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 Тимофієнко Павла Юрійовича
(ПІБ)
академічної групи 126м-22-2
(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 стеження за об'єктами 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
2024

ЗАТВЕРДЖЕНО:
завідувач кафедри

___інформаційних технологій___
___та комп'ютерної інженерії____
(повна назва)

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

«_____»_____ 2024 року

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

студенту _____Тимофієнку П.Ю._____ **академічної групи** ____126м-22-2___
(прізвище та ініціали) (шифр)

спеціальності ___126 « Інформаційні системи та технології »_____

за освітньою-професійною програмою _____

_____«Інформаційні системи та технології»_____

на тему Розробка інформаційної технології стеження за об'єктами 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 стеження за об'єктами у відеоряді.	15.10.2023 – 18.10.2023
Розділ 2	Проектування та розробка інформаційної технології стеження за об'єктами у відеоряді.	19.10.2023 – 15.11.2023
Розділ 3	Тестування створеного продукта, розгортання та підтримка. Оформлення кваліфікаційної роботи.	16.11.2023 – 10.01.2024

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

Дата видачі _____15.10.2023 р._____

Дата подання до екзаменаційної комісії ____17.01.2024 р.____

Прийнято до виконання _____ _____Тимофієнко П.Ю._____
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 101 стор., 40 рис., 1 табл., 4 додатка, 136 джерел.

Об'єкт дослідження: інформаційна технологія стеження за об’єктами у відеоряді.

Предмет дослідження: процес стеження за об’єктами у відеоряді.

Мета магістерської роботи: обґрунтування архітектури інформаційної технології, що забезпечує стеження за об’єктами 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 відстежування об’єктів, здійснено аналіз сучасних алгоритмів стеження за об’єктами.

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

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

Наукова новизна отриманих результатів визначається тим, що вперше обґрунтовано архітектуру інформаційної технології стеження за об’єктами у відеоряді, що підтримує високу точність під час обробки навантажених сцен.

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

ВІДСТЕЖУВАННЯ ОБ’ЄКТІВ, ВІДЕО, СІАМСЬКА НЕЙРОННА МЕРЕЖА, ТРАНСФОРМЕР, CNN, ANCHOR BOX, BOUNDING BOX, MDNET, BOTT.

ABSTRACT

Explanatory note: 101 pages, 40 figures, 1 table, 4 applications, 136 sources.

Object of research: information technology for tracking objects in a video sequence.

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

Purpose of Master's thesis: substantiation of information technology architecture that ensures tracking objects in the 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 tools and technologies of object tracking is given, an analysis of modern object tracking 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 tracking objects in the video sequence is performed, and the results of its experimental research are provided.

Originality of research is associated with first substantiation of architecture of the information technology for object tracking in video sequence, which supports high accuracy during the processing of busy scenes.

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

OBJECT TRACKING, VIDEO, SIAMIAN NEURAL NETWORK, TRANSFORMER, CNN, ANCHOR BOX, BOUNDING BOX, MDNET, BOTT.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

CNN - Convolutional Neural Network

DL - Deep Learning

FP - False Positives

GRU - Gated Recurrent Unit

LSTM - Long Short-Term Memory

MLP - Multi Layers Perceptron

MOT - Multiple Object Tracking

OD - Object Detection

OT - Object Tracking

RNN - Recurrent Neural Network

SOT - Single Object Tracking

SVM - Support Vector Machine

ViT - Vision Transformer

VOT - Visual Object Tracking

ШІ – штучний інтелект

ВСТУП

Актуальність дослідження. В останні роки такі галузі, як аналіз зображень і аналіз відео, розширили коло потенційних застосувань. Дві фундаментальні технології, які домінують у цій сфері, — це комп'ютерний зір та штучний інтелект. Візуальне відстеження об'єктів є ключовим завданням комп'ютерного зору з широким спектром застосування в різних сферах, включаючи відеоспостереження, інтелектуальне транспортування та взаємодію людини з комп'ютером. Метою візуального відстеження об'єктів є точне та послідовне відстеження конкретних цілей у відеоряді. Однак цьому процесу заважають такі фактори, як деформація, оклюзія, швидкий рух і фонові перешкоди.

Виявлення та відстеження об'єктів у режимі реального часу є великою, перспективною та складною областю комп'ютерного зору. Активне використання технології відстеження об'єктів сприяє розробці більш ефективних та конкурентоспроможних алгоритмів. Однак при реалізації відстеження об'єктів у реальному часі досі виникають проблеми, такі, як відстеження в динамічному середовищі, дорогі обчислення для забезпечення продуктивності в реальному часі або відстеження кількох об'єктів за допомогою кількох камер. Тому розробка та вдосконалення технологій стеження за об'єктами в відеоряді залишається актуальним напрямом досліджень.

Об'єкт дослідження – інформаційна технологія стеження за об'єктами у відеоряді.

Предмет дослідження – процес стеження за об'єктами у відеоряді.

Мета й завдання дослідження. Мета роботи полягає у обґрунтуванні архітектури інформаційної технології, що забезпечує стеження за об'єктами 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 стеження за об’єктами в відеоряді;
- обґрунтувати архітектуру інформаційної технології стеження за об’єктами в відеоряді;
- розробити програмне забезпечення інформаційної технології стеження за об’єктами в відеоряді на базі обраної архітектури.

Наукова новизна результатів кваліфікаційної роботи визначається тим, що вперше обґрунтовано архітектуру інформаційної технології стеження за об’єктами у відеоряді, що підтримує високу точність під час обробки навантажених сцен.

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

Робота складається з трьох розділів.

Перший розділ присвячено аналізу тему дослідження та постановці задачі. Наведено огляд засобів та технологій відстежування об’єктів і коротко розглянуті сучасні алгоритми стеження за об’єктами.

В другому розділі наведено проектну складову вирішення завдання. Розглянуто деякі відомі моделі стеження за об’єктами в режимі реального часу, детально описано алгоритми стеження на основі трансформерів та сіамських нейронних мереж. Обґрунтовано вибір інструментів програмної реалізації інформаційної технології.

Третій розділ присвячено розробці інформаційної технології стеження за об’єктами у відеоряді. Виконано реалізацію інформаційної технології.

Створено програмне забезпечення, яке призначено для стеження за об’єктами у відеоряді в режимі реального часу.

РОЗДІЛ 1
АНАЛІЗ СТАНУ ОБЛАСТІ РІШЕННЯ ЗАДАЧІ

1.1. Концепція та технології стеження за об’єктами у відеоряді

Відстеження об’єктів є важливим завданням комп’ютерного зору. А трекери об’єктів є невід’ємною частиною багатьох програм комп’ютерного зору, які обробляють відеопотік камер. Відстеження об’єктів — це програма глибокого навчання, де програма бере початковий набір виявлень об’єктів і розробляє унікальну ідентифікацію для кожного з них, а потім відстежує виявлені об’єкти, коли вони переміщуються кадрами у відео. Іншими словами, відстеження об’єктів — це завдання автоматичної ідентифікації об’єктів на відео та інтерпретації їх як набору траєкторій з високою точністю. Часто навколо об’єкта, який відстежується, є індикація, наприклад, квадрат навколо об’єкта, який показує користувачеві, де на екрані знаходиться об’єкт.

Відстеження об’єктів використовується для різноманітних випадків використання різних типів вхідного відеоматеріалу. Від того, чи буде передбачений вхід зображення чи відео, або відео в реальному часі чи попередньо записане відео, залежить вибір алгоритмів, які використовуються для створення програм відстеження об’єктів. Тип введення також впливає на категорію, випадки використання та застосування відстеження об’єктів.

