DOI: https://doi.org/10.36910/6775-2524-0560-2026-64-32
УДК: 004.93:004.85
Фітель Богдан Станіславович, аспірант
https://orcid.org/0009-0002-6353-2840
Романишин Юрій Михайлович, д.т.н., професор
https://orcid.org/0000-0003-2794-432X
Національний університет «Львівська політехніка», м. Львів, Україна

ДЕТЕКЦІЯ ОБ'ЄКТІВ НА ОСНОВІ СИНТЕТИЧНИХ ДАНИХ: ПОРІВНЯЛЬНА ОЦІНКА ВПЛИВУ АУГМЕНТАЦІЙ ТА АРХІТЕКТУРИ ДЕТЕКТОРА

Фітель Б. С., Романишин Ю. М. Детекція об'єктів на основі синтетичних даних: порівняльна оцінка впливу аугментацій та архітектури детектора. Робота присвячена дослідженню можливості підвищення якості роботи детекторів об'єктів, навчених в умовах повної відсутності реальних зображень у тренувальному датасеті. Основна мета полягає у визначенні того, чи здатні аугментації, що імітують артефакти реальних камер, скорочувати розрив між синтетичними та реальними даними і покращувати показники детекції на реальних зображеннях. Для формування навчального датасету використано Blender - безкоштовний інструмент тривимірного моделювання з вбудованою підтримкою Python-скриптів, що дозволяє процедурно генерувати фотореалістичні сцени з автоматичною розміткою об'єктів. Датасет формувався у трьох варіаціях: без аугментацій, з аугментаціями на тренувальних і валідаційних зображеннях, а також виключно з аугментаціями валідаційних зображень. До аугментацій належать: шум сенсора, JPEG-артефакти, розмиття руху та розфокусування, хроматична аберація, віньєтування та зсув експозиції. Тренування виконувалось на платформі Google Colab на датасеті з 5000 зображень із використанням бібліотек Ultralytics та Torchvision для декількох архітектур детекторів: YOLOv11, YOLOv11-Seg, Faster R-CNN та Mask R-CNN. Отримані моделі тестувались на трьох незалежних датасетах реальних зображень з ручною розміткою. Найкращий результат на реальних даних - mAP50 0.600 (YOLOv11, test1); застосування аугментацій підвищує mAP50 Faster R-CNN у 4 рази при низькій роздільній здатності (0.118 vs 0.506 на test3). Результати демонструють вплив вибору аугментаційної стратегії та архітектури детектора на якість детекції в умовах domain gap між синтетичними та реальними даними.
Ключові слова: синтетичні дані, детекція об'єктів, YOLOv11, YOLOv11-Seg, Faster R-CNN, Mask R-CNN, Blender, автоматична анотація, комп'ютерний зір, domain gap.

Fitel B., Romanyshyn Y. Object detection based on synthetic data: comparative evaluation of augmentation and detector architecture impact. This paper investigates the possibility of improving the performance of object detectors trained in conditions of complete absence of real images in the training dataset. The primary objective is to determine whether augmentations that simulate real camera artifacts can reduce the domain gap between synthetic and real data and improve detection metrics on real-world images. Blender, a free three-dimensional modeling tool with built-in Python scripting support, was used to generate the training dataset. The dataset was produced in three variants: without augmentations, with augmentations applied to both training and validation images, and with augmentations applied to validation images only. The augmentations include sensor noise, JPEG compression artifacts, motion blur and defocus, chromatic aberration, vignetting, and exposure shift. Training was performed on Google Colab using a dataset of 5,000 images with the Ultralytics and Torchvision libraries across multiple detector architectures: YOLOv11, YOLOv11-Seg, Faster R-CNN, and Mask R-CNN. The resulting models were evaluated on three independent real-image datasets with manual annotations. The best result on real data is mAP50 0.600 (YOLOv11, test1); augmentations improve Faster R-CNN mAP50 by a factor of 4 under low resolution (0.118 vs 0.506 on test3). The results demonstrate the impact of augmentation strategy and detector architecture on detection quality under domain gap conditions between synthetic and real data.
Keywords: synthetic data, object detection, YOLOv11, YOLOv11-Seg, Faster R-CNN, Mask R-CNN, Blender, automatic annotation, computer vision, domain gap.

Вступ. Сучасний світ переживає стрімкий розвиток технологій штучного інтелекту, що охоплює практично всі сфери науки та промисловості. Разом із прогресом алгоритмів і моделей машинного навчання невпинно вдосконалюється і апаратне забезпечення, на якому виконуються обчислювально інтенсивні процеси навчання нейронних мереж. Графічні процесори (GPU) нового покоління демонструють багатократне зростання продуктивності порівняно з попередніми поколіннями, надаючи дослідникам та інженерам можливість тренувати дедалі складніші моделі за прийнятний час і з меншими витратами ресурсів.

Паралельно з розвитком обчислювальних потужностей активно прогресує напрям комп'ютерного зору, зокрема системи автоматичного виявлення та класифікації об'єктів на зображеннях - так звані детектори зображень. Сучасні архітектури детекторів, такі як YOLO (You Only Look Once) та Faster R-CNN (Region-based Convolutional Neural Network), демонструють високу точність і швидкість обробки, що робить їх незамінними інструментами у задачах відеоспостереження, промислового контролю якості, медичної діагностики та автономного водіння.

Проте ефективне навчання детекторів зображень вимагає наявності великих обсягів якісно розмічених навчальних даних. Кожне зображення в датасеті має бути забезпечене точними анотаціями – обмежувальними рамками (bounding boxes) та відповідними мітками класів для

кожного об'єкта. Формування таких датасетів традиційно потребує або значних затрат ручної праці досвідчених розмічувачів, або залучення спеціалізованих платних сервісів і моделей штучного інтелекту для автоматичної розмітки, що у свою чергу породжує суттєві фінансові та часові витрати і не завжди гарантує необхідну якість розмітки.

