Вісник Харківського національного університету імені В. Н. Каразіна. Серія «Фізика». Вип. 42, 69–73 (2025)
Journal of V. N. Karazin Kharkiv National University. Series Physics. Iss. 42, 69–73 (2025)
DOI: 10.26565/2222-5617-2025-42 ISSN 2222-5617 (Print)

Original article
In print article
https://doi.org/10.26565/2222-5617-2025-42-09
UDC 004.8, 004.45, 004.9
PACS numbers: 01.40, 02.60, 01.50.H

МЕТОДИКА СТВОРЕННЯ ІНТЕГРОВАНОГО СЕРЕДОВИЩА ДОСЛІДНИКА НА БАЗІ JUPITER NOTEBOOK З ВИКОРИСТАННЯМ НЕЙРОМЕРЕЖ

Ю. В. Литвинов, Г. В. Клименко

Харківський національний університет імені В. Н. Каразіна, майдан Свободи, 4, 61022 Харків, Україна

Надійшла до редакції 03 квітня 2025 р. Переглянуто 10 травня 2025 р.
Прийнято до друку 17 травня 2025 р.

У роботі представлено методику створення інтегрованого середовища дослідника для обробки експериментальних даних на базі Jupyter Notebook із використанням можливостей штучного інтелекту, зокрема генеративних нейромереж. Jupyter Notebook є провідним середовищем серед дослідників завдяки своїй гнучкості, широкому набору бібліотек і зручності в інтеграції різних інструментів аналітики. Основне призначення Jupyter Notebook – створення індивідуальних програмних рішень для дослідників шляхом написання коду понад 40 мовами програмування.

Традиційно розробка дослідницьких інструментів у Jupyter Notebook вимагає написання програмного коду мовою Python. Однак, це може становити певні труднощі для фахівців, які не володіють глибокими навичками програмування. Із стрімким розвитком генеративних нейромереж з'явилася унікальна можливість створювати невеликі, спеціалізовані програми для власних потреб без значного занурення у програмування. Такий підхід не лише спрощує процес створення прикладних програм, але й значно прискорює освоєння мов програмування, знижуючи бар'єр входження до професії розробника. У роботі розглянуто основні можливості Jupyter Notebook, зроблено короткий огляд інтерфейсу Jupyter Notebook, надано елементарні пояснення принципів створення програм з використанням нейромереж. Важливим кроком у створенні інструментів дослідника є складання опису функціоналу програми та організації інтерфейсу. З огляду на те, що створення програмного коду здійснюється з використанням нейромереж, особливу увагу приділено принципам написання промптів для ефективного генерування коду та створення програм для автоматизації обробки даних різних форматів. Наведено конкретні приклади розробки функціональних модулів, що демонструють можливості адаптації нейромережевих моделей для вирішення типових завдань обробки експериментальних даних. Стаття орієнтована на дослідників-експериментаторів, які прагнуть розширити свої аналітичні можливості за допомогою сучасних нейромережевих технологій, уникаючи складного програмування.

Ключові слова: *Jupyter Notebook, нейромережі, Python.*

Як цитувати: *Ю. В. Литвинов, Г. В. Клименко. Методика створення інтегрованого середовища дослідника на базі Jupiter Notebook з використанням нейромереж.* Вісник ХНУ імені В. Н. Каразіна. Серія «Фізика». Вип. 42, 2025, 69–73. https://doi.org/10.26565/2222-5617-2025-42-09

In cites: *Yu. V. Lytvynov, G. V. Klymenko. Methodology for creating an integrated researcher environment based on Jupiter Notebook using neural networks.* Journal of V. N. Karazin Kharkiv National University. Series Physics. Iss. 42, 2025, 69–73. https://doi.org/10.26565/2222-5617-2025-42-09 (in Ukrainian).

ВСТУП

Jupyter Notebook – це інтерактивне обчислювальне середовище, яке дозволяє створювати інструменти дослідника і ділитися документами, що містять код, математичні рівняння, візуалізації та текстові пояснення. Спочатку розроблене як частина проекту IPython, це середовище стало одним із найпоширеніших інструментів для наукових обчислень та обробки даних у різних галузях, зокрема у фізиці [1].

Основною перевагою Jupyter Notebook є можливість інтеграції коду та результатів його виконання безпосередньо у структурі документа. Це дозволяє дослідникам та викладачам не лише розв'язувати задачі й будувати моделі, але й наочно демонструвати логіку обчислень, проводити числові експерименти та аналізувати отримані результати.

