пропонувати вибір найзручнішого режиму роботи з даними;

– F3. Доступність (OLAP виступає посередником). У цьому правилі особливо підкреслюється роль OLAP як прошарку між гетерогенними джерелами даних та їх поданням для кінцевого користувача;

– F4. Пакетний доступ проти інтерпретації. Це правило вимагає, щоб продукт однаково ефективно забезпечував доступ як до власного сховища даних, так і зовнішніх даних;

– F5. Моделі аналізу OLAP. OLAP продукти повинні підтримували чотири моделі аналізу: категоріальний, тлумачний, візуальний та стереотипний;

– F6. Архітектура «клієнт-сервер». Продукт має бути не тільки клієнт-серверним, але й щоб серверний компонент був би досить інтелектуальним для того, щоб різні клієнти могли підключатися до мінімуму зусиль. Це означає, що OLAP має бути доступним з робочого столу користувача;

– F7. Прозорість. Повна відповідність цій вимогі означає, що користувач електронної таблиці здатний отримати всі необхідні дані з сервера OLAP, навіть якщо невідомо звідки вони виходять;

– F8. Забезпечення клієнтської підтримки. Інструменти OLAP повинні забезпечувати одночасний доступ (читання та запис), інтеграцію та конфіденційність. Спеціалізовані особливості;

– F9. Обробка ненормалізованих даних. Необхідність інтеграції між OLAP-сервером та ненормалізованими джерелами даних;

– F10. Збереження результатів OLAP за принципом розміщення в пам'яті окремо від вихідних даних. OLAP-додатки, що працюють у режимі

читання-запису, повинні не впливати безпосередньо на оброблювані дані та дані, модифіковані в OLAP;

– F11. Виняток відсутніх значень. Відсутні значення повинні відрізнятися від нульових значень;

– F12. Обробка відсутніх значень. Усі відсутні значення повинні ігноруватися OLAP-аналізатором без урахування їхнього джерела. Особливості подання звітності;

– F13. Гнучкість формування звітів, тобто виміри мають бути розміщені у звіті так, як це потрібно користувачеві;

– F14: Стандартна продуктивність звітів. Продуктивність формування звітів має істотно падати зі зростанням кількості вимірів і розмірів бази даних;

– F15. Автоматичне налаштування фізичного рівня. OLAP-додатки повинні автоматично налаштовувати свою фізичну схему залежно від типу моделі, обсягів даних та розрідженості бази даних.;

– F16. Універсальність вимірів. Усі виміри мають бути рівноправними, кожен вимір - еквівалентним і в структурі, і в операційних можливостях;

– F17. Необмежену кількість вимірів та рівнів агрегації. Технічно немає продукту, який міг би відповідати цій вимозі, тому що немає такого режиму, як необмежений об'єкт на обмеженому комп'ютері, а у разі прийняття деякого максимуму він повинен забезпечувати принаймні два десятки вимірів;

– F18. Необмежені операції між просторами.

Усі види операцій повинні бути дозволені для будь-яких вимірювань, а не лише для вимірювань типу «показник» – міра.

4.4 Програмна реалізація

Концептуальна схема являє собою узагальнені функціональні та інформаційні компоненти проєктованого програмного продукту, принципи їхньої взаємодії між собою, з користувачем і зовнішнім середовищем. За результатами вищеописаної роботи було спроектовано концептуальну схему системи постановки процесів на виконання та структуризації оперативної пам'яті, яку показано на рисунку 4.2, де показано дві концептуальні частини - компонент оброблення даних, що надійшли в систему, і компонент аналізу ідентифікованих характеристик.

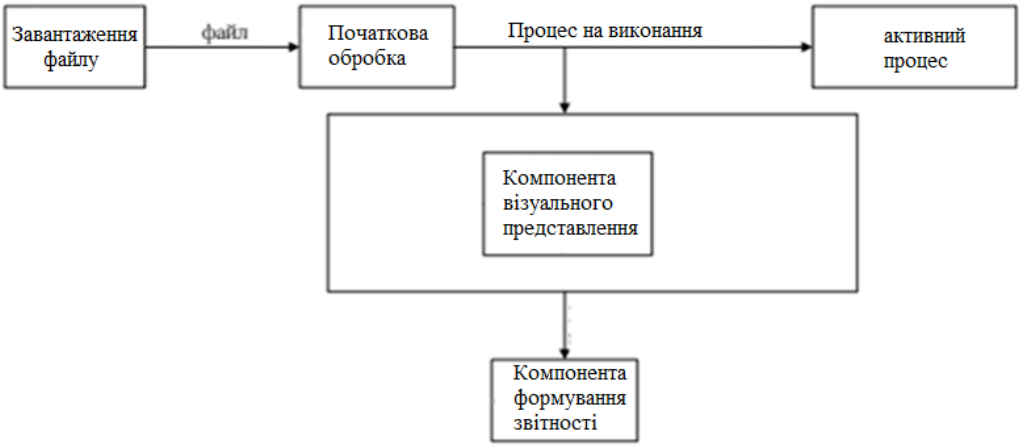

Рисунок 4.2 – Концептуальна схема розроблюваної програмної системи

4.5 Реалізація алгоритму автоматизації підготовки великих даних до аналізу та створення моделей інтелектуального аналізу даних

Виконаємо алгоритмізацію аналізу даних за допомогою технології Tensor Flow в середовищі Google Collab (рисунки 4.3 – 4.4).

