МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В.Н. Каразіна
Факультет математики та інформатики
Кафедра теоретичної та прикладної інформатики

Кваліфікаційна робота

магістра

на тему “**КЛАСИФІКАЦІЯ ЧАСТКОВО ВИДИМИХ ОБ’ЄКТІВ В УМОВАХ НЕВИЗНАЧЕНОСТІ**”

Виконав: студент 2 курсу, групи МФ-61
спеціальність 122 «Комп’ютерні науки»
освітньо-наукова програма «Інформатика»
Лугових Дмитро Юрійович
Керівник Панченко Артем Сергійович
Рецензент ПІБ РЕЦЕНЗЕНТА

Харків – 2026

АНОТАЦІЯ

ЛУГОВИХ ДМИТРО ЮРІЙОВИЧ. КЛАСИФІКАЦІЯ ЧАСТКОВО ВИДИМИХ ОБ'ЄКТІВ В УМОВАХ НЕВИЗНАЧЕНОСТІ – кваліфікаційна робота на здобуття освітнього рівня Магістр – Харківський національний університет імені В. Н. Каразіна, Міністерства освіти і науки України, Харків, 2026.

Дипломна робота присвячена дослідженню та аналізу сучасних методів класифікації у задачах комп'ютерного зору в умовах невизначеності, зумовленої частковим перекриттям об'єктів.

Дослідження охоплює оцінку ефективності сучасних архітектур ШІ для розпізнавання об'єктів з метою визначення їх здатності до роботи в умовах різного ступеня оклюзії та формування узагальненого підходу до аналізу їх поведінки.

Таким чином, запропонований у роботі підхід дозволяє здійснювати комплексне тестування моделей детекції об'єктів у контрольованих умовах невизначеності та виконувати кількісну оцінку їх стійкості до часткової втрати візуальної інформації.

У вступі обґрунтовано актуальність теми, що визначається необхідністю підвищення надійності систем комп'ютерного зору в реальних умовах, де об'єкти часто є частково перекритими. Сформульовано мету дослідження, яка полягає у проведенні порівняльного аналізу сучасних моделей детекції об'єктів щодо їх здатності до класифікації частково видимих об'єктів. Визначено об'єкт дослідження – процеси розпізнавання об'єктів у системах комп'ютерного зору. Предмет дослідження – архітектури (YOLO-сімейства: YOLOv12, YOLOv13, SSD, DEIMv2 та RT-DETRv2) для детекції об'єктів в умовах оклюзії. Методи дослідження базуються на використанні сучасних моделей, аналізі їх архітектурних

особливостей та експериментальній оцінці за допомогою стандартних метрик.

У першому розділі проведено дослідження задачі класифікації об’єктів в умовах оклюзії. Розглянуто альтернативні підходи та сучасні архітектурні рішення, орієнтовані на обробку частково перекритих об’єктів. Особливу увагу приділено аналізу механізмів вирішення складних випадків класифікації, зокрема в умовах значної втрати візуальної інформації, а також досліджено підходи, що підвищують стійкість моделей до невизначеності. Окремо виконано аналіз додаткових архітектур та їх компонентів, що використовуються для загальної задачі класифікації, з метою визначення їх потенціалу для адаптації до умов часткової видимості об’єктів.

У другому розділі розглянуто сучасні архітектури та їх компоненти, що застосовуються для вирішення задачі детекції частково видимих об’єктів. Проаналізовано особливості моделей YOLOv12, YOLOv13, SSD, DEIMv2 та RT-DETRv2, а також їх внутрішні механізми, що впливають на здатність працювати в умовах оклюзії. Описано конфігурацію архітектур YOLOv12, YOLOv13, DEIMv2 та RT-DETRv2 як складову експериментальної реалізації, а також особливості реалізації моделі SSD та її ключових компонентів. Розглянуто організацію наборів даних та супутні етапи їх обробки, включаючи підготовку та адаптацію до умов дослідження. Визначено основні перешкоди, що виникають під час виконання дослідження, та проаналізовано додаткові механізми обробки, спрямовані на підвищення ефективності моделей у складних умовах.

У третьому розділі представлено результати експериментального дослідження можливостей використання обраних архітектур у випадку частково видимих даних. Розглянуто метод оцінювання результатів, що базується на використанні метрик mAP та mAR із врахуванням параметрів IoU, Area та maxDets для комплексного аналізу якості детекції. Наведено

результати первісної оцінки моделей на стандартних умовах розпізнавання, що дозволило визначити базовий рівень їх продуктивності. Сформульовано основні результати роботи моделей на частково видимих даних, проведено порівняльний аналіз їх ефективності залежно від параметру.

У дипломній роботі вирішено актуальну науково-прикладну задачу оцінки ефективності сучасних моделей комп'ютерного зору в складних для класифікації об'єктів умовах.

У результаті дослідження реалізовано експериментальне середовище для тестування моделей комп'ютерного зору та отримано кількісні оцінки їх ефективності в умовах оклюзії.

Наукова новизна та практичне значення одержаних результатів полягає у наступному:

1. Проведено комплексний порівняльний аналіз сучасних моделей детекції об'єктів щодо їх стійкості до часткового перекриття;

2. Адаптовано методологію оцінки ефективності моделей для задач із невизначеністю на основі спеціалізованих датасетів та метрик;

3. Проаналізовано вплив архітектурних особливостей моделей на їх здатність розпізнавати оклюзивні об'єкти;

4. Розроблено навколишнє середовище для обробки даних та безпосереднього використання досліджуваних архітектур.

Сукупність отриманих результатів, їх наукова обґрунтованість та практична значущість дають підстави вважати поставлену задачу дослідження класифікації частково видимих об'єктів в умовах невизначеності вирішеною, а мету роботи – досягнутою.

ABSTRACT

LUHOVYKH DMYTRO YURIIOVYCH. CLASSIFICATION OF PARTIALLY VISIBLE OBJECTS UNDER CONDITIONS OF UNCERTAINTY – Master's qualification thesis – V. N. Karazin Kharkiv National University, Ministry of Education and Science of Ukraine, Kharkiv, 2026.

The thesis is devoted to the study and analysis of modern classification methods in computer vision tasks under uncertainty caused by partial object occlusion.

The research includes the evaluation of the effectiveness of modern AI architectures for object recognition in order to determine their ability to operate under different levels of occlusion and to form a generalized approach to analyzing their behavior.

Thus, the approach proposed in this work enables comprehensive testing of object detection models under controlled uncertainty conditions and provides a quantitative assessment of their robustness to partial loss of visual information.

The introduction substantiates the relevance of the topic, determined by the need to improve the reliability of computer vision systems in real-world conditions, where objects are often partially occluded. The goal of the research is formulated as conducting a comparative analysis of modern object detection models with respect to their ability to classify partially visible objects. The object of research is the process of object recognition in computer vision systems. The subject of research includes object detection architectures (YOLO family: YOLOv12, YOLOv13, SSD, DEIMv2, and RT-DETRv2) under occlusion conditions. Research methods are based on the use of modern models, analysis of their architectural features, and experimental evaluation using standard metrics.

The first chapter investigates the problem of object classification under occlusion conditions. Alternative approaches and modern architectural solutions

aimed at processing partially occluded objects are considered. Special attention is given to analyzing mechanisms for handling complex classification cases, particularly under significant loss of visual information, as well as approaches that improve model robustness to uncertainty. Additionally, further architectures and their components used for general classification tasks are analyzed to determine their potential for adaptation to partial visibility conditions.

The second chapter examines modern architectures and their components used to solve the problem of detecting partially visible objects. The features of YOLOv12, YOLOv13, SSD, DEIMv2, and RT-DETRv2 models are analyzed, along with their internal mechanisms affecting performance under occlusion. The configuration of YOLOv12, YOLOv13, DEIMv2, and RT-DETRv2 architectures is described as part of the experimental implementation, as well as the implementation specifics of the SSD model and its key components. The organization of datasets and related data processing stages, including preparation and adaptation for the study, are discussed. The main challenges encountered during the research are identified, and additional processing mechanisms aimed at improving model performance under complex conditions are analyzed.

The third chapter presents the results of the experimental study on the applicability of the selected architectures to partially visible data. The evaluation methodology is described based on mAP and mAR metrics, taking into account IoU, Area, and maxDets parameters for comprehensive quality assessment. The results of the initial evaluation under standard conditions are provided, establishing baseline performance levels. The main results of model performance on partially visible data are formulated, and a comparative analysis of their effectiveness depending on occlusion conditions is conducted.

The thesis solves an important scientific and applied problem of evaluating the effectiveness of modern computer vision models under conditions that are challenging for object classification.

As a result of the study, an experimental environment for testing computer vision models was developed, and quantitative evaluations of their effectiveness under occlusion conditions were obtained.

The scientific novelty and practical significance of the results are as follows:

1. A comprehensive comparative analysis of modern object detection models in terms of their robustness to partial occlusion has been carried out;

2. The methodology for evaluating model performance in uncertainty conditions based on specialized datasets and metrics has been adapted;

3. The influence of architectural features on the ability of models to recognize occluded objects has been analyzed;

4. An environment for data processing and direct use of the studied architectures has been developed.

The obtained results, their scientific validity, and practical significance allow us to conclude that the research problem of classifying partially visible objects under uncertainty conditions has been successfully solved, and the objectives of the work have been achieved.

ЗМІСТ

ВСТУП

Актуальність теми. Сучасні технології штучного інтелекту, зокрема методи комп'ютерного зору, відіграють ключову роль у вирішенні широкого спектра прикладних задач, таких як автономне керування транспортом, системи відеоспостереження, робототехніка та медична діагностика. Однією з базових задач у цій галузі є класифікація та розпізнавання об'єктів, що передбачає не лише визначення класу об'єкта, а й його локалізацію на зображенні. З розвитком нових архітектур постійно зростає точність і швидкість розпізнавання, проте значною проблемою залишається робота в умовах невизначеності, зокрема при частковому перекритті (оклюзії) об'єктів. У таких випадках моделі стикаються з втратою частини візуальної інформації, що може суттєво погіршувати якість їх роботи. Аналіз сучасних наукових досліджень показує, що хоча існує значна кількість архітектур для детекції об'єктів, питання їх ефективності в умовах оклюзії залишається недостатньо дослідженим. Це обумовлює необхідність проведення комплексного аналізу можливостей різних підходів та визначення їх придатності до роботи в складних умовах.

Мета та задачі дослідження. Метою дипломної роботи є з'ясування можливостей сучасних архітектур для класифікації об'єктів у задачах подолання проблем, пов'язаних із різним ступенем оклюзії.

Для досягнення поставленої мети необхідно вирішити такі задачі:

1. Проаналізувати сучасні архітектури для задач класифікації та детекції об'єктів;

2. Імплементувати обрані архітектури та дослідити специфіку їх використання;

3. Розробити та адаптувати методи обробки даних відповідно до вимог моделей;

4. Підібрати та організувати набори даних для первісної та поглибленої оцінки моделей;

5. Використати або сформувати анотаційні дані для навчання та тестування моделей;

6. Провести валідаційну оцінку архітектур із використанням сучасних метрик якості;

7. Виконати порівняльний аналіз моделей на стандартних та оклюзивних даних;

8. Дослідити ефективність архітектур у складних умовах часткової видимості об’єктів.

Об’єкт дослідження. Процеси використання архітектур для класифікації об’єктів в умовах оклюзії.

Предмет дослідження є методи організації та застосування архітектур і їх компонентів, підготовка та обробка даних, формування анотацій, а також підходи до оцінювання ефективності моделей у задачах класифікації частково видимих об’єктів.

Методи дослідження. У межах дослідження застосовано такі методи та принципи:

1. Методи комп’ютерного зору для побудови та аналізу моделей детекції об’єктів, зокрема використання сучасних архітектур (YOLO, SSD, DEIMv2, RT-DETRv2).

2. Набір внутрішніх архітектурних механізмів для врегулювання обробки даних та налаштування взаємодії їх компонентів, що впливають на здатність працювати з частково видимими об’єктами.

3. Методи підготовки та обробки даних, зокрема організація наборів даних, формування анотацій, попередня обробка зображень та адаптація даних до вимог різних архітектур.

4. Методи оцінювання якості моделей, зокрема використання метрик mAP та mAR із врахуванням параметрів IoU, Area та maxDets для кількісного аналізу результатів.

5. Програмні методи реалізації (мова Python, бібліотека PyTorch) для імплементації, налаштування та тестування моделей.

Практичне значення отриманих результатів. Отримані результати можуть бути використані для вибору ефективних архітектур у задачах комп'ютерного зору, що працюють в умовах невизначеності, а також для подальшого вдосконалення систем розпізнавання об'єктів у прикладних сферах, таких як автономні системи, відеоспостереження та робототехніка.

Структура та обсяг дипломної роботи. Дипломна робота складається зі вступу, трьох розділів, висновків та списку використаних джерел. Основна частина роботи присвячена дослідженню класифікації оклюзивних даних, аналізу архітектур та їх компонентів, а також експериментальному оцінюванню ефективності моделей у задачах розпізнавання частково видимих об'єктів.

1 ДОСЛІДЖЕННЯ КЛАСИФІКАЦІЇ ОКЛЮЗИВНИХ ДАНИХ

1.1 Аналіз механізмів з вирішення важких випадків класифікації

Під час огляду наявних підходів до вирішення саме цього питання можна знайти багато літератури, яка тільки досліджує як інші доступні архітектури або ж підходи долають цю проблему. Компроміс між швидкість та загальним часом грає важливу роль у виконанні детекції з класифікацією об’єктів, саме так можна досягти результатів які б задовольнили обидві сторони. Все ж таки, окрім більш просунутих архітектур до загальної класифікації, є й такі, що націлені на важкі випадки класифікації – оклюзивні випадки. Проблема виникає в середовищах, де такі випадки можуть траплятися надзвичайно часто, аніж у менш навантажених різними об’єктами місцях.

1.1.1 Контекстне розпізнавання об’єктів у важких випадках

У роботі присутнє порівняння з доволі відомою архітектурою під назвою Faster Regions with Convolutional Neural Network (Faster R-CNN) доводячи її неефективність у задачі розпізнавання під час важких випадків класифікації. У ній пропонується подолати деякі обмеження цієї архітектури використовуючи покращену версію згорткової мережі під назвою Compositional Convolutional Neural Networks (CompositionalNets). В основному, ці обмеження стосуються саме механізмів запроваджених у цій архітектурі:

1. Regional Proposal Network (RPN) не виділяє регіони достатньо чітко під час важких випадків видимості об’єктів;

2. Присутня ненадійна класифікація частково видимих об’єктів.

Перший запропонований варіант такої згорткової мережі була недостатньо надійна для такої класифікації, що й призвело до появи даної роботи. Генеративна композиційна модель може чітко розпізнавати об’єкти як композиційні частини, які у поєднанні з системою голосування дозволяє надійну класифікацію засновану на просторової конфігурації декількох видимих частин. Однак, були певні проблеми, які не дозволяли їй розпізнавати об’єкти:

1. Вони не дозволяли явно відокремити отриманий контекст від самого об’єкту. Експерименти свідчили про негативний ефект ще на етапі виділення контексту на тренувальних даних (наприклад, літак зазвичай можна побачити у небі). Якщо об’єкти сильно перекриті, то повинен бути зменшений поріг детекції. У свою чергу це може призвести до впливу на контекст об’єкту та збільшення негативно-позитивних (eng. false-positive) розпізнавань в регіонах, де об’єктів немає (наприклад, якщо повинна бути розпізнана машина з оклюзіями, то літак може бути неправильно розпізнаний у небі, цей ефект можна побачити на рис. 1.1).

2. У таких мережах є недостача механізмів направлених на надійну оцінку границь (eng. bounding box) об’єкту. Експерименти показали, що RPNs не оцінюють границі надійно коли об’єкти знаходяться під частковою оклюзією.

На рис. 1.1 зображено границі під впливом різних підходів, включаючи також істину (eng. ground truth). На цьому рисунку можна побачити наступні результати виділення границь:

- Синя границя – істина;
- Червона границя – Faster R-CNN (RPN+VGG);
- Жовта границя – RPN+CompositionalNet;
- Зелена границя – контекстна CompositionalNet з надійним голосування за границі.

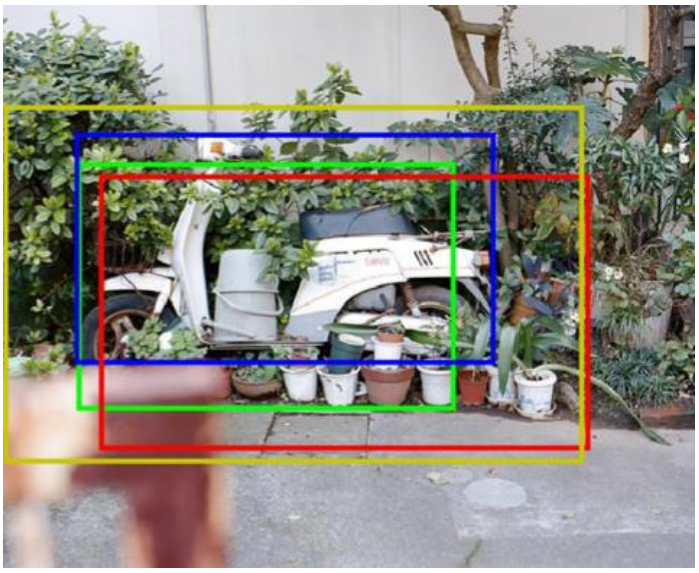


Рис 1.1 – Розпізнавання об’єктів з RPN та з запропонований надійне головування за границі (eng. robust bounding box voting)

Результати демонструють, що RPN підхід помилково локалізує границі об’єкту, у той час як запропонований метод зміг коректно розпізнати потрібний об’єкт у складній ситуації. Досягати такого ефекту допомогло застовування розпізнавального шару та запропонована декомпозиція відображення зображень як поєднання відображення контексту та об’єкту. Контекстне відображення зображення дозволяє контролювати вплив контексту на детекційні результати. Тим паче, надійний механізм голосування дозволяє оцінити границі об’єкту більш чіткіше, а розширена частинна схема голосування (eng. part-based voting scheme) у CompositionalNets робить можливим голосування за два протилежних кути границь разом з центром об’єкту. Загально можна підкреслити наступні внески:

1. Декомпозиція відображення зображення в CompositionalNets як поєднання контексту та об’єкту. Такий підхід демонструє точний контроль впливу контексту об’єкта на результати розпізнавання таким чином збільшуючи надійність класифікації об’єктів за умови сильної оклюзії;

2. Надійна частинний механізм голосування для розраховувань границь, який дозволяє робити такі розрахування точно навіть при важких оклюзіях;

3. Згідно експериментам саме комбінація цих двох механізмів разом допомагають перевершити Faster R-CNN в розпізнаванні об’єктів під час часткових оклюзій з доволі великою різницею.

1.1.1.1 CompositionalNets

Ця мережа є Deep Convolutional Neural Networks (DCNNs) з властивою надійністю до часткових оклюзій та структура, яка нагадує VGG-16, де повністю з’єднана голова (eng. fully connected head) замінена на диференційовану генеративну композиційну модель активацій особливостей $p(F|y)$ та y є категорією об’єкту. Композиційна модель відповідає поєднанню von-Mises-Fisher (vMF) розподіленню. Наведені зміни наведено у формулах 1.1-1.4.

Також є можливість доповнення цієї мережі за допомоги оклюзивної моделі, яка визначає надійну вірогідність того, у якій саме позиції p у зображенні або ж об’єктна модель $p(f_p|\mathcal{A}_{p,y}^m, \Lambda)$ або оклюзивна модель $p(f_p|\beta, \Lambda)$ буде активною. Об’єктна модель підпадає до визначення за наступними формулами 1.5-1.7.

Оклюзивна у свою ж чергу буде поєднаною моделлю, наведено у формулі 1.8.

1.1.1.2 Шар детекції з найдійним голосуванням за границі

Сама мережа не може сама розпізнавати об’єкти, тому зазвичай до неї додають RPNs, які показали доволі погані результати при виділені границь.

Для подолання цієї проблеми було створено надійний частинний механізм голосування заснований на виділенні видимих частин об’єкту. Особливим у цьому нововведенні є декомпозована об’єктна модель $p(F|\Theta_y)$ в поєднання композиційних моделей $p(F|\Theta_y^m)$, де кожне поєднання компоненту відображає об’єкт класу y з урахуванням різного положення. Також варто зауважити те, що CompositionalNets навчаються на зображеннях які були вирізані спираючись на відповідні границі об’єкту. Центруючи об’єкт у зображенні, як це показано на рис 1.2, кожне поєднання компоненту $p(F|\Theta_y^m)$ можна вважати накопиченими голосами (eng. accumulating votes) з частин моделей з метою центрувати об’єкт в карті особливостей (eng. feature map).

Рисунок 1.2 – Контекстна сегментація результатів

Для досягнення такого накопичення голосів для центру об’єкту у всіх позиціях p в карті особливостей F, буду розраховано об’єктна вірогідність при скануванні всіх позицій p та буде згенерована просторова карта вірогідностей (eng. spatial likelihood map) за наступною формулою 1.9.

Проблема розпізнавання в важких умовах полягає у надзвичайно великій кількості частин, які можуть бути перекриті, як це показано на рис

1.3. Вона була вирішена так званим частинним механізмом голосування особливість якого полягає у вивчені додатково поєднання компонентів, що моделює очікувані активації особливостей F навколо кутів границі $p(F_p|\Theta_y^c)$, де $c = \{ct, bl, tr\}$ – центр об'єкта ct, та два протилежних кута границі $\{bl, tr\}$. На рис. 1.3 можна побачити просторову мапу вірогідностей R^c усіх трьох моделей. Далі генеруються границі на основі двох точок з максимальними вірогідностями, що дозволяє виділяти границі чітко незважаючи на великі частини об'єкту перекриті.

Рисунок 1.3 – Результат надійного голосування за границю

1.1.1.3 Контекстний CompositionalNets

Так вийшло, що стандартна версія не може відділити відображення контексту від об'єкта, хоча контекст може бути корисним для розпізнавання об'єктів через зміщення (eng. bias), наприклад, літаки частіше оточений небом. На рис. 1.4 можна побачити, що сильний акцент на контекст може стати оманливим фактором коли об'єкти сильно перекриті, таким чином поріг детекції повинен бути зменшений під час складної оклюзії. Тому контроль впливу на контекстуальні сигнали на детекцію є доволі важливими.

Рис 1.4 – Вплив контексту в детекції під оклюзією

Context-Aware CompositionalNets (CA-CompositionalNets) контролює вплив, розділяючи відображення контексту від зображення в оригінальній CompositionalNets за допомогою представлення мапи особливостей F як поєднання двох моделей, як це наведено у формулі 1.10:

Сегментуючи тренувальні зображення в контекст та об’єкт заснований на доступній граничній анотації, засновано на тому, що будь-яка особливість, яка має поле сприйняття (eng. receptive field) зовні у рамках границь, розглядаються як частини контексту. Після вилучення випадкових особливостей популяції вона кластеризується використовуючи вдосконалений алгоритм K-means++ та отримується асоціативний масив центри особливостей контексту (eng. context feature centers) $E = \{e_q \in \mathbb{R}^D | q = 1, \dots, Q\}$ до цього застосовуємо поріг на косинусі подібності $s(E, f_p) = \max_q[(e_q^T f_p)/(\|e_q\|\|f_p\|)]$ для сегментації контексту та об’єкту для будь-якого тренувального зображення. На рис. 1.2 можна побачити попередньо описану сегментацію.

1.1.2 Адаптивне розпізнавання оклюзії

Зазначимо, що алгоритм заснований на архітектурі You Only Live Once X-small (YOLOX-s), також його структуру можна побачити на рис. 1.5. У роботі ця структура була покращена на предмет поганої детекції перекритих об’єктів з боку транспортних засобів та запропонований адаптивно деформований алгоритм для розпізнавання оклюзій.