Відстеження відео – це програма відстеження об’єктів, де рухомі об’єкти знаходяться у відеоінформації. Таким чином, системи відеоспостереження здатні обробляти відзнятий матеріал у реальному часі, а також записані відеофайли. Процеси, які використовуються для виконання завдань відстеження відео, відрізняються залежно від типу цільового відеовходу.

Різні програми відеоспостереження відіграють важливу роль у відеоаналітиці, у розумінні сцени для безпеки та спостереження, військових, транспортних та інших галузей. Сьогодні методи відстеження відео використовуються у широкому спектрі програм комп’ютерного бачення в режимі реального часу та глибокого навчання.

Візуальне відстеження (Visual Object Tracking, VOT) або візуальне відстеження цілі є науковою темою в області комп’ютерного зору, яка застосовується у багатьох повсякденних сценаріях. Метою візуального відстеження є оцінка майбутнього положення візуальної цілі, яка була ініціалізована без наявності решти відео.

Відстеження зображень призначене для виявлення цільових двовимірних зображень у певному вхідному файлі. Потім це зображення безперервно відстежується, коли воно рухається в налаштуваннях. Таким чином, відстеження зображень ідеально підходить для наборів даних із висококонтрастними зображеннями (наприклад, чорно-білими), асиметрією, невеликою кількістю візерунків і кількома відмінностями між цільовими зображенням та іншими зображеннями в наборі зображень. Відстеження зображень покладається на комп’ютерне зір для виявлення та доповнення зображень після попереднього визначення цільових зображень.

Сучасні методи відстеження об'єктів можна застосувати до відеопотоків у реальному часі практично будь-якої камери. Таким чином, відео з USB-камери або IP-камери можна використовувати для відстеження об’єктів шляхом передачі окремих кадрів алгоритму відстеження. Пропуск кадрів або паралельна обробка є поширеними методами покращення продуктивності відстеження об’єктів за допомогою відео в реальному часі з однієї чи кількох камер.

Між відстеженням і виявленням об'єктів існує тісний зв’язок. Виявлення полягає у визначенні місцезнаходження одного або кількох об’єктів на певному зображенні, тоді як метою відстеження є визначення

місцезнаходження цих об’єктів протягом усього відео, відстеження того, який об’єкт є яким, уздовж кадрів відео. Щоб відстежувати об’єкт, спочатку потрібно надати зображення зазначеного об’єкта в алгоритм відстеження, і це робиться або за допомогою алгоритму виявлення (трекери на основі виявлення), або вручну (трекери без виявлення).

Найпростіший спосіб виконання відстеження полягає в застосуванні алгоритму виявлення до кожного кадру відео, але є кілька причин, чому відстеження є необхідним або корисним:

- відстеження дозволяє підтримувати ідентифікацію об'єктів;

- виявлення є обчислювально дорогим;

- трекери без виявлення дозволяють відстежувати об'єкти, для яких детектор не був навчений;

- відстеження може допомогти вирішити складні типові проблеми, такі як зміна освітлення, розмиття руху, зміна масштабу, оклюзії (коли ціль частково або повністю прихована іншим об’єктом на певний період часу у відео), низька якість зображення.

1.2. Рівні стеження за об’єктами у відеоряді

Алгоритми візуального відстеження об’єктів призначені для оцінки стану (тобто просторового розташування та розміру) об’єкта в даній відеопослідовності. Враховуючи початковий стан цілі в першому кадрі відеопослідовності, ці алгоритми відстежують стан цілі в решті кадрів. VOT можна класифікувати як одиночний і багатооб’єктний підхід [1, 2] на основі кількості цілей, які слід відстежувати. При відстеженні одного об’єкта (Single Object Tracking, SOT) відстежується один екземпляр класу об’єктів, тоді як при відстеженні кількох об’єктів (Multiple Object Tracking, MOT) відстежуються кілька екземплярів класу об’єктів.

1.2.1. Відстеження одного об'єкта

Алгоритми відстеження окремих об’єктів стали популярними та викликали інтерес в останні роки через широкий діапазон застосувань у комп’ютерному зорі, включаючи відеоспостереження [3], доповнену реальність [4], автоматизоване водіння [5], мобільну робототехніку [6], моніторинг дорожнього руху [7], аналіз спортивного відео [8], розуміння сцени [9] та взаємодія людини з комп’ютером [10]. Підхід VOT до одного об’єкта фіксує особливості зовнішнього вигляду цілі в першому кадрі відеопослідовності, а потім використовує його для визначення місцезнаходження цілі в решті кадрів. Незважаючи на те, що в VOT було запропоновано багато підходів, заснованих на зовнішньому вигляді, він ще далекий від досягнення надійності відстеження людей в реальному часі через багато проблем, таких як зовнішній вигляд і варіації поз, оклюзії, розмиття руху, фоновий безлад, подібні відволікаючі об’єкти , і деформація.

Алгоритми відстеження одного об’єкта можна класифікувати та аналізувати кількома способами. На основі функцій, які використовуються в моделі відстеження, підходи SOT можна класифікувати як ручні трекери та трекери на основі глибоких функцій. Створені вручну підходи відстеження на основі функцій [11, 12, 13, 14, 15] виділяють ознаки із зображень відповідно до певного попередньо визначеного вручну алгоритму на основі експертних знань. З іншого боку, трекери на основі глибоких функцій [16, 17, 18, 19] фіксують семантичні підказки з необроблених зображень за допомогою згорткових нейронних мереж (CNN) [20]. Завдяки можливості ієрархічного навчання трекери зовнішнього вигляду на основі глибоких функцій значно перевершують створені вручну трекери на основі функцій [21]. Залежно від того, як трекери відрізняють цільовий об’єкт від його оточення, вони можуть бути дискримінаційними та генеративними трекерами.

Дискримінаційні трекери [22, 23] розглядають SOT як задачу бінарної класифікації та відокремлюють цільовий об’єкт від фону. З іншого боку, більшість генеративних трекерів [24, 25] розглядають SOT як проблему відповідності подібності шляхом пошуку найкращого кандидата, який точно відповідає еталонному шаблону в кожному кадрі. У останніх трекерах архітектура CNN з двома гілками, відома як сіамська мережа [26], використовується для відстеження відповідності подібності. За останні кілька років у SOT було запропоновано велику кількість сіамських трекерів [27, 28, 29, 30, 31, 32, 33, 34], оскільки вони продемонстрували чудову продуктивність із високою обчислювальною ефективністю. Однак дискримінаційна здатність сіамських трекерів є досить поганою [35], оскільки вони зосереджуються лише на навчанні візуального представлення цільового об’єкта для відповідності подібності, ігноруючи фонову та цільову специфічну інформацію, і, отже, показують низьку продуктивність у оклюзії та деформації сцени.

1.2.2. Відстеження кількох об'єктів

Відстеження кількох об’єктів (MOT) спрямоване на розпізнавання, локалізацію та відстеження об’єктів у певній відеопослідовності. Це наріжний камінь динамічного аналізу сцени та життєво важливий для багатьох реальних програм, таких як автономне водіння, доповнена реальність і відеоспостереження. Традиційно тести MOT [36-39] визначають набір семантичних категорій, які становлять об’єкти, що відстежуються в розподілі даних навчання та тестування. Тому потенціал традиційних методів MOT [40-42] обмежений таксономіями цих тестів. Як наслідок, сучасні методи ТО борються з невидимими подіями, що призводить до розриву між ефективністю оцінювання та розгортанням у реальному світі.