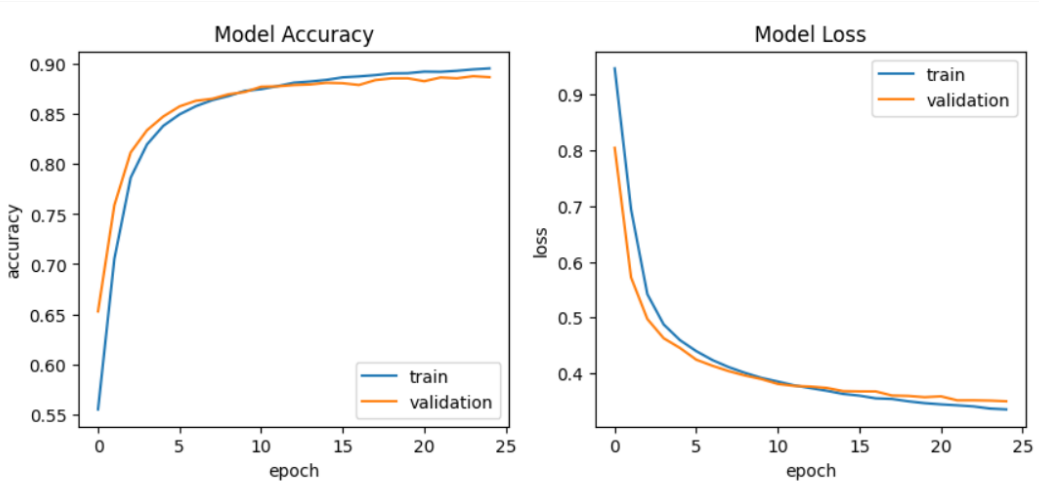

Рисунок 4.3 – Результат навчання мережі та співставлення даних

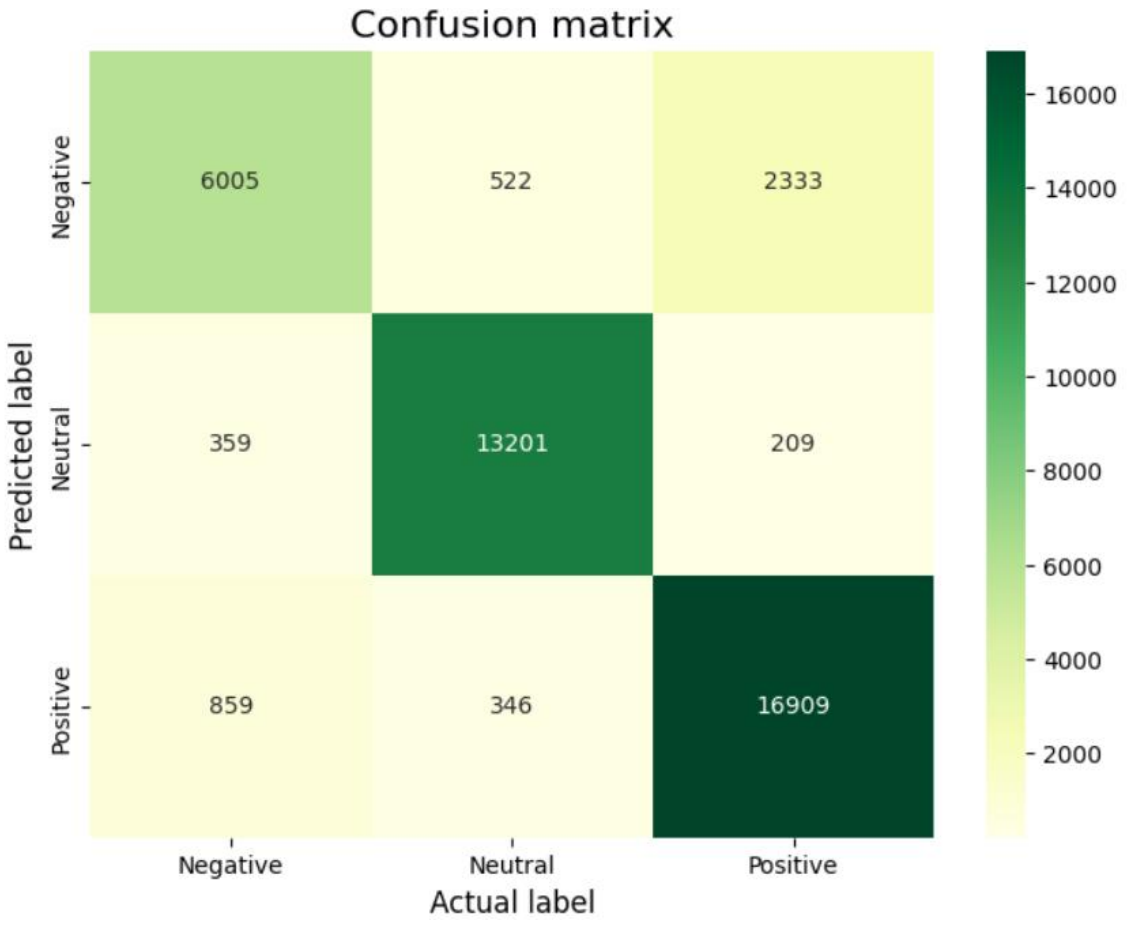

Рисунок 4.4 – Побудова матриці неточностей навчання мережі

4.6 Посібник користувача

Для запуску програми на виконання відкриваємо файл програмного коду в середовищі Python, запуск відбувається при натисканні кнопки F5 (рисунки 4.5 – 4.10). Вхідні тестові дані зібрані в окремому файлі з розширенням CSV.

```
train = pd.read_csv("data/train.csv")
test = pd.read_csv("data/test.csv")
col_for_x = ['mix_mv_avg','5_price_diff','mv_avg_diff', 'avg_quantity','quantity_price','ct_ris
x_train = np.array(train[col_for_x])
x_test = np.array(test[col_for_x])
y_train_encoded = np.array(pd.get_dummies(train.label))
y_test_encoded = np.array(pd.get_dummies(test.label))
y_test = np.array(test.label)

# формування моделі навчання
cnn_input = []
for i in range(x_train.shape[0]-50):
    row = []
    for k in range(50):
        for t in range(8):
            row.append(x_train[i+k,:][t])
    cnn_input.append(row)
x_train_grid = pd.DataFrame(cnn_input)    # df
check = y_train_encoded[0:-50,:]

# тест
cnn_input_test = []
for i in range(x_test.shape[0]-50):
    row = []
    for k in range(50):
        for t in range(8):
            row.append(x_test[i+k,:][t])
    cnn input test.append(row)
```

Рисунок 4.5 – Програма в середовищі Python

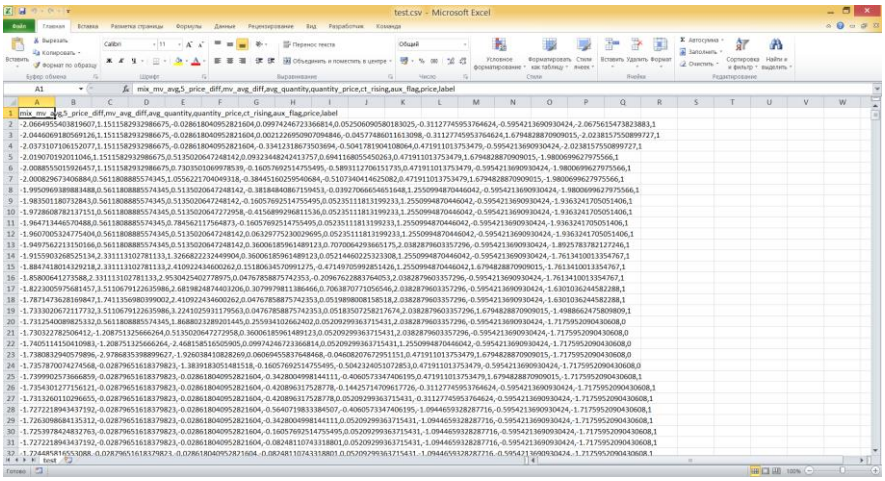

Рисунок 4.6 – Файл тестових даних

Результат навчання моделі зберігається у файлі test.csv.

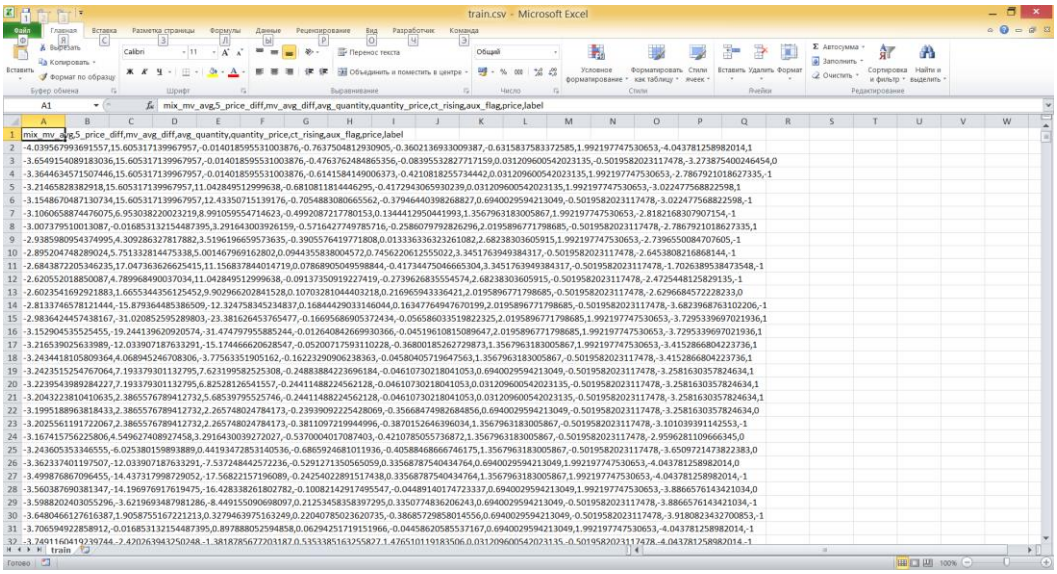

Рисунок 4.7 – Файл з результатами навчання моделі

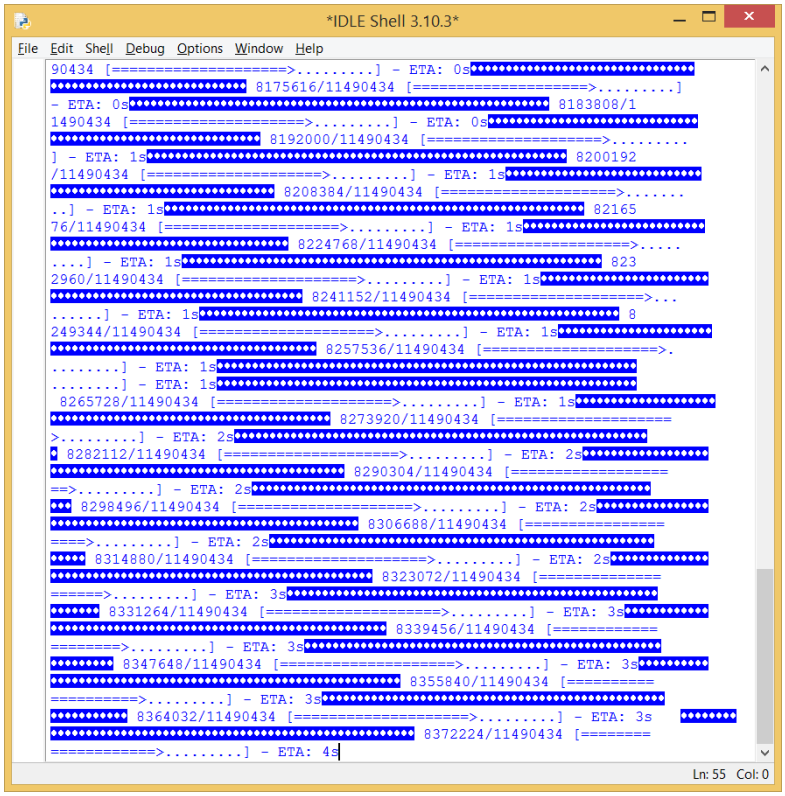

Рисунок 4.8 – Процес навчання

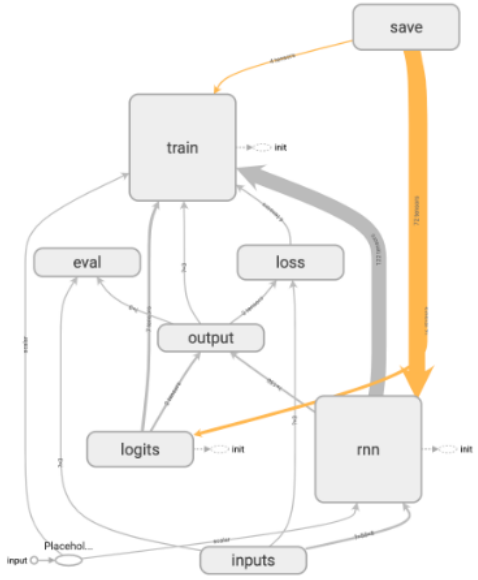

Рисунок 4.9 – Структурна схема моделі

```
Best score: 0.673
Best parameters set:
        criterion: 'entropy'
        min_samples_leaf: 30
        n_estimators: 300
        random_state: 123
Predicted   -1     0    1
Actual
-1         461    96   11
 0         377  3365  777
 1           2   109  632
             precision    recall  f1-score   support

         -1       0.55      0.81      0.65       568
          0       0.94      0.74      0.83      4519
          1       0.45      0.85      0.58       743

avg / total       0.84      0.76      0.78      5830

Accuracy: 0.7646655231560892
Cohen's Kappa: 0.5099966980909575
```

```
Predicted   -1     0    1
Actual
-1         556    11    1
 0         814  3045  660
 1           1    12  730
             precision    recall  f1-score   support

         -1       0.41      0.98      0.57       568
          0       0.99      0.67      0.80      4519
          1       0.52      0.98      0.68       743

avg / total       0.88      0.74      0.77      5830

Accuracy: 0.7428816466552316
Cohen's Kappa: 0.5227722336495331
Took: 8.974967 seconds
```

```
Predicted   -1     0    1
Actual
-1         551    16    1
 0         754  3135  630
 1           2    41  700
             precision    recall  f1-score   support

         -1       0.42      0.97      0.59       568
          0       0.98      0.69      0.81      4519
          1       0.53      0.94      0.68       743

avg / total       0.87      0.75      0.77      5830

Accuracy: 0.7523156089193825
Cohen's Kappa: 0.5279231251958805
```

Рисунок 4.10 – Результат виконання прогнозування за допомогою бібліотеки TensorFlow

Виконуємо завантаження даних програмним методом в середовищі Google Collab (рисунки 4.11 – 4.15).

