УДК 004.93 *http://dx.doi.org/10.30970/vam.2024.34.00000*

МЕТОДИКА ВИКОРИСТАННЯ БІБЛІОТЕК КОМП'ЮТЕРНОГО ЗОРУ ДЛЯ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ АВІАЦІЇ

І. Юзвишин, Ю. Ящук

Львівський національний університет імені Івана Франка,
вул. Університетська 1, Львів, 79000, Україна
e-mail: ivan.yuzvyshyn@lnu.edu.ua, yuriy.yashchuk@lnu.edu.ua

Штучний інтелект (ШІ) поступово перетворюється на ключовий інструмент майбутнього розвитку людства, впливаючи на всі аспекти нашого життя – від економіки та промисловості до науки та побуту. Одним із найважливіших напрямків розвитку ШІ є комп'ютерний зір – галузь, яка дозволяє машинам «бачити» та розуміти візуальну інформацію так само, як це роблять люди. У даній роботі зосереджено увагу на аналізі найбільш поширених сучасних моделей і бібліотек комп'ютерного зору. Зокрема, досліджено кілька готових рішень, такий як YOLO, SSD, та R-CNN, які були протестовані на практичних прикладах без попереднього налаштування. Основна мета полягає у визначенні «підключив і працюй» (від англ. «plug-and-play») підходу, здатного ефективно застосовуватись для задач розпізнавання та класифікації об'єктів без потреби у додатковій конфігурації. Такий підхід дозволяє швидко розгортати моделі та отримувати результати з мінімальними часовими витратами на оптимізацію. Алгоритми протестовано на зображеннях різної якості із використанням середовища Python з метою визначення найоптимальнішого рішення. За результатами експериментів, модель YOLO продемонструвала високу впевненість на зображеннях з одним об'єктом. У той час як SSD показала кращі результати при обробці сцен із кількома об'єктами.

Ключові слова: python, комп'ютерний зір, класифікація об'єктів, виявляння об'єктів, штучний інтелект.

1. Вступ

Комп'ютерний зір базується на складних алгоритмах, які здатні навчатися на великих обсягах даних та поступово покращувати свої результати. Одним із найуспішніших підходів у цій галузі є використання нейронних мереж глибокого навчання, які можуть ефективно обробляти та інтерпретувати зображення. Ці технології знаходять застосування у широкому спектрі галузей – від медицини до автомобільної індустрії, допомагаючи вирішувати завдання від діагностики захворювань до автоматизованого керування транспортом.

Попри значний обсяг наукових досліджень і результатів у цій сфері, шлях від теоретичних ідей до практичного застосування є достатньо тривалим. Це зумовлено кількома факторами, по-перше, складністю алгоритмів і моделей, що потребують значних обчислювальних ресурсів для навчання та оптимізації. По-друге, необхідністю значних масивів якісних даних для навчання моделей, що часто ускладнює процес. По-третє, інтеграція нових технологій у існуючі системи вимагає часу на адаптацію та тестування, щоб забезпечити їхню надійність і ефективність у реальних умовах. Тому проблема вибору правильних інструментів комп'ютерного зору для ефективного збору, точного розпізнавання та обробки зображень з метою виявлення визначених об'єктів без значних витрат часу та ресурсів залишається вкрай актуальною.

Глибоке навчання набуло популярності у 2006 році, коли з'явились перші алгоритми для розпізнавання мовлення – тобто автоматичного перетворення звукової інформації у текстову форму. Це дало поштовх для розвитку й інших напрямків штучного інтелекту, зокрема комп'ютерного зору. Значний внесок у розробку теорії комп'ютерного зору та розпізнавання образів зробили як українські, так і міжнародні вчені, досліджуючи різноманітні підходи до вирішення практичних задач.

Z. Zhao з колегами проаналізували наявні підходи до роботи із зображеннями, такі як виявлення, класифікація та виділення ключових ознак, а також застосування згорткових нейронних мереж (salient object detection) у задачах розпізнавання облич [1]. Значний прорив у цій галузі здійснили T. Yagi, H. Tasnimul, Y. Sato у своїй роботі «Object Instance Identification in Dynamic Environments», де за допомогою комп'ютерного зору і машинного навчання їм вдалось реконструювати тривимірний простір [2]. Схожі дослідження провів W. Liang у роботі «A Fast Defogging Image Recognition Algorithm Based on Bilateral Hybrid Filtering», в якій запропоновано ефективні алгоритми обробки зображень у реальному часі за умов туману або поганої видимості з метою їх подальшого застосування у відеоспостереженні та робототехніці [3].

A. K. Gupta у праці «Salient Object Detection Techniques in Computer Vision» описав підходи до виявлення та локалізації областей зображень, які привертають миттєву зорову увагу людини – сферу інтенсивних досліджень у комп'ютерному зорі [4]. А. Музичук у праці «Про використання кластерної вибірки в алгоритмах візуальної одометрії» розглянув підхід візуальної одометрії для обчислення траєкторії літаючої платформи на основі даних її бортової відеокамери [5]. Лук'янець М. у роботі «Real-Time Data Visualization For IoT Network Systems: Challenges And Strategies For Performance Optimization» досліджував підходи до підвищення продуктивності візуалізації даних у режимі реального часу, зокрема впровадження високопродуктивної обчислювальної інфраструктури, оптимізацію обробки й аналізу даних, а також застосування методів кешування та стиснення даних [6].

