Міністерство освіти і науки України
Національний університет «Києво-Могилянська академія»
Факультет інформатики
Кафедра математики

**Кваліфікаційна робота**

освітній ступінь – бакалавр

на тему: **«ТРЕНУВАННЯ МОДЕЛІ ДЕТЕКЦІЇ ОБ'ЄКТІВ НА ЧАСТКОВО РОЗМІЧЕНИХ ЗОБРАЖЕННЯХ/OBJECT DETECTOR TRAINING WITH PARTIALLY LABELED IMAGES»**

Виконав: студент 4-го року навчання
освітньої програми «Прикладна математика»,
спеціальності 113 Прикладна математика

Шакіров Артем Тимурович

Керівник: Швай Н. О.
кандидат фіз.-мат. наук, доцент

Рецензент:

Кваліфікаційна робота захищена
з оцінкою _____

Секретар ЕК_____
(підпис)

«_____» _____ 20___р.

Київ - 2025

Міністерство освіти і науки України
Національний університет «Києво-Могилянська академія»
Факультет інформатики
Кафедра математики

ЗАТВЕРДЖУЮ
Зав.кафедри математики,
доцент, кандидат фіз.-мат. наук
_____ *Чорней Р.К.*
(підпис)
“____”_____ 2024

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
для кваліфікаційної роботи
студенту 4-го курсу, факультету інформатики
Шакірову Артему Тимуровичу

**Тема:** «Тренування моделі детекції об’єктів на частково розмічених зображеннях/Object detector training with partially labeled images»

**Зміст кваліфікаційної роботи:**

Анотація

Вступ

1. Огляд існуючих моделей детекції об’єктів та методів використання частково розмічених даних у їх навчанні

2. Опис алгоритму детекції об’єктів YOLO та відповідної функції втрат

3. Побудова методу використання частково розмічених даних в модифікованій процедурі навчання алгоритму YOLO

4. Реалізація експериментів та оцінка результатів

Висновки

Список літератури

Дата видачі “____”_____ 2024 Керівник _____
(підпис)

Завдання отримав _____
(підпис)

# Графік підготовки кваліфікаційної роботи до захисту

Графік узгоджено «_____» _____ 2024р.

| № з/п | Перелік робіт | Термін виконання етапу | Підпис наукового керівника | Дата ознайомлення наукового керівника | Примітка |
|---|---|---|---|---|---|
| 1. | Отримання теми кваліфікаційної роботи. | 01.10.2024 | | | |
| 2. | Ознайомлення з темою кваліфікаційної роботи. | 04.10.2024 | | | |
| 3. | Розробка плану та структури роботи. | 14.10.2024 | | | |
| 4. | Робота з науковою літературою. Написання вступу та анотації. | 28.10.2024 | | | |
| 5. | Побудова процедури навчання YOLO із частковою розміткою та експериментування. | 17.03.2025 | | | |
| 6. | Робота над текстовим оформленням теоретичної частини. | 04.04.2025 | | | |
| 7. | Попередній аналіз кваліфікаційної роботи. Виправлення помилок. | 16.05.2025 | | | |
| 8. | Попередній захист кваліфікаційної роботи. | 23.05.2025 | | | |
| 9. | Захист кваліфікаційної роботи. | 06.06.2025 | | | |

Науковий керівник _____
*(ПІБ)*

Виконавець кваліфікаційної роботи _____
*(ПІБ)*

# Зміст

# Анотація

Ця кваліфікаційна робота присвячена розробці ефективної модифікованої процедури навчання алгоритму YOLO на частково розмічених зображеннях. У роботі було розглянуто теоретичні аспекти тренування моделей детекції об'єктів на неповній розмітці та експериментально перевірено запропоновані підходи.

В першому розділі проаналізовано основи машинного навчання в області задач виявлення об'єктів та комп'ютерного зору.

Другий розділ присвячено детальному аналізу архітектури, механізму прогнозування, функції втрат та модифікацій алгоритму YOLO.

У третьому розділі розглянуто методи навчання на даних з частковою розміткою.

Четвертий розділ містить запропоновані модифіковані підходи для тренування алгоритму YOLO на частково розмічених даних.

У п'ятому розділі представлено результати серії експериментів, проведених із використанням модифікованої процедури навчання, та оцінено їх ефективність.

# Основні поняття

- YOLO(You Only Look Once) – алгоритм детекції об'єктів.
- Обмежувальна рамка(bounding box) – прямокутник, який визначає розташування та розмір об'єкта на зображенні.
- Значення впевненості(confidence score) – числова ймовірність, що обмежувальна рамка містить об'єкт певного класу.
- IoU(Intersection over Union) – відношення площі перетину двох обмежувальних рамок до площі їх об'єднання.
- Якорні рамки(anchor boxes) – набір попередньо визначених фіксованих рамок різних розмірів для спрощення процесу прогнозу моделі детекції.
- Гіперпараметр – параметр моделі, який задається перед початком її навчання.
- Катастрофічне забування(catastrophic forgetting) – явище втрати моделлю знань з попереднього тренування при донавчанні на новому наборі даних.
- Згорткова нейронна мережа(CNN) – глибинна нейронна мережа, що використовує згорткові шари для виокремлення ознак.
- Згортковий шар(convolutional layer) – шар нейронної мережі, який застосовує фільтр до вхідного зображення або попереднього шару.
- Повнозв'язний шар(fully-connected layer) – шар нейронної мережі, де кожен нейрон попереднього шару з'єднаний з кожним поточним нейроном.
- Функція активації – нелінійна функція, яка застосовується до виходу кожного нейрона.

# Вступ

## Актуальність

Детекція об'єктів є однією з ключових та провідних задач машинного навчання і комп'ютерного зору. Стрімкий розвиток цієї галузі сприяв появі багатьох технологій, які застосовуються в безпілотному керуванні, відеоспостереженні, аналізі медичних зображень та інших галузях, де візуальний аналіз є критично важливим.

Ефективне навчання алгоритмів детекції об'єктів зазвичай потребує великих обсягів повністю розмічених зображень, але процес повного анотування даних є трудомістким, дорогим та часто потребує ручної роботи або використання допоміжних моделей, які не завжди можуть гарантувати коректність та повноту анотацій всіх зображень. Своєю чергою, це змушує застосовувати частково розмічені дані, що ускладнює процес навчання та може негативно вплинути на точність моделі, зокрема – для об'єктів, представлених у вибірці недостатньо.

У зв'язку з цим, необхідною є розробка методів, що зможуть ефективно та стабільно працювати з частково розміченими наборами даних. В цій роботі розглядається модифікація сучасного алгоритму детекції об'єктів YOLO, який буде адаптовано для навчання на неповній розмітці.

Тема кваліфікаційної роботи є актуальною. Вона спрямована на покращення процесу тренування сучасних моделей комп'ютерного зору, що дозволить розширити їх використання в реальних умовах, де повна розмітка є неможливою або економічно недоцільною.

## Мета, завдання дослідження

Метою роботи є розробка ефективної модифікованої процедури тренування алгоритму YOLO у випадку частково розмічених навчальних даних.

Досягнення поставленої мети передбачає вирішення наступних завдань:

1. Розглянути сучасні методи комп'ютерного зору для задачі детекції об'єктів.

2. Описати архітектуру алгоритму YOLO та проаналізувати його функцію

втрат.

3. Оглянути існуючі підходи використання частково розмічених даних у задачах класифікації та виявлення об'єктів.

4. Розробити метод використання часткових анотацій у задачі детекції об'єктів шляхом модифікації процедури навчання YOLO.

5. Реалізувати експерименти із використанням розроблених підходів.

6. Оцінити результати та ефективність проведених експериментів у порівнянні із базовою моделлю.

# 1 Моделі детекції об'єктів

Детекція об'єктів є однією з головних задач комп'ютерного зору, що поєднує дві менші підзадачі: локалізацію об'єктів на зображенні та їх класифікацію. На відміну від класичних моделей класифікації зображень, де метою є прогнозування однієї категорії для всього зображення, модель детекції повинна вміти виявляти різні категорії одночасно, точно визначаючи координати їх положення.

Із часом та розвитком технологій вимоги до детекторів значно ускладнились. Тепер навчанні алгоритми повинні були забезпечувати свою роботу в реальному часі та враховувати розмір, ракурс і різні стани об'єкту. Окрім цього, треба було ще забезпечити стабільну роботу із частковим перекриттям іншими об'єктами, різним фоном та недостатністю тренувальних даних.

Для розуміння сучасних вимог до моделей, потрібно проаналізувати їх еволюцію – від класичних підходів до нейромережних архітектур.

## 1.1 Перші моделі детекції об'єктів

До широкої популярності глибинного навчання задачі детекції ґрунтувались на традиційних методах машинного навчання. Вони зазвичай складались з двох етапів: сегментації (проходженням по зображенню із фіксованим вікном для генерації регіонів інтересу) та класифікації цих регіонів із використанням стандартних статистичних моделей.

Одним із перших значних проривів в цій сфері стала поява алгоритму Viola-Jones у 2001 році, який використовувався для виявлення облич в режимі реального часу [1]. Його архітектура використовувала гаароподібні ознаки(Haar-like features) та техніку, яка дозволила ефективно обчислювати суму пікселів у прямокутних областях і отримала назву – інтегральне зображення(integral image). Це дозволило зменшити час обробки на 50 – 70% порівняно із класичними методами та стало основою для ранніх систем розпізнавання облич.

Іншим важливим підходом стала комбінація гістограми напрямлених градієнтів(Histogram of Oriented Gradients, HOG) із методом опорних векторів(Support Vector Machine, SVM) [2]. HOG визначав локальні ознаки завдяки розподілу градієнтів яскравості, а SVM класифікував регіони. Така система

показала високу ефективність, бо була стабільною до різних масштабів, та найчастіше використовувалась для детекції пішоходів та транспортних засобів на дорогах.

Незважаючи на певні успіхи та стабільну роботу цих методів, вони не забезпечували достатньої гнучкості через вимоги до ручного налаштування під кожен тип об'єктів. Також ці підходи були вразливими до змін фону, освітлення або ракурсу та потребували значних обчислювальних витрат.

