DOI: https://doi.org/10.15276/ict.02.2025.66
УДК 52-004

Дослідження впливу алгоритмів оптимізації на точність нейронних мереж Yolov11

Ковбаса Сергій Миколайович[1)]
Д-р техніч. наук, доцент, зав. каф. Автоматизації електромеханічних систем та електроприводу
ORCID: https://orcid.org/0000-0002-2954-455X; skovbasa@ukr.net. Scopus Author ID: 55328200100
Холоша Антон Олексійович[1)]
Аспірант кафедри Автоматизації електромеханічних систем та електроприводу
ORCID: https://orcid.org/0009-0003-4858-202X; kholosha.anton@gmail.com
[1)] Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»,
пр. Берестейський, 37, Київ, Україна

ABSTRACT

Візуальна інспекція по результатах детекції зображень є інтенсивно зростаючою складовою систем автоматизації. Машинний зір все ширше використовується у виробничих технологічних лініях різного призначення. Підвищення точності розпізнавання в таких застосуваннях може бути нелегкою задачею, особливо в умовах можливих обмежень, одним з яких може бути обмеження за розміром та вагою, що в свою чергу обмежує потужність комп'ютерних пристроїв, що реалізують детектування та розпізнавання зображень. Можливим рішенням цієї проблеми є підвищення точності розпізнавання за допомогою збільшення варіантів зображень в датасеті та точному налаштуванню гіперпараметрів моделі. У даній статті досліджується ефективність налаштування гіперпараметрів для моделі детекції зображень YOLO (You Only Look Once) котра в подальшому може бути імплементована в системах з обмеженням потужності.

Keywords: штучний інтелект; комп'ютерний зір; гіперпараметри; алгоритми оптимізації; YOLOv11; Roboflow; точність; IoU

Метою роботи є дослідження ефективності налаштування гіперпараметрів моделі детекції YOLO для підвищення точності розпізнавання зображень в умовах можливих апаратних обмежень.

Протягом останніх років виробничі системи все частіше модернізуються в сторону використання комп'ютерного зору, а саме object detection and classification [1]. Багато систем реалізуються в портативному виконанні, що є головною причиною проблеми з корисним простором, що в свою чергу напряму впливає на обчислювальні потужності для комп'ютерного зору. Збільшення точності розпізнавання об'єктів без збільшення розміру моделі детекції (МД) або коштовного збільшення обчислювальних ресурсів це є важливим та актуальним завданням на сьогодні, так як системи покладаються на обмежені апаратні ресурси і збільшення розміру МД призведе до збільшення потреб ресурсів для детекції об'єктів в режимі реального часу. Одним з рішень для покращення детекції є оптимізація існуючої моделі, що є предметом постійних досліджень [2–4]. Досить розповсюдженим підходом до рішення цього завдання є налаштування гіперпараметрів МД, що дозволяє підвищити продуктивність без збільшення її розміру.

Налаштування гіперпараметрів це довготривалий процес по ручному, або автоматичному підборі параметрів, які керують навчанням моделі штучного інтелекту. Для моделей розпізнавання об'єктів, таких як YOLO (You Only Look Once), до початкових гіперпараметрів які можна налаштовувати можна віднести розмір батчів, функції втрат, розмір зображення, та кількість епох навчання, що значно впливає на точність розпізнавання в подальшому [5, 6]. Цей підхід особливо корисний в бюджетних системах, або в системах котрі не можуть бути модернізовані.

Згорткові нейронні мережі (CNN) стали основою сучасних алгоритмів виявлення об'єктів, включаючи YOLO, завдяки їх здатності автоматично навчатися на основі вхідних даних з розміченими ознаками [7, 8]. CNN працюють шляхом застосування згорткових шарів, які діють як фільтри для виявлення локальних ознак, а потім шляхом об'єднання шарів, які зменшують розмірність даних, зберігаючи при цьому найбільш вагомі ознаки. Даний процес

вилучення дозволяє згортковим нейронним мережам справлятися із складними завданнями генерації, класифікації та розпізнавання об'єктів на зображеннях.

Однією з найвідоміших і як результат наймасштабніших моделей по завданням пов’язаних з детекцією об’єктів є вище згадана нейромережа YOLO. Нейромережі розпізнавання образів поділяють на моделі котрі здійснюють розпізнавання в декілька етапів та ті котрі здійснюють детекцію за один прогон зображення. Нейромережа YOLO розглядає виявлення об'єктів як єдине завдання регресії. Через те, що YOLO обробляє зображення за один прохід, вона забезпечує найбільшу швидкодію серед представлених на ринку нейромереж і, в той же час, постійно розвивається. Наприклад, в YOLOv3 було нововведення багатомасштабного виявлення та обмежувальних рамок, що дозволило моделі виявляти об'єкти різних розмірів у співвідношенні сторін [9]. У версії YOLOv11 було ще більше поліпшено модель розпізнавання, впровадивши збільшення обсягу даних і покращено розрахунок функції втрат, що зробило її високоточною і ефективною моделлю для виявлення об'єктів в реальному часі [10].

YOLO має п’ять основних перед тренованих моделей котрі відрізняються швидкістю, кількістю параметрів, потребам по обчислювальним спроможностям, вихідною точністю та класифікуються літерами: n – nano, s – small, m – medium, l – large, x – extra large.

Основні параметри кожного типу МД та метрики на стандартному COCO 2017 датасеті [11] моделей представлені в таблиці 1. Кількість параметрів нелінійно впливає на розмір моделі, тому для найдинамічніших та легких задач обмежуються використання найменшої моделі з подальшим налаштуванням гіперпараметрів, датасету, та варіантів розрахунку втрат. Також можна спостерігати тенденцію зменшення росту точності відповідно до кількості параметрів, що також грає істотну роль в знаходженні варіантів по підвищенню точності не збільшуючи розмір моделі.

