«Системні технології» 3 (164) 2026 «System technologies»
DOI 10.34185/1562-9945-3-164-2026-24
УДК 004.932

В.Ю. Царик, Вік.В. Гнатушенко, Є.В. Ольшанский

ПІДХІД РОЗПІЗНАВАННЯ ЕЛЕМЕНТІВ GUI ЯК ЗОБРАЖЕНЬ НА ОСНОВІ МАШИННОГО НАВЧАННЯ

Анотація. У сучасних дослідженнях у галузі комп'ютерного зору значна увага приділяється задачам виявлення та класифікації об'єктів на зображеннях, зокрема із застосуванням глибокого навчання. Водночас питання застосування цих підходів до розпізнавання елементів графічного інтерфейсу залишається недостатньо дослідженим. У статті розглянуто підхід, що передбачає використання елементів веб-інтерфейсу як вхідних даних для моделей детекції об'єктів як зображень. Запропоновано процес, у межах якого здійснюється отримання зображення за допомогою Selenium, подальша обробка з використанням моделі машинного навчання та передача результатів у форматі JSON для тестування в середовищі PyTest. Проведено порівняльний аналіз моделей Swin-L, EfficientDet-D7, Cascade Mask R-CNN, RetinaNet та YOLO з точки зору їх придатності до задачі розпізнавання елементів інтерфейсу. За результатами дослідження встановлено, що використання моделей YOLO є найбільш доцільним для задачі розпізнавання елементів графічного інтерфейсу користувача. Запропонований підхід дозволяє зменшити залежність від DOM-структури, підвищити стійкість тестових сценаріїв до змін інтерфейсу та знизити витрати на їх підтримку. Отримані результати можуть бути використані для подальшого розвитку систем автоматизованого тестування веб-застосунків.
Ключові слова: розпізнавання, зображення, GUI, модель, глибоке навчання, тестування.

Постановка проблеми. Упродовж останніх десятиліть розпізнавання зображень сформувалося як один із ключових напрямів досліджень у галузях цифрової обробки даних, комп'ютерного зору та штучного інтелекту. Методи автоматичного аналізу візуальних даних широко застосовуються в задачах виявлення та класифікації об'єктів, системах відеоспостереження, автономному керуванні транспортними засобами, а також у медичній діагностиці на основі зображень [1]. Інтенсивний розвиток обчислювальних ресурсів і поява глибоких нейронних мереж сприяли суттєвому підвищенню точності та швидкодії алгоритмів розпізнавання. Водночас, попри значний прогрес, проблема надійного функціонування таких алгоритмів у складних і варіативних реальних умовах залишається відкритою.

Особливої актуальності набуває задача розпізнавання елементів графічного інтерфейсу користувача (Graphical User Interface, GUI) як зображень [2]. Сучасні про-

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

грамні системи, зокрема веб-застосунки, характеризуються високим рівнем складності та динамічності. Архітектура типового веб-застосунку включає кілька взаємопов'язаних рівнів: мову розмітки (HTML або XML), каскадні таблиці стилів (CSS), що визначають візуальне представлення елементів, а також мову програмування JavaScript, яка забезпечує логіку взаємодії та динамічну поведінку інтерфейсу. Така багатокомпонентність зумовлює значну варіативність відображення інтерфейсів навіть за незначних змін у кодовій базі.

Будь-які модифікації структури або стилю веб-сторінки можуть призводити до порушення коректності функціонування інтерфейсу, що, у свою чергу, впливає на досвід користувача та може спричинити втрату клієнтів і зниження прибутків. У цьому контексті автоматизоване тестування користувацького інтерфейсу (UI-тестування) відіграє критично важливу роль у забезпеченні якості програмного забезпечення.

Традиційні підходи до UI-тестування базуються на аналізі DOM-структури документа та використанні селекторів для ідентифікації елементів інтерфейсу. Найбільш поширеними інструментами автоматизації UI-тестування є Selenium WebDriver у поєднанні з тестовими фреймворками, такими як PyTest. Ці засоби дозволяють автоматизувати типові сценарії взаємодії користувача з веб-застосунком, включаючи навігацію між сторінками, введення даних у форми, натискання кнопок та перевірку очікуваних результатів. Однак такі підходи мають низку обмежень, пов'язаних із їхньою залежністю від внутрішньої структури сторінки.

Зокрема, використання XPath- або CSS-селекторів є чутливим до змін у DOM-дереві, це зумовлює необхідність постійного супроводу та оновлення тестових сценаріїв, що підвищує витрати часу і ресурсів. Крім того, традиційні методи не враховують фактичного візуального представлення інтерфейсу, що обмежує їхню здатність виявляти помилки, пов'язані з відображенням елементів.

У зв'язку з цим виникає потреба у розробці альтернативних або доповнювальних підходів до автоматизації UI-тестування, які б базувалися на аналізі візуальної інформації. Одним із перспективних напрямів є застосування методів комп'ютерного зору для розпізнавання елементів інтерфейсу безпосередньо на основі зображень екрану. Такий підхід дозволяє абстрагуватися від внутрішньої реалізації веб-сторінки та орієнтуватися на її кінцеве візуальне представлення, що є більш релевантним з точки зору користувача.

Аналіз останніх досліджень і публікацій. В інтерактивному програмному забезпеченні існують інтерфейси користувача (UI), який використовується для взаємодії із системою. Найпоширенішою формою UI є графічний інтерфейс користувача (GUI) через його візуальну природу, яка дозволяє безпосереднє маніпулювання програмним забезпеченням. У багатьох програмних засобах GUI відіграє важливу роль для спрощення його використання, використовуючи візуальний дизайн та когнітивні аспекти людини, такі як правильне використання кольорів або наскільки людині комфортно працбвати з візуальною структурою. Розробка графічного інтерфейсу користувача, яка не враховує ці два аспекти, може призвести до людських помилок [3].

