UDC 004.415.532 DOI: 10.18664/ikszt.v30i1.326699

БАРКОВСЬКА О. Ю. к.т.н.,
НІ Я. С. к.т.н., ,
ЯНКОВСЬКИЙ О. А. к.т.н., ,
РОМАНЕНКО А. О. магістрант,
ПЕРЕТЯКА Є. О. магістрант
(кафедра електронних обчислювальних машин, Харківський національний університет радіоелектроніки)

Модель системи автоматизованого навантажувального тестування програмних застосунків із використанням методів штучного інтелекту

Abstract.. *Стаття присвячена розробленню моделі системи автоматизованого навантажувального тестування програмних застосунків із використанням методів штучного інтелекту, пропонуючи інноваційний підхід для підвищення ефективності тестування програмного забезпечення в умовах динамічного розвитку сучасних технологій. Метою роботи є розроблення моделі системи автоматизованого навантажувального тестування програмних застосунків. Дослідницькою складовою в запропонованій системі є оцінювання ефективності використання методів штучного інтелекту для розпізнавання реєстраційних форм, що може призвести до скорочення часових і ресурсних затрат, а також забезпечити високу точність і надійність результатів. У роботі досліджено застосування згорткових нейронних мереж (CNN) для розпізнавання форм на основі зображень та оптичного розпізнавання символів (OCR) для аналізу тексту, що дає змогу автоматично ідентифікувати типи вебформ, таких як логін чи реєстрація, для автоматичної генерації в подальших дослідженнях тестових даних за отриманими розпізнаними формами та проведення навантажувального тестування засобами Apache JMeter. Результати демонструють, що CNN досягає точності до 92 % з розпізнаванням складних реєстраційних форм, хоча і з певними коливаннями під час валідації, тоді як OCR виявляється ефективнішим для простих форм логіна з точністю 88.9 %. Інтеграція моделі з JMeter сприяє автоматичному створенню тестових сценаріїв і підвищенню якості тестування. Отримані результати підтверджують ефективність розробленої методики, що дає змогу суттєво зменшити трудовитрати і підвищити якість навантажувального тестування. Запропонований підхід має перспективи подальшого розвитку, зокрема вдосконалення алгоритмів розпізнавання та розширення застосування для різних типів програмних інтерфейсів.*

Ключові слова: *машинне навчання, тестування, навантажувальне тестування, штучний інтелект, CNN, OCR, розпізнавання, генерація, тестові сценарії, JMeter, модель, реєстрація, логін.*

Вступ

Навантажувальне тестування є критичним компонентом у забезпеченні стабільної роботи систем під час реальних умов експлуатації, особливо в епоху масштабних цифрових трансформацій [1], невід'ємною складовою розроблення програмного забезпечення, особливо для вебдодатків [2], які мають постійно підтримувати велику кількість одночасних користувачів без втрати продуктивності. Мета такого тестування – оцінювання поведінки системи під різними рівнями навантаження, виявлення вузьких місць і забезпечення саме тієї стабільності, яку хоче отримати продуктивна команда будь-якої компанії. Проте слід зазначити, що у традиційного підходу для створення навантажувальних тестів є деякі виклики:

- ручне створення сценаріїв (наприклад, для форми логіна з полями «Email» і «Password» потрібно вказати HTTP-запит, заголовки і тестові дані, що займає приблизно годину на одну форму);
- складність сучасних UI через використання динамічних інтерфейсів (AJAX, React, Angular), де елементи з'являються або змінюються асинхронно, що ускладнює точне моделювання поведінки користувача;
- адаптивність до змін, яка призводить до необхідності переписування сценаріїв зі зміною інтерфейсу.

У контексті життєвого циклу розроблення програмного забезпечення (SDLC) тестування займає головну позицію на кожному етапі. За дослідженням [3], штучний інтелект (ШІ) може бути ефективно інтегрований в етапи SDLC, наведені в табл. 1.

Таблиця

Застосування штучного інтелекту на етапах SDLC

Етап SDLC	Застосування ШІ	Очікувані результати
Планування та аналіз вимог	• автоматизований аналіз вимог; • прогнозування ризиків; • оптимізація ресурсів	• підвищення точності оцінювання проєкту; • зменшення ризиків; • ефективний розподіл ресурсів
Проєктування	• генерація архітектурних рішень; • валідація проєктних рішень; • оптимізація структури	• прискорення проєктування; • покращення якості архітектури; • зменшення технічного боргу
Розроблення	• генерація тестових сценаріїв; • передбачення дефектів; • оптимізація коду	• швидше розроблення; • менша кількість помилок; • більш якісний код
Тестування	• автоматизоване тестування; • інтелектуальний аналіз; • прогнозування проблем	• повне покриття тестами; • швидше виявлення помилок; • надійніші результати
Впровадження та підтримка	• моніторинг продуктивності; • предиктивна аналітика; • автоматизація підтримки	• стабільна робота системи; • швидше вирішення проблем; • краща підтримка користувачів

Незважаючи на критичну важливість навантажувального тестування, значна кількість сервісів усе ще покладається на мануальні методики проведення тестування, для яких характерні висока ресурсомісткість і схильність до банальних помилок за рахунок людського фактора. Як зазначено в роботі [4], розроблення автоматизованої моделі навантажувального тестування з можливістю інтеграції методів штучного інтелекту може суттєво трансформувати цей процес, забезпечуючи більш ефективне виконання тестових сценаріїв і підвищення якості кінцевого програмного продукту.

Отже, використання автоматизованих систем тестування, підкріплених технологіями штучного інтелекту, дає змогу не лише підвищити швидкість виконання тестів і зменшити великі затрати на ресурси, але й забезпечує доволі високий рівень точності та надійності результатів, що є критично важливим у динамічному середовищі розроблення програмного забезпечення.

Серед застосунків, призначених для тестування продуктивності та навантаження програмного забезпечення, які забезпечують перевірку, як система працює за великої кількості запитів, користувачів або операцій, можна виділити Apache JMeter, HP LoadRunner, MS Visual Studio, Gatling. Порівняння запропонованих застосунків за критеріями підтримуваних протоколів, вартості та відкритості коду наведено в табл. 2.

