Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)
факультет інформаційних технологій
(факультет)
Кафедра інформаційних технологій та комп'ютерної інженерії
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня магістра
(бакалавра, спеціаліста, магістра)

Студента Дьякова Сергія Петровича
(ПІБ)
академічної групи 126м-22-1
(шифр)
спеціальності 126 «Інформаційні системи та технології»
(код і назва спеціальності)
за освітньо-професійною програмою

«Інформаційні системи та технології»
(офіційна назва)
на тему Розробка інформаційної технології розпізнавання об'єктів у відеоряді
(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	проф. Коротенко Г.М.			
розділів:				

Рецензент	доц. Ширін А.Л.			

Нормоконтролер	проф. Коротенко Г.М..			

Дніпро
2023

ЗАТВЕРДЖЕНО:
завідувач кафедри

___інформаційних технологій___
___та комп'ютерної інженерії____
(повна назва)

_____ __Гнатушенко В.В.__
(підпис) (прізвище, ініціали)

«_____»_____ 2023 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня ____магістр_____
(бакалавра, спеціаліста, магістра)

студенту _____Дьякову С.П._____ **академічної групи** ____126м-22-1____
(прізвище та ініціали) (шифр)

спеціальності ___126 « Інформаційні системи та технології »_____

за освітньою-професійною програмою _____

«Інформаційні системи та технології»_____
на тему Розробка інформаційної технології розпізнавання об'єктів у відеоряді_____
_____

затверджену наказом ректора НТУ «Дніпровська політехніка» від 09.10.2023 р. № 1227-с

Розділ	Зміст	Термін виконання
Розділ 1	Аналіз теми та постановка задачі з розробки інформаційної технології розпізнавання об'єктів у відеоряді.	1.10.2023 – 18.10.2023
Розділ 2	Проектування та розробка інформаційної технології розпізнавання об'єктів у відеоряді.	19.10.2023 – 15.11.2023
Розділ 3	Тестування створеного продукта, розгортання та підтримка. Оформлення кваліфікаційної роботи.	16.11.2023 – 10.12.2023

Завдання видано _____ _____Коротенко Г.М._____
(підпис керівника) (прізвище, ініціали)

Дата видачі _____1.10.2023 р._____

Дата подання до екзаменаційної комісії ____21.12.2023 р.____

Прийнято до виконання _____ _____Дьяков С.П._____
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 95 стор., 17 рис., 1 табл., 4 додатка, 97 джерел.

Об'єкт дослідження: інформаційна технологія розпізнавання об’єктів в відеоряді.

Предмет дослідження: процес розпізнавання об’єктів у відеоряді.

Мета магістерської роботи: обґрунтування архітектури інформаційної технології, що забезпечує розпізнавання об’єктів у відеоряді із оптимальною точністю та швидкодією.

У вступі подано стан проблеми та виконана постановка задачі дослідження.

В першому розділі наведені основні відомості про засоби та технології виявлення об’єктів, здійснено аналіз сучасних алгоритмів розпізнавання об’єктів.

У другому розділі наведена проектна складова вирішення завдання, обґрунтувано вибір інструментів програмної реалізації технології.

У третьому розділі виконано проектування та реалізацію інформаційної технологію розпізнавання об’єктів в відеоряді, та надані результати її експериментального дослідження.

Наукова новизна отриманих результатів визначається тим, що вперше обґрунтовано архітектуру інформаційної технології розпізнавання об’єктів у відеоряді, що підтримує адаптацию класифікатора до масштабування зображення.

Практична цінність результатів полягає в тому, що запропонована в роботі інформаційна технологія дозволяє виконувати розпізнавання об’єктів в відеоряді в режимі реального часу.

РОЗПІЗНАВАННЯ ОБ’ЄКТІВ, КЛАСИФІКАЦІЯ, МАСШТАБУВАННЯ, НЕЙРОННІ МЕРЕЖІ, CNN, SSD, ANCHOR BOX, BOUNDING BOX, VGG16.

ABSTRACT

Explanatory note: 95 pages, 17 figures, 1 table, 4 applications, 97 sources.

Object of research: information technology for object recognition in a video sequence.

Subject of research: the process of recognizing objects in a video sequence.

Purpose of Master's thesis: substantiation of information technology architecture, which ensures object recognition in a video sequence with optimal accuracy and speed.

In the introduction the status of the problem and the formulation of the research task are presented.

In the first section basic information about the means and technologies of object detection is given, and an analysis of modern object recognition algorithms is carried out.

In the second section the design component of problem solution is described, the choice of tools for software implementation of the technology is justified.

In the third section the design and implementation of the information technology for object recognition in the video sequence is performed, and the results of its experimental research are presented.

Originality of research is associated with first substantiation of the information technology architecture for object recognition in the video series, which supports the adaptation of the classifier to the image scaling.

Practical value of the results is that the information technology offered in the work allows recognition of objects in a video sequence in real time.

OBJECT RECOGNITION, CLASSIFICATION, SCALING, NEURAL NETWORKS, CNN, SSD, ANCHOR BOX, BOUNDING BOX, VGG16.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

BB - Bounding Box

CE - Cross-Entropy Loss

CNN - Convolutional Neural Network

DL - Deep Learning

FL - Focal Loss

FPN - Feature Pyramid Network

NMS - Non-Maximum Suppression

OD - Object Detection

RCNN - Region-Based Convolutional Neural Network

ReLU - Rectified Linear Unit Activation Function

RoI - Region of Interest

RPN - Region Proposal Network

SPP - Spatial Pyramid Pooling

SSD - Single Shot Multibox Detector

SVM - Support Vector Machine

YOLO - You Only Look Once

ВСТУП

Актуальність дослідження. Відстеження та виявлення об’єктів є одними з найважливіших завдань комп’ютерного зору, які мають численні застосування у таких сферах, як автономне відстеження транспортних засобів, роботизація, моніторинг руху. За останні роки було проведено велику кількість досліджень за даною тематикою. Однак, наявність таких невирішених задач, як виявлення швидкого руху, розрізнення варіацій освітлення, оклюзії, тощо, дають поштовх до продовження подальших досліджень в цій галузі.

Завдяки постійному вдосконаленню потужного обчислювального обладнання швидко розвивається технологія виявлення об’єктів на основі глибокого навчання. Також стає дедалі гострішою потреба у високоточних системах реального часу. Оскільки досягнення високої точності та ефективності детекторів є кінцевою метою цієї задачі, було розроблено низку напрямків, таких як побудова нової архітектури, вилучення багатих функцій, використання кращих репрезентацій, покращення швидкості обробки, навчання з нуля, методи без прив’язки, вирішення складних проблем зі сценою (малі об’єкти, закриті об’єкти), поєднання одноступеневих і двоступеневих детекторів для отримання кращих результатів, удосконалення методу постобробки NMS, вирішення проблеми дисбалансу мінус-позитив, підвищення точності локалізації та класифікації. Зі збільшенням потужності детекторів об’єктів у сферах безпеки, військової сфери, транспорту, медицини та побуту застосування технологій виявлення об’єктів поступово розширюється. Крім того, в області розпізнавання виникають нові відгалуження. Незважаючи на те, що новітні досягнення в цій галузі є ефективними, залишається ще багато можливостей для подальшого розвитку.

Об'єкт дослідження – інформаційна технологія розпізнавання об’єктів в відеоряді.

Предмет дослідження – процес розпізнавання об’єктів у відеоряді.

Мета й завдання дослідження. Мета роботи полягає у обґрунтуванні архітектури інформаційної технології, що забезпечує розпізнавання об’єктів у відеоряді із оптимальною точністю та швидкодією. Відповідно до мети й предмета дослідження у кваліфікаційній роботі необхідно вирішити наступні завдання:

- дослідити принципи, технології та алгоритми розпізнавання об’єктів в відеоряді;
- обґрунтувати архітектури інформаційної технології розпізнавання об’єктів в відеоряді;
- розробити програмне забезпечення інформаційної технології розпізнавання об’єктів в відеоряді на базі обраної архітектури.

Наукова новизна результатів кваліфікаційної роботи визначається тим, що вперше обґрунтовано архітектуру інформаційної технології розпізнавання об’єктів у відеоряді, що підтримує адаптацию класифікатора до масштабування зображення.

Практична цінність результатів полягає у тому, що запропонована в роботі інформаційна технологія дозволяє виконувати розпізнавання об’єктів в відеоряді в режимі реального часу.

Робота складається з трьох розділів.

Перший розділ присвячено аналізу тему дослідження та постановці задачі. Наведено огляд засобів та технологій виявлення об’єктів і коротко розглянуті сучасні алгоритми розпізнавання об’єктів.

В другому розділі наведено проектну складову вирішення завдання. Розглянуто деякі відомі моделі виявлення об’єктів в режимі реального часу, а також оптимізований алгоритм одиночного мультирамкового детектора. Обґрунтовано вибір інструментів програмної реалізації інформаційної технології.

Третій розділ присвячено розробці інформаційної технології розпізнавання об’єктів в відеоряді. Виконано реалізацію інформаційної технології.

Створено програмне забезпечення, яке призначено для розпізнавання об’єктів у відеоряді в режимі реального часу.

РОЗДІЛ 1
АНАЛІЗ СТАНУ ОБЛАСТІ РІШЕННЯ ЗАДАЧІ

1.1. Алгоритми виявлення об’єктів

Виявлення об’єктів — це важливе завдання комп’ютерного зору, яке має справу з виявленням екземплярів візуальних об’єктів певного класу (таких як люди, тварини чи автомобілі) у цифрових зображеннях. Метою виявлення об’єктів є розробка обчислювальних моделей і методів, які надають одну з найосновніших частин знань, необхідних для програм комп’ютерного зору – місцезнаходження об’єктів. Двома найважливішими показниками для виявлення об’єктів є точність (включаючи точність класифікації та точність локалізації) і швидкість [1].

Виявлення об’єктів служить основою для багатьох інших задач комп’ютерного зору, таких як сегментація екземплярів [2-3], створення підписів до зображень [4-5], відстеження об’єктів [6] тощо. В останні роки швидкий розвиток методів глибокого навчання значно сприяв прогресу виявлення об’єктів, що призвело до видатних проривів і привернуло до нього безпрецедентну увагу дослідників. Виявлення об’єктів зараз широко використовується в багатьох застосунках, таких як автономне водіння, роботизоване бачення, відеоспостереження тощо.

Оскільки різні завдання виявлення мають абсолютно різні цілі та обмеження, їхні труднощі можуть відрізнятися одна від одної. На додаток до деяких загальних проблем в інших задачах комп’ютерного зору, таких як об’єкти під різними кутами зору, освітлення та варіації всередині класу, проблеми з виявленням об’єктів включають такі аспекти: обертання об’єкта та зміни масштабу (наприклад, невеликі об’єкти), точна локалізація об'єкта, виявлення щільних і закритих об'єктів, прискорення виявлення тощо.

За останні два десятиліття розвиток виявлення об’єктів проходив через два історичні періоди: період традиційного виявлення об’єктів (до 2014 року) та період виявлення на основі глибокого навчання (після 2014 року).

Традиційні детектори. Більшість ранніх алгоритмів виявлення об’єктів були побудовані на основі ручних функцій. Через відсутність ефективного представлення зображень у той час людям доводиться створювати складні представлення функцій і різноманітні навички прискорення.

У 2001 році П. Віола та М. Джонс вперше досягли виявлення людських облич у реальному часі без будь-яких обмежень (наприклад, сегментація кольору шкіри) [7, 8]. Працюючи на процесорі Pentium III із тактовою частотою 700 МГц, детектор VJ був у десятки чи навіть сотні разів швидшим за інші алгоритми свого часу за порівнянної точності виявлення. Детектор VJ використовує найпростіший спосіб виявлення, тобто ковзні вікна: він перебирає всі можливі місця та масштаби зображення, щоб побачити, чи є в якомусь вікні людське обличчя. Детектор VJ значно підвищив швидкість виявлення завдяки застосуванню трьох важливих методів: «цілісного зображення», «вибору ознак» і «каскадів виявлення».

У 2005 році Н. Далал і Б. Тріггс запропонували дескриптор ознак гістограми орієнтованих градієнтів (HOG) [9]. HOG можна розглядати як важливе вдосконалення масштабно-інваріантного перетворення ознак і контекстів форми свого часу. Щоб збалансувати інваріантність ознак (включно з трансляцією, масштабом, освітленням тощо) і нелінійністю, дескриптор HOG розроблено для обчислення на щільній сітці рівномірно розташованих комірок і використовує нормалізацію локального контрасту, що перекривається (на «блоках»). Для виявлення об’єктів різного розміру детектор HOG змінює масштаб вхідного зображення кілька разів, зберігаючи розмір вікна виявлення незмінним. Детектор HOG був важливою основою багатьох детекторів об’єктів [10, 11] і великої різноманітності програм комп’ютерного зору протягом багатьох років.

Модель на основі деформованих частин (DPM) спочатку була запропонована P. Felzenszwalb [10] у 2008 році як розширення детектора HOG. Він дотримується філософії виявлення «розділяй і володарюй», де навчання можна просто розглядати як вивчення правильного способу декомпозиції об’єкта, а висновок можна розглядати як сукупність виявлень на різних частинах об’єкта. Пізніше Р. Гіршик розширив зіркову модель до «моделей суміші», щоб мати справу з об’єктами в реальному світі за більш значних варіацій, і вніс ряд інших удосконалень [11, 12].

Незважаючи на те, що сучасні детектори об’єктів значно перевершили DPM за точністю виявлення, багато з них все ще знаходяться під сильним впливом його цінних ідей, наприклад, моделі суміші, жорсткий негативний аналіз, регресія обмежувальної рамки, праймінг контексту тощо.

Двоступеневі детектори на базі CNN. Коли продуктивність ручних функцій стала насиченою, дослідження виявлення об’єктів досягли плато після 2010 року. У 2012 році світ побачив відродження згорткових нейронних мереж [13]. R. Girshick та ін. взяв на себе ініціативу вийти з глухого кута в 2014 році, запропонувавши регіони з функціями CNN (RCNN) [14]. Відтоді виявлення об’єктів почало розвиватися з безпрецедентною швидкістю. В епоху глибокого навчання існує дві групи детекторів: «двоступеневі детектори» та «одноступеневі детектори».

У RCNN процес починається з вилучення набору пропозицій об’єктів (блоків кандидатів на об’єкти) шляхом вибіркового пошуку [15]. Потім кожна пропозиція масштабується до зображення фіксованого розміру та вводиться в модель CNN, попередньо навчену на ImageNet (наприклад, AlexNet [13]), щоб витягнути функції. Нарешті, лінійні класифікатори SVM використовуються для прогнозування присутності об’єкта в кожному регіоні та розпізнавання категорій об’єктів. Хоча RCNN досяг значного прогресу, його недоліки очевидні: обчислення надлишкових функцій для великої кількості пропозицій, що перекриваються (понад 2000 блоків з одного зображення) призводять до

надзвичайно низької швидкості виявлення (14 с на зображення з GPU). Пізніше в тому ж році була запропонована SPPNet [16], яка вирішила цю проблему.

У 2014 році K. He та ін. запропонували просторові пірамідні мережі об'єднання (SPPNet) [16]. Попередні моделі CNN вимагають вхідних даних фіксованого розміру, наприклад, зображення 224x224 для AlexNet [13]. Основним внеском SPPNet є впровадження рівня об’єднання просторових пірамід (SPP), який дозволяє CNN генерувати представлення фіксованої довжини незалежно від розміру зображення/області інтересу без його масштабування. При використанні SPPNet для виявлення об’єктів карти функцій можна обчислити з усього зображення лише один раз, а потім можна створити представлення довільних областей із фіксованою довжиною для навчання детекторів, що дозволяє уникнути повторного обчислення згорткових функцій. Незважаючи на те, що SPPNet значно підвищив швидкість виявлення, у нього все ще є деякі недоліки: по-перше, навчання все ще є багатоетапним, по-друге, SPPNet лише налаштовує повністю пов’язані рівні, просто ігноруючи всі попередні рівні. Пізніше в наступному році була запропонована Fast RCNN [17], яка вирішила ці проблеми.

У 2015 році P. Гіршик запропонував детектор Fast RCNN [17], який є подальшим вдосконаленням R-CNN та SPPNet [16]. Fast RCNN дозволяє одночасно навчати детектор і регресор обмежувальної рамки в одній конфігурації мережі. Хоча Fast-RCNN успішно інтегрує переваги R-CNN і SPPNet, його швидкість виявлення все ще обмежена виявленням пропозиції.

У 2015 році S. Ren та ін. запропонував детектор Faster RCNN [18] - перший детектор глибокого навчання майже в реальному часі. Основним внеском Faster-RCNN є запровадження мережі регіональних пропозицій (RPN). Від R-CNN до Faster RCNN більшість окремих блоків системи виявлення об’єктів, наприклад виявлення пропозицій, виділення ознак, регресія обмежувальної рамки тощо, поступово інтегровано в уніфіковану

структуру наскрізного навчання. Хоча Faster RCNN долає вузьке місце у швидкості Fast RCNN, на наступному етапі виявлення все ще є надлишковість обчислень. Пізніше було запропоновано низку вдосконалень, у тому числі RFCN [19] і Light head RCNN [20].

У 2017 році T.-Y. Lin та ін. була запропонована FPN [21]. До FPN більшість детекторів на основі глибокого навчання запускали виявлення лише на картах функцій верхнього рівня мережі. Хоча функції в більш глибоких шарах CNN є корисними для розпізнавання категорій, це не сприяє локалізації об'єктів. З цією метою у FPN розроблено низхідну архітектуру з бічними зв'язками для побудови семантики високого рівня на всіх рівнях. Оскільки CNN природним чином формує піраміду ознак шляхом прямого поширення, FPN демонструє великі успіхи для виявлення об'єктів із широким розмаїттям масштабів. FPN став основним будівельним блоком багатьох новітніх детекторів.

Одноступеневі детектори на базі CNN. Більшість двоступеневих детекторів дотримуються парадигми обробки «від грубого до точного». Грубий етап прагне покращити здатність запам'ятовування, тоді як точний уточнює локалізацію на основі грубого виявлення та приділяє більше уваги здатності розпізнавання. Вони можуть легко досягти високої точності без будь-яких вдосконалень, але рідко використовуються в техніці через низьку швидкість і величезну складність. Навпаки, одноступеневі детектори можуть отримати всі об'єкти за один крок. Вони підходять для мобільних пристроїв із функціями, які легко розгортаються в режимі реального часу, але їх продуктивність помітно погіршується під час виявлення щільних і малих об'єктів.

You Only Look Once (YOLO) було запропоновано Р. Джозефом та ін. у 2015 р. Це був перший однокаскадний детектор в епоху глибокого навчання [22]. Парадигма YOLO повністю відрізняється від двоступеневих детекторів: застосування однієї нейронної мережі до повного зображення. Ця мережа

ділить зображення на регіони та прогнозує обмежувальні рамки та ймовірності для кожного регіону одночасно. Незважаючи на значне покращення швидкості виявлення, YOLO страждає від падіння точності локалізації порівняно з двоступеневими детекторами, особливо для деяких малих об’єктів. Подальші версії YOLO [23] і запропонований останнім SSD [25] приділяли більше уваги цій проблемі. Нещодавно було запропоновано YOLOv7 [24], який перевершує більшість існуючих детекторів об’єктів щодо швидкості та точності (діапазон від 5 кадрів/с до 160 кадрів/с) завдяки введенню оптимізованих структур, таких як динамічне призначення міток і перепараметризація структури моделі.

Single Shot MultiBox Detector (SSD) [25] був запропонований W. Liu та ін. у 2015 р. Основним внеском SSD є запровадження методів виявлення з кількома посиланнями та різною роздільною здатністю, що значно покращує точність виявлення одноступеневого детектора, особливо для деяких малих об’єктів. SSD має переваги як у швидкості виявлення, так і в точності. Основна відмінність SSD від попередніх детекторів полягає в тому, що SSD виявляє об’єкти різного масштабу на різних рівнях мережі, тоді як попередні виконують виявлення лише на своїх верхніх рівнях.

Незважаючи на високу швидкість і простоту, одноступеневі детектори роками поступалися за точністю двоступеневим детекторам. Т.-Й. Lin та ін. досліджували причини та запропонували RetinaNet у 2017 році [26]. Вони виявили, що центральною причиною є надзвичайний дисбаланс класів переднього плану та заднього плану, який спостерігається під час навчання детекторів щільності. З цією метою в RetinaNet було введено нову функцію втрат під назвою «фокальна втрата», яка змінила стандартну перехресну втрату ентропії, щоб під час навчання детектор зосереджувався на складних, неправильно класифікованих прикладах. Focal Loss дозволяє одноступеневим детекторам досягти порівнянної точності з двоступеневими детекторами, зберігаючи при цьому дуже високу швидкість виявлення.