За таких умов виникає закономірна потреба в автоматизації процесу формування навчальних датасетів із мінімальними людськими та фінансовими затратами [1]. Одним із перспективних підходів є використання виключно синтетичних даних - зображень і відповідних анотацій, згенерованих за допомогою програмних засобів тривимірного моделювання та рендерингу. Пакет Blender, як потужний безкоштовний інструмент з відкритим вихідним кодом, надає широкі можливості для процедурної генерації фотореалістичних сцен із автоматичним формуванням розмітки, що робить його ідеальним кандидатом для вирішення цієї задачі.

Метою роботи є дослідження того, наскільки детектори об'єктів, навчені виключно на синтетичних даних без жодного реального зображення, здатні узагальнюватись на реальний домен, та як на це узагальнення впливають аугментації, що імітують сенсорні артефакти реальних камер. Для досягнення мети вирішуються такі завдання: генерація автоматично розміченого синтетичного датасету у Blender у трьох аугментаційних варіантах; тренування чотирьох архітектур детекторів (YOLOv11, YOLOv11-Seg, Faster R-CNN, Mask R-CNN); порівняльне тестування отриманих моделей на трьох незалежних реальних датасетах із різним ступенем доменного розриву.

Огляд технологій

Детектори об'єктів

YOLOv11 - однопрохідний anchor-free детектор, що локалізує та класифікує об'єкти за єдиний прохід зображення через мережу [2]. Завдяки цьому він суттєво швидший за двоетапні архітектури - залежно від конфігурації від 30 до 100+ FPS. На синтетичному датасеті ця перевага проявляється особливо виразно: чисті, рівномірно освітлені сцени без сильних артефактів добре відповідають умовам, за яких однопрохідний підхід впевнено знаходить об'єкти вже на ранніх епохах. Відсутність необхідності підбирати anchor-параметри під конкретний датасет спрощує навчання та робить модель менш чутливою до специфіки синтетичної геометрії.

YOLOv11-Seg розширює YOLOv11 до instance segmentation: замість прямокутної рамки модель передбачає попіксельний контур кожного окремого об'єкта [2]. Ключова перевага у контексті синтетичних даних полягає в тому, що маски в Blender генеруються автоматично з emission-рендерів без будь-яких додаткових затрат на анотацію - тобто сегментаційний сигнал дістається безкоштовно. Це дозволяє задіяти маскову гілку як додатковий регуляризатор під час навчання, що покращує точність локалізації bounding box порівняно з чистим детектором.

Faster R-CNN - двоетапний детектор: Region Proposal Network спочатку висуває регіони-кандидати, а потім окремий модуль класифікує та уточнює кожен із них. Двоетапний підхід дає вищу точність локалізації на складних і перекритих об'єктах, але ціна - 2-10 FPS. На синтетичному датасеті Faster R-CNN отримує важливу перевагу: backbone ResNet-50 з FPN [3] ініціалізується вагами ImageNet, що забезпечує потужне базове представлення ознак навіть при обмеженій кількості навчальних прикладів. У поєднанні з детально розрахованими bounding box-анотаціями з Blender це дозволяє двоетапній моделі надійно навчатись навіть на сценах зі значним перекриттям фігур.

Mask R-CNN додає до Faster R-CNN маскову гілку: після RoIAlign кожен регіон отримує бінарну маску 28×28 пікселів, обчислену незалежно для кожного класу [4]. Порівняно з прототипним підходом YOLOv11-Seg це забезпечує точніші межі об'єктів, особливо складної форми - наприклад, шахових фігур із характерними силуетами. Анотації у форматі COCO з бінарними масками генеруються з Blender так само автоматично, тому додаткова складність маскової гілки не потребує ручного втручання. Основний компроміс залишається той самий, що й у Faster R-CNN: значно менша швидкість інференсу, що робить модель придатною передусім для офлайн-оцінки якості детекції.

Blender як інструмент генерації синтетичних даних

Blender - це відкрита кросплатформна система для 3D-моделювання, анімації та рендерингу з ліцензією GNU GPL. Програма підтримує два фізично коректні рендерери: Cycles (path tracing з підтримкою GPU через CUDA, OptiX і Metal) та EEVEE (real-time рендер на основі растеризації). Саме поєднання фотореалістичного рендерингу Cycles із повноцінним Python API робить Blender

ідеальним середовищем для процедурної генерації навчальних датасетів: будь-яка дія в інтерфейсі доступна зі скрипту, включно з розміщенням об'єктів, налаштуванням камери, керуванням матеріалами та безпосередньо запуском рендеру. Це дозволяє автоматизувати виробництво тисяч анотованих зображень без жодного ручного втручання.

Ключовою перевагою підходу є те, що разом із рендером автоматично генерується і розмітка: координати bounding box для кожного об'єкта у кадрі обчислюються програмно через проєкцію 3D-вершин об'єкта на площину камери за допомогою вбудованої функції world_to_camera_view. Таким чином, виключається будь-яка ручна анотація, а точність розмітки визначається лише числовою точністю обчислень, а не суб'єктивним судженням розмічувача. Генератор підтримує одночасний експорт у формати YOLO (TXT), Pascal VOC (XML) та COCO, забезпечуючи сумісність із широким спектром навчальних фреймворків.

Огляд суміжних робіт

Використання синтетичних даних для навчання детекторів об'єктів - активно досліджуваний напрям. Ранні роботи демонстрували, що моделі, навчені лише на рендерингах, вже здатні переносити знання на реальні сцени, хоча й зі значним зниженням точності [1]. Широке поширення набули підходи на основі ігрових рушіїв: UnrealCV [5] надає Python-інтерфейс до Unreal Engine для генерації фотореалістичних зображень з автоматичними depth і сегментаційними масками; CARLA використовує той самий рушій для автономного водіння і є одним із найбільш цитованих прикладів sim-to-real трансферу в задачах детекції об'єктів. Ці системи дають дуже якісний рендеринг, але вимагають ліцензованого та дорогого програмного забезпечення і не передбачають гнучкого сценарного контролю поза вбудованим API.