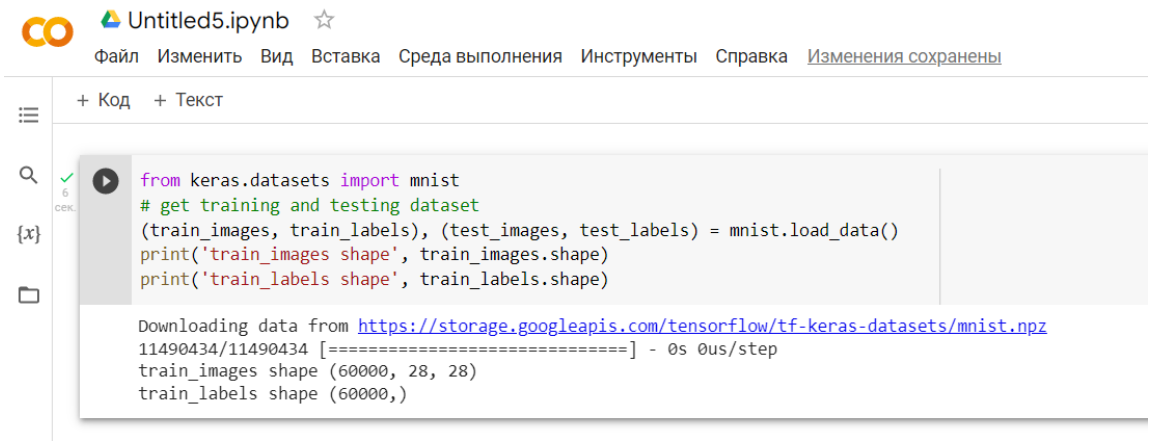

Рисунок 4.11 – Завантаження тестових даних

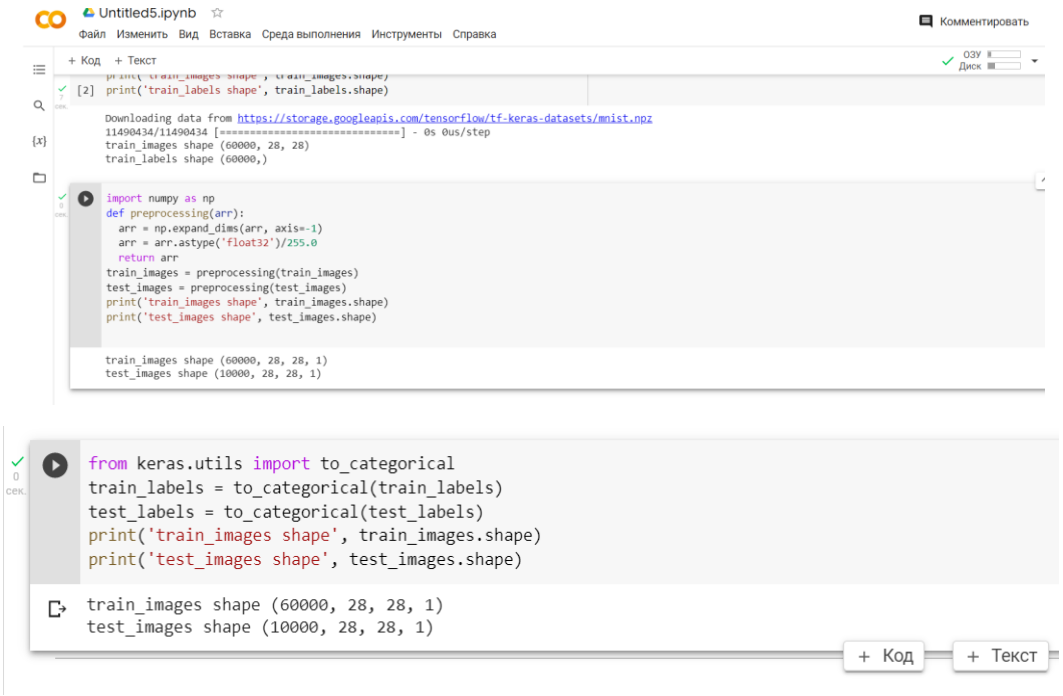

Рисунок 4.12 – Підготовка даних

Побудова моделі.

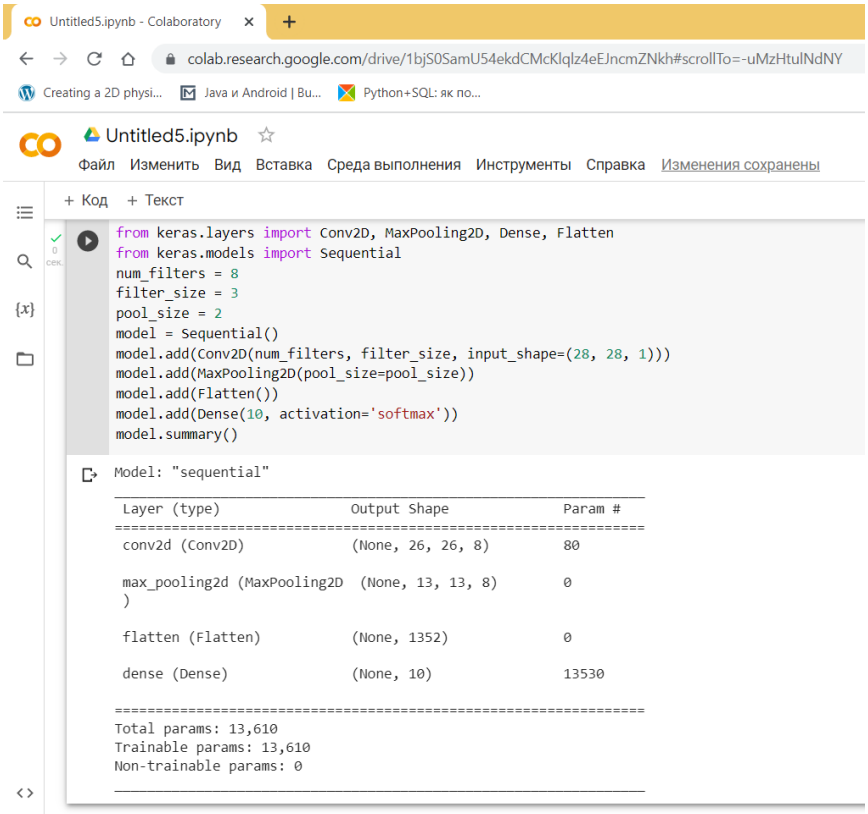

Рисунок 4.13 – Побудова моделі

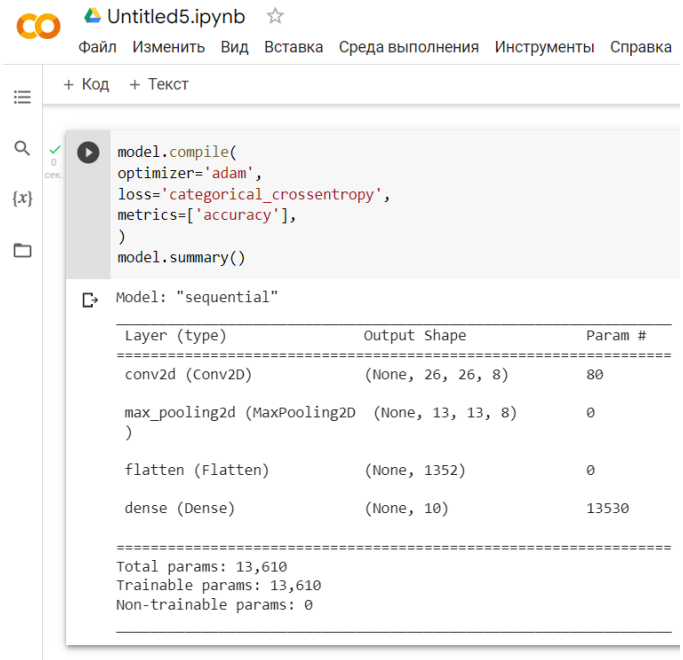

Рисунок 4.14 – Компіляція моделі

```
model.fit(
train_images, #training data
train_labels, #training targets
epochs=5,
batch_size=32,
)
```

```
Epoch 1/5
1875/1875 [==============================] - 20s 10ms/step - loss: 0.3387 - accuracy: 0.9040
Epoch 2/5
1875/1875 [==============================] - 20s 11ms/step - loss: 0.1782 - accuracy: 0.9488
Epoch 3/5
1875/1875 [==============================] - 19s 10ms/step - loss: 0.1306 - accuracy: 0.9624
Epoch 4/5
 708/1875 [==========>...................] - ETA: 11s - loss: 0.1106 - accuracy: 0.9688
```

Рисунок 4.15 – Навчання моделі

Запускаємо на виконання, результати наведено на рисунках 4.16–4.19.