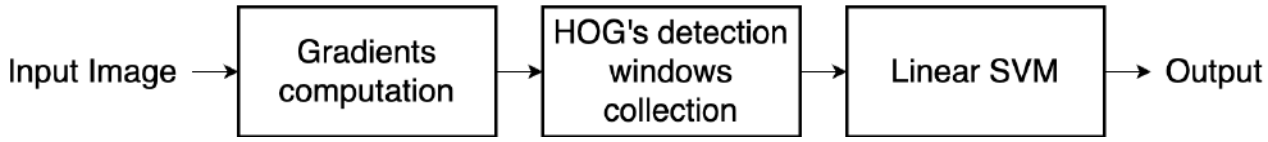



Рис. 1: Схема роботи HOG + SVM

## 1.2 Двоетапні детектори

Із поступовим розвитком глибинного навчання відбулася повна трансформація в методах детекції об'єктів. На відміну від класичних підходів, де потрібно було вручну визначати параметри для виокремлення ознак, нові методи автоматизували цей процес за допомогою згорткових нейронних мереж(CNN). Одним із перших ефективних та вдалих методів, сформувавших основу сучасних моделей, стала архітектура R-CNN(Regions with Convolutional Neural Networks), запропонована Робертом Гіршиком у 2014 році [3].

Головна ідея R-CNN полягала в тому, що традиційний генератор регіонів поєднувався із згортковою нейронною мережею (CNN). Алгоритм вибіркового пошуку (selective search) генерував до 2000 регіонів на зображенні із потенційними об'єктами. Кожен з цих регіонів масштабувався до фіксованого розміру та пропускався через CNN для виділення головних ознак, які потім класифікувалися із використанням методу опорних векторів (SVM) та координати яких уточнювались регресією граничних рамок(bounding box regression).

В порівнянні із першими моделями детекції, R-CNN досягла майже вдвічі кращої середньої точності (mAP) в 53.7% на наборі даних PASCAL VOC 2012, але обробка одного зображення займала близько 50 секунд, що було недостатньо для роботи в реальному часі. Також навчання потребувало окремого

тренування CNN, SVM та регресії рамок, що ускладнювало весь процес.

Для усунення недоліків, у 2015 році було запропоновано нову модель – Fast R-CNN [4]. Ключовим покращенням якої стала оптимізація паралельної обробки завдяки генерації мапи ознак(feature map) з початкового зображення, пропущеного через CNN. Застосовуючи шар регіону інтересу (Region of Interest, RoI), з цієї мапи виокремлювались вектори ознак, що дозволило уникнути повторного обчислення CNN. Класифікація та регресія граничних рамок були об'єднані в єдину мережу.

Ці покращення дозволили досягти швидкості обробки в 2 секунди та збільшили середню точність до 68.8%, однак метод вибіркового пошуку залишався вузьким місцем, бо він все ще вимагав близько 1.5 секунд для кожного зображення.

Революційним кроком стало впровадження згорткової підмережі регіональних пропозицій (Region Proposal Network, RPN) у версії Faster R-CNN [5], що замінило повільний вибірковий пошук. Ця мережа працювала безпосередньо на мапі ознак та самостійно генерувала регіони із ймовірними об'єктами. В результаті це дозволило створити єдину згорткову архітектуру для виявлення, яка значно підвищила швидкість та точність моделі.

Порівняння метрик архітектур наведено в таблиці 1.

| **Параметр** | **R-CNN** | **Fast R-CNN** | **Faster R-CNN** |
|---|---|---|---|
| Час обробки | 50 сек | 2 сек | 0.2 сек |
| mAP | 53.7% | 68.8% | 70.4% |

Табл. 1: Порівняння метрик версій R-CNN

## 1.3 Одноетапні детектори

Двоетапні детектори демонстрували високу точність, але їх швидкість була недостатньою для задач реального часу, як автономне керування або відеоспостереження. Саме тому з'явилася ідея створення одноетапних детекторів, які повинні були поєднати етапи локалізації і класифікації в єдиний крок. Найвідомішими представниками цієї архітектури стали YOLO (You Only Look Once) та SSD (Single Shot Multibox Detector).

### YOLO

Модель YOLO була вперше запропонована Редмоном Джозефом у 2016 році [6]. Вона трансформувала класичну ідею детекції, розглядаючи її як задачу регресії просторових координат та ймовірності класів. Зображення поділялося на сітку розміром $S \times S$, де кожна комірка відповідала за прогноз $B$ кількості координат рамок, впевненості у наявності об'єкта та ймовірності класів.

Цей підхід, що використовував лише один прохід нейронної мережі, дозволив моделі суттєво пришвидшити процес виявлення, обробляючи зображення зі швидкістю 45 кадрів на секунду (FPS).

Попри це, перші версії YOLO поступалися двоетапним детекторам у точності, особливо при присутності на зображенні малих об'єктів.

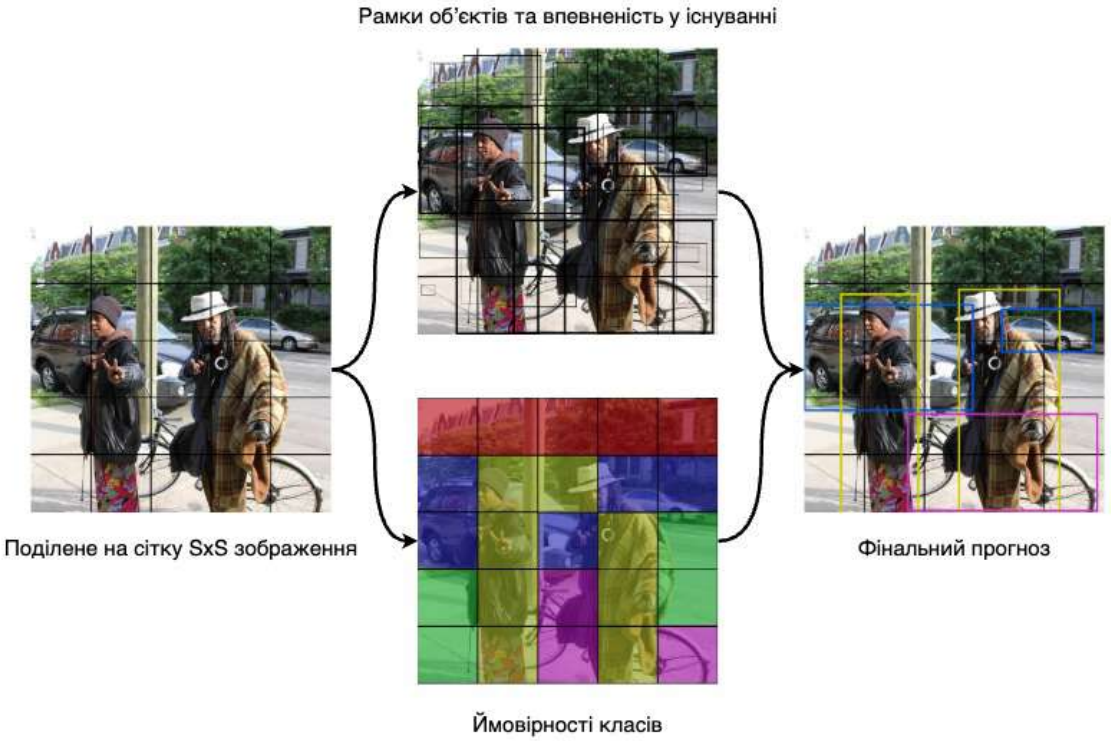



Рис. 2: Ілюстрація роботи YOLO

### SSD

Іншим варіантом одноетапної архітектури стала модель SSD, яка вперше була запропонована в роботі Вея Лю у 2016 році [7]. Вона ґрунтувалась на ідеї базових згорткових мереж, до яких додавались додаткові шари згортки різної

роздільності для роботи з ознаками різних масштабів.

Це дозволило SSD виконувати прогнози з декількох рівнів просторових ознак, що забезпечило кращу локалізацію об'єктів різного розміру. Порівняно з YOLO, SSD показала вищу точність, але зберігала можливість роботи в реальному часі.

Однак, ця архітектура також не була ідеальною, в моделі залишилась проблема із чутливістю до перекриття об'єктів та з'явилась залежність від якісної ініціалізації роздільності для додаткових шарів.

| **Параметр** | **Faster R-CNN** | **YOLOv1** | **SSD** |
|---|---|---|---|
| Швидкість(FPS) | 5 | 45 | 59 |
| mAP | 70.4% | 57.9% | 74.3% |

Табл. 2: Порівняння Faster R-CNN, YOLO та SSD

## 1.4 Сучасні виклики у розробці моделей детекції

Моделі детекції об'єктів досягли значного прогресу протягом останніх років. Використання глибших нейронних мереж, ефективніших методів навчання та оптимізації функцій втрат допомогло їм значно вдосконалитись. Саме тому основний фокус тепер спрямовано на покращення надійності, масштабованості та ефективного використання даних.

### Безякорні детектори

Класичні детектори, зокрема YOLO та SSD, широко використовують техніку якорів(anchor boxes) – заздалегідь заданих рамок різних масштабів, але цей підхід має низку недоліків, таких як дисбаланс між прикладами, бо на 1 об'єкт може припадати до 1000 негативних якорів, та залежність від ретельного налаштування гіперпараметрів.

Сучасні безякорні моделі відмовляються від фіксованих рамок, натомість вони самостійно передбачають центри та розміри об'єктів, що полегшує загальний пайплайн та зменшує кількість гіперпараметрів, необхідних для налаштування, зберігаючи при цьому високу точність.

### Трансформери

Перспективною ідеєю є застосування трансформерів для задач комп'ютерного зору. Однією з перших вдалих архітектур такого типу стала DETR(DEtection TRansformer) від Meta AI [8].

DETR розуміє проблему детекції, як задачу послідовного передбачення за допомогою архітектури кодера-декодера з механізмом самоуваги. На виході модель генерує фіксовану кількість передбачень, що дозволяє їй уникати процесу генерування регіонів.

Головними перевагами цієї архітектури стали: одноетапна детекція та автоматичне відкидання зайвих рамок через глобальну оптимізацію. Недоліком залишилась обчислювальна складність та повільна збіжність, яка потребувала близько 500 епох навчання для ефективної роботи.