JF. Mu у своїй праці «Airplane Recognition and Location On the Airport Based On Computer Vision» представив метод позиціонування, який дозволяє ідентифікувати авіалайнер і визначати його розташування й траєкторію руху відносно інших літаків в аеропорту, що сприяє підвищенню безпеки наземного керування [7]. Також T. Yagi у роботі «Object Instance Identification in Dynamic Environments» досліджував проблеми розпізнавання образів у динамічних середовищах, де об'єкти швидко рухаються або відбувається швидка зміна їхнього розташування [2].

Попри велику кількість публікацій у сфері комп'ютерного зору, більшість із них переважно зосереджені на демонстрації можливостей окремих рішень та підходів. А беручи до уваги той факт, що виробники готових продуктів зазвичай порівнюють свої моделі лише з попередніми версіями власних розробок, це ускладнює об'єктивне оцінювання їхньої ефективності в реальних умовах. Комплексне порівняння різних підходів на єдиному наборі задач залишається рідкістю, хоча саме такий підхід є критично важливим для вибору найбільш придатного інструменту в конкретних сценаріях.

У даній роботі пропонується порівняння доступних алгоритмів на прикладах локалізації літаків із різним рівнем шуму на зображеннях та різною кількістю об'єктів, що стане основою для подальшого вдосконалення. Вибір зображень літаків для дослідження зумовлений їх складною геометрією та різноманіттям форм, що

робить їх ефективними тестовими об'єктами для оцінки можливостей комп'ютерного зору. Літаки мають чіткі контури, складну структуру та велику кількість дрібних деталей, що дозволяє більш точно оцінити продуктивність моделей на реалістичних зображеннях. Крім того, такі об'єкти рідше зустрічаються у відкритих наборах даних, порівняно, наприклад, з автомобілями або пішоходами, тому більшість готових моделей не є повноцінно адаптованими для їх розпізнавання без додаткового навчання або тонкого налаштування.

Для тестування та побудови алгоритмів використано популярну мову програмування високого рівня – Python. Python – це проста для вивчення та гнучка у використанні мова програмування, яка підтримує всі необхідні типи даних для розробки. Її основною перевагою є величезна база наукових бібліотек, що значно спрощує розробку та вдосконалення систем штучного інтелекту та комп'ютерного зору.

2. Огляд використаних у дослідженні інструментів

2.1. Огляд мови програмування Python

Python – це високорівнева, інтерпретована мова програмування (код виконується безпосередньо, без необхідності попередньої компіляції) загального призначення, яка відома своєю простотою і читабельністю коду. Розроблена Гвідо ван Россумом у 1991 році [8].

Python здобув популярність завдяки своїм численним перевагам та широкому спектру застосувань. Простий та чіткий синтаксис робить код легким для читання і розуміння, що у свою чергу знижує бар'єр входу для новачків та підвищує продуктивність досвідчених програмістів.

Варто також згадати, що у Python змінні типізуються динамічно (тип змінної визначається під час виконання програми, а не під час компіляції), що дозволяє гнучко працювати з даними.

Python має велику стандартну бібліотеку, що включає низку корисних модулів для роботи з файлами, мережею, веб-серверами, тестуванням, обробкою тексту. Також існує безліч сторонніх бібліотек і фреймворків для Python, які охоплюють різні області застосування, включаючи веб-розробку (Django, Flask), науку про дані (NumPy, pandas, SciPy), машинне навчання (TensorFlow, PyTorch, scikit-learn), автоматизацію (Selenium, BeautifulSoup), та багато інших.

Python є кросплатформенною мовою, таким чином код, написаний на Python, може виконуватися на різних операційних системах (Windows, macOS, Linux) без змін.

2.2. Огляд бібліотек комп'ютерного зору для розпізнавання об'єктів на базі Python

OpenCV (*Open Source Computer Vision Library*) – бібліотека з відкритим вихідним кодом для комп'ютерного зору та обробки зображень [9]. Вона надає широкий спектр інструментів для аналізу зображень і відео, а також для створення застосунків у таких галузях, як відеоаналітика, машинне навчання та робототехніка.

OpenCV містить безліч алгоритмів для обробки зображень: фільтрація, трансформація, виявлення контурів, розпізнавання об'єктів. Бібліотека підтримує інтеграцію з нейронними мережами через модуль `dnn`, що дозволяє використовувати

попередньо навчені моделі, як-от YOLO, SSD чи Faster R-CNN. Таким чином, OpenCV забезпечує потужні інструменти як для класичних методів, так і для сучасних підходів на основі глибокого навчання.

PyTorch – популярна бібліотека машинного навчання з відкритим кодом, розроблена Facebook AI Research [10]. Вона забезпечує гнучкий і потужний інтерфейс для створення та тренування нейронних мереж. Основною структурою даних у PyTorch є Tensor, що дозволяє ефективно виконувати числові обчислення.

Бібліотека містить модулі: `torch.nn` – для побудови нейронних мереж, `torch.optim` – для реалізації оптимізаторів, torchvision – для обробки зображень і роботи з попередньо навченими моделями (наприклад, SSD, Faster R-CNN). PyTorch широко застосовується як у наукових дослідженнях, так і у виробничих середовищах завдяки своїй гнучкості та зручному API.

TensorFlow – ще одна потужна бібліотека машинного навчання, розроблена Google, що підтримує створення, навчання та розгортання нейронних мереж [11]. TensorFlow включає високорівневий API *Keras* для побудови моделей, а також модулі для роботи з попередньо навченими архітектурами, як-от Inception v3, ResNet, EfficientDet, BERT.