Blender як основа для генерації датасетів використовується у ряді досліджень. BlenderProc [6] - спеціалізований Python-фреймворк поверх Blender, орієнтований на генерацію сцен для 6DoF pose estimation та детекції з HDRI-освітленням і фізичним розміщенням об'єктів. Tang і Jia [7] систематично дослідили корисність синтетичних даних Blender для навчання і довели, що в умовах контрольованого IID-розподілу синтетика добре слугує основою для адаптації до реального домену. На відміну від цих підходів, у цьому дослідженні Blender використовується не тільки для генерації зображень, а й для одночасного формування трьох форматів розмітки (YOLO, Pascal VOC, COCO) і побудови трьох незалежних аугментаційних варіантів датасету з одного рендерингового пайплайну - що дозволяє коректно порівнювати різні конфігурації без повторного рендерингу.

Питання аугментацій для скорочення sim-to-real розриву розглядається у [8], де показано, що моделювання ефектів сенсора (хроматична аберація, шум, розмиття) ефективно зменшує доменний розрив для детекції у міських сценах. Мумуні А. та Мумуні Ф. [1] систематизують широкий спектр методів синтетичної аугментації, від domain randomization до нейронного рендерингу. Однак жодна з розглянутих робіт не порівнює систематично вплив аугментацій на різних архітектурах однопрохідних і двоетапних детекторів у рамках єдиного повністю синтетичного пайплайну - чим і мотивоване це дослідження.

Domain gap та аугментації

Поняття domain gap (доменного розриву) описує невідповідність між розподілами навчальних і тестових даних, що призводить до деградації якості моделі у реальних умовах [9]. У контексті синтетичних даних цей розрив формується кількома незалежними факторами. Текстурний розрив виникає через відмінність поверхонь 3D-моделей від реальних матеріалів із природними дефектами і нерівностями. Освітлювальний розрив - через спрощення фізики освітлення у рендерері порівняно з реальними джерелами. Сенсорний розрив - через відсутність у рендерингових зображеннях артефактів реальних камер: шуму матриці, аберацій оптики, блюру, JPEG-компресії.

Одним із найпростіших і водночас ефективних способів скорочення доменного розриву є domain randomization [9]: під час тренування свідомо вносяться широкі варіації текстур, освітлення, фонів, щоб реальні дані опинились у межах навченого розподілу. Загальний огляд технік аугментації зображень для глибокого навчання, включно з геометричними перетвореннями, колірними операціями та фільтрами, наведено в [10]. Альтернативний підхід - фотореалістична аугментація, яка моделює конкретні артефакти реальних камер [8]. Саме ця стратегія застосована у цій роботі: замість широкої рандомізації всіх параметрів сцени аугментації цілеспрямовано імітують сенсорні ефекти, що дозволяє перевірити, чи достатньо лише візуальної подібності для покращення узагальнення.

Методологія

Генерація синтетичного датасету

Для генерації синтетичного датасету розроблено Python-скрипт, що виконується безпосередньо у середовищі Blender. Скрипт процедурно будує кожен кадр із нуля: вибирає підмножину 3D-моделей шахових фігур, розміщує їх у просторі, налаштовує камеру, освітлення та оточення, виконує рендер і паралельно формує анотації. Весь цикл повторюється для кожного зображення без жодного ручного втручання.

Генератор автоматично сканує всі mesh-об'єкти у сцені Blender і розпізнає шахові фігури за іменем. Це дозволяє завантажувати довільну кількість 3D-наборів: для даного датасету завантажено 4 різних набори шахових фігур, кожна фігура якого отримує той самий class_id незалежно від стилістики 3D об'єкту. Підтримується також ручне призначення аліасів. Таким чином, один і той самий клас представлений різноманітними геометричними варіаціями, що підвищує узагальнювальну здатність детектора.

Фігури розміщуються у вільному просторі на умовній площині, а фоном слугують HDRI-панорами. Це спрощує сцену, надаючи вигляд реального середовища з об'єктами на фоні без затрат на рендеринг геометрії, прибирає одноманітний текстурований фон і змушує модель навчатись розпізнавати фігури краще. На кожному кадрі обирається випадкова підмножина від 4 до 12 фігур (з усіх доступних у сцені). Кут повороту навколо вертикальної осі рандомізується від 0° до 360°, масштаб варіюється від 60% до 150%. З 50% ймовірністю фігура отримує нахил навколо горизонтальних осей (±180°), імітуючи перекинутий або лежачий стан. З 15% ймовірністю генерується порожній кадр (negative sample) без жодної фігури - для зниження false positive спрацювань.

Усі mesh-об'єкти сцени, що не розпізнані як шахові фігури, автоматично вважаються дистракторами. У сцену можна завантажити будь-які 3D-моделі: наприклад, предмети темного кольору, які перекривають або знаходяться позаду чорних шахових фігур, змушуючи модель навчатись розрізняти їх від фону та схожих за кольором об'єктів. На кожному кадрі випадково обирається від 0 до 6 дистракторів, кожен отримує рандомну позицію в радіусі 30 одиниць від центру сцени, випадковий оберт та масштаб (30-120% від оригінального). Дистрактори не потрапляють до анотацій.

Камера обертається навколо центру сцени за сферичними координатами: відстань 25-55 одиниць, кут підйому 20°-70°, азимут 0°-360°. Поле зору FOV рандомізується від 38° до 55°, крен камери - ±8°. Точка спостереження завжди направлена на центр сцени з невеликим випадковим зміщенням (±1.5 одиниці), що унеможливлює однотипні кадри.