Таблиця 1. **Порівняльна характеристика моделей Yolo**

Model	size(pixels)	mAP^{val}_{50-95}	Speed CPU ONNX (ms)	Speed T4 TRT10 (ms)	params (M)	FLOPs (B)
YOLO11n	640	39.5	56.1	1.5	2.6	6.5
YOLO11s	640	47.0	90.0	2.5	9.4	21.5
YOLO11m	640	51.5	183.2	4.7	20.1	68.0
YOLO11l	640	53.4	238.6	6.2	25.3	56.9
YOLO11x	640	54.7	462.8	11.3	56.9	194.9

Незважаючи на щорічний прогрес в архітектурі моделей YOLO, покращення якості розпізнавання образів залишається складним завданням. Прогрес в архітектурі моделей YOLO представлений на рис. 1, що показує мінімальний ріст поміж останніми поколіннями.

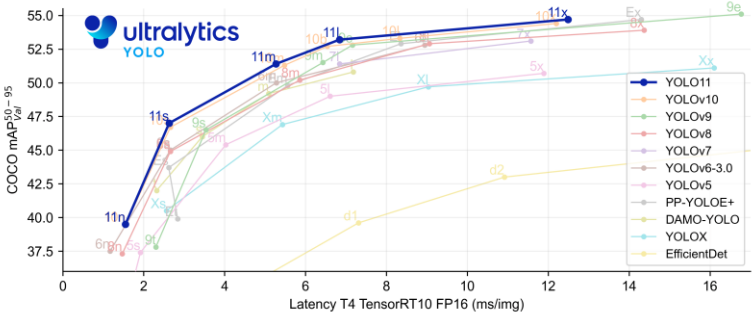

Рис. 1. **Прогрес архітектур Yolo**

Необхідність налаштування моделі перед навчанням, все ще є важливою областю досліджень. Автоматичне налаштування параметрів не завжди є найкращим варіантом, особливо при потребі досягнення точної детекції об’єктів. У даній статті зосереджено увагу, на налаштуванні гіперпараметрів та дослідженні, як вони можуть поліпшити точність МД, таких як YOLO пропонуючи апаратне рішення проблем, пов'язаних з обмеженими розрахунковими можливостями.

В загальному вигляді структура YOLO не змінювалась починаючи від YOLOv3 та в собі має наступні основні блоки. Кожний блок відіграє свою роль у процесі аналізу зображення детекції та класифікації, все це відбувається в прихованих шарах даних блоків.

Базовий блок (Backbone). Цей блок відповідає за витягування певних ознак та особливостей з зображення та за подальшу ідентифікацію даних ознак. Процес є найнижчим щаблем детекції і в той самий час є критично важливим, оскільки перетворює необроблені дані зображення в карти ймовірностей, що фіксують візуальну інформацію.

Шийний блок (Neck). Даний блок поєднує ознаки отримані від попереднього блоку, формує цілісне уявлення про об’єкти на зображені та передає далі до блоку детекції Head. Блок призначений для підвищення здатності розпізнавання малих об'єктів за допомогою використання шарів з різною роздільною здатністю. Підвищення відбувається за допомогою підвищення та локалізації ознак у малих шарах.

Головний блок (Head). Блок відповідає за прогнозування, генерацію обмежувальних рамок, прогнозування класів, та виявлення об’єктів. Він містить механізми, що дозволяють мережі робити кілька прогнозів для заданої області інтересу, покращуючи точність виявлення. Крім того, ця частина включає спрощену стратегію прогнозування, що дозволяє швидше робити висновки, що є критично важливим для додатків, що працюють у режимі реального часу. Ці вдосконалення в шарі прогнозування забезпечують ефективну і точну роботу моделі в динамічних і чутливих до часу сценаріях [13].

Структурна схема з усіма блоками та гілками зображена на рис. 2.