Попередні методи в основному використовували блоки прив’язки для надання посилань на класифікацію та регресію. Об’єкти часто відрізняються за кількістю, місцем розташування, масштабом, співвідношенням тощо. Їм доводиться йти шляхом встановлення великої кількості еталонних блоків, щоб краще відповідати основним істинам для досягнення високої продуктивності. Однак мережа постраждала б від подальшого дисбалансу категорій, великої кількості вручну розроблених гіперпараметрів і тривалого часу конвергенції. Щоб вирішити ці проблеми, H. Law та інші [27] у CornerNet відкидають попередню парадигму виявлення та розглядають завдання як проблему прогнозування ключової точки (кути прямокутника). Після отримання ключових точок CornerNet роз’єднає та перегрупує кутові точки, використовуючи додаткову інформацію про вбудовування, щоб сформувати обмежувальні рамки.

X. Zhou та інші запропонували CenterNet [28] у 2019 році. Він також дотримується парадигми виявлення на основі ключових точок, але усуває дорогі постпроцеси, такі як групове призначення ключових точок і немаксимального придушення (Non-Maximum Suppression, NMS), що створює повністю наскрізну мережу виявлення. CenterNet розглядає об’єкт як одну точку (центр об’єкта) і регресує всі його атрибути (такі як розмір, орієнтація, розташування, поза тощо) на основі опорної центральної точки. Модель може інтегрувати виявлення 3-D об’єктів, оцінку пози людини, навчання оптичного потоку, оцінку глибини та інші завдання в єдину структуру.

Останніми роками «трансформери» глибоко вплинули на всю сферу глибокого навчання, зокрема на сферу комп’ютерного зору. Трансформери відмовляються від традиційного оператора згортки на користь обчислень лише за увагою, щоб подолати обмеження CNN і отримати сприйнятливе поле глобального масштабу. У 2020 році N. Carion та інші запропонували DETR [29], де вони розглядали виявлення об’єктів як проблему прогнозування та запропонували наскрізну мережу виявлення з трансформерами. Наразі

виявлення об’єктів вступило в нову еру, коли об’єкти можна виявляти без використання блоків прив’язки або точок прив’язки. Пізніше X. Zhu та інші запропонували деформований DETR [30] для вирішення проблеми тривалого часу конвергенції DETR та обмеженої продуктивності при виявленні малих об’єктів.

1.2. Концепції та технології розпізнавання об’єктів

1). Багатомасштабне виявлення об’єктів із різними розмірами та різними пропорціями є однією з головних технічних проблем у виявленні об’єктів.

Піраміди характеристик + розсувні вікна: З 2004 року на основі цієї парадигми було створено ряд детекторів етапів, включаючи детектор HOG, DPM тощо. Вони часто ковзають вікном виявлення фіксованого розміру по зображенню, приділяючи мало уваги «різним співвідношенням сторін».

Виявлення за допомогою пропозицій об’єктів: пропозиції об’єктів відносяться до групи незалежних від класу посилань, які, імовірно, містять будь-які об’єкти. Виявлення за допомогою пропозицій об’єктів допомагає уникнути вичерпного пошуку ковзного вікна по зображенню [31].

Глибока регресія та виявлення без прив’язки: Після 2018 року дослідники почали думати про проблему виявлення об’єктів з точки зору виявлення ключових точок. Ці методи часто ґрунтуються на двох ідеях: одна — груповий метод, який виявляє ключові точки (кути, центри або репрезентативні точки), а потім проводить об’єктне групування [27, 32]; інший — безгруповий метод, який розглядає об’єкт як одну/багато точок, а потім регресує атрибути об’єкта (розмір, співвідношення тощо) за посиланням на точки [28].

Виявлення з кількома посиланнями/роздільністю: Виявлення з кількома посиланнями зараз є найбільш використовуваним методом для багатомасштабного виявлення [33, 24, 25, 18]. Основна ідея виявлення кількох

посилань [33, 23, 25, 18] полягає в тому, щоб спочатку визначити набір посилань (так званих якорів, включаючи рамки та точки) у кожному місці зображення, а потім передбачити виявлення на основі цих посилань. Іншою популярною технікою є виявлення з різною роздільною здатністю [21, 25, 34], тобто шляхом виявлення об'єктів різного масштабу на різних рівнях мережі.

2) Праймінг контексту: Візуальні об'єкти зазвичай вбудовані в типовий контекст з навколишнім середовищем. Мозок людини використовує асоціації між об'єктами та середовищем, щоб полегшити візуальне сприйняття та пізнання [35].

Виявлення за допомогою локального контексту: локальний контекст відноситься до візуальної інформації в області, яка оточує об'єкт для виявлення. Нещодавні детектори на основі глибокого навчання також можуть бути покращені за допомогою локального контексту шляхом простого розширення сприйнятливого поля мереж або розміру пропозицій об'єктів [36, 37].

Виявлення за допомогою глобального контексту: глобальний контекст використовує конфігурацію сцени як додаткове джерело інформації для виявлення об'єктів. Для останніх детекторів існує два методи інтеграції глобального контексту. Перший метод полягає у використанні переваг глибокої згортки, розширеної згортки, деформованої згортки, операції об'єднання [38] для отримання великого рецептивного поля (навіть більшого, ніж вхідне зображення). Але тепер дослідники дослідили потенціал застосування механізмів на основі уваги (нелокальних, трансформерів тощо) для досягнення повного сприйнятливого поля зображення та досягли великого успіху [29]. Другий метод полягає в уявленні про глобальний контекст як про послідовну інформацію та її вивчення за допомогою рекурентних нейронних мереж [39].

Інтерактивний контекст: Інтерактивний контекст відноситься до обмежень і залежностей, які передаються між візуальними елементами. Деякі

нещодавні дослідження показали, що сучасні детектори можна вдосконалити, враховуючи контекстні інтерактиви. Деякі нещодавні вдосконалення можна згрупувати у дві категорії, де перша спрямована на дослідження зв'язків між окремими об'єктами [11, 40], а друга — на дослідження залежностей між об'єктами та сценами [41].

3) Жорсткий негативний видобуток: навчання детектора, по суті, є незбалансованою проблемою навчання. У випадку детекторів на основі ковзного вікна дисбаланс між фоном і об'єктами може досягати 107:1 [31]. У цьому випадку використання всіх фонів буде шкідливим для навчання, оскільки величезна кількість легких негативів перевантажить процес навчання. Жорсткий негативний майнінг (HNM) спрямований на подолання цієї проблеми. Щоб полегшити проблему дисбалансу даних під час навчання, такі детектори, як Faster RCNN і YOLO, просто балансують ваги між позитивним і негативним вікнами. Однак пізніше дослідники помітили, що це не може повністю вирішити проблему дисбалансу [26]. З цією метою після 2016 року для виявлення об'єктів було знову введено завантажувальний механізм [25, 42]. Альтернативним удосконаленням є розробка нових функцій втрат [26] шляхом зміни стандартної перехресної втрати ентропії таким чином, щоб вона приділяла більше уваги складним, неправильно класифікованим прикладам [26].

5) Немаксимальне придушення: оскільки сусідні вікна зазвичай мають подібні показники виявлення, немаксимальне придушення використовується як етап постобробки для видалення повторюваних обмежувальних рамок і отримання остаточного результату виявлення. На ранніх етапах виявлення об'єктів NMS не завжди була інтегрована. Це пояснюється тим, що на той час бажаний результат системи виявлення об'єктів був не зовсім ясним.

Жадібний вибір: жадібний вибір є старим, але найпопулярнішим способом виконання NMS. Для набору виявлень, що перекриваються, вибирається обмежувальна рамка з максимальним показником виявлення, а

сусідні рамки видаляються відповідно до попередньо визначеного порогу перекриття. Хоча жадібний відбір зараз став де-факто методом для NMS, він все ще має місце для вдосконалення. По-перше, поле з найбільшим результатом може бути не найкращим. По-друге, він може пригнічувати сусідні об'єкти. Нарешті, він не пригнічує помилкові спрацьовування [43]. Для вирішення зазначених вище проблем запропоновано багато робіт [44, 45].

Агрегація обмежувальних рамок: агрегація BB — це ще одна група методів для NMS [7, 43] з ідеєю об’єднання або кластеризації кількох перекриваючих обмежувальних рамок в одне кінцеве виявлення. Перевага цього типу методу полягає в тому, що він повністю враховує зв’язки об’єктів та їхнє просторове розміщення [46]. Деякі добре відомі детектори використовують цей метод, такі як детектор VJ [7] і Overfeat [47].

NMS на основі навчання: Остання група вдосконалень NMS, які нещодавно привернули велику увагу, це NMS на основі навчання [48, 49]. Основна ідея полягає в тому, щоб розглядати NMS як фільтр для повторної оцінки всіх необроблених виявлень і навчити NMS як частину мережі наскрізним способом або навчити мережу імітувати поведінку NMS. Ці методи продемонстрували багатообіцяючі результати щодо покращення оклюзії та виявлення щільних об’єктів порівняно з традиційними ручними методами NMS.

Детектор без NMS: Щоб вийти з NMS і створити повністю наскрізну навчальну мережу виявлення об’єктів, дослідники розробили серію методів для завершення призначення міток «один-до-одного» (він же один об’єкт із лише одним блоком передбачення) [28, 29]. Ці методи часто дотримуються правила, яке вимагає використання найякіснішого боксу для навчання, щоб отримати безкоштовну NMS. Детектори без NMS більше схожі на систему візуального сприйняття людини, а також є можливим шляхом у майбутнє виявлення об’єктів.

6). Межі виявлення зсувного вікна

Оскільки об’єкт на зображенні може бути однозначно визначений за його верхнім лівим кутом і нижнім правим кутом базової рамки істинності, завдання виявлення може бути еквівалентно сформульовано як проблема локалізації парних ключових точок. Однією з недавніх реалізацій цієї ідеї є прогнозування теплової карти для кутів [27]. Деякі інші методи слідують цій ідеї та використовують більше ключових точок (кут і центр [50], крайні та центральні точки [51], репрезентативні точки [32]), щоб отримати кращу продуктивність. Інша парадигма розглядає об’єкт як точку/точки та безпосередньо передбачає атрибути об’єкта (наприклад, висоту та ширину) без групування. Перевага цього підходу полягає в тому, що його можна реалізувати в рамках семантичної сегментації, і немає необхідності проектувати багатомасштабні блоки прив’язки. Більше того, розглядаючи виявлення об’єктів як встановлене передбачення, DETR [29, 30] повністю вивільняє його в рамці на основі посилань.

7). Надійне виявлення змін обертання та масштабу

Обертання об’єкта зазвичай можна побачити під час виявлення обличчя, виявлення тексту та виявлення об’єктів дистанційного зондування. Найбільш простим вирішенням цієї проблеми є виконання доповнення даних, щоб об’єкт у будь-якій орієнтації міг бути добре охоплений розподілом доповнених даних [52], або навчання незалежних детекторів окремо для кожної орієнтації [53]. Розробка функцій втрат, інваріантних обертанню, є останнім часом популярним рішенням, де додається обмеження на втрати при виявленні, щоб ознака обертаних об’єктів залишалася незмінною [54]. Іншим недавнім рішенням є вивчення геометричних перетворень об’єктів-кандидатів [55]. У двоступеневих детекторах об’єднання ROI має на меті отримати представлення ознак фіксованої довжини для пропозиції об’єкта з будь-яким розташуванням і розміром. Оскільки об’єднання ознак зазвичай виконується в декартових координатах, воно не є інваріантним для перетворення обертання.

Нещодавнє вдосконалення полягає у виконанні об’єднання ROI у полярних координатах, щоб характеристики могли бути стійкими до змін обертання [53].

Нещодавні дослідження були проведені для надійного виявлення масштабу як на етапі навчання, так і на етапі виявлення.

Навчання адаптації до масштабу: сучасні детектори зазвичай змінюють масштаб вхідних зображень до фіксованого розміру та поширюють втрату об’єктів у всіх масштабах. Недоліком цього є те, що виникне проблема дисбалансу масштабу. Побудова піраміди зображення під час виявлення може полегшити цю проблему, але не принципово [19]. Нещодавно вдосконалено нормалізацію масштабу для пірамід зображень (SNIP) [56], яка будує піраміди зображень як на етапах навчання, так і на етапі виявлення та лише повертає втрату деяких вибраних масштабів.

Масштабне адаптивне виявлення: у детекторах на основі CNN розмір і співвідношення сторін якорів зазвичай ретельно розроблені. Недоліком цього є те, що конфігурації не можуть адаптуватися до неочікуваних змін масштабу. Щоб покращити виявлення малих об’єктів, у деяких останніх детекторах пропонуються деякі методи «адаптивного збільшення» для адаптивного збільшення малих об’єктів у «більші» [57]. Ще одним нещодавнім удосконаленням є передбачення розподілу масштабу об’єктів на зображенні, а потім адаптивне змінення масштабу зображення відповідно до нього [58].

8). Покращення локалізації

Для підвищення точності локалізації в останніх детекторах є дві групи методів: 1) уточнення обмежувальної рамки та 2) нові функції втрат для точної локалізації.

Уточнення обмежувальної рамки: найбільш інтуїтивно зрозумілим способом покращити точність локалізації є уточнення обмежувальної рамки, яку можна розглядати як постобробку результатів виявлення. Одним із останніх методів є ітераційна подача результатів виявлення в регресор BB, доки прогноз не зійдеться до правильного місця та розміру [59]. Однак деякі

дослідники також стверджували, що цей метод не гарантує монотонність точності локалізації [59] і може призвести до виродження локалізації, якщо уточнення застосовується кілька разів.

Нові функції втрати для точної локалізації: у більшості сучасних детекторів локалізація об’єкта розглядається як проблема координатної регресії. Однак недоліки цієї парадигми очевидні. По-перше, регресійні втрати не відповідають кінцевій оцінці локалізації, особливо для деяких об'єктів з дуже великим співвідношенням сторін. По-друге, традиційний метод регресії ВВ не забезпечує впевненості локалізації. Коли є кілька ВВ, що перекриваються один з одним, це може призвести до збою в немаксимальному придушенні. Зазначені вище проблеми можна пом'якшити шляхом розробки нових функцій втрат. Найбільш інтуїтивно зрозумілим вдосконаленням є пряме використання IoU як втрати локалізації [44]. Крім того, деякі дослідники також намагалися покращити локалізацію за імовірнісною системою виведення [60]. На відміну від попередніх методів, які безпосередньо передбачають координати рамки, цей метод передбачає розподіл ймовірностей розташування обмежувальної рамки.

9). Навчання з втратою сегментації

Виявлення об'єктів і семантична сегментація є двома основними завданнями комп'ютерного зору. Останні дослідження показують, що виявлення об’єктів можна покращити шляхом навчання з втратою семантичної сегментації.

Щоб покращити виявлення за допомогою сегментації, найпростішим способом є розглядати мережу сегментації як екстрактор фіксованих ознак та інтегрувати її в детектор як допоміжні функції [61]. Перевагою цього підходу є те, що його легко реалізувати, а недоліком є те, що мережа сегментації може принести додаткові обчислення.

Іншим способом є введення додаткової гілки сегментації поверх вихідного детектора та навчання цієї моделі багатозадачним функціям втрат

(seg. + det.) [3, 61]. Перевагою є сег. brunch буде видалено на етапі виведення, і це не вплине на швидкість виявлення. Однак недоліком є те, що для навчання потрібні анотації зображень на рівні пікселів.

10) Метрики

В останні роки найбільш часто використовуваною оцінкою виявлення є «Середня точність (AP)», яка спочатку була представлена у VOC2007. AP визначається як середня точність виявлення за різних відкликань і зазвичай оцінюється за категорією. Середнє AP (mAP), усереднене за всіма категоріями, зазвичай використовується як остаточний показник ефективності. Щоб виміряти точність локалізації об'єкта, IoU між передбачуваним блоком і базовою істинністю використовується для перевірки того, чи перевищує воно попередньо визначене порогове значення, скажімо, 0,5. Якщо так, об'єкт буде ідентифіковано як «виявлений», інакше як «пропущений». Тоді 0,5-IoU mAP став де-факто метрикою для виявлення об'єктів.

Після 2014 року, у зв'язку з впровадженням наборів даних MS-COCO, дослідники почали приділяти більше уваги точності локалізації об'єктів. Замість використання фіксованого порогового значення IoU, MS-COCO AP усереднюється за кількома пороговими значеннями IoU від 0,5 до 0,95, що сприяє точнішій локалізації об'єкта та може мати велике значення для деяких реальних програм.

1.3. Визначення обмежень та постановка задачі

Виявлення об'єктів як проблема комп'ютерного зору за своєю суттю є складною, оскільки ідеальний детектор має забезпечувати як високу точність, так і високу продуктивність за розумних витрат енергії та обчислень.

Точність виявлення та швидкість виведення також залежать як від розмірів зображення, так і від розмірів об'єкта. Хоча вища роздільна здатність зображення забезпечує кращу точність, вилучаючи більше інформації зі сцени,

вона також знижує швидкість виведення. Тому вибір розміру зображення, який забезпечує правильний баланс між точністю та швидкістю, є надзвичайно важливим. Крім того, розміри об’єктів відіграють важливу роль у точності виявлення. У той час як детектори можуть досягти високої точності на середніх і великих об’єктах, майже всім детекторам важко виявити менші об’єкти на сцені [65].

Виявлення/відстеження об’єктів у реальному часі у HD-відео має велике значення для відеоспостереження та автономного водіння. Традиційні детектори об’єктів зазвичай розроблені для виявлення зображення, просто ігноруючи кореляції між кадрами відео. Удосконалення виявлення шляхом дослідження просторової та часової кореляції в умовах обмеження розрахунків є важливим напрямком досліджень.

Щоб забезпечити високу точність, детектор має бути надійним і локалізувати та класифікувати об’єкти реального світу зі значними варіаціями всередині класу (наприклад, варіаціями розміру, форми та типу об’єктів), пози та нежорсткі деформації. Для використання детекторів на основі прив’язок оптимізація прив’язок є проблемою, оскільки вони залежать від набору даних.

Ще одна серйозна проблема — підтримувати постійну продуктивність за різних погодних умов (дощ, сніг, хуртовина) та умов освітлення. У таких програмах, як автономне водіння, детектор також повинен враховувати захаращений фон, багатолюдні сцени та ефекти камери.

Нарешті, глибокі нейронні мережі, як правило, покладаються на підказки швидкого навчання, отже, переобладнуючи для розподілу навчальних даних і не узагальнюючи дані поза розподілом.

Виявлення об’єктів включає завдання як розпізнавання (наприклад, «класифікація об’єктів»), так і локалізації (наприклад, «регресія розташування»). Детектор об’єктів повинен відрізняти об’єкти певних цільових класів від фону на зображенні за допомогою точної локалізації та правильного передбачення категорійної мітки для кожного екземпляра

об’єкта. Передбачено, що обмежувальні рамки або піксельні маски локалізують ці екземпляри цільових об’єктів.

У формалізованому вигляді надано набір із N анотованих зображень $\{x_1, x_2, \ldots, x_N\}$, а для i-го зображення x_i є об’єкти M_i, що належать до категорій C з анотаціями:

$$y_i = \left\{(c_1^i, b_1^i), (c_2^i, b_2^i), \ldots, (c_{M_i}^i, b_{M_i}^i)\right\} \tag{1.1}$$

де c_j^i $\left(c_j^i \in C\right)$ та b_j^i (обмежувальна рамка або піксельна маска об’єкта) позначають категоріальні та просторові мітки j-го об’єкта в x_i відповідно. Детектор f параметризований θ. Для x_i передбачення y_{pred}^i має той самий формат, що й y_i:

$$y_{\text{pred}}^i = \left\{(c_{\text{pred}_1}^i, b_{\text{pred}_1}^i), (c_{\text{pred}_2}^i, b_{\text{pred}_2}^i), \ldots)\right\} \tag{1.2}$$

Нарешті, функція втрат ℓ встановлюється для оптимізації детектора як:

$$\ell(x, \theta) = \frac{1}{N} \sum_{i=1}^{N} \ell(y_{\text{pred}}^i, x_i, y_i; \theta) + \frac{\lambda}{2} \|\theta\|_2^2 \tag{1.3}$$

де другий член є регуляризатором із компромісним параметром λ. Різні функції втрат, такі як softmax loss [63] і focal loss [64], впливають на кінцеву ефективність виявлення.

