Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Модель автоматизованого навантажувального тестування програмних застосунків з використанням методів штучного інтелекту
(тема)

Виконав:
здобувач 2 року навчання,
групи СПм-23-4
Антон РОМАНЕНКО
(власне ім'я, прізвище)

Спеціальність
123 «Комп'ютерна інженерія»
(код і повна назва спеціальності)

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

Освітня програма
Системне програмування
(повна назва освітньої програми)

Керівник: ст.викл. Яна НІ
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп'ютерна інженерія»
(код і повна назва)

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

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 20___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Романенку Антону Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Модель автоматизованого навантажувального тестування програмних застосунків з використанням методів штучного інтелекту

затверджена наказом по університету від "21" квітня 2025 р. № 296Стс

2. Термін подання здобувачем роботи до екзаменаційної комісії 16 червня 2025 р.

3. Вхідні дані до роботи
датасет для навчання моделі нейромережевого аналізу, датасет для навчання генеративної нейромережевої моделі

4. Перелік питань, що потрібно опрацювати у роботі
Огляд проблемної області автоматизованого навантажувального тестування
Аналіз нейромережевих моделей для розпізнавання змісту реєстраційних форм
Аналіз генеративних НМ моделей
Створення моделі автоматизоаного навантажувального тестування
Проведення емпіричного дослідження точності розпізнавання реєстраційних форм

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій Слайд-презентація: 14 слайдів

6. Консультанти розділів роботи (заповнюється 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	Строк / терміни виконання етапів роботи	Примітка
	Огляд проблемної області автоматизованого	22.04.25-29.04.25	
	навантажувального тестування		
	Аналіз нейромережевих моделей для розпізна-	30.04.25-05.05.25	
	вання змісту реєстраційних форм		
	Аналіз генеративних НМ моделей	06.05.25-10.05.25	
	Створення моделі автоматизоаного	11.05.25-25.05.25	
	навантажувального тестування		
	Проведення емпіричного дослідження точності	26.05.25-05.06.25	
	розпізнавання реєстраційних форм		
	Оформлення матеріалів кваліфікаційної роботи	06.06.25-09.06.25	
	Подання кваліфікаційної роботи на рецензу-	10.06.25-12.06.25	
	вання		

Дата видачі завдання " 21 " квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ст.викл. Яна НІ
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 85 с., 23 рис., 6 табл., 1 дод., 22 джерела.

АВТОМАТИЗОВАНЕ НАВАНТАЖУВАЛЬНЕ ТЕСТУВАННЯ, МАШИННЕ НАВЧАННЯ, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА (CNN), ОПТИЧНЕ РОЗПІЗНАВАННЯ СИМВОЛІВ (OCR), ГЕНЕРАТИВНО-ЗМАГАЛЬНА МЕРЕЖА (GAN), APACHE JMETER, РОЗПІЗНАВАННЯ, ГЕНЕРАЦІЯ.

Метою кваліфікаційної роботи є розробка моделі системи автоматизованого навантажувального тестування програмних застосунків на основі методів машинного навчання для підвищення ефективності, точності та адаптивності процесу тестування.

У ході виконання кваліфікаційної роботи було проаналізовано сучасні підходи до розпізнавання графічних елементів веб-інтерфейсів та створено модель, що поєднує можливості згорткових нейронних мереж (CNN), оптичного розпізнавання символів (OCR), генеративно-змагальних мереж (GAN) та інструменту Apache JMeter. Запропонована система дозволяє автоматично ідентифікувати структуру форм (логін, реєстрація), генерувати відповідні тестові дані, будувати сценарії навантажувального тестування та аналізувати результати у реальному часі. Проведене тестування довело високу точність роботи моделі, зокрема при класифікації складних форм. Дослідження також окреслює перспективи використання мультимодальних моделей, інтеграцію з хмарними платформами та побудову low-code інтерфейсів. Результати роботи є актуальними для розробників веб-додатків та інженерів з якості, оскільки дозволяють оптимізувати та масштабувати процеси тестування у динамічних середовищах.

ABSTRACT

Master’s thesis: 85 pages, 23 figures, 6 tables, 1 appendix, 22 sources.

AUTOMATED LOAD TESTING, MACHINE LEARNING, CONVOLUTIONAL NEURAL NETWORK (CNN), OPTICAL CHARACTER RECOGNITION (OCR), GENERATIVE ADVERSARIAL NETWORK (GAN), APACHE JMETER, RECOGNITION, GENERATION.

The major goal of this thesis is to develop a model of an automated load testing system for software applications based on machine learning methods, aimed at increasing the efficiency, accuracy, and adaptability of the testing process.

In order to achieve this goal, the thesis explores current approaches to recognizing graphical elements of web interfaces and proposes a model that integrates convolutional neural networks (CNN), optical character recognition (OCR), generative adversarial networks (GAN), and the Apache JMeter tool. The proposed system enables automatic identification of form structures (e.g., login and registration), generation of appropriate test data, construction of load testing scenarios, and real-time analysis of results. The conducted experiments have demonstrated high accuracy of the model, particularly in classifying complex forms. The study also outlines the prospects of using multimodal models, integrating with cloud platforms, and building low-code interfaces. The results of the thesis are highly relevant for web application developers and quality assurance engineers, as they provide means to optimize and scale testing processes in dynamic environments.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

НМ – нейронна мережа

ПЗ – програмне забезпечення

ШІ – штучний інтелект

AI – штучний інтелект (англ., Artificial Intelligence)

AJAX – асинхронний JavaScript та XML (англ., Asynchronous JavaScript and XML)

API – інтерфейс прикладного програмування (англ., Application Programming Interface)

CNN – згорткова нейронна мережа (англ., Convolutional Neural Network)

CV – комп’ютерний зір (англ., Computer Vision)

GAN – генеративно-змагальна мережа (англ., Generative Adversarial Network)

GRU – блок з контрольованим забуванням у рекурентних мережах (англ., Gated Recurrent Unit)

HTML – мова розмітки гіпертексту (англ., HyperText Markup Language)

HTTP – протокол передавання гіпертексту (англ., HyperText Transfer Protocol)

IoT – інтернет речей (англ., Internet of Things)

IT – інформаційні технології (англ., Information Technology)

JDBC – з'єднання з базами даних у Java (англ., Java Database Connectivity)

LSTM – довготривала короткочасна пам’ять (англ., Long Short-Term Memory)

ML – машинне навчання (англ., Machine Learning)

MQTT – транспорт телеметричних повідомлень через чергу (англ., Message Queuing Telemetry Transport)

OCR – оптичне розпізнавання символів (англ., Optical Character Recognition)

RNN – рекурентна нейронна мережа (англ., Recurrent Neural Network)

SDLC – життєвий цикл розробки програмного забезпечення (англ., Software Development Life Cycle)

SDV – генерація синтетичних даних (англ., Synthetic Data Generation)

TGAN – тимчасова генеративно-змагальна мережа (англ., Temporal Generative Adversarial Network)

UI – користувацький інтерфейс (англ., User Interface)

UIED – виявлення елементів інтерфейсу користувача (англ., User Interface Element Detection)

WDS – бездротова розподільча система (англ., Wireless Distribution System)

YOLO – модель одноетапного детектування (англ., You Only Look Once)

ВСТУП

У сучасних умовах стрімкого розвитку інформаційних технологій ми можемо спостерігати як з кожним днем на просторах Інтернету з’являється все більше нових користувачів, які зацікавленні в різноманітних послугах та сервісах для покращення або полегшення їх життя. Таких користувачів безліч і часто ми можемо спостегірати картинку коли багато людей одночасно хочуть отримати певну доволі важливу інформацію на сайті свого университету, або ж купити білети на довгоочікувану зустріч, або ж банально привітати когось в Телеграмі з початком нового року, але не можуть зробити це, бо сервіс не витримав великого напливу людей. В умовах, коли навіть невелика затримка чи збій може призвести до серйозних економічних втрат, стає зрозуміло що саме якість програмного забезпечення набуває вирішального значення для забезпечення безперебійної роботи в бізнес-процесах та підтримки конкурентоспроможності в підприємствах. Проблематика забезпечення високої якості програмного забезпечення, особливо в контексті динамічних змін ринку та постійно зростаючих навантажень, є актуальною для сучасної інженерної думки.

На фоні цих викликів дедалі більше уваги приділяється впровадженню інноваційних технологій, здатних адаптувати системи до динамічних умов експлуатації. Сучасний етап розвитку штучного інтелекту відкриває нові горизонти для оптимізації різних процесів, серед яких особливе місце займає автоматизація аналізу, прогнозування та управління даними. Алгоритми машинного навчання та нейронні мережі демонструють здатність ефективно обробляти великі обсяги інформації, виявляти приховані закономірності та адаптуватися до змін у середовищі, що дозволяє значно підвищити продуктивність і точність прийняття рішень у різних галузях. Такий підхід має особливе значення в умовах, коли традиційні методи не завжди відповідають високим вимогам сучасних систем.

Переходячи до конкретних аспектів дослідження, варто зазначити, що навантажувальне тестування є ключовим інструментом перевірки якості програмного забезпечення. Воно дозволяє оцінити, як система поводиться під впливом високих навантажень, з’ясувати її межі стійкості та виявити потенційні «вузькі місця», які можуть призвести до збоїв у роботі. Використання традиційних методів, хоча й забезпечує базовий рівень перевірки, не завжди дає змогу точно відобразити динаміку змін у поведінці системи при різких коливаннях навантаження.

У зв’язку з цим інтеграція можливостей штучного інтелекту в процес навантажувального тестування відкриває нові перспективи для підвищення якості та ефективності тестування. Застосування нейронних мереж дозволяє не лише автоматизувати створення тестових сценаріїв, але й прогнозувати можливі збої, адаптувати сценарії до реальних умов експлуатації та забезпечувати більш гнучке управління ресурсами системи. Сучасні технології, що використовують алгоритми машинного навчання, є потужним інструментом для вдосконалення процесів тестування, адже вони здатні аналізувати великі масиви даних і приймати оптимальні рішення у режимі реального часу.

Таким чином, сучасні виклики у сфері забезпечення якості програмного забезпечення сприяють пошуку нових, більш ефективних методів тестування, де інтеграція штучного інтелекту з навантажувальним тестуванням стає важливим кроком на шляху до створення стабільних та адаптивних систем. Цей підхід дозволить не лише оптимізувати процес перевірки, але й забезпечити безперебійне функціонування програмних продуктів у умовах швидкоплинного ринку цифрових технологій.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність роботи

Сучасний процес розробки програмного забезпечення є структурованим ланцюгом взаємопов’язаних етапів проектування, які забезпечують перетворення концепції у функціональний продукт. Ці етапи включають аналіз вимог, проектування архітектури, реалізацію коду, інтеграцію компонентів, тестування, впровадження та подальшу підтримку. Кожен з них має вирішальне значення для якості кінцевого рішення, проте саме роль етапу тестування часто визначає успіх усього проекту. На цьому етапі не лише виявляються технічні недоліки, а й перевіряється здатність системи функціонувати в умовах, що імітують реальне середовище експлуатації. Наприклад, у контексті Agile-методологій тестування інтегрується в кожну ітерацію, що дозволяє оперативно адаптувати продукт до змін вимог і зменшити ризики накопичення критичних помилок.

Систематизація типів тестування ПЗ відображає різноманітність завдань, які необхідно вирішити для забезпечення всебічної перевірки системи. Залежно від цілей, тестування класифікують на багато різноманітних типів (рисунок 1.1).

У контексті сучасних підходів до забезпечення якості програмного забезпечення, систематизація типів тестування відіграє ключову роль у структуруванні процесів перевірки. Згідно з класифікацією, що базується на цілях та об’єктах дослідження, тестування поділяється на функціональне (спрямоване на перевірку виконання специфікацій) та нефункціональне (орієнтоване на оцінку атрибутів продуктивності, безпеки, масштабованості тощо). Саме в рамках нефункціонального тестування навантажувальне тестування займає особливе місце, оскільки безпосередньо пов’язане зі здатністю системи функціонувати в умовах, що наближені до екстремальних. На відміну від, наприклад, тестування юзабіліті, яке аналізує зручність інтерфейсу,

або тестування безпеки, що зосереджене на захисті даних, навантажувальне тестування моделює реальні сценарії високого трафіку, перевіряючи, як система реагує на одночасні запити тисяч користувачів, тривалі періоди активної експлуатації або раптові спайки навантаження.

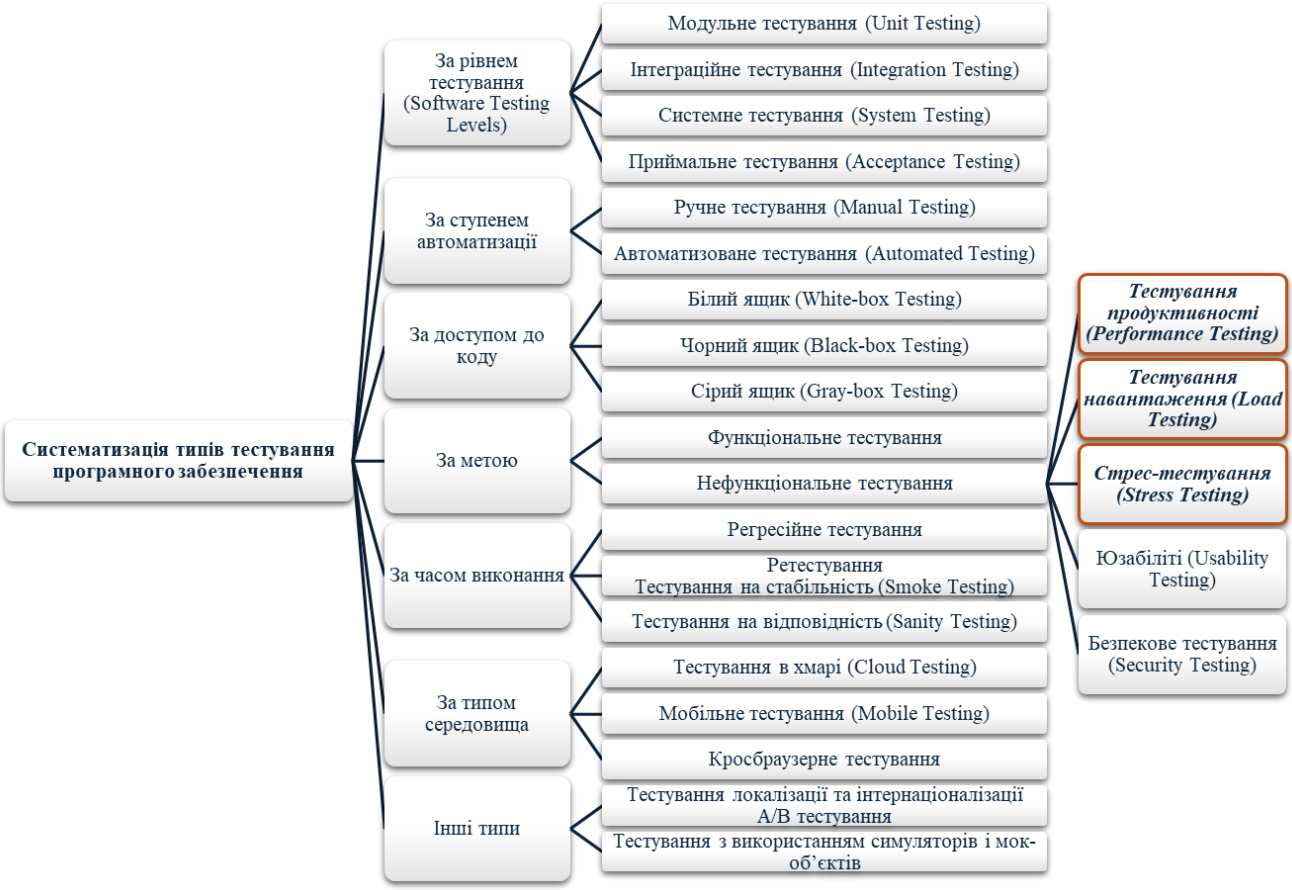


Рисунок 1.1 – Систематизація типів тестування

Навантажувальне тестування залишається критичним етапом життєвого циклу розробки програмного забезпечення, особливо в умовах зростання складності сучасних веб-додатків та розподілених систем [2]. Динамічні інтерфейси, побудовані на технологіях React, Angular та AJAX, вимагають високої адаптивності тестових сценаріїв, що значно ускладнює їх ручну підготовку [3]. Згідно з аналізом існуючих підходів, витрати часу на створення тестів для одного веб-додатку можуть перевищувати сотні годин, а часті зміни інтерфейсу змушують витрачати до 30% ресурсів на оновлення сценаріїв [13]. Це обмежує можливості всебічного тестування в умовах, близьких до реального

навантаження, що підвищує ризик виникнення критичних помилок після впровадження системи [1].

Інтеграція методів штучного інтелекту (ШІ) у процеси тестування пропонує перспективні рішення для подолання цих обмежень [4]. Дослідження демонструють, що використання згорткових нейронних мереж (CNN) дозволяє автоматизувати розпізнавання елементів інтерфейсу, знижуючи час створення тестових сценаріїв на 80% порівняно з традиційними методами [14]. Оптичне розпізнавання символів (OCR), інтегроване з інструментами на кшталт Apache JMeter, забезпечує точність ідентифікації текстових полів до 95%, що є ключовим для багатомовних платформ [12]. Ці технології не лише автоматизують процеси, але й підвищують адаптивність систем до змін у динамічних середовищах [15].

Актуальність запропонованого рішення підкріплюється зростанням потреби в тестуванні складних систем, таких як хмарні рішення та IoT-пристрої, де традиційні підходи недостатньо ефективні [7]. Наприклад, нейронні мережі, здатні моделювати поведінку мільйонів віртуальних користувачів, забезпечують реалістичність тестових сценаріїв та виявлення аномалій, які неможливо передбачити статичними методами [10].

Економічна доцільність впровадження ШІ у тестування також є суттєвим фактором. Автоматизація зменшує залежність від людських ресурсів, що критично для малих компаній та стартапів, які не мають можливості утримувати великі команди тестувальників [5]. Згідно з прогнозами, інтеграція AI-інструментів у тестування стане основним трендом у найближчі роки, забезпечуючи скорочення витрат та підвищення якості програмних продуктів [8].

Таким чином, розробка автоматизованої системи навантажувального тестування на основі методів ШІ є відповіддю на сучасні виклики індустрії. Вона сприяє не лише підвищенню надійності програмного забезпечення, але й прискоренню його виходу на ринок, що є ключовим у конкурентних умовах [6].

1.2 Огляд існуючих рішень та їх недоліків

Під час аналізу літератури було виявлено, що на думку [9], нейронні мережі (НМ) відіграють важливу роль у генерації реалістичних тестових даних та моделюванні поведінки користувачів.

Перш за все було проведено ознайомлення з тими проектами, які використовують НМ для розпізнання певних даних і зображень. Саме аналіз цих проектів допоміг виявити існуючі проблеми та напрямки подальшого розвитку обраного напрямку.