Одним із важливих факторів, що визначають, чи може користувач легко використовувати графічний інтерфейс є зручність використання. Зручність використання – це атрибут якості, який вимірює, наскільки легко вивчити, наскільки ефективно використовувати або наскільки приємний інтерфейс користувача [4, 5]. UCD (User-Centered Design) це ітеративний процес дизайну, де інтерфейс користувача не повинен розроблятися раз і назавжди, а поетапно та ітеративно. Що стосується графічного інтерфейсу користувача, ітеративний дизайн інтерфейсу користувача може призвести до кращої якості навіть у процесі редизайну [6]. На практиці візуальний аспект графічного інтерфейсу розвивається поетапно [7-9].

Щоб вирішити ці проблеми, попередні роботи, що використовували комп'ютерний зір або штучний інтелект загалом, виконували дії, спрямовані на пришвидшення або сприяння процесу розробки графічного інтерфейсу.

Одна з перших робіт, що використовувала техніку комп'ютерного зору, була виконана Рідлем та Амантом [10] та Гібссом та ін. [11], де вони створили системи під назвою SegMan та Lens відповідно. Обидві системи були призначені для оцінки, хоча й вирішували різні проблеми: SegMan спрямована на автоматичне дослідження інтерфейсу користувача, тоді як Lens більше орієнтована на розуміння інтерфейсу користувача (структура та класифікація) та мала на меті допомогти незрячим користувачам у використанні графічного інтерфейсу. Чанг та ін. [12] створили систему, яка допомагає тестувальникам автоматизувати процес тестування графічного інтерфейсу користувача (етап оцінки). Система під назвою Sikuli Test дозволила тестувальникам писати тестові сценарії на основі зображень компонентів графічного інтерфейсу або навіть генерувати візуальні тестові сценарії після одноразового запуску тесту вручну.

Кох та Оуласвірта [13] виконали роботу з оцінки сприйняття макета інтерфейсу користувача на основі алгоритмічного представлення принципів гештальту для оцінки зручності використання, де інтерфейс користувача оцінювався в цілому. У роботах Лю та ін. [14] побудована система, яка генерує семантичну інформацію про інтерфейси користувача зі скріншотів мобільних додатків. Зі зображень визначається ієрархія переглядів, потім вона сегментується за кольорами і, нарешті, семантично анотується на основі класифікації компонентів графічного інтерфейсу. Система використовувала набір даних Rico [15] та зосереджувалася на розробці методів.

Система, що використовує глибоке навчання, розроблена Фернандесом та Дежею [16], представлено роботу з оцінки зручності використання для автоматичної евристичної оцінки веб-сайтів за допомогою згорткової нейронної мережі (CNN). Вони створили набір даних на основі евристичної оцінки, наданої учасниками для різних скріншотів веб-сайтів.

Лу та ін. [17] збирали набір даних графічного інтерфейсу програмного забезпечення та класифікували їх на позитивні або негативні категорії на основі макета сторінки, зручності та яскравості. Інше використання глибокої згорткової мережі виконали Хассан та ін. [18], створивши систему виявлення компонентів графічного інтерфейсу з зображення макета інтерфейсу задля покращення розробки графічного інтер-

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

фейсу. Нгуєн та ін. [19] запропонували систему глибокого навчання DeepUI, яка використовує рекурентні нейронні мережі (RNN) для вивчення шаблону проектування інтерфейсу та генеративно-змагальну мережу (GAN) для створення візуального дизайну з каркасу.

Найпоширенішим способом автоматичної оцінки зручності використання є метод комп'ютерного зору, як це було зроблено в багатьох згаданих дослідженнях. Одним із поширених методів є виявлення компонентів графічного інтерфейсу всередині інтерфейсу користувача, їх класифікація та подальша обробка. Розпізнавання GUI-елементів як зображень передбачає використання алгоритмів детекції об’єктів, сегментації та класифікації. Сучасні методи, засновані на згорткових нейронних мережах (CNN), демонструють високу ефективність у задачах виявлення об’єктів різної природи. На відміну від природних зображень, GUI-елементи часто мають невеликі розміри, високу щільність розміщення та значну стилістичну варіативність. Крім того, один і той самий функціональний елемент (наприклад, кнопка) може мати різні візуальні представлення залежно від дизайну застосунку. Іншою суттєвою проблемою є відсутність стандартизованих і достатньо великих наборів даних для навчання моделей розпізнавання GUI-елементів. На відміну від загальновідомих датасетів у галузі комп’ютерного зору (таких як ImageNet або COCO), у сфері аналізу інтерфейсів кількість відкритих і добре анотованих наборів даних є обмеженою. Це ускладнює порівняння різних методів і знижує відтворюваність результатів досліджень.

Також слід враховувати динамічний характер сучасних інтерфейсів. Елементи можуть змінювати свій стан у відповідь на дії користувача (наприклад, при наведенні курсора або натисканні), що впливає на їхній зовнішній вигляд. Анімації, адаптивний дизайн та використання різних роздільних здатностей екранів додатково ускладнюють задачу розпізнавання. У таких умовах алгоритми повинні бути стійкими до змін масштабу, освітлення, кольорових схем та інших факторів.

Окрему увагу слід приділити питанню інтеграції методів комп’ютерного зору в існуючі процеси тестування. Використання таких методів як доповнення до традиційних інструментів, зокрема Selenium WebDriver, потребує розробки ефективних механізмів взаємодії між різними рівнями системи. Зокрема, необхідно забезпечити коректне зіставлення результатів візуального розпізнавання з логічними діями тестових сценаріїв.