Під час оцінки для оцінки якості локалізації використовується метрика під назвою «перехрестя-об’єднання» (IoU) між об’єктами та прогнозами (тут опускається індекс i):

$$\text{IoU}(b_{\text{pred}}, b_{\text{gt}}) = \frac{\text{Area}(b_{\text{pred}} \bigcap b_{\text{gt}})}{\text{Area}(b_{\text{pred}} \bigcup b_{\text{gt}})} \tag{1.4}$$

Тут b_{gt} відноситься до істиної обмежувальної рамки або маски. Порогове значення IoU Ω встановлюється, щоб визначити, чи повністю прогноз охоплює об’єкт чи ні (тобто IoU ≥ Ω; зазвичай дослідники встановлюють Ω = 0,5). Для виявлення об’єктів передбачення з правильною

категорійною міткою, а також успішне передбачення локалізації (відповідає критеріям IoU) вважаються позитивними, інакше прогнозування є негативним:

$$\text{Prediction} = \begin{cases} \text{Positive} & c_{\text{pred}} = c_{\text{gt}} \text{ and } \text{IoU}(b_{\text{pred}}, b_{\text{gt}}) > \Omega \\ \text{Negative} & \text{otherwise} \end{cases} \tag{1.5}$$

Для оцінки загальної проблеми виявлення об’єктів використовується середня середня точність (mAP) за класами *C*, а в реальних сценаріях, таких як виявлення пішоходів, використовуються різні метрики оцінювання. Окрім точності виявлення, швидкість виведення також є важливою метрикою для оцінки алгоритмів виявлення об’єктів. Зокрема, необхідно виявляти об’єкти у відеопотоці (виявлення в реальному часі), необхідно мати детектор, який може швидко обробити цю інформацію. Таким чином, ефективність детектора також оцінюється за кадрами в секунду (FPS), тобто скільки зображень він може обробити за секунду. Зазвичай детектор, який може досягти швидкості виведення 20 FPS, вважається детектором реального часу.

1.4. Висновки за розділом 1

Незважаючи на величезний прогрес у виявленні об’єктів, досягнутий за попередні роки, більшість детекторів все ще далекі від оптимального співвідношення показників точності та швидкодії. У міру зростання кількості реальних застосунків зростатиме потреба в компактних алгоритмах, які можна використовувати на мобільних і вбудованих пристроях.

В даний час модель глибокого навчання широко поширена у всій області комп’ютерного зору, включаючи загальне виявлення об’єктів і предметно-спеціальне виявлення об’єктів. Більшість найсучасніших детекторів об’єктів використовують мережі глибокого навчання як свою магістраль і мережу виявлення для вилучення характеристик із вхідних зображень (або відео), класифікації та локалізації відповідно.

У даному розділі було описано, як одноступеневі та двоступеневі детектори відрізняються один від одного. Двоступеневі детектори повільніші, їх можна використовувати для застосунків у реальному часі, хоча зазвичай вони більш точні. Однак за останні роки одноступеневі детектори стали такими ж точними та значно швидшими. Актуальним завданням є розробка швидких та набагато більш точних детекторів об'єктів для відео.

РОЗДІЛ 2
МОДЕЛІ ТА МЕТОДИ РОЗВ'ЯЗАННЯ ЗАДАЧІ

2.1. Розпізнавання об'єктів на основі нейромережевої моделі

2.1.1. Згорткові нейронні мережі (CNN)

Нейронні мережі регулярно використовуються в обробці зображень. Зображення фактично є тривимірними тензорами з трьома вимірами: висотою, шириною та глибиною кольору. Щоб використовувати як вхідні дані для звичайної нейронної мережі, тензор має бути зведений до довгого вектора ознак. У такого підходу є два недоліки. Вхідна розмірність є добутком трьох вимірів. Із зображенням 800×600 RGB розмір вхідного вектора мережі вже перевищує мільйон скалярів. По-друге, функції, які вивчає мережа, залежать від положення ознак, тобто виявлення об'єкта в деякій позиції (x_1, y_1) має вивчатися окремо від виявлення того самого об'єкта в позиції (x_2, y_2). Згорткові шари [66] вводять інваріантність позиції в нейронні мережі.

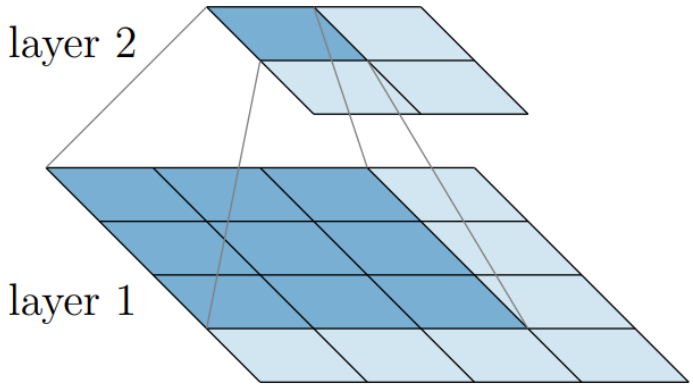


Рис. 2.1. Згортковий шар (кожен нейрон у шарі 2 знаходиться під впливом перекриваючої області нейронів 3×3 у шарі 1).

Нейрони на згортковому шарі сприймають лише невелику прямокутну область у вхідному зображенні. Ті самі параметри переміщуються по зображенню, що робить виявлення однаковими незалежно від місця розташування. Мотивація такої моделі полягає в тому, щоб зменшити

кількість параметрів, які можна навчити, і це виправдано тим фактом, що цікаві особливості природних зображень є локальними та можуть бути захоплені маленькими локальними вікнами. Розподіл ваги та функції локального підключення значно зменшують кількість параметрів, які можна вивчати. Розрахунок виконується за допомогою операції математичної згортки, яка визначена [66] для дискретного двовимірного випадку як

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n), \tag{2.1}$$

де I — вхідне зображення, а K — ядро згортки. Ядро K оптимізується під час навчання.

Як і в повністю зв'язаних мережах, вихідні дані операції згортки потім перетворюються за допомогою функції активації. Накопичування кількох згорткових шарів призводить до того, що перші шари зосереджуються на природних об'єктах, таких як краї, а шари шарів виявляють складніші форми шляхом компонування виявлень попередніх шарів. При відповідній ієрархії нейрони на найвищому рівні активуються, коли виявляють характеристики високого рівня, такі як присутність об'єкта на зображенні.

Глибокі згорткові нейронні мережі створили поле глибокого навчання. Глибокі моделі містять кілька згорткових шарів, де на нейрони, вищі в ієрархії, впливають більші області вхідних зображень. Ця область відома як рецептивне поле.

Рівні об'єднання розділяють вхідні дані на $N \times N$ вікон, що не перекриваються, і зменшують значення в кожному вікні до одного на виході. Дві операції об'єднання, які зазвичай використовуються, — це максимальне об'єднання, де зберігається максимальне значення кожного вікна, і об'єднання середніх значень, коли значення усереднюються разом.

Операція об'єднання виконується окремо для кожного каналу карти функцій, так що результат об'єднання карти $H \times W \times C$, де H – висота, W – ширина, а C – кількість каналів, з 2×2 операція об'єднання призводить до

тензора форми $W/2 \times H/2 \times C$. Шари об'єднання параметризуються розміром вікна N і не містять параметрів, які можна вивчати.

Мотивація об'єднання шарів подвійна. По-перше, вони вводять інваріантність позиції до виявлень, оскільки операція об'єднання відкидає інформацію про розташування значень. Об'єднання також зменшує розмір вихідної карти функцій, що збільшує загальне сприйнятливе поле мережі та сприяє швидшому виведенню та навчанню, хоча й за рахунок просторової точності.

Виявлення об'єктів поєднує як класифікацію, так і локалізацію, надаючи мітки класів і координати обмежувальних рамок примірників об'єктів. Детектори об'єктів на основі згорткової нейронної мережі зазвичай класифікуються на дві категорії, а саме. двоступеневий і одноступеневий методи виявлення.

2.1.2. Одноступеневе виявлення

Одноступеневі детектори складаються з єдиної наскрізної прямої мережі, яка виконує класифікацію та регресію в монолітному середовищі. Ці детектори не мають окремого етапу для створення пропозицій і натомість розглядають усі позиції на зображенні як потенційні пропозиції. Кожен із них використовується для прогнозування ймовірностей класу, розташування обмежувальної рамки та показників достовірності. Показники достовірності визначають, наскільки мережа впевнена щодо прогнозування свого класу. Дві основні категорії одноступеневих детекторів – це детектори на основі прив'язки та детектори на основі ключових точок.

Детектори на основі прив'язки використовують заздалегідь визначені прив'язки (або попередні), щоб допомогти прогнозувати. Яскравими прикладами цього підходу є You Only Look Once [70, 71] і Single Shot Detector [72]. YOLO працює, візуалізуючи вхідне зображення як сітку клітинок, де

кожна клітинка відповідає за прогнозування обмежувальної рамки, якщо центр рамки потрапляє в клітинку. Кожна клітинка сітки передбачає кілька обмежувальних рамок і виводить розташування та мітку класу разом із оцінкою достовірності. SSD був першим однокаскадним детектором, який відповідає точності сучасних двокаскадних детекторів, зберігаючи при цьому швидкість реального часу. SSD прогнозує оцінки та зсуви коробки для фіксованого набору прив'язок різних масштабів у кожному місці на кількох картах функцій, отриманих із мережі піраміди функцій (FPN). FPN полегшує функції мультироздільності [73].

Недоліком детекторів на основі прив'язок є робота з такими гіперпараметрами, як кількість, співвідношення сторін і розмір прив'язок, які сильно залежать від набору даних [69]. Це призвело до введення нової парадигми детекторів об'єктів без прив'язки (также на основі ключових точок). Методи, засновані на ключових точках, сприймають об'єкти як точки замість того, щоб моделювати їх як обмежувальні рамки. Ключові точки, такі як кути або центри об'єктів, оцінюються, а ширина та висота регресують із цих точок замість попередньо визначених прив'язок. Було представлено кілька мереж на основі ключових точок, а саме CornerNet, CenterNet, FCOS, NanoDet і TTFNet [74-78].

Незважаючи на те, що як детектори на основі прив'язки, так і детектори на основі ключових точок досягли надзвичайної точності у загальному виявленні об'єктів, у ньому значною мірою домінували архітектури на основі CNN, яким бракує глобального контексту. Крім того, сучасні детектори зазвичай виконують регресію та класифікацію на великому наборі пропозицій, прив'язок або центрів вікон. Таким чином, їх продуктивність страждає від складних завдань постобробки, таких як NMS (немаксимальне придушення для видалення майже ідентичного набору пропозицій) [69]. Трансформери були представлені як альтернативна архітектурна парадигма CNN. Детектори на основі трансформерів, такі як DETR [79], використовують модулі

самоконтролю, які явно моделюють усі взаємодії між елементами в заданій послідовності, таким чином забезпечуючи глобальний контекст. Загальна конструкція трансформаторів також обходить ручні процеси, такі як NMS, роблячи прямі прогнози на заданому вході.

2.1.3. Двоступеневе виявлення

Двоступеневі детектори складаються з окремої мережі регіональних пропозицій (RPN) разом із класифікацією. Витягнуті ознаки із запропонованої регіону інтересу (ROI) у RPN передаються як голові класифікації, щоб визначити мітки класу, так і голові регресії, щоб визначити обмежувальні рамки [80-84]. Згорточна нейронна мережа на основі регіонів (RCNN) використовує алгоритм вибіркового пошуку для пошуку областей пікселів на зображенні, які можуть представляти об’єкти, а кандидати з короткого списку потім передаються через CNN [80]. Отримані ознаки з CNN стають вхідними даними для опорної векторної машини (SVM), яка класифікує регіони, і для регресора, який передбачає обмежувальні прямокутники. RCNN вимагає поступового багатоетапного навчання і є досить повільним. Щоб подолати недолік RCNN, в Fast-RCNN були запропоновані певні модифікації [81]. По-перше, замість того, щоб використовувати CNN для виділення ознак регіонів, запропонованих вибірковим пошуком, CNN використовується для безпосереднього виділення характеристик всього зображення.

По-друге, класифікатор SVM і регресор замінені повністю пов’язаними шарами. Faster-RCNN запропонував подальші вдосконалення, щоб позбутися повільного алгоритму вибіркового пошуку для пропозицій регіону. Функції, отримані магістральною CNN, надсилаються до додаткової мережі регіональних пропозицій (RPN) на основі CNN, яка надає регіональні пропозиції [82]. Однак, незважаючи на високу точність, вищезгадані двоступеневі методи виявлення не підходять для додатків у режимі реального

часу [85-87]. Згодом було запропоновано двоступеневий детектор під назвою ThunderNet з ефективним RPN у поєднанні з невеликою магістралью для виявлення в реальному часі.

2.1.4. Формулювання виявлення об'єкта

Основні компоненти

Проблема виявлення об’єкта може бути формалізована так [69]: задано довільне зображення I_i та попередньо визначений список класів об’єктів, модель виявлення об’єкта не тільки класифікує тип екземплярів об’єкта, присутніх на зображенні, $\{c_1, c_2, ..., c_m\}$, але також повертає розташування кожного об’єкта у формі обмежувальних рамок $\{b_1, b_2, ..., b_m\}$, де $b_i = \{(x_1, y_1), (x_2, y_2)\}$ — верхня ліва та нижня права координати обмежувальної рамки. Детектори об'єктів, як одноступеневі, так і двоступеневі, зазвичай складаються з екстрактора ознак (надалі магістраль) і головки виявлення. Основою зазвичай є мережа на базі CNN, яка виділяє найвидатніші представлення сцени (від функцій низького до високого рівня). Більшість магістралей використовують шари об’єднання/згортки, щоб поступово зменшувати розмір карт функцій і збільшувати сприйнятливе поле мережі. Потім вихідні карти ознак передаються до головки виявлення, яка виконує класифікацію та регресію для визначення мітки та розташування екземплярів об’єктів (на рис. 2.2 показано загальне виявлення об’єктів).

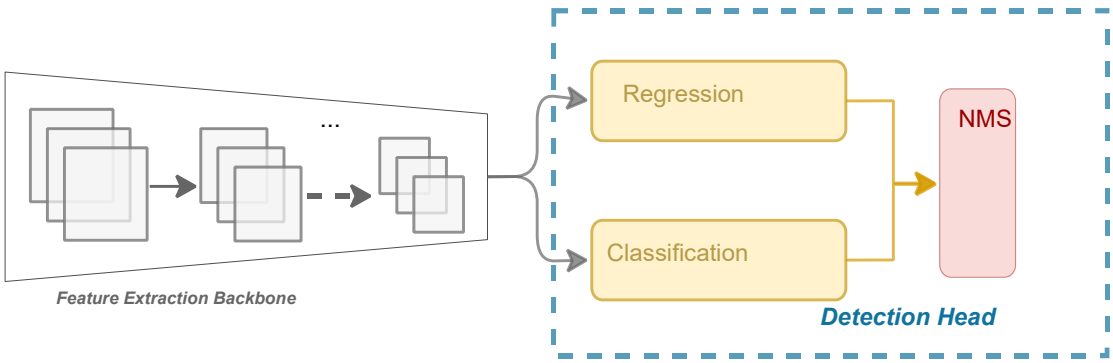


Рис. 2.2. Принципова діаграма основних компонентів детектора об’єктів

Функції втрат

Дві цільові функції зазвичай використовуються для навчання детектора на основі CNN, це класифікація та регресійні втрати. Втрата класифікації зазвичай визначається втратою крос-ентропії (CE):

$$\mathcal{L}_{CE} = -\sum_{i=1}^{n} t_i \log(p_i) \tag{2.2}$$

де t_i — мітка базової істинності, а p_i — м'яка максимальна ймовірність для i-го класу. Однак втрата CE не враховує незбалансовані набори даних, де менш часті приклади набагато важче вивчити порівняно з об'єктами, які часто зустрічаються. Таким чином, Lin et al. [88] запропонував фокусні втрати (FL), які вирішують дисбаланс класів, призначаючи більшу важливість важким зразкам, одночасно зменшуючи внесок втрат легких зразків,

$$\mathcal{L}_{FL} = -\alpha_i (1 - p_i)^{\gamma} \log(p_i) \tag{2.3}$$

де α_i – ваговий параметр, $\gamma \geq 0$ – настроюваний модулюючий параметр.

Втрата регресії зазвичай є втратою L_1 (найменше абсолютне відхилення) або L_2 (помилка найменшого квадрата) на всіх чотирьох координатах обмежувальної рамки між істиною та прогнозованою обмежувальною рамкою.

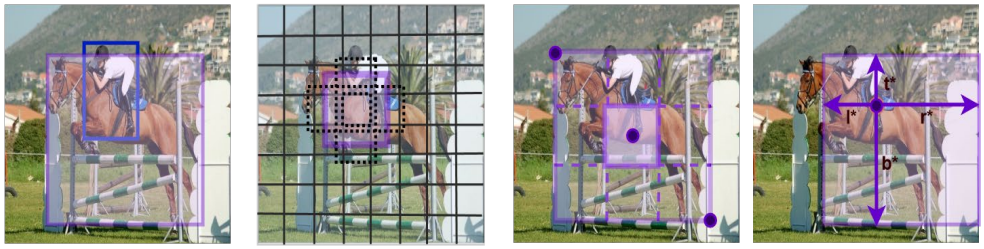
Рамка прив'язки (Anchor Box) та ключова точка (Keypoint)

Методи виявлення об'єктів на основі прив'язки використовують концепцію рамок прив'язки (у літературі їх також називають попередніми рамками). У цьому підході зображення ділиться на сітки, у яких кожну комірку сітки можна призначити кільком попередньо визначеною рамкою прив'язки (рис. 2.3 б). Ці поля визначаються для фіксації масштабу та співвідношення сторін конкретних класів об'єктів і зазвичай вибираються на основі розмірів об'єктів у навчальному наборі даних. Перетин через об'єднання (IoU) обчислюється між прив'язками та обмежувальними прямокутниками, а прив'язка, яка має найбільше перекриття, використовується для прогнозування розташування та класу цього об'єкта. Якщо перекриття рамки

прив’язки точки з опорним полем вище заданого порогового значення, це вважається позитивною прив’язкою. Мережа, замість прямого прогнозування обмежувальних рамок, передбачає зміщення рамок прив’язки і повертає унікальний набір прогнозів для кожної. Використання рамок прив’язки допомагає виявляти кілька об’єктів, об’єкти різного масштабу та об’єкти, що перекриваються.

Проте підходи, засновані на прив’язці, мають два серйозні недоліки [69]. По-перше, потрібна дуже велика кількість рамок прив’язки, щоб забезпечити достатнє перекриття з базовими блоками, і на практиці лише невелика частина перекривається з базовими блоками. Це створює величезний дисбаланс між позитивними та негативними прив’язками, таким чином збільшуючи час навчання. По-друге, розмір, форма та співвідношення сторін рамок прив’язки сильно залежать від набору даних і тому вимагають тонкого налаштування для кожного набору даних. Однак такий вибір робиться за допомогою спеціальної евристики (з використанням базових блоків набору даних), яка стає значно складнішою з багатомасштабними архітектурами, де кожна шкала використовує різні функції та власний набір прив’язок.

Щоб пом’якшити вищезазначені проблеми, пропонуються методи виявлення об’єктів на основі ключових точок, які не використовують прив’язки [74-76]. Проблема виявлення переформульована як попіксельне передбачення, подібне до сегментації. Функції CNN використовуються для створення теплових карт, де піки інтенсивності представляють ключові точки, такі як кути або центри відповідних об’єктів. Поряд з ними існують додаткові гілки, які передбачають розміри рамок (ширину і висоту). Прогнози теплової карти разом із вбудовуваннями використовуються для оцінки правильного розташування та розміру передбачених рамок. Для центральних ключових точок передбачені відстані від центрів до чотирьох сторін обмежувальної рамки об’єкта для виявлення (рис. 2.3 в).



а б в

Рис. 2.3. (а) істиний об’єкт; (б) методи на основі прив’язки; (в) методи на основі ключових точок.

Немаксимальне придушення (NMS)

Детектори об'єктів створюють занадто багато пропозицій, багато з яких зайві. Щоб видалити щільний набір дублікатів передбачень, детектори зазвичай використовують етап постобробки, який називається NMS. Модуль NMS спочатку сортує прогнозовані пропозиції для кожного екземпляра за їх показником достовірності та вибирає пропозицію з найвищою достовірністю [69]. Згодом виконується перекриття Жаккара з майже ідентичним набором пропозицій для кожного екземпляра,

$$IoU(b_m, b_i) = \frac{b_m \cap b_i}{b_m \cup b_i} \tag{2.4}$$

де b_m є пропозицією з найвищою достовірністю, а b_i представляє кожну з майже ідентичних пропозицій для того самого екземпляра. Дублікати видаляються, якщо це значення перевищує встановлений поріг NMS (зазвичай 0,5).