Порівняння сервісів для навантажувального тестування

Інструмент	Open source	Протокол	Ціна
Apache JMeter	✓	HTTP, JDBC, SMTP	Безкоштовно
HP LoadRunner	✗	Широкий спектр	Дорогий (для тестування великих корпоративних систем)
MS Visual Studio	✗	.NET, Web	Відносно дорогий
Gatling	✓	HTTP, WebSocket	Відносно дорогий

Результати порівняльного аналізу показали переваги Apache JMeter [5, 6]. Сервіс є одним із найстабільніших і легких у використанні, надає послуги безкоштовно, підтримує автоматизацію за рахунок багатого функціонала, дає змогу користувачам створювати розширені тестові плани, використовуючи графічний інтерфейс, без необхідності знання програмування. Тестові плани в JMeter складаються з різних компонентів, таких як зразки запитів, логіка контролерів, лісенери для збору та візуалізації даних, таймери, асерції тощо (рис. 1).

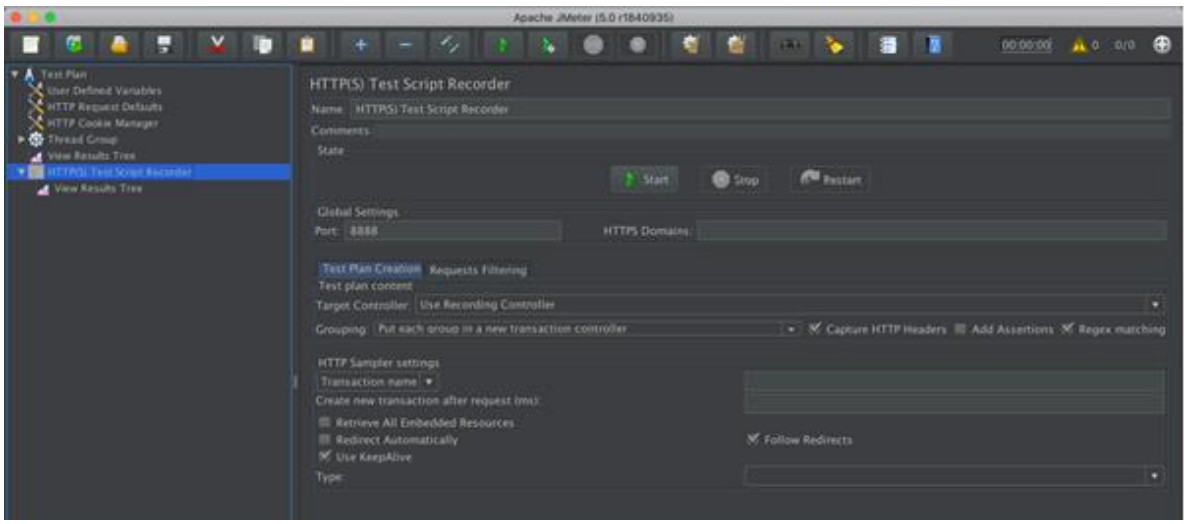


Рис. 1. Інтерфейс Apache Jmeter

Завдяки можливості моделювати різні сценарії користувацьких запитів JMeter є важливим інструментом для забезпечення надійності, стабільності та ефективності вебсервісів і додатків перед запуском їх у виробництво. Це дає змогу розробникам і тестувальникам ідентифікувати і усунути потенційні проблеми з продуктивністю, забезпечуючи кращий досвід користувача.

Аналіз останніх досліджень і публікацій

У сучасному світі розроблення програмного забезпечення все більшого значення набуває використання штучного інтелекту заради полегшення однотипних процесів, які займають багато часу. За останні три роки спостерігають значне зростання його використання і в процесах тестування [7]. Узагальнена систематизація типів тестування [8] наведена в табл. 3.

Систематизація типів тестування

Тип тестування	Підтип	Основні характеристики
Функціональне тестування	• модульне тестування; • інтеграційне тестування; • системне тестування; • приймальне тестування	• перевірка відповідності вимогам; • валідація функціональності; • верифікація взаємодії компонентів; • підтвердження готовності до експлуатації
Нефункціональне тестування	• навантажувальне тестування; • стрес-тестування; • тестування продуктивності; • тестування безпеки; • тестування зручності використання	• оцінювання продуктивності системи; • перевірка стійкості до навантажень; • аналіз швидкодії; • виявлення вразливостей; • оцінювання користувацького досвіду

Під час аналізу літератури було виявлено, що в роботі [9] розглянуто застосування глибоких нейронних мереж (DL) і великих мовних моделей (LLMs) для автоматичного виправлення вразливостей у програмному забезпеченні. Автори проводять емпіричне порівняння продуктивності п’яти різних LLMs і чотирьох APR (Automated Program Repair) моделей, оцінюючи їхню ефективність на наборах даних реальних вразливостей Java (Vul4J і VJBench). У роботі показано, що попередньо навчені великі мовні моделі, зокрема Codex, можуть частково автоматизувати виправлення програмних вразливостей. Проте не розглянуто специфічні особливості навантажувального тестування програмних застосунків, а увагу зосереджено виключно на виправленні вразливостей, що є лише частковою складовою тестування надійності.

Насамперед проведено ознайомлення з тими проєктами, які використовують НМ для розпізнання певних даних і зображень. Саме аналіз цих проєктів допоміг виявити наявні проблеми та подальший розвиток вибраного напряму.

UIED: UI Element Detection (https://github.com/MulongXie/UIED) — модель на основі YOLOv3 для детекції елементів інтерфейсу

(кнопки, текстові поля) для виявлення координат полів на скріншотах.

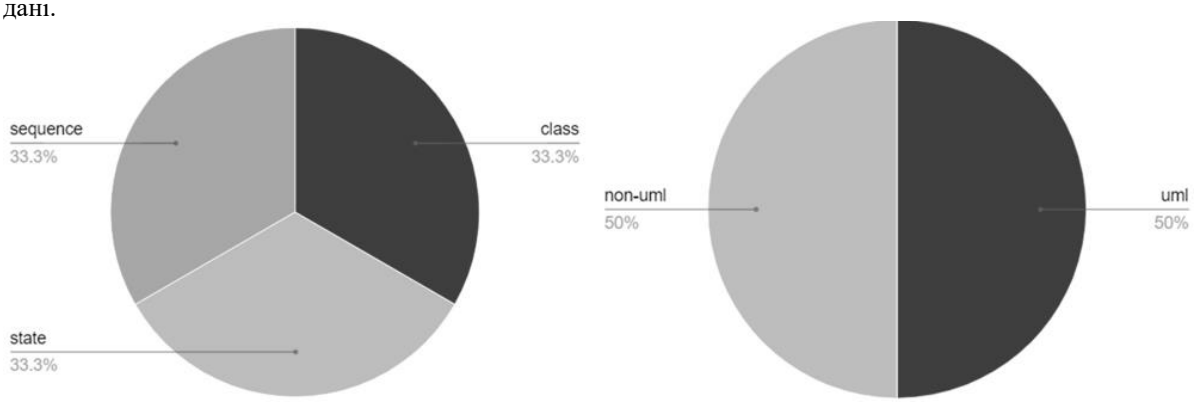
RICO Dataset + CNN (https://arxiv.org/abs/1707.03321_) — модель від Google Brain, навчена розпізнавати типи екранів з точністю до 91 %.