TensorFlow підтримує розгортання на мобільних пристроях через TensorFlow Lite і у виробничих середовищах за допомогою TensorFlow Extended (TFX). Бібліотека активно використовується для задач обробки зображень, природної мови та інших сфер ШІ.

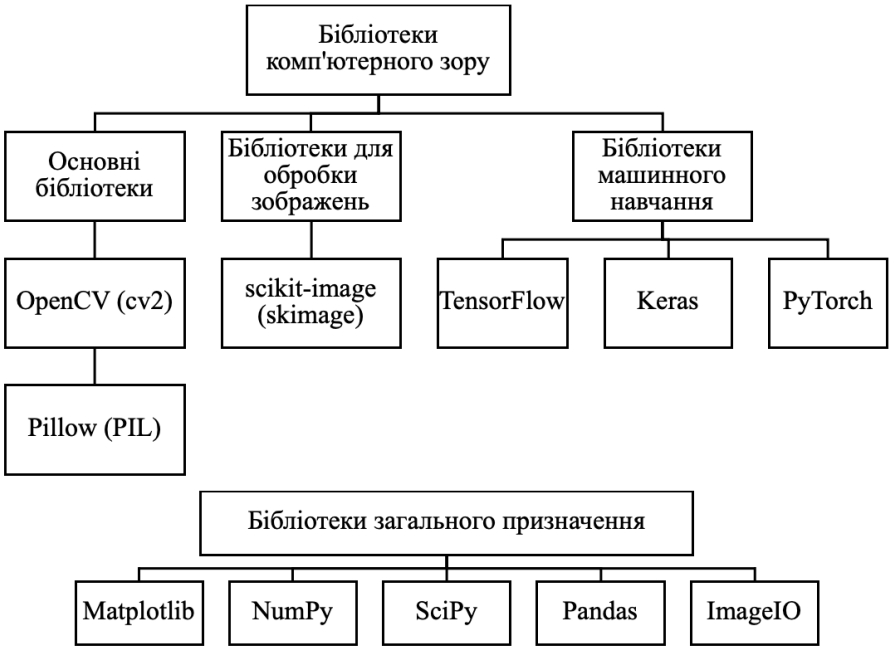

Рис. 1. Структурна схема бібліотек Python, що використовуються для роботи з комп'ютерним зором

2.3. Прискорення навчання моделей комп'ютерного зору

Однією з основних проблем тренування великих нейронних мереж є потреба у проведенні значної кількості обчислень. Порівняно з традиційними CPU, значне пришвидшення можна досягти за допомогою графічних процесорів (GPU). Одним з основних рішень у цій сфері є використання технології CUDA від NVIDIA.

CUDA (*Compute Unified Device Architecture*) – це платформа паралельних обчислень і API, що дозволяє використовувати GPU для загальних обчислень, не пов'язаних з графікою [12]. CUDA дозволяє програмістам писати код на C, C++ та Python, який виконується безпосередньо на GPU, значно пришвидшуючи виконання паралельних задач, таких як обробка зображень, моделювання та машинне навчання.

Для запуску Python-коду на GPU використовується бібліотека CuPy, яка дозволяє легко переносити обчислення з CPU на GPU. Крім того, популярні бібліотеки комп'ютерного зору також мають підтримку CUDA: OpenCV через модулі `opencv_cuda` і `opencv_cudaoptflow`, а TensorFlow і PyTorch – через cuDNN і CUDA Toolkit.

TensorFlow автоматично визначає наявність GPU і використовує його для прискорення обчислень, надаючи інструменти для гнучкого керування ресурсами через `tf.config`. PyTorch також автоматично використовує GPU, дозволяючи переносити тензори між CPU і GPU за допомогою простого методу `.to('cuda')`.

3. Математичні основи алгоритмів комп'ютерного зору

Першим розглянемо підхід до побудови рішення з використанням наявних інструментів комп'ютерного зору. Незалежно від обраної бібліотеки – OpenCV, TensorFlow чи PyTorch, процес попередньої підготовки даних має схожий характер. Після отримання даних для навчання та тестування потрібно вивантажити у пам'ять відповідний формат анотацій та фотографій, у нашому випадку це COCO (Common Objects in Context), який є стандартом у сфері комп'ютерного зору. Його переваги полягають у зручній структурі зберігання анотацій, підтримці різних типів об'єктів та широкій сумісності з популярними бібліотеками. Крім того, доступність відкритих наборів даних у форматі COCO значно спрощує роботу на етапі підготовки даних.

Для кожного зображення за його ідентифікатором завантажується відповідний файл. У випадку з OpenCV це виконується через функцію `cv2.imread`, яка дозволяє зчитувати зображення у вигляді масиву пікселів для подальшої обробки. Аналогічні функції використовуються і в інших фреймворках: наприклад, `tf.io.read_file` у поєднанні з `tf.image.decode_jpeg` для TensorFlow або `PIL.Image.open` у поєднанні з перетворенням у тензор за допомогою `transforms.ToTensor` у PyTorch. Таким чином, базовий підхід до завантаження й обробки зображень є уніфікованим незалежно від обраної платформи.

У первинній обробці необхідно стандартизувати розмір зображення до одного формату, а також, щоб воно відповідало вимогам вхідного розміру моделі. Зазвичай це включає зміну розміру до певного стандартного розміру (до прикладу, 416×416 пікселів для $YOLO$) та нормалізацію піксельних значень.