Однак одна з проблем, пов’язаних із NMS, полягає в тому, що дійсні пропозиції відхиляються, якщо пропозиції (для різних випадків) близькі одна до одної або в деяких випадках збігаються. Це особливо актуально для багатолюдних сцен. Тому для покращення обмеження NMS було запропоновано Soft-NMS. У Soft-NMS оцінки пропозицій виявлення (для інших випадків), які мають трохи менше перекриття з b_m, зменшуються,

водночас гарантуючи, що пропозиції, які мають більше перекриття з b_m, зменшуються більше, щоб можна було видалити дублікати. Це робиться шляхом обчислення добутку оцінки довіри (пропозиції b_i) і негативного значення IoU з b_m, як показано нижче:

$$s_i = \begin{cases} s_i, & \text{if } IoU(b_m, b_i) < threshold \\ s_i(1 - IoU(b_m, b_i)), & \text{if } IoU(b_m, b_i) \geq threshold \end{cases} \tag{2.5}$$

Детектори на основі ключових точок не використовують NMS на основі IoU, оскільки вони обробляють точки на теплових картах замість блоків, що перекриваються. NMS у цих мережах є простою операцією максимального пулу на основі пікових навантажень, яка обчислювально є менш дорогою.

2.2. Технології виявлення об’єктів в режимі реального часу

2.2.1. Одиночний мультирамковий детектор (SSD)

SSD [72] має CNN прямого зв’язку, який створює обмежувальні прямокутники, оцінки достовірності та класифікаційні мітки для кількох екземплярів об’єктів у сцені. SSD використовує численні карти функцій від поступового зменшення роздільної здатності, емулюючи вхідні зображення різних розмірів, водночас розподіляючи обчислення в різних масштабах. У той час як карти функцій з неглибоких шарів використовуються для вивчення грубих особливостей менших об’єктів, характеристики з більш глибоких шарів використовуються для локалізації більших об’єктів на сцені.

Структура SSD [89], як показано на рис. 2.4, побудована на структурі VGG-16, але вона викорінює повністю підключені шари. SSD використовує ідею прив’язки до швидшого R-CNN. Обмежувальна рамка визначається попередніми рамками з різними значеннями та вимірюваннями, потім CNN виділяє характеристики, після чого класифікація та регресія виконуються правильно. Весь метод виконується лише в одному процесі. Мережа SSD вибирає пріоритети, які містять цікавий об’єкт, і знаходить координати, які точно відповідають формі об’єкта. Ці повністю згорткові обмежувальні рамки є такими ж, як рамки прив’язки, що використовуються в мережах регіональних пропозицій. Метод спрощується шляхом зміни шарів згортки 3*3 на шари 1*1 для прогнозування значень конкретного об’єкта, тому проміжний рівень не потрібен.

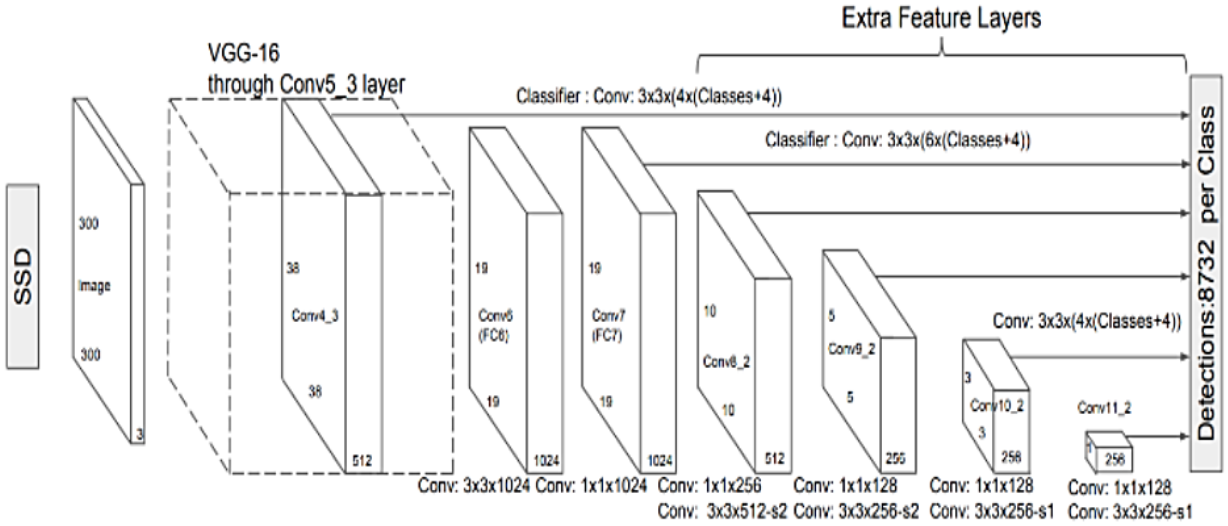


Рис. 2.4. Архітектура SSD

Головка виявлення використовує окремі попередньо визначені прив'язки для кожного масштабу карти об'єктів і об'єднує прогнози всіх прив'язок за замовчуванням у різних масштабах і співвідношеннях сторін (рис. 2.5). Масштаб і розмір прив'язок для кожної карти функцій k визначається як:

$$s_k = s_{min} + \frac{s_{max} - s_{min}}{m - 1}(k - 1) \tag{2.6}$$

де $k \in [1, m]$, а значення за замовчуванням для s_{min} і s_{max} задані як 0,2 і 0,9 відповідно. $m = 6$ карт функцій використовуються в SSD.

SSD створює різноманітний набір прогнозів, що охоплюють екземпляри об'єктів різних форм і розмірів. SSD використовує стратегію відповідності, щоб визначити, які прив'язки відповідають основній істині, а прив'язки – найвище перекриття використовується для прогнозування розташування та класу цього об'єкта.

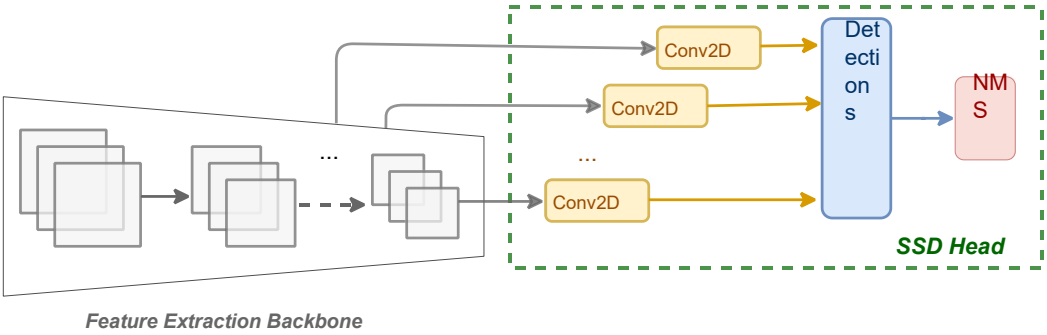


Рис. 2.5. Принципова діаграма одноступеневого анкерного детектора SSD.

Цільова функція походить від об'єкта multibox [90] і розширена для кількох категорій. Загальна цільова функція

$$\mathcal{L} = \frac{1}{N_{pos}}\mathcal{L}_{cls} + \lambda\mathcal{L}_{reg} \tag{2.7}$$

де L_{cls} — втрата перехресної ентропії, L_{reg} — сума втрат L_1 за всіма властивостями обмежувальної рамки для відповідних позитивних прямокутників. N_{pos} — кількість позитивних проб, а λ — балансувальна вага.

Важливо, що дизайн SSD є першою роботою, яка об'єднує різні значення багатьох функцій і з різною роздільною здатністю класифікує об'єкт і покращує якість виявлення об'єкта. Конструкція SSD є першою технікою, яка використовує значення обмежувальної рамки з характеристиками різної роздільної здатності в мережі. Крім того, SSD виявляє кілька об'єктів без спільного використання згорткових шарів. SSD має ту саму мету, що й MultiBox, але він може виявляти кілька категорій за один раз, на відміну від двоступеневого методу. Після цього виконується жорсткий негативний майнінг, де беруться найвищі попередні значення, що призводить до швидшої роботи мережі. Кінцеві результати отримують шляхом виконання немаксимального придушення (NMS) на багатомасштабних обмежувальних прямокутниках.

Тим не менш, SSD не є ефективним у правильному пошуку деяких маленьких об'єктів, але його можна розробити, використовуючи кращу

модель екстрактора функцій, як-от ResNet101, і використовуючи кращу мережу, як-от Stem Block і Dense Block [91]. Крім того, кращий алгоритм виявлення об’єктів SSD наведено Шепін Чжай, Дінгронг Шан, Шухуан Ван і Сусу Донг у своїй статті [92]. Зберігаючи SSD як основний алгоритм, дається спеціальна мережа вилучення DenseNet-S-32-1, яка замінює мережу VGG-16. Для виявлення кількох об’єктів представлений інший механізм, який називається злиттям, щоб поєднати характеристики низького та високого рівнів.

2.2.2. Рамки прив’язки (Anchor Boxes)

Моделі виявлення об’єктів використовують рамки прив’язки для прогнозування обмежувальної рамки. Розуміння та ретельне налаштування рамок прив’язки моделі може бути дуже важливим важелем для покращення продуктивності моделі виявлення об’єктів, особливо якщо у в ній присутні об’єкти неправильної форми.

Під час виявлення об’єктів потрібно п ідентифікувати та локалізувати об’єкти, як вони з’являються на зображенні. Виявлення об’єктів відрізняється від класифікації зображень, оскільки на зображенні може бути кілька об’єктів одного або різних класів, і виявлення об’єктів прагне точно передбачити всі ці об’єкти.

Моделі виявлення об’єктів вирішують це завдання, розбиваючи крок передбачення на дві частини: спочатку вони передбачають обмежувальну рамку за допомогою регресії, а по-друге, передбачають мітку класу за допомогою класифікації.

Щоб передбачити та локалізувати багато різних об’єктів на зображенні, більшість сучасних моделей виявлення об’єктів, таких як EfficientDet,YOLO і SSD, починають із рамок прив’язки, а потім налаштовуються на них (рис.2.6).

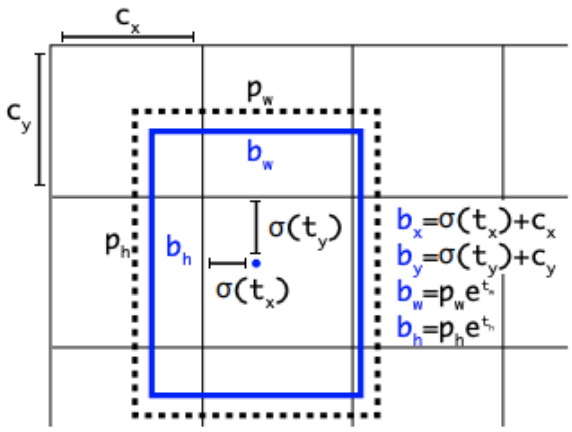


Рис. 2.6. Передбачення обмежувальної рамки на основі рамки прив’язки

Сучасні моделі зазвичай використовують обмежувальні рамки в такому порядку:

- фомується тисячі потенційних прив’язок навколо зображення;
- для кожної рамки прив’язки передбачається деяке зміщення від цієї рамки як рамки-кандидата;
- обчислюється функція втрат на основі істиного прикладу;
- обчислюється ймовірність того, що задана рамка зсуву перекривається реальним об’єктом;
- якщо ця ймовірність більша за 0,5, додається прогноз до функції втрат;
- винагороджуючи та штрафуючи рамки прив’язки, модель повільно підтягується до локалізації лише справжніх об’єктів.

Ось чому, після початкового навчання моделі, можна побачити рамки прив’язки, які з’являтимуться всюди. Після завершення навчання модель робитиме лише високоімовірні ставки на основі зміщень рамок прив’язки, які вона вважає найбільш ймовірно реальними.

2.2.3. Архітектура VGG16

VGG16 відноситься до моделі VGG, яка також називається VGGNet. Це модель згорткової нейронної мережі (CNN), яка підтримує 16 рівнів [67]. Ця модель відрізнялася від попередніх високопродуктивних моделей кількома параметрами. По-перше, він використовував крихітне сприйнятливе поле 3×3 із кроком 1 піксель — для порівняння AlexNet використовував рецептивне поле 11×11 із кроком 4 пікселя. Фільтри 3×3 поєднуються, щоб забезпечити функцію більшого сприйнятливого поля. Перевага використання кількох менших рівнів замість одного великого полягає в тому, що більше нелінійних шарів активації супроводжують шари згортки, покращуючи функції прийняття рішень і дозволяючи мережі швидко об’єднуватися.

По-друге, VGG використовує менший згортковий фільтр, який зменшує тенденцію мережі до надмірної підгонки під час навчальних вправ. Фільтр 3×3 є оптимальним розміром, оскільки менший розмір не може отримувати інформацію зліва-направо та вгору-вниз. Таким чином, VGG є найменшою можливою моделлю для розуміння просторових особливостей зображення. Послідовні згортки 3×3 спрощують керування мережею.

VGG16, як випливає з назви, є 16-рівневою глибокою нейронною мережею. Таким чином, VGG16 є відносно розгалуженою мережею із загальною кількістю 138 мільйонів параметрів — це величезна кількість навіть за сучасними мірками. Однак простота архітектури VGGNet16 є її головною привабливістю.

Архітектура VGGNet включає в себе найважливіші функції згорткової нейронної мережі. Мережа VGG складається з малих згорткових фільтрів. VGG16 має три повністю зв'язані шари та 13 згорткових шарів (рис. 2.7). Щоб бути корисним для класифікації, вихід кінцевого модуля потім зрівнюється до єдиного вектора, який після трьох повністю пов’язаних рівнів остаточно

перетворюється на розподіл ймовірностей між 1000 класами за допомогою функції активації soft-max [66].

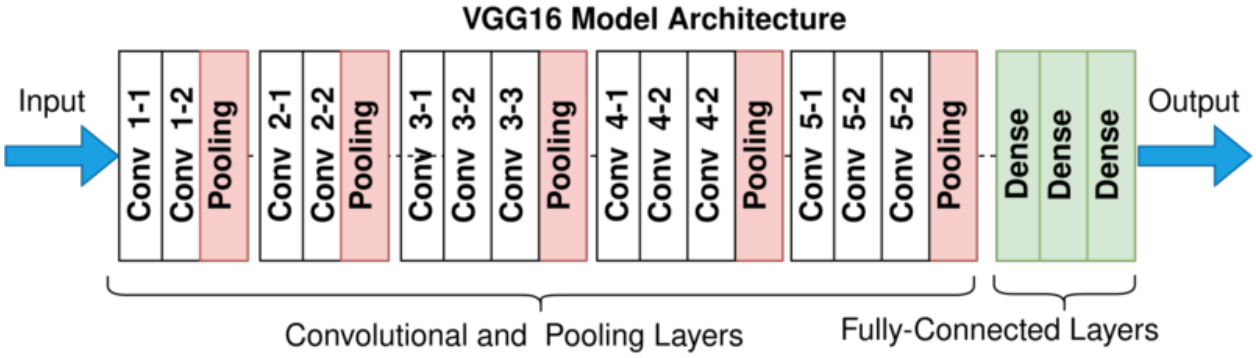


Рис. 2.7. Структура мережі VGG16

Стислий опис архітектури VGG:

Вхід — VGGNet отримує вхідне зображення 224×224. У конкурсі ImageNet творці моделі зберегли постійний розмір вхідного зображення, обрізавши частину 224×224 від центру кожного зображення [68].

Згорткові шари — згорткові фільтри VGG використовують найменше сприйнятливе поле 3×3. VGG також використовує згортковий фільтр 1×1 як лінійне перетворення вхідних даних.

Активація ReLu — наступним є компонент Rectified Linear Unit Activation Function (ReLU), основна інновація AlexNet для скорочення часу навчання. ReLU — це лінійна функція, яка забезпечує відповідний вихід для позитивних входів і виводить нуль для негативних входів. VGG має встановлений крок згортки в 1 піксель, щоб зберегти просторову роздільну здатність після згортки (значення кроку відображає, скільки пікселів «переміщує» фільтр, щоб охопити весь простір зображення).

Приховані рівні — усі приховані рівні мережі VGG використовують ReLU замість локальної нормалізації відповіді, як AlexNet. Останнє збільшує час навчання та споживання пам’яті з невеликим покращенням загальної точності.

Об’єднання шарів – шар об’єднання слідує за кількома згортковими шарами – це допомагає зменшити розмірність і кількість параметрів карт функцій, створених кожним кроком згортки. Об’єднання має вирішальне значення, враховуючи швидке зростання кількості доступних фільтрів з 64 до 128, 256 і, зрештою, до 512 на кінцевих рівнях.

Повністю підключені рівні — VGGNet включає три повністю підключені рівні. Перші два рівні мають по 4096 каналів, а третій рівень має 1000 каналів, по одному для кожного класу.

2.3. Обґрунтування алгоритму виявлення об’єктів у відеоряді

Алгоритм SSD використовує мережу VGG-16 як свою базову мережу. Він змінює повністю зв’язані шари (fc6 і fc7) на шари згортки (conv_fc6 і conv_fc7), а потім додає шари згортки різних розмірів для виконання багатомасштабного прогнозування цілей. Його мережева структура показана на рис. 2.8. На рис. 2.8 видно, що кілька нещодавно доданих шарів згортки за мережею VGG-16 відповідають характерній реакції цілі в різних масштабах. На додаток до рівня функцій conv4_3 самої мережі VGG-16, conv4_3, conv_fc7, conv6_2 використовуються в цілому, conv7_2, conv8_2 і conv9_2 — це шестирівневі функціональні шари для прогнозування багатомасштабних цілей. Модель на основі SSD [93] усуває генерацію регіональних кадрів-кандидатів і використовує повну згортку для прямого виконання регресійного прогнозування щодо інформації про положення та категорію цілей різних масштабів і форм. У той же час прогнози цілей у різних масштабах розосереджені на різних рівнях функцій для реалізації, що значно покращує загальну швидкість виявлення та точність мережі.

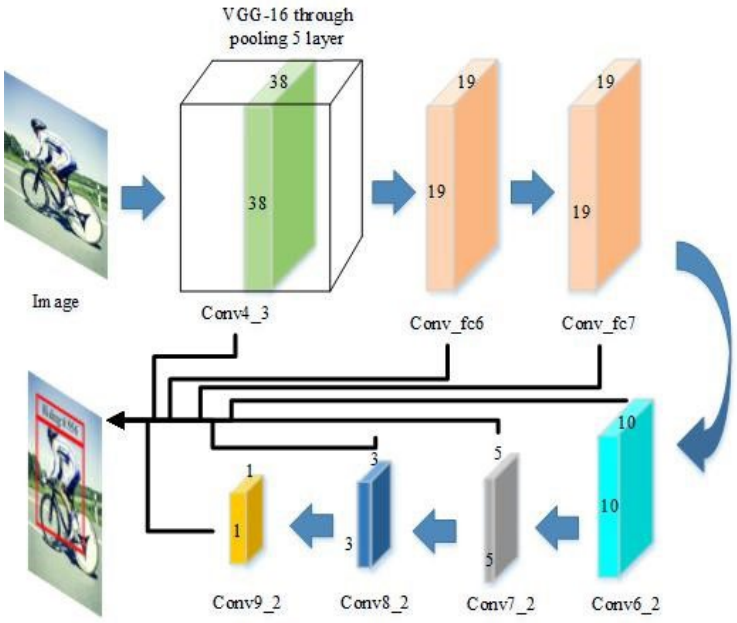


Рис. 2.8. Структура моделі SSD

Модель виявлення цілей SSD поєднує в собі ідею сегментації простору вибірки за прив'язкою в швидшому R-CNN. Цільові передбачувані шари в різних масштабах генеруватимуть форми коробок за замовчуванням з різними масштабами. Ці кадри-кандидати за замовчуванням зосереджені на кожному пікселі відповідного шару ознак і рівномірно розподілені по всьому шару ознак для прогнозування цілей різних масштабів і позицій. Форма прямого регресійного прогнозування повної згортки усуває процес генерації регіональних кадрів-кандидатів, що значно покращує швидкість виявлення мережі SSD. І його структура проста, механізм прогнозування цілі єдиний, а точність виявлення висока. Зберігаючи продуктивність виявлення в реальному часі, він може отримати вищу точність виявлення. Завдяки багатомасштабній структурі об'єднання функцій і структурі прогнозування Inception [93] можна значно підвищити точність виявлення мережі, зберігаючи при цьому високу швидкість виявлення.

Модель SSD реалізує ієрархічне прогнозування багатомасштабних цілей за допомогою шарування функцій. Шари функцій нижчого рівня відповідають за вивчення та прогнозування характеристик цілей меншого масштабу, а шари

ознак вищого рівня відповідають за вивчення та прогнозування особливостей цілей великого масштабу. Ефективне їх поєднання слугує для ефективного підвищення точності позиціонування. Враховуючи вплив масштабу ознак кожного шару, розглядається лише трирівневе злиття ознак. На рис. 2.9 показано структуру моделі SSD на основі багаторівневого об'єднання функцій.