Donut (Document Understanding Transformer) (https://github.com/clovaai/donut) – Vision Transformer для розуміння структурованих документів. Розпізнає текст і його семантику (наприклад розрізняє «Username» і «Password»).

Проєкт DeepTraffic (https://github.com/lexfridman/deeptraffic) — це проєкт, який використовує глибокі нейронні мережі для моделювання та оптимізації руху транспортних засобів. Хоча проєкт не призначений безпосередньо для навантажувального тестування ПЗ, його підхід для моделювання складних систем може бути корисним для розроблення НМ, що генерують тестові дані.

Проєкт Synthetic Data Vault (SDV) (https://github.com/sdv-dev/SDV) — це бібліотека Python для генерації синтетичних даних за допомогою різних моделей, включаючи нейронні мережі. Вона дає змогу створювати реалістичні дані на основі наявних наборів, що може бути корисним для навантажувального тестування.

Проєкт TGAN (https://github.com/sdv-dev/TGAN) — це інструмент для генерації табличних даних за допомогою генеративно-змагальних мереж. Він може бути використаний для створення реалістичних тестових даних для навантажувального тестування. Слід зазначити, що в TGAN гарно запропонована класифікація даних, як і в роботі Irina-Gabriela [10], де дані також розбито на декілька класів і половину з них було використано через UML-діаграму (рис. 2).



а б

Рис. 2. Розподіл класів: а – розподіл типів даних; б – співвідношення джерел даних [8]

Серед недоліків розглянутих застосунків можна виділити те, що деякі з них не є закінченими та передбачають роботу користувача з кодом, що потребує специфічних знань для роботи з Linux.

Виявлені переваги та недоліки наявних рішень і результатів, відображених у наукових роботах, вказують на потребу в комплексному автоматизованому рішенні, яке може швидко, точно і без зайвих втрат генерувати навантажувальні тести, адаптуючись до складності сучасних систем. До критеріїв оцінювання підходів для тестування програмного забезпечення належать час створення одного тесту, точність розпізнавання полів, адаптивність до змін UI, витрати на ресурси, складність впровадження, обробка динамічних UI, гнучкість форматів даних, залежність від людини. Порівняння за наведеними критеріями мануального тестування і тестування на основі ШІ наведено в табл. 4.

Порівняння традиційного підходу і підходу на основі ШІ

Критерій	Мануальний підхід	Підхід на основі ШІ
Час створення одного тесту	Високий (1–2 год на форму залежно від складності)	Низький (5–10 с на форму)
Точність розпізнавання полів	Залежить від уважності тестувальника (70–90 %)	Висока (до 95 % за умови якісного навчання)
Адаптивність до змін UI	Низька (потрібне ручне оновлення сценаріїв)	Висока (перенавчання моделі на нових даних)
Витрати на ресурси	Високі (зарплати, час, навчання персоналу)	Низькі (після налаштування моделі)

Складність впровадження	Низька (базові знання тестування)	Висока (потрібні знання ML і CV)
Обробка динамічних UI	Складна (потребує аналізу коду або скриптів)	Проста (модель розпізнає патерни автоматично)
Гнучкість форматів даних	Обмежена (тільки відомі тестувальнику формати)	Висока (модель адаптована до різних мов/форм)
Залежність від людини	Висока (весь процес залежить від тестувальника)	Низька (автономна робота після навчання)

Вибір методу машинного навчання для розв'язання цих задач спирається на результати досліджень. У роботі [11] досліджено використання великих мовних моделей (LLMs) для різних завдань тестування програмного забезпечення, зокрема генерації тест-кейсів, пошуку багів і виправлення коду. Проте робота не надає конкретних рішень щодо інтеграції методів машинного навчання, зокрема LLM, у сучасні інструменти навантажувального тестування. У роботі [12] зазначено про важливість CNN у тестуванні, особливо для аналізу графічних інтерфейсів.

Проведений аналіз демонструє необхідність розроблення нового рішення, яке б дало змогу зекономити час користувача, можливість використання та проведення навантажувального тестування без досвіду роботи з кодом і поєднало переваги наявних систем із можливостями штучного інтелекту для створення більш ефективного та адаптивного інструменту навантажувального тестування, яке буде поєднувати в собі як розпізнання зображень, так і генерацію тестових даних для використання в розробленні тестових сценаріїв.

Мета та задачі роботи

Метою роботи є розроблення моделі системи автоматизованого навантажувального тестування програмних застосунків. Дослідницькою складовою в запропонованій системі є оцінювання ефективності використання методів штучного інтелекту для розпізнавання реєстраційних форм, що може призвести до скорочення часових і ресурсних затрат, а також забезпечити високу точність і надійність результатів.

Для досягнення поставленої мети слід вирішити такі завдання:

- аналіз сучасних підходів для розпізнавання елементів web-сторінок;
- створення моделі системи автоматизованого навантажувального тестування;
- оцінювання ефективності аналізаторів на основі методів машинного навчання в запропоновану систему для визначення типу форми (логін, реєстрація) на основі зображень реєстраційних форм;
- реалізація ідентифікації текстових міток (наприклад «Email», «Password», «Sign Up» , «Login»).

Кроками подальшого розвитку проєкту є генерація релевантних тестових даних для заповнення форм (наприклад випадкові email-адреси, паролі) і підготовка HTTP-запитів для JMeter, а також обробка результатів навантажувального тесту з JMeter.

У кінцевому результаті проєкту буде розроблено модель системи автоматизованого навантажувального тестування програмного застосунку на основі методів машинного навчання для імітації користувацької взаємодії та навантаження на системи в реальному часі.

Отримані наукові результати. Обговорення результатів

Запропоновану модель системи автоматизованого навантажувального тестування, яка включає методи машинного навчання та OCR для розпізнавання реєстраційних форм, наведено на рис. 3.