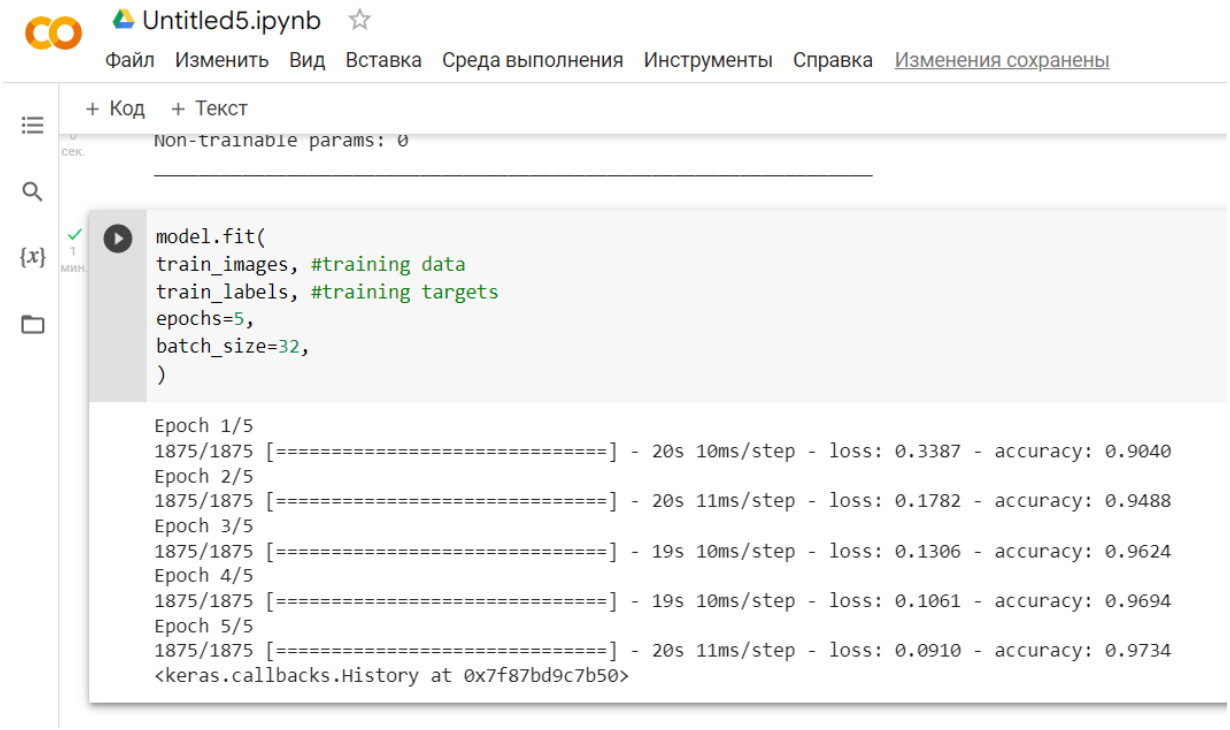

Рисунок 4.16 – Запуск на виконання

Тестування моделі.

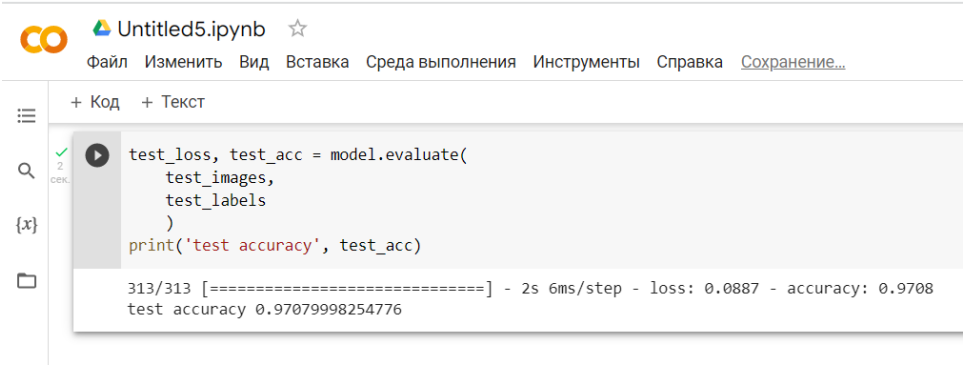

Рисунок 4.17 – Тестування моделі

Рисунок 4.18 – Запуск на виконання

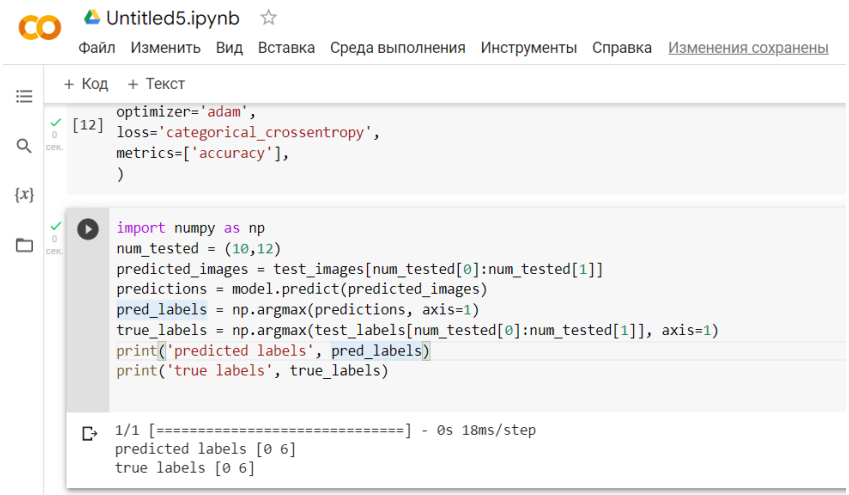

Рисунок 4.19 – Тестування моделі

Виконання зі зміною параметрів CNN (рисунок 4.20).

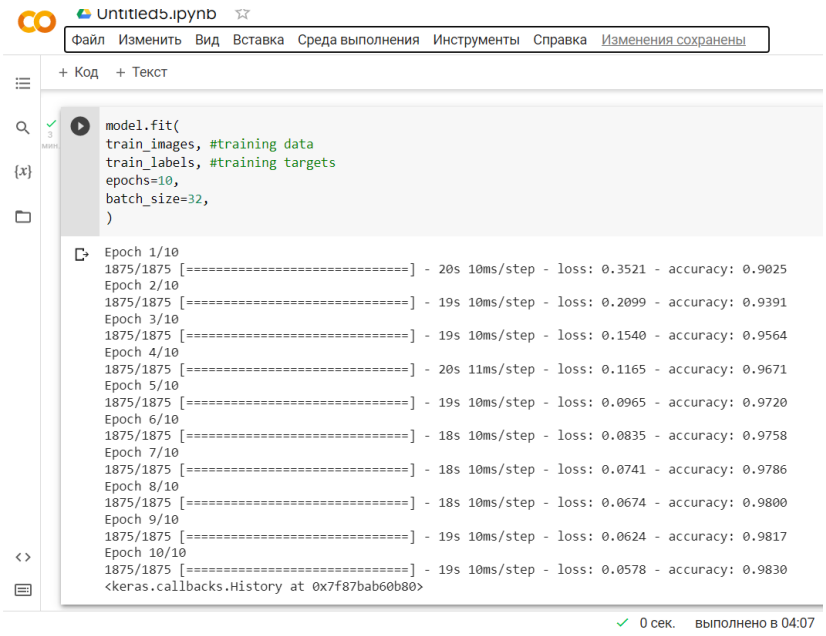

Рисунок 4.20 – Результат виконання моделі

Будуємо графік (рисунки 4.21–4.22).

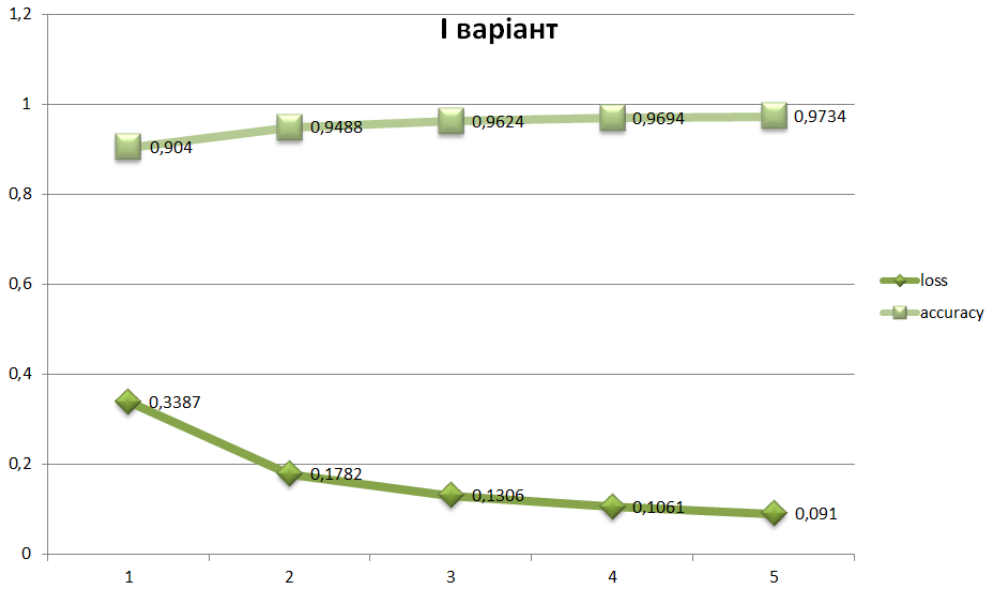

Рисунок 4.21 – Графік параметрів моделі

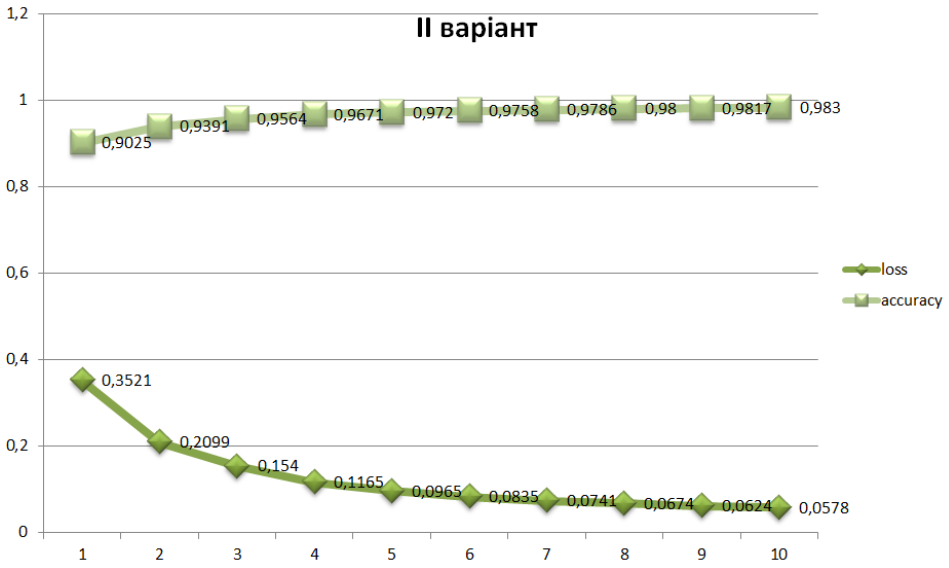

Рисунок 4.22 – Графік параметрів моделі

Змінюємо параметри моделі, виконуючи збільшення кількості ітерацій навчання та розмірності партій даних (рисунок 4.23).

```
optimizer='adam',
loss='categorical_crossentropy',
metrics=['accuracy'],
)
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 23, 23, 16)        592

 max_pooling2d_2 (MaxPooling  (None, 5, 5, 16)         0
 2D)

 flatten_2 (Flatten)         (None, 400)               0

 dense_2 (Dense)             (None, 10)                4010

=================================================================
Total params: 4,602
Trainable params: 4,602
Non-trainable params: 0
_________________________________________________________________
```