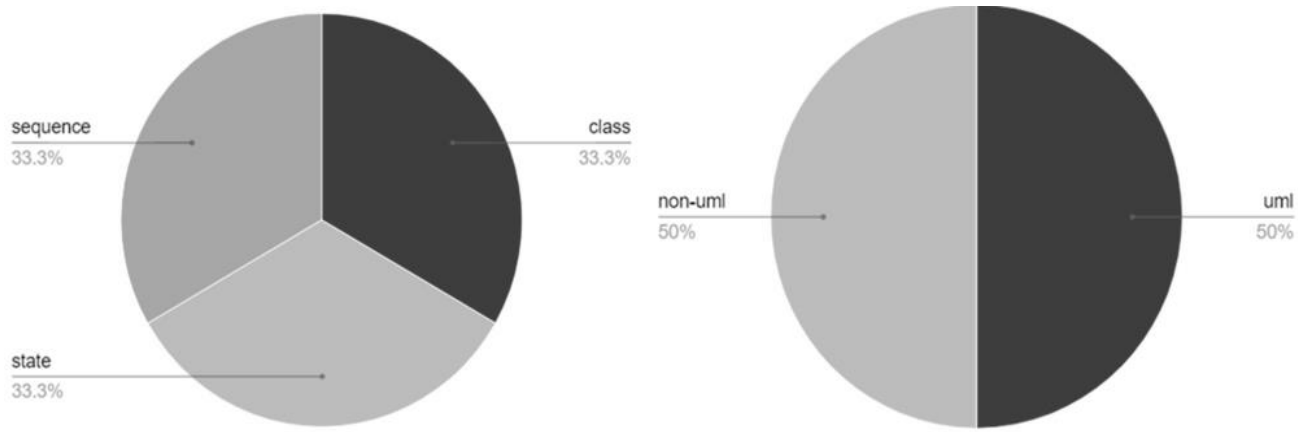
UIED: UI Element Detection (https://github.com/MulongXie/UIED) – модель на основі YOLOv3 для детекції елементів інтерфейсу (кнопки, текстові поля), яка дозволяє виявляти координати полів на скріншотах.

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" та "Password".

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

Проект 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 дозволяє створювати реалістичні дані на основі існуючих наборів, що може бути корисним для навантажувального тестування.



а) б)

Рисунок 1.2 – Розподіл класів даних згідно роботи Irina-Gabriela [8]:
а) розподіл типів даних , б) співвідношення джерел даних

Проект 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. Варто відмітити, що в TGAN гарно запропонована класифікація даних, як і в роботі Irina-Gabriela [10], де дані також були розбито на декілька класів і половину з них було використано через UML-діаграму (рисунок 1.2).

1.3 Мета та задачі дослідження

Метою кваліфікаційної роботи є розробка моделі системи автоматизованого навантажувального тестування програмних застосунків на основі методів машинного навчання для підвищення ефективності, точності та адаптивності процесу тестування.

Для досягнення поставленої мети мають бути вирішені наступні задачі:

- аналіз існуючих підходів до розпізнавання елементів web-сторінок;
- аналіз нейромережевих моделей для розпізнавання змісту реєстраційних форм;
- аналіз генеративних НМ моделей;

- створення моделі автоматизоаного навантажувального тестування;
- проведення емпіричного дослідження точності розпізнавання форм реєстрації;
- реалізація модулю автоматичної генерації навантажувальних тестів;
- аналіз отриманих результатів.

В кінцевому результаті даної роботи 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.

Об’єктом дослідження виступає навантажувальне тестування як критичний етап життєвого циклу розробки програмного забезпечення. Акцент робиться на веб-додатки, які функціонують у умовах високого навантаження, зберігаючи стабільність та продуктивність при одночасній роботі тисяч користувачів. Дослідження охоплює технологічні виклики, пов’язані з динамічними інтерфейсами, змінами архітектури систем та необхідністю оперативної адаптації тестових процесів до нових вимог.

Предметом дослідження є методи та алгоритми штучного інтелекту, інтегровані в процес навантажувального тестування. Зокрема, досліджуються механізми використання згорткових нейронних мереж (CNN) для розпізнавання графічних елементів веб-форм, оптичного розпізнавання символів (OCR) для ідентифікації текстових міток, а також інтеграція цих технологій з інструментом Apache JMeter. Особлива увага приділяється автоматизації генерації тестових даних, оптимізації взаємодії з динамічними інтерфейсами та підвищенню точності тестування в умовах реального навантаження.

Розроблена модель відкриває низку перспективних напрямів для подальших досліджень. Одним із пріоритетів є вдосконалення алгоритмів розпізнавання шляхом застосування трансформерних архітектур (наприклад, Vision Transformer), які можуть підвищити точність класифікації складних комбінованих форм за рахунок аналізу глобальних залежностей між елементами інтерфейсу.

Важливим напрямом є розширення функціоналу OCR-модуля для підтримки нових мов, таких як арабська в’язь, що дозволить адаптувати систему до глобальних ринків. Інтеграція мультимодальних моделей, здатних одночасно обробляти текст, зображення та метадані, може значно покращити адаптивність системи до різноманітних інтерфейсів.

Окремий інтерес становить адаптація моделі до тестування розподілених систем, таких як мікросервісні архітектури та IoT-пристрої. Інтеграція з хмарними платформами (наприклад, Kubernetes) та протоколами IoT (MQTT) дозволить створювати комплексні тестові середовища, що імітують реальні умови експлуатації.

На завершення, варто згадати про потенціал створення low-code платформи з візуальним інтерфейсом, яка дозволить користувачам без глибоких технічних знань налаштовувати тестові сценарії, аналізувати результати через інтерактивні дашборди та автоматично генерувати звіти. Такі інновації можуть стати основою для нового покоління інструментів тестування, орієнтованих на швидкість, гнучкість та доступність.

1.4 Очікувана новизна та практичне значення

Наукова новизна дослідження полягає в інтеграції методів штучного інтелекту, зокрема згорткових нейронних мереж (CNN) та оптичного розпізнавання символів (OCR), у процес автоматизованого навантажувального тестування програмних застосунків. Запропонований підхід вперше поєднує візуальне розпізнавання графічних елементів інтерфейсу (наприклад, форм логіну, реєстрації) з аналізом текстових міток для автоматичної генерації тестових сценаріїв у середовищі Apache JMeter. Новизна також проявляється у використанні глибокого навчання для класифікації динамічних веб-елементів, що змінюються в реальному часі (наприклад, у системах на базі React або Angular), що традиційні інструменти тестування не здатні ефективно обробляти.

Практичне значення роботи полягає у значному зниженні трудомісткості та часу, необхідного для проведення навантажувальних тестів. Завдяки автоматизації розпізнавання елементів інтерфейсу та генерації тестових даних час створення сценаріїв для однієї форми скорочується з 1–2 годин до 5–10 секунд. Це дозволяє тестувальникам зосередитися на аналізі результатів, а не на рутинній підготовці.

Система також підвищує точність тестування: CNN досягає до 92% точності у розпізнаванні складних реєстраційних форм, а OCR демонструє 88.9% ефективності для простих текстових полів. Такі показники значно перевищують можливості ручних методів, де точність залежить від людського фактору.

Ключові практичні переваги включають:

- адаптивність до змін інтерфейсу. Модель здатна автоматично оновлювати тестові сценарії при модифікації UI, що критично для Agile-середовищ, де оновлення відбуваються щотижня;

- підтримка динамічних веб-додатків. Алгоритми ефективно працюють з асинхронними елементами (AJAX, WebSocket), які складно тестувати традиційними методами;

- скорочення витрат. Автоматизація дозволяє компаніям зменшити витрати на оплату праці тестувальників та уникнути фінансових втрат від простоїв через помилки.

Розроблена модель може бути інтегрована у CI/CD-ланцюжки для безперервного тестування, що особливо актуально для фінансового сектору, електронної комерції та медичних систем, де стабільність і безпека є пріоритетами. Наприклад, у банківських додатках система зможе імітувати тисячі транзакцій одночасно, виявляючи вузькі місця до запуску в продуктив.

Перспективи впровадження охоплюють не лише IT-сектор, але й освітній процес: модель може стати основою для навчальних курсів з автоматизації тестування, демонструючи студентам реальні сценарії використання ШІ у розробці ПЗ. Таким чином, дослідження сприяє як технологічному прогресу, так і підготовці кваліфікованих фахівців.

2 ТЕОРЕТИЧНІ ОСНОВИ НАВАНТАЖУВАЛЬНОГО ТЕСТУВАННЯ ПРОГРАМНИХ ЗАСТОСУНКІВ

2.1 Сутність та завдання навантажувального тестування

За даними [1] навантажувальне тестування виступає критичним компонентом у забезпеченні стабільної роботи систем під час реальних умов експлуатації, особливо в епоху масштабних цифрових трансформацій.

Навантажувальне тестування(рисунок 2.1) є невід'ємною частиною розробки програмного забезпечення, особливо для веб-додатків [2] які повинні постійно підтримувати велику кількість одночасних користувачів без втрати продуктивності. //посилання на рис https://www.enterpriseitworld.com/what-is-stress-testing-how-it-works-main-purpose-and-examples/

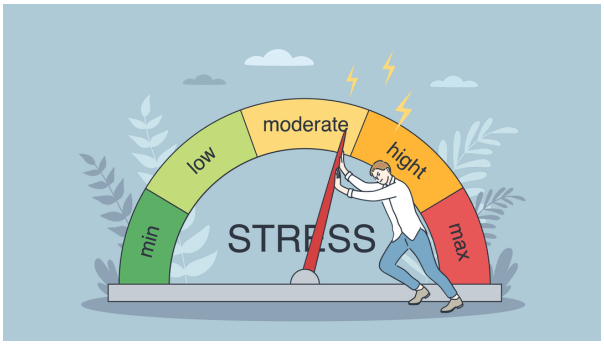


Рисунок 2.1 – Навантажувальне тестування

Основна мета навантажувального тестування – оцінити, як система реагує на збільшення навантаження, визначити її межі продуктивності та виявити вузькі місця, які можуть призвести до збоїв або погіршення роботи. Це допомагає розробникам і менеджерам проєктів краще зрозуміти, на якому рівні навантаження система починає давати збої, і вжити необхідних заходів для усунення проблем[16].

Завдання навантажувального тестування можна узагальнити наступним чином (таблиця 2.1).

Таблиця 2.1 – Завдання навантажувального тестування

Завдання	Складові
1	2
Визначення максимальної пропускної здатності системи	- визначення верхньої межі навантаження, при якому система все ще може функціонувати належним чином; - виявлення точки, де продуктивність системи різко знижується або система повністю відмовляє.
Аналіз поведінки системи під навантаженням	- спостереження за реакцією системи на зростаюче навантаження; - виявлення вузьких місць, де система показує ознаки нестабільності або зниження продуктивності; - визначення критичних точок навантаження, при яких система втрачає працездатність.
Оцінка стійкості системи до навантаження	- перевірка здатності системи витримувати різке збільшення навантаження без втрати функціональності; - визначення, чи може система відновитися після періоду високого навантаження.
Порівняння продуктивності	- оцінка продуктивності системи в різних умовах навантаження; - порівняння результатів тестування з попередніми версіями системи або конкуруючими рішеннями.
Виявлення вузьких місць і оптимізація	- визначення компонентів системи, які становлять «вузькі місця» під навантаженням; - надання рекомендацій щодо оптимізації системи для підвищення її продуктивності.

Продовження таблиці 2.1

11	2
Моделювання реальних сценаріїв використання	- створення тестових сценаріїв, що відображають типові або очікувані умови використання системи; - оцінка продуктивності системи в умовах, наближених до реального світу.

Навантажувальне тестування застосовується на різних етапах життєвого циклу розробки програмного забезпечення, від початкової розробки до розгортання та експлуатації. Воно допомагає виявити проблеми продуктивності на ранніх етапах, коли їх виправлення є більш ефективним і менш витратним.

Крім того, навантажувальне тестування може бути корисним для таких завдань, що визначені на рисунок. 2.2.

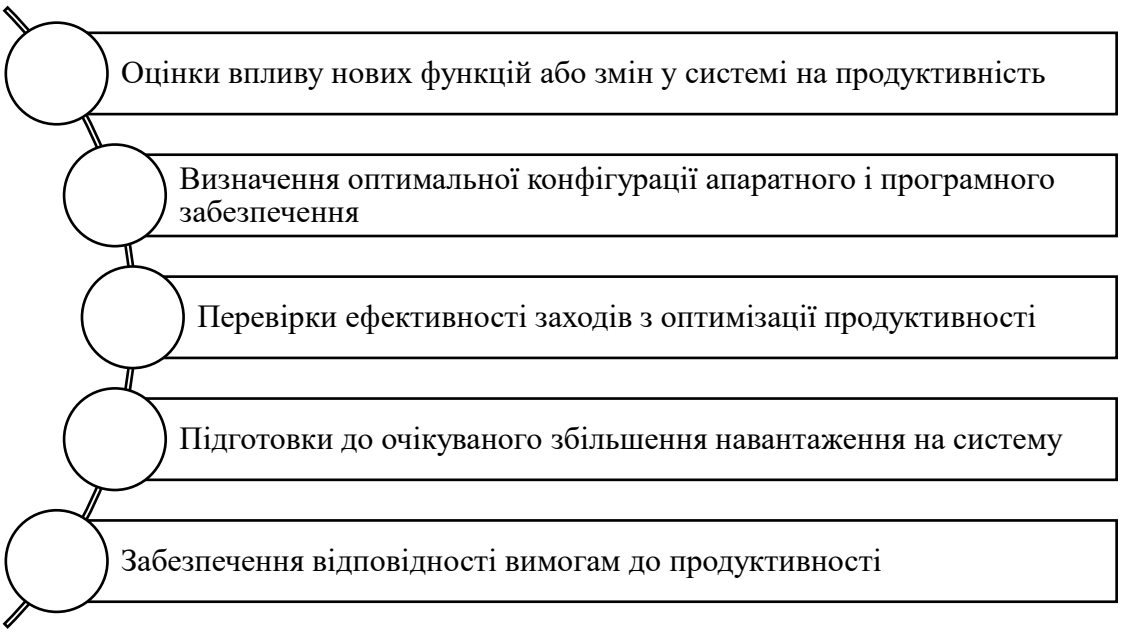


Рисунок 2.2 – Користь навантажувального тестування

Українські дослідники та фахівці в галузі програмної інженерії приділяють значну увагу навантажувальному тестуванню. Нижче наведено думки та напрацювання деяких українських авторів з цієї теми.

Зокрема, Олександр Коваль доцент кафедри програмної інженерії Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського", у своїй статті "Методи та інструменти навантажувального тестування програмного забезпечення" [17] зазначає, що навантажувальне тестування є критично важливим для оцінки продуктивності та стійкості програмних систем. Він наголошує на необхідності використання комплексного підходу, який включає в себе визначення тестових сценаріїв, вибір відповідних інструментів, моніторинг ключових показників продуктивності та аналіз отриманих результатів.

Олексій Гайдаржи у своїй роботі "Тестування продуктивності веб-застосунків: методи та інструменти" [18] детально розглядає особливості навантажувального тестування веб-додатків. Він підкреслює важливість врахування специфіки веб-технологій, таких як HTTP-протокол, масштабованість і розподіленість, при проєктуванні та виконанні навантажувальних тестів.

Ірина Лісовська у статті "Автоматизація навантажувального тестування програмного забезпечення" [19] розглядає переваги автоматизації навантажувального тестування, зокрема можливість виконання тестів у неробочий час, об'єктивність результатів та можливість їх повторення. Вона також наголошує на необхідності інтеграції навантажувального тестування в безперервний процес розробки програмного забезпечення.

Отже, українські фахівці приділяють значну увагу навантажувальному тестуванню, підкреслюючи його важливість для забезпечення якості та продуктивності програмних систем. Вони акцентують увагу на використанні комплексного підходу, врахуванні специфіки технологій, а також на необхідності автоматизації та інтеграції навантажувального тестування в процес розробки.

Таблиця 2.2 – Типи навантажувального тестування

Тип тестування	Опис	Приклад використання
Стрес-тестування	Перевірка системи за умов, що значно перевищують нормальне навантаження.	Імітація 10 000 користувачів при розрахунковій пропускній здатності на 5 000.
Тестування на витривалість	Тривале тестування для виявлення накопичувальних помилок (напр., витоків пам’яті).	48-годинне навантаження веб-сервера.
Спайк-тестування	Раптове збільшення навантаження для перевірки реакції системи.	Збільшення кількості запитів з 100 до 5 000 за 2 хвилини.
Об’ємне тестування	Перевірка обробки великих обсягів даних.	Імпорт CSV-файлу з 1 мільйоном рядків у систему.
Тестування масштабованості	Оцінка здатності системи збільшувати продуктивність при додаванні ресурсів.	Додавання серверів у кластер для покращення обробки запитів.

Саме навантажувальне тестування моє декілька типів, які застосовуються залежно від цілей та особливостей тестування (таблиця 2.2).

2.2 Методи та підходи до навантажувального тестування

Навантажувальне тестування є невід’ємною частиною життєвого циклу розробки програмного забезпечення, спрямованою на забезпечення стабільності системи під навантаженням. З плином часу методи тестування еволюціонували від ручних, ресурсомістких підходів до високотехнологічних автоматизованих рішень, і навіть до прототипів автоматизації з використанням штучного

інтелекту (ШІ). Ця еволюція відображає зростання складності сучасних систем, де динамічні інтерфейси, мікросервісні архітектури та високі вимоги до продуктивності вимагають інноваційних інструментів.

Таблиця 2.3 – Порівняння підходів до навантажувального тестування

Критерій	Мануальний підхід	Підхід на основі ШІ
Час створення одного тесту	Високий (1–2 години на форму залежно від складності)	Низький (5–10 секунд 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 (вимагає аналізу коду або скриптів)	Проста (модель розпізнає патерни автоматично)
Гнучкість форматів даних	Обмежена (тільки відомі тестувальнику формати)	Висока (модель адаптується до різних мов/форм)
Залежність від людини	Висока (весь процес залежить від тестувальника)	Низька (автономна робота після навчання)

За даними дослідження [3], понад 60% компаній стикаються із затримками випуску продуктів через неефективність тестування, що підкреслює актуальність автоматизації. А що вже можна казати якщо буде створене інноваційне рішення на основі ШІ. Далі наведемо порівняння мануального підходу та підходу на основі ШІ (таблиця 2.3).

2.2.1 Традиційні підходи (мануальне тестування)

На початках розвитку навантажувального тестування основним інструментом були ручні методи. Тестувальники імітували навантаження вручну, наприклад, відкриваючи десятки вкладок у браузері або використовуючи прості скрипти на мовах типу Bash чи Python. Ці методи вимагали значних трудовитрат і були неефективними для складних систем.

Наприклад у 2000-х роках для тестування веб-сайту команда могла використовувати скрипт, який відправляв HTTP-запити через curl, але кожен новий сценарій (наприклад, тестування форми оплати) потребував годин ручного налаштування параметрів. Зміна структури сторінки означала повне переписування коду.