Щоб подолати цю прогалину, численні наукові роботи розглядали МОТ в контексті відкритого світу. Зокрема, Osep et al. [43] підходять до загального відстеження об’єктів, спочатку сегментуючи сцену та виконуючи відстеження перед класифікацією. В інших роботах використовувалися класові агностичні локалізатори [44-45] для виконання МОТ на довільних об’єктах. Нещодавно Liu et al. [46] визначив відстеження у відкритому світі, завдання, яке зосереджується на оцінці раніше невидимих об’єктів. Зокрема, це вимагає відстеження будь-яких об'єктів як етапу, що передує класифікації об'єктів. Ця установка пов’язана з двома труднощами. По-перше, у контексті відкритого світу щільне анотування всіх об’єктів є непомірно дорогим. По-друге, без заздалегідь визначеної таксономії категорій поняття того, що таке об’єкт, є неоднозначним. Як наслідок, Liu et al. вдатися до оцінки на основі запам'ятовування, яка обмежена двома способами. Покарання за помилкові спрацьовування (FP) стає неможливим, тобто виміряти точність трекера неможливе. Крім того, оцінюючи відстеження в класово-агностичний спосіб, втрачається можливість оцінити те, наскільки добре трекер може зробити висновок про семантичну категорію об’єкта.

Виявлення об’єктів зазвичай дає набір обмежувальних прямокутників як вихідні дані. Відстеження кількох об’єктів часто не потребує попередньої підготовки щодо зовнішнього вигляду та кількості цілей. Обмежувальні прямокутники визначаються за їх висотою, шириною, координатами та іншими параметрами. Тим часом алгоритми МОТ додатково призначають цільовий ідентифікатор кожній обмежувальній рамці. Цей цільовий ідентифікатор відомий як виявлення, і він важливий, оскільки дозволяє моделі розрізняти об’єкти в межах класу. Наприклад, замість ідентифікації всіх автомобілів на фотографії, де кілька автомобілів присутні лише як «автомобіль», МОТ намагаються ідентифікувати різні автомобілі як такі, що відрізняються один від одного, а не всі підпадають під позначку «автомобіль». Візуальне представлення цієї відмінності показано на рис. 1.1.

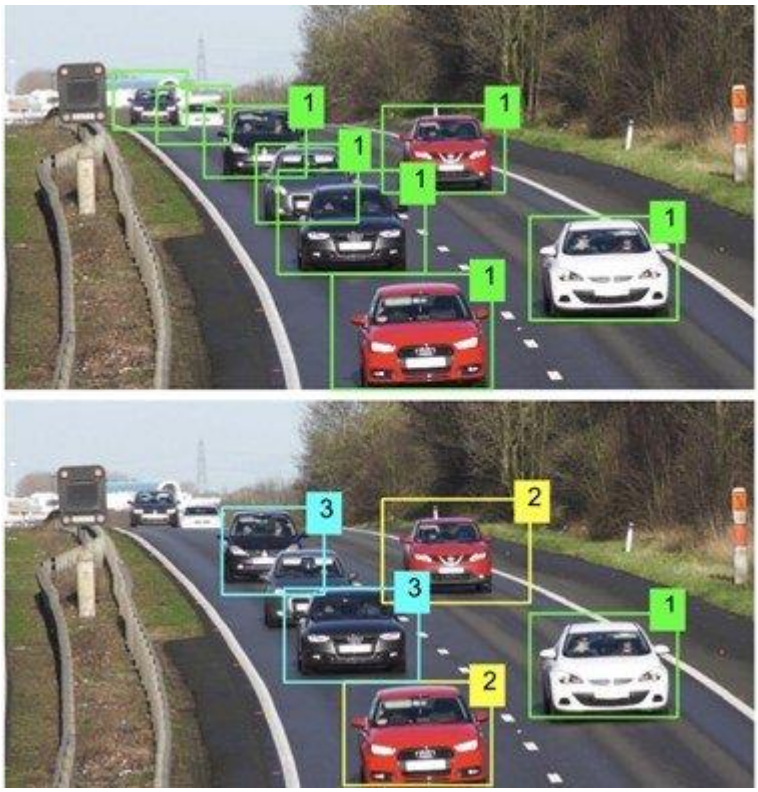

Рис. 1.1. Результат роботи алгоритму виявлення об’єктів (верхнє зображення) та відстеження кількох об’єктів (нижнє зображення) [47]

Більшість алгоритмів MOT включають підхід відстеження за виявленням. Метод відстеження за виявленням включає незалежний детектор, який застосовується до всіх кадрів зображення для отримання ймовірних виявлень, а потім засіб відстеження, який запускається на наборі виявлень. При цьому трекер намагається виконати асоціацію даних (наприклад, пов’язати виявлення для отримання повних траєкторій). Виявлення, отримані з відеовходів, використовуються для керування процесом відстеження шляхом їх з’єднання та призначення ідентичних ідентифікаторів обмежувальним рамкам, що містять ту саму ціль.

Алгоритми пакетного відстеження використовують інформацію з майбутніх відеокадрів під час визначення ідентичності об’єкта в певному кадрі. Алгоритми пакетного відстеження використовують нелокальну інформацію про об’єкт. Ця методологія забезпечує кращу якість відстеження.

В той час як алгоритми пакетного відстеження отримують доступ до майбутніх кадрів, алгоритми онлайн-відстеження використовують лише поточну та минулу інформацію, щоб зробити висновки щодо певного кадру. Онлайн-методи відстеження для виконання MOT зазвичай працюють гірше, ніж пакетні методи, через обмеження онлайн-методів, які залишаються обмеженими поточною структурою. Однак ця методологія іноді необхідна через варіант використання. Наприклад, проблеми в реальному часі, які вимагають відстеження об’єктів, як-от навігація чи автономне водіння, не мають доступу до майбутніх відеокадрів, тому методи онлайн-відстеження все ще є життєздатним варіантом.

Більшість алгоритмів відстеження кількох об’єктів містять базовий набір кроків, які залишаються незмінними.

Крок 1: Позначення або виявлення: на етапі позначення позначаються та виділяються цілі, що представляють інтерес. Алгоритм аналізує вхідні кадри для ідентифікації об’єктів, які належать до цільових класів. Обмежувальні рамки використовуються для виконання виявлень як частини алгоритму.

Крок 2: Рух: алгоритми виділення функцій аналізують виявлення, щоб виділити особливості зовнішнього вигляду та взаємодії. Прогноз руху в більшості випадків використовується для прогнозування наступних позицій кожної відстежуваної цілі.

Крок 3: Відкликання: передбачення ознак використовуються для обчислення показників подібності між позиціями виявлення. Потім ці бали використовуються для пов’язування виявлень, які належать одній цілі. Ідентифікатори призначаються подібним виявленням, а різні ідентифікатори застосовуються до виявлень, які не є частиною пар.

Деякі моделі відстеження об’єктів створюються за допомогою цих кроків окремо один від одного, тоді як інші поєднують і використовують

кроки разом. Ці відмінності в обробці алгоритмів створюють унікальні моделі різної точності.

1.3. Методи відстеження об’єктів у відеоряді

Методи відстеження об’єктів мають різні категорії, наприклад, Fiaz et al. представили комплексне дослідження методів відстеження, яке класифікує методи відстеження на дві групи методів, заснованих на кореляційному фільтрі та некореляційному фільтрі [48]. Li et al. переглянули та порівняли методи глибокого навчання відстеження об’єктів [49], Verma переглянула методи виявлення та відстеження об’єктів і розділила методи відстеження на п’ять категорій: на основі ознак, на основі сегментації, на основі оцінки, на основі зовнішнього вигляду, та засновані на навчанні [50]. У кваліфікаційній роботі буде використовується саме остання група, спираючись також на класифікацію методів відстеження одного об’єкта [51]; детальна характеристика кожного з них наведені нижче.