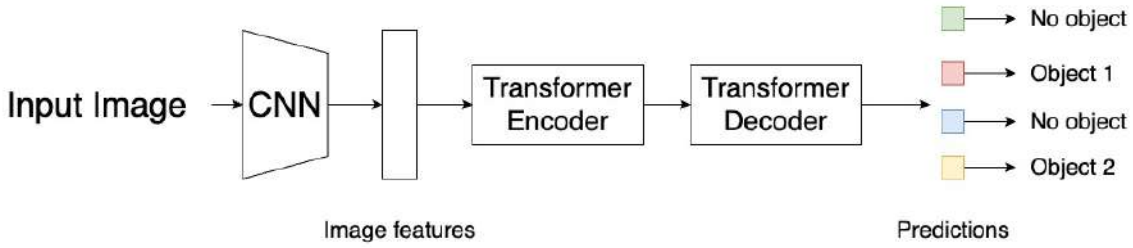



Рис. 3: Архітектура DETR

### Легкі моделі

Зростаючий попит на мобільні та вбудовані застосунки посприяв дослідженню та створенню компактних версій моделей, які зможуть зберегти швидкість та прийнятну точність при значно меншій складності.

Серед розроблених підходів, особливу увагу можна приділити різним версіям YOLO. Наприклад, YOLOv5n, яка зменшила розмір моделі з 40 МБ до 4 МБ, або PP-YOLO [9], що застосовує дистиляцію знань для навчання меншої моделі «учня» на виходах великої «вчителя» із зберіганням точності.

### Навчання на частковій розмітці

Попри всі проблеми, однією із найбільших сьогодні залишається висока вартість та складність повної розмітки даних, що обмежує масштабування досліджень. Для вирішення цього активно досліджуються та розробляються

підходи, які зможуть використовувати частково розмічені дані для ефективного навчання моделей.

Ця робота сфокусована саме на цій проблемі. У ній проекспериментовано із різними методами використання частково розмічених зображень для ефективного навчання моделі.

# 2 Алгоритм детекції об'єктів YOLO

Для досягнення поставлених цілей кваліфікаційної роботи було обрано архітектуру YOLO. Головною причиною стала ефективність моделі в реальному часі, її популярність та відкритий вихідний код, що спрощує процес модифікації і дослідження нових підходів.

## 2.1 Ідея YOLO

Як вже зазначалося раніше, алгоритм YOLO [6] радикально змінив підхід детекції: замість послідовної локалізації регіонів та класифікації кожного, модель вирішувала задачу регресії, де нейронна мережа одночасно передбачала координати об'єктів, впевненість у їх наявності та їх категорію.

Процес виявлення відбувається так: вхідне зображення розбивається на сітку фіксованого розміру $S \times S$ та для кожної комірки мережа передбачує $B$ наборів обмежувальних рамок (bounding box), значень впевненості (confidence score) і ймовірності класів. Такий підхід дозволив детекції YOLO охоплювати зображення повністю та відбуватись за один прохід, значно пришвидшуючи її.

## 2.2 Архітектура YOLO

Архітектура YOLO пройшла через низку змін від версії до версії, але її головна схема залишається незмінною і складається з трьох основних частин: backbone(хребет), neck(шия) та head(голова) [11]. Кожна з них відповідає за свій етап обробки зображення та може бути окремо модифікована, зберігаючи одноетапну структуру.

### Backbone

Головна функція backbone – виділення просторових і семантичних ознак зображення. Це досягається використанням послідовних шарів згорткової мережі, що зменшують розмірність простору зображення, отримуючи його ключові патерни, необхідні для подальшої обробки. Перші версії YOLO v1-v3

використовували архітектуру Darknet [6] зі згортками $3 \times 3$ та $1 \times 1$, що забезпечувало баланс між точністю та швидкістю. Новіші версії перейшли на більш ефективну модифікацію – CSPDarknet [10], яка зменшує обчислювальну складність завдяки розподілу потоку на дві частини, об'єднуючи їх на виході.

### Neck

Neck – проміжна частина між backbone та head, яка поєднує ознаки різної глибини для покращення виявлення об'єктів різного розміру. Для цього застосовувалась структура Feature Pyramid Network(FPN) [11], яку пізніше замінила Path Aggregation Network(PANet) [10]. Ці мережі збирають ознаки з різних шарів backbone та поєднують високорівневі семантичні ознаки із низькорівневими просторовими, що покращує точність всієї моделі, особливо для маленьких об'єктів.

### Head

Head – фінальний модуль, який генерує вихідні прогнози моделі. Він виводить координати обмежувальних рамок, впевненість в існуванні об'єкта та ймовірності класів. Кожен передбачений вектор має таку структуру [6]:

$$[x, y, w, h, confidence, p_1, p_2, \ldots, p_K]$$

де $x, y$ – координати центру рамки, $w, h$ – ширина та висота, $confidence$ – впевненість у наявності об'єкта, $p_c$ – ймовірність належності до класу $c$, $K$ – загальна кількість класів.

Розмір вихідного тензору для $S = 5, B = 2, K = 20$ буде $5 \times 5 \times 50$.

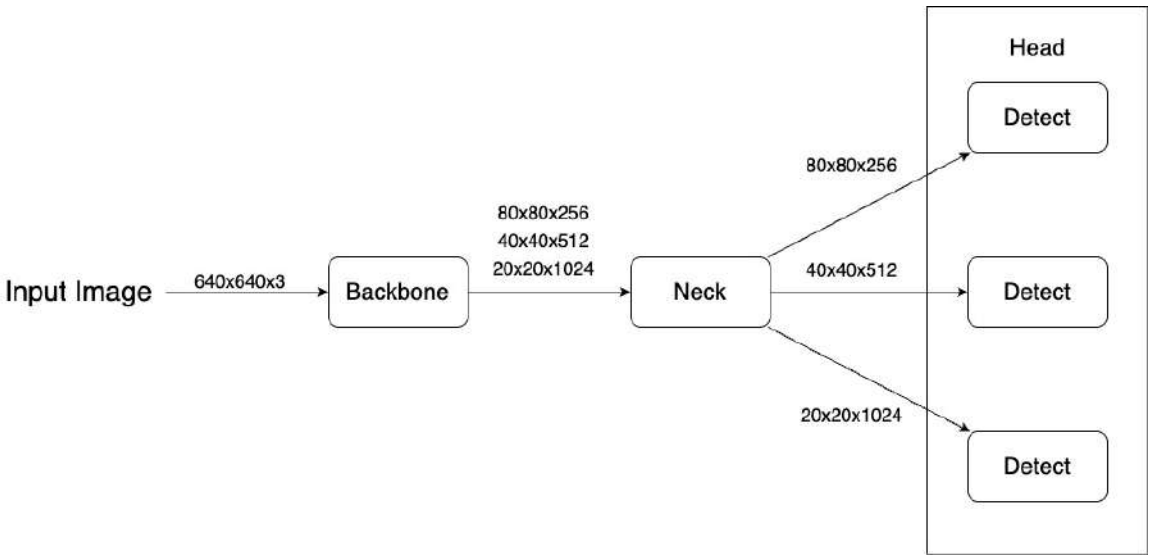



Рис. 4: Архітектура YOLO

## 2.3 Механізм прогнозування

Фінальні прогнози моделі YOLO формуються на вихідних даних модуля head, проте, замість прямого виведення, ці дані вимагають ще додаткової обробки для підвищення точності та уникнення дублювань. В різних версіях YOLO цей механізм відрізняється.

Механізм прогнозування першої версії YOLO складався повністю з повнозв'язного шару (fully-connected layer), що робив передбачення координат напряму для кожної рамки відповідної комірки [6]. Цей підхід працював стабільно, але породжував проблему із повільнішим навчанням моделі, ніж у Faster R-CNN, де використовувалась повністю згорткова інфраструктура, яка видавала зсув від якорних рамок [12]. Тому для наступних версій архітектуру було перероблено під роботу з ними. Прогнозування локації тепер рахувало такі значення:

- Локація рамки: $x = \sigma(t_x) + c_x, y = \sigma(t_y) + c_y$, де $(c_x, c_y)$ – координати верхнього лівого кута комірки, $(t_x, t_y)$ – координати спрогнозованої рамки, $\sigma$ – логістична функція активації.

- Розмір: $w = p_w e^{t_w}, h = p_h e^{t_h}$, де $(p_w, p_h)$ – розміри якоря, $t_w, t_h$ – висота та ширина спрогнозованої рамки.

У версіях, починаючи з YOLOv8, реалізовано безякорний метод, в якому прогнози базуються на прямій регресії [13]:

- Локація рамки: $x = \sigma(t_x) \cdot s + c_x, y = \sigma(t_y) \cdot s + c_y$, де $s$ – розмір мапи ознак (feature map).
- Розмір: $w = t_w \cdot s, h = t_h \cdot s$.

Вихідні значення ймовірності класів [11]:

$$p_i = \sigma(t_c), \text{ де } t_c \text{ – спрогнозована ймовірність класу } c, c = 1, ..., K$$

Значення впевненості в існуванні об'єкта визначається як:

$$confidence = \sigma(t_o), \text{ де } t_o \text{ – спрогнозована ймовірність існування об'єкта}$$

Оскільки модель може спрогнозувати декілька рамок для одного й того ж об'єкта, для усунення дублікатів застосовується метод Non-Maximum Suppression (NMS) [6], який складається з трьох етапів:

1. Сортування за спаданням $confidence$.
2. Обрання рамки із найбільшим значенням.
3. Видалення всіх інших прогнозів, де IoU(Intersection over Union) з обраним прикладом вище заданого порогу.

$$IoU(\hat{B}, B) = \frac{Area(\hat{B} \cap B)}{Area(\hat{B} \cup B)}$$

## 2.4 Функція втрат

Як і в більшості моделей машинного навчання, функція втрат в YOLO відіграє ключову роль в процесі її навчання, оновлюючи ваги мережі. Для цього функція повинна допомогти моделі точно визначити координати рамки об'єктів, навчити відокремлювати об'єкти від фону та правильно їх класифікувати. З цієї причини функція втрат YOLO складається з 3 основних частин [6]: втрати локалізації, втрати об'єктності та класифікаційної втрати.