Для фізиків Jupyter Notebook надає потужні інструменти для чисельного розв'язання диференціальних рівнянь, аналізу великих масивів даних, візуалізації даних, симуляцій фізичних процесів і явищ, інтерактивних демонстрацій. Крім того, середовище надає можливість створювати інтерактивні елементи та використовувати потужні графічні інтерфейси для покращення розуміння та презентації результатів [2].

Таким чином, Jupyter Notebook є не лише інструментом для розрахунків, але й платформою для інтегрованого викладання, навчання та досліджень у галузі фізики.

ІНТЕРФЕЙС JUPYTER NOTEBOOK

Jupyter Notebook, на відміну від багатьох інтегрованих середовищ розробки, має інтерфейс максимально наближений за структурою до інтерфейсів користувацьких програм і складається з кількох основних елементів**.**

1. **Меню (Menu Bar):** доступ до основних функцій – створення нового файлу, збереження, експортування, зміна ядра, налаштування вигляду тощо.
2. **Панель інструментів (Toolbar):** набір основних команд для керування документом: запуск коду, збереження, вставлення комірок, зміна типу комірки.
3. **Робоча область (Notebook):** основна область, де розташовуються комірки. Комірки можуть містити код, текст (Markdown), графічні елементи та інтерактивні віджети.
4. **Комірки (Cells):** базові елементи, що можуть бути кодом або текстом. Кожна комірка виконується окремо і може бути змінена чи переміщена.
5. **Консоль повідомлень:** відображає статус виконання коду та повідомлення про помилки або успішне виконання команд.Такий розподіл інтерфейсу дозволяє зручно організувати процес розробки, тестування та візуалізації розрахунків у єдиному робочому просторі.

НЕЙРОМЕРЕЖІ ТА ПРОМПТИ ДЛЯ ГЕНЕРАЦІЇ PYTHON-КОДУ

Генерацію Python-коду проєктів Jupiter Notebook можна здійснювати з використанням великої кількості нейромереж [6]. Нами тестувалися виключно нейромережі, які мають безкоштовний план для роботи індивідуальних користувачів. Такими є: ChatGPT, Claude, Gemini, Code Llama, CodiumAI, DeepSeek R - 1 та Amazon CodeWhisperer. З огляду на те, що розвиток технологій штучного інтелекту відбувається швидкими темпами, важко визначити переможця із списку вказаних інструментів. Найкращим вибором для початку роботи будуть ChatGPT і Gemini. Gemini останнім часом показує кращі результати ніж ChatGPT. Однак, користуватись варто обома нейромережами. Огляд підходів до промпт програмування описано у багатьох роботах [3, 4]. Згенерований однією мережею код можна надавати іншій для його експертної оцінки, а також для узгодження формулювань промптів – завдань для нейронної мережі. Як же складати промпти для нейромереж? Розглянемо основні вимоги до складання промптів для генерації Python-коду.

1. Звдання формулюйте однозначно, включаючи вказівку на необхідні бібліотеки, структуру даних та тип очікуваного результату.
2. Вказуйте чітку структуру коду: імпортування бібліотек, основна логіка, візуалізація даних.
3. Зазначайте, що код має бути адаптований для Jupyter Notebook, враховуючи можливість виведення графіків, інтерактивних елементів та markdown-комірок.
4. За необхідності надавайте тестові дані або формати вхідних даних, щоб уникнути неоднозначності.
5. Рекомендується просити нейромережу давати розгорнуті коментарі для пояснення ключових етапів коду, особливо якщо це навчальний матеріал.
6. Вказуйте на необхідність обробки можливих помилок (наприклад, при роботі з файлами чи при побудові графіків).

Розглянемо найпростіший приклад роботи у середовищі Jupiter Notebook з використанням нейромереж. Зазначимо, що для початку немає сенсу обирати спеціалізовані нейромережі, призначені для написання програмного коду. Chat gpt або Gemini з успіхом впораються з більшістю завдань. Сформулюємо завдання для нейромережі – «промпт» і введемо його у поле для вводу промптів Chat gpt: «Працюємо у Jupiter Notebook. Напиши код для виводу повідомлення за умови натискання кнопки «Друк»: "Добрий день, це мій перший проєкт у Юпітер Ноутбук". Юпітер ноутбук – англійською мовою, Шрифт Times New Roman, 20, жирний».