Рисунок 4.23 – Зміна параметрів моделі

Результати обчислення свідчать про більш точну апроксимацію даних завдяки кількості ітерацій (рисунок 4.24).

```
[ ] train_images, #training data
    train_labels, #training targets
    epochs=15,
    batch_size=64,
    )
```

```
Epoch 1/15
938/938 [==============================] - 23s 24ms/step - loss: 0.3754 - accuracy: 0.8985
Epoch 2/15
938/938 [==============================] - 22s 24ms/step - loss: 0.1286 - accuracy: 0.9623
Epoch 3/15
938/938 [==============================] - 22s 24ms/step - loss: 0.0986 - accuracy: 0.9706
Epoch 4/15
938/938 [==============================] - 23s 25ms/step - loss: 0.0807 - accuracy: 0.9758
Epoch 5/15
938/938 [==============================] - 22s 24ms/step - loss: 0.0712 - accuracy: 0.9789
Epoch 6/15
938/938 [==============================] - 23s 25ms/step - loss: 0.0633 - accuracy: 0.9811
Epoch 7/15
938/938 [==============================] - 22s 24ms/step - loss: 0.0590 - accuracy: 0.9823
Epoch 8/15
938/938 [==============================] - 22s 24ms/step - loss: 0.0530 - accuracy: 0.9840
Epoch 9/15
938/938 [==============================] - 22s 24ms/step - loss: 0.0500 - accuracy: 0.9845
Epoch 10/15
938/938 [==============================] - 22s 24ms/step - loss: 0.0469 - accuracy: 0.9857
Epoch 11/15
938/938 [==============================] - 22s 24ms/step - loss: 0.0449 - accuracy: 0.9861
Epoch 12/15
938/938 [==============================] - 22s 24ms/step - loss: 0.0429 - accuracy: 0.9866
Epoch 13/15
938/938 [==============================] - 22s 23ms/step - loss: 0.0402 - accuracy: 0.9871
Epoch 14/15
938/938 [==============================] - 22s 23ms/step - loss: 0.0388 - accuracy: 0.9878
Epoch 15/15
938/938 [==============================] - 23s 24ms/step - loss: 0.0373 - accuracy: 0.9880
<keras.callbacks.History at 0x7f87ba2181c0>
```

```
[ ] test_loss, test_acc = model.evaluate(
        test_images,
        test_labels
        )
    print('test accuracy', test_acc)
```

```
313/313 [==============================] - 2s 6ms/step - loss: 0.0528 - accuracy: 0.9830
test accuracy 0.9829999804496765
```

```
[ ] model.save('model.h5')
```

```
test_loss, test_acc = model.evaluate(
    test_images,
    test_labels
    )
print('test accuracy', test_acc)
```

```
313/313 [==============================] - 2s 7ms/step - loss: 0.0528 - accuracy: 0.9830
test accuracy 0.9829999804496765
```

Рисунок 4.24 – Результат моделювання

5 ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ОТРИМАНИХ РІШЕНЬ ПІСЛЯ РОЗРОБКИ ПРОГРАМНОГО ЗАСОБУ ОПРАЦЮВАННЯ ПОТОКІВ ДАНИХ

Періодична та сезонна залежність (сезонність) є іншим загальним типом компонент тимчасового ряду. Періодична залежність може бути формально визначена як кореляційна залежність порядку k між кожним і елементом ряду і (ik) елементом. Її можна виміряти за допомогою автокореляції (тобто кореляції між самими членами ряду); k зазвичай називають лагом (іноді використовують еквівалентні терміни: зсув, запізнення). Якщо помилка виміру не надто велика, то сезонність можна визначити візуально, розглядаючи поведінку членів низки через кожні k часових одиниць.

Автокореляційна корелограма. Сезонні складові часового ряду можуть бути знайдені за допомогою корелограми. Корелограма (автокорелограма) показує чисельно і графічно автокореляційну функцію (АКФ), тобто коефіцієнти автокореляції (та його стандартні помилки) для послідовності лагів з певного діапазону.

Коефіцієнти приватної кореляції більш високих порядків можна визначити через коефіцієнти приватної кореляції нижчих порядків за наступною формулою рекурентної:

$$r_{yx_i(x_1x_2\ldots x_k)} = \frac{r_{yx_i(x_1x_2\ldots x_{k-1})} - r_{yx_k(x_1x_2\ldots x_{k-1})} r_{x_ix_k(x_1x_2\ldots x_{k-1})}}{\sqrt{\left(1-r^2_{yx_k(x_1x_2\ldots x_{k-1})}\right)\cdot\left(1-r^2_{x_ix_k(x_1x_2\ldots x_{k-1})}\right)}} \quad (5.1)$$

Коефіцієнти приватної кореляції широко використовуються на стадії формування моделі при відборі факторів.

При побудові багатофакторної моделі застосовується метод виключення змінних, під час якого будується рівняння регресії з повним

набором змінних, потім розраховується матриця окремих коефіцієнтів кореляції. Далі перевіряється статистична значущість кожного з коефіцієнтів згідно з t-критерієм Ст'юдента. Незалежна змінна, що має найменшу та несуттєву кореляцію із залежною змінною, виключається. Потім будується нове рівняння регресії, і процедура триває до того часу, доки виявиться, що це приватні коефіцієнти кореляції статистично значимі, тобто істотно від нуля.

Перевірка статистичної значущості приватного коефіцієнта кореляції перевірка гіпотези про те, що він дорівнює нулю H0: $r_{yx_i(x_1x_2\ldots x_k)}=0$.

Розраховується статистика:

$$t=\frac{r_{yx_i(x_1x_2\ldots x_k)}}{\sqrt{1-(r_{yx_i(x_1x_2\ldots x_k)})^2}}\cdot\sqrt{n-(k+1)} \qquad (5.2)$$

Висновок про значимість приватного коефіцієнта кореляції робиться при$|t|>t_\varepsilon$де t_εвідповідне табличне значення t-розподілу з (n-(k+1)) ступенями свободи.

Розрахуємо парні лінійні коефіцієнти кореляції, застосовуючи формулу і водночас перевіряючи їхню статистичну значимість.

$$r_{YX_1}=\frac{n\sum_{i=1}^{n}X_1Y-\sum_{i=1}^{n}X_1\sum_{i=1}^{n}Y}{\sqrt{\left[n\sum_{i=1}^{n}X_1^2-\left(\sum_{i=1}^{n}X_1\right)^2\right]\left[n\sum_{i=1}^{n}Y^2-\left(\sum_{i=1}^{n}Y\right)^2\right]}}=$$

$$=\frac{20\cdot 8912{,}57-277{,}2\cdot 454{,}5}{\sqrt{(20\cdot 5860{,}9-76839{,}84)\cdot(20\cdot 18206{,}89-206570{,}3)}}=0{,}6553,$$

$$t=0{,}6553\cdot\sqrt{20-2}\,/\,\sqrt{1-(0{,}6553)^2}=3{,}68,$$

$$r_{YX_2} = \frac{n\sum_{i=1}^{n} X_2 Y - \sum_{i=1}^{n} X_2 \sum_{i=1}^{n} Y}{\sqrt{\left[n\sum_{i=1}^{n} X_2^2 - \left(\sum_{i=1}^{n} X_2\right)^2\right]\left[n\sum_{i=1}^{n} Y^2 - \left(\sum_{i=1}^{n} Y\right)^2\right]}} =$$

$$= \frac{20 \cdot 908{,}56 - 31{,}8 \cdot 454{,}5}{\sqrt{(20 \cdot 61{,}5 - 1011{,}24) \cdot (20 \cdot 18206{,}89 - 206570{,}3)}} = 0{,}6346,$$

$$t = 0{,}6346 \cdot \sqrt{20-2} / \sqrt{1-(0{,}6346)^2} = 3{,}60,$$

$$r_{X_1 X_2} = \frac{n\sum_{i=1}^{n} X_1 X_2 - \sum_{i=1}^{n} X_1 \sum_{i=1}^{n} X_2}{\sqrt{\left[n\sum_{i=1}^{n} X_1^2 - \left(\sum_{i=1}^{n} X_1\right)^2\right]\left[n\sum_{i=1}^{n} X_2^2 - \left(\sum_{i=1}^{n} X_2\right)^2\right]}} =$$

$$= \frac{20 \cdot 8912{,}57 - 277{,}2 \cdot 31{,}8}{\sqrt{(20 \cdot 5860{,}9 - 76839{,}84) \cdot (20 \cdot 61{,}5 - 1011{,}24)}} = 0{,}1247,$$

$$t = 0{,}1247 \cdot \sqrt{20-2} / \sqrt{1-(0{,}1247)^2} = 2{,}80.$$

	y	x_1	x_2
y	1,0	0,6553 (3,68)	0,6346 (3,60)
x_1	0,6553 (3,68)	1,0	0,1247(2,80)
x_2	0,6346(3,60)	0,1247(2,80)	1,0