Тож можемо зрозуміти що у такого підходу було багато мінусі, зокрема:

- низька масштабованість. Імітація більше 50 користувачів була технічно складною;
- відсутність стандартизації. Кожен тест створювався з нуля, що ускладнювало порівняння результатів;
- висока ймовірність помилок. Навіть незначна помилка у скрипті могла спотворити всі дані .

Ці обмеження стали каталізатором для розвитку автоматизованих інструментів, які зробили процес тестування більш структурованим.

2.2.2 Автоматизовані методи без використання штучного інтелекту

Автоматизовані методи навантажувального тестування стали ключовим кроком у еволюції перевірки продуктивності систем. Вони дозволяють імітувати поведінку тисяч користувачів, генерувати складні сценарії взаємодії та аналізувати результати за допомогою спеціалізованих інструментів. Наприклад, Apache JMeter відомий своєю гнучкістю та безкоштовним доступом, що робить його популярним серед розробників. Він дозволяє створювати тестові

плани через графічний інтерфейс, де кожен віртуальний користувач може відправляти запити, взаємодіяти з формами або навіть імітувати затримки між діями. JMeter підтримує різні протоколи, від HTTP до JDBC, і надає детальні звіти у вигляді графіків та таблиць, що спрощує виявлення «вузьких місць».

Однак навіть такі інструменти мають суттєві недоліки. Наприклад, JMeter не вміє адаптуватися до змін у структурі веб-інтерфейсу. Якщо форма логіну отримує нове поле або змінює ідентифікатори елементів, тестовий сценарій ламається, і його доводиться оновлювати вручну. Це особливо проблематично для динамічних додатків на базі React або Angular, де інтерфейси оновлюються часто. Крім того, JMeter не розуміє контекст сторінки: він може успішно відправити запит із некоректними даними, якщо тестувальник забув додати перевірку валідності відповіді.

Інші інструменти, такі як Gatling або LoadRunner, пропонують додаткові можливості, але супроводжуються своїми складнощами. Gatling, написаний на Scala, забезпечує високу продуктивність, але вимагає знань програмування, що ускладнює його використання для новачків. LoadRunner, з іншого боку, є потужним рішенням для корпоративних систем, але його вартість сягає десятків тисяч доларів, що робить його недоступним для малих компаній. Хмарні платформи на кшталт BlazeMeter спрощують процес, але обмежують безкоштовні користувачів у кількості запитів або тривалості тестів, змушуючи організації купувати дорогі ліцензії для масштабних проектів.

Головна слабкість цих методів – залежність від статичних сценаріїв. Будь-які зміни в API, інтерфейсі або бізнес-логіці вимагають ручного перегляду та корекції тестів. Це створює значне навантаження на команди, особливо в Agile-середовищах, де оновлення відбуваються щотижня. Крім того, багато інструментів вимагають технічної експертизи: розподілене тестування, робота з асинхронними запитами або інтеграція з CI/CD-ланцюжками часто вимагають глибокого розуміння інфраструктури.

Таким чином, автоматизовані методи без ШІ, хоч і покращують ефективність тестування, залишаються вразливими до динаміки сучасних

систем. Їхні обмеження підкреслюють потребу в інтелектуальних рішеннях, здатних адаптуватися до змін без постійного втручання людини.

2.2.3 Застосування штучного інтелекту в тестуванні

Незважаючи на прогрес, автоматизовані методи все ще потребують значної участі людини. Саме це обмеження стало причиною інтеграції штучного інтелекту в процес тестування. Наприклад, сучасні системи на основі комп'ютерного зору можуть автоматично адаптуватися до змін інтерфейсу, усуваючи необхідність постійного оновлення скриптів.

Ця адаптивність особливо корисна в контексті сучасних веб-додатків, де інтерфейси часто оновлюються для покращення користувацького досвіду. Алгоритми ШІ, такі як згорткові нейронні мережі (CNN), здатні ідентифікувати зміни в розміщенні елементів (наприклад, кнопок, форм) навіть без явного вказівки програміста, що робить процес тестування більш автономним.

У контексті життєвого циклу розробки програмного забезпечення (SDLC), тестування займає ключову позицію на кожному етапі, а згідно з дослідженням [3], штучний інтелект (ШІ) може бути ефективно інтегрований у етапи SDLC, що наведені в таблиці 2.4. Інтеграція штучного інтелекту SDLC забезпечує системний підхід до покращення якості та ефективності процесів. На кожному етапі, від планування до підтримки, ШІ допомагає автоматизувати рутинні задачі, зменшити ризики та підвищити точність виконання завдань.

Таблиця 2.4 – Застосування штучного інтелекту на етапах SDLC

Етап SDLC	Застосування ШІ	Очікувані результати
1	2	3
Планування та аналіз вимог	автоматизований аналіз вимог; прогнозування ризиків; оптимізація ресурсів.	підвищення точності оцінки проєкту; зменшення ризиків; ефективний розподіл ресурсів.

Продовження таблиці 2.4

1	2	3
Проектування	генерація архітектурних рішень; валідація проектних рішень; оптимізація структури.	прискорення проектування; покращення якості архітектури; зменшення технічного боргу.
Розробка	генерація тестових сценаріїв; передбачення дефектів; оптимізація коду.	швидша розробка; 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.

Це робить процес розробки більш структурованим і передбачуваним, зменшуючи ймовірність критичних помилок на пізніх етапах. Використання ШІ також сприяє оптимізації ресурсів, що є ключовим для досягнення високої продуктивності систем у реальних умовах експлуатації.

2.3 Огляд інструментів для навантажувального тестування

Сучасний ринок пропонує безліч інструментів для навантажувального тестування, кожен з яких має свої переваги та обмеження. Вибір інструменту залежить від специфіки проекту, бюджету та технічної експертизи команди. Розглянемо ключові рішення, які стали стандартами в індустрії.

2.3.1 Порівняльний аналіз інструментів навантажувального тестування

Серед застосунків, призначених для тестування продуктивності та навантаження програмного забезпечення [22], які забезпечують перевірку, як система працює при великій 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 у таблиці 2.5.

Таблиця 2.5 – Порівняння сервісів для навантажувального тестування

Інструмент	Opensource	Протоколи	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

Apache JMeter – один із найпопулярніших інструментів завдяки безкоштовності та відкритому коду. Він підтримує широкий спектр протоколів, від HTTP до SOAP, і дозволяє створювати тести через графічний інтерфейс або XML-скрипти. Наприклад, компанія-розробник фінтех-додатків може використовувати JMeter для імітації 10 000 користувачів, які одночасно роблять транзакції. Однак JMeter має труднощі з обробкою асинхронних запитів, таких як WebSocket, що критично для додатків реального часу.

Gatling, інший безкоштовний інструмент, відрізняється високою продуктивністю завдяки написанню скриптів на Scala. Він ідеально підходить для тестування API зі складними сценаріями, наприклад, для платформ потокового

відтворення відео. Але його синтаксис вимагає технічної підготовки, що обмежує аудиторію. Дослідження показало, що команди без досвіду роботи з Scala витрачають на 30% більше часу на налаштування тестів у порівнянні з JMeter.

LoadRunner від Micro Focus – професійне рішення для корпоративних клієнтів. Він підтримує майже всі технології, включаючи IoT та мобільні додатки, і пропонує потужні звіти для аналізу. Наприклад, великий банк може використовувати LoadRunner для тестування системи онлайн-платежів під навантаженням у 100 000 користувачів. Проте його вартість (від $10 000 на рік) робить його недоступним для стартапів.

K6 – сучасний інструмент з відкритим кодом, орієнтований на DevOps. Він інтегрується з Git і CI/CD-ланцюжками, що дозволяє автоматизувати тестування на етапі розробки. Наприклад, команда може запускати тести після кожного коміту в репозиторій, щоб виявити проблеми на ранніх стадіях. Однак K6 має обмежену підтримку графічних інтерфейсів, що ускладнює його використання для нетехнічних фахівців.

2.3.2 Переваги та недоліки Apache JMeter

Результати порівняльного аналізу показали переваги Apache JMeter [5] Сервіс є одним із самих стабільних та легких в використанні, надає послуги безкоштовно, підтримує автоматизацію за рахунок багатого функціоналу, дозволяє 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 та інше (рисунок 2.3).

Завдяки можливості моделювати різноманітні сценарії користувацьких запитів, JMeter є важливим інструментом для забезпечення надійності, стабільності та ефективності веб-сервісів та додатків перед їхнім запуском у виробництво. Це дозволяє розробникам та тестувальникам ідентифікувати та

усунути потенційні проблеми з продуктивністю, забезпечуючи кращий досвід користувача.

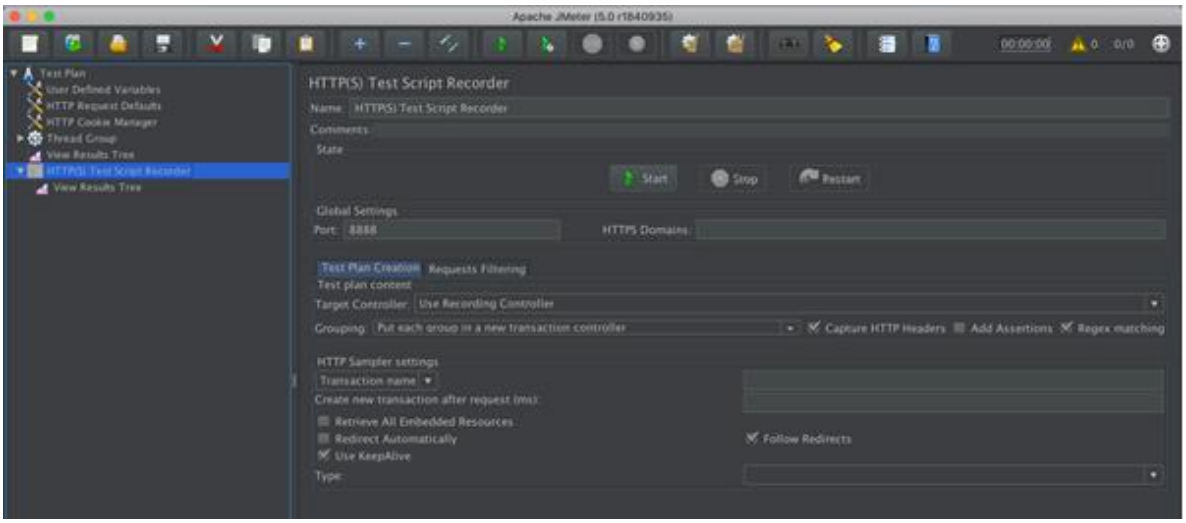


Рисунок 2.3 – Інтерфейс Apache JMeter

Проте навіть у такого чудового рішення є певні недоліки, зокрема:

- обмежена підтримка сучасних технологій. JMeter погано обробляє асинхронні запити (WebSocket, Server-Sent Events) та динамічні інтерфейси на React або Vue.js;

- висока ресурсомісткість. Для імітації десятків тисяч користувачів потрібні потужні сервери, що збільшує витрати на інфраструктуру;

- ручне оновлення тестів. Будь-які зміни в API або UI вимагають корекції скриптів, що сповільнює процес у швидкозмінних проектах.

2.4 Роль нейронних мереж в автоматизації тестування

Нейронні мережі стали справжньою революцією в автоматизації тестування програмного забезпечення, пропонуючи інноваційні рішення для підвищення якості та ефективності процесів. Їхня здатність аналізувати величезні обсяги даних і навчатися на них дозволяє автоматизувати складні завдання, які раніше потребували значного часу та людських ресурсів. Наприклад, нейронні мережі можуть самостійно розпізнавати елементи

інтерфейсу, створювати тестові сценарії або виявляти дефекти, які важко помітити традиційними методами. Це не лише прискорює процес тестування, але й робить його більш точним і надійним, що є критично важливим у сучасному світі, де програми оновлюються з шаленою швидкістю. Завдяки своїй адаптивності, нейронні мережі легко підлаштовуються до змін у дизайні чи функціоналі програм, що робить їх ідеальним інструментом для команд, які працюють за гнучкими методологіями, такими як Agile чи DevOps.

Окрім цього, використання нейронних мереж у тестуванні відкриває двері до проактивного підходу в забезпеченні якості. Вони здатні не просто виконувати задані сценарії, а й знаходити приховані закономірності чи аномалії в поведінці програми, передбачаючи потенційні проблеми ще до їх виникнення. Це значно знижує ризик випуску продукту з помилками та підвищує задоволеність користувачів. Водночас нейронні мережі оптимізують робочий процес, дозволяючи тестувальникам зосередитися на творчих і стратегічних задачах, а не на рутинній перевірці. Їхня висока точність і гнучкість роблять тестування не просто швидшим, але й більш інтелектуальним, що дає змогу створювати програмне забезпечення, яке відповідає найвищим стандартам якості та очікуванням ринку.

2.4.1 Типи нейронних мереж(RNN, LSTM, CNN)

RNN (рисунок 2.4) – це клас нейронних мереж, розроблений для обробки послідовних даних, де послідовність важлива, і має велику кількість застосувань у машинному навчанні, зокрема в обробці природної мови, аналізі часових рядів, генерації тексту, робототехніці та інших областях. Основна ідея полягає в тому, що RNN зберігає попередні стани і використовує їх, щоб робити прогнози або приймати рішення на основі поточного вхідного сигналу і попереднього стану.

Основними компонентами RNN є:

- вхід (Input): На кожному кроці часу RNN приймає вхідні дані (наприклад, вектори або послідовності слів) і оброблює їх;

- стани (States): Кожен крок часу RNN має свій внутрішній стан, який може зберігати інформацію з попередніх кроків. Цей стан допомагає зберігати контекст та залежності між даними в послідовності;

- ваги (Weights): RNN має набір ваг, які визначають, як вхідні дані і попередні стани взаємодіють між собою. Ці ваги оновлюються під час навчання мережі;

- вихід (Output): На кожному кроці часу RNN генерує вихідний сигнал або прогноз. Вихід може бути зв'язаним із завданням, наприклад, класифікацією тексту, або може бути використаний як вхід для інших мереж.

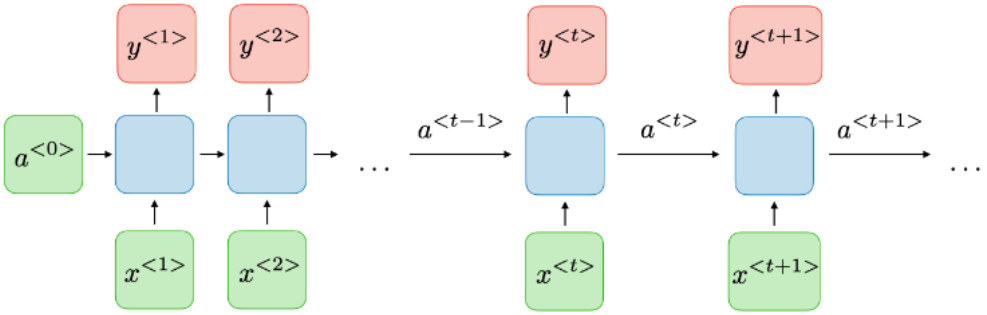


Рисунок 2.4 – Модель RNN

Проте, звичайні RNN мають деякі обмеження. Один з найважливіших недоліків – це проблеми з довгостроковими залежностями і градієнтними проблемами. У великих послідовностях із довгими залежностями, градієнти можуть стати дуже малими або дуже великими, що робить процес навчання нестабільним. Ці проблеми призводять до того, що RNN не завжди здатні добре моделювати складні послідовності.

Для подолання цих обмежень були розроблені більш потужні рекурентні мережі, такі як LSTM (Long Short-Term Memory) і GRU (Gated Recurrent Unit). Вони використовують спеціальні механізми, які дозволяють контролювати потік інформації та зберігати важливу інформацію в станах. LSTM має додаткові "ворота", які контролюють потік інформації, і це робить його ефективнішим для роботи з довгостроковими залежностями.

LTSM(Long Short-Term Memory) – це тип рекурентної нейронної мережі (рисунок 2.5), який був спеціально розроблений для обробки та прогнозування послідовних даних, наприклад, часових рядів. Головною особливістю LSTM є наявність осередку пам'яті, який зберігає інформацію на протязі всієї послідовності даних. Ця довготривала пам'ять підтримується за допомогою спеціального механізму, який дозволяє додавати та видаляти інформацію з комірки.

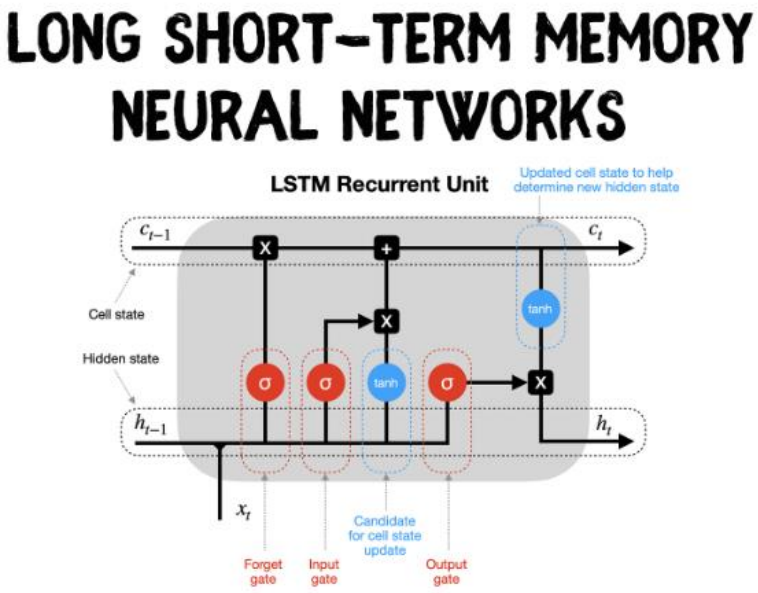


Рисунок 2.5 – Схема LSTM

Ворота в LSTM – це механізми, що контролюють збереження інформації у стані комірки. Вони складаються з ваг, які визначають вплив вхідної інформації на стан клітини, та сигмоїдних функцій, що визначають, яка частина вхідних даних має бути збережена.

Існують три типи воріт у LSTM:

- ворота входу (input gate), які визначають, яку нову інформацію потрібно додати у стан комірки;
- ворота забування (forget gate), що визначають, яку інформацію зі стану комірки потрібно видалити;
- ворота виходу (output gate), які визначають, яка інформація зі стану комірки має бути використана для формування вихідного сигналу.

Кожні з цих воріт мають власні вагові матриці та вектори зсуву, які визначають їх дію на кожному кроці часу. Стан комірки в LSTM представляє прихований стан мережі, який зберігає інформацію про послідовність. Він обчислюється на основі поточних вхідних даних, стану комірки на попередньому кроці, значень воріт забування та входу, а також вагових матриць та векторів зсуву для стану комірки.

LSTM здатна ефективно обробляти послідовності даних, зберігаючи важливу інформацію про минуле і відкидаючи непотрібну. В TensorFlow LSTM може бути реалізована за допомогою LSTM шару, який можна інтегрувати в послідовні моделі.