Згенерований по цьому запиту код слід скопіювати, вставити у комірку Jupiter Notebook і натиснути кнопку «Run». Зверніть увагу на те, що промпт написаний людською мовою без докладних пояснень, але Chat gpt з легкістю виконає завдання. У промпті нічого не

сказано про те, що кнопку «Друк» треба створити. У цьому випадку така неоднозначність у промпті не вплине на працездатність коду. При формулюванні складніших завдань такий підхід не дасть позитивних результатів.

Помилки у коді більш складних і структурованих завдань виникають з низки причин. Якщо узагальнити, то їх можна поділити на три групи.

1. Вади промпту: промпт складено некоректно, занадто загально, непослідовно, відсутні приклади очікуваних результатів.
2. Помилки нейромережі: синтаксичні помилки, галюцинування мережі за рахунок неоднозначності трактування завдання.
3. Помилки, зумовлені особливостями програмного забезпечення: відсутність потрібних бібліотек, несумісні типи даних, несумісність версій програмних продуктів, вплив версії і налаштувань операційної системи та інше [5].

Зауважимо, що в процесі програмування людина стикається з цими ж проблемами. Для зменшення похибок проєкт має створюватись покроково з відладкою кожного наступного елементу проєкту.

Якщо це перші кроки в освоєнні нейромереж, бажано не просто викласти промпт на основі побажань, а для початку «обговорити» з електронним «колегою» проєкт і попросити його дати експертну оцінку свого проєкту. Експериментатор повинен точно знати, що нейромережа правильно зрозуміла завдання. Також у запиті треба попросити дати комментарі до коду і супутні пояснення. Якщо проєкт передбачає наявність складного інтерфейсу і великої кількості функцій, програмний код потрібно генерувати по частинах, додаючи нові функції до вже працюючого коду. Про це також потрібно попередити нейромережу. Нові функції можуть порушувати алгоритм роботи програми. У такому випадку генерується сповіщення про помилку у коді. Текст сповіщення про помилку слід скопіювати і надати нейромережі для аналізу помилки і виправлення коду. Якщо програма вже має великий розмір, краще не генерувати весь код цілком, а генерувати його фрагменти. Новий фрагмент коду потрібно вставити в існуючий текст програми, враховуючи, з яких модулів вона складається. Нейромережа генеруватиме кожен новий фрагмент з повним вмістом глобальних змінних, бібліотек та інших складових коду, тому при вставленні нового інструменту слід уникати дублювання існуючих складових коду. Наприклад, при додаванні нової функції програми у структуру інтерфейсу додається нова кнопка, або слайдер для регулювання відповідного параметра. При цьому у новому згенерованому фрагменті коду буде передбачено завантаження бібліотеки віджетів «import ipywidgets as widgets», яка була вже завантажена при створенні інших елементів інтерфейсу.

Розглянемо складові програми та послідовність цих складових у тексті коду:

- імпорт бібліотек;
- глобальні змінні;
- інтерфейс користувача (віджети);
- функції обробки данних;
- функції інтерактивності;
- відображення інтерфейсу.

Приємною особливістю роботи з нейромережами є те, що під час роботи над проєктом, електронний «співавтор» відповість на будь-які запитання стосовно програмування, підкаже, як зробити програму більш функціональною, запропонує додаткові корисні опції до програми, навчить писати коректні промпти для генерації коду. Щоб отримати кращий результат, при формулюванні завдань запитуйте, як нейромережа зрозуміла чергове завдання і, виходячи з цього, пропонувала кращий варіант досягнення поставленої мети. Існують стандартні засоби і алгоритми розв'язання типових завдань, однак, якщо на це не звернути уваги, нейромережа буде намагатись виконати завдання нераціональним шляхом, який ви їй запропонували.