Мета дослідження. Наукова проблема полягає у розробці та аналізі алгоритмів розпізнавання елементів графічного інтерфейсу як зображень, які б забезпечували високу точність, стійкість до варіацій та ефективність у реальних умовах використання. Необхідно дослідити, яким чином сучасні підходи машинного навчання можуть бути адаптовані до специфіки GUI та інтегровані в процеси автоматизованого тестування.

Метою даного дослідження є аналіз моделей машинного навчання для розпізнавання та класифікації елементів інтерфейсу користувача за для оптимізації процесів тестування веб-додатків.

Викладення основного матеріалу дослідження. Впровадження машинного навчання у сферу тестування веб-додатків приносить значні переваги, допомагаючи авто-

матизувати та покращити процеси тестування. Воно активно впроваджується у різні процеси тестування:

- тестування за допомогою скріншотів - виявлення візуальних відмінностей між сторінками після зміни коду;
- аналіз логів - обробка великих обсягів даних логів для виявлення аномальних подій;
- виявлення елементів – використання AI/ML для ідентифікації певних елементів інтерфейсу користувача на HTML-сторінці замість використання селекторів;
- розробка тестових сценаріїв - використання природної мови або записаних дій користувача для автоматичного створення тестів для програми.

Особливо актуальним є застосування машинного навчання для UI тестування, а саме перевірка, наскільки правильно система відповідає на запити клієнта (користувача).

У більшості випадків перевіряються такі елементи веб-сторінки: кнопки, текст, зображення, поля для введення тексту, чек-бокси. Для кожного елемента повинен бути підготовлений свій тест-план, так як перевірка кожного елемента при UI-тестуванні включає ряд завдань, спрямованих на забезпечення правильної роботи і коректного відображення об'єкта, що тестується, на інтерфейсі програмного додатка або веб-сайту.

Якщо користувач при певних діях очікуючи певний результат отримує помилку або результат, якого він очікував, то тестування було проведено не якісно. Відсутність якісного тестування пов'язана з багатьма факторами: безперервність потоку оновлень у роботі сайту та запуску нових релізів, зміна інтерфейсу користувача; брак часу на створення тестів з повним покриттям, виникнення помилок. Безперервний потік доробок змушує тестувальника безперервно переписувати існуючі автотести, не приділяючи належної уваги створенню нових тест-сценаріїв. Також автотестувальники повинні мати навички розробника, щоб вміти розбиратися не тільки в DOM сайту, а й у різних фреймворках (Рисунок 1).

Саме тому багато компаній відмовляються від запровадження автотестів, залишаючись на ручному тестуванні. Основними проблемами автоматизації UI-тестування для компаній є:

- тривалий час на прогін тестів;
- проблема з інтеграцією різних засобів автоматизації;
- проблема з доступністю середовища;
- проблема із тестовими даними;
- вартість автоматизації та окупність інвестицій;
- проблема стабільності автотестів.

Впровадження комп'ютерного зору автоматизацію UI-тестування має допомогти зняти більшість проблем, описаних вище.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

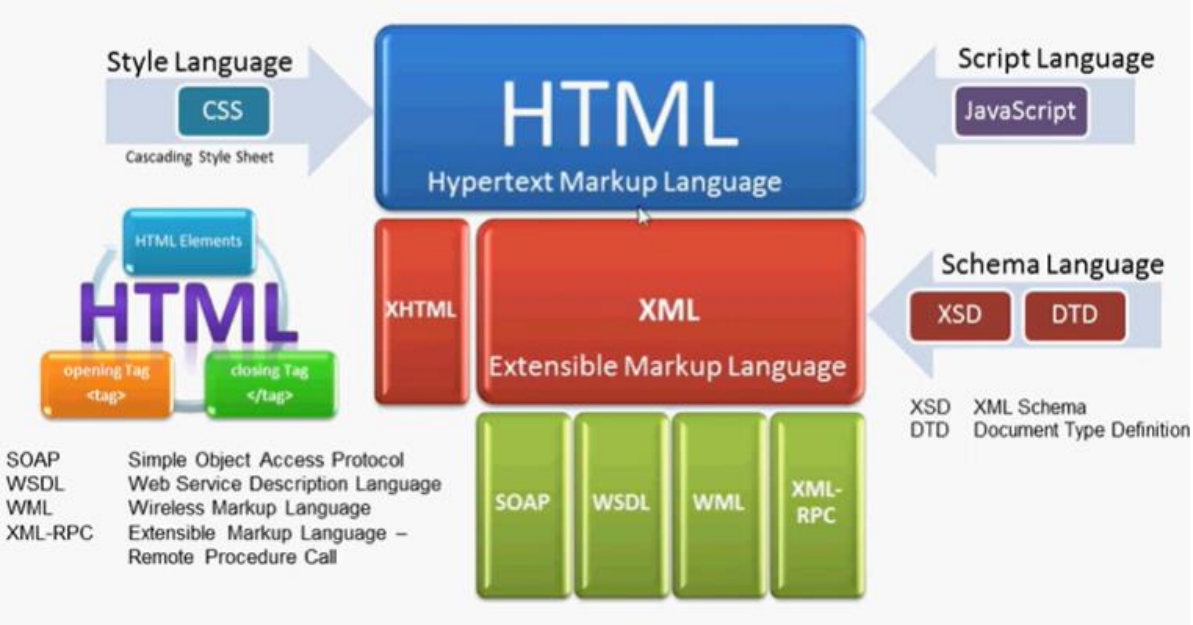

Рисунок 1 – Мови розмітки та програмування веб-сайту [https://www.agilytics.in/post/a-comprehensive-guide-to-markup-languages-and-their-applications]

Процес UI-тестування із застосуванням моделей машинного навчання. Для запропонування оптимальної моделі процесу автоматичного UI-тестування визначено задачу, яка вирішується з використанням алгоритмів машинного навчання і яке місце в бізнес-процесі вона займає. Для цього в роботі проведено порівняльний аналіз поточного бізнес-процесу, що застосовується при автоматизації UI-тестування (AS-IS) та запропонованого бізнес-процесу (TO-BE) (Рисунок 2-3 відповідно).