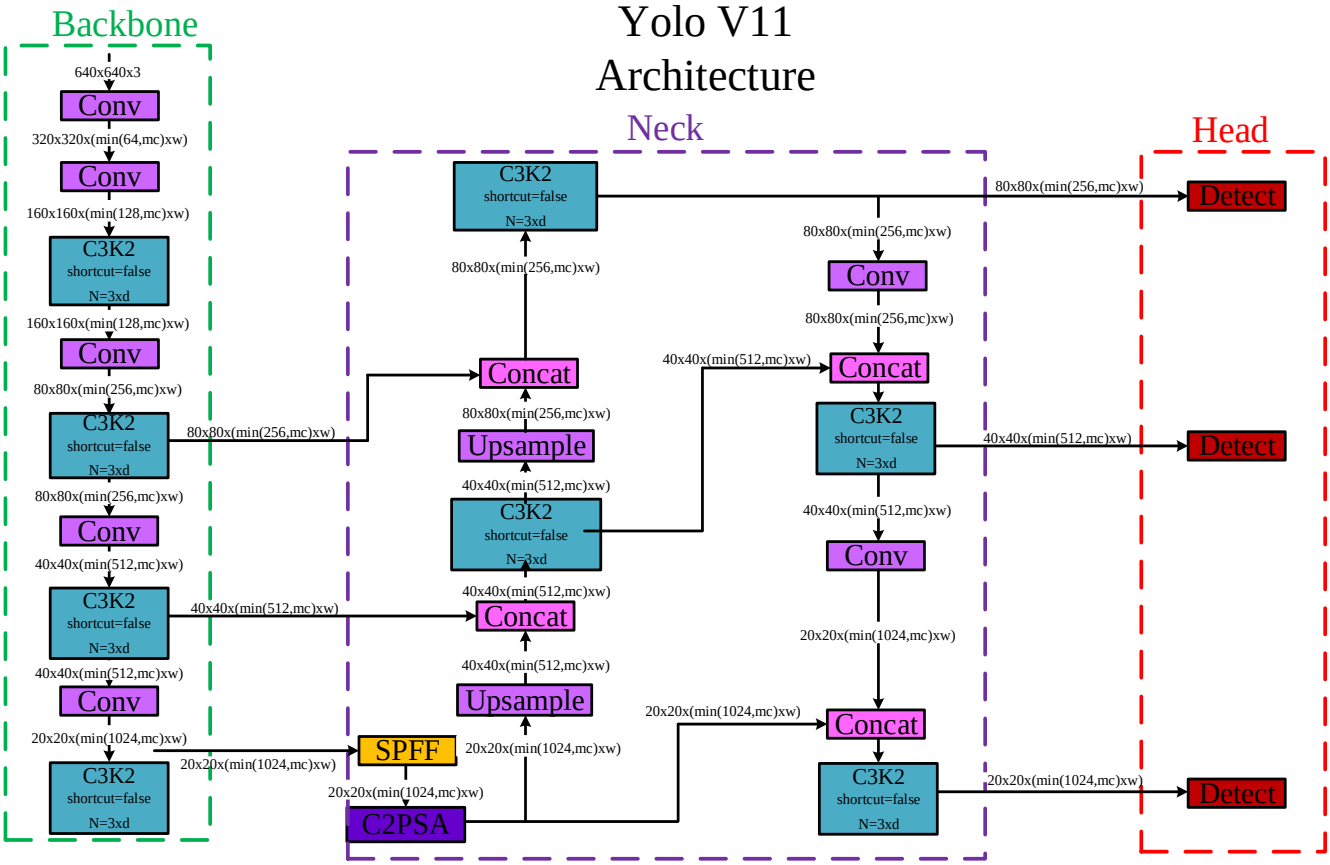

Рис. 2. **Структурна схема Yolov11**

На рисунку 2 зображено структурну схему YOLOv11 в якій базовий блок має наступні внутрішні блоки: Conv(convolution) – блок згортки котрий вилучає основні ознаки зображення, такі як краї, кути текстури при цьому використовує фільтри, щоб зменшити зображення зберігаючи важливу інформацію, готує дані для подальшої обробки більш складними шарами; C3K2 – блок каскадних згорток, який використовує декілька згорток для ще глибшого вилучення та комбінування ознак для кращого виявлення. Шийний блок має наступні внутрішні блоки: Upsample – блок збільшує розмір зображення для поліпшення розпізнавання об’єктів на різних рівнях, критичний блок для детекції малих об’єктів; Concats – блок котрий об’єднує виявлені ознаки з різних рівнів та масштабів; SPFF(Spatial Pyramid Pooling Fusion) – за допомогою концепції просторового пірамідного пулінгу вилучає ознаки на різних масштабах. Головний блок має тільки блоки Detect котрі визначають обмежувальні рамки для кожного об’єкта на зображенні. Він генерує сітку можливих обмежувальних рамок та ймовірності кожного блоку [14].

Особливої уваги в головному блоці мають втрати IoU (Intersection over Union), що є показником оцінювання, який зазвичай використовується для виявлення об'єктів. Втрата IoU, яка враховує різницю між площею справжніх і передбачуваних обмежувальних рамок, визначається як

$$\mathrm{IoU} = \frac{\mathrm{Area\,of\,Overlap}}{\mathrm{Area\,of\,Union}}, \tag{1}$$

де $\mathrm{Area\,of\,Overlap}$ – площа перетину між передбачуваним обмежувальним прямокутником і реальним обмежувальним прямокутником, рис. 3; $\mathrm{Area\,of\,Union}$ – площа об'єднання, тобто загальна площа, яку охоплюють обидва обмежувальні прямокутники.

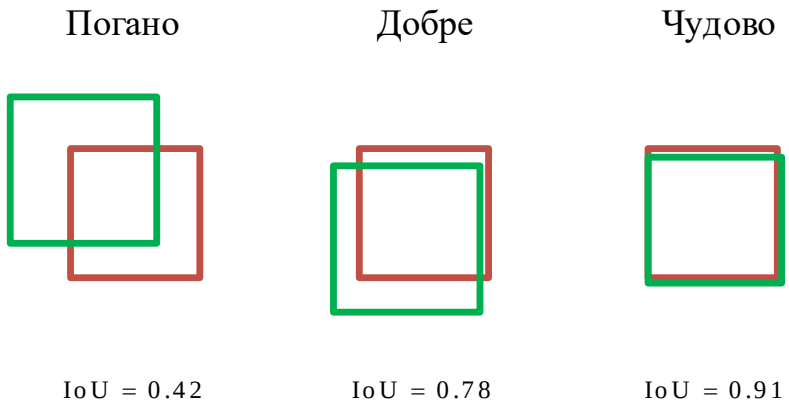

Рис. 3. **Варіанти площі перетину IoU**

Якщо IoU = 1, це означає ідеальну відповідність. Зазвичай значення $\mathrm{IoU} \geq 0.5$ вважається ознакою успішного виявлення об'єкту [15].