Наведемо приклад роботи з нейромережею Gemini. «Розробляємо програму для використання у середовищі Jupiter Notebook. Створюємо код мовою Python. Програма для обробки фотознімків з металографічного мікроскопа. 1. Основні функції програми: завантаження фотознімків з файлу, або з камери мікроскопа; інвертування зображення; бінаризація зображення; виділення границь об'єктів за зміною кольору або контрасту; регулювання порогів чутливості до зміни контрасту або кольору; визначення і виділення кольоровим контуром об'єкта, групи об'єктів, або об'єктів в обмеженій зоні; обчислення довжини контуру, площі, та еквівалентного діаметра виділеного об'єкта або середніх значень цих параметрів для групи виділених об'єктів; зумування зображення; побудова гістограми розподілу об'єктів за еквівалентним діаметром. 2. Інтерфейс програми: інтерфейс з динамічним регулюванням розміру; до складу інтерфейсу входить вікно для відображення знімків; праворуч від вікна відображення знімків розташовуються дві або три вертикальних колонки для органів керування та відображення результатів обчислення; гістограма будуватиметься в окремому вікні. Проаналізуй промпт, розкажи, як ти розумієш завдання, зроби зауваження та пропозиції і склади план, за яким будемо працювати над програмою». Аналіз запиту відбувається протягом долей секунд і його результат відображається на екрані. Нажаль, розмір відповіді Gemini на такий запит не може бути відтворений тут повністю, але вкажемо на істотні моменти, які треба відслідкувати у відповіді нейромережі. Якщо Gemini правильно зрозуміла алгоритм роботи програми, структуру інтерфейсу, запропонувала шляхи розв'язання окремих завдань і спланувала роботу над програмою, можна починати генерувати код. Якщо у користувача немає достатнього досвіду програмування, краще покластися на досвід нейромережі і почати роботу з запиту: «З чого ти

пропонуєш почати?». Gemini запропонує згенерувати інтерфейс з органами керування, а потім покроково прив'язувати функції до органів керування. Після узгодження послідовності дій дайте команду на генерацію коду для створення інтерфейсу. Згенерований код слід скопіювати, вставити в комірку Jupiter Notebook і запустити, натиснувши кнопку «run». Якщо код спрацює і на екрані з'явиться інтерфейс майбутньої програми, його слід перевірити, скласти перелік зауважень щодо розташування елементів інтерфейсу та їх маркування і дати завдання Gemini виправити знайдені недоліки. Під час тестування коду можуть з'являтися сповіщення про помилки. Ці сповіщення треба копіювати і додавати нейромережі до свого текстового промпту про роботу коду. Після створення інтерфейсу програми переходьте до призначення функцій його елементам. Послідовність призначення функцій потрібно виконувати за алгоритмом роботи програми. Не слід братися за наступну функцію, якщо помічені вади в роботі програми. Деякі функції можуть не суміщатися з логікою роботи інших. У такому випадку опишіть проблему в промпті і нейромережа запропонує план вирішення проблеми. Кінцевий результат роботи зі створення програми показано на рис. 1.