AS-IS - описує поточний бізнес-процес застосовується в більшості компаній при автоматизації UI-тестування.

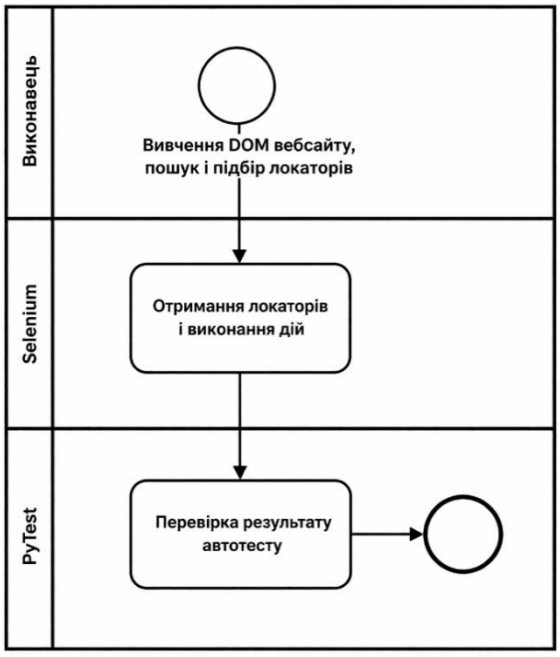

Рисунок 2 - Процес автоматизації UI-тестування AS-IS

У випадку процесу автоматизації UI-тестування за моделлю AS-IS успіх автотесту сильно залежить від уміння тестувальника працювати з DOM і знаходити унікальні елементи, наприклад, якщо в автотесті буде застосовуватися не унікальний елемент, то тест буде виконано з помилками або негативний. Таким чином, автоматизація тестування, що застосовується, повністю не виключає людський фактор. У разі внесення навіть невеликих змін до веб-сторінки, автотести доведеться аналізувати і за необхідності переписувати.

Алгоритм TO-BE визначає порядок дій при повному автоматичному тестуванні, що дозволить навіть при повній зміні інтерфейсу не змінювати автотести і повністю виключити людський фактор.

Для автотестів будуть використовуватись такі інструменти:

- Selenium - інструмент для автоматизації дій із браузером;
- Pytest - інструмент для виконання та перевірки тестових сценаріїв;
- ML - навчена модель машинного навчання.

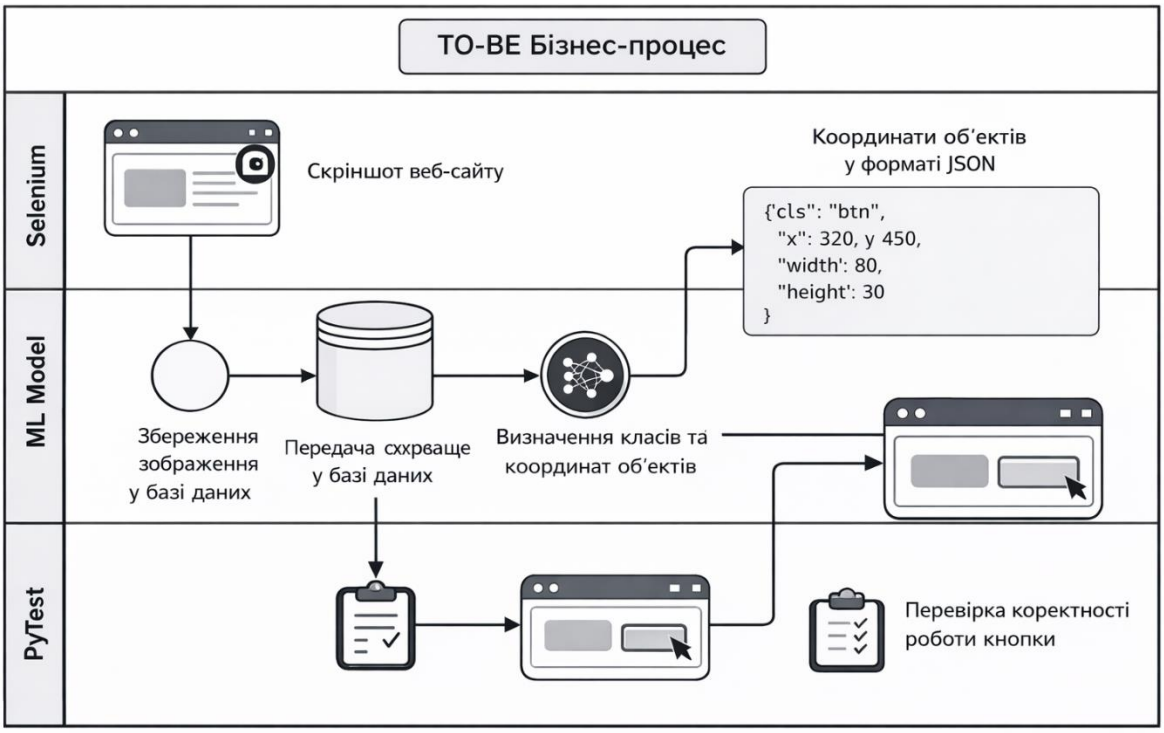

Рисунок 3 – Процес автоматизації UI-тестування TO-BE

У загальному випадку бізнес-процес TO-BE працює в такий спосіб:

1. За допомогою Selenium створюється скріншот веб-сайту, що аналізується, як цифрове зображення.

2. Отриманий зображення зберігається у сховище (базі даних), потім передається в навчену модель машинного навчання.