### Втрата локалізації

Цей компонент відповідає за мінімізацію відхилень між координатами передбачених та справжніх рамок. В перших версіях використовується сума квадратичних відхилень [6]. Щоб зменшити непропорційний вплив великих об'єктів, до ширини та висоти застосовується квадратний корінь.

$$L_{loc} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj}((\hat{x}_i - x_i)^2 + (\hat{y}_i - y_i)^2 + (\sqrt{\hat{w}_i} - \sqrt{w_i})^2 + (\sqrt{\hat{h}_i} - \sqrt{h_i})^2)$$

$\hat{x}, \hat{y}, \hat{w}, \hat{h}$ – передбаченні значення, $x, y, w, h$ – істинні, $\mathbb{1}_{ij}^{obj}$ – індикаторна функція, про існування об'єкта j-й рамці i-ї комірки.

Починаючи з YOLOv4 ця втрата рахується із використанням CIoU [10]:

$$L_{loc} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \cdot L_{CIoU}(\hat{B}_{ij}, B_{ij})$$

$$L_{CIoU}(\hat{B}, B) = 1 - IoU(\hat{B}, B) + \frac{\rho^2(\hat{B}, B)}{c^2} + \alpha v$$

$$v = \frac{4}{\pi^2}(arctan\frac{\hat{w}}{\hat{h}} - arctan\frac{w}{h})^2$$

$$\alpha = \frac{v}{(1 - IoU(\hat{B}, B)) + v}$$

$\rho$ – відстань між центрами, $c$ – діагональ мінімального прямокутника, який може охопити обидві рамки, IoU – Intersection over Union, $B_{ij}$ – справжня рамка.

### Втрата об'єктності

Мета цього компоненту – калібрування впевненості моделі у наявності об'єкта в рамці. Для ранніх версій використовувалась формула [6]:

$$L_{obj} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj}(\hat{C}_i - C_i)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj}(\hat{C}_i - C_i)^2$$

$C \in \{0, 1\}$ – істинна впевненість, $\lambda_{noobj}$ – коефіцієнт балансу дисбалансу класів.

Для версій YOLOv3 і сучасніших застосовується бінарна крос-ентропія [11]:

$$L_{obj} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} -\mathbb{1}_{ij}^{obj}(\hat{C}_i log C_i + (1 - \hat{C}_i) log(1 - C_i)) \\ - \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj}(\hat{C}_i log C_i + (1 - \hat{C}_i) log(1 - C_i))$$

### Класифікаційна втрата

Ця втрата застосовується для максимізації точності розподілу ймовірностей. Як і для попередніх частин, для версій YOLOv1-v2 використана середньоквадратична похибка [6]:

$$L_{cls} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \sum_{c \in K} (\hat{p}_i(c) - p_i(c))^2$$

K – множина всіх класів, p(c) – ймовірність класу c.

В YOLOv3 та всіх наступних – крос-ентропія [11]:

$$L_{cls} = - \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \sum_{c \in K} (\hat{p}_i(c) log(p_i(c)) + (1 - \hat{p}_i(c)) log(1 - p_i(c)))$$

### Загальна функція втрат

Фінальна функція втрат є зваженою сумою всіх трьох компонентів [6]:

$$L = \lambda_{loc} \cdot L_{loc} + \lambda_{obj} \cdot L_{obj} + \lambda_{cls} \cdot L_{cls}$$

$\lambda_{loc}, \lambda_{obj}, \lambda_{cls}$ – вагові коефіцієнти кожної частини, що дозволяють налаштувати баланс між окремими складовими втрат.

## 2.5 Версії YOLO

Починаючи від 2016 року і до сьогодні модель YOLO отримує нові версії, які вдосконалюють її точність, швидкість та можливості застосування. Станом на час написання цієї кваліфікаційної роботи існує більше 10 версій YOLO. Нижче наведено ключові версії та їхні особливості.

### YOLOv1

Перша версія YOLO була створена в 2016 році [6]. Вона реалізувала підхід одноетапного детектора, де класифікація та локалізація проводились за один прохід мережі. Модель працювала в реальному часі зі швидкістю 45 кадрів на секунду (FPS), але мала гіршу точність ніж двоетапні детектори: 57.9% mAP проти 70.4% mAP у Faster R-CNN із набором даних PASCAL VOC 2012.

### YOLOv3

Версія YOLOv3 стала важливим оновленням [11]. Модель отримала можливість робити багатокласову класифікацію та використовувати якорні рамки, які базувались на наборі даних COCO. Було досягнуто балансу між швидкістю та середньою точністю: 57.9% на датасеті COCO зі швидкістю 30 FPS, що було краще за тодішніх конкурентів, наприклад, SSD513 із 50.4% і 8 FPS.

### YOLOv5

Попри неофіційність, бо вона була реалізована сторонньою компанією Ultralytics, YOLOv5 стала надзвичайно популярною. Модель була повністю переписана на бібліотеці PyTorch та отримала модульну архітектуру, що спростило її модифікацію, адаптацію та тренування [14]. Також було реалізовано автоматичний підбір якорів та вже претренована модель могла бути використана в декількох розмірах: від найменшої YOLOv5n(nano) до найбільшої YOLOv5x(extra-large). YOLOv5m отримала 64.1% mAP зі швидкістю 120 кадрів на секунду із COCO.

### YOLOv8

Наступним важливим кроком стала модель YOLOv8 [13], яка повністю відмовилась від якорних рамок, що пришвидшило етап навчання. Результати YOLOv8m виявились також кращими: 66.8% mAP зі швидкістю 160 кадрів на секунду на COCO.

### YOLO11

Останньою стабільною версією стала YOLO11 [15], яка не є революційною, але є продовженням розвитку YOLOv8. Вона імплементує покращену архітектуру хребта та шиї, що дозволяє їй мати кращу точність. Метрики YOLO11m: 68.1% mAP при швидкості 200 кадрів на секунду із набором COCO.

| **Модель** | **Точність** | **Швидкість(FPS)** |
|---|---|---|
| YOLOv1 | 57.9%* | 45 |
| YOLOv3 | 57.9% | 30 |
| YOLOv5 | 64.1% | 120 |
| YOLOv8 | 66.8% | 160 |
| YOLO11 | 68.1% | 200 |

Табл. 3: Порівняння версій YOLO

* – на наборі даних PASCAL VOC

Для дослідження та експериментування із навчанням на частковій розмітці було обрано YOLO11. Ця модель є актуальною на момент написання кваліфікаційної роботи, має високу швидкість і точність та пропонує зручну платформу для модифікації.

# 3 Методи тренування на даних з частковою розміткою

Як зазначалось раніше, ефективне навчання більшості моделей детекції залежить від наявності набору із повністю розміченими даними, але в реальних умовах це не завжди може бути дотримано. Застосування традиційних методів в таких випадках може призвести до проблем катастрофічного забування(catastrophic forgetting) [16] та зміщення моделі в бік домінуючих класів [17]. Для подолання цих обмежень та мінімізування втрати ефективності повинні бути застосовані спеціальні підходи, які будуть спрямовані на використання часткової анотації. Нижче розглянуто деякі з них.

## 3.1 Псевдорозмітка

Одним з популярних способів роботи із частковою анотацією є застосування псевдорозмітки [18] – методу напівкерованого навчання, який автоматично генерує додаткові анотації із впевнених прогнозів вже навченої моделі. У зазначеній роботі [18] автор використовує цей підхід для збільшення навчальної вибірки для покращення результатів моделі в задачі класифікації зображень і отримує підвищення точності.

В цій роботі таке застосування дозволить збільшити кількість розмічених об'єктів у наборі даних, що зменшить залежність від ручної розмітки та потенційно покращить ефективність моделі. Головним недоліком підходу можуть стати хибні мітки у згенерованих даних, які негативно вплинуть на навчання.

## 3.2 Дистиляція знань

Наступним підходом є дистиляція знань [19]. Цей метод також використовує вже попередньо натреновану модель-вчителя, яку модель-студент буде намагатись імітувати на етапі навчання. Для процесу імітації вихідні прогнози вчителя зазвичай використовуються у порівнянні із прогнозами студента та їх розбіжність включається до загальної функції втрат студента окремим компонентом, який треба мінімізувати.

В дослідженні [16] таке застосування дозволяє новій моделі уникнути катастрофічного забування про відсутні в наборі даних класи, що може бути корисним при навчанні на частково розмічених даних. Окремим обмеженням методу є те, що прогрес у навчанні буде досить повільним та ефективність буде напряму залежати від якості моделі-вчителя.

## 3.3 Маскування

Ще одним варіантом є маскування – обмеження інформації, яку отримує модель. У цьому контексті буде виділено два методи:

1. **Маскування зображення**: в кваліфікаційній роботі запропоновано підхід, де модель, завдяки позбуванню від фона, отримує на вхід зображення із лише анатованими областями. Це дозволить відкинути хибне навчання на нерозмічених, але присутніх на фото об'єктах. Недоліком цього підходу може стати перенавчання на деяких прикладах та потенційна втрата здатності моделі відрізняти фон від об'єкта.

2. **Маскування втрат**: у роботі [20] запропоновано альтернативний спосіб, де навчання відбувається на повному зображенні, але при підрахунку функції втрат використовуються лише втрати від спрогнозованих об'єктів, що мали IoU$>$50% із рамкою справжньої розмітки. Цей підхід допоміг в усуненні помилкових штрафів за відсутні анотації та підвищив здатність моделі до навчання на неповних даних. Також було зазначено, що з'явилася проблема хибно позначених міток, коли нерозмічений об'єкт співпадав із розміченим та модель навчалась на цьому прикладі як на правильному.

## 3.4 Комбінація підходів

Найбільш перспективним підходом може виявитись комбінація описаних методів в єдину навчальну схему. Поєднання дозволить використовувати їх сильні сторони та компенсувати обмеження. Головним викликом стане складність пошуку оптимальної комбінації та ретельне налаштування її гіперпараметрів для уникнення прихованих конфліктів.

# 4 Побудова модифікованої процедури навчання

## 4.1 Опис проблеми

В реальних умовах збір повністю анотованих зображень є складним, повільним та затратним процесом. Розмітка одного фото в середньому займає від 3 до 5 хвилин, що для наборів із десятками тисяч зображень відповідає тисячам годин роботи [21]. В результаті, більшість датасетів можуть містити велику кількість частково розмічених зображень, в яких деякі об'єкти не будуть проанотовані, а деякі класи будуть мати недостатньо прикладів для навчання.