На рис. 4 показано спрощений приклад етапів роботи моделі по детекції зображень. Першим етапом роботи МД після отримання зображення є розділення на сітку розміром 80×80 комірок, що здійснюється базовим блоком. Даний розмір сітки дозволяє балансувати між точністю та швидкістю обробки зображення, оскількьки кожний блок оброблюється окремо. Після розділення на комірки та витягування ознак за допомогою вище згаданих блоків, відбувається передача за рахунок, неявно представленого, шийного блоку до головного блоку. Шийний блок підвищує якість детекції масштабованих та різнорозмірних об'єктів. В головному блоці кожна комірка прогнозує кількість обмежувальних рамок, кожен з яких має своє значення достовірності та ймовірність належності до класу певного об'єкта. Далі головний блок передає дані з ймовірностями на обробку де застосовується не максимальне придушення і генерується остаточне передбачення [16,17].

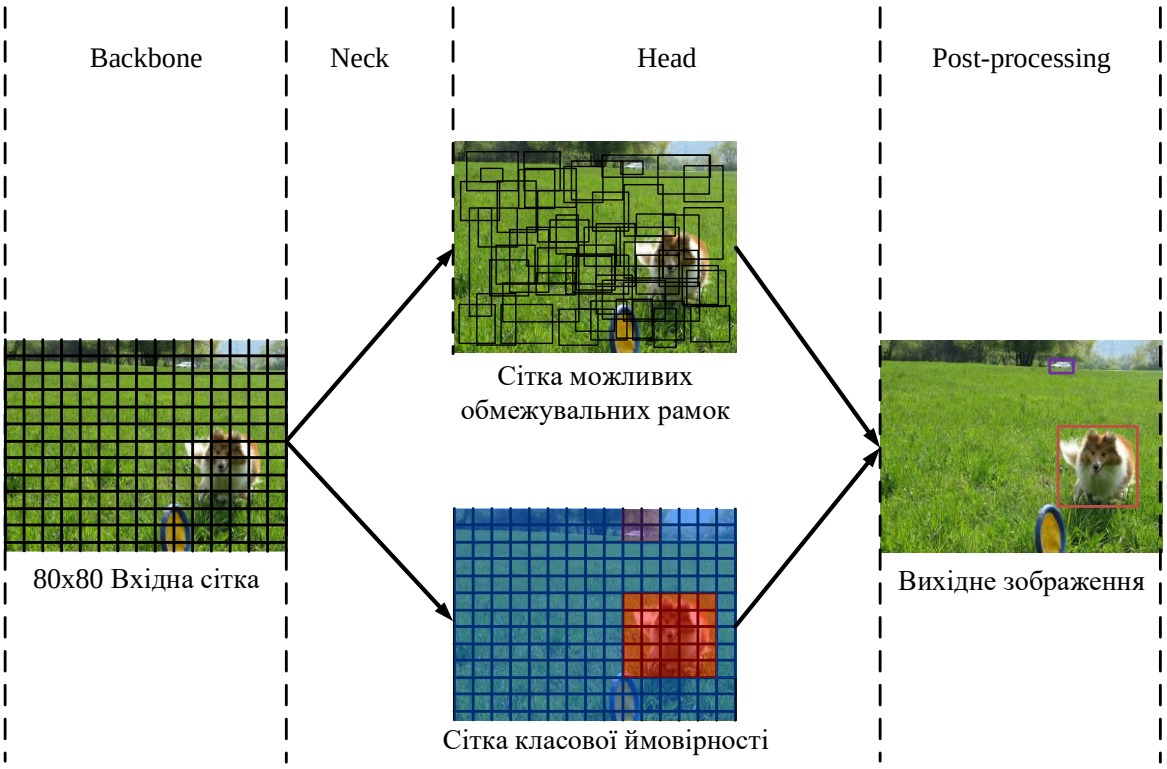

Рис. 4. **Етапи роботи моделі по детекції**

В якості плати детекції образів, для детекції на портативних пристроях можуть бути використані Nvidia Jetson Nano[18] або Raspberry Pi 5 [19] з платою розширення Hailo-8 [20], рис. 5.

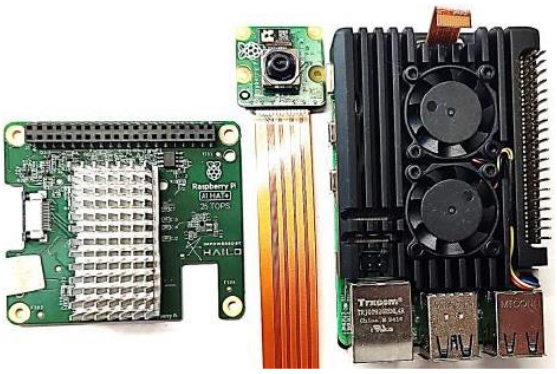

Рис. 5. **Raspberry Pi 5 з платою розширення Hailo-8**

Nvidia Jetson Nano має потужність до 67 TOPS (Tera Operations Per Second) в той час, як Raspberry Pi 5 з платою розширення Hailo-8 26 TOPS, що в портативних пристроях накладає обмеження на розмір та потужність МД. При цьому розміри першої плати 100x80x29мм та споживає до 25Вт. Raspberry з Hailo-8 має розміри 86x56x18мм та споживану потужність до 20Вт, що є більш компактним та енергоефективним рішенням.

Загалом гіперпараметри – це змінні, котрі визначаються користувачем заздалегідь і керують навчальним процесом. Вони не можуть бути змінені під час навчання з боку користувача, або автоматично. Дані параметри можуть бути зміненні лише перед початком навчання, а результати для набору параметрів, які були внесені перед початком тренування, стають відомі після повного навчання моделі, що може зайняти деякий час.