4. Модель визначає та детектує класи на зображені, координати всіх розпізнаних об'єктів та повертає результат у вигляді json формату із зазначенням центру координат об'єкта – x та y і ширину та висоту розпізнаного об'єкта.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

5. Отримані детектори об’єкта передаються в Selenium та за допомогою Pytest перевіряється коректність роботи кнопки.:

Отримані переваги після переходу на запропонований бізнес-процес:

1. виключення роботу з DOM сайту, що унеможливлює помилки, пов'язані з людським фактором;

2. зниження витрат, пов'язаних з переглядом автотестів у разі зміни структури сайту.

Для успішного використання запропонованного бізнес-процесу необхідна модель машинного навчання, яка могла б розпізнавати та детектувати основні елементи вебсторінок.

Моделі машинного навчання для візуального розпізнавання елементів інтерфейсу користувачів. Для розпізнавання елементів інтерфейсу можна використовувати різні алгоритми. На сьогоднішній день найбільш популярними є Swin-L, EfficientDet-D7, Cascade Mask R-CNN, RetinaNet, YOLO.

Архітектура Swin Transformer Large (Swin-L) [20] належить до класу візуальних трансформерів і базується на ієрархічному представленні ознак із використанням механізму локальної самоуваги (self-attention) у межах зсунутих вікон (shifted windows). Такий підхід забезпечує ефективне моделювання як локальних, так і глобальних залежностей у зображенні, що є критично важливим для аналізу складних графічних інтерфейсів, де просторові відношення між елементами можуть мати семантичне значення. У контексті задачі розпізнавання GUI-елементів Swin-L доцільно розглядати як потужний backbone для систем детекції. Його ієрархічна структура дозволяє ефективно обробляти різномасштабні об’єкти, від дрібних іконок до великих контейнерів. Це особливо важливо для веб-інтерфейсів, де щільність розміщення елементів є високою, а варіативність стилів значною. Крім того, використання attention-механізму дозволяє моделі враховувати контекст розташування елементів, що може підвищити точність класифікації (наприклад, розрізнення кнопки та текстового поля на основі оточення). Це є суттєвою перевагою порівняно з класичними згортковими мережами. Водночас, Swin-L характеризується значною обчислювальною складністю та високими вимогами до пам’яті, що обмежує його застосування в системах реального часу, зокрема в автоматизованому UI-тестуванні. Таким чином, його використання є доцільним переважно в офлайн-аналізі або як частина гібридних архітектур.

EfficientDet-D7 [21] є представником сімейства EfficientDet, яке базується на концепції compound scaling, що передбачає узгоджене масштабування глибини, ширини та роздільної здатності мережі. Центральним компонентом є BiFPN (Bidirectional Feature Pyramid Network), який забезпечує ефективне об’єднання багаторівневих ознак. У задачі розпізнавання GUI-елементів EfficientDet-D7 демонструє високу ефективність завдяки здатності працювати з об’єктами різного масштабу. Це є цінним, оскільки елементи інтерфейсу можуть значно відрізнятися за розмірами. BiFPN дозволяє покращити якість детекції дрібних об’єктів, таких як іконки або чекбокси. Додатковою перевагою є відносно краща ефективність використання обчислювальних ресурсів порівняно з іншими високоточними моделями. Це робить EfficientDet-D7 придатним для практичних

застосувань, де необхідно забезпечити баланс між точністю та швидкодією. Однак, найбільша конфігурація (D7) все ще потребує значних ресурсів, що може ускладнювати інтеграцію в процеси. Крім того, модель менш ефективно враховує глобальний контекст порівняно з трансформерними архітектурами.

Cascade Mask R-CNN [22] є розвитком архітектури Mask R-CNN, що використовує каскадну стратегію для поетапного покращення якості детекції. Кожен наступний етап мережі навчається на більш жорстких порогах IoU, що дозволяє підвищити точність локалізації об'єктів. Ця модель є корисною завдяки можливості сегментації масок, що дозволяє не лише визначити координати об'єкта, але й точно відтворити його форму, що є необхідною умовою для складних інтерфейсів із нестандартними елементами. Каскадна структура забезпечує високу точність навіть у випадках перекриття об'єктів або складного фону. Це актуально для сучасних веб-інтерфейсів, де елементи можуть накладатися або мати складні стилістичні ефекти. Основними обмеженнями є висока обчислювальна складність і низька швидкодія, що унеможливлює використання в реальному часі, та модель є складною в реалізації та налаштуванні.

RetinaNet [23] є моделлю детекції, яка використовує функцію втрат focal loss для подолання проблеми дисбалансу класів. Архітектура включає backbone (наприклад, ResNet) і дві підмережі для класифікації та регресії bounding boxes. Вона є ефективною завдяки здатності працювати з нерівномірно представленими класами. Це важливо, оскільки деякі елементи інтерфейсу (наприклад, кнопки) можуть зустрічатися значно частіше, ніж інші. Модель забезпечує хороший баланс між точністю та швидкодією, що робить її придатною для інтеграції в автоматизовані системи тестування. Водночас відсутність механізмів сегментації обмежує її можливості у випадках, коли необхідна точна форма об'єкта.

YOLO [24] є сімейством одностадійних моделей детекції об'єктів, орієнтованих на обробку в реальному часі. Основна ідея полягає у формулюванні задачі детекції як задачі регресії, що дозволяє виконувати обробку зображення за один прохід мережі. У контексті UI-тестування YOLO є найбільш практичним вибором завдяки високій швидкодії та простоті інтеграції. Модель дозволяє швидко визначати координати елементів інтерфейсу та передавати їх у Selenium для подальшої взаємодії. Попри дещо нижчу точність порівняно з двостадійними моделями, сучасні версії YOLO (наприклад, YOLOv5/YOLOv8) демонструють достатній рівень якості для практичних застосувань. Це робить їх придатними для автоматизованого тестування, де швидкість є критичним фактором.