1. Методи на основі функцій

Цей метод є одним із простих способів відстеження об’єктів. Щоб відстежувати об’єкти, спочатку виділяють такі характеристики, як колір, текстура, оптичний потік тощо. Ці виділені ознаки мають бути унікальними, щоб об’єкти можна було легко розрізнити в просторі ознак. Після виділення ознак наступним кроком є пошук найбільш схожого об’єкта в наступному кадрі, використовуючи ці ознаки, використовуючи певний критерій подібності. Одна з проблем цих методів полягає на етапі виділення, оскільки необхідно виділити унікальні, точні та надійні характеристики об’єкта, щоб можна було відрізнити ціль від інших об’єктів. Ось деякі з функцій, які використовуються для відстеження об’єктів:

а) Колір

За ознаками кольору можна показати зовнішній вигляд предмета. Цю функцію можна використовувати різними способами, одним із найпоширеніших методів використання цієї функції є кольорова гістограма. Кольорова гістограма показує розподіл кольорів на зображенні, фактично показує різні кольори та кількість пікселів кожного кольору на зображенні. Недоліком кольорових гістограм є те, що представлення залежить лише від кольору об’єкта та ігнорує форму та текстуру об’єкта, тому два різні об’єкти можуть мати однакову гістограму. У деяких роботах для відстеження використовувалася кольорова гістограма [52, 53]. Zhao [54] використовував гістограми для відстеження; На рис. 1.2. показана блок-схема відстеження за допомогою цього методу.

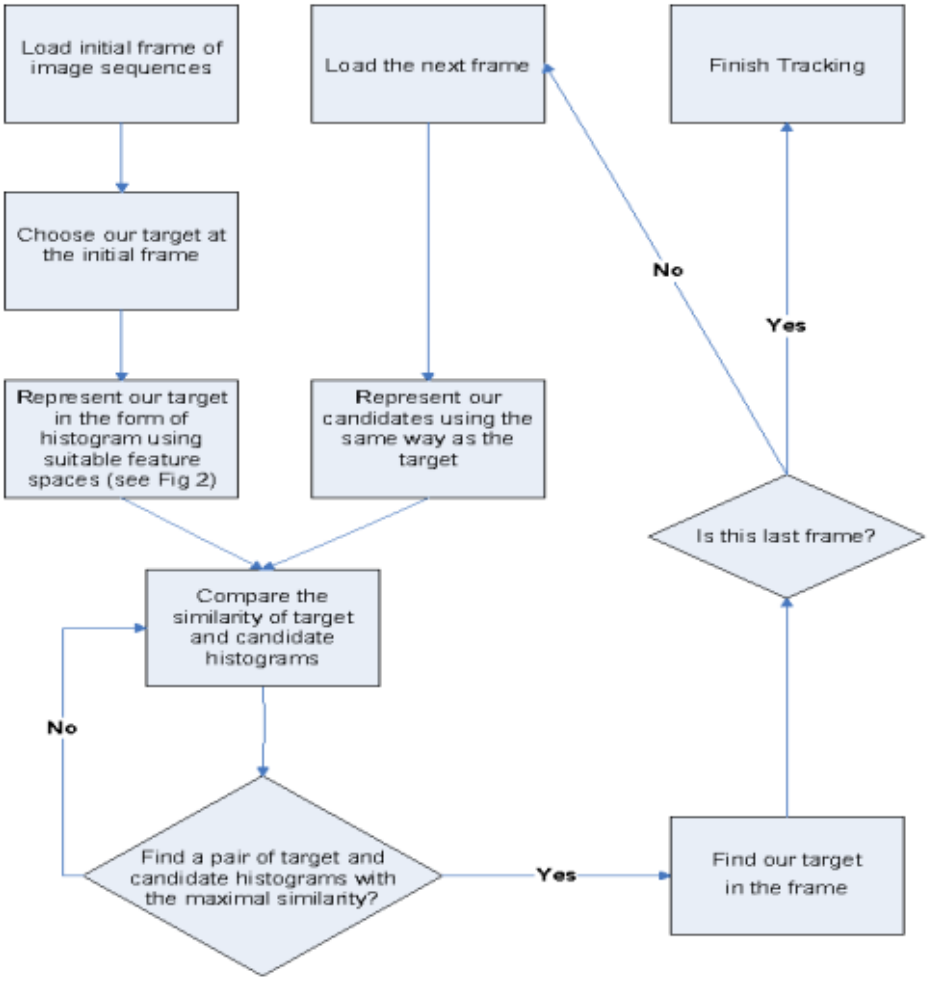

Рис. 1.2. Блок-схема відстеження об’єктів на основі гістограми в просторі градієнтних ознак [54]

Стеження здійснюється таким чином, що в першому кадрі ціль позначена обмежувальною рамкою; потім, як видно на рис. 1.3, ця мета відображається за допомогою гістограми у відповідному просторі ознак. Після входу в наступний кадр ціль слід шукати в цьому кадрі. У новому кадрі метод вводить усіх можливих кандидатів; потім проводиться порівняння, щоб знайти пару з максимальною подібністю між кандидатом і цільовою гістограмою.

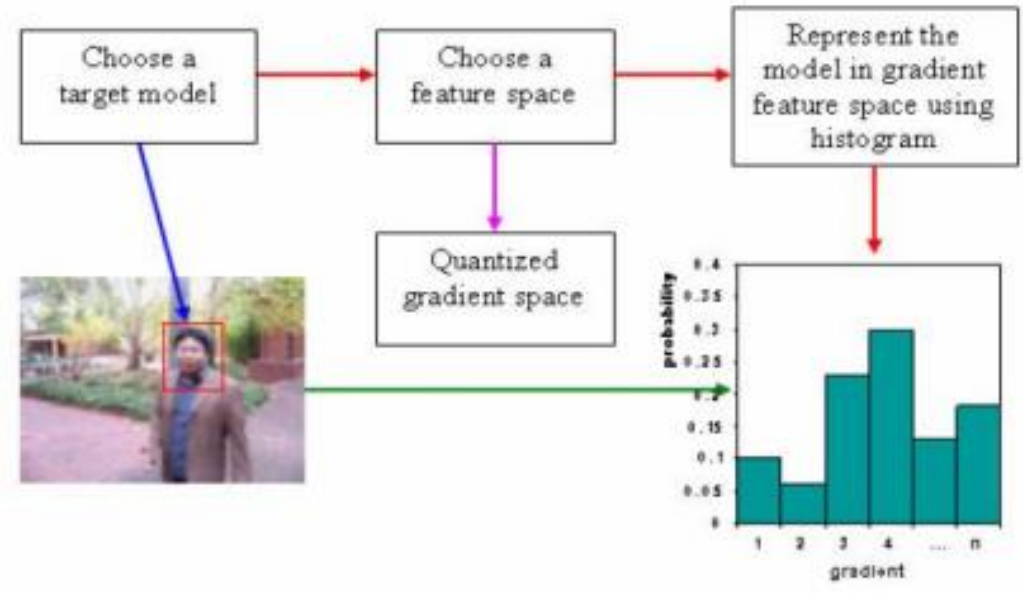

Рис. 1.3. Визначення цільової моделі за допомогою гістограми [54]

Після знаходження цілі вставляється наступний кадр і ті самі кроки повторюються до досягнення останнього кадру.

б) Текстура

Текстура — це повторюваний візерунок інформації або розташування структури з регулярними інтервалами. Характеристики текстури не отримуються безпосередньо, а генеруються методами попередньої обробки зображення.

Функція текстури є важливою характеристикою зображення, її можна використовувати разом із ознакою кольору для опису вмісту зображення або його ділянки. Оскільки ознаки кольору недостатньо для ідентифікації

подібних об’єктів, іноді можна побачити, що різні зображення мають однакову гістограму.