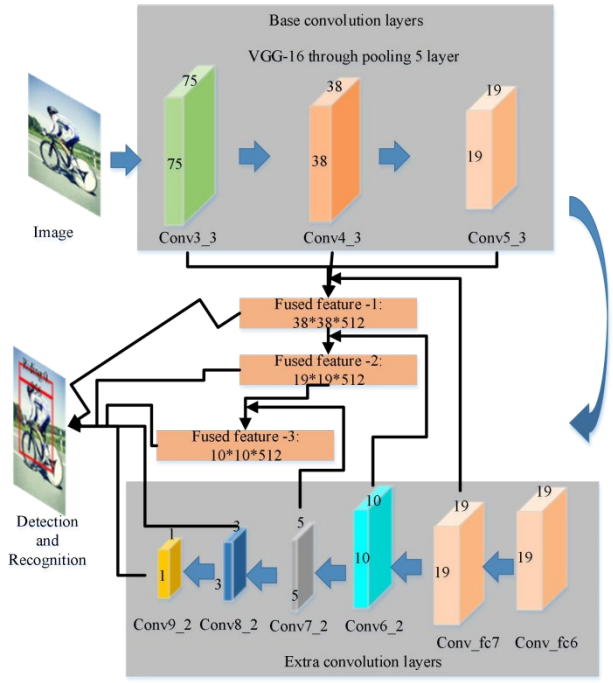


Рис. 2.9. Структура моделі SSD із багаторівневим об'єднанням функцій

Для того, щоб ефективно поєднати витягнуті функції, чутливі до позиції, і контекстну інформацію про функції з цільовими семантичними ознаками, у роботі [93] застосовано метод додавання відповідної функції за пікселем у багатомасштабному об'єкті. На рис. 2.10 показано, що дві різні функції об'єднані попіксельно. Щоб досягти інтеграції двох різних функцій, необхідно переконатися, що розміри та розміри двох функцій залишаються незмінними перед об'єднанням. Під час процесу злиття відповідні позиційні елементи різних характеристик безпосередньо додаються, як показано в рівнянні (2.8).

$$z_j^i = x_j^i + y_j^i \tag{2.8}$$

Розмір і розмір об’єкта після об’єднання відповідають розміру об’єкта до об’єднання.

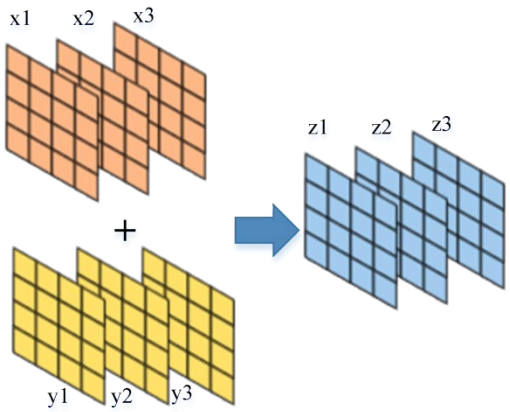


Рис.2.10. Метод об’єднання ознак на основі попіксельного додавання

У моделі злиття функцій SSD детальні характеристики нижнього рівня та контекстна інформація верхнього рівня додаються до цільового рівня прогнозування, а семантичні характеристики цільового рівня прогнозування все ще домінують над функціями злиття. За допомогою методу об’єднання ознак із додаванням відповідного елемента забезпечується цілісність інформації об’єднаних ознак, щоб цільова семантична ознака могла ефективно вибирати та об’єднувати детальну ознаку та контекстну інформацію. Сам цільовий об’єкт забезпечує багатомасштабне злиття об’єктів, таке обмеження робить об’єднані об’єкти більш ефективними для відображення рухомих цілей.

Виходячи з наведеного вище аналізу, об’єднання багатомасштабних функцій за допомогою попіксельної відповідності може ефективно підтримувати передачу багатомасштабної інформації та ефективно поєднувати позиційно-чутливу інформацію, надану детальними функціями, з контекстною інформацією, що надається за семантичними ознаками високого рівня.

У традиційних мережах класифікації та розпізнавання, таких як VGG-16, ResNet-101 тощо, глибина мережі постійно збільшується для виділення

більш абстрактних цільових ознак, тим самим підвищуючи точність розпізнавання мережі. GoogleNet, з іншого боку, застосував інший підхід, використовуючи початкову структуру Inception для збільшення ширини мережі та виділення цільових функцій через каскадування. Початкова структура Inception, яка використовується в GoogleNet, показана на рис. 2.11.

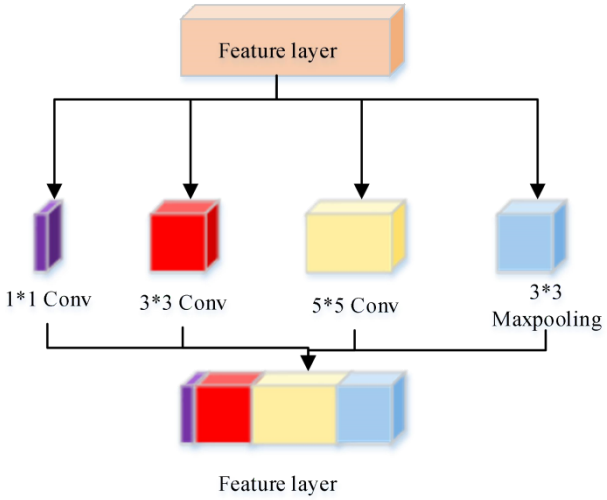


Рис. 2.11. Діаграма структури Inception в мережі GoogleNet

Використання початкової структури для отримання інформації про цільову функцію може значно розширити щільність інформації кожного шару, таким чином ефективно роблячи прогнози для цілі. У цьому розділі кваліфікаційної роботи концепція початкової структури в поєднанні з багатомасштабним вивченням цільових ознак використовується для вирішення проблеми недостатньої семантики кожного рівня прогнозування цілі в мережі SSD, тим самим покращуючи точність виявлення цілі та позиціонування та розпізнавання об’єктів.

Додавши покращену структуру Inception до цільового рівня прогнозування мережі SSD із багатомасштабною структурою об’єднання функцій, вона використовується для покращення семантики функцій кожного рівня прогнозування цілі в мережі, тим самим підвищуючи точність виявлення рухомої цілі людини. і позиціонування.

Завдяки багатомасштабній структурі об’єднання функцій у мережі SSD, рівень згортки функцій conv4_3 головним чином відповідає за вивчення дрібномасштабних цільових функцій. Тому структура Inception лише додається до цільового рівня прогнозування conv4_3, а потім витягує інформацію про цільову функцію. Структура моделі SSD зі структурою прогнозування Inception показана на рис. 2.12.

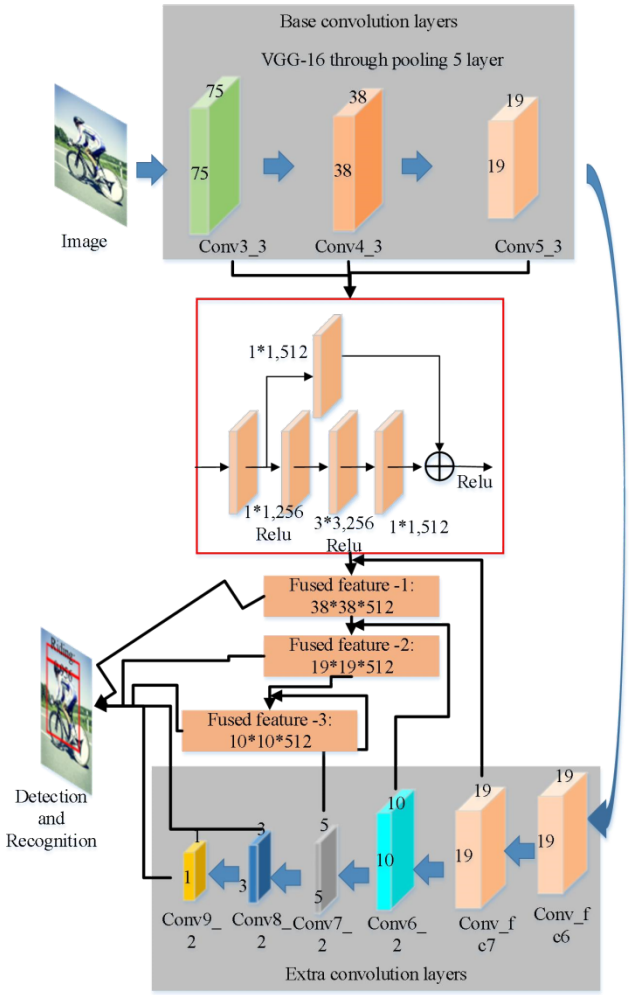


Рис. 2.12. Структура моделі SSD зі структурою прогнозування Inception

Після додавання модуля прогнозування Inception рівня прогнозування conv4_3, він все ще підключений до рівня згортки 3×3 для регресійного прогнозування інформації про позицію та категорію об’єкта.

У процесі навчання використовується функція багатозадачної втрати оригінального SSD. Рівняння (2.9) є функцією втрат, що відповідає запропонованій моделі SSD:

$$L_{x,c,l,g} = \frac{1}{N_{cls}} L_{cls}(x,c) + \frac{\varepsilon}{N_{reg}} L_{loc}(x,l,g) \tag{2.9}$$

У виразі x - це двійковий вектор, складовими елементами якого є $x_{ij}^{p} = \{0,1\}$, який використовується для вказівки того, чи збігається блок-кандидат за умовчанням, помічений *i,* з цільовим дійсним блоком, позначеним під час навчання *j*. Тоді значення x_{ij}^{p} дорівнює 1, що вказує на те, що вікно кандидата за умовчанням є позитивним зразком, інакше значення дорівнює 0, що є негативним зразком. Верхній індекс *p* вказує на категорію цілі, яка відповідає вікну кандидата за замовчуванням. Змінна N_{cls} представляє загальну кількість усіх позитивних і негативних зразків, залучених до розрахунку класифікаційних втрат, а N_{reg} представляє загальну кількість позитивних зразків. Ці два параметри використовуються для нормалізації двох елементів у функції втрат. Змінна α використовується як коефіцієнт балансу для балансування внесків двох членів у функції втрат. Під час навчання функція класифікаційних втрат $L_{cls}(x,c)$ використовує функцію втрат softmax, а її конкретні вирази показані в рівнянні (2.10) і рівнянні (2.11).

$$L_{cls}(x,c) = -\sum_{i \in Pos}^{N} x_{ij}^{p} \log(\hat{c}_i^{p}) - \sum_{i \in Neg} \log(\hat{c}_i^{0}) \tag{2.10}$$

$$\hat{c}_i^{p} = \frac{e^{c_i^{p}}}{\sum_{p} e^{c_i^{p}}} \tag{2.11}$$

Серед них $\hat{c}_i^{p}$ вказує, що мережа оцінює ймовірність кадру-кандидата за замовчуванням у цільовій категорії *Pos*, *Pos* вказує набір міток усіх кадрів-кандидатів за замовчуванням, які відповідають цільовому реальному кадру в навчальному пакеті, а *Neg* вказує, що мітка набір стандартного кадру-кандидата, який не відповідає цільовому істинному кадру. Функція втрат позиційної регресії $L_{loc}\ (x,l,g)$ також використовує плавну функцію втрат L_1,

запропоновану у швидкій RCNN, і її конкретні вирази показано в рівняннях (2.12) – (2.16).

$$L_{loc}(x,l,g) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx,cy,w,h\}} x_{ij}^{k} smooth_{L_1}(l_i^m - \hat{g}_j^m) \tag{2.12}$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx}) / d_i^w \tag{2.13}$$

$$\hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy}) / d_i^h \tag{2.14}$$

$$\hat{g}_j^w = \log(g_j^w / d_i^w) \tag{2.15}$$

$$\hat{g}_j^h = \log(g_j^h / d_i^h) \tag{2.16}$$

Серед них змінна $\hat{g}_j^m$ представляє відносне зміщення позиції між кадром-кандидатом за умовчанням і цільовим істинним кадром у *(x, y, w, h)* чотирьох координатних вимірах. Змінна l_i^m представляє прогнозоване мережею значення інформації про чотиривимірні координати кадру кандидата за замовчуванням. Змінна $\left(d_i^{cx},\ d_i^{cy}\right)$ представляє інформацію про координати центральної позиції стандартного кадру-кандидата. Змінні d_i^w і d_i^h відповідають ширині та висоті поля за замовчуванням *i* відповідно. Змінна x_{ij}^p дорівнює лише 1, коли вікно кандидата за замовчуванням відповідає цільовому вікну істинного значення, що вказує на те, що функція втрат бере участь у обчисленні функції втрат лише тоді, коли поле кандидата за замовчуванням є позитивною вибіркою.

2.4. Висновки за розділом 2

У другому розділі був детально розглянутий алгоритм SSD, який базується на мережі CNN. Він генерує багато зовнішніх кадрів фіксованого розміру та розглядає можливість того, що кожен кадр містить об’єкти, а потім

виконує метод немаксимального придушення для отримання остаточного прогнозованого значення. Після мережі VGG-16 SSD додає шар за шаром карти згорткових функцій зі зменшенням роздільної здатності. Ці карти функцій мають різні сприйнятливі поля, тому SSD може виконувати багатомасштабне виявлення об’єктів, тобто використовувати карти функцій з високою роздільною здатністю для виявлення малих цілей на зображенні, і використовувати карти функцій з низькою роздільною здатністю для виявлення великих цілей. Таким чином, пропонується модель виявлення об’єктів на відео в реальному часі на основі алгоритму SSD. Виявлення об’єктів на основі багатомасштабного об’єднання функцій може покращити точність позиціонування та точність розпізнавання цілей у різних масштабах.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ ОБ’ЄКТІВ У ВІДЕОРЯДІ

3.1. Опис архітектури інформаційної технології розпізнавання об’єктів у відеоряді

Для подальшої роботи за основу було взято архітектуру моделі Single Shot Multibox Detection (SSD). Основним завданням моделі є ефективний пошук та класифікація об’єктів на зображенні з урахуванням масштабу об’єктів. Для роботи з масштабованими об'єктами у даній моделі використовується механізм Anchor Boxes.

Успішним результатом роботи моделі є обведений прямокутником об'єкт інтересу на зображенні з інформацією про те, що це за об'єкт. Приклад результату роботи алгоритму наведено на рис. 3.1.

Рис. 3.1. Результат роботи SSD

Це означає, що модель виконує два завдання - класифікацію і пошук меж об'єкта, що враховується в архітектурі. По суті, модель робить два різних передбачення, кожне з яких має свою функцію втрати. Спрощену архітектуру моделі SSD можна побачити на рис. 3.2.

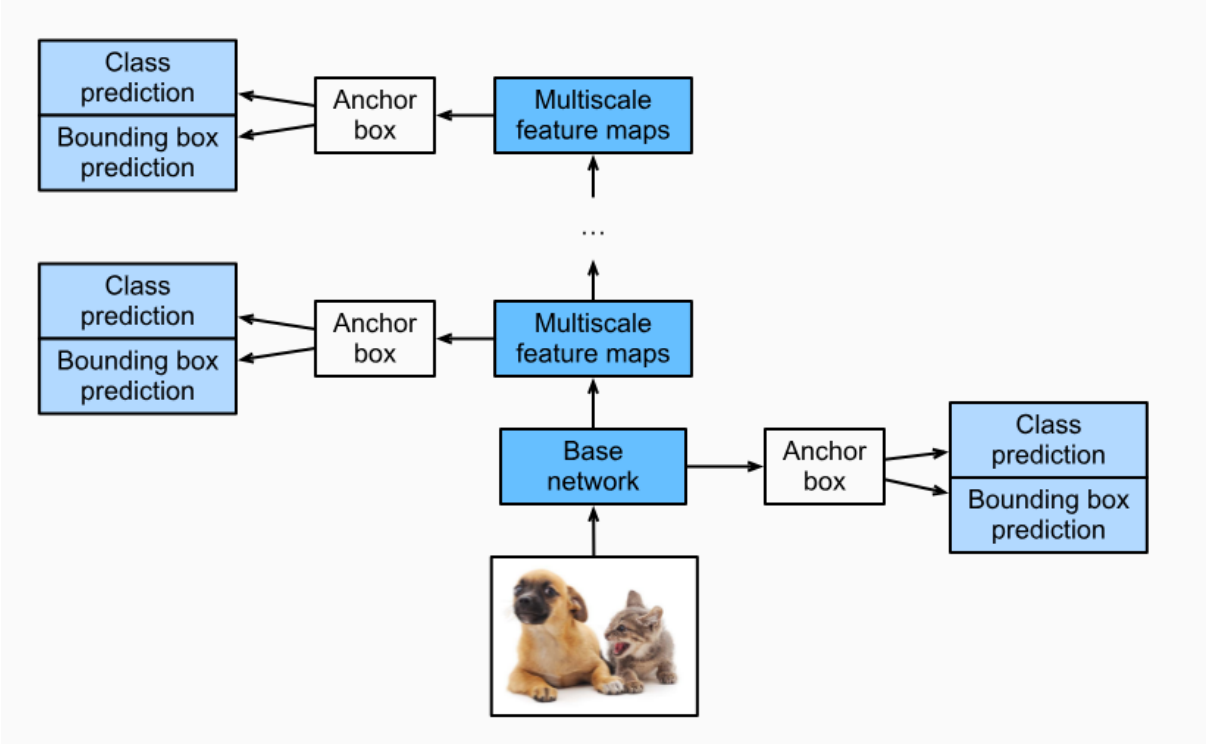


Рис. 3.2. Архітектура моделі SSD

Можна виокремити два основні схожі блоки, з яких складається вся модель: Base Network та Multiscale Feature Maps.

На вході в модель дані передаються в Base Network. Це блок, який найчастіше складається з різних комбінацій згорткових шарів. Звичайною практикою є застосування навчених заздалегідь мереж для виділення особливостей, таких як VGG або ResNet. Оскільки основним завданням блоку є виділення особливостей із зображення для подальшої обробки в системі, не обов'язково навчати цю частину моделі самостійно. Багато компаній надають готові рішення, навчені на великому обсязі якісних даних. В умовах експерименту практично неможливо домогтися аналогічної якості виділення особливостей під час навчання. Виходячи з вище перерахованого, у якості

моделі для проведення експериментів, надалі буде використовуватися навчена заздалегідь модель VGG-16 (рис. 3.3).

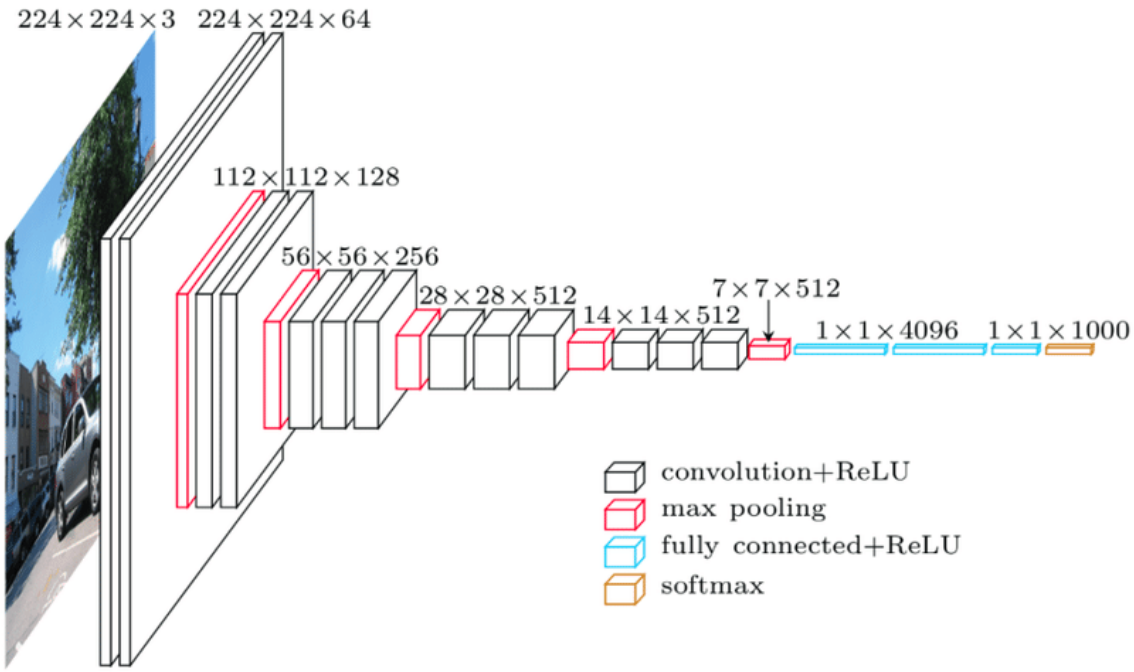


Рис. 3.3. Архітектура блоку VGG-16

Модель буде скорочено, для відповідності з навчальними даними експерименту. Також, для блоку, який був залишений, відключено навчання, щоб не модифікувати готові ваги блоку. Від блоку VGG-16 взяті шари починаючи з першого і закінчуючи шаром max pooling, який приводить дані до розмірності - 28x28x512. Це зроблено для того, щоб було більше можливостей для роботи з масштабуванням зображення.