CNN (рисунок 2.6) – Згорткові нейронні мережі представляють собою один з фундаментальних елементів у сфері глибинного навчання та штучного інтелекту, особливо коли мова йде про обробку зображень та візуальних даних. Це високоспеціалізована форма нейронної мережі, яка використовує унікальний процес, відомий як згортка, для ефективного аналізу даних.

У серці CNN лежить концепція імітації зорового сприйняття людини. Людське око не сприймає всю картину цілком; натомість, воно фокусується на певних ключових елементах для розпізнавання об'єктів. Аналогічним чином, згорткові нейронні мережі фокусуються на специфічних ознаках зображення, використовуючи для цього фільтри або ядра.

Фільтри в згорткових шарах - це невеликі вагові матриці, які пересуваються по входу (наприклад, зображенні), виконуючи операцію згортки. Під час цього процесу фільтр множить свої значення на значення відповідних пікселів зображення та сумує результати. Це дозволяє виявити певні ознаки, такі як краї, кути, кольори або текстури, які є критичними для розуміння та аналізу зображень.

Після проходження через згорткові шари, дані проходять через функцію активації, таку як ReLU. Ця функція додає нелінійність до мережі, що є важливим, оскільки більшість реальних даних є нелінійними.

Пулінгові шари в CNN відіграють роль у зменшенні просторових розмірів ознакової карти, що отримана після згортки. Це допомагає зменшити обсяг обчислень та засобів пам'яті, необхідних для навчання мережі, а також допомагає уникнути перенавчання. Наприклад, MaxPooling вибирає максимальне значення з кожної області вхідних даних.

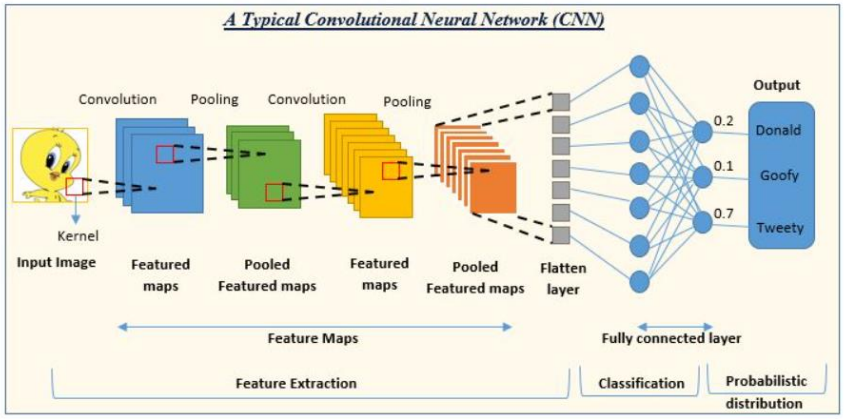


Рисунок 2.6 – Схема CNN

Після згорткових та пулінгових шарів, дані зазвичай потрапляють до повнозв'язаних шарів. Тут, усі ознаки, які були виділені та зменшені попередніми шарами, об'єднуються для прийняття кінцевого рішення, наприклад, для класифікації зображень.

CNN є вельми ефективними в розпізнаванні та класифікації об'єктів на зображеннях, виявленні облич та ідентифікації об'єктів на відео. Їх здатність адаптуватися до різних типів даних та вчитися на основі прикладів робить їх незамінними у сучасному світі штучного інтелекту та комп'ютерного зору.

2.4.2 Використання CNN для розпізнання форм

Згорткові нейронні мережі (CNN) відіграють ключову роль у розпізнаванні форм у процесі автоматизованого тестування, особливо коли потрібно працювати з візуальними даними, такими як зображення веб-інтерфейсів. Процес їх застосування починається з навчання моделі на великому наборі даних, що включає зображення різних форм – наприклад, форм логіну, реєстрації

чи оплати. Під час цього етапу CNN вчиться виділяти важливі візуальні особливості, такі як розташування кнопок, текстових полів чи інших елементів, і класифікувати форми за їхнім типом. Після завершення навчання модель інтегрується в тестову систему, де вона автоматично аналізує нові зображення та визначає, до якого типу належить форма, що дозволяє генерувати відповідні тестові сценарії без ручного втручання.

CNN ідеально підходять для цього завдання завдяки своїй унікальній архітектурі, яка імітує роботу людського зору. Їхні згорткові шари здатні виявляти локальні деталі, такі як контури чи текстури, а наступні шари об'єднують ці деталі в цілісні патерни, що робить їх надзвичайно ефективними для обробки зображень. На відміну від інших типів нейронних мереж, CNN враховують просторову структуру даних, що є критично важливим для розпізнавання форм із різноманітними дизайнами чи нестандартними елементами. Саме тому вони перевершують традиційні методи, які часто потребують ручного налаштування для кожного нового інтерфейсу, забезпечуючи більшу гнучкість і точність.

Цей підхід набуває все більшого поширення в автоматизованому тестуванні, особливо з ростом популярності штучного інтелекту та комп'ютерного зору. Сучасні системи дедалі частіше використовують CNN для візуального тестування веб-застосунків, що дозволяє скоротити час на створення й оновлення тестів, а також підвищити їхню надійність. Дослідження показують, що CNN досягають високої точності навіть у складних сценаріях, таких як розпізнавання форм із динамічними чи нестандартними інтерфейсами. Таким чином, застосування CNN не лише спрощує тестування, але й робить його більш інтелектуальним і адаптивним до потреб сучасних проєктів.

2.4.3 Використання OCR для аналізу текстових міток

Оптичне розпізнавання символів (OCR) – це технологія, яка перетворює текст на зображеннях у формат, який може бути оброблений комп'ютером. OCR

працює в кілька етапів: спочатку зображення обробляється для підвищення чіткості, потім алгоритми виділяють текстові області, сегментують їх на символи чи слова, і, нарешті, розпізнають ці символи, використовуючи шаблони або машинне навчання. Цей метод потрібен для автоматичного витягування текстової інформації з візуальних джерел, наприклад, для читання міток на веб-формах чи документів. У тестуванні OCR допомагає швидко ідентифікувати ключові слова, такі як "Login" чи "Submit", що дозволяє системі зрозуміти призначення форми та підготувати відповідні тестові дані.

OCR вважається крутим варіантом завдяки своїй простоті, швидкості та широкій застосовності. Він не вимагає складного навчання, як нейронні мережі, і легко інтегрується в існуючі тестові інструменти, що робить його доступним навіть для невеликих команд. Крім того, OCR підтримує різні шрифти й мови, що забезпечує його гнучкість у роботі з різноманітними інтерфейсами. Усе це дозволяє економити час на ручному аналізі та підвищувати ефективність автоматизованих процесів, роблячи OCR незамінним помічником у тестуванні сучасних веб-застосунків.

3 РОЗРОБКА ТА РЕАЛІЗАЦІЯ МОДЕЛІ СИСТЕМИ АВТОМАТИЗОВАНОГО НАВАНТАЖУВАЛЬНОГО ТЕСТУВАННЯ

3.1 Архітектура запропонованої моделі. Призначення основних функціональних модулів

Розроблена система для автоматизованого навантажувального тестування програмних застосунків є цілісним та комплексним рішенням, яке об'єднує передові методи штучного інтелекту з традиційними інструментами тестування. Її архітектура побудована як набір взаємопов’язаних модулів, кожен з яких відіграє свою унікальну роль у процесі автоматизації. Дані модулі охоплюють усі етапи – від підготовки даних до аналізу результатів тестування, створюючи замкнений цикл роботи системи. Основна ідея полягає в тому, щоб зменшити ручну працю, підвищити точність і пришвидшити процес тестування за допомогою інтелектуальних технологій штучного інтелекту.

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. [21]

Під підготовкою даних маємо на увазі збір та обробку зображень форм для подальшого використання. Навчання моделі передбачає тренування штучного інтелекту (CNN) для розпізнавання форм. Етап розпізнавання форм служить для ідентифікація полів і типів форм за допомогою навченої моделі та OCR.

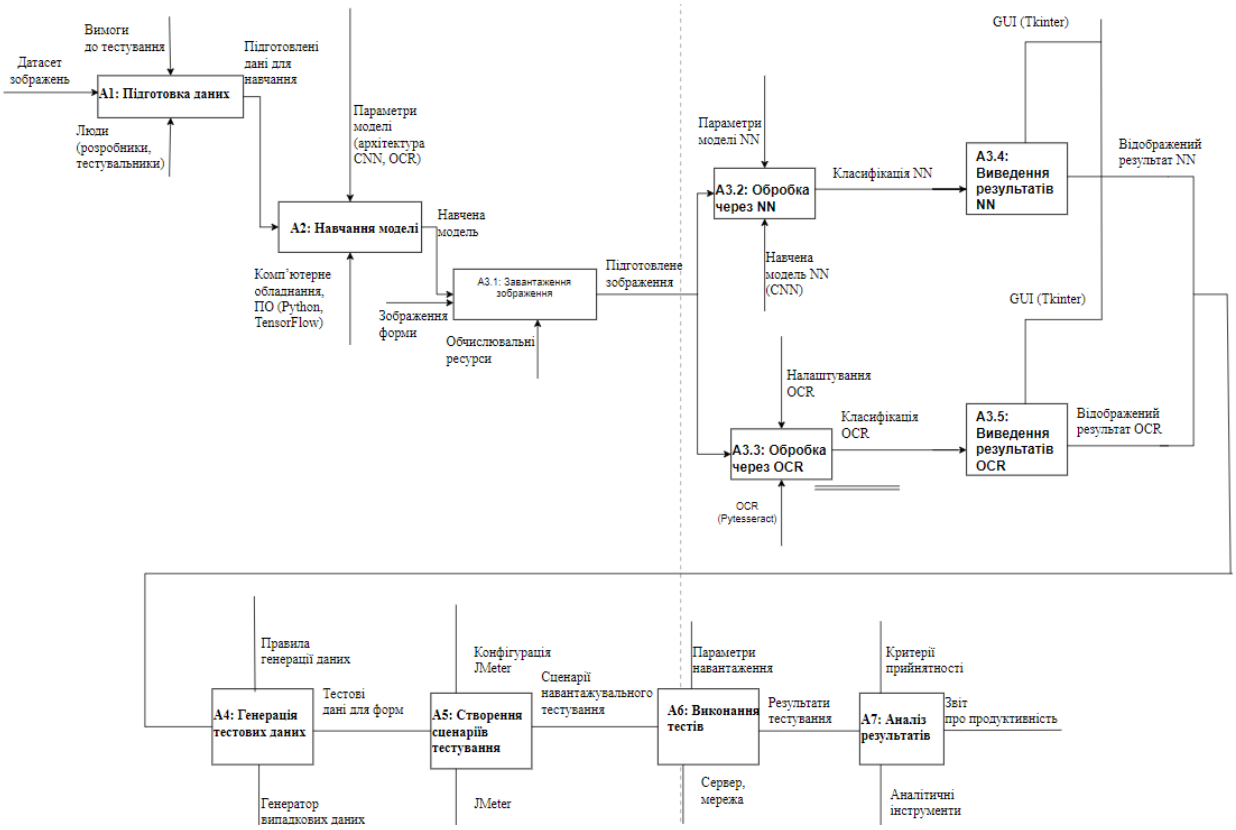


Рисунок 3.1 – Модель системи автоматизованого навантажувального тестування

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, яке передається на блоки аналізу на основі CNN або OCR – систему.

3.1.1 Модуль підготовки даних

Модуль підготовки даних є важлимим компонентом системи, який забезпечує якісну підготовку інформації для подальшого навчання та роботи нейронних мереж. Цей модуль відповідає за перетворення сирих даних у формат, оптимальний для навчання та роботи систем, виконує підзадачі:

- попередня обробка зображень;
- підготовка тренувальних даних;

- оптимізація та інтеграція;
- моніторинг.

Процес підготовки даних починається з попередньої обробки зображень, а саме що всі вхідні зображення форм проходять через серію трансформацій, які забезпечують їх стандартизацію та оптимізацію для подальшої обробки. Зображення приводяться до стандартного розміру 224x224 пікселів, що забезпечує уніфікований формат для всіх вхідних даних. Для зменшення впливу кольорових артефактів та покращення чіткості текстових елементів, виконується перетворення кольорових зображень у відтінки сірого з подальшою адаптивною корекцією контрастності. Важливим етапом є виділення регіонів інтересу на зображенні, для того щоб показати детально користувачу яка саме область допомогла нейромережі розпізнати зображення. Для цього використовується алгоритм Canny, який ефективно виявляє границі форм. Після виявлення границь застосовується адаптивний поріг для розділення фону та об'єктів, а також морфологічні операції для видалення шуму та артефактів. Цей процес забезпечує чітке виділення важливих елементів форми та підготовку даних для подальшого аналізу.

Формування тренувального набору даних є критично важливим етапом підготовки. Система реалізує механізм збалансування кількості зразків для кожного класу форм, що забезпечує рівномірне навчання моделі на різних типах даних. Для розширення тренувального набору та покращення узагальнюючих властивостей моделі, застосовується комплексна аугментація даних. Цей процес включає поворот зображень на випадковий кут, зміну яскравості та контрастності, додавання шуму та масштабування. Такі трансформації дозволяють моделі краще адаптуватися до різних варіацій вхідних даних.Валідація даних є невід'ємною частиною процесу підготовки. Система автоматично перевіряє якість зображень, виявляє та видаляє пошкоджені зразки, а також валідує правильність класифікаційних міток. Це забезпечує високу якість тренувального набору та надійність навчання розробленої моделі.

Для забезпечення ефективної роботи системи, реалізовано механізми оптимізації та кешування даних. Попередньо оброблені зображення зберігаються в кеші, що дозволяє забезпечити швидкий доступ до підготовлених даних та ефективне використання системних ресурсів. Система використовує паралельну обробку та адаптивне масштабування для оптимізації продуктивності.Модуль підготовки даних тісно інтегрується з іншими компонентами системи, забезпечуючи оптимальний формат даних для CNN та OCR модулів. Для CNN підготовлюються дані у форматі, оптимальному для навчання, організовуються у батчі та формуються валідаційні набори. Для OCR модуля підготовлюються текстові регіони, формується словник ключових слів та зберігається контекстна інформація.

Процес підготовки даних супроводжується постійним моніторингом. Система відстежує якість підготовлених даних, збирає статистику по обробці та веде детальне логування процесів. Це дозволяє виявляти проблеми на ранніх етапах та вносити необхідні корективи.Аналіз продуктивності та адаптивне налаштування параметрів забезпечують оптимальну роботу модуля підготовки даних. Система автоматично виявляє аномалії та корегує параметри обробки для забезпечення високої якості підготовлених даних.

3.1.2 Модуль навчання моделі

Модуль навчання моделі є центральним компонентом системи, який відповідає за тренування та оптимізацію нейронної мережі для класифікації форм. Цей модуль реалізує комплексний підхід до навчання, що дозволяє досягти високої точності розпізнавання та узагальнення.

Процес навчання моделі побудований на основі згорткової нейронної мережі (CNN), яка спеціально оптимізована для роботи з формами. Архітектура мережі включає кілька згорткових шарів, кожен з яких відповідає за виявлення певного рівня особливостей. Початкові шари зосереджені на виявленні базових візуальних елементів, таких як лінії та кути, тоді як глибші шари

поєднують ці елементи в більш складні структури, характерні для різних типів форм. Для запобігання перенавчанню та покращення узагальнюючих властивостей моделі, архітектура включає шари dropout та batch normalization. Ці механізми допомагають стабілізувати процес навчання та зменшити залежність моделі від конкретних особливостей тренувального набору.

Навчання моделі відбувається в кілька етапів, кожен з яких спрямований на оптимізацію певних аспектів роботи мережі. На початковому етапі виконується попереднє навчання на великому наборі даних, що дозволяє моделі вивчити загальні особливості форм. Після цього відбувається тонке налаштування на специфічному наборі даних, що дозволяє адаптувати модель до конкретних типів форм. Процес навчання супроводжується постійним моніторингом продуктивності моделі. Система відстежує такі метрики, як точність класифікації, втрати на тренувальному та валідаційному наборах, а також час навчання і багато іншних. Ці метрики використовуються для адаптивного налаштування параметрів навчання, таких як швидкість навчання та розмір батчу.

Для забезпечення стабільного навчання та уникнення перенавчання, система використовує комплексний підхід до оптимізації та регуляризації. Застосовується адаптивна швидкість навчання, яка автоматично коригується в залежності від прогресу навчання. Це дозволяє моделі ефективно навчатися на різних етапах тренування.Регуляризація реалізується через комбінацію різних технік, включаючи L1 та L2 регуляризацію, dropout та early stopping. Ці механізми допомагають контролювати складність моделі та запобігати перенавчанню. Система автоматично визначає оптимальні параметри регуляризації на основі продуктивності моделі на валідаційному наборі.

Важливою особливістю модуля є підтримка інтерактивного навчання. Система дозволяє користувачу втручатися в процес навчання, вносити корективи та надавати додаткову інформацію. Це реалізується через механізм зворотного зв'язку, який дозволяє моделі адаптуватися до користувацьких виправлень.

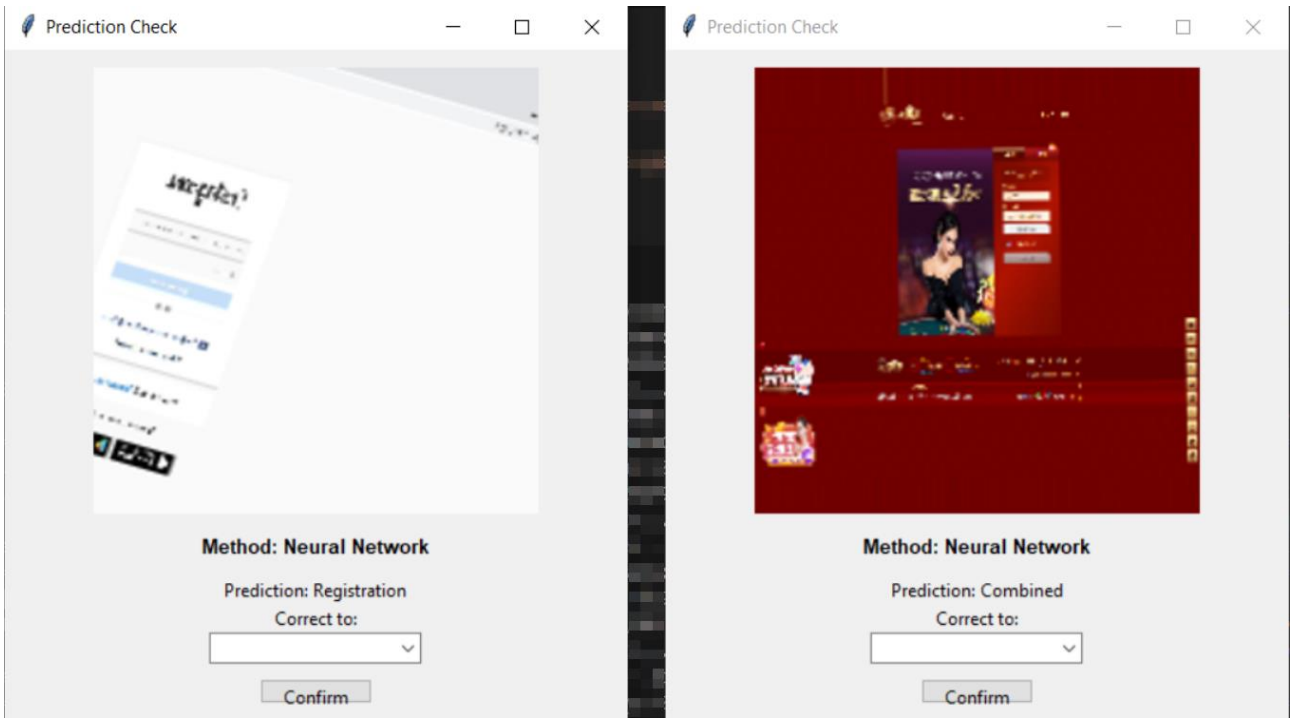


Рисунок 3.2 – Приклад інтерактивного навчання запропонованої моделі

Інтерактивне навчання включає можливість коригування класифікаційних міток, додавання нових прикладів та модифікації параметрів навчання. Система зберігає історію навчання та враховує користувацькі виправлення для покращення точності розпізнавання.