Вейвлет Габора [55] є однією з найбільш вивчених особливостей текстури. Найважливішою властивістю фільтрів Габора є їх незмінність до освітлення, обертання, масштабу та трансляції, що робить їх придатними для відстеження об’єктів. У [56] представлено метод визначення розташування та форми тіла рухомих тварин за допомогою фільтра Габора. Zhao et al. використовували LBP для опису рухомих об’єктів, а також використовували фільтр Калмана для відстеження цілей [57].

На рис. 1.4 показано вейвлети Габора зображення обличчя, які містять серію ядер Габора [58].

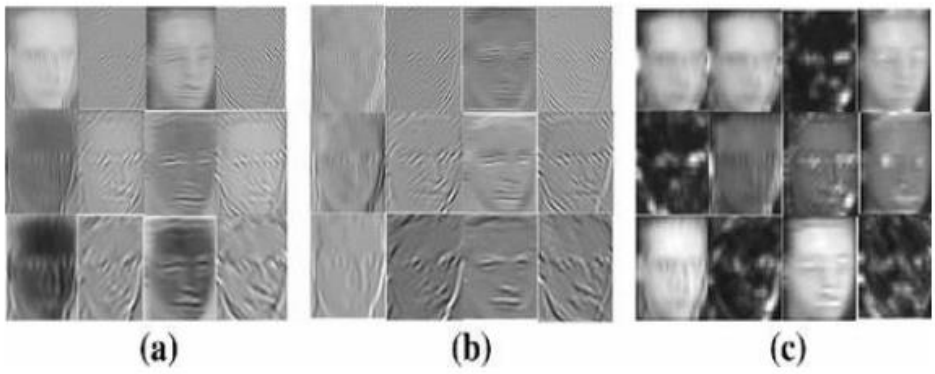

Рис. 1.4. (a): реальна частина ядер Габора. (b): уявна частина ядер Габора. (c): амплітуда ядер Габора [58]

в) Оптичний потік

Оптичний потік — це видимий рух шаблонів яскравості на зображенні. видимий рух може бути спричинений зміною освітлення без будь-якого фактичного руху. Алгоритм оптичного потоку обчислює зміщення шаблонів яскравості від одного кадру до іншого. Алгоритми, які обчислюють зміщення для всіх пікселів зображення, відомі як алгоритми щільного оптичного потоку, тоді як розріджені алгоритми оцінюють напругу зміщення для вибіркової кількості пікселів у зображенні. На рис. 1.5 аналізуються чотири послідовні кадри і для них розраховується оптичний потік. Зелені лінії — це

вектори руху, кінцеві точки яких показано червоною крапкою. Як можна помітити, деякі об’єкти позначені червоною крапкою, яка вказує на те, що об’єкт нерухомий і довжина вектора його руху дорівнює нулю. Блоки, які рухаються більше, мають більші вектори руху.

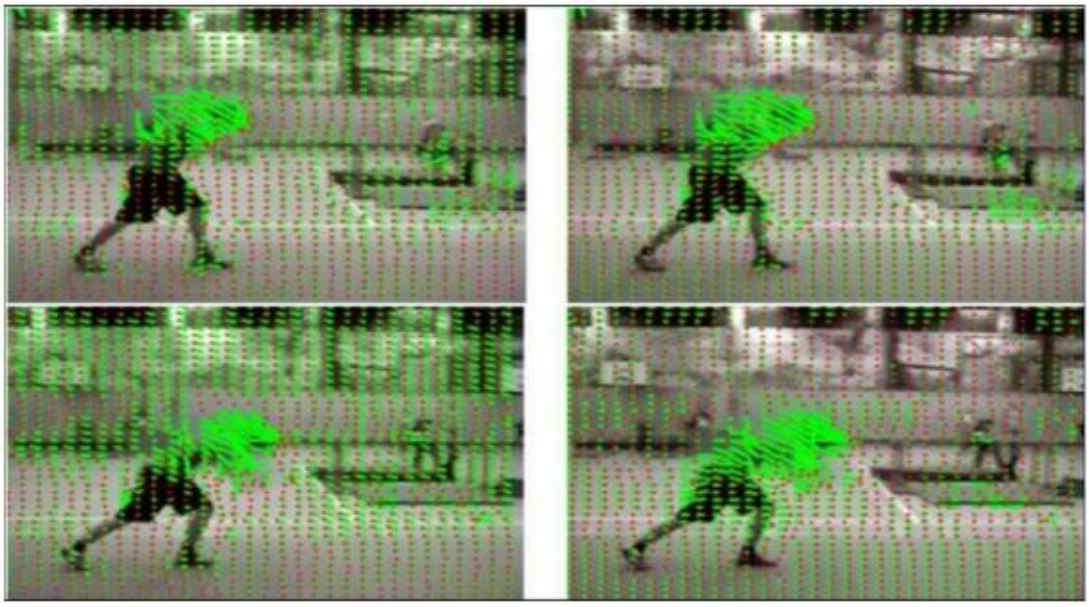

Рис. 1.5. Розрахунок оптичного потоку в кадрах [59]

У деяких роботах для відстеження використовувався оптичний потік [59, 60]. Hariyono et al. [60] представляють метод, заснований на оптичному потоці та HOG, який може виявляти пішоходів зсередини транспортного засобу, що рухається. Для отримання оптичного потоку два послідовних зображення розділені на комірки сітки 14×14 пікселів. Потім слідкують за кожною клітинкою в поточному кадрі, щоб змінити відповідну клітинку в наступному кадрі.

2. Методи на основі сегментації

Сегментування об’єктів переднього плану з відеокадру є фундаментальним і найважливішим кроком у візуальному відстеженні. Сегментація переднього плану виконується для відділення об’єктів переднього плану від фону сцени. Зазвичай об’єкти переднього плану – це об’єкти, які рухаються в сцені. Щоб відстежувати ці об’єкти, їх необхідно

відокремити від фону сцени [50]. Нижче розглядаються деякі методи відстеження об’єктів на основі сегментації.

а) Метод «знизу вгору».

У цьому типі відстеження має бути два окремих завдання: спочатку сегментація переднього плану, а потім відстеження об’єкта. Сегментація переднього плану використовує низькорівневу сегментацію для виділення областей у всіх кадрах, а потім деякі функції витягуються з областей переднього плану та відстежуються відповідно до цих функцій [61], як показано на рис. 1.6.

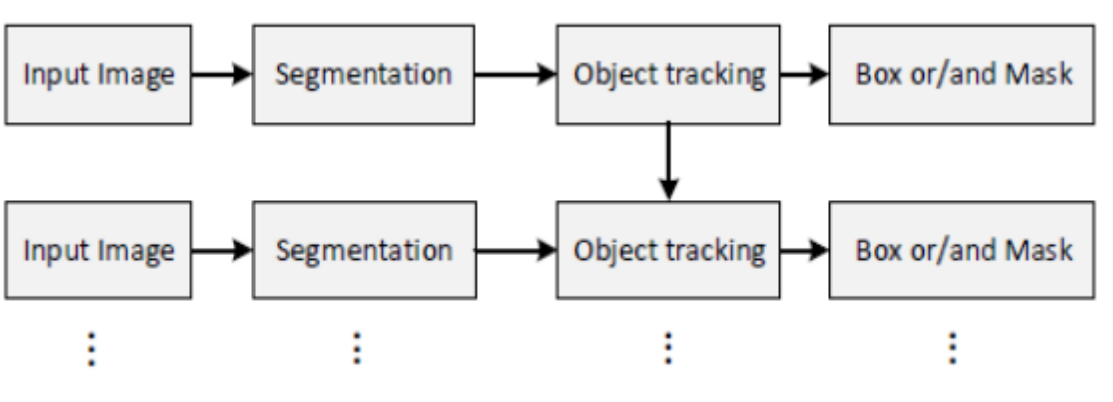

Рис. 1.6. Фреймворк «знизу вгору» [60]

У деяких роботах для відстеження використовувався висхідний метод [62-64]. Son et al. [63] використовує цей метод для відстеження об’єктів, які поширюються назад через фільтри часток, щоб оцінити цільовий стан, і він отримує цільову область шляхом класифікації секцій бажаної області та використовує онлайн-дерево рішень із посиленням градієнта для її класифікації.