Складемо матрицю парних лінійних коефіцієнтів кореляції (у дужках значення t-статистик):

Коефіцієнт кореляції між у і х1, свідчить про прямий статистично значимий зв'язок між вартістю перевезення та вагою вантажу, що перевозиться. Коефіцієнт кореляції між у та х2 також свідчить про прямий та статистично значущий зв'язок між вартістю перевезення та відстанню перевезення. Величина статистично значущого коефіцієнта кореляції між

x1 і x2 означає практичну відсутність взаємозв'язку між відстанню перевезення та вагою вантажу, що суперечить початковим припущенням у тому, що відстань перевезення може бути обумовлено вагою вантажу і навпаки. Розрахуємо коефіцієнти приватної кореляції та перевіримо їх значущість:

$$r_{yx_1(x_2)} = \frac{0{,}6553 - 0{,}6346 \cdot 0{,}1247}{\sqrt{(1-(0{,}6346)^2)\cdot(1-(0{,}1247)^2)}} = 0{,}7513;$$

$$t = \frac{0{,}7513}{\sqrt{1-(0{,}7513)^2}} \cdot \sqrt{20-(2+1)} = 4{,}69,$$

$$r_{yx_2(x_1)} = \frac{0{,}6346 - 0{,}6553 \cdot 0{,}1247}{\sqrt{(1-(0{,}6553)^2)\cdot(1-(0{,}1247)^2)}} = 0{,}7377;$$

$$t = \frac{0{,}7377}{\sqrt{1-(0{,}7377)^2}} \cdot \sqrt{20-(2+1)} = 4{,}51,$$

$$r_{x_1x_2(y)} = \frac{0{,}1247 - 0{,}6553 \cdot 0{,}6346}{\sqrt{(1-(0{,}6553)^2)\cdot(1-(0{,}6346)^2)}} = -0{,}4987;$$

$$t = \frac{-0{,}4987}{\sqrt{1-(-0{,}4987)^2}} \cdot \sqrt{20-(2+1)} = -2{,}37.$$

Складемо матрицю окремих коефіцієнтів кореляції (у дужках значення t-статистик):

	y	x_1	x_2
y	1,0	0,7513 (4,69)	0,7377 (4,51)
x_1	0,7513 (4,69)	1,0	-0,4987 (-2,37)
x_2	0,7377(4,51)	-0,4987 (-2,37)	1,0

Коефіцієнти кореляції показують «чисту» кореляцію пари змінних, що виключає вплив інших змінних, включених у рівняння. Таким чином, найбільш сильним є взаємозв'язок між вартістю перевезення та вагою вантажу. Однак зауважимо, що приватні коефіцієнти кореляції між y та x1, y та x2 свідчать про сильніші взаємозв'язки незалежних змінних із залежною, ніж це показують значення парних коефіцієнтів кореляції. Це сталося тому, що парний коефіцієнт кореляції завищив тісноту зв'язку між x1 та x2, занизивши при цьому тісноту зв'язку між y та x1, y та x2. Зазначимо також, що це приватні коефіцієнти кореляції статистично значущі.

ВИСНОВКИ

Дослідження процедури агрегування критеріїв показало, що результати агрегування сильно залежать від знань експертів та переваг методів дослідження статистичних показників. Навіть якщо завдання вибору вирішується однією людиною, то й у цьому випадку на різних етапах процедури агрегування критеріїв результати можуть більшою чи меншою мірою відрізнятись один від одного залежно від суджень людини, як обумовлених поглядом на проблему з різних сторін, так і продиктованих життєвим досвідом. .

У процесі виконання роботи з'ясувалося, що ступінь агрегування вихідних показників впливає на трудомісткість та пояснення результатів у завданні вибору.

Чим більше буде підсумкових критеріїв, тим складніше зробити правильний вибір та пояснити отримані результати. З іншого боку, занадто мала кількість підсумкових критеріїв у сукупності з неправильним агрегуванням показників на попередніх рівнях ієрархії може призвести до суперечностей, таким, що всі запропоновані варіанти при своїй очевидній відмінності можуть мати однакові оцінки за результатами порівняння.

Традиційні математичні моделі не дозволяють отримати необхідну точність прогнозування фінансових часових рядів, які близькі до випадкових. У зв’язку із цим, розробка моделей та методів, які призначені для прогнозування знаків приростів таких часових рядів має важливе наукове й практичне значення. Дослідження спрямоване на розробку такого методу, який дозволяв би виконувати прогноз знаку приросту часового ряду з максимальною точністю. При цьому може використовуватися апарат фрактального аналізу інформації, підходи інтелектуального аналізу часових рядів, зокрема індексації за методами найближчого сусіда і K-найближчих сусідів тощо.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Анісімов А.В. Інформаційні системи та бази даних: Навчальний посібник для студентів факультету комп'ютерних наук та кібернетики. / Анісімов А.В., Кулябко П.П. − Київ. – 2017. – 110 с.

2. Антоненко В. М. Сучасні інформаційні системи і технології: управління знаннями : навч. Посібник / В. М. Антоненко, С. Д. Мамченко, Ю. В. Рогушина. – Ірпінь : Нац. університет ДПС України, 2016. – 212 с.

3. Воронін А. М. Інформаційні системи прийняття рішень: навчальний посібник. / Воронін А. М., Зіатдінов Ю. К., Климова А. С. − К. : НАУ-друк, 2009. − 136с.

4. Галузинський Г. П. Інформаційні системи у бізнесі. Практикум для індивідуальної роботи: навчметод. посіб. для самост. вивч. Дисципліни. / Галузинський Г. П., Денісова О. О., Писаревська Т. А. − К. : КНЕУ, 2008. − 524с.

5. Годун В.М. Інформаційні системи і технології в статистиці: навч. посіб. / В.М. Годун, Н.С. Орленко, М. А. Сендзюк; за ред. В.Ф. Ситника. – К.: КНЕУ, 2003. – 267 с.

6. Грицунов О. В. Інформаційні системи та технології: навч. посіб. для студентів за напрямом підготовки «Транспортнітехнології» / О. В. Грицунов; Харк. нац. акад. міськ. госп-ва. – Х.: ХНАМГ, 2010. – 222 с.

7. Інформаційні системи в економіці : навч. посібник / Пономаренко В. С., Золотарьова І. О., Бутова Р. К. та ін. – Х. : Вид. ХНЕУ, 2011. – 176 с.

8. Інформаційні системи в промисловості : навчальний посібник / Л. О. Добровольська, О. О. Черевко. – Маріуполь : ПДТУ, 2014. – 238 с.

9. Інформаційні системи в сучасному бізнесі : навчальний посібник / В. С. Пономаренко, І. О. Золотарьова, Р. К. Бутова та ін. – Х. : Вид. ХНЕУ, 2011. – 484 с.

10. Інформаційні системи і технології в банківській сфері: навч. посіб. для студ. спец. 6.050105 "Банківські справи". / Аніловська Г. Я., Чуй І. Р., Вус М. Л., Стоколоса Т. М. − Л. : ЛКА, 2008. – 332 с.

11. Програмування числових методів мовою PYTHON / А. Ю. Дорошенко, за ред. А. В. Анісімова. – ВПЦ "Київський університет", 2013. – 464 с.

12. Шаховська Н. Б. Алгоритми та структури даних / Н. Б. Шаховська, Р.О. Голощук. – Львів : Магнолія-2006. – 2009. – 216 с.

13. Шеховцов В.А. Операційні системи / В.А. Шеховцов. – К. : Видавнича група BHV, 2005. – 576 с.

14. Мнушка О.В., Савченко В.М. Конспект лекцій з дисципліни «Об’єктно-орієнтоване програмування для студентів напрямів підготовки 121 «Інженерія програмного забезпечення» та 122 «Комп’ютерні науки». – Харків, ХНАДУ, 2020.

15. Data model. – URL: https://docs.python.org/3.8/reference/datamodel.html

PEP 0 – Index of Python Enhancement Proposals (PEPs).– URL: https://www.python.org/dev/peps/

16. Links to Python related information in Ukranian. – URL: https://wiki.python.org/moin/UkranianLanguage

17. Stackoverflow. – URL: https://stackoverflow.com/

18. Python Software Foundation. The Python Tutorial [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/tutorial/index.html

19. Python Software Foundation. Python 3.7.12 documentation [Електронний ресурс]. – Режим доступу: https://docs.python.org/3.7/

20. Lutz M. Learning Python, 5th Edition. – O’Reilly Media Inc., 2013. – 1648 p.

21. Chun Wesley J. Core Python Application Programming. Third Edition. – Pearson Education, Inc., 2012.

22. Lutz M. Programming Python, Forth Edition. – O'Reilly Media Inc., 2011.

23. Prometheus: CS50. Вебпрограмування з Python та JavaScript CS50. – Prometheus. – 2021. [Електронний ресурс] – Режим доступу: https://courses.prometheus.org.ua/courses/coursev1:Prometheus+CS50+2021_T1/about

24. Сороко Н.В. Інтеграція сучасних інформаційно-комунікаційних технологій у навчальний процес: зарубіжний та вітчизняний досвід / Наукові записки. – Випуск 77. - Серія: Педагогічні науки. - Кіровоград: РВВ КДПУ ім . В. Винниченка. - 2008. Частина 1. - 354 с., с. 113 - 118.

25. Хмельов О. Г., Хмельова А. В., Інформаційні технології і засоби навчання. — 2009. — №5 (13). — [Електронний ресурс]: сайт Інституту інформаційних технологій і засобів навчання НАПН України. — Режим доступу: http://www.ime.eduua.net/em.html.