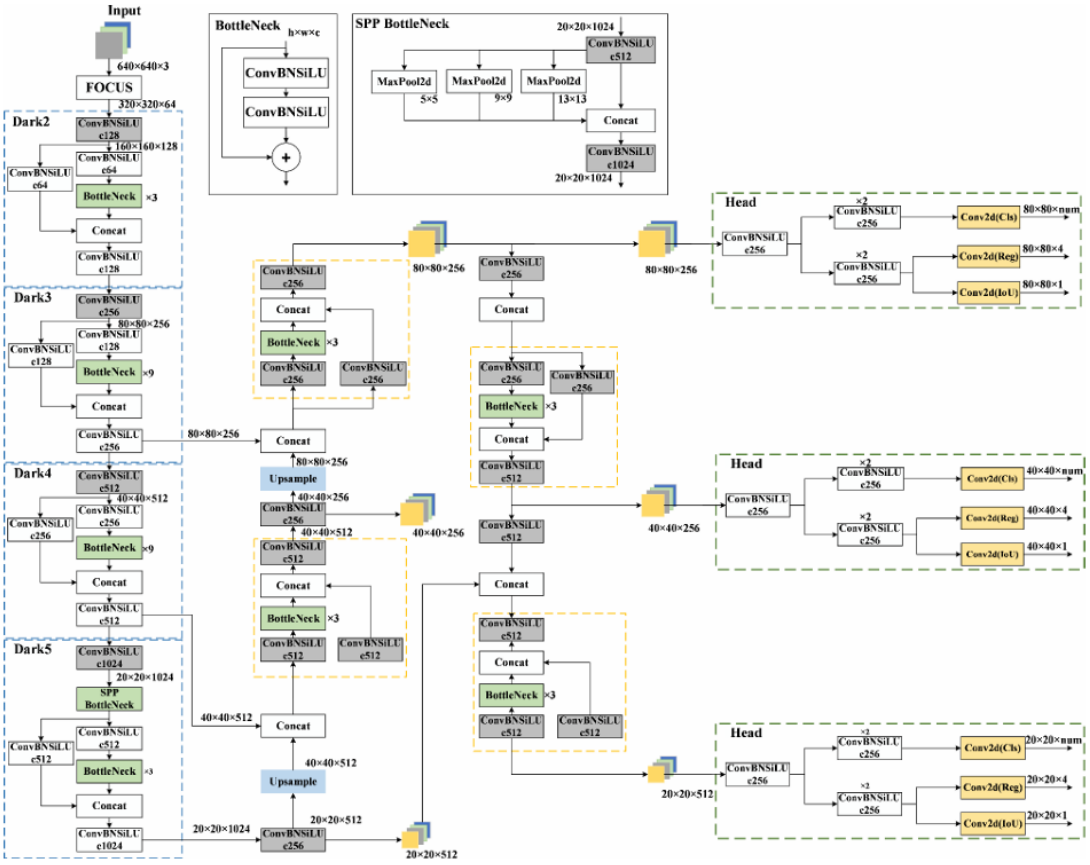


Рисунок 1.5 – Архітектура YOLOX-s

Загально окрім цього можна винести наступні зміни:

1. Використання адаптивно деформованого згорткового механізму у основному блоці (eng. backbone network) YOLOX для ефективного підвищення можливості алгоритму для трансформації складної геометрії для адаптації до складних та динамічних сценаріїв оклюзії та посилення здатності відображення особливостей відносно перекритих об’єктів;

2. Координаційний механізм уваги в шийному блоку (eng. neck part) для подолання обмежень використання ваг у різних каналах та просторових області з метою покращення здатності до злиття залишкового отримання особливостей перекритих цілей;

3. Пропонується регресійна функція втрат на основі накладного IoU (OL-IoU), який працює з перетином та об'єднанням, тобто IoU, та пропорційний механізм штрафів такі як ширина накладання та його висота, які використовуються як для прискорення збіжності границь так й для підвищення точності розпізнавання;

4. Spatial Pyramid Pooling (SPP) замінений на більш ефективну Fast SPP, щоб балансувати розпізнавання у реальному часі та точність розпізнавання оклюзій, які підтвердилися експериментами.

1.1.2.1 Деформативний згортковий механізм

Під час формування особливостей зображень звичайним механізмом згортання, поля сприйняття уособлюють собою всі звичайні сусідні прямокутники з обмеженою можливістю геометричних варіацій. Є певна потреба у алгоритмі з можливістю адаптивно сприймати цільовий регіон та налаштувати такі поля, саме тому положення та фігура перекритих цілей, або ж у цілому об'єктів, не можуть бути передбачені. Deformable Convolutional Networks (DCN) забезпечує адаптивне навчальне зміщення (eng. adaptive learnable offset) для підвищення просторового обирання точок, які дозволять мережі мати здатність до адаптивного моделювання. Процес деформації створений саме для цього та розраховується за наступною формулою 1.11.

Деформативний згортковий механізм використовує згадане зміщення для опису орієнтацій особливостей цільового об'єкту, дозволяючи змістовому полю мережі (eng. network's sense field) не бути обмеженим у

фіксованому діапазоні та бути більш гнучким у адаптації змін в цільовій геометрії. Хоч DCNs не бере на себе більше розрахунків, але при цьому використання таких мереж у великій кількості підвищить час затрачений алгоритмом на передбачення опираючись на нові дані. З метою збалансування точності та швидкості та адаптивної зміни вибіркову позицію згорткового ядра через зміщення, яке можна побачити на рис 1.6, була використана стандартна 3x3 згорткова мережа у шарі основного блоку FEAT3.

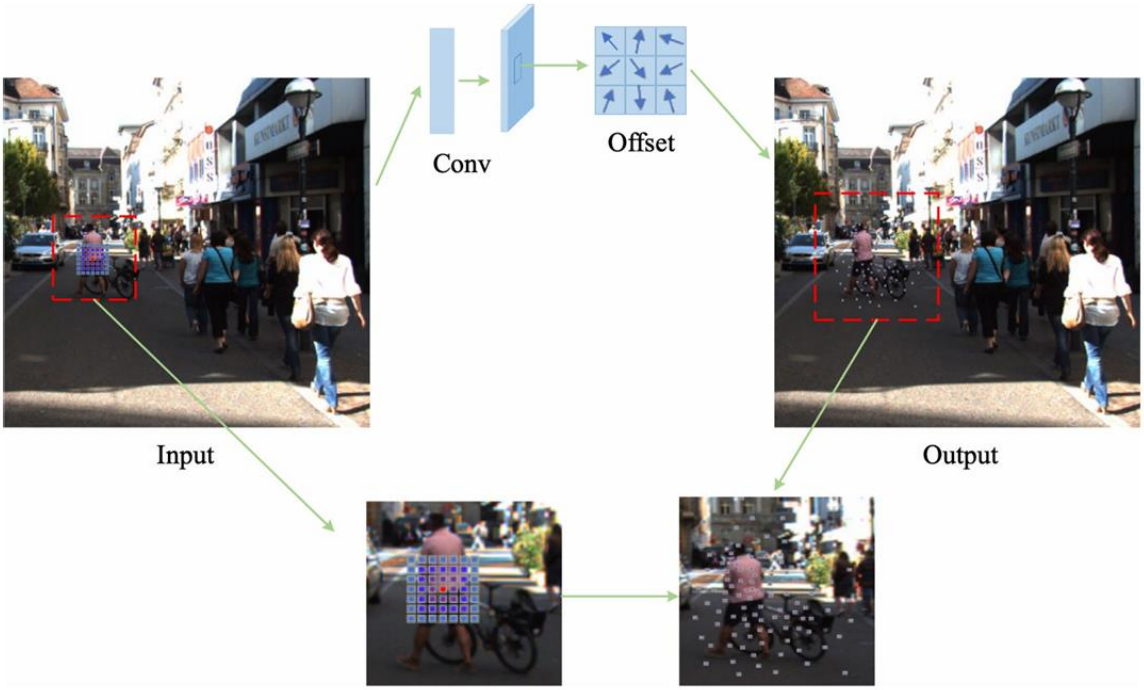


Рисунок 1.6 – Ефект зміщення у деформативно згортковому механізмі

1.1.2.2 Просторовий пірамідний пулінг Fast SSP

Архітектура використовує цю структуру в основному блоці мережі з трьома різними мапами особливостей через три різні розміри ядер для максимального пулінгу: 5x5, 9x9 та 13x13 для вилучення особливостей та подальшого поєднання, яке ефективно збільшує сенсорне поле мережі.

Використання таких ядер породжує проблему зі швидкістю обрахування та точністю, тому швидка версія цієї структури слугує заміною, щоб уникнути їх. Виглядає вона як показано на рис. 1.7.

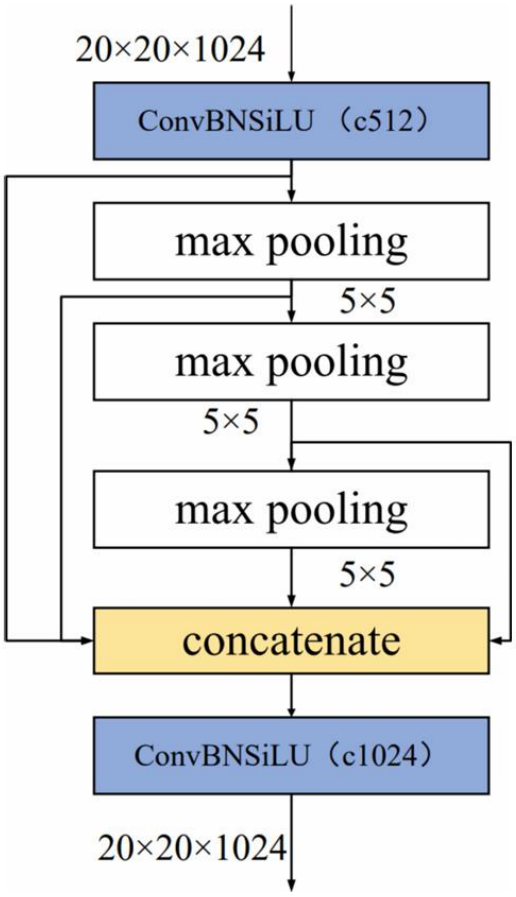


Рис 1.7 – Структура Fast SPP

Вона дозволяє зробити розрахунки трьох згорткових ядер як 5x5 шар пулінгу послідовно, щоб кожні вихідні дані пулінгу стали вхідними для іншого та поєднати особливості під різними змінами в даних при спільній мапі особливостей, яка покращує показник експлуатації параметрів мережі та зменшує складність операції.

1.1.2.3 Координована мережа з attention

Такий вид уваги, де дано два вхідні просторові діапазони пулінгу та використовуються для кодування кожного каналу за горизонтальними та вертикальними координатами, загально можна побачити на рис 1.8.

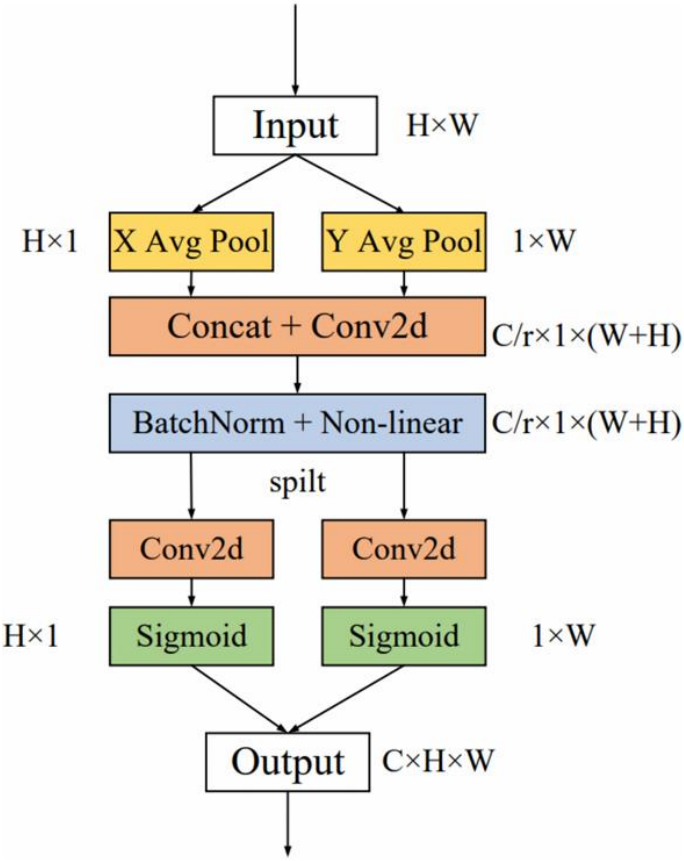


Рисунок 1.8 – Координатний механізм уваги

Вихід для c каналу з висотою h та шириною w може бути описаний наступною формулою 1.12 та 1.13. Після чого особливості поєднуються уздовж двох просторових напрямах для отримання пари проміжний мап особливостей f, що кодують просторову інформацію в горизонтальних та вертикальних напрямках, показано у формулі 1.14.

Після цього використовуються згорткові тензори 1x1 з однаковим каналом F_h та F_w та розраховують ваги уваги для двох просторових напрямів, як й вихід координованої уваги, обрахування відбуваються за формулами 1.15-1.17 відповідно.

Координатна увага може задовольнити потребу в перехопленні довгострокових залежностей уздовж одного напрямку та водночас турбуватися про точну позицію у іншому. Використання цього блоку у вилученні особливостей можна побачити на рис. 1.9.

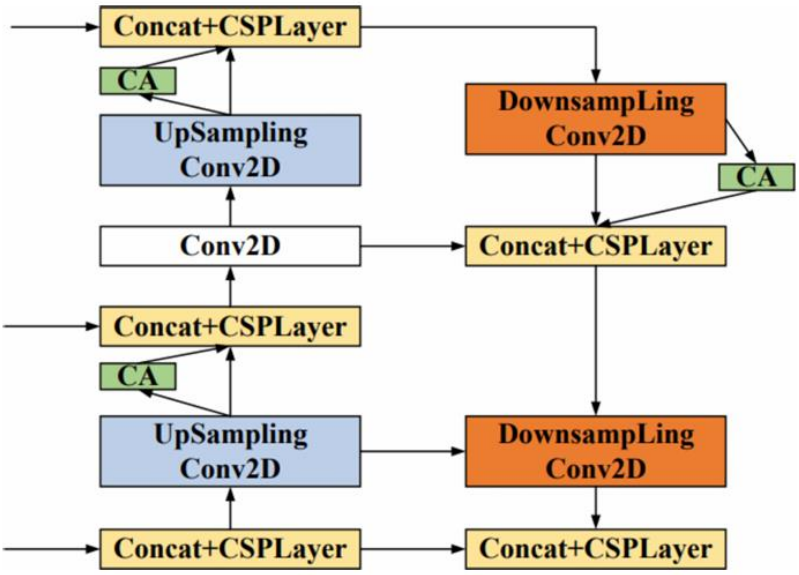


Рисунок 1.9 – Розширена структура вилучення особливостей

1.1.2.4 Регресійна функція втрат OL-IoU

YOLOX використовує динамічне співпадіння позитивної вибірки для отримання особливої точки відповідно до кожного реального кадру та передбаченого кадру з цієї точки. Далі ці два кадри використовуються для розрахунки Intersection over Union (IoU). Однак це не допомагає у точному відображенні перекриття двох кадрів, особливо коли IoU однаковий, але дві різні позиції кадрів показують, що регресійний ефект не однаковий. IoU розраховується наступним чином за формулою 1.12.

Розглянемо механізми, які знадобляться для вирішення проблеми:

• Distance-IoU (DIoU) привносить мінімальну зовнішню прямокутну діагональну дистанцію (eng. the minimum outer rectangular diagonal distance)

між передбаченою та реальною границей для прискорення регресії за евклідовою дистанцією між центроїдами двох границь, які запобігають феномену, що помилка дуже велика для оптимізації двох границь, які розташовані далеко один від одного. Його можна розраховати наступним чином за формулою 1.13.

• Complete IoU (CIoU) додає фактор співвідношення сторін між передбаченими та реальними кадрами на основі DIoU. У цьому випадку враховуючи цей фактор отримуємо наступне за формулами 1.14-1.16.

• Efficient IoU (EIoU) розділяє співвідношення на CIoU та розглядає узгодженість ширини та висоти перекритої частини та ширини та висоти найменшого зовнішнього прямокутника відповідно. Це дозволяє більш чітко виділяти ширину та висоту відносно їх реальної різниці з рівнями впевненості (eng. confidence levels). Розрахування відбуваються за формулою 1.17.

• Overlapping IoU (OL-IoU) узгоджує фактор співвідношення передбаченої рамки до реальної рамки ν_1 та фактор узгодженості аспектно прискорювальної перекритої регресії (eng. consistency factor of the overlapping aspect-accelerated regression) ν_2 на основі EIoU. Якщо ν_1 розраховується як наведено у формулі 1.20, то ν_2 за наступною формулою 1.18.

Тоді OL-IoU буде виглядати наступним чином як це наведено у формулі 1.19.

Логіка $\beta\nu_2$ діє як приближення передбачених кадрів в обох напрямках – ширини та висоти, до реальних кадрів з однаковою швидкістю, тим самим запогібаючи те, що один напрям не може досягти мети у максимізації оптимізації. У той же час, більш логічно прослідковується те, що приближення цільового кадру в одному напрямі може визивати ту ж саму операцію в іншому напрямі. Навіть якщо перекриття збільшить 2D квадратний кут, то це прискорить покращення IoU. На рисунку 1.10 можна

схематично прослідкувати пункт (a) паралельне відношення пов’язане з перетином (eng. intersection-parallel ratio) без додавання βv_2 логіки та (б) показує чи є передбачені кадри більш активними після додавання штрафу ближчими до реального кадру.

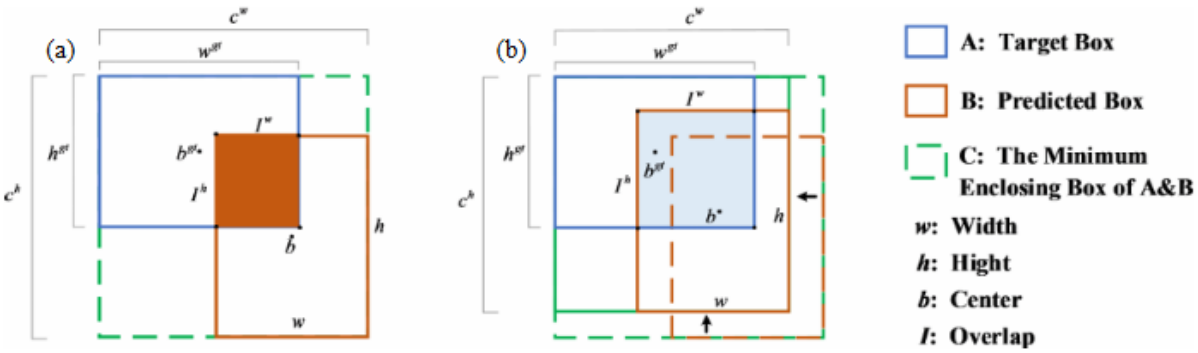


Рисунок 1.10 – Схематична діаграма перекриваючої IoU регресії

Таким чином покращуючи основну архітектуру YOLOX, ми отримаємо OCC-YOLOX (Occluded- YOLOX), яка наведена на рис. 1.11.

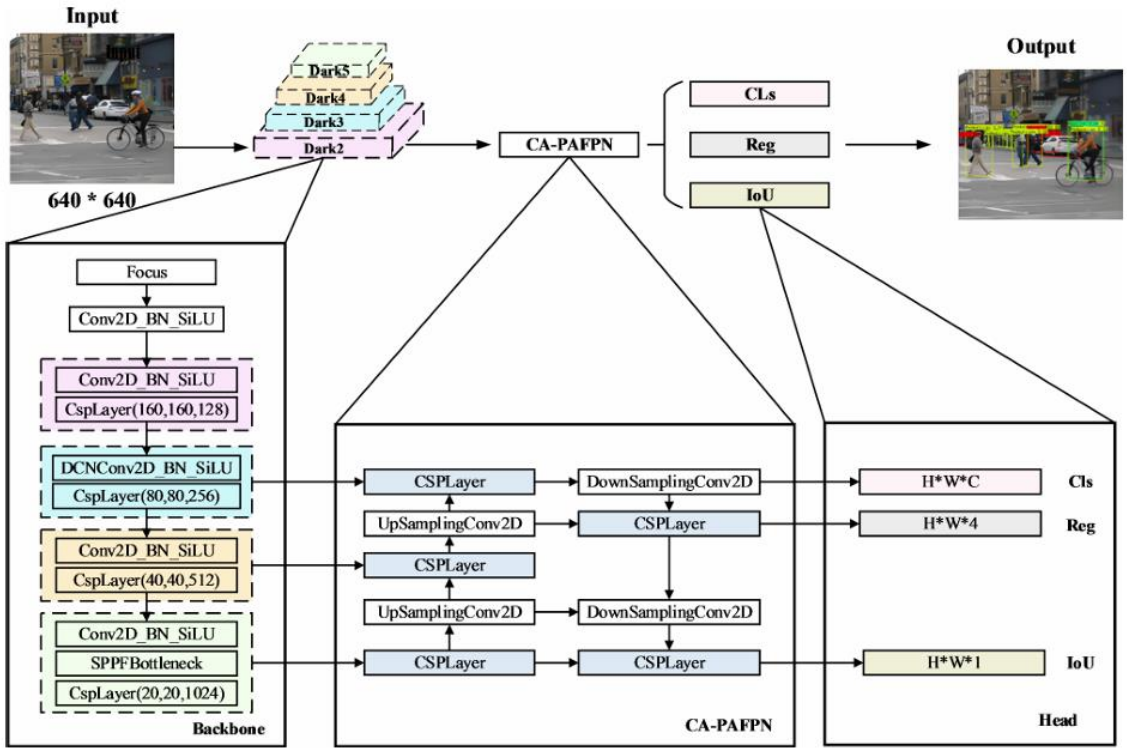


Рис. 1.11 – Структурна модель OCC-YOLOX

1.2 Аналіз додаткових архітектур та механізмів для загальної класифікації

Стандартні механізми для розпізнавання об’єктів у цілому розраховані з врахуванням важких випадків, відрізняються тільки як цей процес відбувається, що й породжує ту саму проблему з оклюзивними даними. Саме цей факт вимагає розглядання усіх модерних архітектур, які працюють у напрямку загальної класифікації об’єктів, що дало б можливість оцінити більш ефективні підходи, які працюють добре в усіх складних ситуаціях. Аналіз таких можливостей може допомоги зрозуміти, які механізми для цього використовувалися раніше, що може розкрити ідею праці окремих компонентів більш детально.

1.2.1 YOLO 26 як архітектурна інновація

З самого початку заснування архітектури у 2016 році вона еволюціонувала через багато різних вдосконалень архітектури, кожна посилається на обмеження її минулої версії у той час інтегруючи переваги зрізання сторін в дизайн нейронних мереж, функції помилок, та ефективне застосування. На відміну від традиційних двоєтапових методів детекції як R-CNN та Faster R-CNN, які відділяли пропозицію регіонів від класифікації, YOLO сформулювала розпізнавання як едину регресійну проблему. Завдяки прямому передбачанню границь та вірогідностей класів за один прохід через згорткову мережу (CNN), YOLO досягла швидкості на рівні реального часу поки зберігши конкурентну точність. Все у 4 версії архітектури було запропоновано Cross-Stage Partial Networks (CSPNet), покращуючи активаційні функції такі як Mish, та продвинуті стратегії тренування

включаючи мозаїчне нарощування даних та CIoU помилку. Ця екосистема міцно зарекомендувала себе як основне сімейство моделей в дослідженнях та застосуваннях у розпізнаванні об’єктів. Ці моделі розглядають самоувагу з багатьма головами (eng. multi-head self-attention), покращену мультимасштабне поєднання (eng. multi-scale fusion), та більш сильніші стратегії регуляризації тренувального процесу. У той час досягають гарних результатів у оцінках, вони також зберегли залежність від Non-Maximum Suppression (NMS) та Distribution Focal Loss (DFL), які впровадили додаткову затримку та проблеми з експортом, особливо для пристроїв з низькою потужністю.