Розглянемо зображення з розмірами $W \times H$, де W позначає ширину, а H – висоту. Для зміни розміру зображення до стандартних розмірів $W' \times H'$, де W' і H' – це

необхідні розміри (у нашому випадку 416×416 пікселів) проведемо маштабування зображення.

Нехай (x, y) – координати пікселя у вихідному зображенні, а $(x\prime, y\prime)$ координати у зміненому зображенні. Перетворення масштабування можна виразити як:

$$\begin{aligned} x' &= \frac{x \cdot W'}{W} \\ y' &= \frac{y \cdot H'}{H} \end{aligned} \tag{1.1}$$

Це забезпечує збереження пропорцій зображення. Коли аспектне співвідношення зображення не відповідає цільовим розмірам, додається заповнення будь-яких порожніх областей, зазвичай чорним кольором.

І відповідно нормалізація, нехай $I(x, y)$ – значення пікселя в позиції (i,j) вихідного зображення. Нормалізація значень пікселів виконується за допомогою наступної формули:

$$I_{\text{norm}}(x, y) = \frac{I(x, y)}{255} \tag{1.2}$$

де

$I_{\text{norm}}(x, y)$ – нормалізоване значення пікселя. Це перетворення масштабує значення пікселів з діапазону $[0, 255]$ до $[0, 1]$, що полегшує обробку нейронними мережами.

Згодом зображення має бути перетворене у відповідний формат для моделі, а саме – у тензор. Це дозволить моделі ефективно працювати з вхідними даними.

Ядро згортки (фільтр) у згорткових нейронних мережах (CNN) є тензором, який використовується для обробки зображень або інших даних. Ядро виконує операцію згортки (convolution), щоб виявити певні характеристики вхідного зображення, такі як краї, текстури чи інші важливі ознаки.

$$\mathbf{K}_c = \begin{pmatrix} k_{1,1,c} & \cdots & k_{1,n,c} \\ \vdots & \ddots & \vdots \\ k_{n,1,c} & \cdots & k_{n,n,c} \end{pmatrix} \tag{1.3}$$

де

$k(i, j)$ – значення ядра на позиції (i, j),
c – колірний канал. У монохромних зображеннях цей показник відсутній, а для RGB маємо 3 матриці, де c варіюється від 1 до 3.

Для перетворення зображення у формат тензора необхідно організувати дані у форматі, придатному для обробки нейронною мережею. Тензор для зображення з розмірами $W' \times H'$ та кількістю кольорових каналів C (для кольорових зображень RGB ядро зазвичай має 3 канали, що відповідають червоному, зеленому і синьому кольорам) буде 4-вимірним тензором T розміром $[N, C, H', W']$, де N – розмір вибірки зображень (зазвичай $N = 1$ для одного зображення).

Значення пікселя в позиції *(x,y)* для каналу k у тензорі визначається як:

$$T(0, k, x, y) = I_{\text{norm}}(x, y, k) \tag{1.4}$$

Після усіх маніпуляцій з обробкою, підготовлене зображення подається на вхід згорткової нейронної мережі. Модель використовує згорткові шари для автоматичного виділення ознак і виявлення складних патернів у даних.

Згорткові шари є основними елементами CNN, відповідальними за автоматичне витягування особливостей з вхідних даних.

Математично операція згортки може бути виражена наступним чином:

$$S(x,y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} K(m,n) \cdot I(x+m, y+n) \tag{1.5}$$

де

S(x,y) – значення пікселя в результаті згортки на позиції *(x,y)*,
K(m,n) – значення ядра згортки (фільтра) на позиції *(m,n)*,
I(x+m,y+n) – значення пікселя вхідного зображення на позиції *(x+m,y+n)*, M та N – розміри ядра згортки.

Далі використовується активаційна функція, яка перетворює лінійні комбінації входів (вхідні дані) у нелінійні виходи (вихідні дані). Це критично важливо, оскільки без нелінійності нейронна мережа не зможе моделювати складні патерни або вирішувати завдання, що вимагають розпізнавання складних нелінійних зв'язків у даних.

Після того, як нейрон отримує лінійну комбінацію входів (наприклад, output = $W \cdot \text{input} + b$, де W – ваги, input – вхідні дані, b – зсув), активаційна функція перетворює цю лінійну комбінацію в значення, яке є виходом нейрона, і він буде використовуватися як вхід для наступного шару мережі. У нашому випадку ми використаємо функцію Leaky ReLU:

$$f(x) = \begin{cases} x, & x > 0 \\ \alpha x, & x \leq 0 \end{cases} \tag{1.7}$$

де

α – *малий додатній параметр (наприклад, 0.01).*

Як результат – модель поверне ймовірні класи об'єктів, їх координати на зображенні та впевненість у передбаченні. Кожен об'єкт у зображенні описується чотирма координатами: центром (x, y), шириною w і висотою h.

$$\begin{aligned} x_{\text{pred}} &= \sigma(t_x) + c_x \\ y_{\text{pred}} &= \sigma(t_y) + c_y \end{aligned} \tag{1.8}$$

де:

t_x – прогнозовані значення параметрів,
σ – сигмоїдна функція $\sigma(x) = \dfrac{1}{1+e^{-x}}$, яка перетворює відхилення t_x і t_y у значення між 0 та 1,
c_x і c_y – координати верхнього лівого кута якоря розпізнавання об'єкта (попередньо визначена, фіксована форма прямокутника, що використовується для оцінки позиції та розміру об'єкта на зображенні у сітці детекції).

$$\begin{aligned} w_{\text{pred}} &= p_w \cdot \exp(t_w) \\ h_{\text{pred}} &= p_h \cdot \exp(t_h) \end{aligned} \tag{1.9}$$

де

t_w і t_h – прогнозовані значення для ширини і висоти об'єкта,
exp – експоненціальна функція, яка перетворює параметри t_w і t_h у додатні значення,
p_w і p_h – розміри якоря, що використовуються для масштабування.