Виділені блоком VGG-16 особливості вхідного зображення передаються в секцію з блоків Multiscale Feature Maps. Multiscale Feature Maps - серія з послідовних блоків, які слугують для зменшення обчислювального навантаження моделі. Цей блок створює певну кількість рамок прив’язки (прямокутників, які центруються на певному пікселі). Кожен блок Multiscale Feature Maps створює рамки прив’язки певного розміру. У тестованій архітектурі використовується 13 блоків Multiscale Feature Maps. У першому блоці будуть найменші рамки, і відповідно їх буде найбільше. В останньому

блоці будуть найбільші рамки прив’язки, і їх буде найменше. Для зручності роботи з вхідними даними, розміри рамок прив’язки будуть приведені до діапазону від 0 до 1, шляхом ділення їхнього розміру на розмір зображення. Таким чином, у першому блоці розмір рамок прив’язки задано як 0.15. У кожному наступному блоці розмір буде збільшуватися з кроком 0.05. В останньому блоці розмір буде 0.8. У створеній для експериментів моделі, рамки прив’язки створюються таким чином:

Для кожного блоку задаються 4 параметри: кількість центральних пікселів за висотою, кількість центральних пікселів за шириною, відносний розмір рамки прив’язки в діапазоні від 0 до 1 і список зі співвідношень висоти до ширини. Також є 2 додаткові параметри, які не впливають на результат роботи блоку, але потрібні для подальшої роботи програми: розмір рамки і кількість каналів.

3.2. Опис процесу класіфікації об’єктів

Створюється чотиривимірна порожня матриця з такою розмірністю: кількість елементів у кошику на кількість каналів на ширину Feature Map і на висоту Feature Map. Перші два виміри: кількість елементів у кошику та кількість каналів - не впливають на створення рамок прив’язки.

Ґрунтуючись на згенерованій інформації в кожному з блоків, включно з найпершим блоком, що використовує VGG-16, модель робить два різних передбачення. Для цього використовуються шар передбачення класу і шар передбачення меж об'єкта. Оскільки це два різні шари, вони розглядаються окремо і кожен з них має свою функцію втрати. Коли визначається клас об'єкта, тоді треба розглянути вміст зображення всередині кожної з рамок прив’язки та визначити оцінку ймовірності відношення вмісту до кожного з класів. Цей блок можна реалізувати через звичайний повністю пов'язаний шар, однак це вартісна в обчислювальному сенсі операція. В такому випадку,

потрібно буде робити класифікацію для *a*w*h* кількості рамок прив’язки, де *a* - кількість рамок прив’язки, згенерованих для кожного центрального пікселя в Feature Map, *w* - ширина Feature Map, *h* - висота Feature Map.

Це число може бути досить великим, і під час класифікації кожного вмісту рамок прив’язки без використання згорткових шарів призведе до надто великої кількості необхідних параметрів, які потрібно навчати. Для того, щоб зробити класифікацію реальною і доступною, використовується кілька модифікацій. По-перше, замість повністю пов'язаних шарів, використовують згорткові шари, що саме по собі вже кардинально зменшує кількість необхідних параметрів. Однак, при використанні згорткових шарів, виникає наступна проблема: в результаті передбачення треба отримати оцінку класу для кожної рамки прив’язки. У разі, коли використовується повністю зв'язаний шар, для досягнення цієї мети у ньому створюється відповідно кількості класів, кількість нейронів на виході - *a*w*h*(q + 1),* де *q* - кількість класів у моделі. Потрібно помножити на *q + 1* для того, щоб враховувати фон. Таким чином, кожен нейрон представляв би конкретну рамку прив’язки в комбінації з конкретним класом.

Щоб замінити повністю пов'язаний шар на згортковий, треба отримати такий висновок із цього шару, який дозволяє робити припущення про відношення кожної рамки прив’язки до кожного класу. У згорткових шарах є канали. Спочатку, коли зображення тільки передається в модель і потрапляє в перший шар, ці канали є каналами кольорів rgba. Надалі вони змінюють свій сенс і містять у собі інформацію про особливості зображення залежно від фільтра, який застосовується до зображення. Іншими словами, кожен канал має свій власний фільтр, що навчається, який виділяє свої особливості зображення. Можна самостійно обирати, скільки каналів буде на виході з кожного із згорткових шарів. Завдяки цьому, можна домогтися схожої логіки з повністю пов'язаним шаром.

У згортковому шарі з передбаченням, створюється *a*(q + 1)* каналів. Це дає змогу для кожного елемента в позиції *(x,y)* мати оцінку для кожної рамки прив’язки і кожного класу. Завдяки даній модифікації, вдалося позбутися повністю пов'язаного шару і отримати приріст до продуктивності, завдяки зменшенню необхідних для навчання параметрів.

Передбачення меж об'єктів працює за схожим принципом, тільки для кожної рамки прив’язки йому потрібна не інформація про класи, а інформація про різницю між справжніми межами рамки і поточними межами. З цього випливає, що буде точно така сама архітектура цього шару, як і в шару з передбаченням класу, тільки кількість каналів, що створюються, на виході буде *a*4*.

Було визначено 4 параметри для кожної рамки прив’язки, оскільки треба враховувати відступи для кожної зі сторін прямокутника. Як раніше згадувалося, обидва передбачення використовують свої функції втрати, однак немає необхідності в тому, щоб застосовувати функцію втрати до кожного шару передбачення, які є в кожному блоці розробленої моделі. Це 28 місць, де треба було б застосовувати функцію втрати. Це не тільки ускладнило б обчислення, а й ускладнило б навчання через зворотне поширення. Натомість об’єднавши виведення всіх шарів із передбаченням, було отримано всього 2 місця, де треба буде застосувати функцію втрати. Для того, щоб це зробити, треба розв'язувати проблему з різними розмірами вихідних матриць.

Через те, що в кожному блоці Multiscale Feature Maps поступово збільшується масштаб, із кожним блоком зменшується кількість рамок прив’язки, що веде до різниці в розмірності результуючих матриць. Для того, щоб конкатенувати ці матриці, їх спочатку вирівнюють. Перший вимір матриці залишається недоторканим (кількість елементів у кошику), а інші три виміри приводяться до одного. Головна проблема полягала в тому, що на початку виконання операції, матриці були чотиривимірні, причому другий, третій та четвертий виміри були неоднакові, що унеможливлює конкатенацію.

Щоб вирівняти матрицю і привести її до двовимірного вигляду, спочатку змінюється розташування вимірів. Вимірювання з каналами переноситься в кінець, після чого робиться вирівнювання. Застосувавши цю операцію до всіх матриць із передбаченнями, з’являється можливість конкатенувати їх уздовж першого виміру. Результатом конкатенації є дві двовимірні матриці, з передбаченнями класу і передбаченнями меж об'єкта. Ці матриці можна передавати у функції втрати. Для класифікації об'єкта використовується функція кроссентропії (формула 3.1), це звичайна практика для моделей, у яких більше, ніж 2 можливих класи для передбачення.

$$-\sum_{i=1}^{C} y_i \cdot \log(\hat{y}_i) \tag{3.1}$$

Для передбачення меж класу використовується функція абсолютної різниці між передбаченням і правильною інформацією. Для правильної роботи функції, використовується маска, яка відсікає всі негативні рамки прив’язки і рамки прив’язки зі зсувом. Цю функцію втрати наведено нижче:

$$\text{Mask} \cdot \left| \hat{y} - y \right| \tag{3.2}$$

На виході для передбачення класу була отримана матриця такої розмірності: кількість елементів у кошику на кількість передбачених класів. Для передбачення меж об'єкта, у другому вимірі буде в 4 рази більше елементів. Обидві ці матриці усереднюються за першим виміром. Таким чином було отримано два вектори зі значеннями втрати. Однак моделі в нейронних системах використовують скаляр для відстеження функції втрати і для зворотного поширення, і для його обчислення достатньо обчислити суму елементів векторів.

3.3. Вхідні дані

Для експериментів було використано датасет Traffic Signs Detection з онлайн ресурсу kaggle. Цей датасет може вільно використовуватися для некомерційних цілей. Датасет відразу розбитий на три частини:

Тренувальні зображення - зображення, які використовуються для навчання моделі, найбільша частина датасету.

Контрольні зображення - зображення, які використовуються для обчислення функції втрати.

Тестові зображення - зображення, які не використовуються в процесі навчання, вони потрібні для перевірки готової моделі на інформації, яку модель не бачила в процесі навчання.

Усі зображення були приведені до одного розміру - 416x416. Для роботи з експериментальною моделлю, розмір зображень із датасету було приведено до 224*224*3 для відповідності вхідному формату блоку VGG-16 (з 3 - колірними каналами rgb).

Усього в датасеті 3530 тренувальних зображень, 801 контрольне зображення і 638 тестових зображень. Датасет містить такі класи: Green Light, Red Light, Speed Limit 10, Speed Limit 100, Speed Limit 110, Speed Limit 120, Speed Limit 20, Speed Limit 30, Speed Limit 40, Speed Limit 50, Speed Limit 60, Speed Limit 70, Speed Limit 80, Speed Limit 90, Stop. Усі вони наведені в числовий формат для подальшої роботи. Крім зображень, датасет містить текстові файли формату .txt. У кожному такому файлі міститься 5 чисел в одному рядку, розділених пропуском. Перше число - відповідає класу об'єкта, наступні 4 числа потрібні для визначення меж об'єкта. Внизу наведено приклад зображення з датасету:

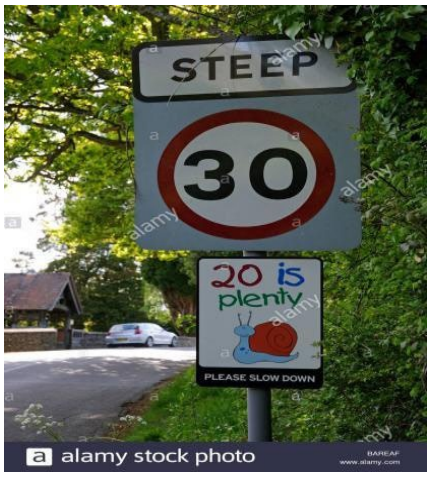


Рис. 3.4. Приклад зображення з датасету Traffic Signs Detection

Змісту текстового файлу для зображення, наведеного на рис. 3.4, виглядає наступним чином: "7 0.5769230769230769 0.3605769230769231 0.41586538461538464 0.27524038461538464".

3.4. Дослідження розробленої моделі

Для проведеного експерименту використовувалася така конфігурація персонального комп'ютера:

- Процесор - i7-7700k
- Відеокарта - RTX 3090
- Оперативна пам'ять - 32 гігабайти ddr4
- Твердотілий накопичувач - SSD Samsung Evo

Навчання відбувалося у віртуальній системі WSL 2 - Ubuntu. WSL було обрано через те, що цей спосіб віртуалізації питомо підтримується в останніх версіях Windows.

Фреймворком для реалізації моделі було обрано фреймворк pytorch мовою програмування Python. Python - одна з найпопулярніших мов програмування для розв'язання задач штучного інтелекту. Вона має велику кількість бібліотек, фреймворків та інструментів для роботи зі штучним інтелектом, як-от: numpy, pandas, matplotlib, tensorflow, pytorch, mxnet або jax. Як було зазначено вище, основним фреймворком було обрано pytorch. Це

фреймворк для штучного інтелекту, який володіє всім необхідним функціоналом для створення моделі запропонованої в експерименті. Так само під час експерименту використовуються numpy і matplotlib. numpy використовувався для модифікації та роботи з даними, своєю чергою matplotlib був потрібен для візуалізації процесу навчання - графіки функції втрати, візуалізація відсотка правильних пророкувань на кроці валідації, відображення зображень із накладеними передбаченими межами об'єкта й оцінкою його класу.

Далі буде наведено результати навчання моделі. Одним із найголовніших параметрів під час навчання є функція втрати. На рис. 3.5 зображено графік функції втрати перших 20 епох.

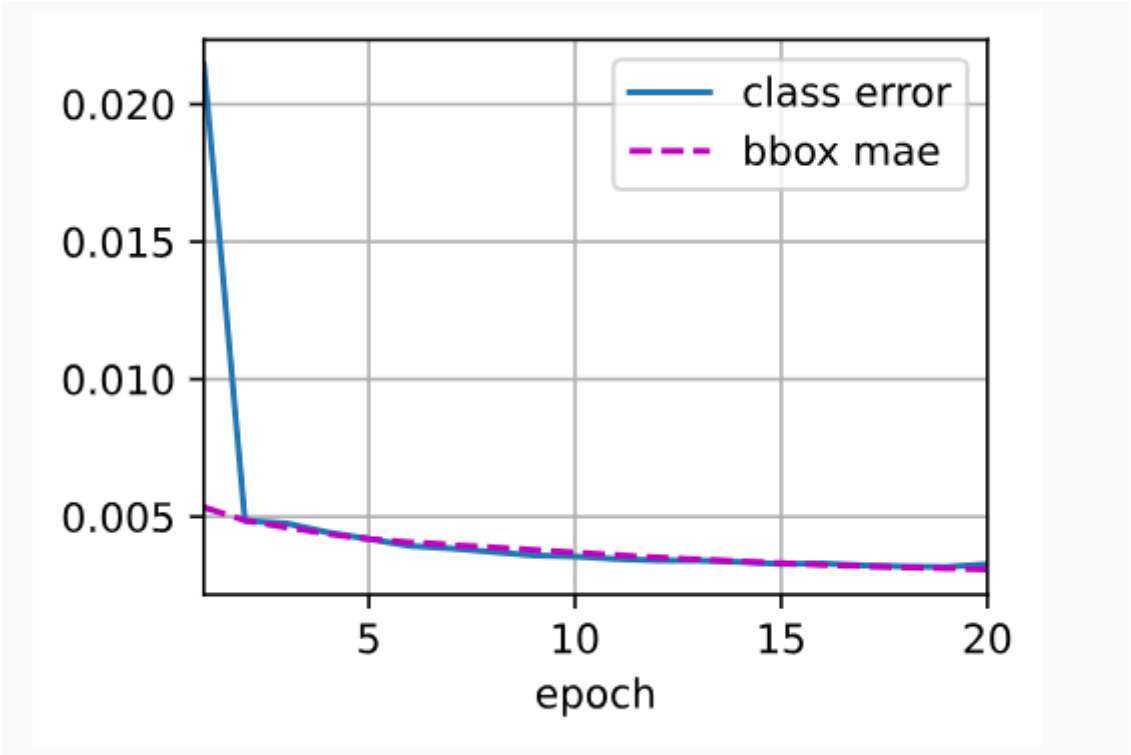


Рис. 3.5. Графік функції втрати перших 20 епох

Такий невеликий діапазон епох було обрано через те, що найбільший вплив на функцію втрати відбувається в перших епохах, після чого зниження результату функції сильно зменшується. Виходячи з графіка, видно, що функція досить швидко набуває маленьких значень.

Це також означає, що датасет можна поліпшити надалі, шляхом додавання більшої кількості даних або застосуванням до нього аугментації. Теоретично це має допомогти поліпшити точність передбачення, оскільки швидке зменшення функції втрати сигналізує про можливе перенавчання. Було проведено порівняння моделі за продуктивністю та якістю передбачення. Для наочності, модель порівнювалася з наявними аналогами: Faster R-CNN, Faster YOLO, YOLO(VGG16). Кожна модель обробляла відео, розбите на кадри. Таким чином, крім точності, з’являється можливість протестувати швидкість роботи алгоритму. В табл. 3.1 наведені результати роботи моделей:

Таблиця 3.1.

Порівняння результатів роботи алгоритмів розпізнавання

Модель	mAP	FPS	Boxes	Вхідна роздільна здатність
Faster R-CNN	73.2	7	~6000	1000 x 600
Faster YOLO	53.7	155	98	448 x 448
YOLO (VGG16)	66.4	21	98	448 x 448
SSD	73.5	29	8732	224 x 224

Значення стовпчиків таблиці:

Модель - використовувана архітектура, створена під час експерименту модель - SSD

mAp - середня помилка роботи, показує, наскільки точно модель визначає клас і межі об'єкта

FPS - кількість оброблюваних зображень на секунду

Boxes - скільки об'єктів змогла розпізнати модель

Вхідна роздільна здатність - розмір вхідного зображення. Чим вона вища, тим складніше обробляти зображення, але водночас воно міститиме більше деталей.

Виходячи з табл. 3.1, можна зробити висновок, що попри низьку роздільну здатність зображення, що тягне за собою втрату інформації, модель, яка була протестована, успішно знаходить велику кількість об'єктів із доволі високою точністю - 73.5. Помітно, що модель відносно повільно працює. Наразі вдалося опрацьовувати відеопотік зі швидкістю 29 кадрів на секунду. Проаналізувавши приклади інших наявних моделей Single Shot Multibox Detection, видно, що 29 кадрів - результат нижчий за середній. До такого результату призвела більша кількість блоків multiscale feature maps. У моделі було додано більше даних блоків, ніж зазвичай використовується в спробі отримати якісніші передбачення. Однак виходячи з тестів, ця зміна не призвела до значного поліпшення точності, проте помітно зменшила швидкість передбачення.

3.5. Висновки за розділом 3

В рамках запропонованої інформаційної технології розпізнавання об’єктів, для виконання класифікації з масштабуванням над кадрами у відеопотоці було розроблено програмне забезпечення на основі архітектури Single Shot Multibox Detection, яка здатна одночасно знаходити та визначати кілька об’єктів. Задля зменшення кількості обчислень були внесені модифікації у вихідний шар нейронної мережі.

В результаті спроби покращити точність розпізнавання шляхом додавання додаткових шарів multiscale feature maps було з’ясовано, що суттєвого впливу на точність не відбулося, проте швидкодія суттєво впала. Таким чином, для практичного застосування варто використовувати зазначену модель зі стандартною кількістю multiscale feature maps та перебудованим вихідним шаром нейронів.

ВИСНОВКИ

У даній роботі була запропонована інформаційна технологія розпізнавання об’єктів в відеоряді. Були розглянуті аналітична і програмна складові запропонованої технології.

Основні принципи побудови запропонованої інформаційної технології розпізнавання об’єктів в відеоряді:

- використання у якості основи архітектуру Single Shot Multibox Detection, з мережею VGG-16 у якості базової мережі;
- модифікація вхідного шару нейронної мережі;
- об’єднання виведення всіх шарів із передбаченням, задля скорочення кількості застосувань функції втрати;
- вирівнювання розмірності результуючих матриць кожного блоку Multiscale Feature Maps.

Наукова новизна отриманих результатів кваліфікаційної роботи визначається тим, що вперше обґрунтовано архітектуру інформаційної технології розпізнавання об’єктів у відеоряді, що підтримує адаптацию класифікатора до масштабування зображення. Завдяки виконаним модифікаціям алгоритму, вдалося отримати приріст до продуктивності моделі, завдяки зменшенню необхідних для навчання параметрів.

Практична цінність результатів полягає в тому, що запропонована в роботі інформаційна технологія дозволяє виконувати розпізнавання об’єктів в відеоряді в режимі реального часу.

Незважаючи на наявний значний прогрес у виявленні об’єктів, більшість детекторів все ще далекі від оптимального співвідношення показників точності та швидкодії. Існує потреба в компактних алгоритмах розпізнавання об’єктів на відео, які можна використовувати на мобільних і вбудованих пристроях. Розгляд цього питання є актуальним напрямком для подальших досліджень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Z. Zou, K. Chen, Z. Shi, Y. Guo and J. Ye, "Object Detection in 20 Years: A Survey," in Proceedings of the IEEE, vol. 111, no. 3, pp. 257-276, March 2023, doi: 10.1109/JPROC.2023.3238524.

2. J. Dai, K. He, and J. Sun, “Instance-aware semantic segmentation via multi-task network cascades,” in CVPR, 2016, pp. 3150–3158.

3. K. He, G. Gkioxari, P. Dollar, and R. Girshick, “Mask´ r-cnn,” in ICCV. IEEE, 2017, pp. 2980–2988.

4. A. Karpathy and L. Fei-Fei, “Deep visual-semantic alignments for generating image descriptions,” in CVPR, 2015, pp. 3128–3137.

5. Q. Wu, C. Shen, P. Wang, A. Dick, and A. van den Hengel, “Image captioning and visual question answering based on attributes and external knowledge,” IEEE transactions on pattern analysis and machine intelligence, vol. 40, no. 6, pp. 1367–1381, 2018.

6. K. Kang, H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, X. Wang et al., “T-cnn: Tubelets with convolutional neural networks for object detection from videos,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 28, no. 10, pp. 2896–2907, 2018.