На етапах генерації передбачень 26 версія позбавилася NMS, що дає нативні комплексні передбачення видаляючи головну проблему постобробки у супроводі з зменшенням дисперсії, спрощення налаштування порогу на етапі застосування. Зі сторони регресії, було видалено DFL, налаштувавши дистрибутивне декодування границь в більш легке та підходяще для різної техніки. Також важливі запровадження у тренувальному процесі стабільності у вигляді progressive loss balancing (ProgLoss) та підвищення точності для малих об’єктів у вигляді small-target-aware label assignment (STAL). ProgLoss здатен адаптувати цільові ваги для запобігання домінації легкими прикладами у тренувальному процесі пізніше, у той час як STAL пріоритезує позначання для крихітних та перекритих об’єктів, покращуючи відклик за складних ситуацій, листях, або ж розмиття під час руху, які звичні у повітрі, робототехніці та трансляцій у реальному часі смарт камерами. Оптимізація заснована на MuSGD, гібриді який поєднує узагальнення звичайної SGD з моментумом за Muon-style методами для швидкого та гладкішого сходження та більшої надійності площини при масштабуванні (eng. plateaus across scales).

Окрім розпізнавання об’єктів є й інші можливості, концентруючись на цьому аспекті архітектури можна виділити:

• Anchor-free, замість створення певних завчасно створених кріплень до границь, орієнтується на ключові точки та особливості об’єкту, як центр. Приклад можна побачити на рис. 1.12;

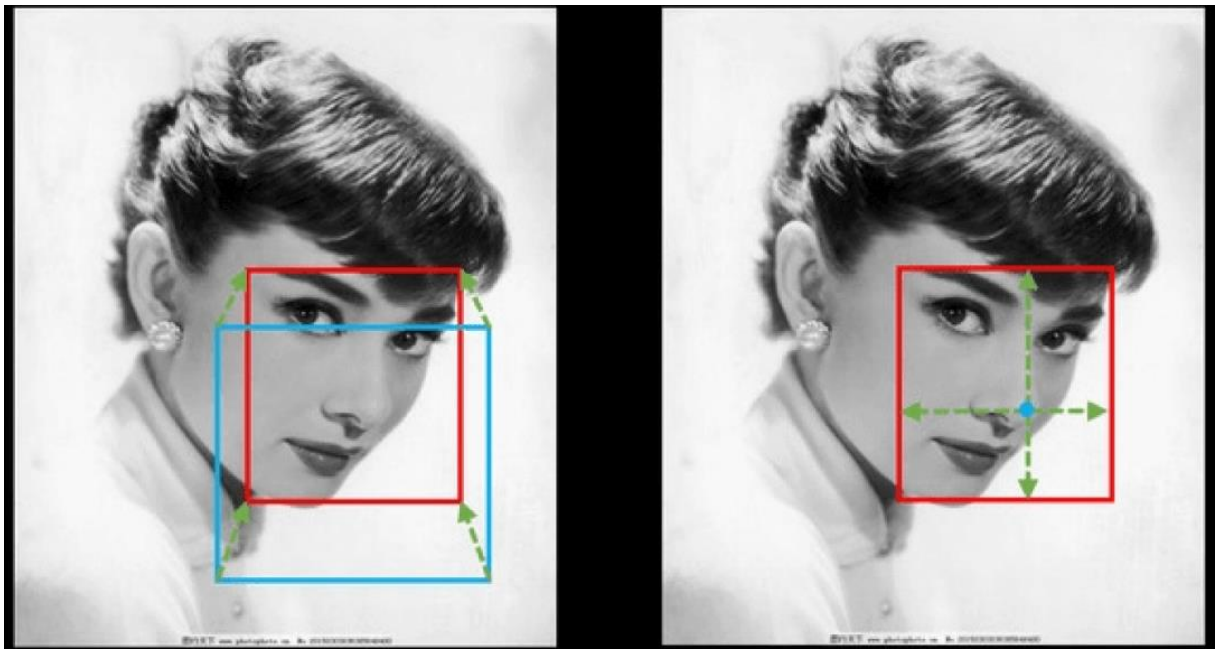


Рисунок 1.12 – Порівняння Anchor-based та Anchor-free

• NMS-free границь та оцінок, що запобігає множинної детекції одного й того ж об’єкту, обираючи тільки найкращу. На рис. 1.13 можна побачити приклад такої поведінки.

Рисунок 1.13 – Принцип механізму NMS

1.2.1.1 Архітектура та її особливості

Процес починається з потрапляння вхідних даних, як відео або зображення, які в першу чергу проходять предобробку такі як зміну розміру та нормалізація даних, які будуть підходящими для передбачання. Всю архітектуру включаючи вище описаний механізм можна спостерігати на рис. 1.14. Потім дані будуть йти у основний блок для отримання особливостей, оброблених ієрархією згорткових мереж. Архітектура також генерує мультимасштабні карти ознак, які зберігають семантичну різноманітність як для великих так й малих об’єктів. Ці особливості пізніше поєднуються в шийному блоці, де інформація використовується якомога ефективніше. Останнім етапом є проведення специфічної обробки для детекції в регресійній голові (eng. regression head), яка створює границі та вірогідності до класів без залежності від NMS.

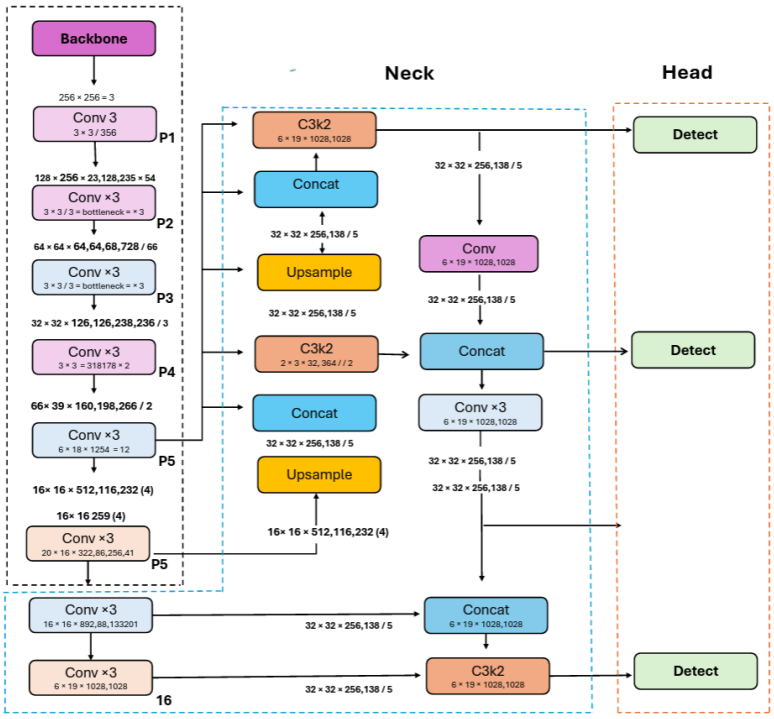


Рисунок 1.14 – Головна архітектурна діаграма YOLO26 для розпізнавання об’єктів

Один із цінних спрощень в архітектурі знаменувала себе видалення DFL модулю, який був розроблений для покращення регресії границь за допомоги прогнозування розподілу імовірностей для координат границь, таким чином забезпечуючи більш точну локалізацію об’єктів. На практиці, DFL потребує спеціалізованої обробки під час отримання передбачень та експорту моделі, які ускладнюють роботу з застосуванням в техніці. З видаленням DFL, архітектура спрощується, роблячи створення границь більш прямим регресійним підходом без втрати ефективності. Більш того, видалення цього модулю значно зменшує затримку отримання передбачень та покращує кроссплатформну сумісність. Це можна побачити на рис. 1.15 в секції (a).

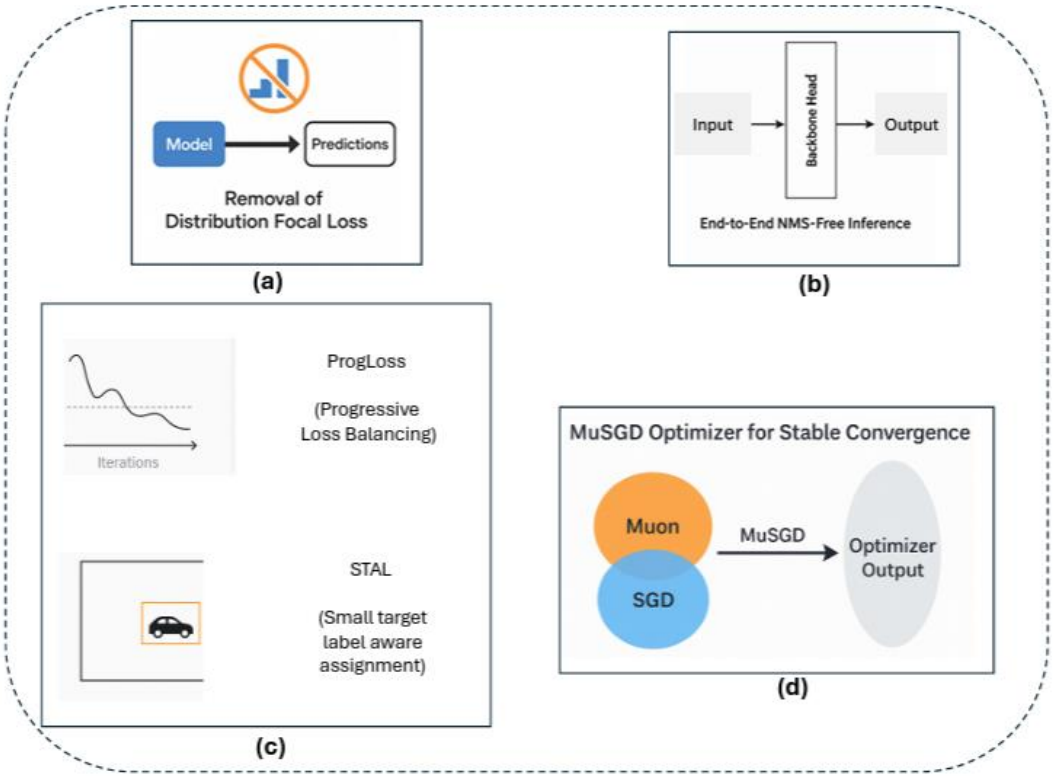


Рисунок 1.15 – Ключові покращення YOLO26

Також окремим покращенням стало комплексне отримування передбачень без NMS механізму, так як він додає додаткову затримку в послідовне виконання та потребує ручного налаштування параметрів таких яг поріг IoU. Ця версія була фундаментально перебудувала механізм передбачень у блоці голови для отримання кращих передбачень границь без необхідності в NMS механізмі. Більш того, його комплексний дизайн не тільки допомагає зменшити складність отримання передбачень, але й позбутися залежності від мануальних порогів, покращуючи інтеграцію в виробничі системи. На рис 1.15 в секції (б) продемонстровано новий механізм. Окрім самої швидкості, такий підхід приносить кращу відтворюваність та портативне застосування у рамках низької потреби у розширеній обробці.

Також раніш обговорені механізми STAL та ProgLoss, які долають проблеми пов’язані з стабільністю та розпізнаванням малих об’єктів. Як вже обговорювалося, вони приносять дуже великий вклад в архітектуру, що відображено на рис 1.15 секція (с), разом вони забезпечують суттєве підвищення точність на датасетах з малими та перекритими об’єктами як Common Objects in Contexts (COCO) та Unmanned Aerial Vehicles (UAV). Таким чином, сучасна архітектура досягла схожого або ліпших покращень з більш легким способом, підходящим до сучасних потребувань ринку штучного інтелекту (ШІ), забезпечуючи надійну детекцію малих об’єктів залишаючи ефективність та портативність сімейства архітектури.

Останньою інновацією стало введення покращеної системи оптимізації MuSGD на основі Stochastic Gradient Descent (SGD) та оптимізатором Muon, яка використовується для оптимізації тренування Large Language Model (LLM). Короткий принцип наведено на рис 1.15 у

секції (d). Оптимізатор використовує надійність та узагальнення здатність SGD поки застосовують властивості Muon, дозволяючи швидку збіжність та більш стабільну оптимізація для різних датасетів. Такий гібридний оптимізатор відображає важливий тренд у глибинному навчанні перехресне перевагу між Natural Language Processing (NLP) та комп'ютерним зором. Він покращує надійність зберігаючи просту філософію архітектури, що у свою чергу свідчить про короткі цикли розробки, менше тренувань та більше спроможності до передбачення у сценаріях застосування.

1.2.2 Оптимізована YOLO NAS для розпізнавання об'єктів у реальному часі

YOLO-NAS є сучасною моделлю для виявлення об'єктів, яка представляє новий етап розвитку сімейства YOLO та поєднує у собі передові підходи до побудови нейронних мереж. Її ключовою особливістю є використання технології пошуку архітектури нейронних мереж (Neural Architecture Search, NAS), що дозволяє автоматично створювати ефективні архітектури без необхідності ручного проєктування. Завдяки цьому вдається подолати обмеження попередніх моделей і досягти значного покращення співвідношення між точністю та затримкою. Крім того, YOLO-NAS підтримує квантизацію, що робить її особливо придатною для практичного застосування в умовах обмежених ресурсів і реального часу. Для досягнення високої продуктивності модель використовує вибіркову квантизацію та блоки, обізнані з квантизацією (eng. quantization-aware blocks). Такий підхід дозволяє зменшити втрати точності навіть після перетворення моделі у формат INT8. Завдяки цьому YOLO-NAS демонструє кращі результати порівняно з іншими моделями сімейства YOLO, зберігаючи високу точність при значному зменшенні обчислювальних витрат. Основу архітектури становлять компоненти Quantization-Aware

RepVGG (QA-RepVGG), які забезпечують ефективну репараметризацію та підтримку післятренувальної квантизації (eng post-training quantization, PTQ). Додаткові модулі, такі як QSP і QCI, сприяють стабільній роботі моделі в умовах обмеженої розрядності. Загалом архітектура виглядає як це наведено на рис. 1.16.

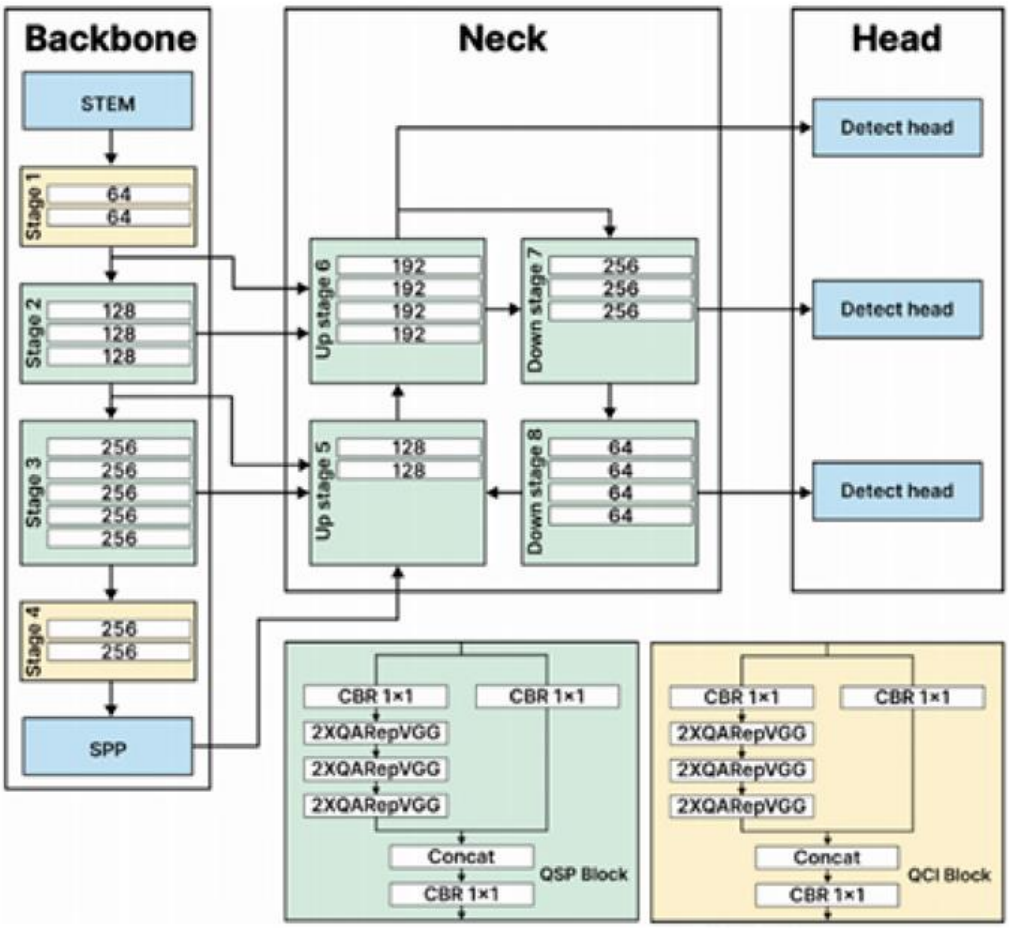


Рисунок 1.16 – Архітектура YOLO-NAS

Технологія NAS, яка лежить в основі YOLO-NAS, є підгалуззю автоматичного машинного навчання (AutoML) і дозволяє систематично враховувати такі фактори, як точність прогнозування, швидкість обчислень та доступність ресурсів. Це значно спрощує процес розробки моделей і дозволяє отримувати рішення, які часто перевершують створені вручну. У задачах комп’ютерного зору NAS уже довела свою ефективність, особливо

в задачах класифікації зображень та виявлення об’єктів. Вона зменшує потребу в експертних знаннях і дозволяє автоматично знаходити оптимальні конфігурації мережі, що є важливим для застосувань у мобільних пристроях та системах реального часу.

Важливим етапом оптимізації YOLO-NAS є обрізання моделей (eng. pruning), яке спрямоване на зменшення надлишковості нейронної мережі. Основна ідея цього підходу полягає у визначенні менш важливих параметрів за допомогою оцінки значущості ознак (eng. feature score) та їх подальшому видаленні. Це дозволяє створити компактнішу модель, яка потребує менше пам’яті та обчислювальних ресурсів. Процес pruning включає три основні етапи: навчання розрідженості (eng. sparsity learning), обрізання та калібрування (eng. fine-tuning). На першому етапі модель навчається визначати менш значущі параметри, після чого вони видаляються, а на завершальному етапі виконується перенавчання для відновлення продуктивності. Обрізання може впливати не лише на окремі ваги, а й на структуру мережі, включаючи видалення цілих фільтрів або шарів. Вибір компонентів для видалення здійснюється на основі показника суми абсолютних значень ваг (eng. sum absolute weights, SAW), де найменші значення відповідають найменш значущим ознакам. Процес виконується ітеративно, з оцінкою втрат після кожного кроку. Якщо втрата перевищує допустимий поріг, процес зупиняється, і модель проходить додаткове навчання. Такий підхід дозволяє зберегти баланс між зменшенням складності та точністю моделі. У випадку YOLO-NAS pruning прихованих шарів дозволяє значно зменшити розмір моделі, хоча це може призводити до певного зниження точності, яке компенсується за допомогою перенесення навчання (eng. transfer learning).

Ще одним важливим компонентом оптимізації є використання алгоритму штучної бджолиної колонії (eng. Artificial Bee Colony, ABC) для налаштування гіперпараметрів. Гіперпараметри визначають поведінку

моделі під час навчання і включають такі параметри, як швидкість навчання, розмір пакета, коефіцієнт регуляризації та параметри аугментації даних. Алгоритм ABC, натхненний поведінкою бджіл, дозволяє ефективно досліджувати простір можливих рішень і знаходити оптимальні конфігурації. У межах ABC використовуються три типи агентів: робочі бджоли, спостерігачі та розвідники. Робочі бджоли досліджують поточні рішення, спостерігачі обирають найкращі з них на основі ймовірності, а розвідники шукають нові рішення випадковим чином. Така структура забезпечує баланс між глобальним пошуком і локальною оптимізацією. У дослідженні використовується вдосконалена версія алгоритму – Enhanced ABC (EABC), яка покращує швидкість збіжності та зменшує ризик передчасного застрягання в локальних мінімумах. Оптимізація за допомогою ABC базується на функції якості, що враховує як точність (mAP), так і затримку під час інференсу. Це дозволяє знайти компроміс між точністю та швидкістю, що є критично важливим для задач реального часу. Результати експериментів показали, що використання ABC дозволяє підвищити точність моделі приблизно на 4.1% порівняно зі стандартними налаштуваннями, а також забезпечує більш стабільні результати. У порівнянні з випадковим пошуком і ручним налаштуванням, ABC демонструє кращу ефективність і узгоджується з принципами теореми «No Free Lunch» (NFL), яка підкреслює важливість адаптації алгоритмів до конкретних задач.

Додатковим фактором підвищення ефективності YOLO-NAS є використання функції активації Mish. Вона є гладкою, неперервною та немонотонною, що забезпечує стабільніше навчання та кращий потік градієнтів. На відміну від ReLU, яка обнуляє від’ємні значення, Mish зберігає частину негативної інформації, що сприяє кращому узагальненню моделі. Це дозволяє уникнути проблеми «мертвих нейронів» і покращує здатність моделі адаптуватися до нових даних. Порівняно з іншими

функціями активації, такими як Swish, Mish забезпечує кращий баланс між обчислювальною ефективністю та точністю. Вона сприяє більш стабільній збіжності під час навчання та підвищує якість виявлення об’єктів, особливо в складних умовах. Використання Mish у YOLO-NAS дозволяє досягти кращих результатів без значного збільшення обчислювальних витрат.

1.2.2.1 Запропонована оптимізована модель YOLO-NAS

У даній роботі запропоновано вдосконалену та оптимізовану версію моделі YOLO-NAS, яка базується на оригінальній архітектурі, але містить низку суттєвих покращень. Основна мета такого підходу – підвищення загальної точності виявлення об’єктів, а також вирішення типових проблем, що виникають у задачах комп’ютерного зору, зокрема перекриття об’єктів та складні умови освітлення.

Оригінальна модель YOLO-NAS слугує базовою архітектурою, на основі якої будується вдосконалена версія. Запропонована модель включає додаткові структурні зміни, що дозволяють більш ефективно витягувати ознаки та покращувати якість передбачень. Глобальна структура моделі демонструє інтеграцію кількох сучасних підходів, які разом забезпечують більш високу продуктивність, як це зображено на рис. 1.17.