Рис. 3. Діаграма IDEF0 розробленої моделі системи

Автоматизоване навантажувальне тестування — це головна функція, яка охоплює весь процес. Вона отримує початкові дані (зображення та вимоги) і видає кінцевий звіт про продуктивність програмного тестованого застосунку. Процес автоматизованого навантажувального тестування включає основні етапи (рис. 4): підготовка даних, навчання моделі, розпізнавання форм, завантаження зображення, обробка через NN, обробка через OCR, виведення результатів NN, виведення результатів OCR, генерація тестових даних, створення сценаріїв тестування, виконання тестів, аналіз результатів.

Процес починається з A1, де готуються дані, які потім передаються в A2 для навчання моделі. Навчена модель з A2 використовується в A3 для розпізнавання форм. Результати розпізнавання з A3 стають входом для A4, де генеруються тестові дані. Тестові дані з A4 передаються в A5 для створення сценаріїв тестування. Сценарії з A5 використовуються в A6 для виконання тестів. Нарешті, результати тестів з A6 аналізуються в A7, щоб отримати звіт.

Для проведення досліджень, направлених на оцінювання ефективності аналізаторів на основі методів машинного навчання в запропоновану систему, для визначення типу форми (логін, реєстрація) на основі зображення реєстраційних форм було зібрано зображення форм для подальшого використання. Навчання моделі передбачає тренування штучного інтелекту (CNN) для розпізнавання форм для ідентифікації полів і типів форм за допомогою навченої моделі та OCR.

Подальші кроки описують тестування навченої моделі на реальних тестових даних. Зображення завантажується з файлу та готується для обробки. Це може бути зміна розміру чи перетворення формату. На виході цього етапу отримуємо підготовлене зображення, передане на блоки аналізу на основі CNN або OCR-системи.

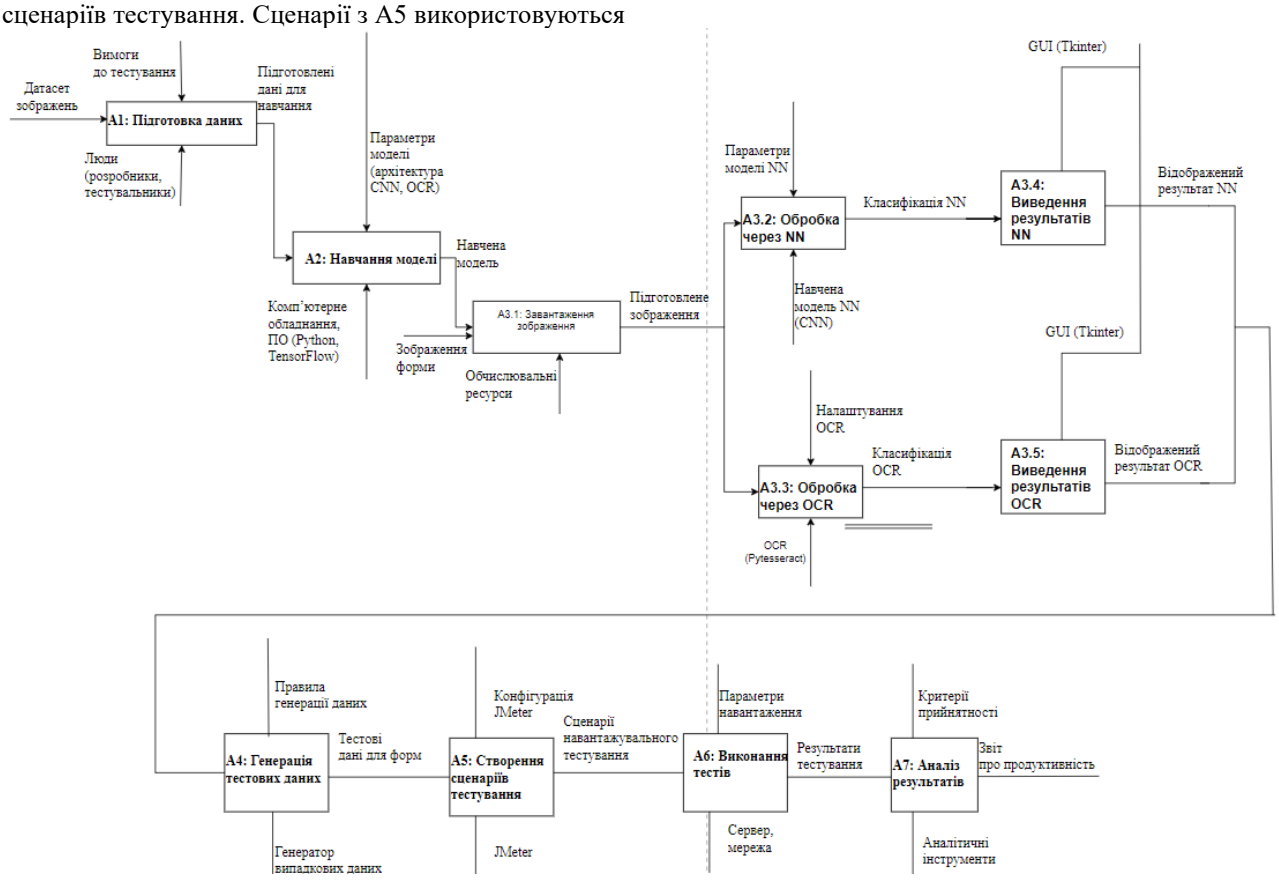


Рис. 4. Декомпозиція розробленої моделі системи автоматизованого навантажувального тестування

Далі як дослідження передбачена паралельна робота двох класифікаторів – аналіз завантаженого зображення на основі нейронної мережі (CNN) та OCR (Pytesseract). НМ аналізує зображення і видає класифікацію (наприклад Login, Registration, Combined). OCR (Pytesseract) розпізнає текст на зображенні і класифікує форму за ключовими словами.

Для НМ виведеними результатами є клас і впевненість мережі в отриманому результаті. Для OCR вихідним результатом є розпізнаний текст.