Параметри t – це виходи нейронної мережі на відповідному шарі, що характеризують зміщення та масштабування відносно якоря. Це дозволяє моделі адаптуватися до варіацій об'єктів у зображеннях. Параметри t обчислюються безпосередньо моделлю як результати згорткових шарів і перетворюються у фінальні координати та розміри об'єкта за допомогою наведених вище функцій.

На етапі пост-обробки необхідно відфільтрувати результати передбачення за порогом впевненості, щоб зосередитися лише на об'єктах, передбачених із високою впевненістю. Завершальним кроком є тестування та візуалізація результатів – окреслення прямокутників на зображенні або відео, що охоплюють виявлені об'єкти. Загальний процес обробки зображення, використаного для навчання та тестування моделі комп'ютерного зору, наведено на рисунку 2.

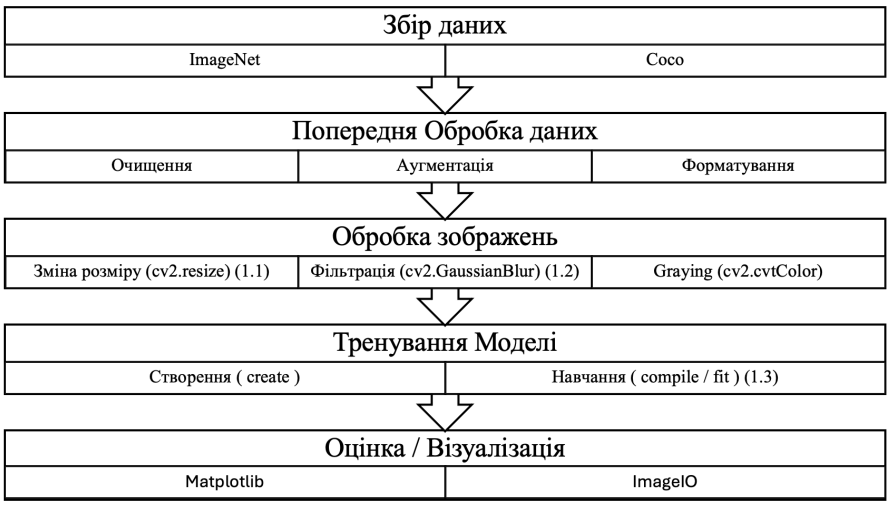

Рис. 2. Структурна схема етапів обробки зображення

4. Результати досліджень

Для ілюстрації застосування процедур визначення заданого об'єкта на зображенні із використанням запропонованих методик обрано базу даних супутникових знімків авіації Kaggle [13] (кількість 1000 шт.), де зображені один або декілька літаків різної чіткості та розмірів (рис. 3-5). Зображення попередньо були очищені та аугментовані з використанням базових методів трансформації, що дозволяє штучно

збільшити обсяг навчального набору та покращити стійкість моделей до варіацій у даних. Серед застосованих технік – зміна яскравості, обертання, горизонтальне віддзеркалення та випадкове масштабування.

Через фокус на тестуванні готових моделей із мінімальними налаштуваннями, використовувались стандартні параметри аугментації, без глибокої оптимізації під конкретний набір зображень, див. Таблиця 1.

Таблиця 1

Застосовані методи аугментації зображень

Метод	**Діапазон / Параметри**
Яскравість	±20%
Горизонтальне віддзеркалення	50% ймовірність
Обертання	$-15°$ до $+15°$
Масштабування	90% – 110%
Обрізання (crop)	До 10% від розміру зображення
Додавання шуму	Легкий гаусів шум

Для експериментів було відібрано обмежену кількість репрезентативних зображень, що дало змогу сфокусуватись на якісному аналізі результатів замість обробки великого обсягу даних. Обрані зображення охоплювали різні сценарії: зміну масштабу, кількість об'єктів та присутність шуму. Опишемо детально два основних сценарії. Сценарій А: зображення з одним літаком та відносно хорошою якістю зображення. Сценарій Б: зображення з декількома літаками та гіршою якістю.