Додаток А

Код програми

```
# ---------------------- Підключення бібліотек ------------------------------
------
import tensorflow as tf
import pandas as pd
import os as os
import numpy as np
#from sklearn.model_selection import train_test_split
#import matplotlib.pyplot as plt
import time
from sklearn import metrics
os.environ['TF_CPP_MIN_LOG_LEVEL']='2'

# change working directory
working_dir = 'E:/code 13_05'
os.chdir(working_dir)

# ---------------- Читання даних ----------------------
# read in data
train = pd.read_csv("data/train.csv")
test = pd.read_csv("data/test.csv")
col_for_x = ['mix_mv_avg','5_price_diff','mv_avg_diff',
'avg_quantity','quantity_price','ct_rising','aux_flag', 'price']
x_train = np.array(train[col_for_x])
x_test = np.array(test[col_for_x])
y_train_encoded = np.array(pd.get_dummies(train.label))
y_test_encoded = np.array(pd.get_dummies(test.label))
```

```
y_test = np.array(test.label)

# формування моделі навчання
cnn_input = []
for i in range(x_train.shape[0]-50):
    row = []
    for k in range(50):
        for t in range(8):
            row.append(x_train[i+k,:][t])
    cnn_input.append(row)
x_train_grid = pd.DataFrame(cnn_input)   # df
check = y_train_encoded[0:-50,:]

# тест
cnn_input_test = []
for i in range(x_test.shape[0]-50):
    row = []
    for k in range(50):
        for t in range(8):
            row.append(x_test[i+k,:][t])
    cnn_input_test.append(row)
x_test_grid = pd.DataFrame(cnn_input_test)
y_test_grid = y_test[0:-50]
y_test_grid_encoded = y_test_encoded[0:-50, :]


height = 50
width = 8
channels = 1
```

```
n_inputs = 400  # 50*8


conv1_fmaps = 36    # 36 # 18
conv1_ksize = 2
conv1_stride = 1
conv1_pad = 'SAME'

conv2_fmaps = 72
conv2_ksize = 2
conv2_stride = 1
conv2_pad = 'SAME'


pool3_dropout_rate = 0.25
pool3_fmaps = conv2_fmaps


n_fc1 = 128      # 144
fc1_dropout_rate = 0.5 # 0.5


n_outputs = 3
learning_rate = 0.001


batch_size = 50

tf.compat.v1.reset_default_graph()
```

```
tf.compat.v1.disable_eager_execution()
with tf.name_scope("inputs"):
    X = tf.compat.v1.placeholder(tf.float32, shape=[None, n_inputs], name="X")
    X_reshaped = tf.reshape(X, shape=[-1, height, width, channels])
    y = tf.compat.v1.placeholder(tf.int32, shape=[None, n_outputs], name="y")
    training = tf.compat.v1.placeholder_with_default(False, shape=[], name='training')


conv1 = tf.compat.v1.layers.conv2d(X_reshaped, filters=conv1_fmaps, kernel_size=conv1_ksize,
                                   strides=conv1_stride, padding=conv1_pad, activation=tf.nn.relu,
                                   name='conv1')
conv2 = tf.compat.v1.layers.conv2d(conv1, filters=conv2_fmaps, kernel_size=conv2_ksize,
                                   strides=conv2_stride, padding=conv2_pad, activation=tf.nn.relu,
                                   name='conv2')

with tf.name_scope("pool3"):
    pool3 = tf.nn.max_pool(conv2, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding="VALID") #1, 2, 1, 1
    pool3_flat = tf.reshape(pool3, shape=[-1, pool3_fmaps*25*4]) # will go to fully connected dense layers, so reshape to 1-d tensor
```

```
        pool3_flat_drop = tf.compat.v1.layers.dropout(pool3_flat, pool3_dropout_rate, training=training) # training placeholder, False.


    with tf.name_scope("fc1"):
        fc1 = tf.compat.v1.layers.dense(pool3_flat_drop, n_fc1, activation=tf.nn.relu, name="fc1")# width of layer: n_fc1
        fc1_drop = tf.compat.v1.layers.dropout(fc1, fc1_dropout_rate, training=training)

    with tf.name_scope("output"):
        logits = tf.compat.v1.layers.dense(fc1_drop, n_outputs, name="output")
        Y_proba = tf.nn.softmax(logits, name="Y_proba")

    with tf.name_scope("loss"):
        xentropy = tf.nn.softmax_cross_entropy_with_logits(logits=logits, labels=y)
        loss = tf.reduce_mean(xentropy, name="loss")
        loss_summary = tf.summary.scalar('log_loss', loss)

    with tf.name_scope("train"):
        optimizer = tf.optimizers.Adam(learning_rate) # GradientDescentOptimizer #MomentumOptimizer , momentum=0.9
        training_op = optimizer.minimize(loss,var_list = None)

    with tf.name_scope("eval"):
        correctPrediction = tf.equal(tf.argmax(Y_proba, 1), tf.argmax(y, 1))
        accuracy = tf.reduce_mean(tf.cast(correctPrediction, tf.float32))
        accuracy_summary = tf.summary.scalar('accuracy', accuracy)
```

```
    #    correct = tf.nn.in_top_k(logits, y, 1)
    #    accuracy = tf.reduce_mean(tf.cast(correct, tf.float32))

    with tf.name_scope("init_and_save"):
        init = tf.global_variables_initializer()
        saver = tf.train.Saver()

    def get_model_params():
        gvars = tf.get_collection(tf.GraphKeys.GLOBAL_VARIABLES)  # get all the variables in the computation graph
        return {gvar.op.name: value for gvar, value in zip(gvars, tf.get_default_session().run(gvars))}

    def restore_model_params(model_params):
        gvar_names = list(model_params.keys())
        assign_ops = {gvar_name: tf.get_default_graph().get_operation_by_name(gvar_name + "/Assign")
                      for gvar_name in gvar_names}
        init_values = {gvar_name: assign_op.inputs[1] for gvar_name, assign_op in assign_ops.items()}
        feed_dict = {init_values[gvar_name]: model_params[gvar_name] for gvar_name in gvar_names}
        tf.get_default_session().run(assign_ops, feed_dict=feed_dict)

    def batch_func(X, y, batch_size):
        batches = []
        for i in range(0, len(X), batch_size):
            X_batch = X[i: i+batch_size]
            y_batch = y[i: i+batch_size]
```

```
            mini_batch = (X_batch, y_batch)
            batches.append(mini_batch)
        return batches


n_epochs = 10
iteration = 0

best_loss_val = np.infty
check_interval = 10  # 500
checks_since_last_progress = 0
max_checks_without_progress = 10  # 20
best_model_params = None

merged = tf.summary.merge_all()


start = time.time()
with tf.Session() as sess:
    init.run()
    train_writer = tf.summary.FileWriter('./graphs/train', tf.get_default_graph()) # train_dropout_0.2 #train_layer_2
    test_writer = tf.summary.FileWriter('./graphs/test', tf.get_default_graph()) # train_MomentumOptimizer_optimizer
    for epoch in range(n_epochs):
        for a_batch in batch_func(x_train_grid, check, batch_size):
            (X_batch, y_batch) = a_batch
            sess.run(training_op, feed_dict={X: X_batch, y: y_batch, training: True})
```

```
            test_summary, acc = sess.run([merged, accuracy], feed_dict={X: x_test_grid, y: y_test_grid_encoded})
            test_writer.add_summary(test_summary, epoch)
            print('Accuracy at step %s: %s' % (epoch, acc))


            train_summary, _ = sess.run([merged, training_op], feed_dict={X:X_batch, y:y_batch}) # , training_op
            train_writer.add_summary(train_summary, epoch)

        train_writer.close()
        test_writer.close()

        save_path = saver.save(sess, "./CNN_stock_model.ckpt")


    with tf.Session() as sess:
        saver.restore(sess, "./CNN_stock_model.ckpt")
        Z = Y_proba.eval(feed_dict = {X: x_test_grid})
        y_pred = np.argmax(Z, axis = 1) - 1   # np.argmax return the index, e.g. 0, 1, 2 for -1, 0, 1, so subtract 1.
    print(pd.crosstab(y_test_grid, y_pred, rownames=['Actual'], colnames=['Predicted']))
    print(metrics.classification_report(y_test_grid,y_pred))
    print("Accuracy: " + str(metrics.accuracy_score(y_test_grid,y_pred)))
    print("Cohen's Kappa: " + str(metrics.cohen_kappa_score(y_test_grid,y_pred)))
    print('Took: %f seconds' %(time.time() - start))
```

```
'''
    if best_model_params:
        restore_model_params(best_model_params)
    acc_test = accuracy.eval(feed_dict={X: x_test_grid, y: y_test_grid})
    print("Final accuracy on test set:", acc_test)
  '''


{
  "nbformat": 4,
  "nbformat_minor": 0,
  "metadata": {
    "colab": {
      "provenance": []
    },
    "kernelspec": {
      "name": "python3",
      "display_name": "Python 3"
    },
    "language_info": {
      "name": "python"
    }
  },
  "cells": [
    {
      "cell_type": "code",
      "execution_count": 2,
      "metadata": {
        "colab": {
```

```
      "base_uri": "https://localhost:8080/"
    },
    "id": "f7TMn5kFquGL",
    "outputId": "013379a3-54da-463d-954f-9617a4664937"
   },
   "outputs": [
    {
     "output_type": "stream",
     "name": "stdout",
     "text": [
      "Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz\n",
      "11490434/11490434 [==============================] - 0s 0us/step\n",
      "train_images shape (60000, 28, 28)\n",
      "train_labels shape (60000,)\n"
     ]
    }
   ],
   "source": [
    "from keras.datasets import mnist\n",
    "# get training and testing dataset\n",
    "(train_images, train_labels), (test_images, test_labels) = mnist.load_data()\n",
    "print('train_images shape', train_images.shape)\n",
    "print('train_labels shape', train_labels.shape)"
   ]
  },
  {
```

```
"cell_type": "code",
"source": [
  "import numpy as np\n",
  "def preprocessing(arr):\n",
  "  arr = np.expand_dims(arr, axis=-1)\n",
  "  arr = arr.astype('float32')/255.0\n",
  "  return arr\n",
  "train_images = preprocessing(train_images)\n",
  "test_images = preprocessing(test_images)\n",
  "print('train_images shape', train_images.shape)\n",
  "print('test_images shape', test_images.shape)\n"
],
"metadata": {
  "colab": {
    "base_uri": "https://localhost:8080/"
  },