б) Метод об’єднання

У методі «знизу вгору» сегментація переднього плану та відстеження є двома окремими завданнями; одна з проблем цього методу полягає в тому, що помилка сегментації поширюється вперед, викликаючи відстеження помилок. Щоб вирішити цю проблему, як показано на рис. 1.7, дослідники

об’єднали сегментацію переднього плану та метод відстеження, що покращило продуктивність відстеження.

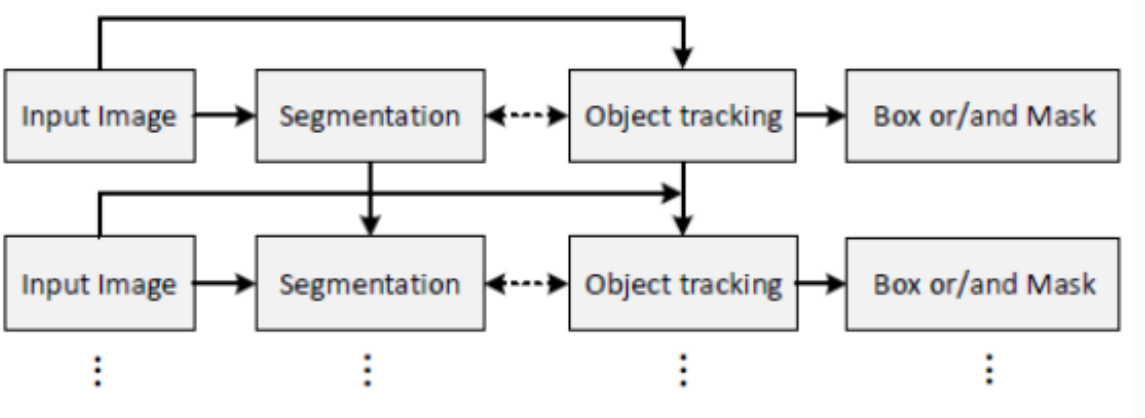

Рис. 1.7. Структура об’єднання [60]

У деяких роботах для відстеження використовується метод об’єднання [65-67]. Schubert et al. [67] використовували цей метод для відстеження. У статті [67] використовується базовий метод, який складається з трьох етапів, як показано на рис. 1.8. На першому кроці модель зовнішнього вигляду будується за допомогою ймовірнісного формулювання. На другому кроці модель сегментується за допомогою цієї моделі; і на третьому кроці виконується відстеження.

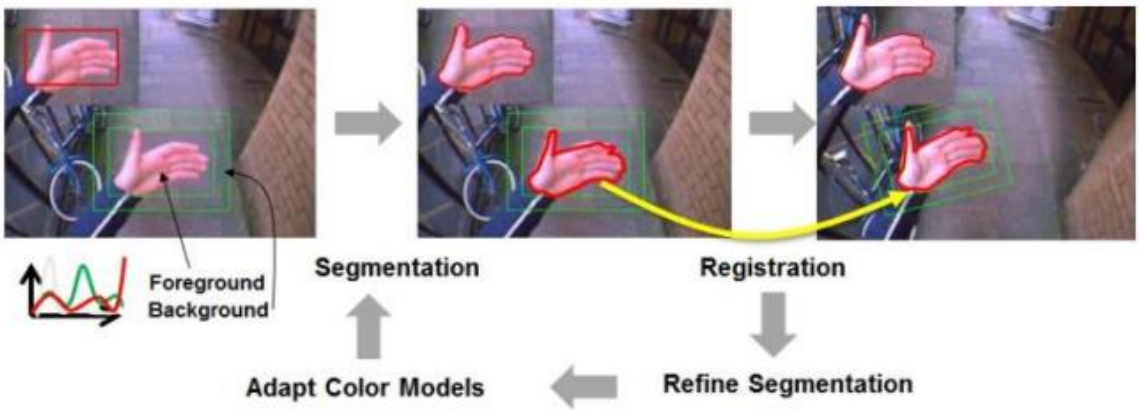

Рис. 1.8. Огляд прикладу об’єднаного відстеження [67]

3. Методи на основі оцінки

Методи оцінки формулюють задачу відстеження до задачі оцінки, в якій об’єкт представлено вектором стану. Вектор стану описує динамічну

поведінку системи, наприклад положення та швидкість об’єкта. Загальна структура проблеми оцінки динамічного режиму взята з байєсівських методів [50]. Байєсовські фільтри дозволяють цілі постійно оновлювати своє положення в системі координат на основі останніх даних датчиків. Цей алгоритм є рекурсивним і складається з двох кроків: прогнозування та оновлення.

На етапі передбачення оцінюється нове положення цілі на наступному кроці за допомогою моделі стану, тоді як на етапі оновлення використовується поточне спостереження для оновлення положення цілі за допомогою моделі спостереження. Етапи передбачення та оновлення виконуються для кожного кадру відео. Ось кілька прикладів цього методу:

а) Фільтр Калмана

Для використання фільтра Калмана [68] у супроводі об’єкта необхідно розробити динамічну модель руху цілі. Фільтр Калмана використовується для оцінки положення лінійної системи, припускаючи, що помилки є Гаусовими. У багатьох випадках динамічні моделі є нелінійними, тому в цьому випадку фільтр Калмана не використовується, а використовуються інші відповідні алгоритми. Одним із таких алгоритмів є розширений фільтр Калмана. Структура використання фільтра Калмана показана на рис. 1.9.

Фільтр Калмана має важливі функції, якими може скористатися відстеження, зокрема:

• передбачення майбутнього розташування об'єкта;

• корекція прогнозу на основі нових вимірювань;

• зменшення шуму, спричинене неправильними виявленнями.

Фільтр Калмана складається з двох етапів - прогнозування та оновлення. На етапі прогнозування попередні моделі використовуються для прогнозування поточної позиції. Етап оновлення використовує поточні вимірювання для корекції положення.