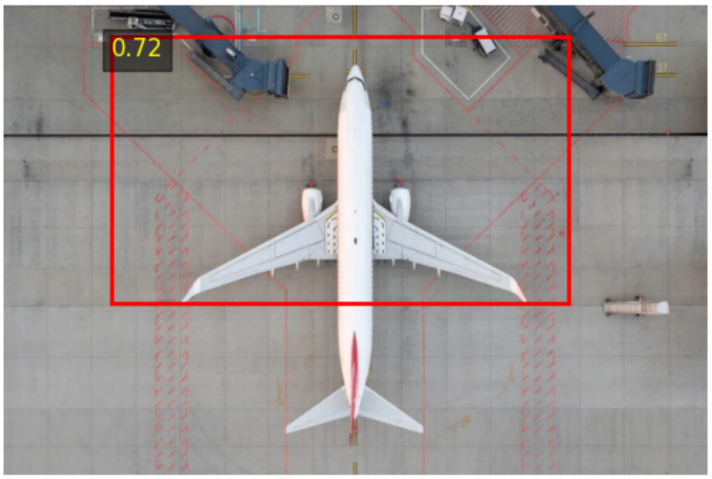

Рис. 3. Результати тестування моделі PyTorch + YOLO

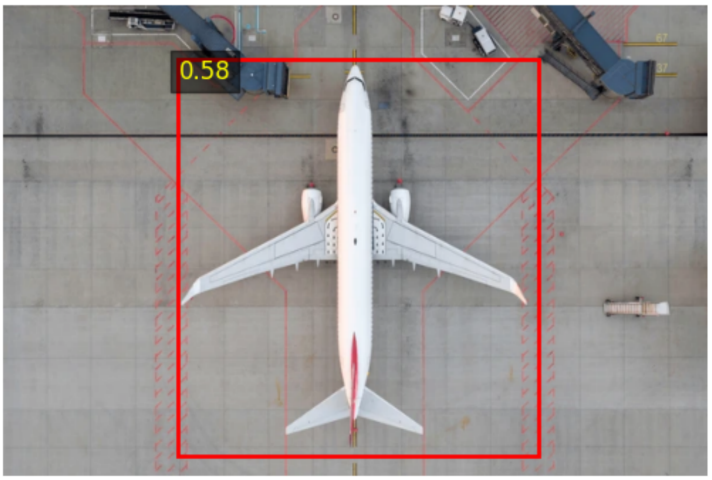

Рис. 4. Результати тестування моделі PyTorch + CNN

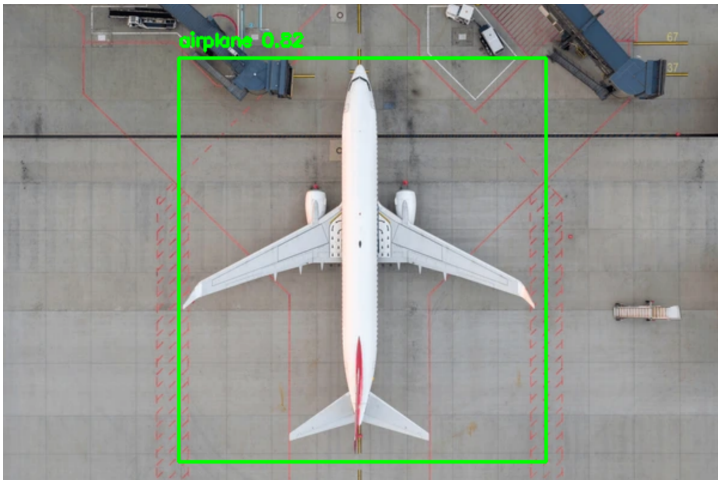

Рис. 5. Результати тестування моделі OpenCV + YOLO

У таблиці 2 наведено загальний результат застосування методів виявлення об'єктів для сценарію А, а у таблиці 3 – результат реалізації методу визначення для сценарію Б.

У цьому дослідженні основним результатом роботи моделі є показник впевненості (confidence score) – числовий показник, який відображає ступінь впевненості моделі в тому, що виявлений прямокутник містить об'єкт певного класу. Значення впевненості варіюється від 0 до 1, де більші значення означають вищу впевненість. Для відсіювання хибних спрацьовувань використовується поріг confidence τ, і лише об'єкти з confidence $\geq \tau$ вважаються коректними передбаченнями.

Таблиця 2

Результати тестування моделей для сценарію А

Бібліотека	**Модель**	**Впевненість**
OpenCV	YOLO	0.82
	SSD	0.79
	R-CNN	0.30
PyTorch	YOLO	0.58
	SSD	0.72
	R-CNN	0.30
TensorFlow	ResNet	–
	YOLO	0.58

Згідно з таблицею результатів виявлення літаків на різних зображеннях, впевненість алгоритмів варіюється залежно від використаної бібліотеки. OpenCV показує найвищу впевненість для моделі YOLO, досягаючи 0.82, а для моделі SSD – 0.79. Однак впевненість моделі R-CNN в OpenCV значно нижча – лише 0.30. У PyTorch результати для YOLO і SSD є нижчими порівняно з OpenCV – 0.58 для YOLO і 0.72 для SSD, тоді як впевненість R-CNN залишається на рівні 0.30, що відповідає результатам OpenCV. TensorFlow продемонстрував впевненість 0.58 для YOLO, подібну до PyTorch. Таким чином, OpenCV виявляється найбільш ефективним для виявлення літаків за допомогою YOLO і SSD, тоді як результати PyTorch і TensorFlow для YOLO є менш вражаючими, а R-CNN в усіх випадках показує низьку впевненість.

Таблиця 3

Результати тестування моделей для сценарію Б

Бібліотека	**Модель**	**Впевненість**
OpenCV	YOLO	Літак 1: 0.56
		Літак 2: 0.54
		Літак 3: –
PyTorch	YOLO	Літак 1: 0.46
		Літак 2: 0.44
		Літак 3: 0.33
	SSD	Літак 1: 0.98
		Літак 2: 0.98
		Літак 3: 0.92
	R-CNN	Літак 1: 0.45
		Літак 2: 0.34
		Літак 3: 0.21
TensorFlow	ResNet	–
	YOLO	Літак 1: 0.46
		Літак 2: 0.44
		Літак 3: 0.33

Аналіз Таблиці 3 дозволяє зробити висновок, що бібліотека OpenCV у поєднанні з моделлю YOLO демонструє помірні результати: впевненість становить 0.56 та 0.54 для перших двох літаків, відповідно. Проте модель не змогла розпізнати третій, останній об'єкт. У середовищі PyTorch модель YOLO показує нижчі результати – 0.46 і 0.44 для перших двох об'єктів та 0.33 для третього. На рис 6–7 наведено візуалізацію результатів.