7. P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in CVPR, vol. 1. IEEE, 2001, pp. I–I.

8. P. Viola and M. J. Jones, “Robust real-time face detection,” International journal of computer vision, vol. 57, no. 2, pp. 137–154, 2004.

9. N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in CVPR, vol. 1. IEEE, 2005, pp. 886–893.

10. P. Felzenszwalb, D. McAllester, and D. Ramanan, “A discriminatively trained, multiscale, deformable part model,” in CVPR. IEEE, 2008, pp. 1–8.

11. P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, “Object detection with discriminatively trained part-based models,” IEEE

transactions on pattern analysis and machine intelligence, vol. 32, no. 9, pp. 1627–1645, 2010.

12. R. B. Girshick, From rigid templates to grammars: Object detection with structured models. Citeseer, 2012.

13. A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Advances in neural information processing systems, 2012, pp. 1097–1105.

14. R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Region-based convolutional networks for accurate object detection and segmentation,” IEEE transactions on pattern analysis and machine intelligence, vol. 38, no. 1, pp. 142–158, 2016.

15. J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, “Selective search for object recognition,” International journal of computer vision, vol. 104, no. 2, pp. 154–171, 2013.

16. K. He, X. Zhang, S. Ren, and J. Sun, “Spatial pyramid pooling in deep convolutional networks for visual recognition,” in ECCV. Springer, 2014, pp. 346–361.

17. R. Girshick, “Fast r-cnn,” in ICCV, 2015, pp. 1440– 1448.

18. S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: towards real-time object detection with region proposal networks,” IEEE Transactions on Pattern Analysis & Machine Intelligence, no. 6, pp. 1137–1149, 2017.

19. J. Dai, Y. Li, K. He, and J. Sun, “R-fcn: Object detection via region-based fully convolutional networks,” in Advances in neural information processing systems, 2016, pp. 379–387.

20. Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, and J. Sun, “Light-head r-cnn: In defense of two-stage object detector,” arXiv preprint arXiv:1711.07264, 2017.

21. T.-Y. Lin, P. Dollar, R. B. Girshick, K. He, B. Hariharan,´ and S. J. Belongie, “Feature pyramid networks for object detection.” in CVPR, vol. 1, no. 2, 2017, p. 4.

22. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in CVPR, 2016, pp. 779–788.

23. A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “Yolov4: Optimal speed and accuracy of object detection,” arXiv preprint arXiv:2004.10934, 2020.

24. C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, “Yolov7: Trainable bag-of-freebies sets new state-ofthe-art for real-time object detectors,” arXiv preprint arXiv:2207.02696, 2022.

25. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in ECCV. Springer, 2016, pp. 21–37.

26. T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar,´ “Focal loss for dense object detection,” IEEE transactions on pattern analysis and machine intelligence, 2018.

27. H. Law and J. Deng, “Cornernet: Detecting objects as paired keypoints,” in Proceedings of the European conference on computer vision (ECCV), 2018, pp. 734– 750.

28. X. Zhou, D. Wang, and P. Krahenb¨ uhl, “Objects as¨ points,” arXiv preprint arXiv:1904.07850, 2019.

29. N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-end object detection with transformers,” in European Conference on Computer Vision. Springer, 2020, pp. 213–229.

30. X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, “Deformable detr: Deformable transformers for end-toend object detection,” arXiv preprint arXiv:2010.04159, 2020.

31. J. Hosang, R. Benenson, P. Dollar, and B. Schiele,´ “What makes for effective detection proposals?” IEEE transactions on pattern analysis and machine intelligence, vol. 38, no. 4, pp. 814–830, 2016.

32. Z. Yang, S. Liu, H. Hu, L. Wang, and S. Lin, “Reppoints: Point set representation for object detection,” in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 9657–9666.

33. S. Ren, K. He, R. Girshick, and J. Sun, “Faster rcnn: Towards real-time object detection with region proposal networks,” in Advances in neural information processing systems, 2015, pp. 91–99.

34. Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, Lin, and B. Guo, “Swin transformer: Hierarchical vi-sion transformer using shifted windows,” arXiv preprint arXiv:2103.14030, 2021.

35. S. K. Divvala, D. Hoiem, J. H. Hays, A. A. Efros, and M. Hebert, “An empirical study of context in object detection,” in CVPR. IEEE, 2009, pp. 1271–1278.

36. C. Chen, M.-Y. Liu, O. Tuzel, and J. Xiao, “R-cnn for small object detection,” in Asian conference on computer vision. Springer, 2016, pp. 214–230.

37. X. Zeng, W. Ouyang, J. Yan, H. Li, T. Xiao, K. Wang, Y. Liu, Y. Zhou, B. Yang, Z. Wang et al., “Crafting gbdnet for object detection,” IEEE transactions on pattern analysis and machine intelligence, vol. 40, no. 9, pp. 2109–2123, 2018.

38. Y. Li, Y. Chen, N. Wang, and Z. Zhang, “Scale-aware trident networks for object detection,” arXiv preprint arXiv:1901.01892, 2019.

39. J. Li, Y. Wei, X. Liang, J. Dong, T. Xu, J. Feng, and Yan, “Attentive contexts for object detection,” IEEE Transactions on Multimedia, vol. 19, no. 5, pp. 944– 954, 2017.

40. L. V. Pato, R. Negrinho, and P. M. Q. Aguiar, “Seeing without looking: Contextual rescoring of object detections for ap maximization,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020.

41. Y. Liu, R. Wang, S. Shan, and X. Chen, “Structure inference net: Object detection using scene-level context and instance-level relationships,” in CVPR, 2018, pp. 6985–6994.

42. S. Zhang, L. Wen, X. Bian, Z. Lei, and S. Z. Li, “Singleshot refinement neural network for object detection,” in CVPR, 2018.

43. R. Rothe, M. Guillaumin, and L. Van Gool, “Nonmaximum suppression for object detection by passing messages between windows,” in Asian Conference on Computer Vision. Springer, 2014, pp. 290–306.

44. Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, “Distance-iou loss: Faster and better learning for bounding box regression,” in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 07, 2020, pp. 12993–13000.

45. Y. He, C. Zhu, J. Wang, M. Savvides, and X. Zhang, “Bounding box regression with uncertainty for accurate object detection,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 2888–2897.

46. R. Solovyev, W. Wang, and T. Gabruseva, “Weighted boxes fusion: Ensembling boxes from different object detection models,” Image and Vision

47. P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, “Overfeat: Integrated recognition, localization and detection using convolutional networks,” arXiv preprint arXiv:1312.6229, 2013.

48. H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei, “Relation networks for object detection,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 3588–3597.

49. Z. Tan, X. Nie, Q. Qian, N. Li, and H. Li, “Learning to rank proposals for object detection,” in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 8273–8281.

50. K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, “Centernet: Keypoint triplets for object detection,” in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 6569–6578.

51. X. Zhou, J. Zhuo, and P. Krahenbuhl, “Bottom-up object detection by grouping extreme and center points,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 850–859.

52. H. Zhu, X. Chen, W. Dai, K. Fu, Q. Ye, and J. Jiao, “Orientation robust object detection in aerial images using deep convolutional neural network,” in ICIP. IEEE, 2015, pp. 3735–3739.

53. B. Cai, Z. Jiang, H. Zhang, Y. Yao, and S. Nie, “Online exemplar-based fully convolutional network for aircraft detection in remote sensing images,” IEEE Geoscience and Remote Sensing Letters, no. 99, pp. 1–5, 2018.

54. G. Cheng, J. Han, P. Zhou, and D. Xu, “Learning rotation-invariant and fisher discriminative convolutional neural networks for object detection,” IEEE Transactions on Image Processing, vol. 28, no. 1, pp. 265–278, 2018.

55. J. Ding, N. Xue, Y. Long, G.-S. Xia, and Q. Lu, “Learning roi transformer for oriented object detection in aerial images,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 2849–2858.

56. B. Singh and L. S. Davis, “An analysis of scale invariance in object detection–snip,” in CVPR, 2018, pp. 3578–3587.

57. M. Gao, R. Yu, A. Li, V. I. Morariu, and L. S. Davis, “Dynamic zoom-in network for fast object detection in large images,” in CVPR, 2018.

Y. Lu, T. Javidi, and S. Lazebnik, “Adaptive object detection using adjacency and zoom prediction,” in CVPR, 2016, pp. 2351–2359.

58. S. Qiao, W. Shen, W. Qiu, C. Liu, and A. L. Yuille, “Scalenet: Guiding object proposal generation in supermarkets and beyond.” in ICCV, 2017, pp. 1809–1818.

59. Z. Cai and N. Vasconcelos, “Cascade r-cnn: Delving into high quality object detection,” in CVPR, vol. 1, no. 2, 2018, p. 10.

60. S. Gidaris and N. Komodakis, “Locnet: Improving localization accuracy for object detection,” in CVPR, 2016, pp. 789–798.

61. S. Brahmbhatt, H. I. Christensen, and J. Hays, “Stuffnet: Using ‘stuff’to improve object detection,” in Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on. IEEE, 2017, pp. 934–943.

62. X. Wu, D. Sahoo, S. C.H. Hoi, Recent advances in deep learning for object detection, Neurocomputing, Vol. 396, 2020, pp. 39-64, ISSN 0925-2312

63. R. Girshick, Fast r-cnn, in: ICCV, 2015.

64. T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollar, Focal loss for dense ´ object detection, in: ICCV, 2017

65. Y. Liu, P. Sun, N. Wergeles, and Y. Shang. A survey and performance evaluation of deep learning methods for small object detection. Expert Systems with Applications, 172:114602, 2021. ISSN 0957-4174.

66. I. Goodfellow, Y. Bengio, and A. Courville. Deep Learning. MIT Press, 2016.

67. K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. CoRR, abs/1409.1556, 2014.

68. Large Scale Visual Recognition Challenge 2014 (ILSVRC2014). URL http://www.image-net.org/challenges/LSVRC/2014/

69. E. Arani et al. “A Comprehensive Study of Real-Time Object Detection Networks Across Multiple Domains: A Survey.” Trans. Mach. Learn. Res, 2022.

70. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779–788, 2016.

71. J. Redmon and A. Farhadi. Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767, 2018.

72. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, Ch.-Y. Fu, and A. C Berg. Ssd: Single shot multibox detector. In European conference on computer vision, pp. 21–37. Springer, 2016.

73. T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017a.

74. H. Law and J. Deng. Cornernet: Detecting objects as paired keypoints. In The European Conference on Computer Vision (ECCV), September 2018.

75. X. Zhou, D. Wang, and Ph. Krähenbühl. Objects as points. CoRR, abs/1904.07850, 2019a. URL http://arxiv.org/abs/1904.07850.

76. Zh. Tian, Ch. Shen, H. Chen, and T. He. Fcos: Fully convolutional one-stage object detection. In The IEEE International Conference on Computer Vision (ICCV), October 2019.

77. R. Lyu. Super fast and lightweight anchor-free object detection model. real-time on mobile devices. 2020. URL https://github.com/RangiLyu/nanodet.

78. Z. Liu, T. Zheng, G. Xu, Zh. Yang, H. Liu, and D. Cai. Training-time-friendly network for real-time object detection. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pp. 11685–11692, 2020.

79. N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. arXiv preprint arXiv:2005.12872, 2020.

80. R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 580–587, 2014.

81. R. Girshick. Fast K-cnn. In The IEEE International Conference on Computer Vision (ICCV), December 2015.

82. Sh. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (eds.), Advances in Neural Information Processing Systems 28, pp. 91–99. Curran Associates, Inc., 2015.

83. K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask r-cnn. In The IEEE International Conference on Computer Vision (ICCV), Oct 2017.

84. K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 37(9):1904–1916, Sep. 2015. ISSN 0162-8828. doi: 10.1109/TPAMI.2015.2389824.

85. L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen. Deep learning for generic object detection: A survey. arXiv preprint arXiv:1809.02165, 2018a.

86. Zh.-Q. Zhao, P. Zheng, Sh.-T. Xu, and X. Wu. Object detection with deep learning: A review. arXiv preprint arXiv:1807.05511, 2018b.

87. Zh. Zou, Zh. Shi, Y. Guo, and J. Ye. Object detection in 20 years: A survey. arXiv preprint arXiv:1905.05055, 2019.

88. T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Oct 2017b.

89. J. Anand & M. Dr. Divyakant. A Comparative Study of Various Object Detection Algorithms and Performance Analysis. International Journal Of Computer Sciences And Engineering. 8. 158-163, 2020. 10.26438/ijcse/v8i10.158163.

90. K. He, X. Zhang, Sh. Ren, and J. Sun. Deep residual learning for image recognition. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.

91. Z. Shen, Z. Liu, J. Li, Y. Jiang, Y. Chen, X. Xue, “DSOD: Learning Deeply Supervised Object Detectors from Scratch”, IEEE International Conference on Computer Vision (ICCV), Venice, Italy, pp.1937-1945, 2017.

92. S. Zhai, D. Shang, S. Wang, S. Dong, “DF-SSD: An Improved SSD Object Detection Algorithm Based on DenseNet and Feature Fusion”, IEEE Access, Vol.8, pp.24344-24357, 2020.

93. M. Gong and Y. Shu, "Real-Time Detection and Motion Recognition of Human Moving Objects Based on Deep Learning and Multi-Scale Feature Fusion in Video," in IEEE Access, vol. 8, pp. 25811-25822, 2020, doi: 10.1109/ACCESS.2020.2971283.

94. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання : (ГОСТ 7.1-2003, IDT) : ДСТУ ГОСТ 7.1:2006. – Чинний з 2007–07–01. – К. : Держспоживстандарт України, 2007. – 47 с. – (Система

стандартів з інформації, бібліотечної та видавничої справи) (Національний стандарт України).

95. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення : ДСТУ 3008-95. – Чинний від 1996–01–01. – К. : Держстандарт України, 1996. – 39 с.

96. ДСТУ 2394-94 Інформація та документація. Комплектування фонду, бібліографічний опис, аналіз документів. Терміни та визначення. – Чинний від 01.01.1995. — Київ : Держстандарт України, 1994. – 88 с.

97. Скорочення слів в українській мові у бібліографічному описі. Загальні правила та вимоги : ДСТУ 3582-97. – Чинний від 1998–07–01. – К. : Держстандарт України, 1998. – 24 с. – (Державний стандарт України).

98. Атестація здобувачів вищої освіти. Методичні рекомендації до виконання кваліфікаційної роботи магістра студентами галузі знань 12 Інформаційні технології спеціальності 126 Інформаційні системи та технології / Г.М. Коротенко, К.Л. Сергєєва; М-во освіти і науки України, Нац. техн. ун-т «Дніпровська політехніка». – Дніпро: НТУ «ДП», 2020. – 48 с.

ДОДАТОК А

Відомість матеріалів кваліфікаційної роботи

		Позначення	Найменування	Кільк. аркушів	Примітка
1					
2			Документація		
3					
4		**ІСТ.КР 23.03.ДА.ПЗ**	Пояснювальна записка	95	
5					
6			Презентація	17	
7					
8			Диск CD с презентацией	1	

Зм.	Ар-куш	№ докум	Підпис	Дата	**ІСТ.КР 23.03.ДА.ПЗ**			
Розроб.		С.П. Дьяков			**Матеріали кваліфіка-ційної роботи**	Літ.	Аркуш	Аркушів
						Н	1	1
Керівник		Г.М. Коротенко				**НТУ «ДП», 12; 126м-22-1**		
Рецензент		А.Л. Ширін						
Н.контр.		Г.М. Коротенко						
Зав.каф.		В.В. Гнатушенко						