На кожному кадрі випадково обирається один із 10 пресетів освітлення: daylight, golden_hour, overcast, dramatic, cool_studio, warm_interior, night, colored, backlit, top_down. Три джерела світла (основне Sun, Fill та Rim Area) отримують рандомні значення інтенсивності (0.2-5.0), кольорової температури (2200-9500 K) та позиції. HDRI-карта - 360°-панорама у форматі .hdr або .exr - обирається випадково з директорії на кожному кадрі та слугує одночасно фоном і фізично коректним джерелом освітлення у рендерері Cycles. Горизонтальна ротація HDRI рандомізується від 0° до 360°, інтенсивність - в діапазоні 0.15-1.2.

Генератор паралельно формує три варіанти датасету в трьох окремих директоріях: clean - без будь-яких постефектів; effects_val_only - постефекти застосовуються лише до валідаційних зображень; effects_train_val - постефекти на обох підвибірках. Аугментації включають шум сенсора, JPEG-артефакти, розмиття руху або розфокусування, хроматичну аберацію, віньєтування, зсув експозиції та балансу білого [8]. Такий підхід дозволяє оцінити вплив аугментацій окремо на тренувальну і валідаційну підвибірки без повторного рендерингу (Рис.1).

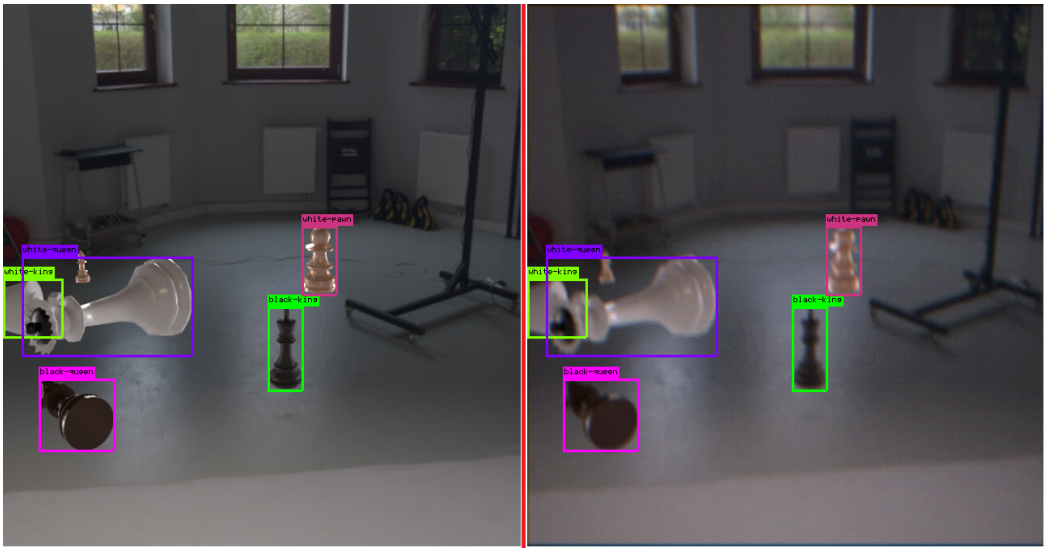

Рис. 1. Демонстрація формування розміток у вигляді bounding box для зображення без та з аугментаціями

Кожна фігура-кандидат проходить перевірку через visible mask gate. Для неї виконуються два швидкі emission-рендери у режимі EEVEE з одним семплом: амодальна маска (фігура ізольована, всі інші меші приховані) та видима маска (фігура серед реальної сцени з усіма оклюзіями). Фігура включається до анотацій лише якщо видимих пікселів ≥ 120 і вони складають ≥ 30% від амодального силуету. Bounding box обчислюється безпосередньо з видимої маски, що гарантує точну відповідність рамки реально видимій частині об'єкта (Рис.2).

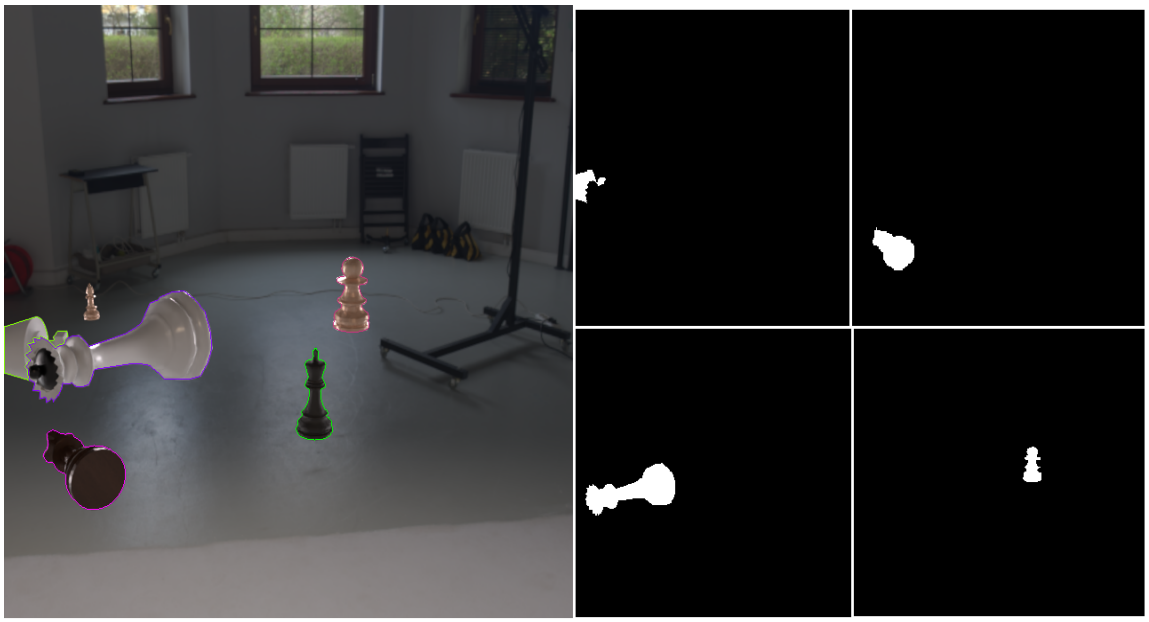

Рис. 2. Формування розмітки для сегментаційних моделей і маски, за допомогою яких вони формуються

Для YOLOv11 та YOLOv11-Seg формуються txt-файли у форматі YOLO, для Faster R-CNN - Pascal VOC XML, для Mask R-CNN - бінарні instance-маски з JSONL-записами, які далі конвертуються у COCO-формат окремим скриптом. Датасет автоматично розбивається на train (80%) та val (20%).