Рис. 6. Приклад тестування моделей, де присутні більше, ніж один літак з використанням моделі YOLO та PyTorch

Рис. 7. Приклад тестування моделей, де присутні більше, ніж один літак з використанням моделі SSD та PyTorch

Водночас важливо зазначити, що модель змогла виявити всі три об'єкти, попри нижчу впевненість. Модель SSD у PyTorch демонструє значно вищу ефективність: впевненість становить 0.98 для перших двох літаків і 0.92 для третього, що свідчить про її надійність та здатність точно розпізнавати об'єкти. Натомість модель R-CNN у PyTorch показує низькі результати – 0.45, 0.34 і 0.21 відповідно для трьох зображень. У TensorFlow модель YOLO демонструє аналогічну продуктивність до PyTorch YOLO – з впевненість 0.46, 0.44 та 0.33 для відповідних випадків. Таким чином, модель SSD у середовищі PyTorch виявляється найбільш ефективною для задач виявлення літаків, тоді як YOLO в OpenCV і TensorFlow забезпечує подібні, але менш точні результати. На рис. 8 наведено візуалізацію результатів.

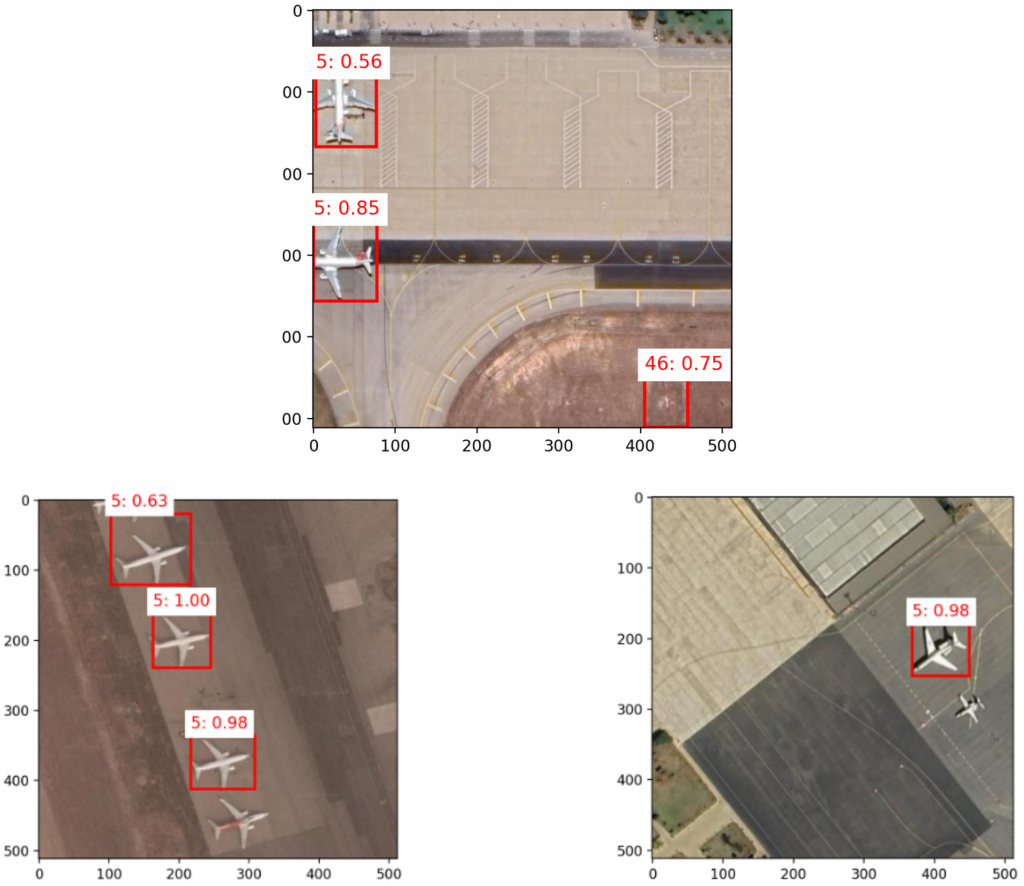

Рис. 8. Приклад тестування моделі TensorFlow на зображеннях, де присутні більше, ніж один літак

5. Висновок

Отже, штучний інтелект і комп'ютерний зір стають основоположними технологіями у розвитку сучасних інновацій, зокрема в таких сферах, як керування та автономна навігація, автоматизація виробництва та інших промислових процесів, а також аналітика великих даних. Завдяки потужним алгоритмам машинного

та глибокого навчання, машини та інші пристрої з комп'ютерним зором можуть виконувати завдання, раніше доступні лише людині, що відкриває нові можливості для підвищення ефективності, точності та безпеки. Важливим аспектом у цьому контексті є чітке визначення об'єктів і завдань, що дозволяє максимально ефективно використовувати ресурси та технології. Для реалізації проектів у галузі комп'ютерного зору широко використовуються різноманітні бібліотеки Python, які забезпечують зручні інструменти для розробки, навчання та впровадження моделей штучного інтелекту. Серед них варто виділити OpenCV – для обробки та аналізу зображень, TensorFlow і Keras – для побудови та навчання нейронних мереж, PyTorch – як потужну платформу для досліджень та експериментів з глибоким навчанням, а також YOLO – для швидкого та ефективного розпізнавання об'єктів у реальному часі. Використання цих бібліотек дозволяє розробникам створювати високотехнологічні рішення, які можуть ефективно виконувати складні завдання в різних сферах застосування – від розпізнавання образів до автономної навігації.