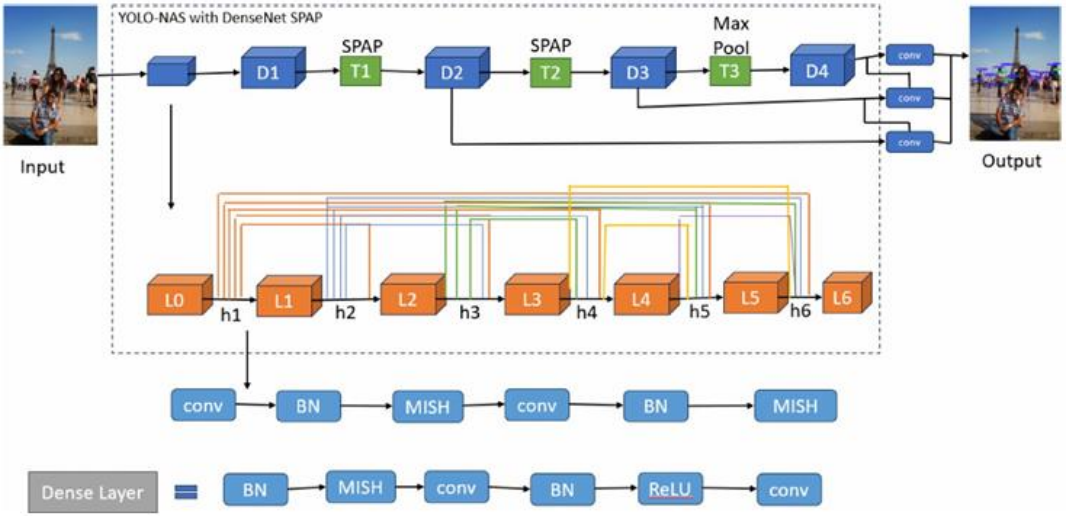


Рисунок 1.17 – Загальна запропонована оптимізована калібрована архітектура YOLO-NAS

Одним із ключових елементів є використання DenseNet – щільно з’єднаної нейронної мережі, яка сприяє кращому поширенню градієнтів і повторному використанню ознак. Це дозволяє уникнути втрати інформації під час проходження через глибокі шари мережі. Крім того, у модель інтегровано механізм Spatial Pyramid Average Pooling (SPAP), який забезпечує отримання ознак на різних масштабах і покращує здатність моделі працювати з об’єктами різного розміру.

Ще одним важливим компонентом є використання функції активації MISH, яка замінює традиційні підходи на кшталт ReLU. Завдяки своїй гладкості та здатності зберігати негативні значення, MISH забезпечує кращу передачу градієнтів і стабільніше навчання. У поєднанні з методом обрізання (pruning) це дозволяє досягти високої ефективності без значного збільшення обчислювальних витрат.

Метод pruning, застосований у цій моделі, сприяє зменшенню кількості параметрів і глибини мережі. Це робить модель легшою та швидшою, що особливо важливо для застосувань у реальному часі. Водночас завдяки додатковому налаштуванню вдається зберегти або навіть покращити точність виявлення.

Архітектура моделі побудована таким чином, щоб забезпечити ефективне витягування багатомасштабних ознак (eng. multi-scale features). Це досягається шляхом стратегічного з’єднання різних компонентів мережі, що дозволяє обробляти інформацію з різних рівнів деталізації. У результаті модель краще розпізнає як великі, так і дрібні об’єкти, навіть у складних умовах.

Особливу увагу приділено скороченню часу передбачень що є критичним для систем реального часу. Завдяки оптимізованій архітектурі модель здатна швидко обробляти зображення, не жертвуючи при цьому точністю. Це робить її придатною для використання в таких сферах, як відеоспостереження, автономні транспортні засоби або мобільні додатки.

Крім того, запропонована модель адаптована до роботи в умовах, де об’єкти можуть бути частково приховані або освітлення є нестабільним. Це досягається завдяки покращеній здатності до узагальнення та використанню більш гнучких механізмів обробки ознак.

1.2.2.2 Оптимізована функція Mish

Оптимізована функція активації MISH є важливим компонентом підвищення продуктивності моделі YOLO-NAS і використовується як частина комплексної архітектурної стратегії. Вона інтегрується разом із такими підходами, як Spatial Pyramid Average Pooling (SPAP) і DenseNet, що дозволяє максимально реалізувати її переваги порівняно з традиційними функціями активації, зокрема ReLU. Вдосконалена функція виглядає наступним чином за формулою 1.20.

У структурі мережі застосовуються блоки Conv-BatchNorm-MISH (CBM), де функція MISH виконує роль нелінійності. Це забезпечує більш плавний потік градієнта та зменшує ризик різких змін активації. У типовій конфігурації спочатку використовується згортка 1×1 разом із Batch Normalization і MISH для обробки вхідних даних, після чого застосовується шар Conv-BatchNorm-ReLU для масштабування сигналу, а далі – згортка 3×3 для глибшого виділення ознак.

Додатково результати згорткових шарів можуть розділятися на частини з подальшим об’єднанням, що покращує представлення ознак. MISH характеризується гладкістю, немонотонністю та ефективним

поширенням градієнтів, що сприяє кращій збіжності. Поєднання MISH і ReLU, зокрема в DenseNet-блоках, забезпечує баланс між стабільністю, обчислювальною ефективністю та узагальнюючою здатністю моделі.

1.2.2.3 DenseNet із Spatial Pyramid Average Pooling

Однією з ключових проблем класичних моделей YOLO є поступова втрата інформації про ознаки під час проходження сигналу через глибокі шари мережі. Для вирішення цієї проблеми у запропонованій роботі використовується архітектура DenseNet (щільно з'єднана нейронна мережа). Її основна перевага полягає в тому, що кожен шар отримує на вхід не лише вихід попереднього шару, а й усіх попередніх шарів. Така щільна зв'язаність забезпечує сильний потік градієнтів і дозволяє ефективно повторно використовувати ознаки. Це, у свою чергу, сприяє формуванню більш багатих і різноманітних представлень даних.

У межах DenseNet кожен шар (позначений як L_i) складається з операцій згортки, нормалізації та функції активації MISH. Вихід кожного шару формується на основі комбінації попередніх ознак, що дозволяє накопичувати інформацію та уникати її втрати. Крім того, кожен шар може генерувати певну кількість карт ознак. Загалом така поведінка показана у формулах 1.21-1.22.

Кілька таких щільних блоків об'єднуються для створення повноцінної мережі DenseNet. Між цими блоками розташовані так звані перехідні шари (transition layers), які включають операції згортки та пулінгу. Вони допомагають зменшити розмірність даних і контролювати складність моделі.

Важливим доповненням до DenseNet у цій роботі є використання Spatial Pyramid Average Pooling (SPAP). Цей механізм замінює традиційний max pooling на average pooling у структурі просторової піраміди. Основна

ідея полягає в тому, щоб отримувати ознаки зображення на різних масштабах шляхом усереднення значень, а не вибору лише максимальних.

SPAP генерує кілька карт ознак різного розміру, наприклад 1×256, 4×256 та 9×256. Ці карти потім об’єднуються, утворюючи більш повне представлення зображення. У результаті формується вектор ознак, який передається до наступних шарів мережі. Такий підхід дозволяє моделі краще враховувати як глобальні, так і локальні характеристики об’єктів.

Використання average pooling замість max pooling має важливу перевагу: воно згладжує зображення та зменшує вплив шуму або різких змін освітлення. Це особливо корисно при роботі з різними наборами даних, де умови зйомки можуть суттєво відрізнятися. На відміну від max pooling, який зосереджується лише на найяскравіших пікселях, average pooling враховує загальний контекст.

SPAP інтегрується в перші рівні DenseNet, що дозволяє моделі з самого початку формувати багатомасштабні ознаки. Це покращує здатність до виявлення об’єктів різного розміру, включаючи дрібні або частково перекриті.

Крім того, така архітектура сприяє кращій регуляризації моделі та зменшує ризик перенавчання. Завдяки ефективному використанню ознак і покращеному градієнтному потоку модель стає більш стійкою до варіацій у даних.

1.2.2.4 DenseNet-SPAP об’єднання

Об’єднання DenseNet і SPAP у єдину архітектуру є важливим етапом підвищення ефективності моделі YOLO-NAS. Основна ідея такого злиття (fusion) полягає в поєднанні переваг обох підходів: здатності DenseNet до ефективного повторного використання ознак і можливості SPAP витягувати багатомасштабні просторові характеристики.

У цій архітектурі DenseNet використовується як backbone, формуючи ієрархічні ознаки різних рівнів – від низькорівневих до високорівневих. Завдяки щільним зв’язкам між шарами забезпечується ефективна передача інформації, яка далі надходить до модуля SPAP. Він виконує багатомасштабне середнє пулінгування (eng. multi-scale average pooling), що дозволяє зберігати контекст навіть для об’єктів різного розміру.

У результаті формується представлення ознак, яке поєднує локальну та глобальну семантичну інформацію. Це представлення передається до detection head YOLO-NAS, де виконується фінальне виявлення об’єктів. Такий підхід підвищує точність і стійкість моделі, зокрема в умовах перекриття об’єктів або змінного освітлення.

Крім того, таке об’єднання покращує чутливість до просторової роздільної здатності та узагальнюючу здатність моделі. При цьому обчислювальні витрати зростають незначно, що дозволяє зберігати баланс між швидкістю та продуктивністю в задачах реального часу.

1.2.2.5 Алгоритм ABC разом із YOLO-NAS

Алгоритм ABC інтегрується в процес оптимізації моделі YOLO-NAS на кількох етапах, забезпечуючи покращення як точності, так і ефективності обчислень. Оскільки оптимізація гіперпараметрів є ресурсоємним процесом, у дослідженні використовується версія моделі YOLO-NAS-L, яка характеризується швидшим навчанням порівняно з іншими варіантами.

На початковому етапі в систему завантажується користувацький набір даних, який використовується як для навчання, так і для тестування моделі. Далі визначається функція якості (fitness function), яка оцінює продуктивність моделі. У цьому випадку використовується метрика mAP (mean Average Precision), яка є стандартом для задач виявлення об’єктів і дозволяє комплексно оцінити точність.

Далі алгоритм ABC використовується для пошуку оптимальних значень гіперпараметрів, таких як швидкість навчання (learning rate), розмір партії (batch size), поріг впевненості (confidence threshold) та інші. У процесі ітераційного пошуку ABC генерує різні варіанти параметрів і оцінює їх за допомогою функції mAP за формулою 1.24. Таким чином, алгоритм досліджує простір рішень і поступово наближається до оптимальної конфігурації.

Паралельно з оптимізацією гіперпараметрів застосовується метод обрізання, який зменшує кількість параметрів моделі та підвищує її обчислювальну ефективність. Це дозволяє створити легшу модель без суттєвої втрати точності. Після знаходження оптимальної конфігурації модель повторно навчається та додатково оптимізується.

Важливим аспектом є налаштування параметрів самого алгоритму ABC. Зокрема, визначаються такі значення, як розмір популяції, ліміт спроб, кількість оцінок та інші. У дослідженні оптимальними виявилися значення: 70 для популяції, 90 для ліміту та 7000 для кількості оцінок. Проте ці параметри можуть змінюватися залежно від розміру та складності набору даних. Алгоритм працює в межах заданих діапазонів гіперпараметрів. Наприклад, для швидкості навчання можуть бути встановлені верхня та нижня межі (від 1e-1 до 1e-5). Користувач може звузити ці межі для прискорення процесу оптимізації, якщо має попередній досвід або знання про задачу.

Метрика mAP використовується не лише як критерій оцінки, але й як цільова функція для оптимізації. У деяких випадках можуть також використовуватися додаткові метрики, такі як F1-score, залежно від вимог задачі.

1.2.3 RT-DETRv4 як компроміс між Vision Foundation Models та ресурсомісткими детекторами

У даній роботі основна увага приділяється застосуванню запропонованого підходу до детекторів об’єктів у реальному часі, побудованих на архітектурі DETR, зокрема до моделі RT-DETR. Запропонована структура включає кілька взаємопов’язаних модулів, які інтегруються в базову архітектуру та підвищують її ефективність. Основою виступає RT-DETR із гібридним енкодером, що забезпечує обробку багатомасштабних ознак. Загальну архітектуру наведено на рис. 1.18.

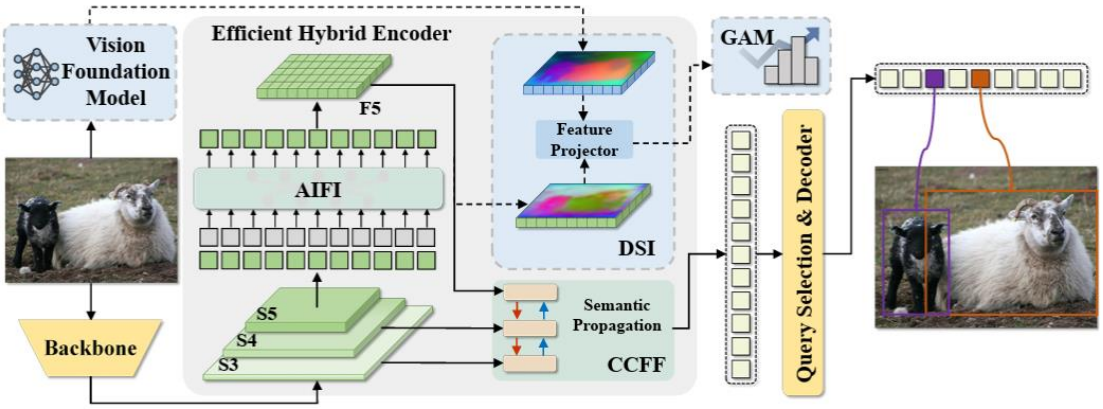


Рисунок 1.18 – Архітектура RT-DETRv4

Енкодер працює з картами ознак S3, S4 і S5, отриманими з CNN-бекбону, та складається з двох ключових компонентів:

1. Attention-based Intra-scale Feature Interaction (AIFI), який застосовує self-attention до найглибшої карти $S_5 \in \mathbb{R}^{H/32 \times W/32 \times C_5}$. Це дозволяє ефективно захоплювати глобальний контекст і довгострокові залежності при збереженні обчислювальної ефективності. У результаті формується збагачена ознака F5 із високим рівнем семантичної інформації.

2. CNN-based Cross-scale Feature Fusion (CCFF), який забезпечує злиття ознак різних масштабів. Зокрема, ознака F_5 поєднується з S_3 і S_4, що

дозволяє передати високорівневу семантику до нижчих рівнів. У результаті формуються багатомасштабні представлення P_3, P_4 та P_5, які використовуються декодером для детекції об'єктів.

Важливим аспектом є якість ознаки F_5, оскільки вона виступає основним носієм глобальної семантики. Це формує проблему «семантичного вузького місця F_5» (eng. «F_5 Semantic Bottleneck»), яка виникає через те, що модуль AIFI отримує лише непрямий нагляд через градієнти, що проходять через декодер і CCFF.

Для усунення цього обмеження запропоновано модуль Deep Semantic Injector (DSI), який використовується під час навчання. Він забезпечує пряме семантичне узгодження F_5 із представленнями, отриманими з VFM, що покращує якість ознак і сприяє ефективнішому поширенню градієнтів.

Додатково застосовується механізм Gradient-guided Adaptive Modulation (GAM), який динамічно регулює вагу функції втрат λ на основі градієнтної статистики. Це дозволяє збалансувати процес детекції та семантичного нагляду. У результаті запропонований підхід усуває вузьке місце F_5 та підвищує загальну продуктивність RT-DETR.

1.2.3.1 Deep Semantic Injector

Для подолання проблеми «семантичного вузького місця F5» запропоновано модуль Deep Semantic Injector (DSI), який виступає легким компонентом і використовується виключно на етапі навчання. Його основне призначення полягає в забезпеченні прямого семантичного нагляду над картою ознак F_5 шляхом її узгодження з представленнями, отриманими від потужної моделі-вчителя $\mathcal{T}$.

У процесі роботи модель-вчитель генерує високоякісне представлення ознак FT, яке містить глибоку семантичну інформацію. Зазвичай такі представлення формуються архітектурами типу Vision

Transformer (ViT), що ефективно захоплюють глобальний контекст і довгострокові залежності. Проте між ознаками F_5 та $F_{\mathcal{T}}$ існують відмінності у просторовій роздільній здатності та кількості каналів, що ускладнює їх безпосереднє порівняння.

Для вирішення цієї проблеми використовується компонент Feature Projector $\mathcal{P}$, який виконує вирівнювання (eng. alignment) представлень. Спочатку вихід моделі-вчителя, представлений у вигляді послідовності токенів, перетворюється у двовимірну структуру, після чого масштабується до розмірів F_5. Паралельно виконується адаптація каналів F_5 до семантичного простору $F_{\mathcal{T}}$, що забезпечує коректне узгодження. Повний процес проекції можна описати за формулами 1.25-1.27.

У межах DSI застосовуються різні стратегії інтеграції семантичної інформації, зокрема узгодження на різних рівнях або безпосередньо на виході AIFI. Важливою особливістю є відсутність відокремлення градієнтів, що дозволяє сигналу помилки поширюватися через всю мережу. Для оптимізації використовується функція втрат cosine similarity loss, яка максимізує подібність між F_5 і $F_{\mathcal{T}}$.

1.2.3.2 Gradient-guided Adaptive Modulation

Для забезпечення стабільного та адаптивного семантичного нагляду в процесі навчання запропоновано механізм Gradient-guided Adaptive Modulation (GAM). Його основне призначення динамічне регулювання внеску семантичного компонента (зокрема модуля AIFI та втрати DSI) у загальну оптимізацію моделі. На відміну від традиційних підходів, які орієнтуються лише на значення функції втрат, GAM використовує інформацію про градієнти, що дозволяє точніше контролювати процес навчання.

Ключова ідея GAM полягає в тому, щоб оцінювати відносний внесок кожного компонента моделі через норму його градієнтів. Для цього на кожному кроці навчання обчислюється L1-норма градієнтів для основних частин моделі, включаючи бекбон, модуль AIFI, модуль CCFF і декодер. Після цього визначається загальна величина градієнтів і обчислюється частка, яку становить внесок AIFI відносно всієї моделі.

Ця частка, градієнтне співвідношення, відображає, наскільки сильно AIFI впливає на процес оптимізації на поточному етапі. Далі значення усереднюються по всіх кроках у межах однієї епохи, що дозволяє отримати стабільну оцінку ролі AIFI в навчанні.

У механізмі GAM використовуються два важливі гіперпараметри:

1. **Цільове співвідношення ($\boldsymbol{\rho}$)**, яке визначає бажаний рівень впливу AIFI на оптимізацію;

2. **Інтервал допуску ($\boldsymbol{\delta}$)**, який задає допустимі межі відхилення від цільового значення. Разом вони формують діапазон $[\rho - \delta, \rho + \delta]$, у межах якого внесок AIFI вважається оптимальним.

Після завершення кожної епохи GAM перевіряє, чи поточне середнє значення градієнтного співвідношення знаходиться в межах цього інтервалу. Якщо так – ваговий коефіцієнт λ, який контролює вплив DSI, залишається без змін. Якщо ж значення виходить за межі, λ коригується таким чином, щоб у наступній епосі наблизити внесок AIFI до бажаного діапазону.

Важливою особливістю цього механізму є те, що корекція λ спрямована не до центру інтервалу, а до його меж. Це пояснюється тим, що лише частина градієнтів AIFI походить від DSI, тому така стратегія забезпечує більш стабільну збіжність і запобігає коливанням під час навчання.

Гіперпараметри ρ і δ відіграють важливу роль у керуванні динамікою навчання. Значення ρ визначає бажану інтенсивність семантичного нагляду,

тоді як δ контролює баланс між швидкістю адаптації та стабільністю. Менше значення δ дозволяє швидше реагувати на зміни, але може призвести до нестабільності. Більше значення, навпаки, забезпечує плавнішу, але повільнішу адаптацію.

Практичні результати показують, що використання GAM дозволяє досягти стабільної збіжності моделі без необхідності додаткового складного налаштування. Механізм автоматично регулює баланс між основною задачею детекції та семантичним узгодженням, що значно спрощує процес навчання.

2 РОБОТА З АРХІТЕКТУРАМИ ТА КОМПОНЕНТАМИ ДЛЯ ВИРІШЕННЯ ПРОБЛЕМИ ЧАСТКОВО ВИДИМИХ ОБ'ЄКТІВ

2.1 Огляд архітектур та спроможності до вирішення поставленої задачі

Звісно, що концентрація тільки на аспекті перекритості об'єктів не є фактором успіху в загальному розпізнаванні об'єктів. Саме тому постає питання про принципи, які можуть бути більш доцільними для такого розпізнавання та до цього ж бути придатними до такого роду класифікації у реальному часі. В основному для загального розуміння реалізації та використання архітектур є необхідність у з'ясуванні їх специфіки роботи та наявних механізмів взаємодії, що й будуть описані в цьому розділі.

Окремої уваги потребує опис типів для вирішення проблеми розпізнавання об'єктів. Усі вони маючи свої види взаємодії мають різну структуру компонентів взаємодія яких впливає на їх унікальність в підході. На рис. 2.1 можна спостерігати більшість відомих видів архітектур для трьох типів.

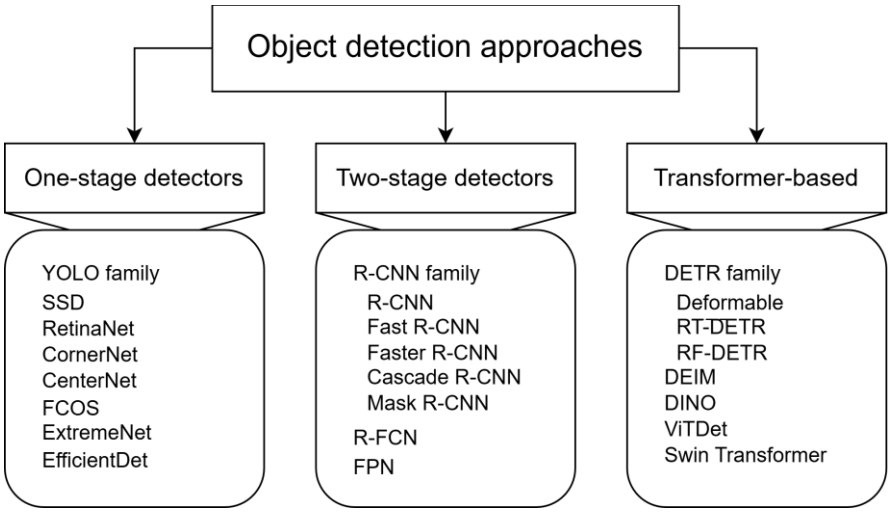


Рисунок 2.1 – Типи детекторів та види архітектур

Важливо також підмітити, що у кожній реалізації була використана імплементація з найбільшої кількістю параметрів для більш доцільного порівняння та дослідження максимальних спроможностей у межах заданої проблеми.

За допомогою JSON/YAML-файлу визначається архітектура, яка формує обчислювальний граф динамічно, використовуючи універсальний механізм побудови моделі, де кожен шар описується через джерело вхідних даних, кількість повторень, тип модуля та набір параметрів. У результаті архітектура у деяких випадках не жорстко закодована у програмному коді, а створюється під час ініціалізації моделі шляхом послідовного інтерпретування конфігурації.

2.1.1 Принцип роботи YOLO

Алгоритм виявлення об’єктів YOLO став важливим проривом у сфері комп’ютерного зору та виявлення об’єктів у реальному часі. На відміну від попередніх підходів, він об’єднав процеси локалізації об’єктів і їх класифікації в межах однієї нейронної мережі, що значно зменшило обчислювальну складність і підвищило швидкість роботи систем. Архітектура YOLO передбачає поділ зображення на сітку, де кожна комірка одночасно прогнозує координати об’єктів і ймовірність належності до певного класу, забезпечуючи наскрізне навчання моделі.

У сфері автономного транспорту та безпеки руху YOLO широко використовується для аналізу навколишнього середовища в реальному часі. Алгоритм допомагає виявляти пішоходів, транспортні засоби, дорожні знаки та потенційні небезпеки, що сприяє уникненню зіткнень і підвищує ефективність керування як наземним транспортом, так і авіаційними системами. Завдяки швидкому опрацюванню даних технологія сприяє зменшенню кількості аварій і підвищенню загального рівня безпеки.