Для YOLO і більшості сучасних моделей детекції, що спираються на повністю розмічені дані, це може створити додаткові проблеми:

- Модель буде вважати всі нерозмічені об'єкти фоном і навіть при правильному прогнозі буде отримано хибний сигнал про відсутність.
- При дисбалансі класів буде відбуватись зміщення в сторону частих класів, що призведе до погіршення метрик рідкісних класів .

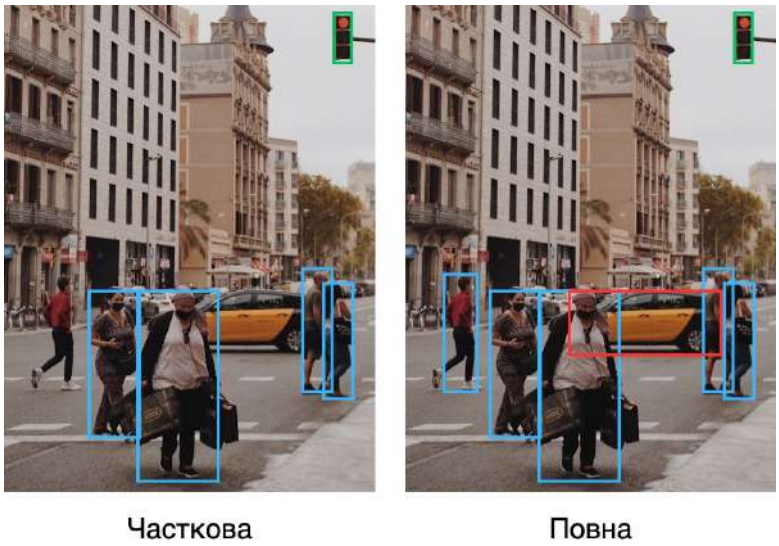



Рис. 5: Приклад часткової та повної розмітки

Нехай $D_{full} = \{(I_n, A_n)\}$ – повністю розмічений датасет, де $I_n$ – зображення, а $A_n$ – множина анотацій до нього, $A_n \subseteq \{(c, b) \mid c \in \{c_0, \ldots, c_K\}, b \in \mathbb{R}^4\}$.

Задачею є пошук підходу до навчання моделі детекції $f$ на наборі із частково розміченими зображеннями $D_{partial} = \{(I_n, A'_n)\}$, де $A'_n \subseteq A_n$, та $D_{full}$ такого, що максимізує метрики mAP моделі $f$ на тестовій вибірці $D_{test}$. Для здійснення цього було запропоновано та перевірено модифіковані методи тренування.

## 4.2 Підготовка даних

Проведення наближеного до реальної ситуації дослідження потребує використання великого набору даних із багатьма різними класами, бажано більше 5000 зображень з повною розміткою. В початковій моделі, яка буде використовуватись для порівняння із запропонованими підходами, треба використати перероблений датасет, в якому треба обрати один клас із набору (наприклад, «potted plant») та видалити значну частину його зображень із анотаціями. Це дозволить створити умову низькорепрезентованого класу, який потребуватиме в дотренуванні та виступить гарним індикатором здатності моделі до навчання.

Етап дотренування буде базуватись на частково анотованому датасеті, який в цьому випадку буде містити лише розмітку обраного класу та відповідні зображення. Підготовлені дані дозволять провести експеримент, в якому модель буде навчатись на неповній розмітці, що і є однією із задач цієї роботи.

## 4.3 Адаптація функції втрат

Для подолання проблем із процесом навчання на частковій розмітці можна виділити два підходи, які будуть напряму модифікувати структуру навчання та функцію втрат: маскування втрати нерозмічених класів та дистиляція знань.

### Маскування втрати нерозмічених класів

Ідея цього методу – уникнення хибного штрафування за передбачення нерозмічених об'єктів. Спосіб базується на роботі [20], але має модифіковану реалізацію, бо на етапі створення пакету (batch) для навчання кожне зображення отримує вектор $M \in \{0, 1\}^K$, де $K$ – кількість класів, $M_c$ – окрема

змінна вектора, що зазначає присутність або відсутність класу $c$ у розмітці зображення. Під час обчислення функції втрат цей вектор застосовується для маскування кожного компонента втрати, де передбачення робиться для класу, який відсутній в анотації.

$$L_{loc} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \cdot M_{c_{ij}} \cdot L_{CIoU}(\hat{B}_{ij}, B_{ij})$$

$$L_{obj} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} -\mathbb{1}_{ij}^{obj} \cdot M_{c_{ij}} \cdot [\hat{C}_i log C_i + (1 - \hat{C}_i) log(1 - C_i)]$$
$$- \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} \cdot M_{c_{ij}} \cdot [(\hat{C}_i log C_i + (1 - \hat{C}_i) log(1 - C_i)]$$

$$L_{cls} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \sum_{c \in K} M_c \cdot [\hat{p}_i(c) log(p_i(c)) + (1 - \hat{p}_i(c)) log(1 - p_i(c))]$$

$c_{ij}$ – клас об'єкта в j-й рамці i-ї комірки.

Таким чином, цей підхід виключає вплив відсутніх анотацій на потенційно правильні прогнози моделі. Це повинно зменшити ефект відсутньої розмітки окремих класів на навчання, що може допомогти в усуненні перетренування на рідкісному класі.

### Дистиляція знань

Цей метод використовує модель-вчителя, яка передасть свої знання моделі-студенту та дозволить йому не забувати присутні нерозмічені об'єкти на етапі донавчання. Застосована реалізація способу із дослідження [16]. Для цього потрібно мати вчителя і на цю роль ідеально підійде попередньо навчена модель із одним низькорепрезентованим класом, втрату від якого потрібно прибрати, щоб студент міг швидше навчитись розпізнавати цей об'єкт. Механізм передачі знань буде робитись через окремі компоненти втрат для цього підходу:

$$L_{KD\_cls} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} D_{KL}(\hat{P}_{ij}^{teacher} || \hat{P}_{ij}^{student})$$

де $D_{KL}$ – розходження Кульбака-Лейблера, $\hat{P}_{ij}$ – передбачені ймовірності(без рідкісного класу) для рамки j комірки i.

$$L_{KD\_loc} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} (\hat{B}_{ij}^{teacher} - \hat{B}_{ij}^{student})^2$$

де $\hat{B}_{ij}$ – передбачені координати рамок.

$$L_{new} = L + \lambda_{KD\_cls} \cdot L_{KD\_cls} + \lambda_{KD\_loc} \cdot L_{KD\_loc}$$

Базуючись на результатах [16], це дозволить моделі навчитись на нових даних та допоможе уникнути проблеми катастрофічного забування, використовуючи знання старої моделі.

## 4.4 Альтернативні способи навчання моделі

На жаль, не всі моделі мають відкритий вихідний код або зрозумілу документацію, що може унеможливити їх пряму модифікацію. Через це з'являється потреба в методах, які зможуть адаптувати дані для навчання детектора. В цьому розділі запропоновано три таких підходи.

### Комбінація повних і частково розмічених даних

В цьому підході запропоновано змінити дані для дотренування, об'єднавши обидва датасети повної та часткової розмітки. Таким чином, анотація все ще буде залишатись частковою, але матиме набагато більше прикладів, що дозволить моделі не перенавчитись та потенційно уникнути катастрофічного забування.

$$D = D_{full} \cup D_{partial}$$

де $D$ – набір даних, який складається з анотацій та відповідних зображень.

Найбільшу проблему для цього методу може становити важливість наявності великої частки повністю розмічених зображень, що не завжди досяжно в реальних умовах.

### Псевдорозмітка неповних анотацій

Як і в [18], імплементація псевдорозмітки буде використовувати початкову модель, яка згенерує власні анотації зображенням, для яких вже присутні неповні дані. Обов'язковою умовою також буде те, що прогнози моделі не будуть робитись для вже існуючих у розмітці класів та повинні мати значення впевненості вище обраного порогу $\theta = 0.6$. Отримана нова розмітка буде додана до часткової.

Нехай $f$ – натренована модель детекції для класів $c_0, \ldots, c_K$, а $\{I_n\}$ – набір зображень, де розмічено лише клас $c_0$ табличною функцією $T$. Тоді псевдорозмітка – результат наступної функції:

$$T_{pseudo}(I_n) = (f_\theta(I_n) \setminus \{(c, b, s) \mid c = c_0\}) \cup T(I_n)$$

де $f_\theta(I_n) \subseteq \{(c, b, s) \mid s \geq \theta\}$ – набір прогнозів моделі із впевненістю більше порогу, $(c, b, s)$ – прогнози моделі вигляду «клас-рамка-впевненість».

Як результат, новий детектор зможе навчатись на псевдорозмічених даних, які теоретично будуть нагадувати повні. Очевидною перевагою цього підходу є те, що він взагалі не потребує додаткових даних, лише натреновану модель.

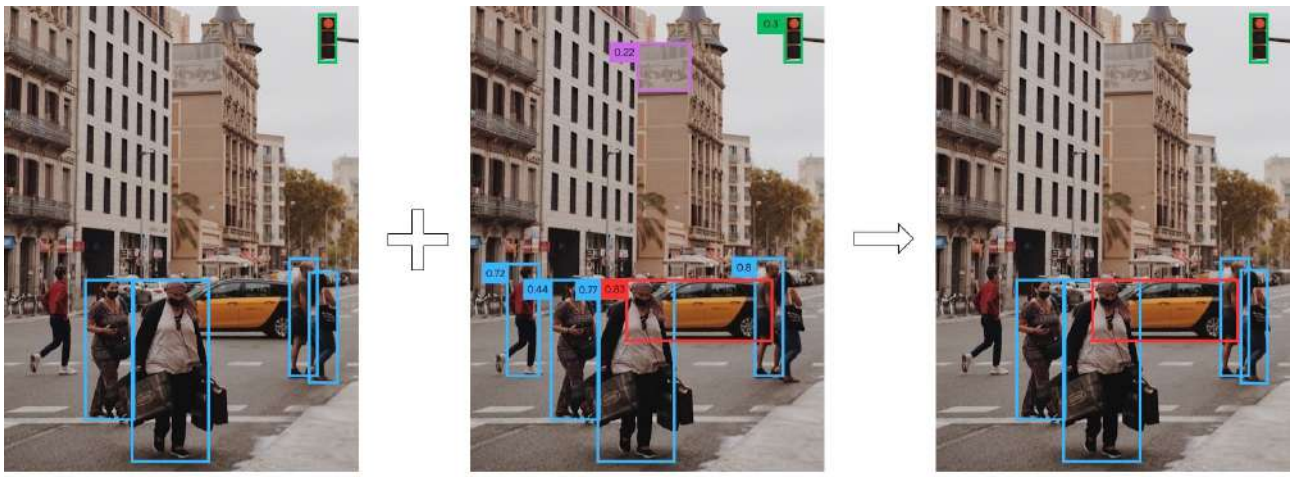



Рис. 6: Приклад псевдорозмітки

### Маскування фону зображення

Маскування фону – ще один із запропонованих способів, який використовує тільки модифікацію даних для зменшення негативного впливу неповних анотацій. Для реалізації потрібно переробити зображення так, щоб на них видимими залишились тільки розмічені об'єкти. Це можна зробити двома шляхами: затемненням фону або його розмиттям.