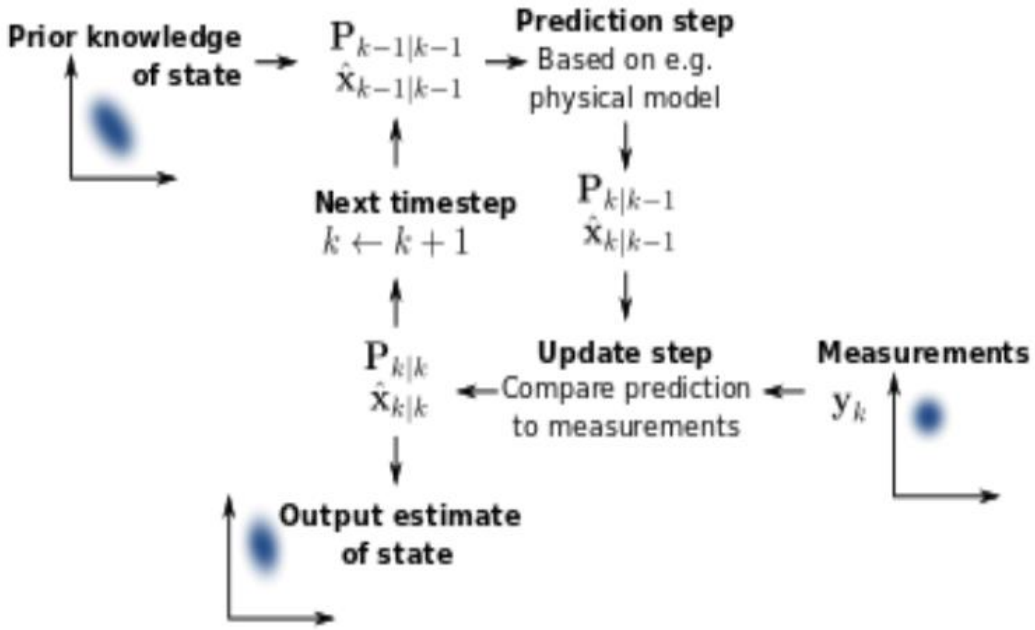

Рис. 1.9. Каркас фільтра Калмана

У деяких роботах для відстеження використовувався фільтр Калмана [69-71]. Наприклад, Gunjal et al. [69] використовували фільтр Калмана для відстеження об’єктів, де кожен об’єкт можна відстежувати з вибраного кадру. У виділеному кадрі можна вибрати об’єкт для відстеження та відслідковувати наступні кадри.

б) Фільтр часток

Більшість проблем із відстеженням є нелінійними. Тому для вирішення таких проблем було розглянуто фільтр часток. Фільтр часток — це рекурсивний статистичний метод розрахунку Монте-Карло, який часто використовується для моделей вимірювання негаусового шуму. Основна ідея фільтра частинок показує розподіл набору часток.

Кожна частка має вагу ймовірності, яка представляє ймовірність вибірки цієї частки на основі функції щільності ймовірності. Однією з проблем цього методу є те, що частки, які мають більшу ймовірність, вибираються кілька разів, і для вирішення цієї проблеми використовується повторна вибірка. Схема використання фільтрів часток показана на рис. 1.10.

Фільтри часток також використовуються в [72, 73]. Утворюється кілька часток, кожна з яких має вагу, яка вказує на якість конкретної частинки. Оцінка бажаної змінної виходить із зваженої суми часток. Він складається з двох важливих кроків: прогнозування та оновлення. Під час передбачення кожна частка модифікується відповідно до моделі стану. На етапі оновлення вага кожної частки переоцінюється новими даними. Метод виключає повторний відбір легких часток і повторює частки з більшою вагою.

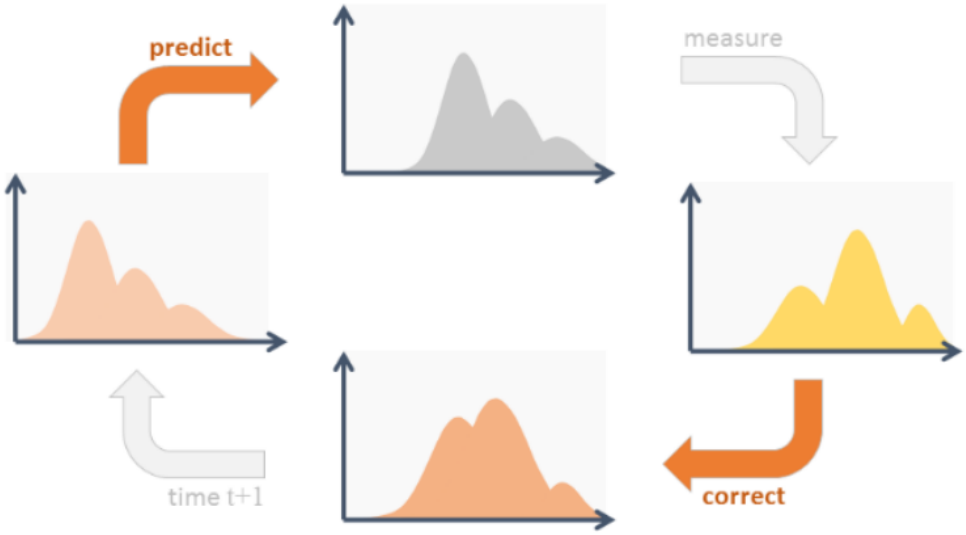

Рис. 1.10. Каркас фільтра часток

Крім того, Ding et al. [74] представляють відстеження об’єктів на основі фільтрації часток на основі гістограми. Основна мета цього методу - дослідити зовнішні зміни, особливо стан і яскравість об'єкта. На рис. 1.11 показана блок-схема цього методу.

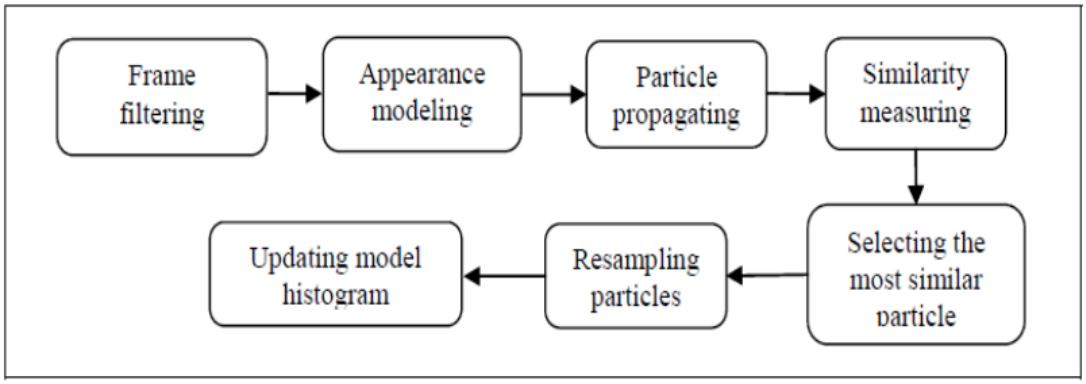

Рис. 1.11. Блок-схема фільтра часток на основі гістограми [74]

Ідея цього методу полягає у використанні черги цільових гістограм FIFO для подолання змін і трансформацій яскравості об’єкта. Гістограма моделі обчислюється на основі середньозваженого значення гістограми та обчислюється рекурсивно.

4. Методи, засновані на навчанні

У методах, заснованих на навчанні, особливості та зовнішній вигляд різних цілей і їх прогнозування вивчаються в наступних кадрах, а потім протягом тестового часу, на основі вивченого, вони можуть виявити об’єкт і відстежити його в наступних кадрах. Методи, засновані на навчанні, часто поділяють на три типи: генеративне, дискримінаційне та навчання з підкріпленням.

а) Дискримінаційні методи

Дискримінаційні трекери зазвичай розглядають стеження як проблему класифікації, яка відрізняє ціль від фону. Дискримінаційне навчання поділяється на дві категорії: поверхневе навчання та глибоке навчання.

Поверхневе навчання

Методи поверхневого навчання – це тип методів, які використовують попередньо визначені функції, і вони не можуть витягувати функції.

Відстеження об’єкта можна розглядати як процес прийняття рішень, у якому класифікація навчається, щоб дізнатися про розрізнення цілі та фону. Після навчання, під час тестування, він вирішує, чи є об’єкт цільовим, чи ні. Ці методи використовують попередньо визначені характеристики різних об’єктів для ідентифікації об’єктів, а потім різні методи, такі як опорні векторні машини [75-76], випадковий ліс [77], дерево рішень [78], використовуються для класифікації.