Отримані дані подаються на модуль генерації тестових даних для створення даних для заповнення форм на основі розпізнаних полів і подальшого

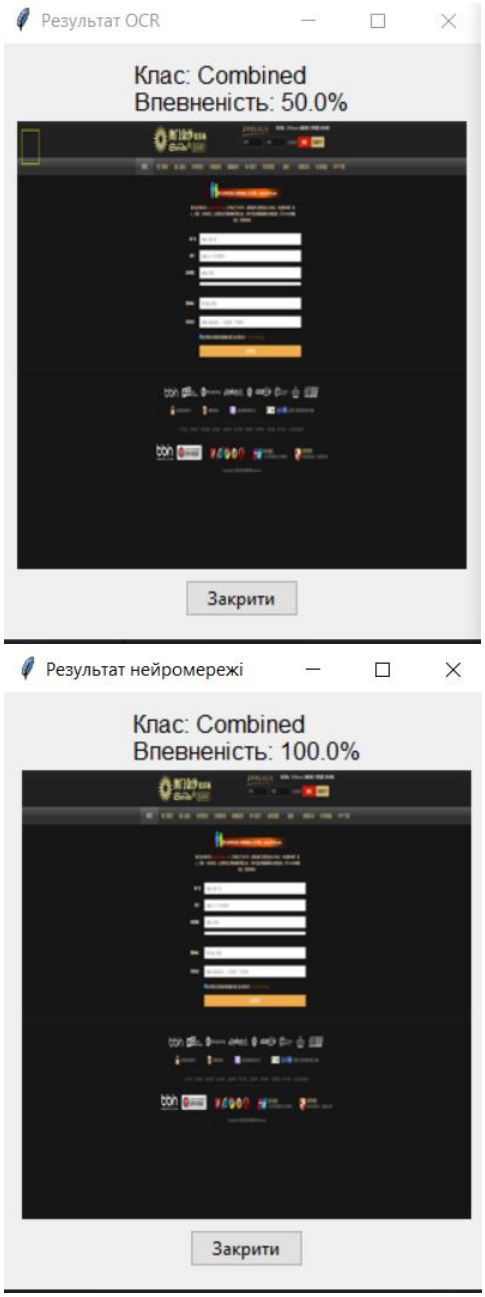
створення сценаріїв тестування для інструменту JMeter.

Підготовлені навантажувальні тести запускають на сервері для оцінювання продуктивності системи та підготовки звіту.

Для навчання і тренування вибраної нейронної мережі зі згортковою архітектурою використано наявний датасет (UI Elements Detection for Gambling Websites) з 1000 зображень вебформ із реальних сайтів, призначений для навчання моделей комп'ютерного зору, які можуть аналізувати, ідентифікувати і розпізнавати елементи інтерфейсу користувача (UI). Характеристики датасету:

- варіації кольорових схем, макетів і мовних версій, динамічні та статичні UI-елементи;
- зображення анотовані ключовими елементами UI з координатами bounding box для кожного елемента, такими як кнопки реєстрації/входу, поля введення (логін, пароль, сума ставки), меню навігації, кнопки ставок і виплат, таймери, банери з акціями, логотипи сайтів;
- метаданими є такі поля: URL-адреса джерела (опціонально), час збору даних, категорія сайту (ставки, казино, слоти тощо).

У моделі також запропонована інтеграція Tesseract OCR для розпізнавання текстових міток полів, що дало змогу підвищити точність і практичність системи. На першому етапі зображення проходить попередню обробку: конвертацію у відтінки сірого для спрощення аналізу, застосування гаусового розмиття з ядром 5×5 для усунення шуму та адаптивну бінаризацію за методом Оцу, щоб чітко відокремити текст від фону. Ці кроки підготовки забезпечують оптимальні умови для подальшого розпізнавання. Використовуючи бібліотеку Tesseract, система аналізує оброблене зображення, ідентифікує текстові фрагменти разом із їхніми координатами (bounding boxes) і порівнює їх із заздалегідь визначеним словником ключових міток, таких як «email», «password», «sign in» тощо. Ця процедура враховує багатомовність інтерфейсів, що критично важливо для роботи з глобальними платформами. Наступний крок — локалізація полів введення, реалізований через «розумне вирізання». Наприклад, для мітки «Password» алгоритм аналізує просторові взаємозв'язки між текстом і графічними елементами, використовуючи оператор Собеля для виявлення градієнтів яскравості. Це дає змогу точно визначити межі поля навіть у випадках часткового перекриття або нестандартного дизайну, уникаючи помилок, властивих традиційним методам на основі bounding boxes.

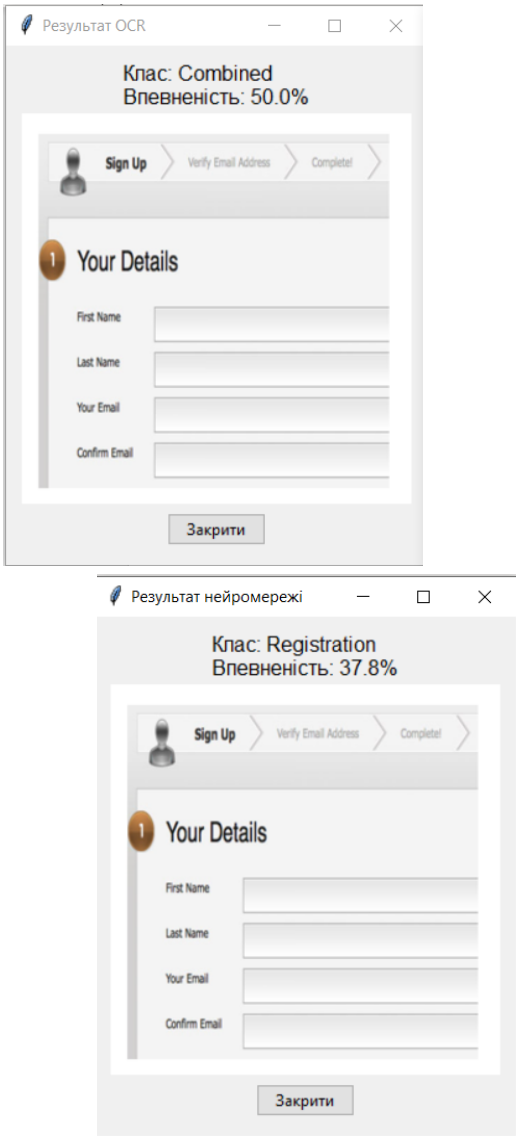


а б

Рис. 5. Результат розпізнавання комбінації логіна і реєстрації:
а – на основі OCR; б – на основі НМ

Порівняльні дослідження точності розпізнавання реєстраційних форм на основі CNN і OCR проведені на 30 валідаційних тестах, під час яких виявлено, що OCR на 32 % краще працює саме з формою Login, у той час як CNN краще розпізнає форми реєстрації за рахунок великого об'єму даних із датасету. Для комбінованої форми обидва методи працюють коректно, показуючи, проте, різну точність (рис. 5).