У медицині YOLO застосовується як інструмент підтримки лікарів під час діагностики. Модель використовується для виявлення онкологічних захворювань, аналізу зображень шкіри та автоматичного розпізнавання лікарських препаратів, що допомагає підвищити точність діагнозів і оптимізувати лікувальні процеси.

Промислові підприємства застосовують YOLO для автоматизованого контролю якості продукції, виявлення дефектів поверхонь і моніторингу технічного стану конструкцій. Це дозволяє підвищити надійність виробництва та забезпечити стабільну якість продукції.

Крім того, технологія активно використовується у системах відеоспостереження, сільському господарстві та дистанційному зондуванні Землі. Вона допомагає виявляти підозрілі дії, аналізувати стан посівів, розпізнавати шкідників і досліджувати супутникові знімки для міського планування й екологічного моніторингу. Загалом YOLO демонструє універсальність і значний внесок у вирішення сучасних технологічних і суспільних завдань.

Загально можна відмітити наступний прогрес у покращені архітектури, що наведено на рис 2.2.

У роботі також наявна реалізація 13 версії архітектури зі значними покращеннями у вигляді Full-Pipeline Feature Aggregationand-Distribution (FullPAD) та Hypergraph-based Adaptive Correlation Enhancement (HyperACE) механізмів для покращення поширення градієнту та розпізнавання об’єктів. Наразі, у розділі 1, ми також познайомилися з його останньою версією та вдосконаленнями деяких ключових моментів, які прискорили обробку даних пов’язану з його внутрішніми механізмами.

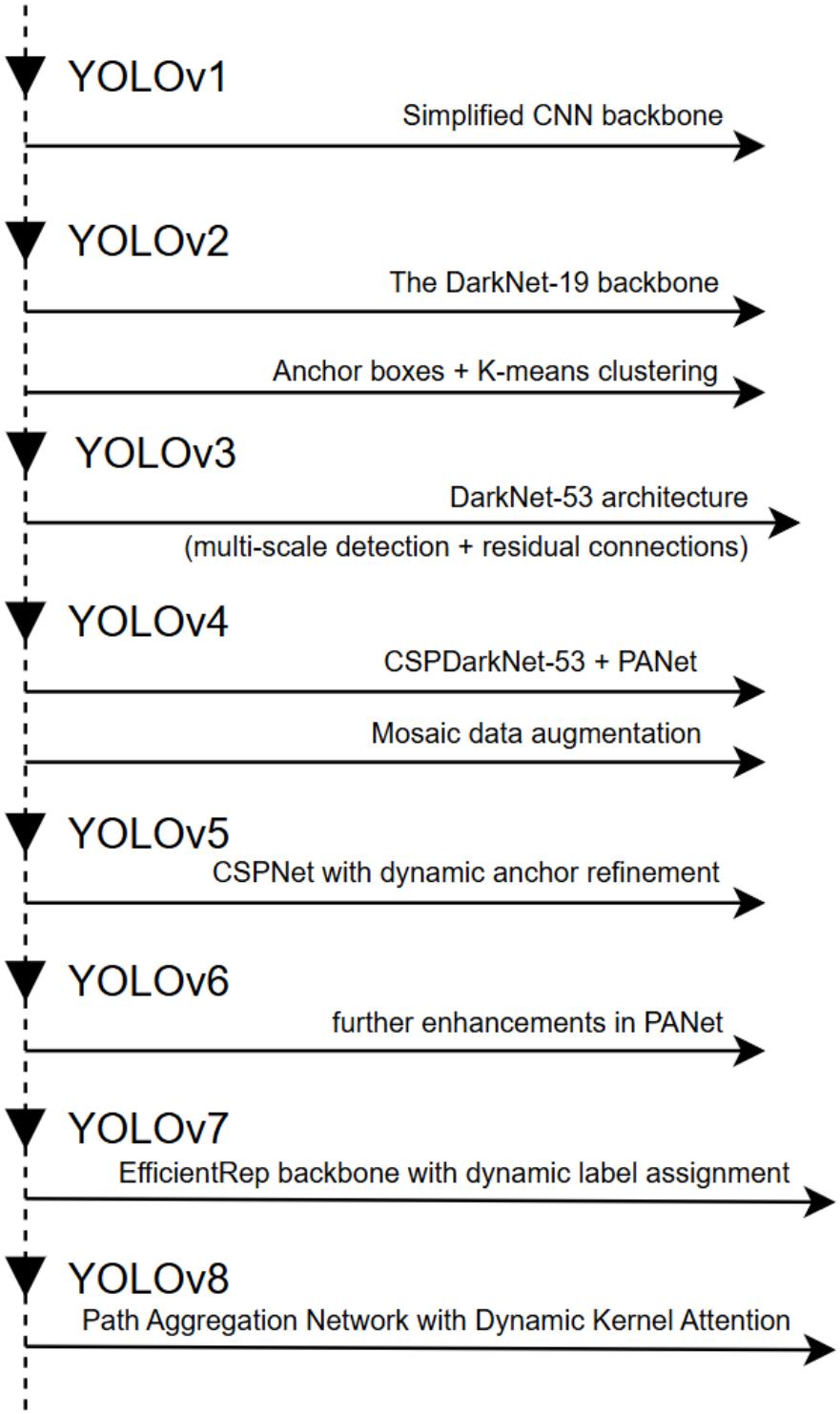


Рисунок 2.2 – Прогрес YOLO архітектури

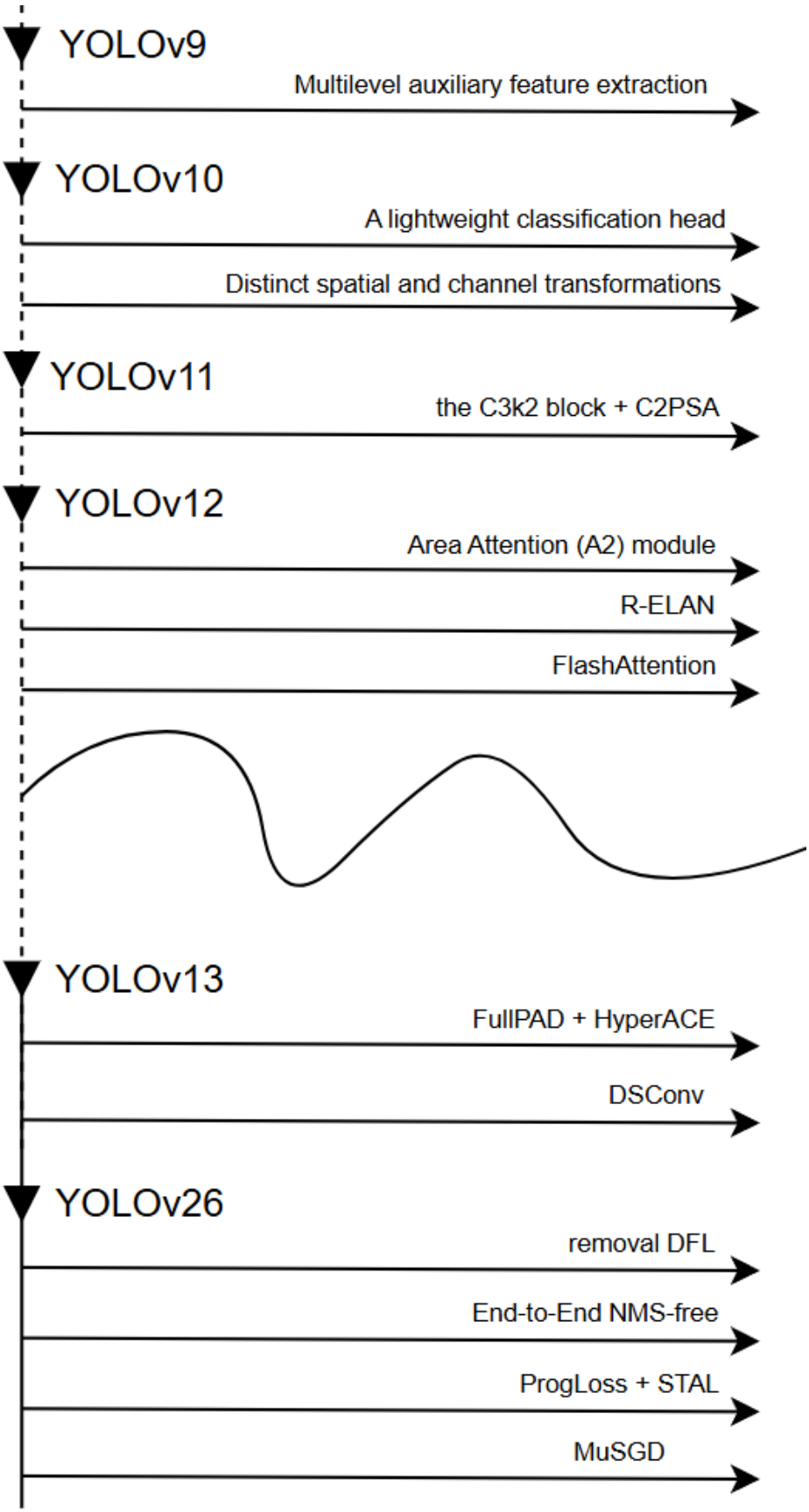


Рисунок 2.2 – Прогрес YOLO архітектури, аркуш 2

Реалізація архітектури моделі детекції об'єктів YOLO здійснюється на основі модульної конфігурації, описаної у YAML-файлі, який визначає структуру backbone, neck/head, блок передбачень та параметри масштабування моделі. Архітектура побудована відповідно до одноетапної парадигми детекції, у якій екстракція ознак, їх агрегація та передбачення виконуються в межах єдиної обчислювальної мережі. Далі задається система compound scaling – набір коефіцієнтів масштабування глибини та ширини мережі. Для варіанту x (Extra Large) використовується найбільші множники ширини та обмеження максимальної кількості каналів. Це означає, що кількість каналів у шарах збільшується відносно базової конфігурації, що підвищує репрезентаційну здатність моделі, але водночас збільшує обчислювальну складність. Подальша обробка на рівні передбачень зводила анотаційний формат до формату COCO, що включає координати об’єктів та індекси класів

2.1.1.1 YOLO 13

Архітектура спрямована на усунення ключових обмежень попередніх підходів, пов’язаних із локальним характером обробки ознак у згорткових мережах. У класичних CNN-орієнтованих детекторах кожна просторова позиція карти ознак переважно використовує локальний контекст, тому при частковому перекритті об'єкта інформативні ознаки можуть втрачатися або змішуватися з фоном чи сусідніми об'єктами, що ускладнює моделювання довгих просторових залежностей. Вона пропонує архітектурні зміни, зображені на рис 2.3, які дозволяють враховувати глобальний контекст сцени без суттєвого зростання обчислювальної складності, що критично для задач реального часу.

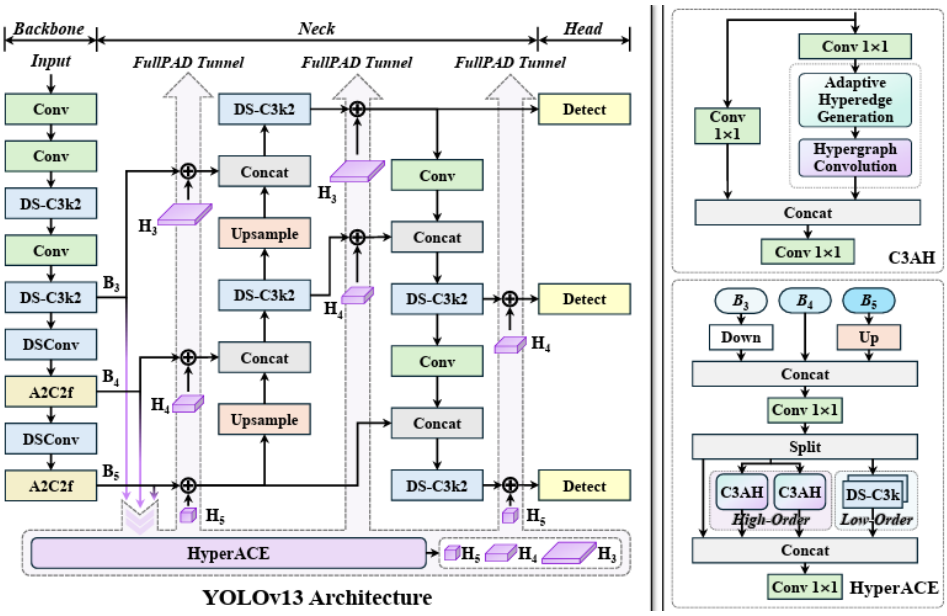


Рисунок 2.3 – Архітектура запропонованої YOLO 13

Основою архітектури є механізм HyperACE. Його головна ідея полягає у представленні просторових ознак не у вигляді звичайного графа попарних зв'язків, а у вигляді гіперграфа, де один зв'язок може одночасно об'єднувати декілька просторових позицій. Замість аналізу лише попарних взаємодій між окремими позиціями, модель формує узагальнені зв'язки між групами просторових елементів, що дозволяє враховувати інформацію з видимих частин об'єкта навіть тоді, коли інші його частини закриті. Також на відміну від класичного самоуваги із квадратичною складністю відносно кількості пікселів, HyperACE використовує обмежену кількість агрегуючих зв'язків, що зменшує обчислювальні витрати та дозволяє зберегти швидкість передбачень.

З точки зору реалізації на Graphic Process Unit (GPU) цей механізм обробки ознак може бути ефективно зіставлений з оптимізованими операціями attention, тому під час виконання моделі доцільно використовувати технологію Flash Attention (FA). На відміну від стандартної реалізації, яка потребує явного обчислення та збереження повної матриці взаємодій, FA виконує обчислення блоками безпосередньо в

швидкій пам'яті GPU, мінімізуючи звернення до глобальної пам'яті. Такий підхід зменшує обсяг проміжних тензорів і суттєво скорочує використання відеопам'яті. Корисність FA повинна бути помітною при роботі з великими розмірами вхідних зображень або збільшеним розміром партій (eng. batch size), оскільки дозволяє уникнути квадратичного зростання витрат пам'яті, характерного для класичних механізмів уваги, і забезпечує стабільну швидкодію моделі під час передбачення та валідації. На жаль, дослідження у цьому напрямку не дає чіткої відповіді щодо цього, технологія ще має свої недоліки, які не дозволяють працювати їй достатньо ефективно.

Другим ключовим компонентом є FullPAD. У традиційних його архітектурах інформаційний потік має переважно односпрямований характер – від backbone до neck і далі до detection head, а міжмасштабна взаємодія відбувається лише у піраміді особливостей (eng. feature pyramid). Це особливо важливо у випадках щільних сцен, де кілька об'єктів одного класу частково перекривають один одного. Інформація, отримана на вищих рівнях абстракції, повертається до нижчих рівнів, допомагаючи мережі відокремлювати межі об'єктів навіть за умов сильного перекриття. FullPAD змінює цю парадигму, забезпечуючи розповсюдження глобально покращених ознак по всьому обчислювальному графу мережі. Практично це реалізується через розширену систему пропусків з'єднань (eng. skip-connections), повторне введення узагальнених ознак у різні рівні мережі та двонапрямну передачу інформації між масштабами. У результаті кожен рівень мережі отримує доступ до глобального контексту, а не лише до локально сформованих ознак.

Додатковою особливістю YOLOv13 є переробка згорткових блоків із використанням depthwise-separable convolutions (DSConvs). Таке рішення дозволяє істотно зменшити кількість параметрів і обчислень без втрати просторової виразності представлень. У свою чергу це означає кращу масштабованість великих варіантів моделі розрахованих на велику кількість

параметрів, яка має значну глибину, але залишається придатною для практичного використання завдяки оптимізованій структурі обчислень.

Під час реального виконання моделі процес обробки виглядає як послідовність етапів: зображення проходить через згортковий backbone, де формується базове багатомасштабне представлення; далі блоки HyperACE підсилюють кореляції між віддаленими просторовими регіонами; після цього FullPAD розповсюджує отримані глобально узгоджені ознаки через усю мережу; на етапі neck виконується багатомасштабне злиття, а detection head формує остаточні передбачення класів і координат об'єктів.

Ключовим фактором підвищення ефективності боротьби з перекритими об'єктами є використання глобальних кореляцій ознак і багатомасштабного розповсюдження контекстної інформації, що дозволяє моделі реконструювати цілісне представлення об'єкта навіть при неповній видимості.

2.1.1.2 Конфігурація архітектури YOLO 13 як частина реалізації

Архітектура починається з параметра nc: 80, що означає кількість класів об’єктів відповідно до датасету COCO. Backbone відповідає за витягування ознак із зображення. На початкових етапах застосовуються послідовні згорткові шари зі кроком (eng. stride) 2, які виконують просторове зменшення роздільної здатності (eng. downsampling) і формують ієрархію ознак P1/2, P2/4, P3/8 тощо. Уже на ранніх рівнях використовуються блоки DSC3k2, які є оптимізованими згортковими структурами з розділенням каналів та зменшеним коефіцієнтом внутрішнього розширення 0.25. Такі блоки дозволяють зберігати інформаційну ємність представлення при значно меншій кількості параметрів.

Подальше зменшення просторової роздільності виконується через DSConv, що істотно скорочують кількість операцій множення-додавання порівняно зі стандартними згортками. На глибших рівнях backbone використовуються модулі A2C2f, які реалізують розширене агрегування ознак із внутрішніми залишковими зв'язками. Ці блоки покращують передачу градієнтів і дозволяють моделі ефективно поєднувати локальні та глобальні контекстні характеристики, що особливо важливо для складних сцен із перекритими об'єктами.

Head реалізує багатомасштабне злиття ознак. Центральним компонентом є модуль HyperACE, який приймає ознаки одразу з трьох рівнів backbone (P3, P4, P5). Він виконує адаптивне контекстне узгодження між масштабами, що дозволяє одночасно враховувати деталі дрібних об'єктів і глобальну сцену. Така взаємодія є критичною для задач детекції, де об'єкти можуть перекриватися або мати значну різницю в розмірах.

Після цього застосовується каскад операцій Upsample і DownsampleConv, які формують двонапрямлений потік ознак, подібний до Feature Pyramid Network (FPN). Ознаки багаторазово об'єднуються через операції Concat, що забезпечує повторне використання інформації з різних рівнів глибини мережі. Важливу роль відіграють блоки FullPAD_Tunnel, які виконують узгодження просторових і каналових характеристик перед об'єднанням тензорів. Фактично вони створюють стабільні канали передачі інформації між рівнями піраміди, мінімізуючи втрати контексту.

На кожному рівні після конкатенації знову застосовуються блоки DSC3k2, що дозволяє моделі переобчислити вже об'єднані ознаки та адаптувати їх до відповідного масштабу. Таким чином формується три фінальні карти ознак: P3 (для малих об'єктів), P4 (середніх) і P5 (великих).

Фінальний шар Detect отримує ці три карти ознак і виконує передбачення координат границь, ймовірностей класів та об'єктності. Багатомасштабний підхід дозволяє моделі ефективно працювати зі сценами

різної складності, включаючи частково перекриті об’єкти, оскільки різні рівні мережі спеціалізуються на різних просторових масштабах.

Загалом реалізація конфігурація архітектури виглядає як це показано на рис. 2.4.

```yaml
nc: 80 # number of classes
scales: # model compound scaling constants, i.e. 'model=yolov13n.yaml' will call yolov13.yaml with scale 'n'
  # [depth, width, max_channels]
  n: [0.50, 0.25, 1024]   # Nano
  s: [0.50, 0.50, 1024]   # Small
  l: [1.00, 1.00, 512]    # Large
  x: [1.00, 1.50, 512]    # Extra Large

backbone:
  # [from, repeats, module, args]
  - [-1, 1, Conv,  [64, 3, 2]] # 0-P1/2
  - [-1, 1, Conv,  [128, 3, 2, 1, 2]] # 1-P2/4
  - [-1, 2, DSC3k2,  [256, False, 0.25]]
  - [-1, 1, Conv,  [256, 3, 2, 1, 4]] # 3-P3/8
  - [-1, 2, DSC3k2,  [512, False, 0.25]]
  - [-1, 1, DSConv,  [512, 3, 2]] # 5-P4/16
  - [-1, 4, A2C2f, [512, True, 4]]
  - [-1, 1, DSConv,  [1024, 3, 2]] # 7-P5/32
  - [-1, 4, A2C2f, [1024, True, 1]] # 8

head:
  - [[4, 6, 8], 2, HyperACE, [512, 8, True, True, 0.5, 1, "both"]]
  - [-1, 1, nn.Upsample, [None, 2, "nearest"]]
  - [ 9, 1, DownsampleConv, []]
  - [[6, 9], 1, FullPAD_Tunnel, []]  #12
  - [[4, 10], 1, FullPAD_Tunnel, []]  #13
  - [[8, 11], 1, FullPAD_Tunnel, []] #14

  - [-1, 1, nn.Upsample, [None, 2, "nearest"]]
  - [[-1, 12], 1, Concat, [1]] # cat backbone P4
  - [-1, 2, DSC3k2, [512, True]] # 17
  - [[-1, 9], 1, FullPAD_Tunnel, []]  #18

  - [17, 1, nn.Upsample, [None, 2, "nearest"]]
  - [[-1, 13], 1, Concat, [1]] # cat backbone P3
  - [-1, 2, DSC3k2, [256, True]] # 21
  - [10, 1, Conv, [256, 1, 1]]
  - [[21, 22], 1, FullPAD_Tunnel, []]  #23

  - [-1, 1, Conv, [256, 3, 2]]
  - [[-1, 18], 1, Concat, [1]] # cat head P4
  - [-1, 2, DSC3k2, [512, True]] # 26
  - [[-1, 9], 1, FullPAD_Tunnel, []]

  - [26, 1, Conv, [512, 3, 2]]
  - [[-1, 14], 1, Concat, [1]] # cat head P5
  - [-1, 2, DSC3k2, [1024,True]] # 30 (P5/32-large)
  - [[-1, 11], 1, FullPAD_Tunnel, []]

  - [[23, 27, 31], 1, Detect, [nc]] # Detect(P3, P4, P5)
```

Рисунок 2.4 – Конфігурація YOLO 13 для розпізнавання об’єктів

Практична реалізація та оцінювання моделі визначаються викликом model.val(data='coco.yaml', batch=16, imgsz=704).

Параметр data='coco.yaml' задає конфігурацію датасету COCO: шляхи до зображень, анотацій і список із 80 класів. Метод val() переводить модель

у режим оцінювання, вимикає оновлення ваг і виконує пронозування на валідаційній або тестовій вибірці з автоматичним обчисленням метрик.

Параметр batch=16 означає, що одночасно обробляється 16 зображень. Це дозволяє ефективніше використовувати GPU завдяки паралельним обчисленням, але збільшує споживання відеопам’яті. Розмір imgsz=704 визначає масштабування вхідних зображень до квадратної роздільності 704×704 пікселів перед подачею у мережу. Збільшений розмір входу покращує детекцію дрібних і перекритих об’єктів, оскільки зберігається більше просторових деталей, хоча це також підвищує обчислювальне навантаження.

З точки зору GPU-реалізації така конфігурація добре узгоджується з використанням оптимізованих attention-обчислень, зокрема Flash Attention. Оскільки частина модулів head виконує інтенсивне міжмасштабне узгодження ознак, блокове виконання attention-операцій у швидкій пам’яті GPU дозволяє зменшити обсяг проміжних тензорів і стабілізувати використання пам’яті при великих розмірах зображень та розмір партій, що безпосередньо впливає на швидкість валідації.

Безпосередньо реалізацію можна переглянути на рисунку А.9.

2.1.1.3 YOLO 12

На відміну від попередньої версії основний акцент зроблено на інтеграції механізмів attention без втрати швидкодії, характерної для класичних моделей YOLO. Їх інтеграція дозволяє моделі враховувати залежності між віддаленими ділянками зображення, що особливо важливо для складних сцен і малих об’єктів. Це дає змогу моделі відновлювати семантичний контекст навіть тоді, коли частина об’єкта закрита іншим об’єктом або фоном.