З проведеного аналізу зроблено висновок, що використання YOLO забезпечує суттєве зменшення обчислювальної складності та затримки обробки, що є важливим у контексті автоматизованого тестування інтерфейсів. У запропонованому бізнес-процесі TO-BE, де передбачається ітеративне виконання тестових сценаріїв із взаємодією користувача, швидкість отримання координат елементів інтерфейсу безпосередньо впливає на загальну продуктивність системи. Це є основною перевагою порівняно з іншими моделями, такими як Swin Transformer або Cascade Mask R-CNN, які, незважаючи на вищу

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

точність, характеризуються значно більшими обчислювальними витратами. Також важливим аспектом є інтеграційна сумісність, YOLO формує результати у вигляді набору обмежувальних рамок (bounding boxes) із відповідними координатами та класами об'єктів. Такий формат природно узгоджується з потребами автоматизованого тестування, де необхідно передавати координати елементів у Selenium для подальшої взаємодії (наприклад, імітації натискання кнопок). У запропонованому алгоритмі результати роботи моделі представлені у форматі JSON, що містить координати центру об'єкта, а також його ширину і висоту. YOLO безпосередньо підтримує подібне представлення, що мінімізує необхідність додаткової обробки даних. Модель може бути донавчена (fine-tuned) на специфічних наборах даних, що містять елементи інтерфейсу, характерні для конкретного веб-застосунку або домену. Це дозволяє підвищити точність розпізнавання без необхідності створення надмірно великих і складних моделей. Умови задачі, де інтерфейси часто мають повторювані патерни, сприяють ефективному перенавчанню моделі.

У даній предметній області критичним є не ідеальне відтворення меж об'єкта, а коректне визначення області, достатньої для взаємодії (наприклад, кліку по кнопці), тому у цьому контексті YOLO забезпечує оптимальний баланс між точністю та ефективністю.

Висновки. У межах дослідження підходів до розпізнавання елементів графічного інтерфейсу користувача (GUI) як цифрових зображень особливу увагу приділено вибору моделі машинного навчання, здатної забезпечити ефективну інтеграцію в процес автоматизованого UI-тестування. З огляду на специфіку предметної області, ключовими вимогами до моделі є точність локалізації об'єктів, стійкість до варіацій візуального представлення інтерфейсів, можливість практичної інтеграції з існуючими інструментами, зокрема Selenium та PyTest. Проведений аналіз сучасних архітектур моделей дозволяє стверджувати, що моделі сімейства YOLO (You Only Look Once) найбільш повно відповідають зазначеним вимогам.

Використання YOLO сприяє усуненню залежності від DOM-структури вебсторінки. Традиційні підходи до UI-тестування базуються на використанні селекторів, які є чутливими до змін у коді. Натомість запропонований підхід тестування вебзастосунків TO-BE дозволяє працювати безпосередньо з візуальним представленням інтерфейсу. YOLO, як ефективний детектор об'єктів, виступає ключовим компонентом цього підходу, забезпечуючи надійне визначення позицій елементів незалежно від їхньої внутрішньої реалізації.

Разом з тим, слід враховувати і певні обмеження YOLO, яка може демонструвати знижену точність при детекції дуже дрібних або щільно розташованих елементів, що характерно для деяких інтерфейсів. Однак ці обмеження можуть бути частково компенсовані за рахунок підвищення роздільної здатності вхідних зображень або використання більш сучасних версій моделі.

ЛІТЕРАТУРА

1. Paschalis Tsirtsakis, Georgios Zacharis, George S. Maraslidis, George F. Fragulis, Deep learning for object recognition: A comprehensive review of models and algorithms, Interna-

tional Journal of Cognitive Computing in Engineering, Volume 6, 2025, Pages 298-312, ISSN 2666-3074, https://doi.org/10.1016/j.ijcce.2025.01.004.
2. Вік.В.Гнатушенко, В.Ю. Царик. Дослідження методів виділення графічних об'єктів на вебсайтах для оцінки якості інтерфейсу / Вік.В.Гнатушенко, В.Ю. Царик // Системні технологій. Регіональний міжвузівський збірник наукових праць. – Випуск 3 (140). – Дніпро, 2022. – С.143-154 DOI 10.34185/1562-9945-3-140-2022-12
3. J. Shariat and C. S. Saucier, Tragic Design: The Impact of Bad Product Design and How to Fix It, First. Sebastopol: O'Reilly Media, 2017.
4. D. Norman and J. Nielsen, "Usability 101: Introduction to Usability," 2012. [Online]. Available: https://www.nngroup.com/articles/usability-101-introduction-to-usability/.
5. D. A. Norman, The Design of Everyday Things. USA: Basic Books, Inc., 2002. J.
6. Nielsen, "Iterative User-Interface Design," Computer (Long. Beach. Calif)., vol. 26, no. 11, doi.org/10.1109/2.241424. 1993, doi.org/10.1109/2.241424.
7. J. Cao, K. Zieba, and M. Ellis, The Ultimate Guide to Prototyping. Mountain View: UXPin Studio, 2015.
8. S. Minhas, "User Experience Design Process," 2018. [Online]. Available: https://uxplanet.org/user-experiencedesign-process-d91df1a45916.
9. C. Murphy, "A Comprehensive Guide To Wireframing And Prototyping," 2018. [Online]. Available: https://www.smashingmagazine.com/2018/03/guide-wireframing-prototyping/#top.
10. M. O. Riedl and R. St Amant, "Toward Automated Exploration of Interactive Systems," 2002.
11. K. Gibbs, T. Winograd, and N. Scott, "Lens: A System for Visual Interpretation of Graphical User Interfaces," 2002.
12. T.-H. Chang, T. Yeh, and R. C. Miller, "GUI Testing Using Computer Vision," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2022.
13. J. Koch and A. Oulasvirta, "Computational layout perception using Gestalt laws," in Conference on Human Factors in Computing Systems - Proceedings, 2016, vol. 07-12-May-2016, pp. 1423–1429, 10.1145/2851581.2892537. doi: 10.1145/2851581.2892537
14. T. F. Liu, M. Craft, J. Situ, E. Yumer, R. Mech, and R. Kumar, "Learning design semantics for mobile apps," in UIST 2018 - Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology, 2018, pp. 569–579, doi: 10.1145/3242587.3242650.
15. B. Deka et al., "Rico: A mobile app dataset for building data-driven design applications," in UIST 2017 - Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology, 2017, pp. 845–854, 10.1145/3126594.3126651.
doi: 10.1145/3126594.3126651
16. R. A. Fernandez, J. A. Deja, and B. P. V. Samson, "Automating heuristic evaluation of websites using convolutional neural networks," in Conference on Human Factors in Computing Systems - Proceedings, 2018, pp. 9–12, doi: 10.1145/3205851.3205854.
17. H. Lu, L. Wang, M. Ye, K. Yan, and Q. Jin, "DNN-based Image Classification for Software GUI Testing," in 2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