Нехай $\mathbb{1}_A(x, y)$ – індикаторна функція існування анотації для координати $(x, y)$, $\{I_n\}$ – набір зображень, $\{A_n\}$ – набір анотацій. Тоді набір із замаскованим фоном:

$$I_n^{masked} = I_n(x, y) \cdot \mathbb{1}_{A_n}(x, y)$$

де $I_n(x, y)$ – значення пікселя координати $(x, y)$.

Запропонований метод дозволить моделі зосередитись на розмічених зонах, що може посприяти уникненню катастрофічного забування.

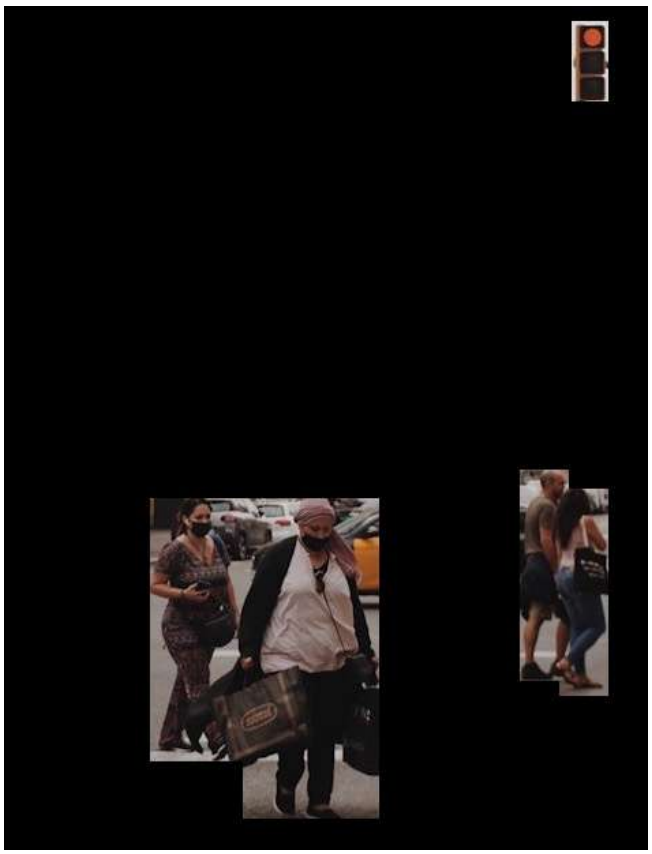


Рис. 7: Приклад маскування фону частково розміченого зображення

# 5 Експериментування

Перевірку та оцінку ефективності запропонованих підходів було проведено завдяки серії експериментів із використанням моделі **YOLO11 v8.3.91** та датасету **PASCAL VOC 2012**. Обрання моделі цієї версії зумовлено тим, що вона була останньою під час проведення всіх досліджень. Обраний набір даних є одним із найпоширеніших у сфері комп'ютерного зору, він має 11540 повністю анотованих зображень із 20 різними класами для навчання та 4952 зображення для тестування. Кожен розмічений об'єкт може належати тільки до однієї з категорій.

На етапі підготовки даних тренувальна вибірка була поділена на 2 частини:

- **Базова модель**: для початкового навчання було використано лише 20% анотацій та зображень із класом «potted plant». Решта навчальної вибірки залишилась незмінною.

- **Додаткове тренування**: застосовано всі зображення із об'єктами «potted plant» та їх розміткою. Всі інші анотації, які не є обраним класом, видалено. Також прибрано зображення, де залишився тільки задній фон.

Дослідження проводилось у середовищі Google Colab із використанням графічного процесора NVIDIA Tesla T4. Реалізація здійснювалась на мові програмування – Python із бібліотекою Ultralytics.

Модель мала наступні гіперпараметри: розмір зображень (imgsz) – 640 × 640, кількість епох тренування (epochs) – 10, оптимізатор (optimizer) – auto (AdamW(lr=0.000417, momentum=0.9)), всі інші параметри – стандартні.

## 5.1 Результати експериментів

Оцінка результатів проводилась за допомогою метрик **mAP50**, що показує середню точність на рівні IoU в 50%, та **mAP50-95**, що демонструє середню точність на різних масштабах IoU. Для кожного підходу наведено результати цих метрик для всіх класів та для обраного класу «potted plant». Окремо надано графік F1 міри на різних порогах впевненості для базової моделі та

найефективнішого підходу. Графіки F1 міри для всіх експериментів надано в додатку А.

| **Підхід** | **Всі класи** | | **Обраний клас** | |
|---|---|---|---|---|
| | mAP50 | mAP50–95 | mAP50 | mAP50–95 |
| Базова модель | 0.785 | 0.599 | 0.291 | 0.136 |
| Маскування класів | 0.275 | 0.166 | 0.430 | 0.268 |
| Дистиляція знань | 0.644 | 0.453 | 0.298 | 0.177 |
| Повна/часткова розмітка | 0.720 | 0.524 | 0.439 | 0.271 |
| Псевдорозмітка | 0.696 | 0.507 | 0.438 | 0.266 |
| Маскування фону | 0.156 | 0.090 | 0.028 | 0.012 |
| Дистиляція та м. класів | 0.737 | 0.553 | 0.300 | 0.139 |
| Дистиляція та п/ч розмітка | 0.766 | 0.579 | 0.294 | 0.145 |
| М. класів та п/ч розмітка | 0.667 | 0.487 | 0.426 | 0.229 |
| М. фону та п/ч розмітка | 0.367 | 0.199 | 0.114 | 0.044 |