У роботі [21] показано, що для оптимізації продуктивності YOLO можна налаштувати кілька гіперпараметрів, таких як розмір вхідного зображення, кількість епох, розмір батчу, швидкість навчання та початковий імпульс. Крім того, налаштування гіперпараметрів, таких

як швидкість навчання та початковий імпульс, може скоротити час навчання та покращити показники точності моделі.

Вибір гіперпараметрів відіграє значну роль в здатності моделі до навчання і підбір здійснюється індивідуально до кожного нового датасету. З практичної сторони неправильно визначені гіперпараметри можуть бути причиною погіршення якості детекції та супроводжуватись явищами недотренованості та перетренованості.

В таблиці 2 нижче наведені основні гіперпараметри для моделей detection.

Таблиця 2. **Гіперпараметри для моделей Yolov11**

Параметр	Тип	Діапазон	Опис
lr0	float	(1e-5,1e-1)	Початкова швидкість навчання на початку тренування. Більш низькі значення забезпечують більш стабільне навчання, але більш повільну збіжність
lrf	float	(0.01,1.0)	Коефіцієнт кінцевої швидкості навчання як частка від lr0. Контролює, наскільки зменшується швидкість навчання під час тренування
mom-entum	float	(0.6,0.98)	Фактор імпульсу SGD. Більш високі значення допомагають підтримувати постійний напрямок градієнта і можуть прискорити збіжність
weight_ decay	float	(0.0,0.001)	Коефіцієнт L2-регуляризації для запобігання перенавчання. Великі значення забезпечують більш сильну регуляризацію
warmup_ epochs	float	(0.0,5.0)	Кількість епох для лінійного розігріву швидкості навчання. Допомагає запобігти нестабільності на ранніх етапах навчання
warmup_ mom-entum	float	(0.0,0.95)	Початковий імпульс під час фази розігріву навчання. Поступово збільшується до кінцевого значення імпульсу
warmup_ bias_lr	float	(1e-5,1e-3)	Швидкість навчання для параметрів зміщення під час фази розігріву, що допомагає стабілізувати навчання моделі в початкових епохах
box	float	(1,10)	Вага втрат обмежувального прямокутника в загальній функції втрат. Балансує регресію обмежувального прямокутника і класифікацію
cls	float	(0.2,4.0)	Вага втрат класифікації в загальній функції втрат. Більш високі значення підкреслюють правильне передбачення класу
dfl	float	(0.5,2.5)	Вага distribution focal loss, що використовується в деяких версіях YOLO для точної класифікації

При навчанні нейронних мереж підбір гіперпараметрів можна здійснювати як вручну так і використовувати алгоритми оптимізації. В YOLO найбільше використання отримали три

алгоритми оптимізації: SGD (стохастичний градієнтний спуск), Adam (адаптивна оцінка моменту навчання), and AdamW (покращений варіант Adam).

SGD – це широко використовуваний алгоритм оптимізації в машинному навчанні та навчанні нейронних мереж. За замовчуванням, при навчанні моделі без вибору алгоритму оптимізації даний оптимізатор використовується за замовчуванням.

SGD працює шляхом коригування параметрів моделі (θ_t) для зменшення помилки $L(\theta_t)$ під час навчання. Це досягається шляхом обчислення, наскільки змінюється помилка моделі у відповідь на невеликі зміни її параметрів. Це називається градієнтом, який показує напрямок руху для зменшення помилки.

SGD оновлює параметри моделі, переміщуючи їх у напрямку, протилежному до цього градієнта. Розмір кожного кроку контролюється значенням, яке називається швидкістю навчання (η) і визначає, наскільки великими або малими будуть коригування:

$$\theta_t = \theta_{t+1} - \eta \cdot \nabla_\theta L(\theta_t), \tag{2}$$

де, θ_t – параметри моделі; θ_{t+1} – оновлений параметр після кроку навчання; η – параметри швидкості навчання; $L(\theta_t)$ – функція втрат (loss function) для поточних параметрів моделі; $\nabla_\theta L(\theta_t)$ – градієнт функції втрат за параметрами θ.

Для кожного прикладу навчання модель здійснює оновлення на основі помилки для цього прикладу. Згодом це допомагає моделі зменшити загальну помилку і покращити прогнозування.

Таким чином, SGD допомагає моделі навчатися, крок за кроком коригуючи її внутрішні налаштування на основі інформації про те, наскільки поточні налаштування впливають на загальну помилку [21].

Оптимізатори Adam і AdamW базуються на концепціях SGD, але впроваджують методи адаптивного регулювання швидкості навчання. Ці оптимізатори використовуються для регулювання параметрів мережі шляхом обчислення градієнтів, подібно до SGD. Однак AdamW впроваджує регуляризацію параметрів шляхом їхнього зменшення, що загалом призводить до швидшого згортання до оптимального результату.

Adam і AdamW є розширеннями SGD, що включають адаптивну оцінку моментів, а саме середнє значення попередніх градієнтів (m_t та v_t). Вони обчислюють ковзні середні значення градієнтів, позначені, як m_t, і квадратичні градієнти, позначені, як v_t, як показано в рівняннях (3) і (4) відповідно. Ці ковзні середні значення представляють перший і другий моменти градієнтів. Параметри оновлюються за допомогою правила Adam, наведеного в рівнянні (5).

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \tag{3}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2, \tag{4}$$

В (5)-(6) коефіцієнт β_1 контролює експоненціальну швидкість згасання для оцінки першого моменту, коефіцієнт β_2 контролює швидкість згасання для оцінки другого моменту, а g_t є градієнтом, обчисленим на поточній партії даних. Ці коефіцієнти використовуються для оновлення m_t і v_t під час процесу навчання, що в кінцевому підсумку призводить до поліпшення збіжності в оптимізації [21].