OpenCV та TensorFlow демонструють хороші результати для моделі YOLO, але модель SSD у PyTorch є найбільш ефективною, забезпечуючи найвищу впевненість у виявленні літаків на зображеннях. Модель R-CNN показує найнижчу впевненість в усіх випадках, що вказує на її обмежену ефективність у цьому контексті. Тож для завдань виявлення літаків, особливо в умовах, коли впевненість є критично важливою, використання моделі SSD у PyTorch є найбільш рекомендованим варіантом.

Список використаної літератури

1. Zhao Z.Q. Object Detection With Deep Learning: A Review / Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, Xindong Wu // IEEE Transactions on Neural Networks and Learning Systems. – 2018. – Vol. 30. – P. 3212–3232.
2. Yagi T. Object Instance Identification in Dynamic Environments / Takuma Yagi, Md Tasnimul Hasan, Yoichi Sato // arXiv:2206.05319. – 2022.
3. Liang W. A Fast Defogging Image Recognition Algorithm Based on Bilateral Hybrid Filtering / Wei Liang, Jing Long, Kuan-Ching Li, Jianbo Xu, Nanjun Ma, Xia Lei // ACM Transactions on Multimedia Computing. – Communications, and Applications. – 2021. – Vol. 17, № 2. – P. 1–16.
4. Gupta A.K. Salient Object Detection Techniques in Computer Vision-A Survey / Ashish Kumar Gupta, Ayan Seal, Mukesh Prasad, Pritee Khanna // Entropy. – 2020. – Vol. 22, № 10, – 1174.
5. Іванов С. Про використання кластерної вибірки в алгоритмах візуальної одометрії / С. Іванов, А. Музичук // Вісник Львівського університету. – Серія прикладна математика та інформатика. – 2023. – № 31. – С. 45–55.
6. Lukianets M. Real-Time Data Visualization For IoT Network Systems: Challenges and Strategies For Performance Optimization / Mykhailo Lukianets, Yevgeniya Sulema // System Technologies. – 2024. – Vol. 5, № 148. – P. 52–61.
7. Mu J. Airplane Recognition and Location On the Airport Based On Computer Vision / JianFeng Mu, YanYang Wang // Proceedings of the 2nd International Conference on Electronics. – Network and Computer Engineering. – 2016. – P. 416–422.
8. Official Python Documentation // [https://docs.python.org/3/]. – Accessed: 24.08.2024.
9. Official OpenCV Documentation // [https://docs.opencv.org/4.x/index.html]. – Accessed: 24.08.2024.
10. Official PyTorch Documentation // [https://pytorch.org/docs/stable/index.html]. – Accessed: 24.08.2024.

11. Official TensorFlow Documentation // [https://www.tensorflow.org/]. – Accessed: 24.08.2024.
12. Official CUDA Documentation // [https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html]. – Accessed: 24.08.2024.
13. Kaggle Database // [https://www.kaggle.com/]. – Accessed: 24.08.2024.

Стаття: надійшла до редколегії 03.03.2025
доопрацьована 24.04.2025
прийнята до друку 15.05.2025

METHODS OF USING COMPUTER VISION LIBRARIES FOR RECOGNITION OF AVIATION OBJECTS

I. Yuzvyshyn, Y. Yashchuk

Ivan Franko National University of Lviv,
1, Universytetska str., 79000, Lviv, Ukraine,
e-mail: ivan.yuzvyshyn@lnu.edu.ua, yuriy.yashchuk@lnu.edu.ua

Artificial Intelligence (AI) is progressively becoming a fundamental technology shaping the future development of humanity, influencing every sector including economics, science, healthcare, and everyday life. The seamless integration of AI into diverse domains unlocks new opportunities for innovation, efficiency, and automation. Among the various AI branches, computer vision stands out as a main field that enables machines to interpret, analyze, and understand visual data similarly to human perception. This capability drives numerous applications, ranging from autonomous vehicles and medical imaging to security systems and industrial automation.

This research focuses on an in-depth analysis of the most widely used modern computer vision models and libraries that are ready-to-use with minimal additional configuration, making them accessible for a broad spectrum of practical applications. We explore their efficiency and confidence in object recognition and classification tasks, with a specific emphasis on aviation images. The datasets include images of airplanes captured under varying conditions and qualities, which provide a comprehensive challenge for detection algorithms.

Utilizing the Python as programming environment, we implement and evaluate several state-of-the-art models, including YOLO, SSD, and R-CNN, across popular deep learning frameworks such as TensorFlow, PyTorch, and OpenCV. Performance metrics, including precision and recall, are used to assess each model's ability to accurately detect and classify aviation objects. The results indicate differences in model effectiveness depending on the chosen framework and dataset characteristics.

Our findings aim to assist developers and researchers in selecting the most suitable computer vision solutions for aviation-related applications, highlighting the strengths and limitations of each approach. This work contributes to advancing AI-driven computer vision systems by providing practical insights and recommendations for optimizing object detection performance in real-world scenarios.

Key words: python, computer vision, object detection, object classification, artificial intelligence.