Табл. 4: Порівняння результатів експериментів

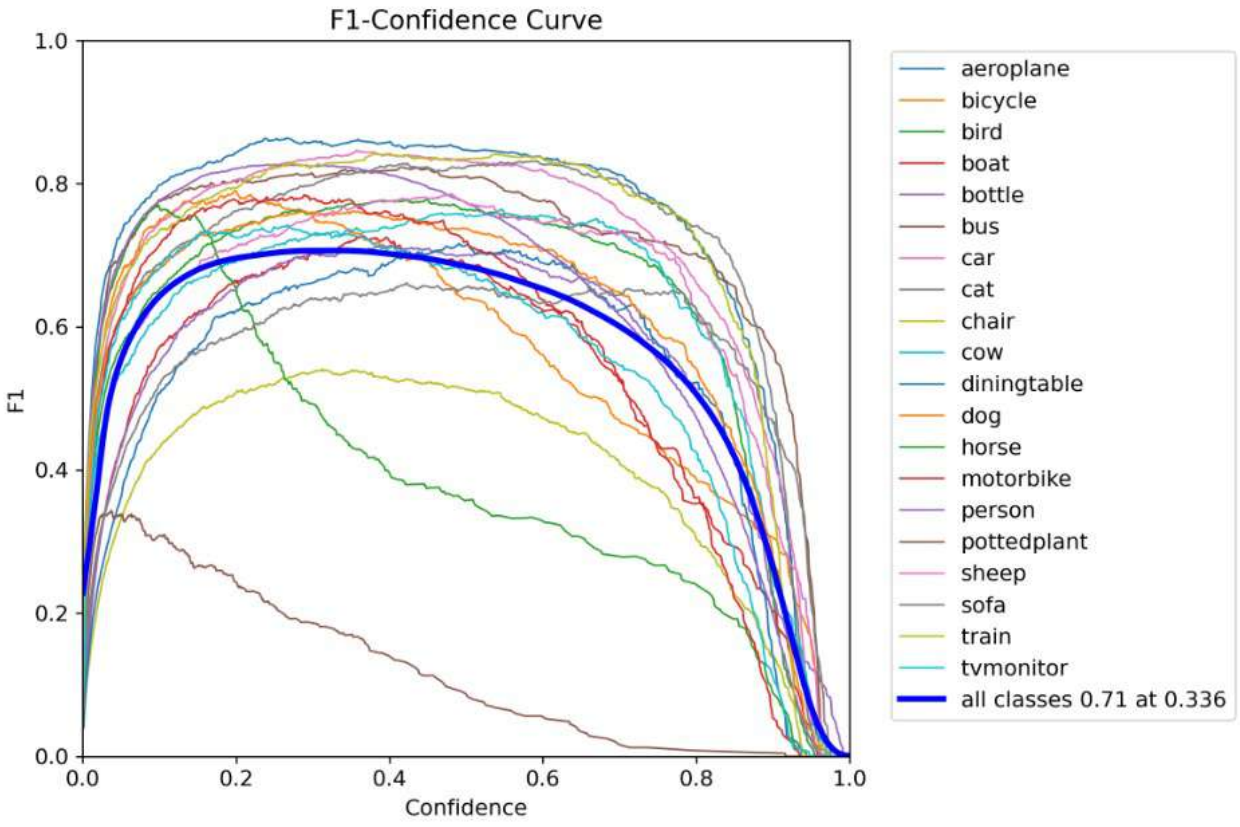



Рис. 8: Крива F1 для базової моделі

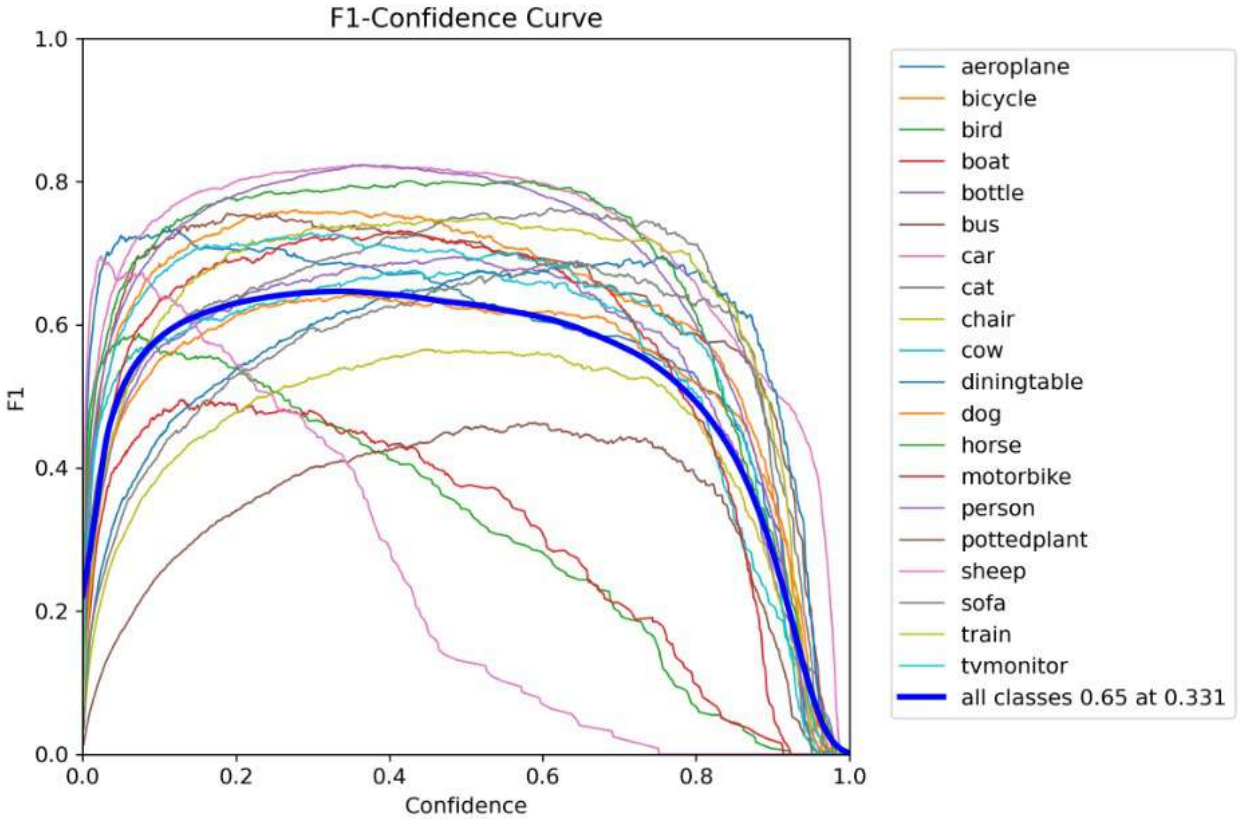



Рис. 9: Крива F1 для підходу комбінації повної та часткової розмітки

## 5.2 Аналіз результатів

- **Базова модель**: продемонстровано високу загальну точність, але низькі результати для обраного класу.
- **Маскування втрати нерозмічених класів**: суттєво погіршена mAP для всіх класів, але значно покращені метрики класу «potted plant». Покращення також супроводжено падінням значення влучності, що свідчить про перенавчання та зміщення моделі в сторону рідкісного класу.
- **Дистиляція знань**: видно погіршення в загальній точності та невелике покращення розпізнавання обраного класу. Покращення зумовлено погіршенням повноти, що може казати про малу кількість загальних передбачень «potted plant».
- **Комбінація повної і часткової розмітки**: збережено адекватну загальну точність та значно покращено результати «potted plant».

- **Псевдорозмітка даних**: mAP всіх класів трохи знизилось, точність обраного класу суттєво збільшена. Результати схожі на навчання на комбінації часткової та повної розмітки.

- **Маскування фону**: видно колосальне погіршення всіх метрик.

- **Комбінація дистиляції знань та маскування втрати нерозмічених класів**: підхід отримав зменшену точність на всіх класах та невелике покращення результатів обраного класу.

- **Комбінація дистиляції знань та повністю/частково розмічених даних**: комбінація продемонструвала одну з найкращих метрик загальної точності та невелике збільшення метрики для «potted plant».

- **Комбінація маскування втрати нерозмічених класів та повністю/частково розмічених даних**: отримано незначне погіршення загальних метрик, але гарне покращення точності обраного класу.

- **Комбінація маскування фону та повністю/частково розмічених даних**: порівняно із минулою спробою маскування фону, отримано кращі, але дуже незадовільні результати.

## 5.3 Підсумок

Проведені експерименти підтвердили, що навчання на частково розмічених даних можливе і є потенційно ефективним.

Найефективнішими підходами виявились псевдорозмітка та використання комбінації повністю і частково розмічених класів. Вони змогли забезпечити адекватні загальні результати моделі та значно покращити результати класу «potted plant».

Перспективними підходами можна вважати дистиляцію знань та комбінацію маскування втрат для нерозмічених класів із повною/частковою розміткою, що допомогли зберегти загальні результати на високому рівні та трохи підвищити метрики для обраного класу. Проте ці методи мають дуже повільну швидкість навчання, тому вони потребують подальшого налаштування для їх ще більшого вдосконалення.

Найменш ефективним методом можна вважати маскування фону, що в обох тестах негативно вплинуло на обидві метрики і тому воно не рекомендується для подібних задач.

Окрім оцінки ефективності, проаналізовано також складність впровадження кожного з підходів, що дозволяє оцінити їх практичну доцільність.

Найпростішими є методи, які вимагають базову модифікацію набору даних. Такими є об'єднання повної/часткової розмітки, яка потребувала комбінації вже створених вибірок, та маскування фону, де перетворення зображень також не викликало проблем.

Методи середньої складності мають додаткові кроки для підготовки датасету. Наприклад, псевдорозмітка потребувала використання навченої моделі для створення додаткових анотацій, які потім акуратно комбінувались із присутніми даними.

Найбільш вимогливими виявились підходи, які потребували прямого втручання в структуру навчання – маскування втрати нерозмічених класів та дистиляція знань. Обидва методи вимагали розуміння логіки алгоритму, ретельного дослідження функції втрат моделі та точної модифікації коду.

# Висновки

У цій кваліфікаційній роботі було розглянуто підходи до модифікації процесу тренування моделі детекції об'єктів YOLO для роботи із частково розміченими даними. Мета дослідження полягала в експериментальній спробі створення такого сценарію навчання алгоритму, який зможе подолати обмеження при тренуванні на частково розмічених зображеннях без значної втрати якості детекції.

Для досягнення мети було запропоновано та реалізовано наступні підходи:

- Модифікація функції втрат із ігноруванням нерозмічених класів.
- Дистиляція знань.
- Використання комбінації наборів повністю і частково розмічених даних.
- Псевдорозмітка.
- Маскування фону зображення.

Результати експериментів дозволили зробити наступні висновки:

- Тренування моделі YOLO на частково розмічених даних без значної втрати ефективності можливе.
- Модифікація функції втрат дозволяє частково зберегти загальну точність моделі, але для досягнення високоякісного навчання лише цього недостатньо.
- Отримання найкращих результатів вимагає зменшення частки частково розмічених даних у всьому датасеті або їх найбільш наближену конвертацію до повної розмітки.

Отже, проведені дослідження підтверджують доцільність застосування частково розмічених даних для навчання моделей детекції об'єктів YOLO із використанням запропонованих підходів. Ці результати можуть бути корисними в практичному застосуванні, наприклад, коли повна розмітка є неможливою або занадто витратною.

Також, ці дослідження можуть слугувати відправною точкою для подальших робіт, оскільки жоден із запропонованих методів не виявився ідеальним та мав свої недоліки. Рекомендаціями для подальших досліджень будуть:

- **Комбінація та налаштування запропонованих підходів**: було перевірено комбінації тільки невеликої частини методів, що дозволяє проводити додаткові експерименти із пошуком найоптимальнішого поєднання. Окрім цього, більша частина гіперпараметрів моделі та деяких підходів окремо не налаштовуваласть, що також може бути ключем до пошуку найбільш ефективного підходу.

- **Проведення експерименту із іншими моделями**: всі експерименти проводились із архітектурою YOLO11, але вона не є панацеєю. В майбутніх роботах бажано сфокусуватись на проведені досліджень із двоетапними детекторами або альтернативними одноетапними моделями.

- **Модифікація процесу навчання алгоритму**: оскільки модифікація самої моделі проходила здебільшого для функції втрат, сама стратегія навчання майже не зачіпалася, але вона може слугувати одним із потенційних шляхів для знаходження результативних підходів до навчання на частковій розмітці.

# Список літератури


1. Viola P., Jones M. Rapid object detection using a boosted cascade of simple features // Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001). – 2001. – Vol. 1. – P. 511-518.

2. Dalal N., Triggs B. Histograms of oriented gradients for human detection // Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005). – 2005. – Vol. 1. – P. 886-893.

3. Girshick R., Donahue J., Darrell T., Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2014. – P. 580-587.

4. Girshick R. Fast R-CNN // Proceedings of the IEEE International Conference on Computer Vision (ICCV). – 2015. – P. 1440-1448.

5. Ren S., He K., Girshick R., Sun J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks // Advances in Neural Information Processing Systems (NeurIPS). – 2015. – Vol. 28.

6. Redmon J.. Divvala S., Girshick R., Farhadi A. You Only Look Once: Unified, Real-Time Object Detection // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2016. – P. 779-788.

7. Liu W., Anguelov D., Erhan D., Szegedy C., Reed S., Fu C.Y., Berg A.C. SSD: Single Shot MultiBox Detector // Proceedings of the European Conference on Computer Vision (ECCV). – 2016. – P. 21-37.