Ключовим елементом реалізації є механізм Area Attention (AA). Його основна ідея полягає в тому, що attention обчислюється не між усіма окремими пікселями або токенами, як у класичних transformer-архітектурах, а між більшими просторовими областями мапи особливостей. Враховуючи таку поведінку, модель може використовувати контекст із неперекритих частин об'єкта для реконструкції його повного представлення. Таким чином, навіть якщо частина об'єкта закрита, інформація з інших областей сцени допомагає правильно ідентифікувати його. Далі attention виконується між цими регіонами. Такий підхід суттєво зменшує обчислювальну складність і обсяг пам'яті, необхідний для роботи моделі, що робить можливим використання attention у real-time детекторі.

Другим важливим компонентом архітектури є блоки Residual Efficient Layer Aggregation Network (R-ELAN), які відповідають за ефективну агрегацію ознак. Вони також сприяють боротьбі з перекриттям, оскільки вони забезпечують ефективне повторне використання ознак і стабільніше поширення інформації між шарами. Завдяки цьому модель краще зберігає слабкі або частково втрачені сигнали, характерні для перекритих об'єктів. Замість простого послідовного проходження через згорткові шари використовується багатогілкова структура з об'єднанням результатів і залишковими з'єднаннями. Це дозволяє повторно використовувати вже обчислені ознаки, покращує поширення градієнтів під час навчання та підвищує репрезентативну здатність мережі без значного збільшення кількості параметрів.

Модуль neck у YOLO 12 виконує багатомасштабне об'єднання ознак, аналогічно підходам FPN або Path Aggregation Network (PAN), однак із додатковим використанням attention-керованого злиття інформації, щоби розділяти близько розташовані об'єкти навіть за значного перекриття, оскільки кожен із них отримує власне представлення у просторі ознак. Це допомагає ефективніше передавати контекст між різними рівнями

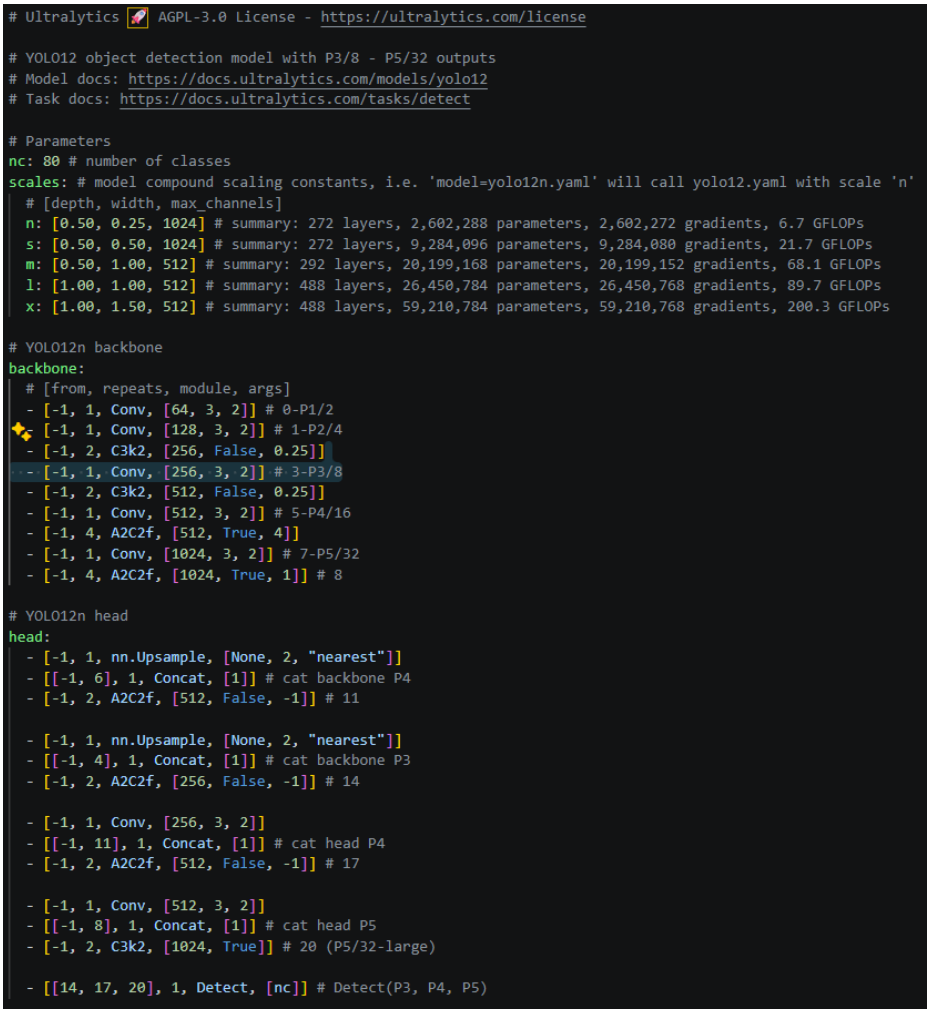
просторової роздільності та покращує детекцію об’єктів різного масштабу. Особливо помітним є покращення роботи з малими об’єктами, які традиційно є складними для одноетапних детекторів. Поєднання високорівневих семантичних ознак із низькорівневими просторовими деталями дозволяє точніше локалізувати об’єкти, які частково накладаються один на одного. Завдяки цьому навіть частково видимі об’єкти можуть бути розпізнані через їх високорівневі семантичні характеристики, а не лише через повний контур або форму.

Detection head зберігає загальну логіку YOLO-підходу: модель виконує одночасне передбачення координат границь і класів об’єктів у межах одного проходу мережі. Оскільки вона отримує більш контекстуально насичені мапи особливостей, вона здатна точніше розділяти близько розташовані границь і зменшувати кількість помилок, пов’язаних із злиттям кількох об’єктів в одну детекцію. Архітектура залишається anchor-free і не зазнає радикальних змін, оскільки основне покращення точності досягається завдяки більш якісним ознакам, сформованим попередніми рівнями мережі.

Особливістю YOLO 12 є орієнтація на апаратну ефективність. Архітектура проєктувалася з урахуванням особливостей сучасних GPU, зокрема мінімізації доступів до пам’яті та можливості використання оптимізованих attention-алгоритмів, таких як FA. Обмеження області attention та оптимізована структура блоків дозволяють уникнути квадратичного росту складності, характерного для класичних transformer-моделей. Загалом її можна розглядати як гібридну архітектуру, у якій attention стає центральним механізмом формування представлень, але реалізований у спосіб, сумісний із вимогами реального часу.

2.1.1.4 Конфігурація архітектури YOLO 12 як частина реалізації

В обох конфігураціях визначено однакову кількість класів, що дорівнює 80. Усі відмінності можна прослідкувати на рис. 2.5, так же як й компоненти реалізації архітектури 12 версії.



```yaml
# Ultralytics 🚀 AGPL-3.0 License - https://ultralytics.com/license

# YOLO12 object detection model with P3/8 - P5/32 outputs
# Model docs: https://docs.ultralytics.com/models/yolo12
# Task docs: https://docs.ultralytics.com/tasks/detect

# Parameters
nc: 80 # number of classes
scales: # model compound scaling constants, i.e. 'model=yolo12n.yaml' will call yolo12.yaml with scale 'n'
  # [depth, width, max_channels]
  n: [0.50, 0.25, 1024] # summary: 272 layers, 2,602,288 parameters, 2,602,272 gradients, 6.7 GFLOPs
  s: [0.50, 0.50, 1024] # summary: 272 layers, 9,284,096 parameters, 9,284,080 gradients, 21.7 GFLOPs
  m: [0.50, 1.00, 512] # summary: 292 layers, 20,199,168 parameters, 20,199,152 gradients, 68.1 GFLOPs
  l: [1.00, 1.00, 512] # summary: 488 layers, 26,450,784 parameters, 26,450,768 gradients, 89.7 GFLOPs
  x: [1.00, 1.50, 512] # summary: 488 layers, 59,210,784 parameters, 59,210,768 gradients, 200.3 GFLOPs

# YOLO12n backbone
backbone:
  # [from, repeats, module, args]
  - [-1, 1, Conv, [64, 3, 2]] # 0-P1/2
  - [-1, 1, Conv, [128, 3, 2]] # 1-P2/4
  - [-1, 2, C3k2, [256, False, 0.25]]
  - [-1, 1, Conv, [256, 3, 2]] # 3-P3/8
  - [-1, 2, C3k2, [512, False, 0.25]]
  - [-1, 1, Conv, [512, 3, 2]] # 5-P4/16
  - [-1, 4, A2C2f, [512, True, 4]]
  - [-1, 1, Conv, [1024, 3, 2]] # 7-P5/32
  - [-1, 4, A2C2f, [1024, True, 1]] # 8

# YOLO12n head
head:
  - [-1, 1, nn.Upsample, [None, 2, "nearest"]]
  - [[-1, 6], 1, Concat, [1]] # cat backbone P4
  - [-1, 2, A2C2f, [512, False, -1]] # 11

  - [-1, 1, nn.Upsample, [None, 2, "nearest"]]
  - [[-1, 4], 1, Concat, [1]] # cat backbone P3
  - [-1, 2, A2C2f, [256, False, -1]] # 14

  - [-1, 1, Conv, [256, 3, 2]]
  - [[-1, 11], 1, Concat, [1]] # cat head P4
  - [-1, 2, A2C2f, [512, False, -1]] # 17

  - [-1, 1, Conv, [512, 3, 2]]
  - [[-1, 8], 1, Concat, [1]] # cat head P5
  - [-1, 2, C3k2, [1024, True]] # 20 (P5/32-large)

  - [[14, 17, 20], 1, Detect, [nc]] # Detect(P3, P4, P5)
```

Рисунок 2.5 – Конфігурація YOLO 12 для розпізнавання об’єктів

Основні відмінності починаються вже на рівні backbone. Початкові етапи виділення ознак реалізовані через стандартні згорткові шари, які виконують поступове зменшення просторової роздільності та збільшення кількості каналів. Подальша обробка виконується блоками C3k2, що реалізують CSP-подібну агрегацію із частковими залишковими з’єднаннями.

Попри ці зміни, обидві архітектури зберігають ключові рівні ознак P3, P4 та P5 із коефіцієнтами зменшення роздільності 8, 16 та 32 відповідно. Це забезпечує сумісність детекційної частини та збереження багатомасштабної логіки роботи.

Спільною рисою моделей є використання блоків A2C2f у глибоких рівнях backbone. У YOLO 12 ці блоки виступають основним механізмом покращеної агрегації контексту та забезпечують поєднання локальних і глобальних ознак.

У head блоці побудований FPN із двонаправленою агрегацією. Ознаки глибоких рівнів послідовно збільшуються за допомогою upsampling, конкатенуються з відповідними рівнями backbone і проходять через блоки A2C2f для адаптивного злиття інформації. Після цього виконується зворотний шлях із повторним downsampling, що формує остаточні карти ознак для трьох масштабів детекції.

На відміну від більш складних схем повторного використання ознак у новіших архітектурах, у YOLO 12 застосовуються прості skip-connections, які передають інформацію безпосередньо між окремими шарами. Такі з’єднання забезпечують базове збереження просторових і семантичних ознак під час проходження сигналу мережею, але не формують щільної системи багаторівневого обміну даними. У результаті обчислювальний граф має відносно просту структуру, де передача ознак відбувається переважно послідовно, з обмеженим повторним перерозподілом інформації між різними масштабами.

Зменшення просторової роздільності виконується переважно в backbone мережі, де поступово зменшується розмір ознак під час їх глибшої обробки, а також частково під час формування багатомасштабної піраміди ознак. У head мережі окремі спеціалізовані модулі для додаткового контрольованого downsampling не використовуються, тому порядок обробки інформації залишається більш традиційним: спочатку формується

ієрархія ознак різних масштабів, після чого вони без додаткового зменшення роздільності застосовуються для задачі детекції.

У YOLO 12 head побудований переважно на стандартних згорткових блоках без широкого використання спеціалізованих полегшених модулів на кшталт DSC3k2. Баланс між точністю та швидкодією у YOLO 12 забезпечується спрощеною архітектурою, де обчислювальні витрати залишаються контрольованими завдяки помірній складності обчислювального графа.

Detect модуль у двох архітектурах залишається концептуально однаковим. Обидві моделі використовують anchor-free передбачення та формують результати на трьох масштабах. Різниця полягає не у самій процедурі детекції, а у якості сформованих ознак, які подаються на цей модуль. При цьому валідаційна частина залишається однаковою в обох випадка, окрім вірогідної різниці у передбаченні та роботі у компонентах цієї версії.

Безпосередньо реалізацію можна переглянути на рисунку А.9.

2.1.2 Принцип роботи RT-DETRv2

Архітектура RT-DETRv2 належить до класу transformer-орієнтованих моделей детекції об'єктів, які виконують передбачення у форматі end-to-end без використання додаткових етапів обробки результатів. Основною метою її розробки є поєднання високої точності Detection Transformer (DETR)-подібних моделей із швидкодією, достатньою для застосування в режимі реального часу. На відміну від ранніх архітектур DETR, дана модель спроєктована з урахуванням практичних обмежень обчислювальних ресурсів та особливостей виконання на сучасних GPU.

Архітектурно модель складається з послідовності функціональних модулів, які утворюють єдиний обчислювальний граф: модуль виділення

ознак у backbone блоці, гібридний encoder, transformer-decoder із запитами об'єктів та модуль передбачення параметрів об'єктів. Вхідне зображення проходить через ці компоненти лише один раз, що забезпечує детекцію без багатоетапної генерації кандидатних областей.

Першим етапом обробки є backbone-мережа згорткового типу, відповідальна за екстракцію багаторівневих ознак із зображення. У реалізації RT-DETRv2 backbone виконує не лише роль початкового перетворення даних, а й оптимізується з точки зору обчислювальної ефективності. Особливістю є формування декількох масштабів ознак без складних пірамідальних структур, характерних для традиційних детекторів. Це дозволяє зменшити кількість операцій переміщення даних у пам'яті, що є критичним фактором для досягнення низької затримки під час отримання передбачень. Вихідні мапи ознак передаються безпосередньо до encoder-частини моделі.

Ключовим елементом архітектури є гібридний encoder, який вирішує головну проблему класичних DETR – високу обчислювальну складність глобального self-attention. У стандартних transformer-encoder кожен просторовий елемент взаємодіє з усіма іншими, що призводить до квадратичного зростання складності відносно кількості токенів. У RT-DETRv2 ця проблема розв'язується шляхом розділення обробки на локальні та міжмасштабні взаємодії. Поєднання локальної внутрішньомасштабної взаємодії та міжмасштабного обміну інформацією дозволяє моделі одночасно враховувати дрібні текстурні деталі та глобальну структуру сцени. Для перекритих об’єктів це критично, оскільки частина об’єкта може бути видимою лише на певному масштабі ознак.

У межах одного масштабу застосовується обмежена взаємодія, яка моделює локальні залежності між сусідніми ознаками. Глобальний attention використовується лише для агрегування інформації між різними рівнями ознак. Такий підхід дозволяє значно зменшити кількість attention-операцій

без втрати здатності моделі інтегрувати глобальний контекст сцени. З реалізаційної точки зору це означає меншу кількість reshaping-операцій тензорів і кращу відповідність виконанню GPU-ядер, що напряму впливає на швидкість роботи.

Після encoder-етапу ознаки передаються до transformer-decoder, який працює за принципом object queries. На відміну від класичних детекторів, де спочатку генеруються області-кандидати (eng. anchor boxes), у RT-DETRv2 використовується фіксований набір навчуваних векторів-запитів. Кожен такий запит навчається представляти один потенційний об'єкт сцени. Decoder ітеративно уточнює ці представлення, використовуючи механізм cross-attention між запитами та закодованими ознаками зображення. Завдяки механізму cross-attention, decoder одночасно аналізує повну карту ознак, що дозволяє моделі розрізняти навіть сильно перекриті об’єкти.

Важливою реалізаційною особливістю є фіксована кількість запитів, що забезпечує стабільну обчислювальну складність незалежно від кількості об'єктів на зображенні. Це значно спрощує оптимізацію отримання передбачень та робить час виконання передбачуваним, що критично для real-time систем.

Detection head модуль архітектури перетворює вихід decoder у параметри детекції. Він складається з двох незалежних гілок: класифікації та регресії координат границь. Результати не проходять етап NMS, що покращує детекцію перекритих об’єктів. Замість цього під час навчання використовується алгоритм Hungarian matching, який встановлює однозначну відповідність між передбаченнями моделі та реальними об'єктами. Завдяки цьому модель навчається генерувати неперекривні та узгоджені передбачення без необхідності постобробки. Це змушує кожен object query спеціалізуватися на окремому об’єкті, навіть якщо вони сильно перекриваються. У результаті модель навчається уникати дублювання передбачень і чітко розділяти близько розташовані об’єкти.

Суттєвою особливістю RT-DETRv2 є використання підходу, який автори описують як bag-of-freebies. Йдеться про набір удосконалень процесу навчання, що підвищують точність моделі без зміни обчислювального графа під час передбачень. До них належать модифікації функцій втрат, покращені стратегії аугментації даних, стабілізація нормалізації ознак і вдосконалені методи призначення цільових об'єктів. Завдяки цьому формуються ознакові представлення, здатні відокремлювати об’єкти навіть за мінімальної видимості.

2.1.2.1 Конфігурація архітектури RT-DETRv2 як частина реалізації

Перед подачею до моделі вхідні зображення проходять етап попередньої обробки. Зображення автоматично змінюють розмір до фіксованої, що забезпечує сталу обчислювальну складність незалежно від початкового розміру даних. Масштабування виконується з використанням інтерполяції, яка дозволяє мінімізувати втрати деталей. Значення пікселів нормалізуються, а додаткова нормалізація за середнім значенням та стандартним відхиленням не застосовується, оскільки стабілізація розподілу ознак забезпечується внутрішніми механізмами мережі. Це все наведено за конфігурацією на наступному рис. 2.6.

Першим етапом обробки є backbone-мережа, реалізована як модифікована ResNet-подібна архітектура зі 101 шаром. Вона складається з чотирьох послідовних етапів згорткових блоків різної глибини, які поступово зменшують просторову роздільну здатність ознак. Для подальшої обробки використовуються три останні рівні ознак, що відповідають масштабам із коефіцієнтами зменшення 8, 16 та 32 відносно початкового зображення.

```
{
  "activation_dropout": 0.0,
  "activation_function": "silu",
  "anchor_image_size": null,
  "architectures": [
    "RtDetrV2ForObjectDetection"
  ],
  "attention_dropout": 0.0,
  "auxiliary_loss": true,
  "backbone": null,
  "backbone_config": {
    "depths": [
      3,
      4,
      23,
      3
    ],
    "model_type": "rt_detr_resnet",
    "out_features": [
      "stage2",
      "stage3",
      "stage4"
    ],
    "out_indices": [
      2,
      3,
      4
    ]
  },
  "backbone_kwargs": null,
  "batch_norm_eps": 1e-05,
  "box_noise_scale": 1.0,
  "d_model": 256,
  "decoder_activation_function": "relu",
  "decoder_attention_heads": 8,
  "decoder_ffn_dim": 1024,
  "decoder_in_channels": [
    384,
    384,
    384
  ],
  "decoder_layers": 6,
  "decoder_method": "default",
  "decoder_n_levels": 3,
  "decoder_n_points": 4,
  "decoder_offset_scale": 0.5,
  "disable_custom_kernels": true,
  "dropout": 0.0,
  "encode_proj_layers": [
    2
  ],
"encoder_activation_function": "gelu",
  "encoder_attention_heads": 8,
  "encoder_ffn_dim": 2048,
  "encoder_hidden_dim": 384,
  "encoder_in_channels": [
    512,
    1024,
    2048
  ],
  "encoder_layers": 1,
  "eos_coefficient": 0.0001,
  "eval_size": null,
  "feat_strides": [
    8,
    16,
    32
  ],
  "focal_loss_alpha": 0.75,
  "focal_loss_gamma": 2.0,
  "freeze_backbone_batch_norms": true,
  "hidden_expansion": 1.0,
  "id2label": {
    ... (id-based transformation)
  },
  "initializer_bias_prior_prob": null,
  "initializer_range": 0.01,
  "is_encoder_decoder": true,
  "label2id": {
    ... (label-based transformation)
  },
  "label_noise_ratio": 0.5,
  "layer_norm_eps": 1e-05,
  "learn_initial_query": false,
  "matcher_alpha": 0.25,
  "matcher_bbox_cost": 5.0,
  "matcher_class_cost": 2.0,
  "matcher_gamma": 2.0,
  "matcher_giou_cost": 2.0,
  "model_type": "rt_detr_v2",
  "normalize_before": false,
  "num_denoising": 100,
  "num_feature_levels": 3,
  "num_queries": 300,
  "positional_encoding_temperature": 10000,
  "torch_dtype": "float32",
  "transformers_version": "4.49.0.dev0",
  "use_focal_loss": true,
  "use_pretrained_backbone": false,
  "use_timm_backbone": false,
  "weight_loss_bbox": 5.0,
  "weight_loss_giou": 2.0,
  "weight_loss_vfl": 1.0,
  "with_box_refine": true

}
```


Рисунок 2.6 – Конфігурація RT-DETRv2 для розпізнавання об’єктів

Вихідні карти ознак мають різну кількість каналів, тому перед передачею у transformer вони приводяться до спільного представлення. Batch normalization у backbone фіксується під час навчання, що дозволяє уникнути нестабільності статистик при обмеженому розмірі пакета даних.

Ознаки різних масштабів проходять через проекційні шари, які перетворюють їх у єдиний прихований простір encoder. Це забезпечує можливість одночасної обробки багатомасштабної інформації без

збільшення складності attention-операцій. Після цього дані передаються до transformer-encoder.

Encoder у даній конфігурації має лише один шар, що є оптимізаційним рішенням для забезпечення роботи в режимі реального часу. Незважаючи на малу глибину, encoder використовує багатоголовий self-attention та повнозв'язну мережу великої розмірності, завдяки чому відбувається ефективне змішування інформації між усіма просторовими позиціями та масштабами ознак. Як функція активації використовується GELU, яка забезпечує плавну нелінійність і стабільні градієнти під час навчання. Dropout у encoder не застосовується, що зменшує випадковість обчислень і робить поведінку моделі більш детермінованою.

Основна частина детекції реалізована у transformer-decoder. Decoder складається з шести послідовних шарів, які працюють із фіксованим набором із 300 object queries. Кожен запит є векторним представленням потенційного об'єкта сцени. На відміну від традиційних детекторів, де кількість кандидатів залежить від кількості позицій на зображенні, тут обчислювальна складність залишається сталою. Decoder використовує механізм deformable attention, який аналізує лише обмежену кількість інформативних точок на кожному масштабі ознак. Це значно зменшує обчислювальні витрати порівняно з повним attention та дозволяє ефективно працювати з високою роздільною здатністю.

У процесі проходження через decoder кожен шар поступово уточнює координати об'єкта. Такий механізм ітеративного уточнення дозволяє спочатку оцінити приблизне положення об'єкта, а потім покращувати його межі на наступних рівнях обробки. Для нелінійності у decoder використовується функція ReLU, яка забезпечує швидші обчислення та менше навантаження на пам'ять.

Під час навчання застосовується спеціальний механізм денойзингу. До навчальних даних додаються зашумлені копії об'єктів із

модифікованими координатами та мітками. Це змушує модель швидше навчатися правильному співставленню запитів із реальними об’єктами та підвищує стійкість до складних сцен, зокрема перекриттів або часткової видимості.