Тестові датасети

Для оцінки узагальнювальної здатності моделей використано три незалежні реальні датасети, зібрані окремо від навчальних даних. Усі три набори розмічені за допомогою

сегментаційного інструменту Roboflow і містять 12 класів шахових фігур, що відповідають навчальному датасету. Розмітка виконувалась вручну для кожного зображення.

Test1 - власноруч зібраний датасет із 400 зображень. Основна мета при зборі - максимальна різноманітність: зображення зроблені з різними фігурами, на різних фонах, при різному освітленні та з різних ракурсів. Акцент зроблено на зйомці окремих фігур на нейтральних фонах, що максимально наближає умови до синтетичних сцен (об'єкти без шахової дошки). Якість зображень навмисно варіювалась - від чітких знімків до дещо розмитих. Саме тому test1 демонструє найвищі показники mAP серед трьох наборів.

Test2 - 2000 зображень, завантажених з Roboflow. Зображення зроблені переважно з одного ракурсу (вид на шахову дошку ззверху або під кутом під час гри), роздільна здатність - переважно 1920×1080 пікселів. Умови зйомки наближені до реального використання: фігури стоять на дошці, часто перекриваються, є задній план. Цей набір представляє типову сцену «шахова партія», що відрізняється від синтетичного пайплайну як постановкою, так і текстурами.

Test3 - 2000 зображень, також завантажених з Roboflow, аналогічних за сценарієм до test2, але з суттєво нижчою роздільною здатністю: 480×288 пікселів. Такий масштаб відповідає умовам потокового відео або камер низького класу. Низька роздільна здатність є основним джерелом додаткового доменного розриву для YOLO-моделей, яким при тестуванні подаються зображення, масштабовані до 640×640 - що вносить артефакти інтерполяції. Двоетапні моделі, що обробляють зображення без фіксованої зміни розміру, зберігають стабільнішу продуктивність при цьому переході.

Налаштування тренування

Тренування всіх моделей виконувалось на платформі Google Colab із GPU-прискоренням. Нижче наведено зведену таблицю гіперпараметрів та конфігурацій для кожної архітектури.

Таблиця 5. Гіперпараметри тренування моделей

Параметр	**Faster R-CNN (ResNet50-FPN)**	**Mask R-CNN (ResNet50-FPN)**	**YOLOv11 (detection)**	**YOLOv11-Seg**
Фреймворк	PyTorch (torchvision)	PyTorch (torchvision)	Ultralytics YOLOv11	Ultralytics YOLOv11-seg
Batch size	16	16	32	32
Початкова learning rate	0.01	0.005	авто (Ultralytics)*	авто (Ultralytics)*
Оптимізатор	SGD (momentum=0.937, wd=5e-4)	SGD (momentum=0.937, wd=5e-4)	AdamW (за замовчуванням)	AdamW (за замовчуванням)
Кількість епох	20	15	60	60
Розмір зображення	оригінальний	оригінальний	640×640	640×640
Аугментації тренування	немає (стандартне масштабування)	немає (стандартне масштабування)	degrees=5°, mosaic=1.0, mixup=0.1, copy_paste=0.1, hsv=0.0	degrees=5°, mosaic=1.0, mixup=0.1, copy_paste=0.1, hsv=0.0
Patience (рання зупинка)	не використовується	не використовується	7 епох	7 епох
Resume тренування	так	так	ні	ні
Кількість класів	12	12	12	12
Розподіл train/val	4000 / 1000	4000 / 1000	4000 / 1000	4000 / 1000

Параметр	Faster R-CNN (ResNet50-FPN)	Mask R-CNN (ResNet50-FPN)	YOLOv11 (detection)	YOLOv11-Seg
Формат анотацій	Pascal VOC (XML)	COCO JSON	YOLO (TXT)	YOLO-seg (TXT+masks)
Seed	42	42	42	42
Пристрій	GPU (cuda)	GPU (cuda)	GPU (cuda)	GPU (cuda)

*У YOLOv11/YOLOv11-Seg learning rate керується внутрішнім scheduler (починається з 0.01 для AdamW і зменшується за косинусним законом). Параметр lr0 не задавався явно.

Для всіх трьох конфігурацій датасету (clean / train_val / val) гіперпараметри тренування залишались незмінними. Варіювався лише сам набір даних - наявність або відсутність спотворень у тренувальній та валідаційній вибірках. Параметри кольору в YOLO-аугментаціях (hsv_h, hsv_s, hsv_v) свідомо вимкнені до нуля: колір є критичною ознакою для розрізнення білих і чорних шахових фігур, тому кольорові спотворення на рівні фреймворку недоречні.

Метрики оцінки якості детекції

Для оцінки використовується стандартний набір метрик детекції об'єктів. mAP@0.5 (mean Average Precision при IoU threshold 0.5) відображає середню точність по всіх класах при відносно м'якому критерії перекриття: передбачений bounding box вважається правильним, якщо IoU з ground truth ≥ 0.5. mAP@0.5:0.95 обчислюється аналогічно, але усереднюється по діапазону порогів IoU від 0.5 до 0.95 з кроком 0.05 і є суворішою метрикою, що штрафує за неточну локалізацію. Precision показує частку правильних передбачень серед усіх передбачень моделі: P = TP / (TP + FP). Recall - частку знайдених об'єктів серед усіх реальних: R = TP / (TP + FN). F1 є гармонійним середнім Precision і Recall і дозволяє оцінити баланс між ними в одному числі. Поріг впевненості (threshold) при тестуванні встановлено 0.25 для всіх архітектур, IoU threshold для NMS - 0.45.