Computing, Internet of People and Smart City Innovation (Smart-World/SCALCOM/UIC/ATC/CBDC om/IOP/SCI), 2018, pp. 1818–1823.
18. S. Hassan, M. Arya, U. Bhardwaj, and S. Kole, “Extraction and Classification of User Interface Components from an Image,” Int. J. Pure Appl. Math., vol. 118, no. 24, 2018.
19. T. T. Nguyen, P. M. Vu, H. V. Pham, and T. T. Nguyen, “Deep learning UI design patterns of mobile apps,” in Proceedings - International Conference on Software Engineering, 2018, pp. 65–68, doi: 10.1145/3183399.3183422.
20. Z. Liu et al., “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows,” Proc. IEEE ICCV, 2021. DOI: 10.1109/ICCV48922.2021.00986
21. M. Tan, R. Pang, Q. V. Le, “EfficientDet: Scalable and Efficient Object Detection,” Proc. IEEE CVPR, 2020. DOI: 10.1109/CVPR42600.2020.01039
22. Z. Cai, N. Vasconcelos, “Cascade R-CNN: High Quality Object Detection and Instance Segmentation,” IEEE TPAMI, 2021. DOI: 10.1109/TPAMI.2019.2956516
23. T.-Y. Lin et al., “Focal Loss for Dense Object Detection,” IEEE TPAMI, 2020. DOI: 10.1109/TPAMI.2018.2858826
24. J. Redmon et al., “You Only Look Once: Unified, Real-Time Object Detection,” Proc. IEEE CVPR, 2016. DOI: 10.1109/CVPR.2016.91

REFERENCES

1. Paschalis Tsirtsakis, Georgios Zacharis, George S. Maraslidis, George F. Fragulis, Deep learning for object recognition: A comprehensive review of models and algorithms, International Journal of Cognitive Computing in Engineering, Volume 6, 2025, Pages 298-312, ISSN 2666-3074, https://doi.org/10.1016/j.ijcce.2025.01.004.
2. Vik.V.Hnatushenko, V.Yu. Tsaryk. Research on methods for highlighting graphic objects on websites to assess interface quality/ Vik.V.Hnatushenko, V.Yu. Tsaryk // System technologies. Regional interuniversity collection of scientific papers. – Issue 3 (140). – Dnipro, 2022. – S.143-154 DOI 10.34185/1562-9945-3-140-2022-12
3. J. Shariat and C. S. Saucier, Tragic Design: The Impact of Bad Product Design and How to Fix It, First. Sebastopol: O’Reilly Media, 2017.
4. D. Norman and J. Nielsen, “Usability 101: Introduction to Usability,” 2012. [Online]. Available: https://www.nngroup.com/articles/usability-101-introduction-to-usability/.
5. D. A. Norman, The Design of Everyday Things. USA: Basic Books, Inc., 2002. J.
6. Nielsen, “Iterative User-Interface Design,” Computer (Long. Beach. Calif)., vol. 26, no. 11, doi.org/10.1109/2.241424. 1993, doi.org/10.1109/2.241424.
7. J. Cao, K. Zieba, and M. Ellis, The Ultimate Guide to Prototyping. Mountain View: UXPin Studio, 2015.
8. S. Minhas, “User Experience Design Process,” 2018. [Online]. Available: https://uxplanet.org/user-experiencedesign-process-d91df1a45916.
9. C. Murphy, “A Comprehensive Guide To Wireframing And Prototyping,” 2018. [Online]. Available: https://www.smashingmagazine.com/2018/03/guide-wireframing-prototyping/#top.
10. M. O. Riedl and R. St Amant, “Toward Automated Exploration of Interactive Systems,” 2002.