Фінальний модуль моделі виконує класифікацію об’єктів і регресію координат границь. Для навчання використовується комбінована функція втрат, яка включає втрати координат, узагальнену IoU-втрату (GIoU) та Focal Loss (FL) для класифікації. Обчислення за ними наведено за формулами 2.1-2.2 відповідно. Використання FL дозволяє зменшити вплив великої кількості простих негативних прикладів і зосередити навчання на складних об’єктах.

Безпосередньо реалізацію можна переглянути на рисунку А.6.

2.1.3 Принцип роботи DEIMv2

Архітектура побудована для задач детекції об’єктів у реальному часі, яка поєднує підхід DETR із використанням попередньо навчених моделей, зокрема DINOv3. Основна ідея полягає в тому, щоб створити єдину архітектурну схему, яка може масштабуватись під різні обчислювальні ресурси без зміни базових принципів побудови.

Ключовим компонентом є backbone, який у більших моделях базується на DINOv3. Однак DINOv3 генерує лише одномасштабні ознаки, що є обмеженням для задач детекції, де необхідна багатомасштабність. Для вирішення цієї проблеми в архітектуру введено спеціальний модуль Spatial Tuning Adapter (STA). Він перетворює одномасштабні ознаки у багатомасштабні та одночасно додає локальні деталі до глобальних представлень. У випадку перекриття це особливо важливо, оскільки різні частини об’єкта можуть бути видимі на різних масштабах. STA дозволяє поєднати ці фрагменти в узгоджене представлення, що підвищує

ймовірність коректної детекції навіть при частковій видимості. Для моделей, орієнтованих на обмежені ресурси (наприклад, edge або mobile), використовується альтернативний backbone – HGNetv2.

Декодер у DEIMv2 є спрощеною версією DETR-подібної структури. У ньому зменшено кількість шарів уваги та оптимізовано обробку запитів, що безпосередньо впливає на зменшення затримки під час отримання передбачень. Це дозволяє досягти балансу між точністю та швидкодією, що є критично важливим для застосувань у реальному часі.

Важливу роль у навчанні відіграє механізм Dense One-to-One (Dense O2O) matching. Він відповідає за зіставлення передбачень моделі з еталонними об’єктами та забезпечує ефективне end-to-end навчання без необхідності складної постобробки. У порівнянні з класичними DETR-підходами, цей механізм забезпечує більш щільний сигнал навчання і покращує збіжність моделі. Додатково використовується підхід Multi-Assignment Learning (MAL), який дозволяє одному об’єкту відповідати кільком передбаченням на етапі навчання.

Архітектура також реалізує єдину стратегію масштабування. Як вже згадувалося, великі моделі використовують DINOv3 разом зі STA, тоді як компактні версії переходять на HGNetv2 із застосуванням pruning. При цьому загальна логіка побудови моделі залишається незмінною, що значно спрощує підтримку та розгортання у виробничих умовах.

Як у всіх розглянутих архітектурах уся система побудована за принципом модульності. Backbone відповідає за витяг ознак, адаптер – за їх перетворення у зручний для детекції формат, а декодер – за фінальне передбачення об’єктів. Такий розподіл дозволяє незалежно оптимізувати кожну частину. Крім того, архітектура орієнтована на ефективне використання параметрів і враховує особливості апаратного забезпечення, що робить її придатною як для потужних GPU, так і для обмежених пристроїв.

У підсумку, DEIMv2 можна розглядати як гібрид, який полягає не у створенні принципово нового детектора, а в ефективній адаптації вже наявних потужних представлень до вимог задачі детекції в реальному часі.

2.1.3.1 Конфігурація архітектури DEIMv2 як частина реалізації

Реалізація архітектури побудована як послідовна модульна система, що складається з backbone з адаптерами, гібридного енкодера та трансформерного декодера. Усі компоненти узгоджені між собою через єдину розмірність ознак – 256 каналів, що дозволяє уникнути додаткових перетворень і спрощує передачу даних між модулями. Побудову можна побачити на рис. 2.7.

На першому етапі використовується backbone на основі моделі DINOv3. Вона генерує ознаки з розмірністю 256, які одразу придатні для подальшої обробки. Однак ці ознаки є одномасштабними, тому для їх адаптації до задачі детекції застосовуються STAs. У реалізації вибрано три точки підключення до трансформера backbone – на шарах із індексами 5, 8 та 11. Це дозволяє витягнути ознаки різного рівня абстракції. Далі кожен із цих виходів проходить через легкі згорткові перетворення та приводиться до єдиної розмірності 256 каналів. У результаті формується трирівневе багатомасштабне представлення з кроками 8, 16 та 32, яке замінює класичну піраміду ознак.

Отримані багатомасштабні ознаки передаються до гібридного енкодера. Його роль полягає в інтеграції та додатковому уточненні ознак перед подачею у декодер.

```
{
  "DEIMTransformer": {
    "activation": "silu",
    "box_noise_scale": 1.0,
    "cross_attn_method": "default",
    "dim_feedforward": 2048,
    "eval_idx": -1,
    "eval_spatial_size": [
      640,
      640
    ],
    "feat_channels": [
      256,
      256,
      256
    ],
    "feat_strides": [
      8,
      16,
      32
    ],
    "hidden_dim": 256,
    "label_noise_ratio": 0.5,
    "layer_scale": 1,
    "mlp_act": "silu",
    "num_denoising": 100,
    "num_layers": 6,
    "num_levels": 3,
    "num_points": [
      3,
      6,
      3
    ],
    "num_queries": 300,
    "query_select_method": "default",
    "reg_max": 32,
    "reg_scale": 4
  },
  "DINOv3STAs": {
    "conv_inplane": 64,
    "embed_dim": 256,
    "hidden_dim": 256,
    "interaction_indexes": [
      5,
      8,
      11
    ],
    "name": "dinov3_vits16plus",
    "num_heads": null
  },
  "HybridEncoder": {
    "act": "silu",
    "csp_type": "csp2",
    "depth_mult": 1.37,
    "dim_feedforward": 1024,
    "dropout": 0.0,
    "enc_act": "gelu",
    "expansion": 1.25,
```


Рисунок 2.7 – Конфігурація архітектури DEIMv2

Рисунок 2.7 – Конфігурація архітектури DEIMv2, аркуш 2

Енкодер працює з трьома рівнями, кожен із яких має 256 каналів, і об’єднує їх за допомогою операції сумування. Архітектурно він містить лише один трансформерний шар із вісьмома головами уваги та feedforward-блоком розміром 1024, що робить його легким з точки зору обчислень. Для активацій використовується комбінація GELU у трансформерній частині та SiLU у згорткових блоках. Додатково застосовується CSP-подібна структура з помірним розширенням каналів і збільшенням глибини, що підвищує виразну здатність без значного зростання складності. При цьому трансформерна обробка застосовується лише до одного, найглибшого рівня ознак (з кроком 32), тоді як інші рівні обробляються простішими операціями. Такий підхід дозволяє зменшити затримку без суттєвої втрати якості.

Далі ознаки подаються до трансформерного декодера DEIMTransformer, який відповідає за безпосередню детекцію об’єктів. У моделі використовується 300 запитів, що визначає максимальну кількість можливих передбачень. Для покращення навчання застосовується механізм

денойзингу зі 100 додатковими запитами, який стабілізує процес оптимізації. Декодер складається з шести шарів, кожен з яких включає self-attention, cross-attention та feedforward-блоки. Усі обчислення виконуються в просторі розмірності 256, а feedforward-шари мають розмір 2048. Як функцію активації використано SiLU.

Для роботи з багатомасштабними ознаками у декодері застосовується deformable attention. Для кожного рівня визначено кількість точок семплювання (3, 6 та 3 відповідно), що дозволяє ефективно агрегувати інформацію з різних масштабів без значного збільшення обчислювальних витрат. Додатково під час навчання вводиться шум у мітки та координати об’єктів (з коефіцієнтами 0.5 та 1.0 відповідно), що покращує узагальнюючу здатність моделі.

Для регресії координат використовується дискретизований підхід: координати передбачаються як розподіли з максимальною кількістю бінів та масштабуванням. Це дозволяє досягти більшої точності порівняно з прямою регресією.

На фінальному етапі постпроцесор відбирає 300 найкращих передбачень. Завдяки DETR-подібному підходу модель не потребує класичного NMS, оскільки механізм навчання вже забезпечує відсутність дублювань.

Безпосередньо реалізацію можна переглянути на рисунку А.3.

2.1.4 Принцип роботи SSD

Уся система працює як єдиний модуль: на вхід подається зображення, а на виході одразу отримуються координати границь і ймовірності класів. Основою SSD є згорткова нейронна мережа, яка використовується як екстрактор ознак, тобто backbone. У класичному варіанті використовується модифікована VGG16, та поверх цієї базової мережі додаються додаткові

згорткові шари, які поступово зменшують просторову роздільну здатність ознак. У реалізації це виглядає як послідовність мап ознак різного масштабу.

Важливою особливістю є використання кількох мап ознак для детекції об’єктів різних розмірів. Ранні шари з високою роздільною здатністю використовуються для виявлення малих об’єктів, тоді як глибші шари – для великих. Це реалізується шляхом застосування однакових за структурою prediction heads до кожного з цих шарів.

Ключовим елементом реалізації є вже згадані anchor boxes. Для кожної позиції на кожній карті ознак генерується кілька прямокутників із різними масштабами та співвідношеннями сторін. Таким чином, ще до навчання визначається велика множина потенційних границь. Мережа не генерує їх динамічно, а лише коригує ці заздалегідь задані варіанти. Для кожного такого з них мережа передбачає два типи значень: по-перше, зсуви координат, які уточнюють положення рамки, і по-друге, ймовірності належності до кожного класу. Це реалізується через окремі згорткові фільтри для локалізації та класифікації, які застосовуються до кожної карти ознак.

Під час навчання важливим етапом є зіставлення (eng. matching) між реальними та anchor об’єктами. Для цього використовується метрика IoU. Anchor boxes, які достатньо добре перекривають реальні об’єкти, вважаються позитивними прикладами, інші – негативними. Оскільки негативних прикладів значно більше, використовується стратегія hard negative mining, яка залишає лише найскладніші з них для обчислення функції втрат.

Функція втрат у SSD є комбінованою: вона складається з двох частин – втрат локалізації, зазвичай Smooth L1, та втрат класифікації softmax. Обидві частини обчислюються одночасно та нормалізуються за кількістю позитивних прикладів.

На етапі передбачення модель працює дуже просто: після одного проходу мережі отримані передбачення декодуються у координати границь, фільтруються за порогом ймовірності та обробляються за допомогою NMS.

SSD є привабливою архітектурою завдяки своїй простоті, високій швидкості та можливості повної паралелізації обчислень, але й має певні обмеження: чутливість до малих об’єктів, залежність від правильного підбору параметрів anchor boxes і необхідність балансування позитивних і негативних прикладів під час навчання.

2.1.4.1 Реалізація SSD та її компонентів

Була використана стандартна реалізація у бібліотеці pytorch, де й можна її переглянути. Архітектура складається з кількох основних компонентів: backbone, модуля побудови багатомасштабних ознак, голів передбачення (classification та regression), генератора anchor boxes та блоку постобробки результатів. Як backbone використовується VGG16, але в модифікованому вигляді – повнозв’язні шари видаляються і замінюються на згорткові. Зокрема, шар FC6 реалізовано як згортку з дилатацією (eng. dilated convolution), що дозволяє збільшити поле сприйняття без втрати роздільної здатності, а FC7 – як звичайну згортку 1×1.

Після backbone додаються додаткові згорткові блоки, які формують набір карт ознак різного масштабу. Кожен наступний блок зменшує просторову роздільну здатність і збільшує поле сприйняття. У результаті формується піраміда ознак, де різні рівні відповідають за виявлення об’єктів різного розміру.

Для кожної карти ознак генеруються anchor boxes за допомогою модуля DefaultBoxGenerator. Кількість, масштаби та співвідношення сторін цих рамок задаються заздалегідь і відрізняються для кожного рівня

піраміди. Їх генерація відбувається під час виконання моделі і залежить від розмірів карт ознак.

Prediction heads реалізовані у вигляді окремих згорткових шарів для кожної карти ознак.. Важливою деталлю є те, що результати з різних карт ознак перетворюються до єдиного формату, після чого об’єднуються.

Під час навчання використовується стандартна процедура зіставлення. Кожному anchor призначається або відповідний об’єкт, або він вважається фоновим. При цьому координати кодуються та масштабується за допомогою BoxCoder, що є важливим для стабільності навчання.

Під час отримання передбачень, модель працює у кілька етапів. Спочатку результати класифікації проходять через softmax. Потім координати рамок декодуються з урахуванням anchor boxes. Далі застосовується фільтрація за порогом впевненості та обмеження кількості кандидатів . На фінальному етапі використовується NMS, який видаляє дублюючі рамки, залишаючи лише найкращі.

Окремим важливим компонентом є блок transform, який відповідає за підготовку вхідних даних. Усі зображення масштабуються до фіксованого розміру 300×300, нормалізуються та об’єднуються в партію.

Безпосередньо реалізацію можна переглянути на рисунку А.7.

2.2 Додаткові механізми обробки

Для деяких архітектур в рамках їх реалізація була потреба в виконанні додаткової обробки даних для сумісності з обробкою архітектури. Повний програмний пайплайн включає три основні етапи: попередню обробку зображень, виконання передбачення моделі та збереження отриманих результатів у базі даних MongoDB. Додатково реалізовано механізм поетапної обробки даних для запобігання перевантаженню оперативної пам’яті під час роботи з великими наборами зображень.

На етапі попередньої обробки зображення зчитуються зі списку файлів, що містить шляхи до зображень датасету. Для кожного зображення виконується відкриття файлу та передача його до відповідного обробника, який забезпечує приведення даних до формату, необхідного для архітектури. Зображення масштабуються до фіксованого розміру пікселів, в залежності від спроможностей самої архітектури. Використання фіксованого розміру входу забезпечує стабільність обчислювального графа.

Окрема увага приділяється обробці зображень у відтінках сірого. Оскільки backbone моделей очікує триканальний RGB-вхід, одноканальні зображення перетворюються у псевдо-RGB шляхом дублювання каналу яскравості у три канали. Це дозволяє використовувати модель без модифікації архітектури та без повторного навчання ваг.

Разом із підготовленими тензорами зберігається додаткова інформація, зокрема оригінальний розмір зображення та ідентифікатор зображення. Збереження початкових розмірів є необхідним, оскільки модель під час отримання передбачень працює з масштабованими зображеннями, а повернення координат об’єктів до реальних розмірів виконується на етапі постобробки. Для зменшення використання пам’яті після обробки кожного зображення примусово виконується очищення невикористаних об’єктів, що є важливим через значні обсяги проміжних тензорів.

Після завершення попередньої обробки виконується передбачення моделі, а після виконання прямого проходу результати проходять постобробку за допомогою image_processor. На цьому етапі координати границь, які були отримані у нормалізованій формі відносно масштабованого зображення, перетворюються у координати піксельного простору з урахуванням реального розміру зображення. Також застосовується поріг довіри для відбору об’єктів.

Збереження результатів здійснюється в базу даних MongoDB. Для кожного передбаченого об’єкта створюється окремий запис у колекції. Такий підхід дозволяє зберігати повний набір передбачень без попередньої агрегації, що спрощує подальший аналіз, обчислення метрик або зміну порогів відбору без повторного запуску отримання передбачень.

З огляду на високі вимоги до оперативної пам’яті, реалізовано механізм часткової обробки датасету. Повний список зображень розбивається на декілька частин, кожна з яких обробляється окремо. Такий підхід дозволяє працювати з великими наборами даних без перевищення доступної пам’яті. Після завершення обробки кожної частини виконується перехід до наступного сегмента, що забезпечує контрольоване використання ресурсів системи.

2.3 Організація наборів даних та подальші компоненти обробки

У межах даної роботи за основу побудови датасетів, що забезпечують масштабованість та зручність подальшої обробки, було взято стандартну побудову у межах бібліотеки ultralytics, включаючи гнучкі конфігураційні файли. Структуру наведено на рис. 2.8, з прикладом з папкою coco.

Створення yaml-файлу було обов’язковим для роботи YOLO 13, так як вона не входить до стандартних архітектур бібліотеки, але при цьому реалізовано на її основі. Текстові ж фали описують місце знаходження необхідних зображень для їх обробки.

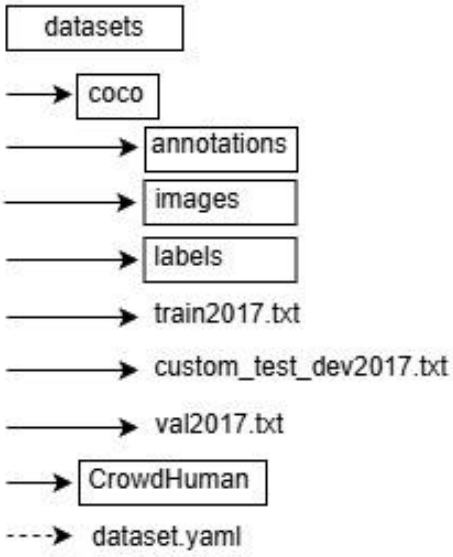


Рисунок 2.8 – Структура датасетів

Папка labels слугує для зберігання текстових файлів типу YOLO, що включає клас та сегментацію об’єктів, папка annotations – вже перетворені на основі сегментів анотаційні файли.

Досліджуючи структуру та файли можна було помітити відсутність тестової анотаційних даних, що викликало потребу у створенні його власноруч. Варто зауважити, що створення такого роду даних є річчю суб’єктивною через координатні положення границь. Через це було вирішено використання механізм автоанотації з використання моделі YOLO 13x, та їх сегментації – Segment Anything Model (SAM) 2.1 large, що можна вважити достатнім для порівняння архітектур між собою через те, що кінцеве формування границь відбувається за сегментами. Зокрема, використано розширений підхід до формування тестового набору на основі 41к зразків у форматі COCO, замість базових 20к. Зазвичай використовують саме 20к даних, але в нашому випадку тест роботи моделей буде нести інший порівняльний характер – відносно моделей, а не даних. Тому для більшої надійності у отриманих результатах було вирішено розширити ці дані, про це каже файл custom_test_dev2017.txt.

Першочерговим етапом є формування структури зберігання даних, яка враховує розподіл на тренувальні, валідаційні та тестові дані. Для підтримки різні JSON форматів анотації, важливою складовою є стандартизація представлення даних для забезпечення сумісності між вихідними даними до зручного формату, в моєму випадку COCO формату. Взагалі між різними типами формату полягає у різниці формування координат для границь та формування так званих анотаційних фактичних даних (eng. the ground truth annotations) та передбачень (eng. predictions). На рис 2.9 наведено як саме виглядає структура анотації та різницю між форматами.

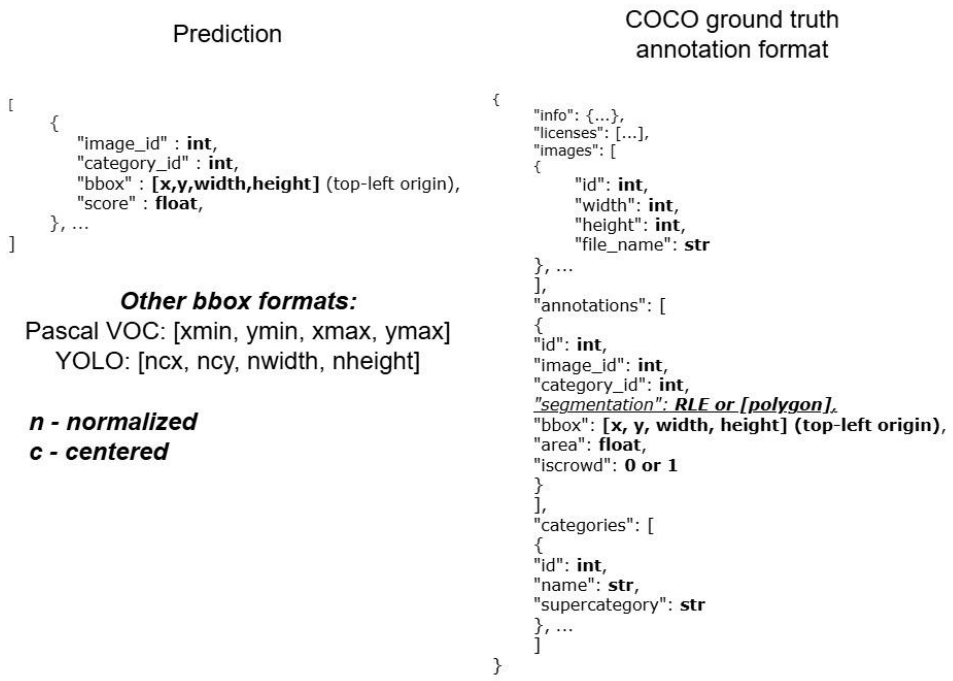


Рисунок 2.9 – Структура анотацій та різні типи координат границь

Важливим аспектом є інтеграція датасету CrowdHuman. Для цього було реалізовано процес конвертації вихідного формату CrowdHuman у формат, сумісний із COCO. Це включає перетворення анотацій з оригінального формату Object Detection Ground Truth (ODGT) у JSON-

структуру, що забезпечує узгодженість між різними наборами даних та спрощує їх спільне використання. Для цього був знайдений скрипт, який дозволяв обробляти даний формат файлу. На основі оброблених даних CrowdHuman було сформовано датасет, структурно подібний до COCO, що дозволяє використовувати його для задач оцінювання моделей у стандартизованому середовищі.

Додатково розглянуто питання роботи з великими обсягами даних, зокрема після їх експорту з бази даних MongoDB. Було використано механізм для роботи з важкими файлами розмір яких доходив до близько 1 ГБ або більше, що дозволило ефективно працювати навіть із значними обсягами інформації. За допомогою JSON оброблювача jq стала можлива безпосередня фільтрація даних для проведення подальших маніпуляцій над даними. Таким чином, запропонована такі кроки забезпечують узгодженість форматів та підготовки даних для оцінювання моделей.

Усі пов’язані з описаними процесами перетворювання та обробка наведена на рис. А.1, А.2, А.5, А.8.

2.4 Перешкоди до виконання дослідження

У процесі виконання дослідження виник ряд обмежень та технічних перешкод, які впливали на різні аспекти виконання роботи.

Однією з ключових проблем став підбір підходящого датасету для проведення експериментів великого обсягу Objects365 (приблизно 1 ТБ), щоб більш детально допоміг дослідити поведінку архітектур в рамках 365 класів.

Пов’язана проблема також стосувалася й датасету з перекритими даними. Один з доцільних варіантів було би застосування спеціалізованого датасету Image Recognition Under Occlusion (IRUO), який був доступний тыльки в рамках. Обмежений доступ до даних ускладнював їх

використання, а також вимагав додаткових процедур для отримання дозволів і доступу до відповідних ресурсів. Аналогічно, отримання деяких попередньо навчених моделей також потребувало окремих запитів, що уповільнювало експерименти.

Суттєві труднощі виникали під час роботи з програмною реалізацією моделей, зокрема через складність архітектур та необхідність компіляції окремих компонентів. Деякі модулі мали специфічні залежності, що вимагало точного налаштування середовища виконання. Через це деякі з архітектур не вдалося використати у роботі.

Окремою проблемою стала несумісність версій програмного забезпечення, зокрема інтерпретатора Python та пов’язаних бібліотек. Частина досліджень у наукових джерелах проводилась із використанням конкретних версій технологій, що не завжди узгоджувались із сучасними інструментами. Це потребувало або адаптації коду, або відтворення середовища з використанням застарілих версій**.** Наприклад, як у випадку з CenterNet архітектурою де всі тести проводилися на Ubuntu 16.04 з Anaconda Python 3.6 та з використанням PyTorch v0.4.1, та було за потрібне скомпілювати певний вид згорткової мережі, де й виникли проблеми.