Результати та аналіз

Зведена таблиця характеристик моделей

На основі логів тренувань (синтетичний датасет, 5000 зображень, 4000 train / 1000 val) отримано наступні результати:

Таблиця 1. Результати тренування моделей на синтетичному датасеті

Модель	Варіант аугментацій*	Найкращий box mAP@0.5	Епоха	Фінальний mAP (остання епоха)	Час 1 епохи (сек)	Фінальний train loss	Фінальний val loss
Faster R-CNN (ResNet50-FPN)	clean (немає)	0.858	7	0.843	~180	0.080	0.238
	train_val (обидві)	0.803	11	0.794	~180	0.097	0.296
	val (лише val)	0.670	9	0.611	~180	0.070	0.581
Mask R-CNN (ResNet50-FPN)	clean	0.862	11	0.850	~220	0.131	0.295
	train_val	0.808	13	0.804	~220	0.138	0.376
	val	0.657	9	0.610	~220	0.140	0.749
YOLOv11 (detection)	clean	0.944	49	0.932	~40	0.354	0.233 (val/box)
	train_val	0.928	49	0.915	~40	0.405	0.305 (val/box)

Модель	Варіант аугментацій*	Найкращий box mAP@0.5	Епоха	Фінальний mAP (остання епоха)	Час 1 епохи (сек)	Фінальний train loss	Фінальний val loss
	val	0.850	48	0.847	~40	0.354	0.526 (val/box)
YOLOv11-Seg (detection box)	clean	0.952	45	0.943	~60	0.245 (train/box)	0.200 (val/box)
	train_val	0.925	46	0.922	~60	0.302	0.268 (val/box)
	val	0.815	24	0.800	~60	0.283	0.569 (val/box)

*clean - аугментації не застосовувались (синтетичні зображення без спотворень)
*train_val - аугментації (розмиття, шуми, спотворення) застосовані і до тренувальної, і до валідаційної вибірки
*val - аугментації застосовані тільки до валідаційної вибірки (тренування на чистій синтетиці)

Метрики на синтетичній валідації виглядають обнадійливо: mAP@0.5 для YOLO-моделей сягає 0.944-0.952, а для двоетапних - 0.858-0.862 на чистому датасеті. Однак ці цифри відображають якість роботи в межах того самого синтетичного розподілу і не прогнозують поведінку на реальних зображеннях. Показовою є конфігурація val_only - коли аугментації застосовані лише до валідаційної вибірки, а тренування відбувалось на чистій синтетиці: навіть у межах синтетичного домену mAP падає на 20-25% відносно clean, наочно ілюструючи ефект доменного розриву ще до переходу на реальні дані. Реальна картина розкривається в тестуванні на трьох незалежних наборах.

Тестування на реальних зображеннях

Тестування проводилось на трьох незалежних реальних наборах, описаних у підрозділі 3.2. Для кожного набору всі 12 навчених конфігурацій (4 архітектури × 3 варіанти датасету) оцінювались за однаковим протоколом при конфіденс threshold 0.25 та NMS IoU 0.45.

Тестовий набір test1 (висока схожість із синтетичним доменом)

Таблиця 2. Результати тестування на наборі test1

Архітектура	Конфігурація	mAP50 (box)	mAP50-95 (box)	Precision	Recall	F1
YOLOv11	clean	0.600	0.518	0.672	0.558	0.610
	train_val	0.549	0.469	0.623	0.555	0.587
	val_only	0.547	0.486	0.664	0.524	0.586
YOLOv11-Seg	clean	0.541	0.492	0.605	0.546	0.574
	train_val	0.597	0.545	0.675	0.575	0.621
	val_only	0.503	0.447	0.594	0.484	0.533
Faster R-CNN	clean	0.318	0.256	0.253	0.462	0.327
	train_val	0.346	0.275	0.279	0.491	0.356
	val	0.273	0.226	0.244	0.408	0.305
Mask R-CNN	clean	0.304	0.250	0.302	0.464	0.366
	train_val	0.358	0.292	0.317	0.515	0.393
	val_only	0.297	0.245	0.294	0.455	0.357

Набір test1 є найбільш наближеним до синтетичного тренувального домену: висока роздільна здатність, подібне освітлення, відсутність значних шумів і артефактів стиснення. За таких умов доменний розрив мінімальний, і всі моделі, включно з навченими на чистих синтетичних

даних, демонструють відносно хорошу якість детекції. Найвищий показник mAP50 досягає YOLOv11-clean (0.600), проте YOLOv11-Seg із аугментаціями на тренуванні практично не поступається (0.597) і додатково надає сегментаційну маску. Двоетапні моделі очікувано поступаються однопрохідним аналогам: найкращий результат серед них у Mask R-CNN train_val (0.358), що майже вдвічі нижче за лідера. Загалом test1 підтверджує, що за близьких до синтетики умов аугментації не є обов'язковими, хоча й не погіршують результат.