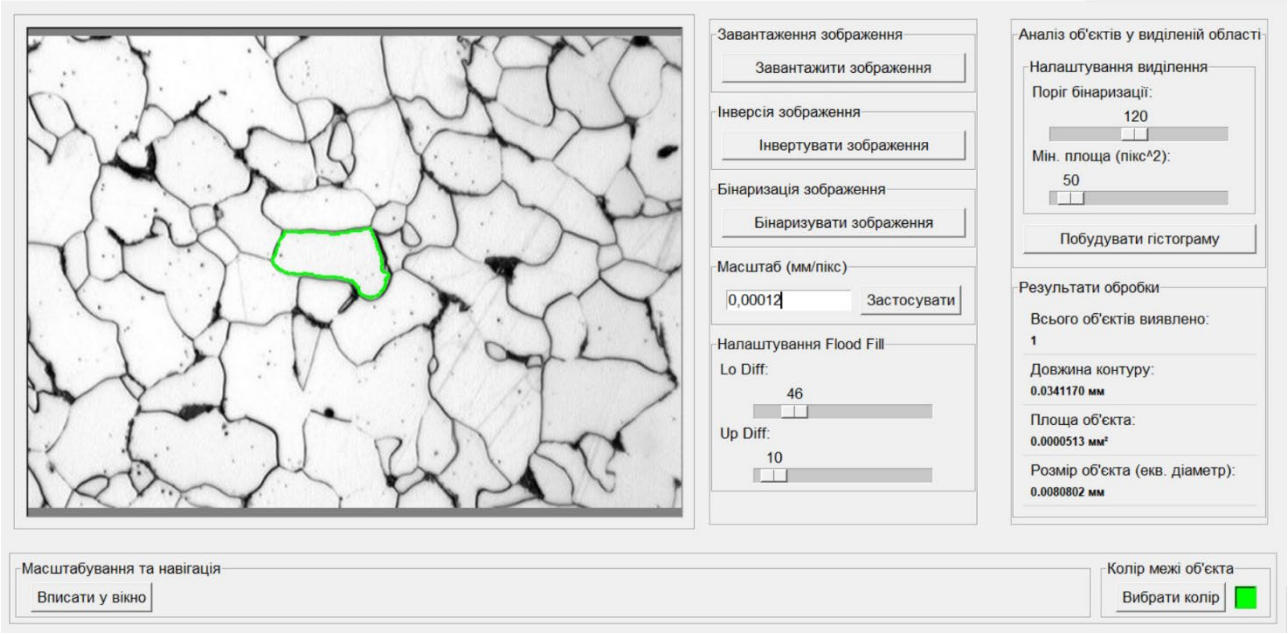

Рис. 1. Інтерфейс програми для обробки зображень і визначення параметрів кристалічної ґратки з виділеним фрагментом зображення.

Fig. 1. Interface of the program for image processing and determination of crystal lattice parameters with a selected image fragment.

ВИСНОВКИ

У роботі викладено основні методичні прийоми роботи зі створення програм для обробки експериментальних даних з використанням загальнодоступних нейромереж і інтегрованого середовища Jupiter Notebook. Розглянуто принципи створення промптів для генерування програмного коду Python, причини помилок та способи запобігання їх виникненню. Надано приклад створеної програми. Запропонований підхід до створення програм сприяє оволодінню мовами програмування, що підвищує професійний рівень дослідника.

КОНФЛІКТ ІНТЕРЕСІВ

Автори повідомляють про відсутність конфлікту інтересів.

CONFLICT OF INTEREST

The authors declare that they have no conflict of interests.

REFERENCES

1. J. Perkel. Nature, 563, 145 (2018). https://doi.org/10.1038/d41586-018-07196-1
2. T. Rule, B. Birmingham, D. Clayton, et al. PLoS Comput Biol 15(7) (2019). https://doi.org/10.1371/journal.pcbi.1007007
3. S. White. arXiv, 2302.11382 (2023). https://doi.org/10.48550/arXiv.2302.11382
4. Laria Reynolds, Kyle McDonell. arXiv, 2102.07350 (2021). https://doi.org/10.48550/arXiv.2102.07350
5. M. Chen, J. Tworek, H. Jun, et al. arXiv, 2107.03374 (2021). https://doi.org/10.48550/arXiv.2107.03374

METHODOLOGY FOR CREATING AN INTEGRATED RESEARCH ENVIRONMENT BASED ON JUPYTER NOTEBOOK USING NEURAL NETWORKS

Yu. V. Lytvynov, H. V. Klymenko

V. N. Karazin Kharkiv National University, Svobody Square 4, 61022 Kharkiv, Ukraine

Received on April 03, 2025. Revised May 10, 2025.
Accepted for publication May 17, 2025.

This paper presents a methodology for creating an integrated research environment for experimental data processing based on Jupyter Notebook with the use of artificial intelligence tools, particularly generative neural networks. Jupyter Notebook is a leading platform among researchers due to its flexibility, broad library ecosystem, and ease of integration with various analytical tools. Its primary purpose is to enable researchers to create customized software solutions by writing code in more than 40 programming languages.

Traditionally, the development of research tools in Jupyter Notebook requires coding in the Python programming language. However, this can pose challenges for specialists who lack deep programming skills. With the rapid development of generative neural networks, a unique opportunity has emerged to create small, specialized programs for personal use without significant immersion in coding. Moreover, this approach not only simplifies the creation of applied software but also significantly accelerates the acquisition of programming skills, lowering the entry barrier into the development profession.The paper reviews the key capabilities of Jupyter Notebook, provides a brief overview of its interface, and offers basic explanations of the principles of program creation using neural networks. A crucial step in building research tools is the formulation of software functionality and interface design. Given that code generation is performed using neural networks, particular attention is paid to prompt engineering principles for effective code generation and the creation of applications for automating the processing of data in various formats. Specific examples of functional module development are presented, demonstrating the adaptability of neural network models to address typical experimental data processing tasks. The article is intended for experimental researchers seeking to enhance their analytical capabilities using modern neural network technologies while avoiding complex programming.

Keywords: *Jupyter Notebook, neural networks, Python.*