Для остаточної формули оновлення параметра необхідно ввести корегування зміщення моментів, котрі розраховуються за формулами:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}; \tag{7}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}; \tag{8}$$

Оновлення параметрів з використанням корегованих моментів та градієнтів:

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{\hat{v}_t} + \varepsilon} + \hat{m}_t. \tag{9}$$

де ε – мале значення для запобігання ділення на нуль.

Для навчання моделі YOLO11n використано інструмент Google Colab, який надає доступ до серверних потужностей на обмежений строк. Навчання також відбувалось на локальному апаратному забезпеченні з процесором AMD Ryzen 5 5600, графічним прискорювачем NVIDIA RTX 3060 і 32 ГБ оперативної пам'яті DDR4.

У формуванні датасету використано стандартний COCO 2017 [11] датасет котрий був обмежений класами: person, car, motorcycle, bus, train, truck, traffic light, stop sign. Загальна кількість зображень за даними класами становить 50000. Для забезпечення балансу між класами датасету кількість зображень класу person зменшено оскільки він істотно переважає за чисельністю. В результаті отримано 27000 зображень які в подальшому були розділені на три частини: 18900 зображень (70 %) використовувалися для навчання, 5400 (20 %) – для валідації та 2700 (10 %) – для остаточного тестування.

Під час налаштування навчання було використано наступні параметри котрі класифікуються як гіперпараметри проте мають просте налаштування. До них відносяться розмір батчу, який був обраний 64, що повністю заповнює тільки виділену відеопамять без залучення додаткового буфера. Моделі навчалися на 30 епох, що є середнім значення для COCO датасету.

Гіперпараметри налаштовувались за допомогою трьох алгоритмів оптимізації AdamW, SGD та Adam, які в кожному прогоні мали 10 ітерацій. Дана кількість ітерацій є достатньою, щоб показати основні відмінності роботи даних алгоритмів оптимізації та вплив гіперпараметрів на точність моделі.

Основним показником, що використовується для оцінки ефективності кожної моделі, є середня точність (mAP), яка має вирішальне значення для систем виявлення об'єктів. На рисунку 6 представлені ітерації алгоритмів оптимізації, звідки видно, що алгоритм оптимізації SGD перевершує інші алгоритми за інтегральним показником якості моделі Fitess, що характеризується середнім значенням точності при $\text{IoU} = 0.5$ та $0.5 \le \text{IoU} \le 0.95$.

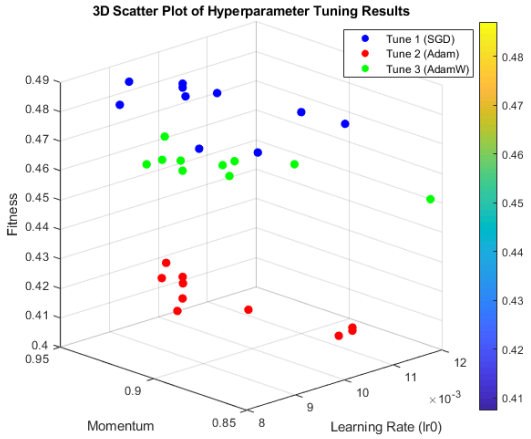

Рис. 6. **Ітерації алгоритмів оптимізації**

З рис. 7, порівняння метрик серед алгоритмів оптимізації, видно що SGD перевершив Adam, так і AdamW за показниками середнього значення точності при порозі $\text{IoU} = 0.5$

(mAP50) на 18.2%, усередненого значення точності на порогах $0.5 \le \text{IoU} \le 0.95$(mAP50-95) на 13.2 % та абсолютної точності передбачення на 5 % що сигналізує про більш точний підбір гіперпараметрів, особливо швидкості навчання, початкового імпульсу та спаду коефіцієнт L2-регуляризації.

При налаштуванні гіперпараметрів та навчанні МД з метою поліпшення її характеристик помічено суттєве збільшення часу, що витрачається на підбір гіперпараметрів. В результаті використання алгоритмів оптимізації загальний час на підбір параметрів для одного алгоритму оптимізації тривав 65000 секунд, тобто приблизно 18 годин. Таке значення часу передбачає потребу в значних розрахункових потужностях для налаштування гіперпараметрів.

Одним з ключових висновків експериментів стала значна різниця в mAP для різних варіацій гіперпараметрів заданих за допомогою алгоритмів оптимізації.