Тестовий набір test2 (помірний доменний розрив)

Таблиця 3. Результати тестування на наборі test2

Архітектура	Конфігурація	mAP50 (box)	mAP50-95 (box)	Precision	Recall	F1
YOLOv11	clean	0.474	0.322	0.554	0.463	0.505
	train_val	0.557	0.390	0.596	0.529	0.560
	val_only	0.447	0.313	0.572	0.426	0.488
YOLOv11-Seg	clean	0.358	0.268	0.433	0.345	0.384
	train_val	0.518	0.392	0.553	0.459	0.501
	val_only	0.422	0.306	0.455	0.377	0.412
Faster R-CNN	clean	0.418	0.260	0.481	0.386	0.428
	train_val	0.428	0.271	0.525	0.445	0.481
	val	0.395	0.268	0.469	0.325	0.384
Mask R-CNN	clean	0.373	0.252	0.476	0.376	0.420
	train_val	0.367	0.260	0.515	0.383	0.439
	val_only	0.372	0.258	0.479	0.368	0.416

Набір test2 представляє помірний рівень відмінності від синтетики: присутні варіації фону, тіней, балансу білого та незначні шуми. Доменний розрив вже достатній, щоб проявити перевагу аугментацій. У всіх без винятку архітектур конфігурація train_val демонструє вищі показники, ніж чиста модель. Найкращий абсолютний результат - YOLOv11 train_val (mAP50 0.557, F1 0.560). Сегментаційна версія YOLOv11-Seg train_val також отримує значний приріст від аугментацій (+0.160 mAP50 відносно clean), досягаючи 0.518. Двоетапні моделі показали близькі результати в усіх конфігураціях (0.37-0.43), суттєво не змінюючись від аугментацій. Test2 наочно демонструє, що для однопрохідних YOLO-архітектур аугментації є критичним інструментом підвищення стійкості, тоді як двоетапні детектори менш чутливі до помірного розриву.

Тестовий набір test3 (низька роздільна здатність, екстремальний доменний розрив)

Таблиця 4. Результати тестування на наборі test3

Архітектура	Конфігурація	mAP50 (box)	mAP50-95 (box)	Precision	Recall	F1
YOLOv11	clean	0.343	0.247	0.424	0.348	0.382
	train_val	0.335	0.230	0.421	0.367	0.392
	val_only	0.346	0.251	0.447	0.341	0.387
YOLOv11-Seg	clean	0.320	0.242	0.417	0.303	0.351
	train_val	0.323	0.247	0.431	0.302	0.355
	val_only	0.298	0.214	0.438	0.272	0.335
Faster R-CNN	clean	0.118	0.084	0.526	0.130	0.208
	train_val	0.506	0.333	0.582	0.539	0.560
	val	0.122	0.089	0.518	0.112	0.183
Mask R-CNN	clean	0.159	0.113	0.567	0.173	0.265
	train_val	0.437	0.287	0.572	0.510	0.539
	val_only	0.147	0.105	0.548	0.156	0.243

Набір test3 є найскладнішим випробуванням: роздільна здатність зображень (480×288) суттєво нижча за тренувальну (640×640), що створює екстремальний доменний розрив. Усі моделі без аугментацій (clean), а також усі однопрохідні YOLO-версії навіть з аугментаціями виявились практично непрацездатними: mAP50 не перевищує 0.35, recall подекуди падає нижче 0.15. Натомість Faster R-CNN train_val демонструє виняткову стійкість - mAP50 0.506, F1 0.560 - і є єдиною моделлю, яка зберігає прийнятну якість за такого падіння роздільності. Друге місце посідає Mask R-CNN train_val (0.437), також суттєво випереджаючи YOLO-моделі. Такий результат пояснюється двоетапною архітектурою та RoIAlign, що краще зберігає просторову інформацію при низькій роздільності. Test3 є вирішальним доказом: для сценаріїв із потенційно низькою якістю вхідних зображень першочергове значення має не аугментаційна стратегія, а вибір архітектури - і безальтернативним вибором є Faster R-CNN з аугментаціями.

Висновки та перспективи подальшого дослідження

Дослідження підтвердило принципову можливість отримання детекторів, здатних працювати на реальних зображеннях, без єдиного реального навчального прикладу - лише на основі автоматично розміченого синтетичного датасету з Blender. Нижче сформульовано основні висновки.

Аугментації є обов'язковою умовою для реального застосування. На всіх тестових наборах конфігурація train_val перевершує або порівняння з clean. Для двоетапних моделей за значного доменного розриву (test3) покращення сягає 4 разів. Навчання лише на чистій синтетиці без аугментацій сенсорних ефектів не дає узагальнення поза синтетичним доменом.

Вибір архітектури залежить від умов. YOLOv11 (detect) забезпечує найкраще співвідношення якості та швидкості на наборах із помірним доменним розривом (test1, test2). При низькій роздільній здатності (test3) безальтернативним лідером є Faster R-CNN (train_val), тоді як YOLO-моделі суттєво поступаються.

Сегментаційні моделі не покращують детекцію боксів. YOLOv11-Seg та Mask R-CNN не перевищують bbox mAP50 своїх суто детекційних аналогів. Їх варто використовувати тільки якщо задача вимагає попіксельної маски.

Перспективи. Серед напрямків подальшого розвитку варто виокремити такі. По-перше, перенесення пайплайну на інші предметні області (авіація, промислові деталі, медичні інструменти, роботизовані маніпулятори) - шахи є зручним прикладом через фіксований клас і характерні силуети, але підхід є загальним. По-друге, дослідження методів domain adaptation - fine-tuning на невеликій кількості реальних зображень поверх синтетично-претренованих ваг. По-третє, збільшення обсягу і варіативності синтетичного датасету: більша кількість 3D-наборів фігур, різноманітніші матеріали та HDRI-панорами. Перспективним є також дослідження впливу якості 3D-моделей (low-poly vs high-poly) на якість детекції та оцінка, наскільки спрощена геометрія впливає на transfer до реального домену.

Обмеження дослідження. По-перше, пайплайн перевірено виключно на фіксованому наборі об'єктів - шахових фігурах - з обмеженою варіативністю 3D-моделей (4 набори). Результати можуть не переноситись на класи з більшою міжекземплярною варіативністю. По-друге, параметри аугментацій (рівень шуму, сила розмиття тощо) підібрані евристично і не оптимізувались - систематичний підбір міг би покращити результати. По-третє, абсолютні значення mAP50 на всіх трьох реальних датасетах залишаються відносно низькими (0.1-0.6), що відображає реальні межі методу за умови малої кількості варіацій навчальних зразків. По-четверте, Ultralytics і torchvision використовують різні внутрішні реалізації NMS, метрик та аугментацій, тому пряме числове порівняння між групами YOLO та R-CNN слід трактувати з обережністю. Нарешті, використання Google Colab унеможливлює точний контроль апаратного забезпечення між запусками, що може вносити варіативність у час тренування, хоча не впливає на якість моделей.