Крім того, з метою вирішення проблеми з недостатньою кількістю оперативної пам’яті було прибрано передбачення зі значенням рівня впевненості менше 5 відсотків у тестових даних. Це могло незначно вплинути на репрезентативність, однак дозволило забезпечити практичну здійсненність експериментів у наявних умовах. Для опрацювання даних був використаний той же препроцесор jq.

3 РЕЗУЛЬТАТИ МОЖЛИВОСТІ ВИКОРИСТАННЯ АРХІТЕКТУР У ВИПАДКУ ЧАСТКОВО ВИДИМИХ ДАНИХ

3.1 Метод оцінювання результатів

Оцінювання якості побудованих моделей здійснюється з використанням стандартних метрик, прийнятих у задачах детекції об’єктів. Основну увагу приділено метрикам середньої точності (mAP) та середньої повноти (mAR), які забезпечують комплексну характеристику ефективності моделі.

Ключовим елементом обчислення обох метрик є показник IoU (Intersection over Union), який визначає ступінь перекриття між передбаченою та еталонною обмежувальними рамками. Результати оцінювання залежать від вибору порогових значень IoU, яка обчислюється за формулою 1.17, зазвичай використовується діапазон значень (наприклад, від 0.5 до 0.95 з кроком 0.05 або ж @0.5:0.95), що дозволяє отримати більш узагальнену оцінку якості детекції, але й існують інші значення, які здатні оцінити модель у більш складних випадках.

Важливим доповненням до оцінювання є рівень впевненості (eng. confidence level), який визначає ймовірність того, що передбачена моделлю детекція є коректною. Для кожного знайденого об’єкта модель генерує відповідне значення впевненості, яке використовується для фільтрації результатів. Зміна порогу впевненості безпосередньо впливає на баланс між точністю та повнотою: підвищення порогу зменшує кількість хибнопозитивних детекцій, але може призвести до втрати частини істинних об’єктів, і навпаки.

Метрика mAP відображає середню точність детекції об’єктів і обчислюється на основі значень Precision–Recall для різних порогів. Вона враховує як здатність моделі правильно ідентифікувати об’єкти. У свою

чергу, mAR характеризує повноту виявлення об’єктів і показує, яка частка реальних об’єктів була знайдена моделлю. У формулах 3.1-3.4 можна побачити як вони працюють включаючи стандарті метрики AP та AR.

Додатково враховуються параметри maxDets та Area. Параметр maxDets визначає максимальну кількість детекцій, які враховуються під час оцінювання для одного зображення, що впливає на обчислення повноти. Параметр Area дозволяє аналізувати якість детекції для об’єктів різного розміру (малих, середніх та великих, або ж усіх одразу), що є важливим для комплексного аналізу поведінки моделі.

Для проведення оцінювання використовується стандартний інструментарій Faster-COCO-Eval, який забезпечує уніфіковану процедуру обчислення метрик та дозволяє порівнювати результати з іншими дослідженнями, при цьому працюючи 3-4 рази швидше, порівнянно з стандртним pycocotools. Використання цього підходу гарантує відтворюваність експериментів і відповідність загальноприйнятим практикам у галузі. Реалізацію оцінки наведено на рис. А.4.

3.2 Результати первісної оцінки

Первісна оцінка моделей проводилась на валідаційній та тестовій вибірках із використанням метрик mAP та mAR у різних конфігураціях її параметрів. Їх загальне призначення можна трактувати так:

- IoU визначає точність локалізації об’єкта – тобто, наскільки добре передбачена рамка співпадає з еталонною. Вищі пороги IoU означають більш строгі вимоги до точності;
- Розмір об’єктів дозволяє оцінити, як модель працює в різних умовах. Зазвичай малі об’єкти розпізнаються значно гірше, ніж великі;

• Параметр maxDets обмежує кількість детекцій на зображення. Менші значення підвищують precision, але знижують recall, тоді як більші – навпаки.

Результати валідаційної вибірки можна переглянути на рис. 3.1.

				VALIDATION				
				Models				
				YOLO12	YOLO13	SSD	RT-DETRv2	DEIMv2
Metric	IoU	Area	maxDets	(imsz=700x700,bs=16)	(imsz=700x700,bs=16)	(imsz=300x300,bs=1)	(imsz=700x700,bs=1)	(imsz=640x640,bs=1)
mean Average Precision (mAP)	0.50:0.95	all	100	**0.553**	0.551	0.251	0.481	0.534
mean Average Precision (mAP)	0.5	all	100	**0.725**	0.725	0.415	0.708	0.700
mean Average Precision (mAP)	0.75	all	100	**0.604**	0.601	0.262	0.508	0.585
mean Average Precision (mAP)	0.50:0.95	small	100	**0.402**	0.394	0.055	0.254	0.352
mean Average Precision (mAP)	0.50:0.95	meduim	100	**0.613**	0.608	0.268	0.532	0.592
mean Average Precision (mAP)	0.50:0.95	large	100	0.704	0.700	0.435	0.703	**0.713**
mean Average Recall (mAR)	0.50:0.95	all	1	**0.399**	0.401	0.239	0.360	0.396
mean Average Recall (mAR)	0.50:0.95	all	10	**0.669**	0.662	0.344	0.590	0.653
mean Average Recall (mAR)	0.50:0.95	all	100	**0.715**	0.708	0.365	0.649	0.712
mean Average Recall (mAR)	0.50:0.95	small	100	**0.562**	0.560	0.088	0.409	0.540
mean Average Recall (mAR)	0.50:0.95	meduim	100	**0.771**	0.762	0.406	0.710	0.763
mean Average Recall (mAR)	0.50:0.95	large	100	0.847	0.853	0.602	**0.874**	0.863
mean Average Recall (mAR)	0.5	all	100	**0.905**	0.903	0.604	0.900	0.900
mean Average Recall (mAR)	0.75	all	100	**0.779**	0.773	0.368	0.688	0.774

Рисунок 3.1 – Результати оцінок на валідаційних даних

На валідаційній вибірці найкращі результати за метрикою mAP (IoU = 0.50:0.95) продемонстрували моделі YOLOv12 та YOLOv13, які мають практично однакову точність. Модель DEIMv2 показала близький результат, тоді як RT-DETRv2 та SSD поступаються за якістю детекції.

Результати тестової вибірки можна переглянути на рис. 3.2.

				TEST				
				Models				
				YOLO12	YOLO13	SSD	RT-DETRv2	DEIMv2
Metric	IoU	Area	maxDets	(imsz=700x700,bs=16)	(imsz=700x700,bs=16)	(imsz=300x300,bs=1)	(imsz=700x700,bs=1)	(imsz=640x640,bs=1)
mean Average Precision (mAP)	0.50:0.95	all	100	**0.702**	0.777	0.291	0.590	0.615
mean Average Precision (mAP)	0.5	all	100	**0.865**	0.945	0.479	0.805	0.767
mean Average Precision (mAP)	0.75	all	100	**0.784**	0.879	0.305	0.636	0.684
mean Average Precision (mAP)	0.50:0.95	small	100	**0.486**	0.572	0.035	0.292	0.388
mean Average Precision (mAP)	0.50:0.95	meduim	100	**0.684**	0.765	0.210	0.564	0.592
mean Average Precision (mAP)	0.50:0.95	large	100	**0.798**	0.877	0.458	0.746	0.718
mean Average Recall (mAR)	0.50:0.95	all	1	**0.482**	0.510	0.265	0.430	0.448
mean Average Recall (mAR)	0.50:0.95	all	10	**0.787**	0.814	0.383	0.697	0.730
mean Average Recall (mAR)	0.50:0.95	all	100	**0.829**	0.848	0.402	0.745	0.776
mean Average Recall (mAR)	0.50:0.95	small	100	**0.686**	0.709	0.073	0.502	0.626
mean Average Recall (mAR)	0.50:0.95	meduim	100	**0.832**	0.845	0.345	0.745	0.771
mean Average Recall (mAR)	0.50:0.95	large	100	**0.897**	0.928	0.589	0.872	0.842
mean Average Recall (mAR)	0.5	all	100	**0.985**	0.998	0.649	0.958	0.931
mean Average Recall (mAR)	0.75	all	100	**0.924**	0.949	0.413	0.812	0.863

Рисунок 3.2 – Результати оцінок на тестових даних

На тестовій вибірці спостерігається суттєве зростання значень mAP для всіх моделей. Найвищий результат демонструє YOLOv13, що логічно у разі застосування її при автоанотації. Саме тому нам важливо подивитися різницю між різними моделями, нечіпляючи цю модель. За нею слідує YOLOv12, як її найближча версія у сімействі. Інші моделі також показують покращення: DEIMv2, RT-DETRv2, SSD. Водночас слід враховувати, що підвищення показників може зумовлене використанням автоанотації, яка базується на YOLOv13, що потенційно створює певне зміщення оцінки на користь подібних архітектур.

При більш детальному порівняння за певними параметрами можна зробити наступні висновки:

• При IoU = 0.5 всі моделі демонструють значно вищі значення точності. При більш строгому порозі IoU = 0.75 значення знижуються, однак

загальна ієрархія моделей зберігається. Це свідчить про стабільність моделей навіть при підвищених вимогах до точності локалізації.

- Для малих об’єктів всі моделі демонструють суттєво гірші результати. Найкраща модель YOLOv13 на тесті, тоді як SSD майже не справляється із задачею. Це вказує на складність детекції дрібних об’єктів. Для середніх і великих об’єктів спостерігається значне покращення. Наприклад, для великих об’єктів лідери YOLOv13 та YOLOv12. У цій категорії також добре себе показує RT-DETRv2.

- За метрикою mAR на валідації найкращі результати при maxDets=100 демонструють YOLOv12 та YOLOv13. На тесті ці значення зростають до 0.829 та 0.848 відповідно. Зі збільшенням параметра maxDets спостерігається зростання повноти для всіх моделей, що є закономірним. Особливо це помітно при переході від maxDets=1 до maxDets=100. Для великих об’єктів всі моделі досягають найвищих значень mAR, тоді як для малих – значення суттєво нижчі.

Отримані результати демонструють, що моделі сімейства YOLO (особливо YOLOv13) є найбільш ефективними серед розглянутих як за точністю, так і за повнотою. DETR-подібні підходи показують стабільні, але дещо нижчі результати, тоді як SSD значно поступається сучасним архітектурам.

Основною проблемною зоною для всіх моделей залишається детекція малих об’єктів, що підтверджується низькими значеннями як mAP, так і mAR у відповідній категорії.

3.3 Основні результати роботи на частково видимих даних

Оцінювання моделей на частково видимих об’єктах показало суттєве зниження якості детекції порівняно з повністю видимими даними. Це

підтверджує складність задачі в умовах перекриття об’єктів. Загальні результати можна побачити на рис. 3.3.

OCCLUDED								
				Models				
				YOLO12	YOLO13	SSD	RT-DETRv2	DEIMv2
Metric	IoU	Area	maxDets	(imsz=700x700,bs=16)	(imsz=700x700,bs=16)	(imsz=300x300,bs=1)	(imsz=700x700,bs=1)	(imsz=640x640,bs=1)
mean Average Precision (mAP)	0.50:0.95	all	100	0.202	0.195	0.107	0.166	**0.205**
mean Average Precision (mAP)	0.5	all	100	**0.452**	0.436	0.313	0.411	0.450
mean Average Precision (mAP)	0.75	all	100	0.121	0.155	0.052	0.115	**0.164**
mean Average Precision (mAP)	0.50:0.95	small	100	**0.323**	0.317	0.174	0.261	0.321
mean Average Precision (mAP)	0.50:0.95	meduim	100	-1.000	-1.000	-1.000	-1.000	-1.000
mean Average Precision (mAP)	0.50:0.95	large	100	-1.000	-1.000	-1.000	-1.000	-1.000
mean Average Recall (mAR)	0.50:0.95	all	1	0.026	0.025	0.020	0.025	**0.027**
mean Average Recall (mAR)	0.50:0.95	all	10	0.191	0.187	0.118	0.166	**0.193**
mean Average Recall (mAR)	0.50:0.95	all	100	0.346	0.342	0.184	0.285	**0.347**
mean Average Recall (mAR)	0.50:0.95	small	100	0.346	0.342	0.184	0.285	**0.347**
mean Average Recall (mAR)	0.50:0.95	meduim	100	-1.000	-1.000	-1.000	-1.000	-1.000
mean Average Recall (mAR)	0.50:0.95	large	100	-1.000	-1.000	-1.000	-1.000	-1.000
mean Average Recall (mAR)	0.5	all	100	0.651	0.645	0.471	0.603	**0.652**
mean Average Recall (mAR)	0.75	all	100	**0.320**	0.314	0.120	0.236	0.319

Рисунок 3.3 – Результати оцінок на перекритих даних

За основною метрикою mAP (IoU = 0.50:0.95) найкращі результати демонструють моделі DEIMv2 та YOLOv12, яка трохи відстає, тоді як YOLOv13 має дещо нижчий показник. Модель RT-DETRv2 показує середній результат, а SSD значно поступається.

При менш строгому порозі IoU = 0.5 всі моделі демонструють зростання точності. Найкращі результати мають YOLOv12 та DEIMv2, тоді як YOLOv13 і RT-DETRv2 показують дещо нижчі значення. При більш строгому IoU = 0.75 точність суттєво знижується для всіх моделей, де лідером стає DEIMv2, що свідчить про кращу локалізацію у складних умовах.

Аналіз за розмірами об’єктів показує, що оцінювання фактично доступне лише для малих об’єктів, де найкращі результати демонструють YOLOv12 та DEIMv2. Для середніх і великих об’єктів метрики відсутні, що вказує на недостатню кількість таких об’єктів у вибірці або неможливість коректного оцінювання.

За метрикою повноти mAR спостерігається аналогічна тенденція. При maxDets=100 найкращі результати демонструє DEIMv2, практично на рівні з YOLOv12 та YOLOv13. SSD знову показує найгірші результати, а для малих об’єктів значення mAR збігаються із загальними.

Зі зменшенням параметра maxDets повнота різко падає для всіх моделей, що свідчить про необхідність врахування більшої кількості детекцій у складних сценах із перекриттями.

ВИСНОВКИ

У магістерській роботі вирішено актуальну науково-прикладну задачу дослідження ефективності сучасних архітектур комп’ютерного зору для класифікації та детекції об’єктів в умовах часткової оклюзії. У результаті проведеного дослідження сформовано комплексний підхід до аналізу моделей, підготовки даних та оцінювання їх якості в умовах втрати частини візуальної інформації. Отримані результати створюють надійну основу для подальших досліджень і вдосконалення систем розпізнавання об’єктів у складних умовах.

Усі поставлені у вступі задачі дослідження були успішно вирішені:

1. Проведено аналіз сучасних архітектур для задач класифікації, визначено їх ключові особливості та обмеження;
2. Досліджено обрані моделі та специфіку їх практичного застосування;
3. Розроблено та адаптовано методи підготовки й обробки даних відповідно до вимог різних архітектур;
4. Підібрано та організовано набори даних для базового та поглибленого оцінювання моделей, зокрема з урахуванням оклюзії;
5. Сформовано та використано анотаційні дані для валідації та тестування моделей;
6. Проведено валідаційне оцінювання архітектур із використанням сучасних метрик якості (mAP, mAR) ;
7. Виконано порівняльний аналіз моделей на стандартних та оклюзивних даних;
8. Досліджено ефективність архітектур у складних умовах часткової видимості об’єктів.

Основні наукові та практичні результати роботи полягають у наступному:

- **Теоретичне обґрунтування та аналіз.** На основі дослідження сучасних підходів у галузі комп’ютерного зору систематизовано знання про особливості роботи архітектур детекції об’єктів в умовах оклюзії. Визначено ключові фактори, що впливають на точність моделей, зокрема здатність враховувати контекстну інформацію та обробляти частково втрачені ознаки;

- **Програмна імплементація.** У роботі використано офіційні реалізації та готові програмні рішення сучасних архітектур на базі мови програмування Python і бібліотеки PyTorch. Такий підхід дозволив зосередитися безпосередньо на дослідженні ефективності моделей, їх налаштуванні та адаптації до різних типів даних і умов експерименту, забезпечивши коректність порівняння та відтворюваність отриманих результатів;

- **Експериментальний аналіз ефективності.** Проведене дослідження показало суттєвий вплив рівня оклюзії на якість розпізнавання. Встановлено, що різні архітектури демонструють різну стійкість до втрати візуальної інформації, а використання відповідних методів підготовки даних може частково компенсувати негативний вплив оклюзії;

- **Практична цінність.** Отримані результати можуть бути використані для вибору та оптимізації моделей у прикладних системах комп’ютерного зору. Запропоновані підходи дозволяють підвищити надійність роботи систем у реальних умовах, зокрема в задачах відеоспостереження, автономного керування та робототехніки.

Перспективи подальших досліджень полягають у розробці нових методів підвищення стійкості моделей до оклюзії, зокрема шляхом використання ансамблевих підходів, удосконалення механізмів уваги, а також інтеграції моделей із методами генеративного відновлення втрачених частин зображення. Крім того, доцільним є розширення експериментальної

бази та дослідження поведінки моделей у ще більш складних і наближених до реальних умов сценаріях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Lin T.-Y., Goyal P., Girshick R., He K., Dollár P. Focal Loss for Dense Object Detection. URL: https://arxiv.org/abs/1708.02002 (дата звернення: 28.04.2026).

2. Gupta C., Gill N. S., Gulia P., Kumar A., Karamti H., Moges D. M. An Optimized YOLO NAS Based Framework for Realtime Object Detection // Scientific Reports. 2025. URL: https://www.nature.com/articles/s41598-025-17919-w (дата звернення: 01.03.2026).

3. Information retrieval – Average precision // Wikipedia. URL: https://en.wikipedia.org/w/index.php?title=Information_retrieval&oldid=793358396#Average_precision (дата звернення: 20.04.2026).

4. Zhang Z., He T., Zhang H., Zhang Z., Xie J., Li M. Bag of Freebies for Training Object Detection Neural Networks. URL: https://arxiv.org/pdf/1902.04103 (дата звернення: 12.04.2026).

5. Roboflow. Object Detection Metrics. URL: https://blog.roboflow.com/object-detection-metrics/ (дата звернення: 22.01.2026).

6. Roboflow. What Is an Anchor Box? URL: https://blog.roboflow.com/what-is-an-anchor-box/ (дата звернення: 03.03.2026).

7. Liu W., Anguelov D., Erhan D., Szegedy C., Reed S., Fu C.-Y., Berg A. C. SSD: Single Shot MultiBox Detector. URL: https://arxiv.org/pdf/1512.02325 (дата звернення: 10.01.2026).

8. Rezatofighi H., Tsoi N., Gwak J. Y. та ін. Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression. URL: https://giou.stanford.edu/GIoU.pdf (дата звернення: 26.04.2026).

9. Tri D., Daniel Y. F., Stefano E., Arti R., Christofer R. FLASHATTENTION: Fast and Memory-Efficient Exact Attention with IO-

Awareness. URL: https://arxiv.org/pdf/2205.14135 (дата звернення: 11.03.2026).

10. Golden A., Hsia S., Sun F., Acun B., Hosmer B., Lee Y., DeVito Z., Johnson J., Wei G.-Y., Brooks D., Wu C.-J. Is Flash Attention Stable? URL: https://arxiv.org/pdf/2405.02803 (дата звернення: 11.03.2026).

11. Tan M., Pang R., Le Q. V. EfficientDet: Scalable and Efficient Object Detection. Google Research, Brain Team. URL: https://arxiv.org/pdf/1911.09070 (дата звернення: 25.01.2026).

12. Milvus. What Are the Different Types of Object Detection Models? URL: https://milvus.io/ai-quick-reference/what-are-the-different-types-of-object-detection-models (дата звернення: 22.02.2026).

13. Kassaw K., Luzi F., Collins L. M., Malof J. M. Are Deep Learning Models Robust to Partial Object Occlusion in Visual Recognition Tasks? URL: https://arxiv.org/pdf/2409.10775 (дата звернення: 02.03.2026).

14. The Python Code. Non-Maximum Suppression using OpenCV in Python. URL: https://thepythoncode.com/article/non-maximum-suppression-using-opencv-in-python (дата звернення: 22.01.2026).

15. Sapkota R., Cheppally R. H., Sharda A. YOLO26: Key Architectural Enhancements and Performance Benchmarking for Real-Time Object Detection. URL: https://arxiv.org/pdf/2509.25164 (дата звернення: 03.04.2026).

16. Kortylewski A., He J., Liu Q., Yuille A. Compositional Convolutional Neural Networks: A Deep Architecture with Innate Robustness to Partial Occlusion. URL: https://arxiv.org/abs/2003.04490 (дата звернення: 08.03.2026).

17. Guo B., Zhang H., Wang H., Li X., Jin L. Adaptive Occlusion Object Detection Algorithm Based on OL-IoU. // Scientific Reports. 2024. URL: https://www.nature.com/articles/s41598-024-78959-2 (дата звернення: 07.01.2026).

18. Wang A., Sun Y., Kortylewski A., Yuille A. Robust Object Detection under Occlusion with Context-Aware CompositionalNets. URL: https://arxiv.org/abs/2005.11643 (дата звернення: 20.02.2026).

19. Huang S., Hou Y., Liu L., Yu X., Shen X. Real-Time Object Detection Meets DINOv3. URL: https://arxiv.org/pdf/2509.20787 (дата звернення: 11.03.2026).

20. Shao J., Loy C. C., Wang X. CrowdHuman Dataset // GitHub. URL: https://github.com/Asthestarsfalll/CrowdHuman (дата звернення: 01.03.2026).

21. Albumentations Documentation. Bounding Boxes Augmentations. URL: https://albumentations.ai/docs/3-basic-usage/bounding-boxes-augmentations/ (дата звернення: 12.02.2026).

22. Ultralytics. How to train bounding box and polygon data together? Issue #19912. URL: https://github.com/ultralytics/ultralytics/issues/19912 (дата звернення: 12.02.2026).

23. Lv W., Zhao Y., Chang Q., Huang K., Wang G., Liu Y. RT-DETRv2: Improved Baseline with Bag-of-Freebies for Real-Time Detection Transformer. URL: https://arxiv.org/abs/2407.17140 (дата звернення: 28.03.2026).

24. Liao Z., Zhao Y., Shan X., Yan Y., Liu C., Lu L., Ji X., Chen J. RT-DETRv4: Painlessly Furthering Real-Time Object Detection with Vision Foundation Models. URL: https://arxiv.org/abs/2510.25257 (дата звернення: 20.02.2026).

25. FlashAttention GitHub Issue #1622. flash_attn_2_cuda.cpython-310-x86_64-linux-gnu.so: undefined symbol. URL: https://github.com/Dao-AILab/flash-attention/issues/1622 (дата звернення: 15.03.2026).

26. Lei M., Li S., Wu Y., Zheng X., Hu G., Du S., Wu Z., Gao Y. YOLOv13: Real-Time Object Detection with Hypergraph-Enhanced Adaptive Visual Perception. URL: https://arxiv.org/abs/2506.17733 (дата звернення: 13.03.2026).

27. Khanam R., Hussain M. YOLOV11: An Overview of the Key Architectural Enhancements. URL: https://arxiv.org/pdf/2410.17725 (дата звернення: 21.02.2026).

28. Alif M. A. R., Hussain M. YOLOV12: A Breakdown of the Key Architectural Features. URL: https://arxiv.org/pdf/2502.14740v1 (дата звернення: 21.02.2026).

29. Ge Z., Liu S., Wang F., Li Z., Sun J. YOLOX: Exceeding YOLO Series in 2021. URL: https://arxiv.org/pdf/2107.08430 (дата звернення: 22.02.2026).

Додаток А

Програмна реалізація