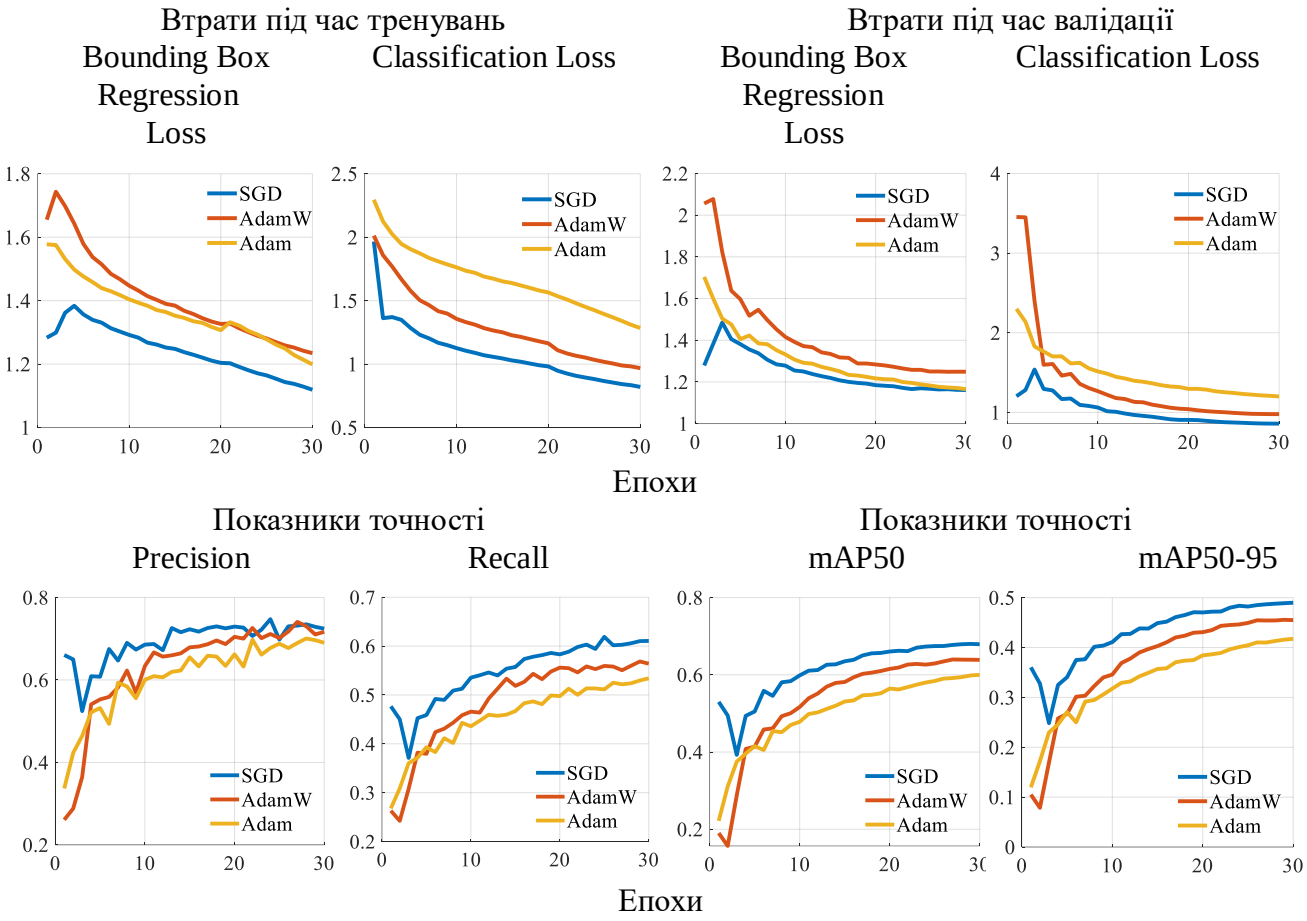

Рис 7. **Порівняння метрик серед алгоритмів оптимізації**

З рис. 7 порівняння метрик можна помітити, збільшення mAP безпосередньо корелює з поліпшенням локалізації об'єктів та точності класифікації, що є надзвичайно важливим для виявлення об'єктів. Значення втрат на навчання та валідацію мають схожу тенденцію де втрати в SGD найменші та становлять 1.12 на навчання та 0.82 на валідацію, що свідчить про ефективніше коригування помилки в процесі тренування. SGD показує найвищі показники Precision при початкових епохах, проте в останній період навчання AdamW показує схожі результати та становлять 0.72. Показник Recall в SGD становить 0.61 в зрівнянні з 0.53 в Adam, що свідчить про велику кількість позитивних детекцій в SGD.

Висновки. За результатами дослідження процедур налаштування гіперпараметрів та тренування МД, встановлено, що алгоритм оптимізації SGD показав найкращі результати за

показником mAP – 0.49, продемонструвавши значну перевагу в точності від найгіршого варіанту mAP – 0.41 в Adam. Проте, процес підбору гіперпараметрів вимагав значних обчислювальних ресурсів та тривав близько 18 годин на локальному обладнанні.

Отримані результати свідчать про доцільність використання алгоритму оптимізації SGD для отримання найкращих познаків точності. Однак складність підбору параметрів зводиться до наявності достатніх обчислювальних ресурсів.

СПИСОК ЛІТЕРАТУРИ

1. “Object Detection in Industrial Machine Vision Applications - HD Vision Systems”. *HD Vision Systems*. – URL: https://www.hdvisionsystems.com/en/blog/object-detection-in-industrial-machine-vision-applications/#:~:text=Object%20detection%20is%20a%20key,improving%20safety%20in%20manufacturing%20environments (дата звернення: 14.10.2025).

2. Özcan İ., Altun Y., Parlak C. “Improving YOLO Detection Performance of Autonomous Vehicles in Adverse Weather Conditions Using Metaheuristic Algorithms”. *Applied Sciences*. 2024. 14 (13): 5841. DOI: https://doi.org/10.3390/app14135841.

3. Poureskandar R., Razzagzadeh S. “Improving Object Detection Performance through YOLOv8: A Comprehensive Training and Evaluation Study”. – URL: https://www.researchgate.net/publication/391803447_Improving_Object_Detection_Performance_through_YOLOv8_A_Comprehensive_Training_and_Evaluation_Study (дата звернення: 28.09.2025).

4. Ding D., Deng Z., Yang R. “YOLO-TC: An Optimized Detection Model for Monitoring Safety-Critical Small Objects in Tower Crane Operations”. *Algorithms*. 2025; 18 (1): 27. DOI: https://doi.org/10.3390/a18010027.