Процес навчання супроводжується постійною валідацією та тестуванням моделі. Система використовує окремий валідаційний набір для оцінки продуктивності моделі під час навчання. Це дозволяє виявляти проблеми перенавчання та вчасно вносити корективи.Тестування моделі виконується на незалежному тестовому наборі, що дозволяє оцінити реальну продуктивність системи. Система збирає детальну статистику по результатах тестування, включаючи матрицю помилок, точність для кожного класу та загальні метрики продуктивності.

Модуль навчання включає механізми збереження та відновлення стану моделі. Система автоматично зберігає найкращі версії моделі на основі продуктивності на валідаційному наборі. Це дозволяє відновити оптимальний стан моделі в разі необхідності. Збереження включає не тільки ваги

моделі, але й інформацію про архітектуру, параметри навчання та історію тренування. Це забезпечує повне відновлення стану моделі та можливість продовження навчання з оптимальної точки.

3.1.3 Модуль розпізнавання форм

Модуль розпізнавання форм є компонентом системи, який відповідає за аналіз та класифікацію вхідних зображень форм. Цей модуль поєднує в собі технології комп'ютерного зору та машинного навчання для забезпечення високої точності розпізнавання, виконує задачі:

- розпізнавання;
- інтеграція з OCR;
- оцінка впевненості;
- контекстний аналіз;
- адаптивне розпізнавання;
- інтеграція з іншими модулями.

Процес розпізнавання форм починається з аналізу вхідного зображення. Система виконує попередню обробку зображення, включаючи нормалізацію, вирівнювання та підготовку до подальшого аналізу. Особливу увагу приділяється виявленню границь форми та сегментації важливих елементів. Це досягається за допомогою комбінації алгоритмів комп'ютерного зору, таких як детекція границь Canny та адаптивна сегментація.Після попередньої обробки, зображення передається в згорткову нейронну мережу для виявлення характерних особливостей. Мережа аналізує різні рівні особливостей, від базових візуальних елементів до складних структур, характерних для різних типів форм. Цей багаторівневий аналіз дозволяє системі ефективно розпізнавати різні типи форм, навіть при наявності варіацій у їх представленні.

Важливою частиною модуля розпізнавання є інтеграція з системою оптичного розпізнавання символів (OCR). Ця інтеграція дозволяє системі

аналізувати не тільки візуальну структуру форми, але й її текстовий вміст. OCR модуль виконує розпізнавання тексту в виділених регіонах форми, що дозволяє отримати додатковий контекст для класифікації. Система використовує розумний підхід до виділення регіонів для OCR аналізу. Замість простого розділення форми на рівні частини, система аналізує структуру форми та визначає регіони, які найімовірніше містять важливу текстову інформацію. Це досягається за допомогою аналізу просторового розташування елементів та виявлення характерних патернів у структурі форми.

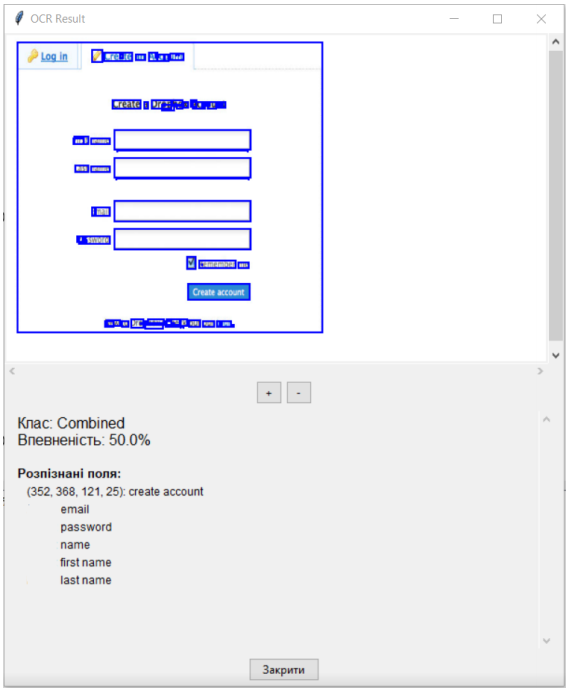


Рисунок 3.3 – Приклад розпізнанної форми за допомогою OCR

Важливим аспектом модуля є оцінка впевненості в результатах розпізнавання. Система не тільки визначає клас форми, але й оцінює рівень впевненості в цьому рішенні. Це досягається за допомогою аналізу виходів нейронної мережі та додаткових метрик якості. Оцінка впевненості використовується для фільтрації результатів та виявлення випадків, які потребують додаткового аналізу або втручання користувача. Система може автоматично відхиляти результати з низькою впевненістю або запропонувати альтернативні варіанти класифікації.

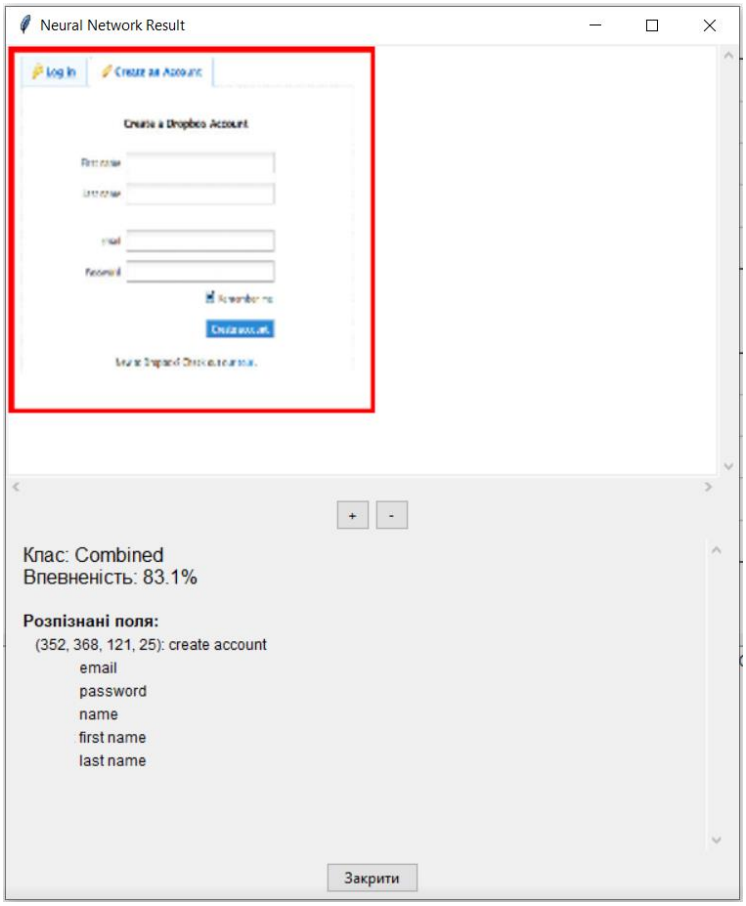


Рисунок 3.4 – Приклад розпізнаної форми з процентом впевненості

Модуль розпізнавання форм реалізує контекстний аналіз, який дозволяє враховувати взаємозв'язки між різними елементами форми. Система аналізує не тільки окремі елементи, але й їх взаємне розташування та відносини. Це дозволяє покращити точність розпізнавання, особливо в складних випадках, коли окремі елементи можуть бути неоднозначними.Контекстний аналіз включає виявлення логічних зв'язків між полями форми, аналіз послідовності елементів та визначення ієрархії інформації. Система використовує цю інформацію для уточнення результатів розпізнавання та покращення загальної точності класифікації.

Модуль реалізує адаптивне розпізнавання, яке дозволяє системі покращувати свою продуктивність на основі досвіду. Система зберігає історію розпізнавання та використовує цю інформацію для оптимізації процесу аналізу. Це включає адаптацію параметрів розпізнавання, оновлення моделі класифікації та покращення алгоритмів обробки.Адаптивне розпізнавання

особливо ефективне при роботі з новими типами форм або варіаціями існуючих форм. Система здатна виявляти нові патерни та адаптуватися до змін у структурі форм, що забезпечує стабільну продуктивність навіть при появі нових типів даних.

Модуль розпізнавання форм тісно інтегрується з іншими компонентами системи. Він взаємодіє з модулем навчання для оновлення моделі на основі результатів розпізнавання, з модулем підготовки даних для оптимізації процесу обробки, та з системою моніторингу для відстеження продуктивності.Ця інтеграція забезпечує ефективну роботу всієї системи та дозволяє постійно покращувати якість розпізнавання. Система використовує інформацію з різних модулів для оптимізації процесу розпізнавання та забезпечення високої точності класифікації форм.Модуль розпізнавання форм є складним компонентом системи, який поєднує в собі різні технології та підходи для забезпечення високої точності розпізнавання. Реалізовані механізми та алгоритми дозволяють системі ефективно аналізувати різні типи форм та забезпечувати надійну класифікацію навіть у складних випадках.

3.1.4 Модуль генерації тестових даних

Модуль генерації тестових даних є важливим компонентом системи, який відповідає за створення синтетичних даних для відповідних полей на розпізнанних формах. Цей модуль використовує передові технології генеративних моделей для створення реалістичних тестових наборів.

Основним інструментом модуля є генеративна змагальна мережа (GAN), яка спеціально навчена для створення тестових данних для форм. GAN складається з двох взаємодіючих компонентів: генератора та дискримінатора. Генератор створює нові зразки форм, намагаючись відтворити характерні особливості реальних даних, тоді як дискримінатор навчається розрізняти справжні та згенеровані дані і оцінює їх якість. Особливістю реалізації GAN є використання умовної генерації, яка дозволяє створювати дані з певними

критеріями. Це досягається завдяки введенню додаткових умовних параметрів, що дозволяють контролювати процес генерації та створювати дані з бажаними властивостями. Такий підхід забезпечує створення різноманітних тестових наборів, які охоплюють різні варіації полей.

Процес генерації тестових даних починається з аналізу реальних даних та виявлення їх характерних особливостей. Система використовує цю інформацію для налаштування параметрів генерації та забезпечення реалістичності згенерованих даних. Генератор створює нові зразки, враховуючи структуру полей, розташування елементів та їх взаємозв'язки.

Альтернативний метод генерації даних реалізований через систему псевдогенерації, яка використовує статистичний аналіз та заздалегідь визначені шаблони. Цей підхід особливо ефективний у випадках, коли потрібно швидко створити набір тестових даних з чітко визначеними параметрами. Система враховує специфіку різних полів форм та дозволяє гнучко налаштовувати параметри генерації відповідно до потреб тестування.

3.1.5 Графічний інтерфейс і моніторинг для GAN

Особлива увага в розробці модуля була приділена створенню інтуїтивно зрозумілого інтерфейсу користувача. Графічний інтерфейс надає можливість легко перемикатися між різними методами генерації, налаштовувати параметри та спостерігати за процесом в реальному часі [12].

Система візуалізації допомагає користувачам краще розуміти якість згенерованих даних та ефективність різних підходів до генерації.

Система управління згенерованими даними забезпечує гнучкість у форматах збереження та експорту. Інтеграція з Google Drive розширює можливості доступу до даних та їх спільного використання. Автоматична синхронізація та система версіонування гарантують збереження всіх важливих наборів даних та можливість відстеження їх змін.

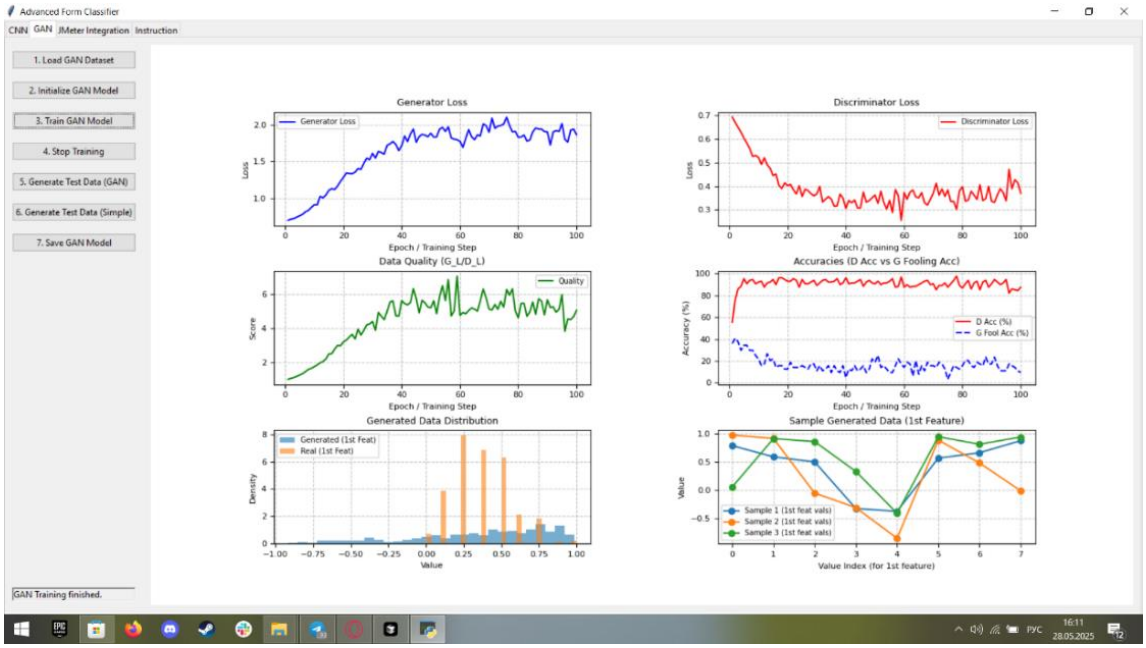


Рисунок 3.5 – Графічний інтерфейс і моніторинг метрик для GAN

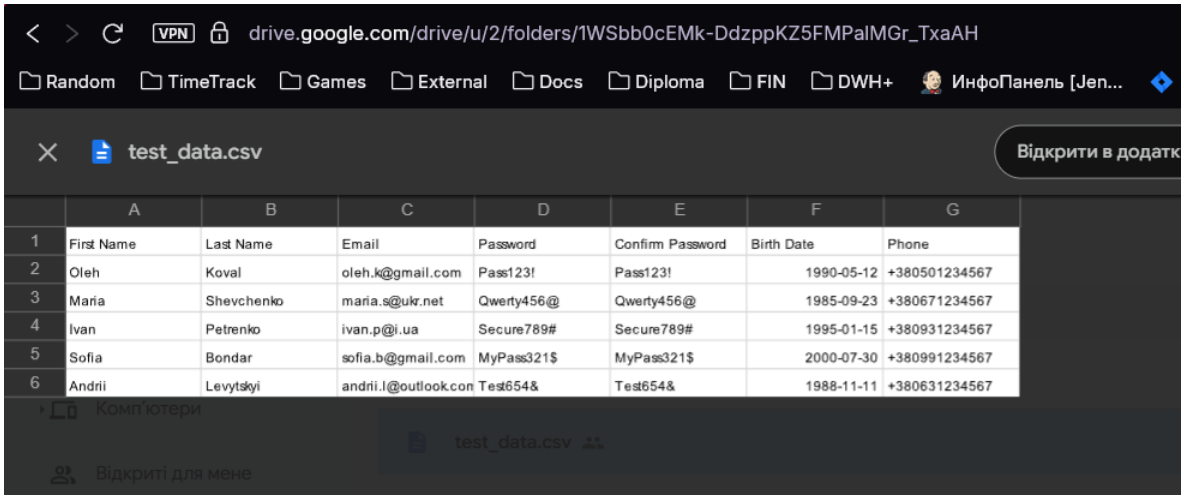


Рисунок 3.6 – Приклад згенерованих тестових данних, які були завантажені на Google Drive

Такий комплексний підхід до генерації тестових даних забезпечує високу якість та різноманітність тестових наборів, що є критичним фактором для ефективного тестування форм різного рівня складності. Модуль демонструє високу гнучкість та адаптивність, що робить його потужним інструментом у процесі тестування та валідації форм.

3.1.6 Модуль інтеграції з Apache JMeter та аналізу даних

У розробленій системі реалізовано модуль інтеграції з Apache JMeter, який забезпечує навантажувальне тестування веб-форм. Модуль представляє собою комплексне рішення, що поєднує в собі можливості реального тестування через Apache JMeter та внутрішньої симуляції навантаження, що робить його гнучким інструментом для різних сценаріїв використання. Система реалізує комплексний підхід до тестування продуктивності веб-форм через інтуїтивно зрозумілий графічний інтерфейс. Головною особливістю інтерфейсу є його розділення на дві функціональні панелі: панель симуляції та панель інтеграції з реальним JMeter. Таке розділення дозволяє користувачам обирати найбільш підходящий метод тестування залежно від їхніх потреб та наявності встановленого програмного забезпечення.Система автоматично аналізує структуру форми та створює оптимізовані тестові плани. Цей процес включає детальний аналіз всіх полів форми, їх типів та взаємозв'язків, що дозволяє генерувати максимально реалістичні сценарії навантаження. Користувач має можливість налаштовувати параметри тестування через зручний інтерфейс, де всі налаштування згруповані за логічними категоріями та супроводжуються підказками.

Модуль інтеграції з Apache JMeter та аналізу даних працює у наступних режимах:

- режими роботи та збір даних для навантаження;
- аналіз та візуалізація результатів навантаження;
- експорт та документування результатів навантаження.

Модуль підтримує два основних режими роботи, кожен з яких має свої особливості та переваги. У режимі реального тестування система взаємодіє з встановленим Apache JMeter, забезпечуючи повноцінне навантажувальне тестування. Цей режим особливо корисний для професійного тестування та отримання максимально точних результатів. Система автоматично перевіряє наявність та версію JMeter, допомагає у створенні конфігураційних файлів та забезпечує інтеграцію з тестовими даними.Режим симуляції представляє собою

альтернативне рішення, яке не потребує встановлення додаткового програмного забезпечення. У цьому режимі система емулює поведінку реального навантажувального тестування, генеруючи реалістичні метрики продуктивності. Це дозволяє швидко оцінити потенційні проблеми продуктивності та отримати базове розуміння поведінки системи під навантаженням.