З рис. 5 видно, що OCR і CNN мають доволі велику різницю в розпізнавальній здатності до комбінованих форм, це може бути пояснене тим, що для OCR доволі складно даються складні комбінації за рахунок схожості заготовлених слів для пошуку, чого не можна сказати про CNN, яка показує з ними великий рівень впевненості за рахунок того, що тренувальний набір дає змогу НМ навчитися відрізняти такі форми.

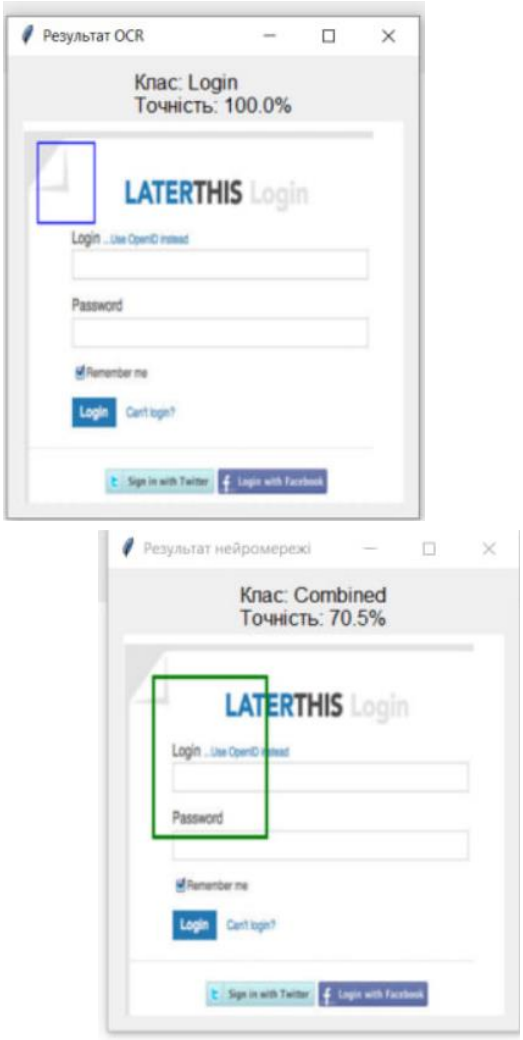


а б

Рис. 6. Результат розпізнавання форми реєстрації:
а – на основі OCR; б – на основі НМ

З рис. 6 видно, що OCR все ж таки нелегко даються форми реєстрації за рахунок їхньої складності і тому на таких формах цього методу завжди буде невисокий рівень впевненості, чого не можна сказати про CNN, яка навіть за нижчої впевненості дає коректний результат. Отже, можна сказати, що обидва методи мають потенціал для покращень.

З рис. 7 стає зрозуміло, що в OCR високий рівень впевненості саме для форми логіна, тому що такі форми досить легкі, їх можна без проблем розпізнати, але, як видно, НМ вони даються складніше, бо вона більше пристосована для складних форм.



а б

Рис. 7. Результат розпізнавання форми логіна:
а – на основі OCR; б – на основі НМ

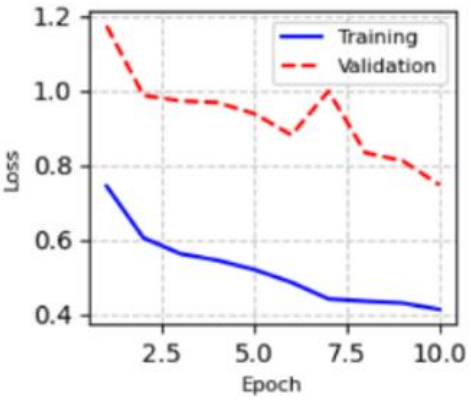
Нижче наведено детальний аналіз результатів тренування нейромережі (NN) на основі згорткової архітектури (CNN) порівняно з традиційним оптичним розпізнаванням символів (OCR). Модель складається з трьох згорткових шарів (Conv2D), кожен із яких має фільтри розміром 3x3 і функцію активації ReLU. Кількість фільтрів у цих шарах становить 32, 64 і 128 відповідно, що дає змогу мережі поступово витягувати більш складні ознаки із зображень. Після кожного згорткового шару застосовано шар максимального пулінгу

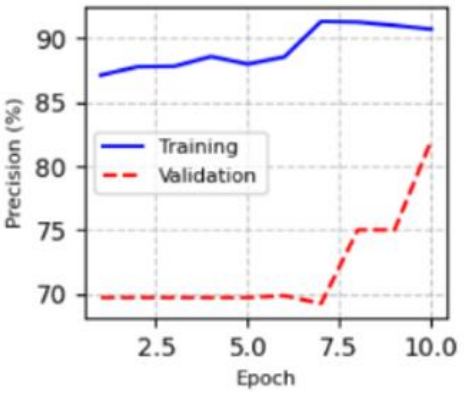
(MaxPooling2D) з розміром вікна 2x2 для зменшення просторових розмірів даних і зниження обчислювальної складності. Далі йде шар Flatten, який перетворює багатовимірні дані у вектор, після чого додано повнозв'язний шар (Dense) із 512 нейронами та функцією активації ReLU. Нарешті, вихідний шар складається з трьох нейронів (відповідно до кількості класів: Login, Registration, Combined) із функцією активації softmax, що забезпечує ймовірнісний розподіл для класифікації.

Модель була скомпільована з використанням оптимізатора Adam, який є адаптивним методом стохастичного градієнтного спуску, і функції вартості categorical_crossentropy, що є стандартним вибором для задач багатокласової класифікації. Для оцінювання якості моделі під час навчання використано метрику точності (accuracy).

Для реалізації моделі та обробки даних використано такі програмні засоби: бібліотека TensorFlow із модулем Keras для побудови та навчання CNN, OpenCV для попередньої обробки зображень, Tesseract OCR для розпізнавання тексту на вебформах, Tkinter для створення графічного інтерфейсу користувача, Matplotlib для візуалізації графіків навчання, Faker для генерації тестових даних, а також xml.etree.ElementTree для створення XML-файлів, які заплановано використовувати в подальшій інтеграції з JMeter.