  "id": "6bH7ZYhKKsiX",
  "outputId": "7b1b3eef-ead1-48fa-c72d-952da6e45c96"
},
"execution_count": 3,
"outputs": [
  {
    "output_type": "stream",
    "name": "stdout",
    "text": [
      "train_images shape (60000, 28, 28, 1)\n",
      "test_images shape (10000, 28, 28, 1)\n"
    ]
  }
```

```
  ]
 },
 {
  "cell_type": "code",
  "source": [
   "from keras.utils import to_categorical\n",
   "train_labels = to_categorical(train_labels)\n",
   "test_labels = to_categorical(test_labels)\n",
   "print('train_images shape', train_images.shape)\n",
   "print('test_images shape', test_images.shape)"
  ],
  "metadata": {
   "colab": {
    "base_uri": "https://localhost:8080/"
   },
   "id": "auA5vwtULK35",
   "outputId": "abcc0a2a-c659-428a-e4eb-a95e775ae455"
  },
  "execution_count": 4,
  "outputs": [
   {
    "output_type": "stream",
    "name": "stdout",
    "text": [
     "train_images shape (60000, 28, 28, 1)\n",
     "test_images shape (10000, 28, 28, 1)\n"
    ]
   }
  ]
```

```
},
{
 "cell_type": "code",
 "source": [
  "from keras.layers import Conv2D, MaxPooling2D, Dense, Flatten\n",
  "from keras.models import Sequential\n",
  "num_filters = 8\n",
  "filter_size = 3\n",
  "pool_size = 2\n",
  "model = Sequential()\n",
  "model.add(Conv2D(num_filters, filter_size, input_shape=(28, 28, 1)))\n",
  "model.add(MaxPooling2D(pool_size=pool_size))\n",
  "model.add(Flatten())\n",
  "model.add(Dense(10, activation='softmax'))\n",
  "model.summary()"
 ],
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/"
  },
  "id": "-uMzHtulNdNY",
  "outputId": "a9e90931-42c1-472c-802c-8d28662e3ba6"
 },
 "execution_count": 5,
 "outputs": [
  {
   "output_type": "stream",
```

```
          "name": "stdout",
          "text": [
           "Model: \"sequential\"\n",
"_________________________________________________________________\n",
           " Layer (type)                 Output Shape              Param #   \n",
"=================================================================\n",
           " conv2d (Conv2D)              (None, 26, 26, 8)         80        \n",
           "                                                                 \n",
           " max_pooling2d (MaxPooling2D  (None, 13, 13, 8)         0         \n",
           " )                                                               \n",
           "                                                                 \n",
           " flatten (Flatten)            (None, 1352)              0         \n",
           "                                                                 \n",
           " dense (Dense)                (None, 10)                13530     \n",
           "                                                                 \n",
"=================================================================\n",
           "Total params: 13,610\n",
           "Trainable params: 13,610\n",
           "Non-trainable params: 0\n",
"_________________________________________________________________\n"
```

```
      ]
     }
   ]
},
{
 "cell_type": "code",
 "source": [
   "model.compile(\n",
   "optimizer='adam',\n",
   "loss='categorical_crossentropy',\n",
   "metrics=['accuracy'],\n",
   ")\n",
   "model.summary()"
 ],
 "metadata": {
   "colab": {
     "base_uri": "https://localhost:8080/"
   },
   "id": "9BOwTaOUNnxO",
   "outputId": "6b91a9ba-7362-48af-9ec7-c2d6c3c3a467"
 },
 "execution_count": 6,
 "outputs": [
   {
     "output_type": "stream",
     "name": "stdout",
     "text": [
       "Model: \"sequential\"\n",
```

```
"_________________________________________________________________\n",
                " Layer (type)                 Output Shape              Param #   \n",
"=================================================================\n",
                " conv2d (Conv2D)              (None, 26, 26, 8)         80        \n",
                "                                                                 \n",
                " max_pooling2d (MaxPooling2D  (None, 13, 13, 8)         0         \n",
                " )                                                               \n",
                "                                                                 \n",
                " flatten (Flatten)             (None, 1352)              0         \n",
                "                                                                 \n",
                " dense (Dense)                 (None, 10)                13530     \n",
                "                                                                 \n",
"=================================================================\n",
                "Total params: 13,610\n",
                "Trainable params: 13,610\n",
                "Non-trainable params: 0\n",
"_________________________________________________________________\n"
              ]
            }
          ]
```

```
},
{
 "cell_type": "code",
 "source": [
  "model.fit(\n",
  "train_images, #training data\n",
  "train_labels, #training targets\n",
  "epochs=5,\n",
  "batch_size=32,\n",
  ")"
 ],
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/"
  },
  "id": "PnTGzujLN5M9",
  "outputId": "bcfb68da-c49b-4c71-b334-d81d5fe4dddd"
 },
 "execution_count": 7,
 "outputs": [
  {
   "output_type": "stream",
   "name": "stdout",
   "text": [
    "Epoch 1/5\n",
    "1875/1875 [==============================] - 20s
10ms/step - loss: 0.3387 - accuracy: 0.9040\n",
    "Epoch 2/5\n",
```

```
        "1875/1875 [==============================] - 20s
11ms/step - loss: 0.1782 - accuracy: 0.9488\n",
        "Epoch 3/5\n",
        "1875/1875 [==============================] - 19s
10ms/step - loss: 0.1306 - accuracy: 0.9624\n",
        "Epoch 4/5\n",
        "1875/1875 [==============================] - 19s
10ms/step - loss: 0.1061 - accuracy: 0.9694\n",
        "Epoch 5/5\n",
        "1875/1875 [==============================] - 20s
11ms/step - loss: 0.0910 - accuracy: 0.9734\n"
       ]
      },
      {
       "output_type": "execute_result",
       "data": {
        "text/plain": [
         "<keras.callbacks.History at 0x7f87bd9c7b50>"
        ]
       },
       "metadata": {},
       "execution_count": 7
      }
     ]
    },
    {
     "cell_type": "code",
     "source": [
      "test_loss, test_acc = model.evaluate(\n",
```

```
    "    test_images,\n",
    "    test_labels\n",
    "    )\n",
    "print('test accuracy', test_acc)"
   ],
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "B3AQowbZO-8x",
    "outputId": "f65aa29d-8819-4ec4-b051-a8dfb9ba3466"
   },
   "execution_count": 8,
   "outputs": [
    {
     "output_type": "stream",
     "name": "stdout",
     "text": [
      "313/313 [==============================] - 2s 6ms/step - loss: 0.0887 - accuracy: 0.9708\n",
      "test accuracy 0.97079998254776\n"
     ]
    }
   ]
  },
  {
   "cell_type": "code",
   "source": [
    "model.save('model.h5')"
```

```
  ],
  "metadata": {
    "id": "9fNixdb5SFnr"
  },
  "execution_count": 9,
  "outputs": []
},
{
  "cell_type": "code",
  "source": [
    "test_loss, test_acc = model.evaluate(\n",
    "    test_images,\n",
    "    test_labels\n",
    "    )\n",
    "print('test accuracy', test_acc)"
  ],
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/"
    },
    "id": "LyU4Vx5ESiNL",
    "outputId": "26d6f0f9-6dd6-46e1-cbe0-9202f8c475a8"
  },
  "execution_count": 10,
  "outputs": [
    {
      "output_type": "stream",
      "name": "stdout",
      "text": [
```

```
"313/313 [==============================] - 2s 6ms/step - loss: 0.0887 - accuracy: 0.9708\n",
"test accuracy 0.97079998254776\n"
]
}
]
},
{
"cell_type": "code",
"source": [
"from keras.models import load_model\n",
"loaded_model = load_model('model.h5')\n",
"model.compile(\n",
"optimizer='adam',\n",
"loss='categorical_crossentropy',\n",
"metrics=['accuracy'],\n",
")"
],
"metadata": {
"id": "KYdwkvNASkU9"
},
"execution_count": 12,
"outputs": []
},
{
"cell_type": "code",
"source": [
"import numpy as np\n",
"num_tested = (10,12)\n",
"predicted_images = test_images[num_tested[0]:num_tested[1]]\n",
"predictions = model.predict(predicted_images)\n",
```

```
            "pred_labels = np.argmax(predictions, axis=1)\n",
            "true_labels = np.argmax(test_labels[num_tested[0]:num_tested[1]], axis=1)\n",
            "print('predicted labels', pred_labels)\n",
            "print('true labels', true_labels)\n"
          ],
          "metadata": {
            "colab": {
              "base_uri": "https://localhost:8080/"
            },
            "id": "IixAsL_JSujP",
            "outputId": "cf91de8a-9dee-4710-9ac5-2c54e680a89c"
          },
          "execution_count": 15,
          "outputs": [
            {
              "output_type": "stream",
              "name": "stdout",
              "text": [
                "1/1 [==============================] - 0s 18ms/step\n",
                "predicted labels [0 6]\n",
                "true labels [0 6]\n"
              ]
            }
          ]
        }
      ]
    }
```

ДОДАТОК Б

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	113 с.
	Інші документи	
2.	Презентаційні матеріали	__ плакатів

.	.	Прізвище та ініціали.	Підп.	Дата		
Розробив		Ляшенко Д.Л.			Шифр групи	Код напр./спец.
Перевірив		Дейнеко А.О.			ДСзм-21-1	122
Н.контр.		Малєєва І.А.			ХНУРЕ кафедра ШІ	
Затв.		Філатов В.О.				

Еволюційна інкрементна нейронна мережа в задачах опрацювання потоків даних