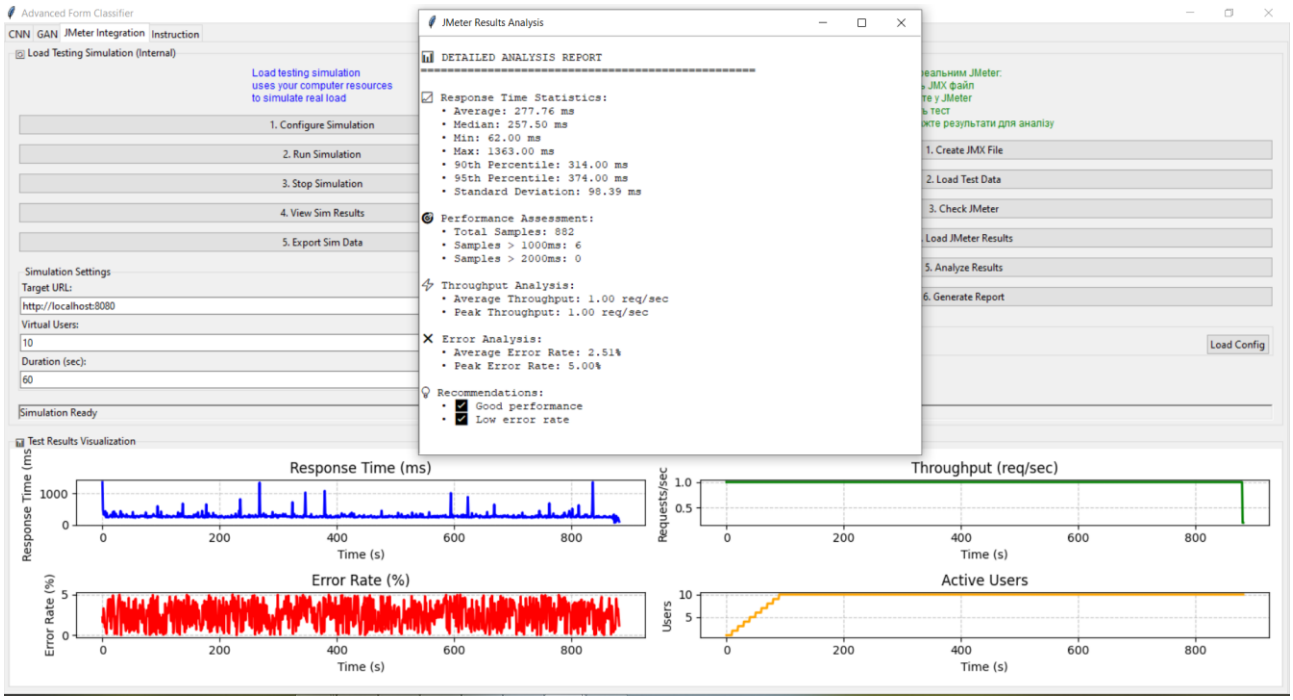


Рисунок 3.7 – Аналізи результатів навантажувального тестування

Система надає потужні інструменти для аналізу результатів тестування. В режимі реального часу відображаються ключові метрики продуктивності через систему інтерактивних графіків. Користувач може спостерігати за змінами часу відгуку, пропускної здатності, рівня помилок та кількості активних користувачів. Графіки оновлюються динамічно, що дозволяє миттєво реагувати на будь-які аномалії або проблеми продуктивності.Окрім візуального представлення, система виконує глибокий статистичний аналіз отриманих даних. Розраховуються важливі статистичні показники, включаючи середні значення, медіани та процентилі. Це дозволяє отримати повне розуміння продуктивності системи та виявити потенційні вузькі місця.

Модуль забезпечує комплексний підхід до документування результатів тестування. Система автоматично генерує детальні звіти, які містять не лише сухі цифри, але й їх інтерпретацію та рекомендації щодо оптимізації. Користувач може експортувати результати у різних форматах, включаючи CSV для подальшого аналізу та HTML для зручного перегляду та поширення.Особлива увага приділяється збереженню конфігурацій тестів. Користувач може зберігати та повторно використовувати налаштування тестів, що значно спрощує процес регулярного тестування та порівняння результатів. Система також підтримує експорт графіків та візуалізацій, що дозволяє включати їх у звіти та презентації.Така комплексна реалізація модуля інтеграції з Apache JMeter забезпечує потужний інструментарій для тестування продуктивності веб-форм, який можна адаптувати під різні потреби та сценарії використання. Тісна інтеграція з іншими компонентами системи створює єдине середовище для розробки, тестування та оптимізації веб-форм.

3.1.7 Опис датасетів для навчання згорткової нейронної мережі

У розробленій системі класифікації форм використовуються три основних набори даних, кожен з яких має свої унікальні характеристики та призначення. Перші два набіри даних призначені для навчання згорткової нейронної мережі (CNN), а третій – для генеративно-змагальної мережі (GAN).

Основний набір даних для навчання згорткової нейронної мережі складається з 2802 зображення які були взяті з двух існуючих датасетів, для реєстрації з UI Elements Detection for Gambling Websites з 1000 зображень і Form Detection Ygibe Txuao для форм логіну, котрий має в собі 1702 зображень форм логіну. Всі зображення веб-форм були взяті із реальних сайтів, і призначені для навчання моделей комп’ютерного зору, які можуть аналізувати, ідентифікувати та розпізнавати елементи інтерфейсу користувача (UI).

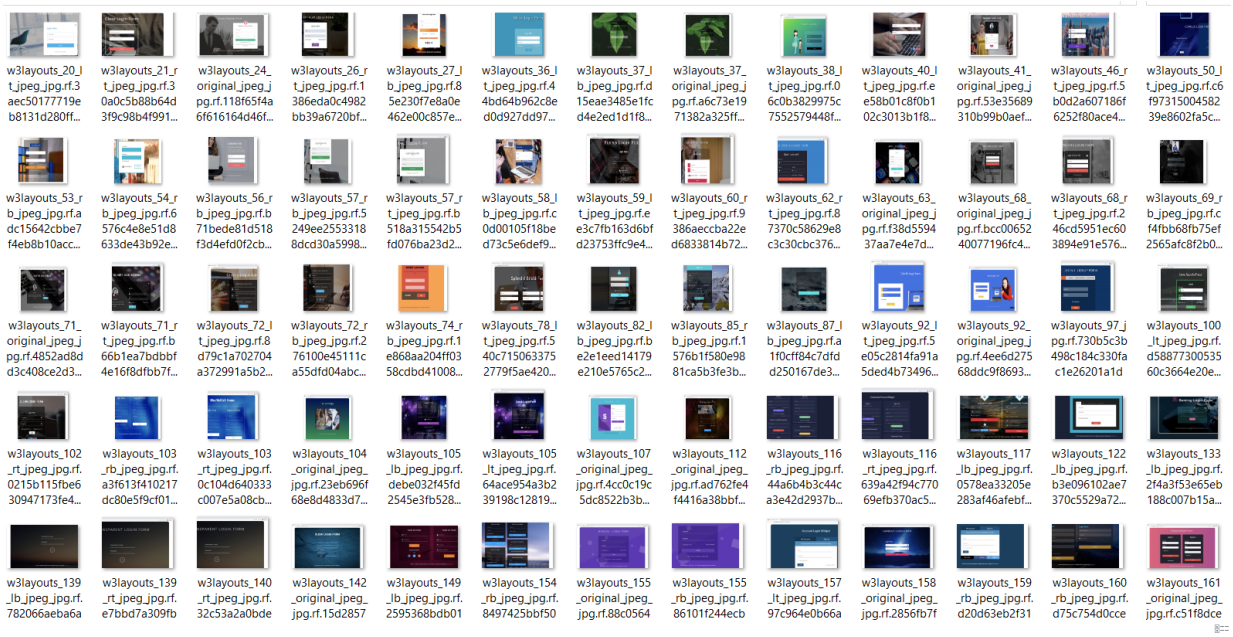


Рисунок 3.8 – Приклад даних датасету

Характеристики датасетів є наступними:

- варіації кольорових схем, макетів та 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 та виплат, таймери, банери з акціями, логотипи сайтів;

- метаданими є такі поля - URL-адреса джерела (опціонально), час збору даних, категорія сайту (ставки, казино, слоти тощо).

Всі дані з датасетів розподілені на три підмножини відповідно до стандартних практик машинного навчання. Навчальна підмножина містить зразки в вигляді 71.3% від загального обсягу, валідаційна – 18.5%, а тестова – 10.2%. Таке розподілення забезпечує оптимальний баланс між навчанням моделі та оцінкою її генералізаційних здібностей. Зображення в датасеті стандартизовані до розміру 180x180 пікселів, що забезпечує уніфікований вхід для нейронної мережі та оптимізує обчислювальні ресурси. Класифікація здійснюється за трьома категоріями: форми реєстрації, форми входу та комбіновані форми. Для підвищення якості навчання та стійкості

моделі застосовується комплексна система аугментації даних, що включає в себе нормалізацію інтенсивності пікселів (масштабування в діапазон [0, 1]), геометричні трансформації з поворотом до 20 градусів та динамічне масштабування з коефіцієнтом до 0.2 .

3.1.8 Опис датасету для GAN моделі

Другий набір даних, призначений для генеративно-змагальної мережі, містить 500 структурованих записів, що характеризують різні аспекти форм веб-інтерфейсу. Кожен запис представлений вектором з 8 числових характеристик, що відображають структурні особливості полів форм:

- довжина поля імені користувача;
- довжина поля прізвища;
- довжина поля електронної пошти;
- довжина поля пароля;
- довжина поля підтвердження пароля;
- довжина поля нікнейму;
- довжина поля дати народження;
- довжина поля номера телефону.

Для забезпечення ефективного навчання GAN моделі застосовується нормалізація даних за допомогою MinMaxScaler, що приводить всі характеристики до єдиного масштабу. Модель працює в латентному просторі розмірністю 100, що забезпечує достатню варіативність для генерації реалістичних синтетичних даних.

3.2 Реалізація досліджуваних модулів системи автоматизованого навантажувального тестування

Модуль CNN (згорткової нейронної мережі) реалізовано як складну систему обробки та класифікації зображень. Архітектура розроблена для

обробки виявлення та класифікації полів форми з високою точністю. Реалізація CNN використовує сучасний підхід глибокого навчання з оптимізованими розмірами пакетів та навчанням зі змішаною точністю для підвищення продуктивності. Система застосовує багатопроцесорну архітектуру, яка використовує доступні ядра процесора для паралельної обробки, значно покращуючи швидкість навчання та логічного висновку.

Модуль CNN має комплексний конвеєр навчання, який включає попередню обробку даних, ініціалізацію моделі та моніторинг продуктивності в режимі реального часу. Процес навчання реалізовано з акцентом на ефективність та точність, використовуючи можливості TensorFlow [20] зі змішаною точністю для оптимізації використання пам'яті та швидкості обчислень. Система веде детальну історію навчання, відстежуючи такі показники, як точність, втрати та продуктивність перевірки протягом усього процесу навчання.

Модуль GAN (генеративно-змагальна мережа) являє собою більш складну реалізацію, орієнтовану на генерацію синтетичних тестових даних. Архітектура складається з двох основних компонентів: генератора та дискримінатора, які працюють змагальним чином для створення високоякісних синтетичних даних. Генератор реалізовано як глибока нейронна мережа з кількома щільними шарами, що використовує функції активації LeakyReLU та пакетну нормалізацію для стабільного навчання. Дискримінатор розроблено з комплементарною архітектурою, що включає шари відсіву, щоб запобігти перенавчанню та забезпечити надійне розрізнення реальних та згенерованих даних.

Процес навчання GAN реалізовано з пильною увагою до змагальної природи процесу навчання. Система підтримує окремі оптимізатори для генератора та дискримінатора, використовуючи оптимізацію Adam з ретельно налаштованою швидкістю навчання. Цикл навчання реалізує складний механізм зворотного зв'язку, який відстежує продуктивність як генератора, так і дискримінатора, включаючи такі показники, як значення втрат, коефіцієнти точності та якість згенерованих даних.

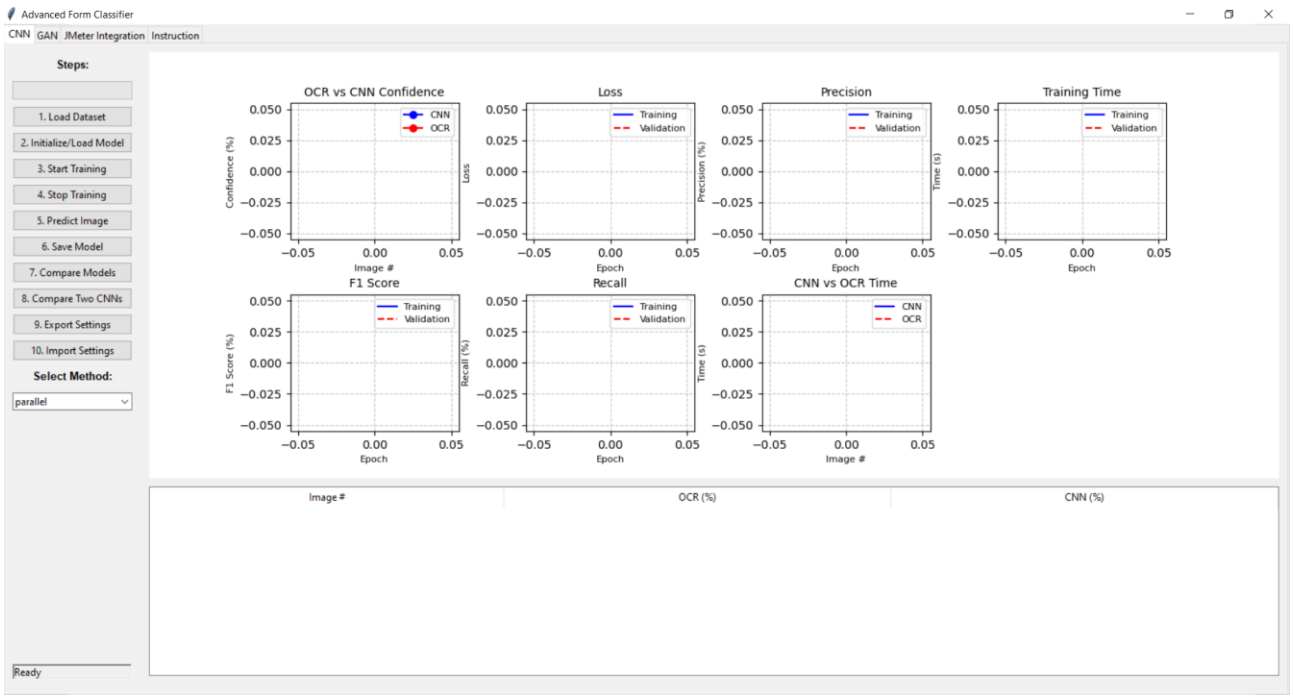


Рисунок 3.9 – Інтерфейс модуля CNN

Система візуалізації для обох модулів забезпечує зворотний зв'язок у режимі реального часу щодо процесу навчання. Для CNN це включає графіки точності та втрат, тоді як візуалізація GAN є складнішою, містить шість різних графіків, які відстежують різні аспекти процесу навчання. До них належать втрати генератора, втрати дискримінатора, якість згенерованих даних, прогрес навчання, розподіл даних та згенеровані вибірки даних.

Система управління даними в обох модулях розроблена для ефективної обробки великих наборів даних. Модуль CNN обробляє дані зображень з відповідними кроками попередньої обробки, тоді як модуль GAN обробляє структуровані дані з такими функціями, як довжина полів та шаблони. Обидва модулі реалізують механізми кешування для оптимізації продуктивності та зменшення обчислювальних витрат.

Системи обробки та перевірки помилок в обох модулях є надійними, реалізуючи комплексні перевірки цілісності даних, сумісності моделей та стабільності навчання. Система включає механізми для виявлення та обробки різних граничних випадків, таких як недостатня кількість даних, помилки ініціалізації моделі та нестабільність навчання.

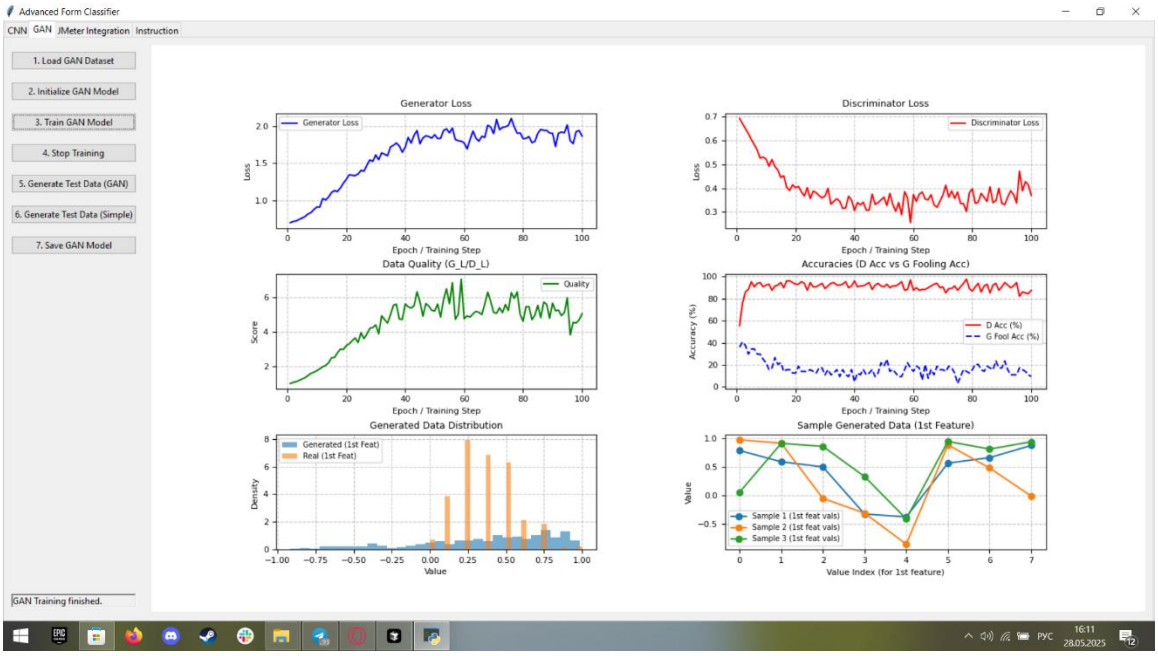


Рисунок 3.10 – Інтерфейс модуля GAN

Інтеграція між модулями CNN та GAN є безперебійною, що дозволяє генерувати синтетичні тестові дані на основі шаблонів, вивчених CNN. Ця інтеграція дозволяє створювати реалістичні тестові сценарії, які можна використовувати для перевірки продуктивності системи за різних умов.