Отримані результати описані на наданих скріншотах, які демонструють ключові метрики, такі як точність, втрати і час тренування. Також подано порівняльну характеристику між CNN та OCR з урахуванням переваг багатопоточності та ефективності моделі (рис. 8).





а

б

Рис. 8. Результати тренування та валідації описаної вище нейронної мережі:
а – графік втрат точності розпізнавання; б – графік точності розпізнавання

Проводячи експерименти, спостерігали перенавчання з неправильним розподілом датасету, що є поширеною проблемою в машинному навчанні, коли модель надмірно адаптована до тренувальних даних і втрачає здатність узагальнювати на нових даних. Тому для боротьби з цією проблемою застосовано методи, які діють на різних рівнях архітектури моделі та процесу її навчання – L2-регуляризацію з коефіцієнтом 0.001 застосовано до всіх трьох згорткових шарів, що формують основу архітектури; техніку Dropout використано після щільного шару з 256 нейронами з імовірністю відключення 0.5 і після щільного шару з 128 нейронами з імовірністю 0.3; підхід для автоматичного зниження швидкості навчання ReduceLROnPlateau з коефіцієнтом 0.2 для уникнення застрявань у локальних мінімумах і більш плавної збіжності; рання зупинка EarlyStopping для припинення навчання моделі, якщо валідаційні втрати не зменшаться протягом трьох епох.

Застосувавши наведені додаткові обмеження під час навчання нейронної мережі із загортковою архітектурою, досягнуто результати навчання, які наведено на рис. 8: навчання відбувається в напрямку зменшення втрат і збільшення точності розпізнавання реєстраційних форм, демонструючи здатність моделі адаптуватися, бо валідаційна точність після п'яти-шести епох значно покращується.

Процес навчання є достатньо стабільним, проте мережа не достатньо гарно узагальнює результати на валідаційних даних, про що можна судити за великою розбіжністю між тренувальними та валідаційними показниками. Причиною того, що модель може краще розпізнавати один клас, а не інший, може бути дисбаланс вибірки. Але підготовка збалансованої вибірки є складним завданням у випадку з реєстраційними формами, тому дослідження точності розпізнавання технікою OCR може показати кращі результати (табл. 5).

Порівняльна таблиця результатів розпізнання CNN та OCR

Номер картинки	Форма	Точність CNN, %	Точність OCR, %
1	Реєстрація	39.2	62.0
2	Комбінована	38.1	50.0
3	Комбінована	59.0	35.2
4	Логін	85.8	94.0
5	Логін	39.4	98.0
6	Реєстрація	91.4	50.0
7	Реєстрація	87.5	60.0
8	Логін	39.9	94.0
9	Комбінована	50.2	77.8
10	Реєстрація	57.4	51.0
11	Логін	85.8	94.5

Порівняльні результати показують, що загальна точність розпізнавання залежить від типу форми: логін-форми розпізнають найкраще обидва методи (CNN і OCR), маючи точність у діапазоні 85-98 %; комбіновані форми (містять елементи як реєстрації, так і логіна) розпізнавані гірше – точність CNN коливається від 38 до 59 %, OCR – від 35 до 77 %.

Отже, OCR у середньому працює стабільніше для текстових форм. Великий діапазон точності від 38 до 91 % свідчить про залежність від структури форми та недостатнє узагальнення моделі.

Оцінювання часу обробки запиту, тобто розпізнавання реєстраційних форм, використовуючи нейромережевий підхід або OCR, показує, що NN демонструє високу стабільність і швидкість, що робить його ефективнішим для задач, де важлива швидкість розпізнавання. OCR менш передбачуваний – його час обробки залежить від особливостей кожного зображення, що може бути проблемою для реального використання в системах реального часу (рис. 9).

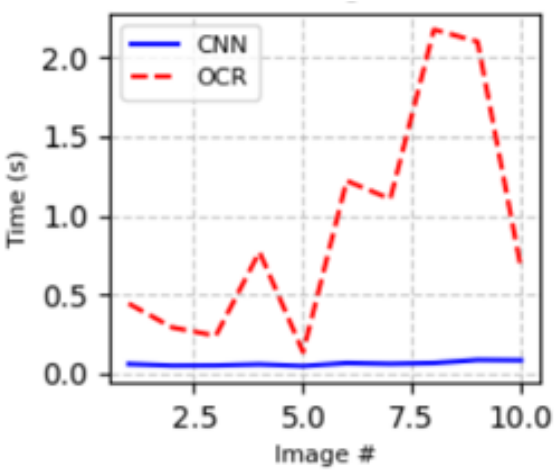


Рис. 9. Час обробки зображень двома методами: CNN та OCR

Висновки

Проаналізовано досить великий об’єм інформації, від наукових статей, присвячених теоретичним прототипам НМ та аналізу доречного використання ШІ у тестуванні, до різних повноцінних проєктів, недоліки яких послугували доволі сильним поштовхом для покращення розробленої моделі. У результаті дослідження розроблено модель системи автоматизованого навантажувального тестування програмних застосунків. Запропонована система включає аналізатор на основі методів машинного навчання для визначення типу форми на вебсторінці на основі зображення. Результат аналізу подано за допомогою графічного інтерфейсу, реалізованого через Tkinter. Для порівняння було досліджено ефективність розпізнавання полів реєстраційних форм на основі OCR. Обидва підходи довели здатність ідентифікації текстових міток типу «Email» або «Password» вісьмома різними мовами.

Кроками для подальшого розвитку проєкту є генерація релевантних тестових даних для заповнення відповідних форм за допомогою нового типу НМ, як GAN або LTSM, і підготовка HTTP-запитів для JMeter для подальшої обробки результатів найнавантажувального тесту для визначення його ефективності і можливості вдосконалення протестованого проєкту.

Список використаних джерел

1. Hourani, Hussam, et al. The Impact of Artificial Intelligence on Software Testing. *IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*, IEEE. 2019. P. 565–570. *DOI.org (Crossref):* https://doi.org/10.1109/JEEIT.2019.8717439.

2. Pu Yunming and Mingna Xu. Load Testing for Web Applications. *First International Conference on Information Science and Engineering*, IEEE. 2009. P. 2954–2957. *DOI.org (Crossref)*: https://doi.org/10.1109/ICISE.2009.720.

3. Barenkamp Marco et al. Applications of AI in Classical Software Engineering. *AI Perspectives*. Dec. 2020. Vol. 2, no. 1. P. 1. *DOI.org (Crossref)*: https://doi.org/10.1186/s42467-020-00005-4.