Список бібліографічного опису

1. Мумуні А., Мумуні Ф., Геррар Н. К. Огляд методів аугментації синтетичних даних у комп'ютерному зорі. Machine Intelligence Research. 2024. https://doi.org/10.1007/s11633-022-1411-7
2. Редмон Дж., Дівала С., Гіршік Р., Фархаді А. You Only Look Once: уніфікована детекція об'єктів у реальному часі. Матеріали IEEE конференції CVPR. 2016. С. 779–788. https://doi.org/10.1109/CVPR.2016.91
3. Рен Ш., Хе К., Гіршік Р., Сунь Дж. Faster R-CNN: до детекції об'єктів у реальному часі за допомогою мереж пропозиції регіонів. Advances in Neural Information Processing Systems. 2015. Т. 28. С. 91–99. https://doi.org/10.48550/arXiv.1506.01497

4. Хе К., Гкіоксарі Г., Доллар П., Гіршік Р. Mask R-CNN. Матеріали IEEE ICCV. Венеція, 2017. С. 2980–2988. https://doi.org/10.1109/ICCV.2017.322
5. Цю В., Юіл А. UnrealCV: підключення комп'ютерного зору до Unreal Engine. Матеріали ECCV 2016 Workshops. LNCS. Т. 9915. С. 909–916. https://doi.org/10.1007/978-3-319-49409-8_75
6. Деннінгер М., Зундермайєр М., Вінкельбауер Д., Зідан Ю., Олефір Д. та ін. BlenderProc: процедурний Blender-пайплайн для фотореалістичного рендерингу. arXiv:1911.01911. 2019. https://doi.org/10.48550/arXiv.1911.01911
7. Танг Г., Цзя К. Новий бенчмарк: про корисність синтетичних даних Blender для навчання з учителем та адаптації до цільового домену. Матеріали IEEE/CVF конференції CVPR. 2023. С. 15954–15964. https://doi.org/10.1109/CVPR52729.2023.01531
8. Карлсон А., Скіннер К. А., Васудеван Р., Джонсон-Робертсон М. Sensor Transfer: оптимальна аугментація зображень ефектами сенсора для адаптації sim-to-real. IEEE Robotics and Automation Letters. 2019. Т. 4, № 3. С. 2431–2438. https://doi.org/10.1109/LRA.2019.2896470
9. Тобін Дж., Фонг Р., Рей А., Шнайдер Дж., Заремба В., Еббіл П. Domain Randomization для перенесення глибоких нейронних мереж із симуляції у реальний світ. Матеріали IEEE/RSJ IROS. 2017. С. 23–30. https://doi.org/10.1109/IROS.2017.8202133
10. Шортен К., Хошгофтаар Т. М. Огляд методів аугментації зображень для глибокого навчання. Journal of Big Data. 2019. Т. 6, № 1. С. 60. https://doi.org/10.1186/s40537-019-0197-0

References

1. Mumuni A., Mumuni F., Gerrar N. K. A survey of synthetic data augmentation methods in computer vision. Machine Intelligence Research. 2024. https://doi.org/10.1007/s11633-022-1411-7
2. Redmon J., Divvala S., Girshick R., Farhadi A. You Only Look Once: Unified, Real-Time Object Detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016. P. 779–788. https://doi.org/10.1109/CVPR.2016.91
3. Ren S., He K., Girshick R., Sun J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. Advances in Neural Information Processing Systems. 2015. Vol. 28. P. 91–99. https://doi.org/10.48550/arXiv.1506.01497
4. He K., Gkioxari G., Dollár P., Girshick R. Mask R-CNN. Proceedings of the IEEE International Conference on Computer Vision (ICCV). Venice, 2017. P. 2980–2988. https://doi.org/10.1109/ICCV.2017.322
5. Qiu W., Yuille A. UnrealCV: Connecting Computer Vision to Unreal Engine. Computer Vision – ECCV 2016 Workshops. LNCS. Vol. 9915. P. 909–916. https://doi.org/10.1007/978-3-319-49409-8_75
6. Denninger M., Sundermeyer M., Winkelbauer D., Zidan Y., Olefir D. et al. BlenderProc: A Procedural Blender Pipeline for Photorealistic Training Image Generation. arXiv:1911.01911. 2019. https://doi.org/10.48550/arXiv.1911.01911
7. Tang H., Jia K. A New Benchmark: On the Utility of Synthetic Data with Blender for Bare Supervised Learning and Downstream Domain Adaptation. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2023. P. 15954–15964. https://doi.org/10.1109/CVPR52729.2023.01531
8. Carlson A., Skinner K. A., Vasudevan R., Johnson-Roberson M. Sensor Transfer: Learning Optimal Sensor Effect Image Augmentation for Sim-to-Real Domain Adaptation. IEEE Robotics and Automation Letters. 2019. Vol. 4, No. 3. P. 2431–2438. https://doi.org/10.1109/LRA.2019.2896470
9. Tobin J., Fong R., Ray A., Schneider J., Zaremba W., Abbeel P. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. Proceedings of IEEE/RSJ IROS. 2017. P. 23–30. https://doi.org/10.1109/IROS.2017.8202133
10. Shorten C., Khoshgoftaar T. M. A Survey on Image Data Augmentation for Deep Learning. Journal of Big Data. 2019. Vol. 6, No. 1. P. 60. https://doi.org/10.1186/s40537-019-0197-0

Історія статті:
Отримано: 02.06.2026 Доопрацьовано: 12.09.2026 Прийнято до друку: 23.09.2026 Опубліковано: 29.09.2026