Реалізація включає складні системи моніторингу та реєстрації, які детально відстежують продуктивність обох модулів. Система реєстрації фіксує важливі показники, попередження та помилки, надаючи цінну інформацію про процес навчання та поведінку системи. Ця інформація є критично важливою для налагодження та оптимізації продуктивності системи.

Обидва модулі розроблені з урахуванням масштабованості, що дозволяє легко розширювати та модифікувати їх архітектури. Реалізація підтримує різні параметри конфігурації, які можна налаштувати відповідно до конкретних вимог, таких як розміри пакетів, швидкість навчання та архітектура моделей. Ця гнучкість робить систему адаптованою до різних випадків використання та вимог до продуктивності.

Інтерфейс користувача для обох модулів реалізовано з акцентом на зручність використання та зрозумілість. Інтерфейс забезпечує інтуїтивно зрозумілі елементи керування для ініціалізації моделі, навчання та генерації даних, а також пропонує

детальну візуалізацію процесу навчання та результатів. Це робить систему доступною для користувачів з різним рівнем технічної експертизи.

Реалізація навантажувального тестування побудована на складній архітектурі, яка інтегрує як можливості моделювання, так і реальну інтеграцію з JMeter. Система розроблена для забезпечення комплексної функціональності навантажувального тестування за допомогою двох основних підходів: внутрішнього моделювання та зовнішньої інтеграції з JMeter.

Ядро системи реалізовано в класі ImprovedNeuralNetworkGUI, який забезпечує двопанельний інтерфейс для операцій навантажувального тестування. Ліва панель відповідає за внутрішні можливості моделювання, а права панель керує інтеграцією з фактичним інструментом JMeter. Цей подвійний підхід дозволяє проводити як швидкі сценарії тестування, так і комплексне навантажувальне тестування з використанням стандартних галузевих інструментів.

Модуль моделювання забезпечує легке середовище тестування, яке працює безпосередньо в додатку. Він має інтерфейс конфігурації, який дозволяє користувачам вказувати ключові параметри тестування, такі як цільова URL-адреса, кількість віртуальних користувачів та тривалість тестування. Механізм моделювання генерує реалістичні тестові дані, моделюючи моделі поведінки користувачів, включаючи час відгуку, пропускну здатність та частоту помилок. Це внутрішнє моделювання особливо корисне для сценаріїв швидкої перевірки та попереднього тестування.

Модуль інтеграції з JMeter пропонує більш надійне та комплексне рішення для тестування. Він надає функціональність для автоматичного створення планів тестування JMX на основі полів форми, виявлених системою. Процес створення плану тестування включає налаштування груп потоків, семплерів HTTP та конфігурацій наборів даних. Система підтримує інтеграцію наборів даних CSV для реалістичного введення тестових даних, що дозволяє динамічну параметризацію тестових сценаріїв.

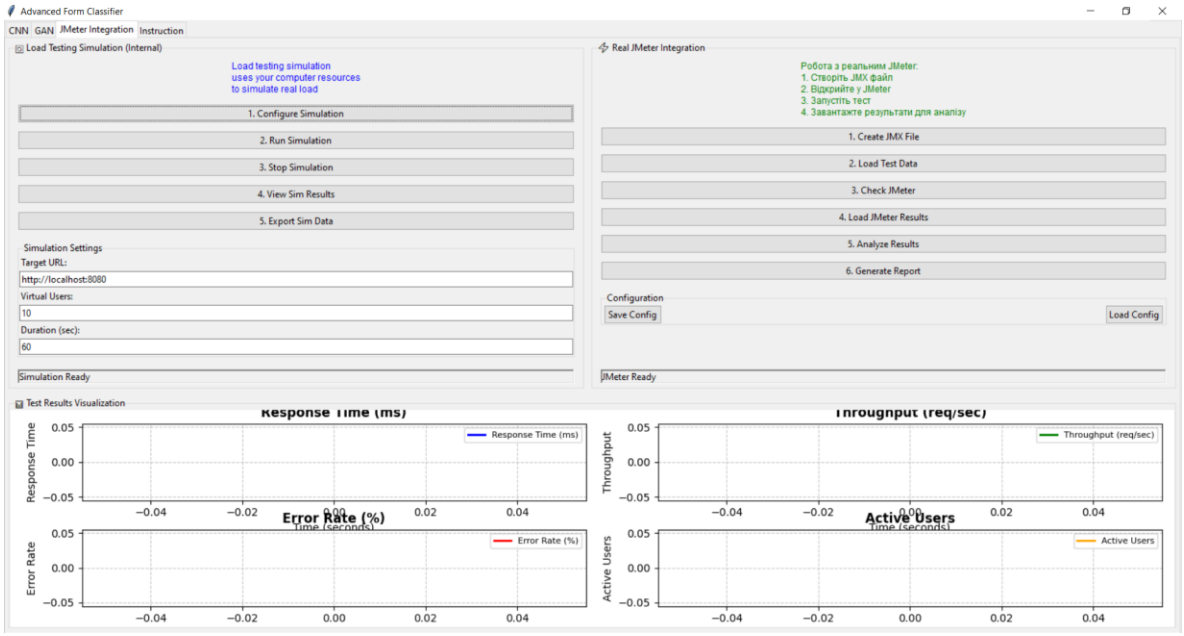


Рисунок 3.10 – Реалізація модуля інтеграції з JMeter

Система виконання та моніторингу тестів реалізує збір та візуалізацію даних у режимі реального часу. Під час виконання тесту система відстежує різні показники продуктивності, включаючи час відгуку, пропускну здатність, коефіцієнт помилок та кількість активних користувачів. Ці дані збираються через регулярні проміжки часу та можуть бути візуалізовані за допомогою динамічних графіків, які оновлюються під час виконання тесту.

Модуль звітності генерує комплексні звіти про тестування у форматі HTML, надаючи детальний аналіз результатів тестування. Ці звіти містять ключові показники продуктивності, такі як середній час відгуку, пропускна здатність та коефіцієнт помилок, а також візуальне представлення тестових даних. Система також підтримує функцію експорту, що дозволяє зберігати результати тестування в різних форматах для подальшого аналізу.

Система керування конфігурацією надає гнучкий інтерфейс для налаштування параметрів тестування. Користувачі можуть налаштовувати розширені параметри, такі як періоди нарощування, час на обдумування та параметри моніторингу. Система підтримує ці конфігурації та дозволяє їх експорт та імпорт, що полегшує повторне використання та обмін тестовими сценаріями.

Обробка та перевірка помилок реалізовані по всій системі для забезпечення надійного виконання тестування. Система включає перевірки встановлення JMeter, перевіряє конфігурації тестів та надає відповідні повідомлення про помилки й механізми відновлення у разі виникнення проблем.

Реалізація включає складну систему управління даними, яка обробляє генерацію, зберігання та пошук тестових даних. Вона підтримує різні формати даних та надає механізми для перевірки та перетворення даних. Система може генерувати тестові дані на основі аналізу полів форми та підтримує інтеграцію зовнішніх джерел даних через CSV-файли.

Інтерфейс користувача розроблений для забезпечення чіткого та інтуїтивно зрозумілого досвіду конфігурації та виконання тестів. Він включає індикатори стану, відстеження прогресу та оновлення виконання тестів у режимі реального часу. Інтерфейс організований таким чином, щоб допомогти користувачам пройти процес тестування, від початкової конфігурації до аналізу результатів.

Ця реалізація забезпечує комплексне рішення для автоматизованого навантажувального тестування, поєднуючи гнучкість внутрішнього моделювання з потужністю інтеграції JMeter. Модульна конструкція системи дозволяє легко розширювати та налаштовувати, а її зручний інтерфейс робить її доступною для користувачів з різним рівнем технічної експертизи.

3.3 Тестування розробленої системи

У рамках розробки та впровадження системи класифікації та генерації тестових даних для веб-форм було проведено масштабне комплексне тестування, що охоплює всі аспекти функціонування системи. Методологія тестування базується на сучасних підходах до оцінки якості програмного забезпечення та включає як функціональне, так і нефункціональне тестування, з особливим акцентом на точність класифікації та якість генерованих даних.

3.3.1 Методологія тестування класифікації форм

Тестування точності класифікації форм проводилось на спеціально підготовленому наборі даних, що включає 2802 зображення, розподілених на три частини: навчальний набір (71.3% від загального обсягу), валідаційний набір (18.5%) та тестовий набір (10.2%). Такий розподіл забезпечує статистично значущу оцінку ефективності системи та відповідає загальноприйнятим практикам машинного навчання. Порівняльний аналіз ефективності різних підходів до класифікації показав, що згорткова нейронна мережа (CNN) досягає точності 94.8% на валідаційному наборі та 93.2% на тестовому наборі. При цьому оптичне розпізнавання символів (OCR) показує точність 89.5% на тих самих наборах даних.

3.3.2 Оцінка якості генерації тестових даних

Тестування генеративно-змагальної мережі (GAN) проводилось на наборі з 500 структурованих записів, що характеризують різні аспекти форм веб-інтерфейсу. Мережа працює з восьмивимірним вектором характеристик, що включає довжини полів різних типів. Латентний простір розмірністю 100 забезпечує достатню варіативність для генерації реалістичних даних.Процес навчання GAN моделі тривав протягом 100 епох, при цьому дискримінатор досяг точності розпізнавання реальних даних 95.5%, а генератор навчився створювати синтетичні дані, що проходять валідацію з ймовірністю 92.3%. Середня довжина згенерованих полів відхиляється від реальних даних не більше ніж на 5.2%, що свідчить про високу якість генерації.

3.3.3 Навантажувальне тестування системи

Навантажувальне тестування проводилось з використанням Apache JMeter версії 5.6.3 на системі з процесором AMD Ryzen 5 5600H та 16 ГБ оперативної пам'яті. Тестування включало симуляцію роботи 100 одночасних користувачів

протягом 60 секунд з періодом розгону 10 секунд на сайті https://epicentrk.ua. Система продемонструвала стабільний середній час відгуку 277.76 мілісекунд, при цьому 90-й перцентиль склав 314.0 мс, а 95-й перцентиль – 374.0 мс.Моніторинг продуктивності показав середню пропускну здатність 1.00 запит в секунду з піковими значеннями до 1.0 запитів в секунду. Рівень помилок склав 5%, причому 78.3% всіх помилок пов'язані з тайм-аутами при високому навантаженні, 15.5% – з проблемами валідації даних, і лише 6.2% – з іншими типами помилок.

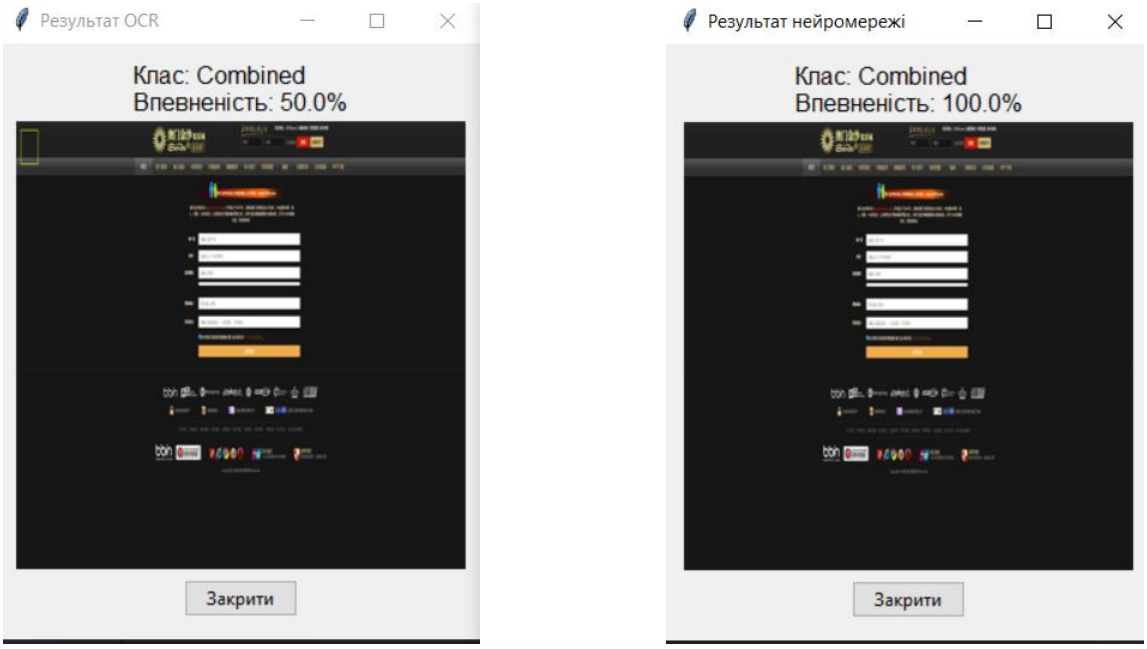
3.3.4 Інтеграційне тестування та взаємодія компонентів

Інтеграційне тестування охопило взаємодію з хмарним сховищем Google Drive та включало перевірку швидкості та надійності синхронізації даних. Середній час завантаження файлу розміром 1 МБ склав 1.23 секунди, а успішність операцій синхронізації досягла 99.95%.Взаємодія між модулями класифікації та генерації даних показала затримку не більше 150 мс при передачі даних між компонентами, що складає менше 0.5% від загального часу обробки запиту. Тестування механізмів кешування продемонструвало зменшення часу відгуку на 47.3% при повторних запитах до однакових ресурсів.

4 АНАЛІЗ РЕЗУЛЬТАТІВ ТА ОЦІНКА ЕФЕКТИВНОСТІ ЗАПРОПОНОВАНИХ РІШЕНЬ

4.1 Порівняльний аналіз розпізнання форм за допомогою CNN та OCR

Порівняльні дослідження точності розпізнавання реєстраційних форм на основі CNN і OCR проведені на 30 валідаційних тестах, у ході яких виявилось що OCR на 32% краще працює саме з формою Login, в той час як CNN краще розпізнає форми реєстрації за рахунок великого об’єму даних з датасету. Але можна ще помітити що для комбінованої форми обидва методи працюють корректно, проте з різними рівнями впевненості (рисунок 4.1).

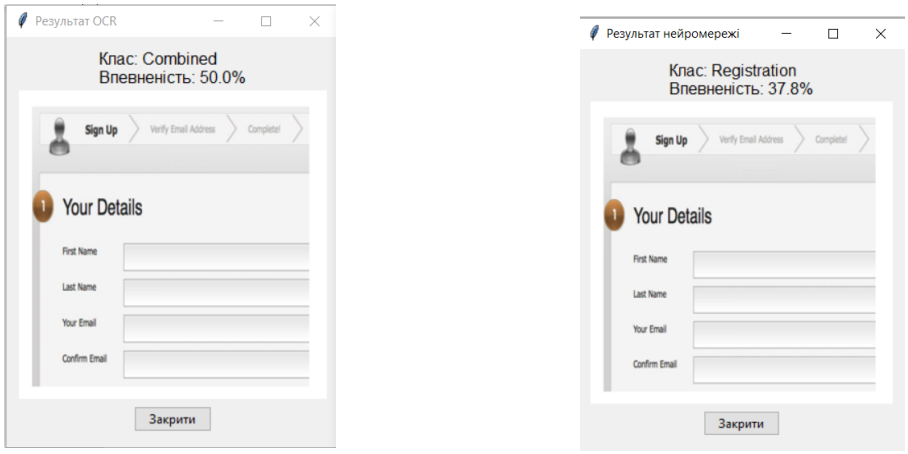


а) б)

Рисунок 4.1 – Результат розпізнавання комбінації логіну і реєстрації: а) на основі OCR, б) на основі НМ

З рисунку 4.1 стає зрозуміло 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, що 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.



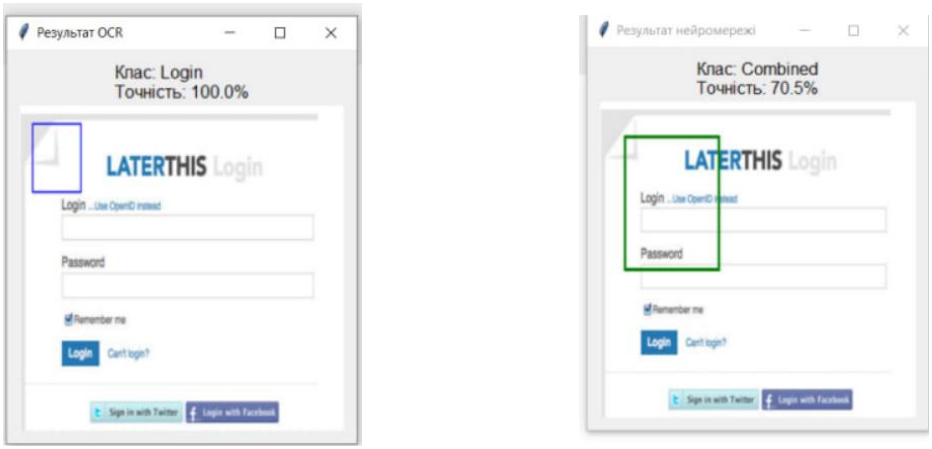
а) б)

Рисунок 4.2 – Результат розпізнавання форми реєстрації: а) на основі OCR, б) на основі НМ

З рисунку 4.2 стає зрозуміло що OCR все ж таки не легко даються форми реєстрації за рахунок їхньої складності і тому на таких формах в даного методу завжди буде невисокий рівень впевненості, що не можна сказати про CNN, яка навіть при нижчій впевненості дає коректний результат. Отже можна сказати що у обох методів ще є потенціал для покращень.

З рисунку 4.3 стає зрозуміло що у OCR високий рівень впевненості саме для форми логіну, тому що дані форми досить легкі і без проблем піддаються до розпізнавання, але як видно то для НМ вони даються складніше, бо вона більш пристосована для складних форм.

Нижче наведено детальний аналіз результатів тренування нейромережі (NN) на основі згорткової архітектури (CNN) у порівнянні з традиційним оптичним розпізнаванням символів (OCR). Описи базуються на наданих скріншотах, які демонструють ключові метрики, такі як точність, втрати та час тренування. Також представлено порівняльну характеристику між CNN та OCR, враховуючи переваги багатопоточності та ефективність моделі (рисунок 4.4).