4. Baqar M., Khanda R. The Future of Software Testing: AI-Powered Test Case Generation and Validation //arXiv preprint arXiv:2409.05808. 2024. https://doi.org/10.48550/arXiv.2409.05808.

5. Tiwari Vatsya et al. Analytical Evaluation of Web Performance Testing Tools: Apache JMeter and SoapUI. *IEEE 12th International Conference on Communication Systems and Network Technologies (CSNT)*, IEEE. 2023. P. 519–523. *DOI.org (Crossref)*: https://doi.org/10.1109/CSNT57126.2023.10134699.

6. Beben Sutara, Sandy Shultan Gunawan. COMPARATIVE ANALYSIS OF REST API PERFORMANCE BETWEEN EXPRESS.JS FRAMEWORK AND HAPI.JS USING APACHE JMETER. *Jurnal Riset Teknik Informatika*. 2024. 1(1). P. 19–26.

7. Kacheru Goutham. AI-POWERED TEST AUTOMATION FRAMEWORKS: CHOOSING THE RIGHT TOOLS. *INTERNATIONAL JOURNAL OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING*. Jan. 2025. Vol. 3, no. 2. P. 221–30. DOI.org (Crossref): https://doi.org/10.34218/IJAIML_03_02_018.

8. Okezie F. et al. A Critical Analysis of Software Testing Tools. *Journal of Physics: Conference Series*. Dec. 2019. Vol. 1378, no. 4. P. 042030. *DOI.org (Crossref)*: https://doi.org/10.1088/1742-6596/1378/4/042030.

9. Wu Yi et al. How Effective Are Neural Networks for Fixing Security Vulnerabilities. *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, ACM. 2023. P. 1282–94. https://doi.org/10.1145/3597926.3598135.

10. Nedelcu Irina-Gabriela and Anca Daniela Ionita. Evaluating the Conformity to Types of Unified Modeling Language Diagrams with Feature-Based Neural Networks. *Applied Sciences*. Oct. 2024. Vol. 14, no. 20. P. 9470. *DOI.org (Crossref)*: https://doi.org/10.3390/app14209470.

11. Junjie Wang, Yuchao Huang, Chunyang Chen, Zhe Liu, Song Wang, and Qing Wang. Software Testing With Large Language Models: Survey, Landscape, and Vision. *IEEE Trans. Softw. Eng*. 50. 4 (April 2024). 911–936. https://doi.org/10.1109/TSE.2024.3368208.

12. Wydyanto and Maria Ulfa. Exploring Text Recognition Segmentation and Detection in Natural Scene Images. *INTI Journal*. Dec. 2024. Vol. JODS, no. 2024. DOI.org (Crossref): https://doi.org/10.61453/jods.v2024no66.

Model of an Automated Load Testing System for Software Applications Using Artificial Intelligence Methods

Abstract. *The article is dedicated to the development of a model for an automated load testing system for software applications using artificial intelligence methods, offering an innovative approach to enhancing software testing efficiency in the context of the dynamic advancement of modern technologies. The objective of this study is to develop a model of an automated load testing system for software applications.*

The research component of the proposed system involves evaluating the effectiveness of artificial intelligence methods for recognizing registration forms, which can lead to a reduction in time and resource costs while ensuring high accuracy and reliability of results. The study investigates the application of convolutional neural networks (CNNs) for image-based form recognition and optical character recognition (OCR) for text analysis, enabling the automatic identification of web form types, such as login or registration. This allows for the automated generation of test data based on recognized forms in subsequent research and the execution of load testing using Apache JMeter. The results demonstrate that CNN achieves an accuracy of up to 92 % in recognizing complex registration forms, albeit with certain fluctuations during validation, whereas OCR proves to be more effective for simpler login forms, with an accuracy of 88.9 %. The integration of the model with JMeter facilitates the automatic creation of test scenarios and enhances the quality of testing. The obtained results confirm the effectiveness of the proposed methodology, enabling a significant reduction in labor costs and an improvement in load testing quality. The proposed approach has promising prospects for further development, particularly in improving recognition algorithms and expanding its application to various types of software interfaces.

Keywords*: machine learning, software testing, load testing, artificial intelligence, CNN, OCR, recognition, generation, test scenarios, JMeter, model, registration, login.*

About the authors

Барковська Олеся Юріївна, кандидат технічних наук, доцент кафедри електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Харків, Україна. E-mail: olesia.barkovska@nure.ua. ORCID ID http://orcid.org/0000-0001-7496-4353.
Olesia Barkovska, Candidate of Technical Sciences, Associate Professor at the Department of Electronic Computers, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine.
E-mail: olesia.barkovska@nure.ua. ORCID ID http://orcid.org/0000-0001-7496-4353.

Ні Яна Самвелівна, кандидат технічних наук, старший викладач кафедри електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Харків, Україна. E-mail: yana.movsesian@nure.ua. ORCID ID http://orcid.org/0000-0002-1352-700X.
Yana Ni, Candidate of Technical Sciences, Senior Lecturer of the Department of Electronic Computers, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine.
E-mail: yana.movsesian@nure.ua. ORCID ID http://orcid.org/0000-0002-1352-700X.

Янковський Олександр Аркадійович, кандидат технічних наук, доцент кафедри електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Харків, Україна.
E-mail: oleksandr.yankovskyi@nure.ua. ORCID ID https://orcid.org/0000-0002-1268-0029.
Oleksandr Yankovskyi, Candidate of Technical Sciences, Associate Professor at the Department of Electronic Computers, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine.
E-mail: oleksandr.yankovskyi@nure.ua. ORCID ID https://orcid.org/0000-0002-1268-0029.

Романенко Антон Олександрович, магістрант кафедри електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Харків, Україна.
E-mail: anton.romanenko@nure.ua. ORCID ID https://orcid.org/0009-0009-2758-0758.
Anton Romanenko, master's student of the Department of Electronic Computers, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine.
E-mail: anton.romanenko@nure.ua. ORCID ID https://orcid.org/0009-0009-2758-0758.

Перетяка Євгеній Олександрович, магістрант кафедри електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Харків, Україна.
E-mail: yevhenii.peretiaka@nure.ua. ORCID ID https://orcid.org/0009-0005-1074-2651.

Yevhenii Peretiaka, master's student of the Department of Electronic Computers, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine.
E-mail: yevhenii.peretiaka@nure.ua. ORCID ID https://orcid.org/0009-0005-1074-2651.