11. K. Gibbs, T. Winograd, and N. Scott, “Lens: A System for Visual Interpretation of Graphical User Interfaces,” 2002.
12. T.-H. Chang, T. Yeh, and R. C. Miller, “GUI Testing Using Computer Vision,” in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2022.
13. J. Koch and A. Oulasvirta, “Computational layout perception using Gestalt laws,” in Conference on Human Factors in Computing Systems - Proceedings, 2016, vol. 07-12-May-2016, pp. 1423–1429, 10.1145/2851581.2892537. doi: 10.1145/2851581.2892537
14. T. F. Liu, M. Craft, J. Situ, E. Yumer, R. Mech, and R. Kumar, “Learning design semantics for mobile apps,” in UIST 2018 - Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology, 2018, pp. 569–579, doi: 10.1145/3242587.3242650.
15. B. Deka et al., “Rico: A mobile app dataset for building data-driven design applications,” in UIST 2017 - Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology, 2017, pp. 845–854, 10.1145/3126594.3126651. doi: 10.1145/3126594.3126651
16. R. A. Fernandez, J. A. Deja, and B. P. V. Samson, “Automating heuristic evaluation of websites using convolutional neural networks,” in Conference on Human Factors in Computing Systems - Proceedings, 2018, pp. 9–12, doi: 10.1145/3205851.3205854.
17. H. Lu, L. Wang, M. Ye, K. Yan, and Q. Jin, “DNN-based Image Classification for Software GUI Testing,” in 2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDC om/IOP/SCI), 2018, pp. 1818–1823.
18. S. Hassan, M. Arya, U. Bhardwaj, and S. Kole, “Extraction and Classification of User Interface Components from an Image,” Int. J. Pure Appl. Math., vol. 118, no. 24, 2018.
19. T. T. Nguyen, P. M. Vu, H. V. Pham, and T. T. Nguyen, “Deep learning UI design patterns of mobile apps,” in Proceedings - International Conference on Software Engineering, 2018, pp. 65–68, doi: 10.1145/3183399.3183422.
20. Z. Liu et al., “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows,” Proc. IEEE ICCV, 2021. DOI: 10.1109/ICCV48922.2021.00986
21. M. Tan, R. Pang, Q. V. Le, “EfficientDet: Scalable and Efficient Object Detection,” Proc. IEEE CVPR, 2020. DOI: 10.1109/CVPR42600.2020.01039
22. Z. Cai, N. Vasconcelos, “Cascade R-CNN: High Quality Object Detection and Instance Segmentation,” IEEE TPAMI, 2021. DOI: 10.1109/TPAMI.2019.2956516
23. T.-Y. Lin et al., “Focal Loss for Dense Object Detection,” IEEE TPAMI, 2020. DOI: 10.1109/TPAMI.2018.2858826
24. J. Redmon et al., “You Only Look Once: Unified, Real-Time Object Detection,” Proc. IEEE CVPR, 2016. DOI: 10.1109/CVPR.2016.91

Received 20.04.2026.
Accepted 24.04.2026.
Published 30.04.2026

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

An approach to recognizing GUI elements as images based on machine learning

Analysis of Recent Research and Publications. In recent years, computer vision tasks, in particular, object detection and classification in images, have undergone significant development thanks to the use of deep learning methods. The most common approaches are the YOLO, RetinaNet, EfficientDet, Cascade Mask R-CNN, and Swin Transformer models, which demonstrate high efficiency in object recognition tasks of varying complexity. At the same time, the issue of applying these models to recognize graphical user interface (GUI) elements in the context of automated testing of web applications remains insufficiently covered. Traditional approaches to UI testing are based on DOM structure analysis, which makes them vulnerable to changes in layout and complicates test support.

Research Objective. The purpose of this work is to analyze modern methods of recognizing objects in images and justify the choice of an effective machine learning model for the task of recognizing graphical user interface elements.

Presentation of the main research material. The paper proposes an approach to recognizing interface elements based on the analysis of web page screenshots as digital images. The proposed process involves obtaining a screenshot using the Selenium tool, saving the image in storage, and then transferring it to a machine learning model for processing. As a result, the model determines the classes of objects, their coordinates, and sizes, generating output data in JSON format. The obtained coordinates are used to interact with interface elements in an automated testing environment (PyTest).

A comparative analysis of modern object detection models is conducted, in particular Swin-L, EfficientDet-D7, Cascade Mask R-CNN, RetinaNet, and YOLO. Particular attention is paid to models of the YOLO family, which provide high image processing speed while maintaining a sufficient level of accuracy. In the context of the interface element recognition problem, this characteristic is important, since it allows integrating the model into the automated testing process without significantly increasing the test execution time. In addition, YOLO provides a convenient format of output data (bounding boxes), which is directly used for interaction with interface elements.

Conclusions. As a result of the conducted research, it was found that the use of computer vision methods is a promising direction for increasing the efficiency of automated UI testing. The proposed approach allows you to abandon the dependence on the DOM structure, which increases the resistance of tests to interface changes and reduces the costs of their support. Among the considered models, the most appropriate for practical application is YOLO, which provides the optimal balance between speed, accuracy and ease of integration. The results obtained can be used for the further development of intelligent web application testing systems.

Keywords: recognition, image, GUI, model, deep learning, testing.

Царик Владислав Юрійович – старший викладач кафедри інформаційних технологій і систем Українського державного університету науки і технологій.
ORCID: https://orcid.org/0000-0001-6449-8037
Гнатушенко Вікторія Володимирівна – доктор технічних наук, професор, професор кафедри інформаційних технологій і систем Українського державного університету науки і технологій.
ORCID: https://orcid.org/0000-0001-5304-4144
Ольшанський Евген Володимирович – аспірант кафедри інформаційних технологій і систем Українського державного університету науки і технологій.
ORCID: https://orcid.org/0009-0008-7100-673X

Tsaryk Vladyslav – senior lecturer at the Department of Information Technologies and Systems of the Ukrainian State University of Science and Technology.
ORCID: https://orcid.org/0000-0001-6449-8037
Hnatushenko Viktoriia – Doctor of engineering's sciences, Professor, Professor of Department of Information Technologies and Systems, Ukrainian State University of Science and Technology.
ORCID: https://orcid.org/0000-0001-5304-4144
Olshanskyi Evhen – postgraduate student of the Department of Information Technologies and Systems of the Ukrainian State University of Science and Technology.
ORCID: https://orcid.org/0009-0008-7100-673X

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)