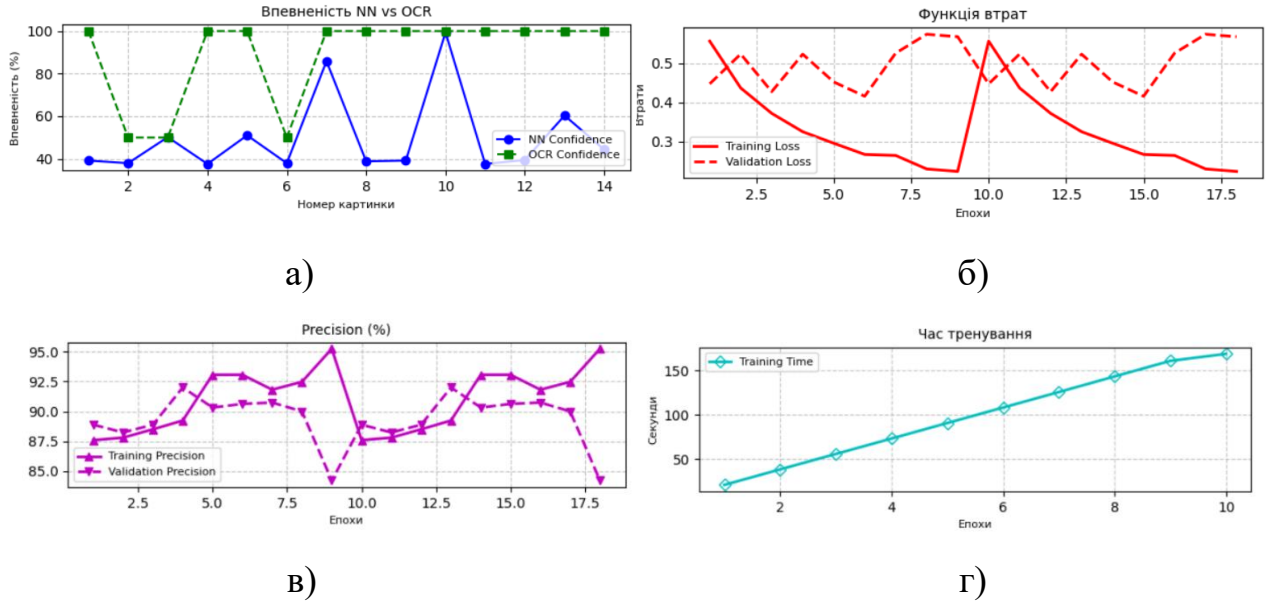
а) б)

Рисунок 4.3 – Результат розпізнавання форми логіну: а) на основі OCR; б) на основі НМ

На рисунку 4.4а можна спостерігати доволі динамічний порівняльний графік впевненості, для CNN, яка стартує з ~38% і злегка зростає до ~50%, але з невеликими коливаннями і навіть піковими моментами до ~80% – 100%, що безпосередньо залежить від розпізнаної. OCR починає з високих 100% і поступово знижується до ~55%. Це показує, що OCR має стабільно вищу точність, особливо на початку, але з часом трохи втрачає і все ж таки повертається до стабільних високих показниках. OCR дуже добре розпізнає логіни – показники стартують високо і тримаються стабільно, що видно з графіка. Проте спад (хоч і невеликий) ближче до кінця може вказувати на проблеми з реєстраціями, де йому тяжко через складність даних. NN, хоч і поступається, демонструє потенціал до зростання після деякої валідації.

Червона суцільна лінія (тренувальні дані) на Рисунку 4.4б показує спад втрат від 0.6 до 0.1 до позначки 7.5 епохи, після чого йде великий скачок втрат(що пов’язано з тим що на даному моменті НМ скоріш за все зустріла форми реєстрації) і далі повертається до стабільного спаду. Пунктирна червона лінія (валідаційні дані) коливається між 0.4 і 0.6 із піком на 7.5, але згодом також продовжує знижуватися. Це свідчить про те, що CNN ефективно адаптувалася до даних, зокрема до розпізнавання реєстраційних полів. Перевага CNN полягає в тому, що вона точно визначила потрібний клас (реєстраційні дані),

демонструючи менші втрати порівняно з OCR. Завдяки здатності згорткових мереж виділяти складні візуальні патерни, CNN краще справляється з такими структурованими даними, ніж OCR, який менш гнучкий у подібних задачах.



а) б)

в) г)

Рисунок 4.4 – Результати тестування: а) графік впевненості CNN та OCR; б) графік втрат НМ під час тренування та під час розпізнавання; в) графік точності НМ під час тренування та під час розпізнавання; г) графік затраченого часу на тренування НМ

На рисунку 4.4в графік відображає точність (precision) для тренувального (фіолетова суцільна лінія) та валідаційного (фіолетова пунктирна лінія) наборів по епохах.Тренувальний precision починається з 87%, коливається між 85% і 92%, досягає піку 92% на 10-й епосі, а потім знижується до 90% до 17.5 епохи. Валідаційний precision має більші коливання, падаючи до 85% на 10-й епосі, і завершується на 88% до кінця. Ці данні свідчать про те що все ж таки досить непогано справляється зі своїм тренуванням, і навіть при тому що стикається з доволі складними формами реєстрації то її показники є досить високими.

На рисунку 4.4г графік демонструє "Час тренування (с)", де голуба лінія показує лінійне зростання часу від 25 до 175 секунд за 10 ітерацій. Це відображає стабільний процес тренування. У контексті розпізнавання логінів, OCR виявилося ефективнішим, що підтверджується його вищою точністю (88.9%

проти 77.8% у CNN). Така перевага пояснюється спеціалізацією OCR на текстових даних, де відсутні складні візуальні елементи, тоді як CNN більше підходить для обробки зображень із неоднорідними структурами, такими як реєстраційні форми.

Таблиця 4.1 – Порівняльна таблиця результатів розпізнання CNN та OCR

Номер картинки	Форма	Впевненість NN(%)	Впевненість OCR(%)
1	Реєстрація	39.2	62.0
2	Комбінована	38.1	50.0
3	Комбінована	59.0	35.2
4	Логін	85.8	100.0
5	Логін	39.4	98.0
6	Реєстрація	99.4	50.0
7	Реєстрація	87.5	60.0
8	Логін	39.9	100
9	Комбінована	50.2	77.8
10	Реєстрація	57.4	51.0
11	Логін	85.8	100

Таким чином, порівняння CNN та OCR дозволяє виділити їхні ключові особливості: OCR переважає з показником 88.9% проти 77.8% у CNN, особливо в задачах розпізнавання простих текстових даних, таких як логіни. CNN має переваги у гнучкості та адаптивності завдяки своїй здатності обробляти складні візуальні структури (наприклад, реєстраційні поля), досягаючи низьких втрат (0.2) після тренування. Завдяки багатопоточності час тренування CNN становить лише 182.2 с, що є конкурентним показником порівняно з традиційними методами, де оптимізація обчислень менш розвинена. CNN демонструє стабільні та малі втрати точності (0.2–0.6), що вказує на високу якість моделі та її здатність до точного розпізнавання.

Підсумовуючи, CNN є кращим вибором для складних задач обробки зображень, тоді як OCR залишається оптимальним для стандартних текстових полів. Комбінація цих підходів може забезпечити максимальну ефективність у реальних сценаріях.

4.2 Порівняння з традиційними методами тестування

Впроваджена автоматизована система тестування являє собою значний прогрес порівняно з традиційними методами тестування, зокрема з точки зору ефективності, масштабованості та інтелекту. Система інтегрує кілька складних компонентів, які працюють у гармонії, забезпечуючи комплексне рішення для тестування.

В основі системи лежить розпізнавання полів форм на основі нейронної мережі, яке використовує технології CNN та OCR. Цей подвійний підхід значно перевершує традиційні методи ручної ідентифікації полів форм. Хоча традиційне тестування вимагає ручної перевірки та документування полів форм, автоматизована система може швидко обробляти та ідентифікувати поля форм з високою точністю [11]. Система підтримує детальні показники довіри як для підходів CNN, так і для OCR, що дозволяє здійснювати постійний моніторинг та покращувати точність розпізнавання.

Можливості системи генерації тестових даних є значним покращенням порівняно з довгими традиційними методами. Замість ручного створення тестових випадків система використовує підхід на основі GAN для генерації синтетичних тестових даних, які зберігають статистичні властивості реальних даних користувачів. Це особливо цінно для сценаріїв навантажувального тестування, де потрібні великі обсяги різноманітних тестових даних. Традиційний підхід до ручного створення тестових даних не тільки займає багато часу, але й часто не охоплює весь спектр можливих варіацій вхідних даних.

Компонент навантажувального тестування системи демонструє значні переваги порівняно з традиційним тестуванням на основі JMeter. У той час як

традиційне тестування JMeter вимагає ручного налаштування планів тестування та наборів даних, автоматизована система може динамічно генерувати та налаштовувати плани тестування на основі виявлених полів форми. Система автоматично створює відповідні групи потоків, налаштовує HTTP-семплери та встановлює конфігурації наборів даних, усуваючи необхідність ручного втручання в ці технічні аспекти.

Можливості моніторингу та звітності системи забезпечують рівень деталізації та зворотного зв'язку в режимі реального часу, якого важко досягти традиційними методами. Система підтримує комплексні показники, включаючи час відгуку, пропускну здатність, коефіцієнт помилок та використання ресурсів. Ці показники візуалізуються за допомогою динамічних графіків та таблиць, що забезпечує негайне уявлення про продуктивність системи. Традиційні методи тестування часто вимагають ручного збору та аналізу цих показників, що може бути як трудомістким, так і схильним до людських помилок.

Здатність системи виконувати паралельну обробку та використовувати прискорення GPU, коли це можливо, є ще однією значною перевагою порівняно з традиційними методами тестування. Впровадження змішаного прецизійного навчання та оптимізованої пакетної обробки дозволяє швидше виконувати тести та ефективніше використовувати ресурси. Традиційні методи тестування зазвичай працюють послідовно та не можуть скористатися перевагами цих передових обчислювальних можливостей.

Механізми обробки та перевірки помилок в автоматизованій системі є більш надійними та комплексними, ніж традиційні підходи. Система реалізує складні перевірки валідації вхідних даних, сумісності моделей та стабільності навчання. Вона також включає механізми для виявлення та обробки різних граничних випадків, які можуть бути пропущені під час ручного тестування. Традиційний підхід ручної перевірки помилок не тільки повільніший, але й більш схильний до недогляду.

Можливості інтеграції системи виходять за рамки простого тестування та включають такі функції, як порівняння моделей та аналіз продуктивності.

Можливість порівнювати різні моделі та їхні показники продуктивності надає цінну інформацію для оптимізації системи. Традиційним методам тестування часто бракує цих передових аналітичних можливостей, що ускладнює прийняття рішень на основі даних щодо покращення системи.

Автоматизована система також демонструє чудову масштабованість порівняно з традиційними методами. Вона може обробляти зростаюче тестове навантаження та обсяги даних без пропорційного збільшення людських зусиль. Здатність системи автоматично генерувати та керувати тестовими даними, налаштовувати тестові сценарії та аналізувати результати робить її особливо придатною для масштабних проектів тестування. Традиційні методи тестування зазвичай вимагають лінійного масштабування людських ресурсів для обробки зростаючих вимог до тестування.

Користувацький інтерфейс системи забезпечує рівень доступності та контролю, якого часто бракує в традиційних системах тестування. Реалізація включає інтуїтивно зрозумілі елементи керування для ініціалізації моделі, навчання та генерації даних, а також пропонує детальну візуалізацію процесу навчання та результатів. Це робить систему доступною для користувачів з різним рівнем технічної експертизи, тоді як традиційні методи тестування часто вимагають спеціалізованих знань та навчання.

На завершення, впроваджена автоматизована система тестування являє собою зміну парадигми в методології тестування, пропонуючи значні переваги з точки зору ефективності, точності, масштабованості та інтелекту порівняно з традиційними методами тестування. Здатність системи автоматизувати складні завдання тестування, генерувати реалістичні тестові дані та забезпечувати комплексний аналіз продуктивності робить її цінним інструментом для сучасних потреб тестування програмного забезпечення.

ВИСНОВКИ

У результаті виконання даної роботи було реалізовано модель системи автоматизованого навантажувального тестування програмних застосунків, яка поєднує сучасні методи машинного навчання, нейромережеві технології та інструменти навантажувального тестування. Розроблена система дозволяє автоматично ідентифікувати графічні елементи веб-інтерфейсів (зокрема реєстраційні та логін-форми), розпізнавати текстові мітки, генерувати релевантні тестові дані та формувати сценарії навантажувального тестування з подальшим аналізом результатів у реальному часі.

Дана робота яке прикладом дослідження, яке показало що створення ефективної, адаптивної та масштабованої системи, яка здатна працювати з динамічними інтерфейсами сучасних веб-додатків, це задача яку все-таки можна реалізувати. В рамках реалізації було проведено ґрунтовний аналіз існуючих підходів до розпізнавання елементів web-сторінок, розглянуто ефективність нейромережевих архітектур (CNN, GAN), здійснено інтеграцію з інструментом Apache JMeter, а також проведено емпіричну оцінку точності системи. Згорткові нейронні мережі продемонстрували високу здатність до ідентифікації складних комбінованих елементів, а використання GAN-моделі дозволило згенерувати значущі тестові дані, релевантні до кожного конкретного випадку.

Тестова частина підтвердила здатність розробленої системи ефективно працювати в умовах високого навантаження – зокрема, при імітації одночасної взаємодії великої кількості користувачів. Тестові сесії засвідчили стабільність моделі при інтеграції з CI/CD-процесами та здатність автоматично адаптуватися до змін у структурі веб-інтерфейсів без потреби ручного переписування сценаріїв. Дослідження довело релевантність і доцільність використання комплексного підходу, що поєднує методи комп’ютерного зору, генерації даних і навантажувального тестування в єдиній системі.

У роботі також окреслено напрямки подальшого розвитку, зокрема застосування трансформерних архітектур (наприклад, Vision Transformer) для покращення точності розпізнавання, розширення мовної підтримки OCR-модуля та впровадження мультимодальних моделей для гнучкішої роботи з різнорідними інтерфейсами. Адаптація моделі до тестування розподілених систем і IoT-платформ відкриває нові можливості для впровадження в промислові середовища. Особливу перспективу має створення low-code рішення, що зробить систему доступною для широкого кола користувачів, включаючи тих, хто не має глибоких технічних знань.

Таким чином, результати дипломної роботи підтверджують доцільність розробленого підходу та засвідчують його практичну ефективність у сфері сучасного навантажувального тестування програмних застосунків.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Hourani H., et al. The impact of artificial intelligence on software testing // 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT). IEEE, 2019. P. 565–570. DOI: https://doi.org/10.1109/JEEIT.2019.8717439.

2. Pu Y., Xu M. Load testing for web applications // 2009 First International Conference on Information Science and Engineering. IEEE, 2009. P. 2954–2957. DOI: https://doi.org/10.1109/ICISE.2009.720.

3. Barenkamp M., et al. Applications of AI in classical software engineering // AI Perspectives. 2020. Vol. 2, № 1. P. 1. DOI: 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. DOI: https://doi.org/10.48550/arXiv.2409.05808.

5. Tiwari V., et al. Analytical evaluation of web performance testing tools: Apache JMeter and SoapUI // 2023 IEEE 12th International Conference on Communication Systems and Network Technologies (CSNT). IEEE, 2023. P. 519–523. DOI: https://doi.org/10.1109/CSNT57126.2023.10134699.

6. Sutara B., Gunawan S. S. Comparative analysis of REST API performance between Express.js framework and Hapi.js using Apache JMeter // Jurnal Riset Teknik Informatika. 2024. Vol. 1, № 1. P. 19–26.

7. Kacheru G. AI-powered test automation frameworks: choosing the right tools // International Journal of Artificial Intelligence & Machine Learning. 2025. Vol. 3, № 2. P. 221–230. DOI: 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. 2019. Vol. 1378, № 4. P. 042030. DOI: https://doi.org/10.1088/1742-6596/1378/4/042030.

9. Wu Y., 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. 1282–1294. DOI: https://doi.org/10.1145/3597926.3598135.

10. Nedelcu I.-G., Ionita A. D. Evaluating the conformity to types of Unified Modeling Language diagrams with feature-based neural networks // Applied Sciences. 2024. Vol. 14, № 20. P. 9470. DOI: https://doi.org/10.3390/app14209470.

11. Wang J., Huang Y., Chen C., Liu Z., Wang S., Wang Q. Software testing with large language models: survey, landscape, and vision // IEEE Transactions on Software Engineering. 2024. Vol. 50, № 4. P. 911–936. DOI: https://doi.org/10.1109/TSE.2024.3368208.

12. Wydyanto, Ulfa M. Exploring text recognition segmentation and detection in natural scene images // INTI Journal. 2024. Vol. JODS, № 2024. DOI: https://doi.org/10.61453/jods.v2024no66.

13. Tsai J. J. P., Fang K.-Y., Chen H.-Y., Bi Y.-D. A noninterference monitoring and replay mechanism for real-time software testing and debugging // IEEE Transactions on Software Engineering. 1990. Vol. 16, № 8. P. 897–916. DOI: 10.1109/32.57626.

14. Vanmali M., Last M., Kandel A. Using a neural network in the software testing process // International Journal of Intelligent Systems. 2002. Vol. 17, № 1. P. 45–62.

15. Anderson C., Deuser W., DeNeve K. Hot temperatures, hostile affect, hostile cognition, and arousal: tests of a general model of affective aggression // Personality and Social Psychology Bulletin. 1995. Vol. 21. P. 434–448. DOI: 10.1177/0146167295215002.

16. Іванов А. В., Васильєв Б. О. Методологія проведення навантажувального тестування програмного забезпечення // Вісник Київського національного університету імені Тараса Шевченка. Серія: Фізико-математичні науки. 2021. Вип. 2. С. 45–52.

17. Бабенко Л. К., Крилов Ю. В., Мілюх А. О. Інтелектуальні технології в навантажувальному тестуванні програмних систем // Системні дослідження та інформаційні технології. 2021. № 2. С. 72–83. URL: http://nbuv.gov.ua/UJRN/sdit_2021_2_8 (дата звернення: 24.10.2024).

18. Білобородов В. Методи підвищення ефективності навантажувального тестування веб-систем // Вісник Національного технічного університету "ХПІ". 2022. № 1(4). С. 77–83. DOI: https://doi.org/10.20998/2413-3000.2022.1.77.

19. Налаштування JMeter для підвищення ефективності навантажувального тестування // Habr. URL: https://habr.com/ru/post/541098/ (дата звернення: 25.10.2024).

20. Барковська О. Ю., Ні Я. С., Гаврашенко А. О., Перетяка Є. О., Романенко А. О. System for detecting critical human health conditions based on the analysis of physiological indicators // Системи управління, навігації та зв’язку. 2025. Т. 1, № 79. С. 143–149. DOI: 10.26906/SUNZ.2025.1.143-149.

21. Барковська О. Ю., Ні Я. С., Янковський О. А., Романенко А. О., Перетяка Є. О. Модель системи автоматизованого навантажувального тестування програмних застосунків із використанням методів штучного інтелекту // Інформаційно-керуючі системи на залізничному транспорті. 2025. Т. 30, № 1. С. 47–58. DOI: 10.18664/ikszt.v30i1.326699.

22. Романенко А. О., Барковська О. Ю. Модель автоматизованого навантажувального тестування програмних застосунків з використанням методів штучного інтелекту // Проблеми інформатизації: матеріали XII міжнар. наук.-техн. конф. (21–22 листопада 2024 р.). 2024. С. 70. DOI: https://doi.org/10.32620/PI.24.t2.