5. “Ultralytics YOLO Hyperparameter Tuning Guide Ultralytics”. *Ultralytics YOLO Docs*. – URL: https://docs.ultralytics.com/ru/guides/hyperparameter-tuning/#default-search-space-description (дата звернення: 28.09.2025).

6. Arnold C., Biedebach L., Küpfer A., Neunhoeffer M. “The role of hyperparameters in machine learning models and how to tune them”. *Political Science Research and Methods*. 2024; 12 (4): 841–848. DOI: https://doi.org/10.1017/psrm.2023.61.

7. LeCun Y., Bengio Y., Hinton G. “Deep learning”. *Nature*. 2015; 521 (7553): 436–444. DOI: https://doi.org/10.1038/nature14539.

8. Pateriya P. N., et al. “Deep Residual Networks for Image Recognition*”. International Journal of Innovative Research in Computer and Communication Engineering*. 2023; 11 (09): 10742–10747. DOI: https://doi.org/10.15680/ijircce.2023.1109026.

9. Liu X., Gan H., Yan Y. “Study on Improvement of YOLOv3 Algorithm”. J*ournal of Physics: Conference Series*. 2021; 1884 (1): 012031. DOI: https://doi.org/10.1088/1742-6596/1884/1/012031.

10. Xu J., et al. “Gesture Object Detection and Recognition Based on YOLOv11”. *Applied and Computational Engineering*. 2025; 133 (1): 81–89. DOI: https://doi.org/10.54254/2755-2721/2025.20604.

11. “COCO 2017 dataset. COCO - Common Objects in Context”. – URL: https://cocodataset.org/#home – [Accessed: 28.09.2025].

12. “Ultralytics. YOLO11”. *Ultralytics YOLO Docs*. – URL: https://docs.ultralytics.com/ru/models/yolo11 – [Accessed: 28.09.2025].

13. Rao S. N. “YOLOv11 Explained: Next-Level Object Detection with Enhanced Speed and Accuracy. Medium”. – URL: https://medium.com/@nikhil-rao-20/yolov11-explained-next-level-object-detection-with-enhanced-speed-and-accuracy-2dbe2d376f71 – [Accessed: 28.09.2025].

14. Rao S. N. “YOLOv11 Explained: Next-Level Object Detection with Enhanced Speed and Accuracy”. *Medium*. – URL: https://medium.com/@nikhil-rao-20/yolov11-explained-next-level-object-detection-with-enhanced-speed-and-accuracy-2dbe2d376f71. – [Accessed: 14.09.2025].

15. Terven J., CordovaEsparza D. “A Comprehensive Review of YOLO: From YOLOv1 to YOLOv8 and Beyond”. 2023. DOI: 10.48550/arXiv.2304.00501.

16. “Ultralytics. YOLO Performance Metrics”. *Ultralytics YOLO Docs*. – URL: https://docs.ultralytics.com/guides/yolo-performance-metrics . – [Accessed: 28.09.2025].

17. Deepika H. C., et al. “An Overview of You Only Look Once: Unified, Real-Time Object Detection”. *International Journal for Research in Applied Science and Engineering Technology*. 2020; 8 (6): 607–609. DOI: https://doi.org/10.22214/ijraset.2020.6098.

18. “NVIDIA Jetson Nano”. *NVIDIA*. – URL: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/product-development . – [Accessed: 10.09.2025).

19. “Raspberry-pi-5”. *raspberrypi*. – URL: https://www.raspberrypi.com/products/raspberry-pi-5 . – [Accessed: 10.09.2025].

20. “Hailo-8 AI Accelerator”. *Hailo*. – URL: https://hailo.ai/products/ai-accelerators/hailo-8-ai-accelerator . – [Accessed: 10.09.2025].

21. Isa I. S., Rosli M. S. A., Yusof U. K., Maruzuki M. I. F., Sulaiman S. N. “Optimizing the Hyperparameter Tuning of YOLOv5 for Underwater Detection”. *IEEE Access*. 2022; 10: 52818-52831. DOI: https://doi.org/10.1109/access.2022.3174583.

DOI: https://doi.org/10.15276/ict.02.2025.66
UDC 52-004

Research on the impact of optimization algorithms on the accuracy of Yolov11 neural networks

Serhii M. Kovbasa[1)]
PhD, Associate Professor, Head of the Department of Automation of Electromechanical Systems and Electric Drives
ORCID: https://orcid.org/0000-0002-2954-455X; skovbasa@ukr.net. Scopus ID: 55328200100
Anton O. Kholosha[1)]
Postgraduate Student of the Department of Automation of Electromechanical Systems and Electric Drives
ORCID: https://orcid.org/0009-0003-4858-202X; kholosha.anton@gmail.com
[1)] National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”, 37, Beresteiskyi Ave. Kyiv, Ukraine

ABSTRACT

Visual inspection and positioning based on image detection results is a rapidly growing component of automation systems. Machine vision is increasingly used in production lines for various purposes. Improving recognition accuracy in such applications can be a difficult task, especially in conditions of possible limitations, one of which may be size and weight restrictions, which in turn limit the power of computer devices that implement image detection and recognition. A possible solution to this problem is to improve recognition accuracy by increasing the number of image variants in the dataset and fine-tuning the model's hyperparameters. This article investigates the effectiveness of hyperparameter tuning for the YOLO (You Only Look Once) image detection model, which can be further implemented in electromechanical systems for positioning moving objects in space.

Keywords: Artificial intelligence; computer vision; hyperparameters; optimizers; YOLOv11; Roboflow; accuracy; IoU