8. Carion N., Massa F., Synnaeve G., Usunier N., Kirillov A., Zagoruyko S. End-to-End Object Detection with Transformers // arXiv preprint arXiv:2005.12872. – 2020. – URL: https://arxiv.org/abs/2005.12872

9. Long X., Deng K., Wang G., Zhang Y., Dang Q., Gao Y., Shen H., Ren J., Han S., Ding E., Wen S. PP-YOLO: An Effective and Efficient Implementati-

on of Object Detector // arXiv preprint arXiv:2007.12099. – 2020. – URL: https://arxiv.org/abs/2007.12099

10. Bochkovskiy A., Wang C.-Y., Liao H.-Y. M. YOLOv4: Optimal Speed and Accuracy of Object Detection // arXiv preprint arXiv:2004.10934. – 2020. – URL: https://arxiv.org/abs/2004.10934

11. Redmon J., Farhadi A. YOLOv3: An Incremental Improvement // arXiv preprint arXiv:1804.02767. – 2018. – URL: https://arxiv.org/abs/1804.02767

12. Redmon J., Farhadi A. YOLO9000: Better, Faster, Stronger // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2017. – P. 7263-7271.

13. Ultralytics. YOLOv8: Official Documentation. – 2023. – URL: https://docs.ultralytics.com/models/yolov8/

14. Ultralytics. YOLOv5: Open Source Object Detection Model. – 2020. – URL: https://github.com/ultralytics/yolov5

15. Ultralytics. YOLO11: Official Documentation. – 2024. – URL: https://docs.ultralytics.com/models/yolo11/

16. Li Z., Hoiem D. Learning without Forgetting // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 2017. – Vol. 40, No. 12. – P. 2935-2947.

17. Lin T.-Y., Goyal P., Girshick R., He K., Dollár P. Focal Loss for Dense Object Detection // Proceedings of the IEEE International Conference on Computer Vision (ICCV). – 2017. – P. 2980-2988.

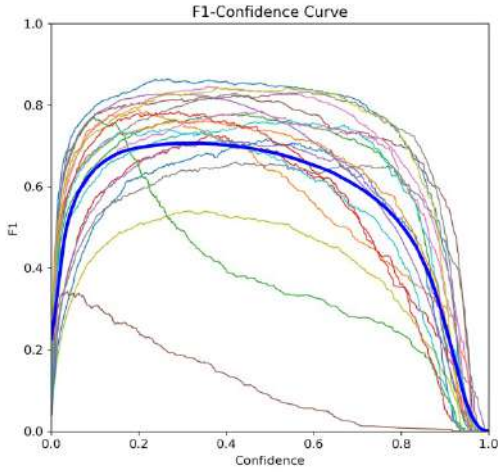
18. Lee D.-H. Pseudo-Label: The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks // ICML 2013 Workshop: Challenges in Representation Learning. – 2013.

19. Hinton G., Vinyals O., Dean J. Distilling the Knowledge in a Neural Network // arXiv preprint arXiv:1503.02531. – 2015. – URL: https://arxiv.org/abs/1503.02531

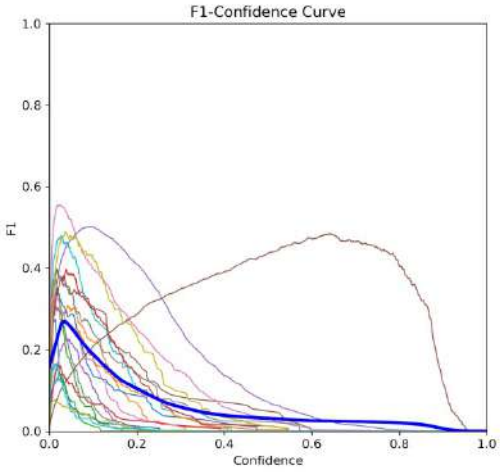
20. Wu Z., Bodla N., Singh B., Najibi M., Chellappa R., Davis L. S. Soft Sampling for Robust Object Detection // arXiv preprint arXiv:1806.06986. – 2018. – URL: https://arxiv.org/abs/1806.06986

21. Lin T.-Y., Maire M., Belongie S., Hays J., Perona P., Ramanan D., Dollár P., Zitnick C. L. Microsoft COCO: Common Objects in Context // Lecture Notes in Computer Science. – 2014. – Vol. 8693. – P. 740-755.

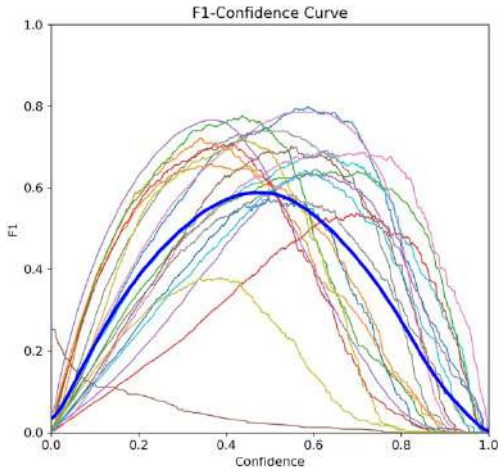
# Додаток А



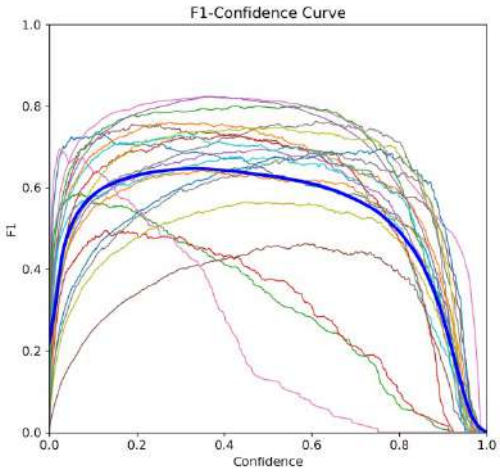

Базова



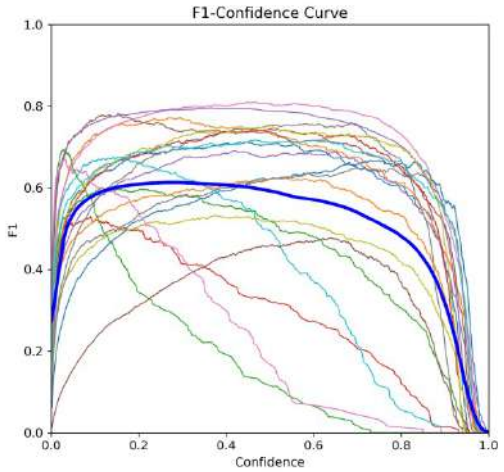

Маскування класів



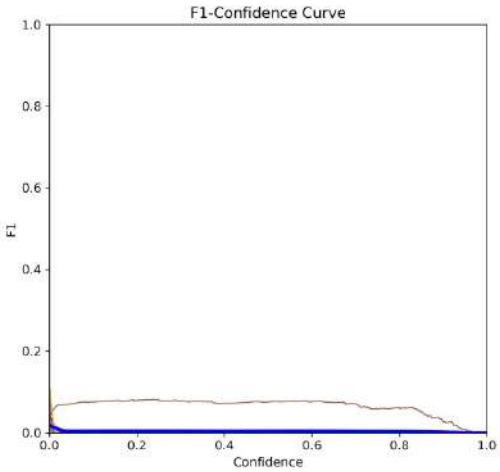

Дистиляція



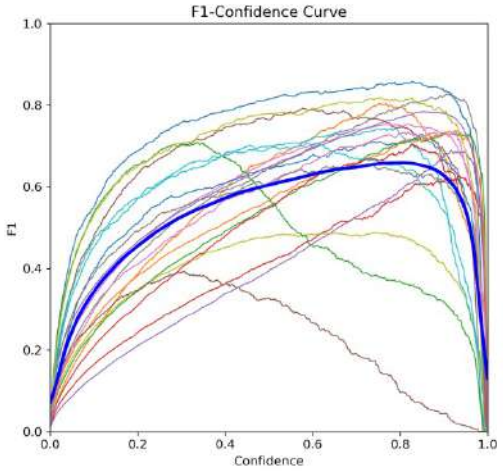

Повна/часткова розмітка



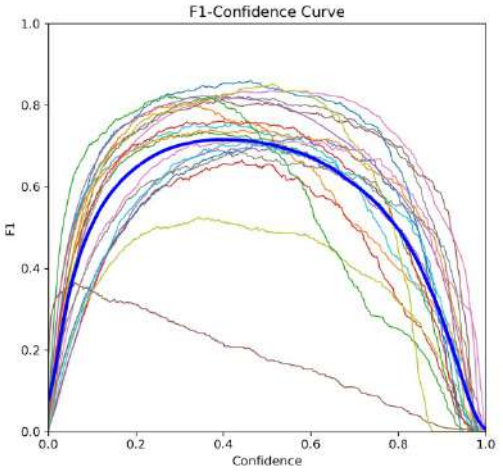

Псевдорозмітка



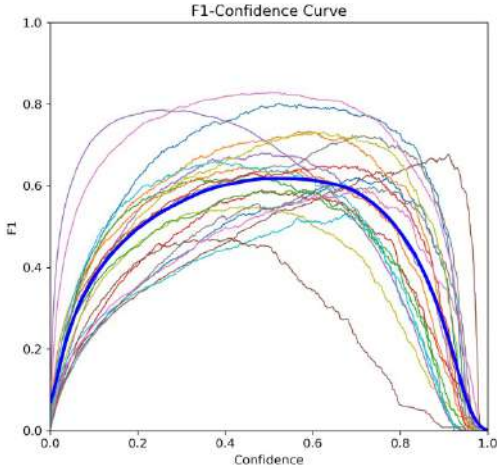

Маскування фону



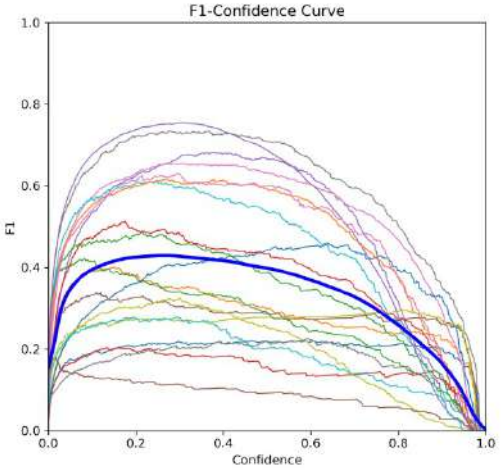

Дистиляція +
Маскування класів




Дистиляція +
Повна/часткова розмітка




Маскування класів +
Повна/часткова розмітка




Маскування фону +
Повна/часткова розмітка