ДОДАТОК Б

Програмний код застосунку

main.py

```
import torch
import torchvision
from torch import nn
from torch.nn import functional as F
from matplotlib_inline import backend_inline
from IPython import display

# help functions for plots

def use_svg_display():
    """Use the svg format to display a plot in Jupyter."""
    backend_inline.set_matplotlib_formats('svg')

def set_figsize(figsize=(3.5, 2.5)):
    """Set the figure size for matplotlib."""
    use_svg_display()
    plt.rcParams['figure.figsize'] = figsize

def set_axes(axes, xlabel, ylabel, xlim, ylim, xscale, yscale, legend):
    """Set the axes for matplotlib."""
    axes.set_xlabel(xlabel), axes.set_ylabel(ylabel)
    axes.set_xscale(xscale), axes.set_yscale(yscale)
    axes.set_xlim(xlim),     axes.set_ylim(ylim)
    if legend:
        axes.legend(legend)
    axes.grid()

def plot(X, Y=None, xlabel=None, ylabel=None, legend=[], xlim=None,
         ylim=None, xscale='linear', yscale='linear',
         fmts=('-', 'm--', 'g-.', 'r:'), figsize=(3.5, 2.5), axes=None):
    """Plot data points."""

    def has_one_axis(X):  # True if X (tensor or list) has 1 axis
        return (hasattr(X, "ndim") and X.ndim == 1 or isinstance(X, list)
                and not hasattr(X[0], "__len__"))
```

```
    if has_one_axis(X): X = [X]
    if Y is None:
        X, Y = [[]] * len(X), X
    elif has_one_axis(Y):
        Y = [Y]
    if len(X) != len(Y):
        X = X * len(Y)

    set_figsize(figsize)
    if axes is None:
        axes = plt.gca()
    axes.cla()
    for x, y, fmt in zip(X, Y, fmts):
        axes.plot(x,y,fmt) if len(x) else axes.plot(y,fmt)
    set_axes(axes, xlabel, ylabel, xlim, ylim, xscale, yscale, legend)



class ProgressBoard():
    """The board that plots data points in animation."""
    def __init__(self, xlabel=None, ylabel=None, xlim=None,
                 ylim=None, xscale='linear', yscale='linear',
                 ls=['-', '--', '-.', ':'], colors=['C0', 'C1', 'C2', 'C3'],
                 fig=None, axes=None, figsize=(3.5, 2.5), display=True):
        self.save_hyperparameters()

    def draw(self, x, y, label, every_n=1):
        raise NotImplemented


    def draw(self, x, y, label, every_n=1):
        """Defined in :numref:`sec_utils`"""
        Point = collections.namedtuple('Point', ['x', 'y'])
        if not hasattr(self, 'raw_points'):
            self.raw_points = collections.OrderedDict()
            self.data = collections.OrderedDict()
        if label not in self.raw_points:
            self.raw_points[label] = []
            self.data[label] = []
        points = self.raw_points[label]
        line = self.data[label]
```

```
        points.append(Point(x, y))
        if len(points) != every_n:
            return
        mean = lambda x: sum(x) / len(x)
        line.append(Point(mean([p.x for p in points]),
                          mean([p.y for p in points])))
        points.clear()
        if not self.display:
            return
        use_svg_display()
        if self.fig is None:
            self.fig = plt.figure(figsize=self.figsize)
        plt_lines, labels = [], []
        for (k, v), ls, color in zip(self.data.items(), self.ls, self.colors):
            plt_lines.append(plt.plot([p.x for p in v], [p.y for p in v],
                                      linestyle=ls, color=color)[0])
            labels.append(k)
        axes = self.axes if self.axes else plt.gca()
        if self.xlim: axes.set_xlim(self.xlim)
        if self.ylim: axes.set_ylim(self.ylim)
        if not self.xlabel: self.xlabel = self.x
        axes.set_xlabel(self.xlabel)
        axes.set_ylabel(self.ylabel)
        axes.set_xscale(self.xscale)
        axes.set_yscale(self.yscale)
        axes.legend(plt_lines, labels)
        display.display(self.fig)
        display.clear_output(wait=True)


# Block of help functions for work with anchor boxes
# and help layer functions


def boxIou(boxes1, boxes2):
    """Compute pairwise IoU across two lists of anchor or bounding boxes.

    Defined in :numref:`sec_anchor`"""
    box_area = lambda boxes: ((boxes[:, 2] - boxes[:, 0]) *
                              (boxes[:, 3] - boxes[:, 1]))
    # Shape of `boxes1`, `boxes2`, `areas1`, `areas2`: (no. of boxes1, 4),
    # (no. of boxes2, 4), (no. of boxes1,), (no. of boxes2,)
```

```
    areas1 = box_area(boxes1)
    areas2 = box_area(boxes2)
    # Shape of `inter_upperlefts`, `inter_lowerrights`, `inters`: (no. of
    # boxes1, no. of boxes2, 2)
    inter_upperlefts = torch.max(boxes1[:, None, :2], boxes2[:, :2])
    inter_lowerrights = torch.min(boxes1[:, None, 2:], boxes2[:, 2:])
    inters = (inter_lowerrights - inter_upperlefts).clamp(min=0)
    # Shape of `inter_areas` and `union_areas`: (no. of boxes1, no. of boxes2)
    inter_areas = inters[:, :, 0] * inters[:, :, 1]
    union_areas = areas1[:, None] + areas2 - inter_areas
    return inter_areas / union_areas

def offsetBoxes(anchors, assigned_bb, eps=1e-6):
    """Transform for anchor box offsets."""
    c_anc = box_corner_to_center(anchors)
    c_assigned_bb = box_corner_to_center(assigned_bb)
    offset_xy = 10 * (c_assigned_bb[:, :2] - c_anc[:, :2]) / c_anc[:, 2:]
    offset_wh = 5 * log(eps + c_assigned_bb[:, 2:] / c_anc[:, 2:])
    offset = concat([offset_xy, offset_wh], axis=1)
    return offset

def assignAnchorToBbox(ground_truth, anchors, device, iou_threshold=0.5):
    """Assign closest ground-truth bounding boxes to anchor boxes."""
    num_anchors, num_gt_boxes = anchors.shape[0], ground_truth.shape[0]
    # Element x_ij in the i-th row and j-th column is the IoU of the anchor
    # box i and the ground-truth bounding box j
    jaccard = boxIou(anchors, ground_truth)
    # Initialize the tensor to hold the assigned ground-truth bounding box for
    # each anchor
    anchors_bbox_map = torch.full((num_anchors,), -1, dtype=torch.long,
                                  device=device)
    # Assign ground-truth bounding boxes according to the threshold
    max_ious, indices = torch.max(jaccard, dim=1)
    anc_i = torch.nonzero(max_ious >= iou_threshold).reshape(-1)
    box_j = indices[max_ious >= iou_threshold]
    anchors_bbox_map[anc_i] = box_j
    col_discard = torch.full((num_anchors,), -1)
    row_discard = torch.full((num_gt_boxes,), -1)
    for _ in range(num_gt_boxes):
        max_idx = torch.argmax(jaccard)  # Find the largest IoU
```

```
        box_idx = (max_idx % num_gt_boxes).long()
        anc_idx = (max_idx / num_gt_boxes).long()
        anchors_bbox_map[anc_idx] = box_idx
        jaccard[:, box_idx] = col_discard
        jaccard[anc_idx, :] = row_discard
    return anchors_bbox_map


def multiboxTarget(anchors, labels):
    """Label anchor boxes using ground-truth bounding boxes.

    Defined in :numref:`subsec_labeling-anchor-boxes`"""
    batch_size, anchors = labels.shape[0], anchors.squeeze(0)
    batch_offset, batch_mask, batch_class_labels = [], [], []
    device, num_anchors = anchors.device, anchors.shape[0]
    for i in range(batch_size):
        label = labels[i, :, :]
        anchors_bbox_map = assignAnchorToBbox(
            label[:, 1:], anchors, device)
        bbox_mask = ((anchors_bbox_map >= 0).float().unsqueeze(-1)).repeat(
            1, 4)
        # Initialize class labels and assigned bounding box coordinates with
        # zeros
        class_labels = torch.zeros(num_anchors, dtype=torch.long,
                                   device=device)
        assigned_bb = torch.zeros((num_anchors, 4), dtype=torch.float32,
                                  device=device)
        # Label classes of anchor boxes using their assigned ground-truth
        # bounding boxes. If an anchor box is not assigned any, we label its
        # class as background (the value remains zero)
        indices_true = torch.nonzero(anchors_bbox_map >= 0)
        bb_idx = anchors_bbox_map[indices_true]
        class_labels[indices_true] = label[bb_idx, 0].long() + 1
        assigned_bb[indices_true] = label[bb_idx, 1:]
        # Offset transformation
        offset = offsetBoxes(anchors, assigned_bb) * bbox_mask
        batch_offset.append(offset.reshape(-1))
        batch_mask.append(bbox_mask.reshape(-1))
        batch_class_labels.append(class_labels)
    bbox_offset = torch.stack(batch_offset)
    bbox_mask = torch.stack(batch_mask)
```

```
    class_labels = torch.stack(batch_class_labels)
    return (bbox_offset, bbox_mask, class_labels)


def multiboxPrior(data, sizes, ratios):
    """Generate anchor boxes with different shapes centered on each pixel."""
    in_height, in_width = data.shape[-2:]
    device, num_sizes, num_ratios = data.device, len(sizes), len(ratios)
    boxes_per_pixel = (num_sizes + num_ratios - 1)
    size_tensor = tensor(sizes, device=device)
    ratio_tensor = tensor(ratios, device=device)
    # Offsets are required to move the anchor to the center of a pixel. Since
    # a pixel has height=1 and width=1, we choose to offset our centers by 0.5
    offset_h, offset_w = 0.5, 0.5
    steps_h = 1.0 / in_height  # Scaled steps in y axis
    steps_w = 1.0 / in_width  # Scaled steps in x axis

    # Generate all center points for the anchor boxes
    center_h = (torch.arange(in_height, device=device) + offset_h) * steps_h
    center_w = (torch.arange(in_width, device=device) + offset_w) * steps_w
    shift_y, shift_x = torch.meshgrid(center_h, center_w, indexing='ij')
    shift_y, shift_x = shift_y.reshape(-1), shift_x.reshape(-1)

    # Generate `boxes_per_pixel` number of heights and widths that are later
    # used to create anchor box corner coordinates (xmin, xmax, ymin, ymax)
    w = torch.cat((size_tensor * torch.sqrt(ratio_tensor[0]),
                   sizes[0] * torch.sqrt(ratio_tensor[1:])))\
                   * in_height / in_width  # Handle rectangular inputs
    h = torch.cat((size_tensor / torch.sqrt(ratio_tensor[0]),
                   sizes[0] / torch.sqrt(ratio_tensor[1:])))
    # Divide by 2 to get half height and half width
    anchor_manipulations = torch.stack((-w, -h, w, h)).T.repeat(
                                    in_height * in_width, 1) / 2

    # Each center point will have `boxes_per_pixel` number of anchor boxes, so
    # generate a grid of all anchor box centers with `boxes_per_pixel` repeats
    out_grid = torch.stack([shift_x, shift_y, shift_x, shift_y],
               dim=1).repeat_interleave(boxes_per_pixel, dim=0)
    output = out_grid + anchor_manipulations
    return output.unsqueeze(0)
```

```
# trainer for model code
class Trainer():
    """The base class for training models with data.

    Defined in :numref:`subsec_oo-design-models`"""
    def __init__(self, max_epochs, num_gpus=0, gradient_clip_val=0):
        self.save_hyperparameters()
        assert num_gpus == 0, 'No GPU support yet'

    def prepare_data(self, data):
        self.train_dataloader = data.train_dataloader()
        self.val_dataloader = data.val_dataloader()
        self.num_train_batches = len(self.train_dataloader)
        self.num_val_batches = (len(self.val_dataloader)
                                if self.val_dataloader is not None else 0)

    def prepare_model(self, model):
        model.trainer = self
        model.board.xlim = [0, self.max_epochs]
        self.model = model

    def fit(self, model, data):
        self.prepare_data(data)
        self.prepare_model(model)
        self.optim = model.configure_optimizers()
        self.epoch = 0
        self.train_batch_idx = 0
        self.val_batch_idx = 0
        for self.epoch in range(self.max_epochs):
            self.fit_epoch()

    def fit_epoch(self):
        raise NotImplementedError

    def prepare_batch(self, batch):
        """Defined in :numref:`sec_linear_scratch`"""
        return batch

    def fit_epoch(self):
        """Defined in :numref:`sec_linear_scratch`"""
```

```
    self.model.train()
    for batch in self.train_dataloader:
        loss = self.model.training_step(self.prepare_batch(batch))
        self.optim.zero_grad()
        with torch.no_grad():
            loss.backward()
            if self.gradient_clip_val > 0:  # To be discussed later
                self.clip_gradients(self.gradient_clip_val, self.model)
            self.optim.step()
        self.train_batch_idx += 1
    if self.val_dataloader is None:
        return
    self.model.eval()
    for batch in self.val_dataloader:
        with torch.no_grad():
            self.model.validation_step(self.prepare_batch(batch))
        self.val_batch_idx += 1


def __init__(self, max_epochs, num_gpus=0, gradient_clip_val=0):
    """Defined in :numref:`sec_use_gpu`"""
    self.save_hyperparameters()
    self.gpus = [gpu(i) for i in range(min(num_gpus, num_gpus()))]


def prepare_batch(self, batch):
    """Defined in :numref:`sec_use_gpu`"""
    if self.gpus:
        batch = [to(a, self.gpus[0]) for a in batch]
    return batch


def prepare_model(self, model):
    """Defined in :numref:`sec_use_gpu`"""
    model.trainer = self
    model.board.xlim = [0, self.max_epochs]
    if self.gpus:
        model.to(self.gpus[0])
    self.model = model


def clip_gradients(self, grad_clip_val, model):
```

```
        """Defined in :numref:`sec_rnn-scratch`"""
        params = [p for p in model.parameters() if p.requires_grad]
        norm = torch.sqrt(sum(torch.sum((p.grad ** 2)) for p in params))
        if norm > grad_clip_val:
            for param in params:
                param.grad[:] *= grad_clip_val / norm

# main model functions

def concatPreds(preds):
    return torch.cat([torch.flatten(p.permute(0, 2, 3, 1), start_dim=1) for p in preds], dim=1)

def blkForward(X, blk, size, ratio, cls_predictor, bbox_predictor):
    Y = blk(X)
    # generate anchor boxes
    anchors = multiboxPrior(Y, sizes=size, ratios=ratio)
    cls_preds = cls_predictor(Y)
    bbox_preds = bbox_predictor(Y)
    return (Y, anchors, cls_preds, bbox_preds)

# Help blok for featue map layer
def downSampleBlk(in_channels, out_channels):
    blk = []
    for _ in range(2):
        blk.append(nn.Conv2d(in_channels, out_channels,
                             kernel_size=3, padding=1))
        blk.append(nn.BatchNorm2d(out_channels))
        blk.append(nn.ReLU())
        in_channels = out_channels
    blk.append(nn.MaxPool2d(2))
    return nn.Sequential(*blk)

def base_net():
    blk = []
    num_filters = [3, 16, 32, 64]
    for i in range(len(num_filters) - 1):
        blk.append(downSampleBlk(num_filters[i], num_filters[i+1]))
    return nn.Sequential(*blk)

def get_blk(i):
```

```
    if i == 0:
        blk = base_net()
    elif i == 1:
        blk = downSampleBlk(64, 128)
    elif i == 4:
        blk = nn.AdaptiveMaxPool2d((1,1))
    else:
        blk = downSampleBlk(128, 128)
    return blk

sizes = [[0.1, 0.1], [0.2, 0.2], [0.3, 0.3], [0.4, 0.4],
         [0.5, 0.5], [0.6, 0.6], [0.7, 0.7], [0.8, 0.8],
         [0.9, 0.9]]
ratios = [[1, 2, 0.5]] * 9
num_anchors = len(sizes[0]) + len(ratios[0]) - 1

class TinySSD(nn.Module):
    def __init__(self, num_classes, **kwargs):
        super(TinySSD, self).__init__(**kwargs)
        self.num_classes = num_classes
        idx_to_in_channels = [64, 128, 128, 128, 128]
        for i in range(9):
            # Feature map for current scale
            setattr(self, f'blk_{i}', get_blk(i))
            # Set numer of output channels to num_anchors * (num_classes + 1) for prediction with conv layer
            setattr(self, f'cls_{i}', nn.Conv2d(idx_to_in_channels[i],
                                                num_anchors * (num_classes + 1), kernel_size=3, padding=1))
            # Conv layer for bounding box prediction
            setattr(self, f'bbox_{i}', nn.Conv2d(idx_to_in_channels[i],
                                                 num_anchors * 4, kernel_size=3, padding=1))

    def forward(self, X):
        anchors, cls_preds, bbox_preds = [None] * 5, [None] * 5, [None] * 5
        for i in range(5):
            # Here `getattr(self, 'blk_%d' % i)` accesses `self.blk_i`
            X, anchors[i], cls_preds[i], bbox_preds[i] = blkForward(
                X, getattr(self, f'blk_{i}'), sizes[i], ratios[i],
                getattr(self, f'cls_{i}'), getattr(self, f'bbox_{i}'))
        anchors = torch.cat(anchors, dim=1)
        cls_preds = concatPreds(cls_preds)
```

```
        cls_preds = cls_preds.reshape(
            cls_preds.shape[0], -1, self.num_classes + 1)
        bbox_preds = concatPreds(bbox_preds)
        return anchors, cls_preds, bbox_preds


def calcLoss(cls_preds, cls_labels, bbox_preds, bbox_labels, bbox_masks):
    batch_size, num_classes = cls_preds.shape[0], cls_preds.shape[2]
    cls = nn.CrossEntropyLoss(cls_preds.reshape(-1, num_classes),
                  cls_labels.reshape(-1)).reshape(batch_size, -1).mean(dim=1)
    bbox = nn.L1Loss(bbox_preds * bbox_masks,
                   bbox_labels * bbox_masks).mean(dim=1)
    return cls + bbox


def clsEval(cls_preds, cls_labels):
    # Because the class prediction results are on the final dimension,
    # `argmax` needs to specify this dimension
    return float((cls_preds.argmax(dim=-1).type(
        cls_labels.dtype) == cls_labels).sum())


def bboxEval(bbox_preds, bbox_labels, bbox_masks):
    return float((torch.abs((bbox_labels - bbox_preds) * bbox_masks)).sum())


def __main__():
    numEpochs = 10000
    net = TinySSD(num_classes=1)
    trainIter = 64 # batch size
    trainer = torch.optim.SGD(net.parameters(), lr=0.2, weight_decay=5e-4)


    for epoch in range(numEpochs):
        # Sum of training accuracy, no. of examples in sum of training accuracy,
        # Sum of absolute error, no. of examples in sum of absolute error
        metric = Accumulator(4)
        net.train()
        for features, target in trainIter:
            trainer.zero_grad()
            X, Y = features.to(device), target.to(device)
            # Generate multiscale anchor boxes and predict their classes and
            # offsets
            anchors, cls_preds, bbox_preds = net(X)
            # Label the classes and offsets of these anchor boxes
```

```
        bbox_labels, bbox_masks, cls_labels = multiboxTarget(anchors, Y)
        # Calculate the loss function using the predicted and labeled values
        # of the classes and offsets
        l = calcLoss(cls_preds, cls_labels, bbox_preds, bbox_labels,
                     bbox_masks)
        l.mean().backward()
        trainer.step()
        metric.add(clsEval(cls_preds, cls_labels), cls_labels.numel(),
                   bboxEval(bbox_preds, bbox_labels, bbox_masks),
                   bbox_labels.numel())
    cls_err, bbox_mae = 1 - metric[0] / metric[1], metric[2] / metric[3]
print(f'class err {cls_err:.2e}, bbox mae {bbox_mae:.2e}')
```

ДОДАТОК В

ВІДГУК

на кваліфікаційну роботу рівня магістра
«Розробка інформаційної технології розпізнавання об'єктів у відеоряді»
студента групи 126м-22-1 Дьякова Сергія Петровича

1 Мета даної кваліфікаційної роботи – розробка інформаційної технології розпізнавання об’єктів у відеоряді.

2 Обрана тема актуальна тому, що відстеження та виявлення об’єктів є одними з найважливіших завдань реалізації технологій комп’ютерного зору, які мають численні застосування у таких сферах, як автономне відстеження транспортних засобів, роботизація, моніторинг руху, розпізнавання облич і т.і.

3 Тема кваліфікаційної роботи відповідного рівня безпосередньо пов'язана з об'єктом діяльності магістра спеціальності 126 «Інформаційні системи та технології» галузі знань 12 «Інформаційні технології» – створення інформаційних технологій різного призначення і застосування.

4 Явища і процеси, що досліджуються в даній кваліфікаційній роботі і обрані для моделювання, оцінювання та реалізації – віднесені в освітньо-кваліфікаційній характеристиці магістрів до класу дослідних та евристичних, рішення яких заснована на знаково-понятійних уміннях.

5 Робота складається з трьох розділів. Перший розділ присвячений дослідженню засобів та технологій виявлення об’єктів, а також здійснено аналіз сучасних алгоритмів розпізнавання об’єктів. У другому розділі наведено проектну складову вирішення завдання та обґрунтувано вибір інструментів програмної реалізації технології. У третьому розділі виконано проектування та реалізацію інформаційної технологію розпізнавання об’єктів в відеоряді, та надані результати її експериментального дослідження. Оригінальність отриманих в роботі результатів та їх наукова новизна полягають у наступному:

- досліджено і обгрунтовано вибір архітектури моделі пошуку та класифікації об’єктів на зображенні та інструментів програмної реалізації технології;
- порівняно технології виявлення об’єктів в режимі реального часу;
- запропоновано сновні принципи побудови відповідної інформаційної технології розпізнавання об’єктів в відеоряді на основі наступних підходів:

• використання у якості основи архітектуру Single Shot Multibox Detection, з мережею VGG-16 у якості базової;

• модифікації вхідного шару нейронної мережі;

• об’єднання виведення всіх шарів із передбаченням, задля скорочення кількості застосувань функції втрати;

• вирівнювання розмірності результуючих матриць кожного блоку Multiscale Feature Maps;

- розроблено та реалізовано відповідну інформаційну технологію розпізнавання об’єктів у відеоряді.

6 Практичне значення результатів роботи полягає в тому, що інформаційна технологія дозволяє виконувати розпізнавання об’єктів в відеоряді в режимі реального часу.

7 Практичні результати кваліфікаційної роботи отримані із застосуванням відповідних технічних і програмних засобів, мови Python, а також програмних продуктів MS Word і MS PowerPoint на інформаційно-технологічній платформі Windows.

8 Оформлення графічних матеріалів до кваліфікаційної роботи рівня магістр виконано на сучасному рівні і відповідає вимогам, що пред'являються до рівня виконання робіт даної кваліфікації.

9 Ступінь самостійності виконання кваліфікаційної роботи достатньо висока.

10 Деякі дискусійні положення та недоліки, які мають місце в роботі:

а) недостатньо повно визначено кінцеву функціональну придатність даної розробки;

б) неповно уявлено, як архітектура моделі пошуку та класифікації об’єктів пов’язана з архітектурою інформаційної технології;

в) у роботі мають місце поодинокі орфографічні та стилістичні помилки.

Незважаючи на вищевказані зауваження, кваліфікаційна робота в цілому заслуговує оцінки «___відмінно (95 балів)_____» та присвоєння здобувачу відповідної кваліфікації.

Керівник кваліфікаційної роботи,
проф. кафедри ІТКІ, д.т.н. Г.М. Коротенко

ДОДАТОК Г

РЕЦЕНЗІЯ

на кваліфікаційну роботу рівня магістра
«Розробка інформаційної технології розпізнавання об'єктів у відеоряді»
студента групи 126м-22-1 Дьякова Сергія Петровича

Розглянута робота присвячена обґрунтування архітектури інформаційної технології, що забезпечує розпізнавання об’єктів у відеоряді із оптимальною точністю та швидкодією.

Завдання і зміст кваліфікаційної роботи відповідає головній цілі - перевірці знань і ступеня підготовленості студента за фахом 126 «Інформаційні системи та технології» галузі знань 12 «Інформаційні технології».

Зміст пояснювальної записки кваліфікаційної роботи відповідає необхідним критеріям та затвердженій темі.

Актуальність обраної теми обумовлена тим, що дослідження ефективності застосування інформаційних технологій і відповідних програмних засобів продовжують розвиватися на базі розширення їхнього спектру.

Повнота і глибина вирішення задач, поставлених в завданні на кваліфікаційну роботу є достатньою.

Оформлення пояснювальної записки кваліфікаційної роботи виконано в повній відповідності з діючими стандартами і нормативними вимогами.

Наукова новизна результатів кваліфікаційної роботи роботи визначається тим, що обґрунтовано архітектуру інформаційної технології розпізнавання об’єктів у відеоряді, що підтримує адаптацию класифікатора до масштабування зображення.

Практичне значення результатів роботи полягає у тому, що запропонована в роботі інформаційна технологія дозволяє виконувати розпізнавання об’єктів в відеоряді в режимі реального часу.

До числа загальних зауважень і недоліків роботи слід віднести:

1) недостатньо конкретизовано зв’язок кількості та особливостей компонентів архітектри моделі з відповідною архітектурою інформацйної технології;

2) не до кінця розкриті функціональні особливості реалізації та результатів практичного застосування розробленої технології;

3) дещо спрощений опис кінцевих результатів роботи.

Однак, зазначені зауваження не здійснюють істотного впливу на підсумкові результати кваліфікаційної роботи і не знижують її безумовну практичну та наукову цінність.

Таким чином, слід зробити висновок, що кваліфікаційна робота в цілому заслуговує оцінки «_____», а її виконавець присвоєння відповідної кваліфікації.

Рецензент, доцент кафедри програмного
забезпечення комп’ютерних систем НТУ
«ДП», к.т.н. А.Л. Ширін