Tian et al. [79] розглядають відстеження як проблему бінарної класифікації, а Support Vector Machine (SVM) обрано як основну класифікацію. При відстеженні об’єкта, як показано на рис. 1.12, цільова область визначається як позитивні дані, а навколишнє середовище

визначається як негативні дані. Мета полягає в тому, щоб навчити класифікатор SVM, який може класифікувати позитивні та негативні дані в нові кадри. Починаючи з першого кадру, позитивні та негативні приклади використовуються для вивчення класифікації SVM. Тоді область пошуку можна оцінити в наступному кадрі. Нарешті, цільова область вибирається в наступному кадрі з максимальною локальною оцінкою в області пошуку.

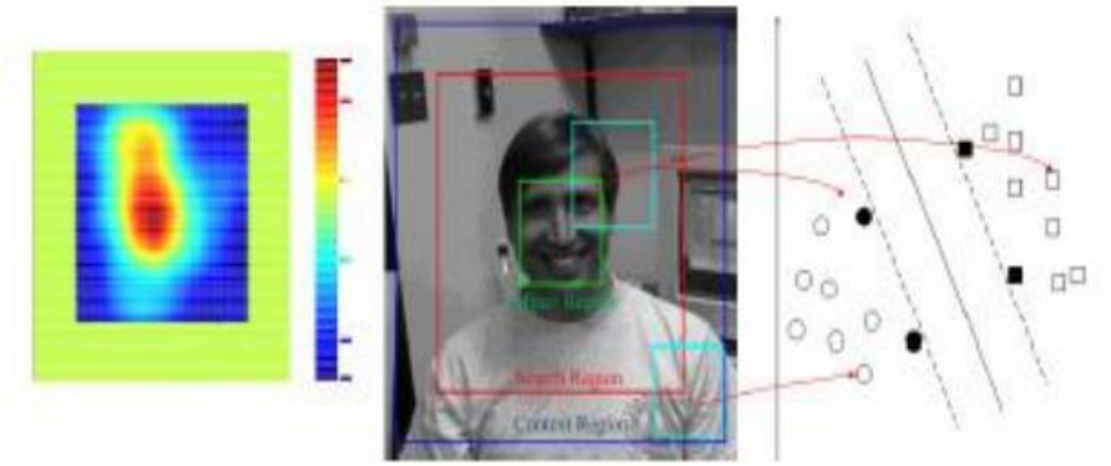

Рис. 1.12. Використання машини опорних векторів у відстеженні [79]

Глибоке навчання

Поверхненве навчання з меншою кількістю рівнів передбачає модель, але глибоке навчання має забагато рівнів. Інша відмінність полягає в тому, що поверхневе навчання потребує важливих і дискримінаційних характеристик, витягнутих професіоналами, але глибоке навчання саме виділяє ці важливі особливості. За останні роки глибоке навчання досягло вражаючих успіхів у різних сферах, включаючи комп’ютерне зір. Однією зі сфер, де глибоке навчання вплинуло на нього та підвищило його точність, є відстеження об’єктів. Методи глибокого навчання поділяються на методи на основі вилучення ознак, і наскрізні методи.

Методи на основі вилучення ознак

Wang et al. показують, що виділення ознак також є дуже важливим питанням у розробці надійного трекера [80]. Оскільки методи глибокого

навчання продемонстрували великі можливості у вилученні ознак і класифікації об’єктів, можна зробити висновок, що використання глибокого навчання може покращити продуктивність відстеження. Завдяки успіху глибоких ознак у класифікації зображень деякі з методів глибокого мережевого відстеження використовуються для виділення ознак, відомих як мережа вилучення ознак.

Ці методи розділяють секції виявлення та відстеження. Виявлення використовує методи глибокого навчання, які можуть виділяти глибокі особливості, але для відстеження ці методи можна класифікувати на дві категорії: відстеження за допомогою класичних методів і відстеження за допомогою глибоких методів.

Bu et al. [81] використовували Faster RCNN, KCF2 і фільтри відстеження Калмана. Chahyati et al. [82] використовували Faster RCNN і сіамську мережу. Agarwal і Suryavanshi [83] використовували Faster RCNN і GOTURN. Ghaemmaghami [84] використовував SSD, YOLO та LSTM, і приклад архітектури цього методу показано на рис. 1.13.

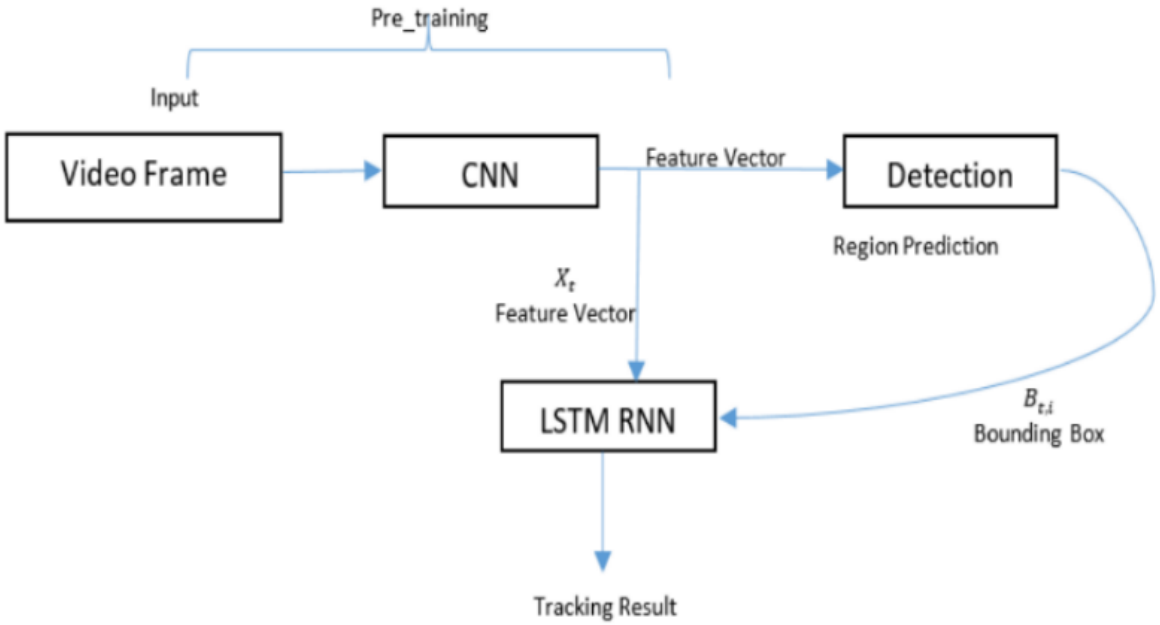

Рис. 1.13. Приклад відстеження на основі вилучення ознак [84]

Наскрізні методи

Наскрізні методи навчають мережу проводити як вилучення ознак, так і оцінку кандидатів. Вони класифікуються за трьома категоріями: сіамські трекери, трекери навчання на основі виправлень (патч-навчання) та трекери на основі графів.

Сіамський трекер

Сіамські мережі мають два входи і один вихід. Він фіксує двох вхідних даних та вимірює подібність між двома зображеннями, щоб визначити, чи існують однакові об’єкти на двох зображеннях чи ні.

Ці типи мереж здатні вивчати подібності та спільні риси. Є деякі документи, які використовують метод сіамського трекера для відстеження, наприклад [85-86]. Прикладом цього трекера є трекер GOTURN. Цей трекер запропонував метод офлайн-навчання нейронних мереж, які можуть виявляти нові об’єкти зі швидкістю 100 кадрів в секунду під час тесту [87]. Цей трекер використовує просту нейронну мережу прямого зв’язку та не вимагає онлайн-навчання. Трекер вивчає загальний зв’язок між рухом об’єкта та зовнішнім виглядом і може використовуватися для відстеження нових об’єктів, яких немає в навчальному наборі. Архітектура цієї мережі показана на рис. 1.14.

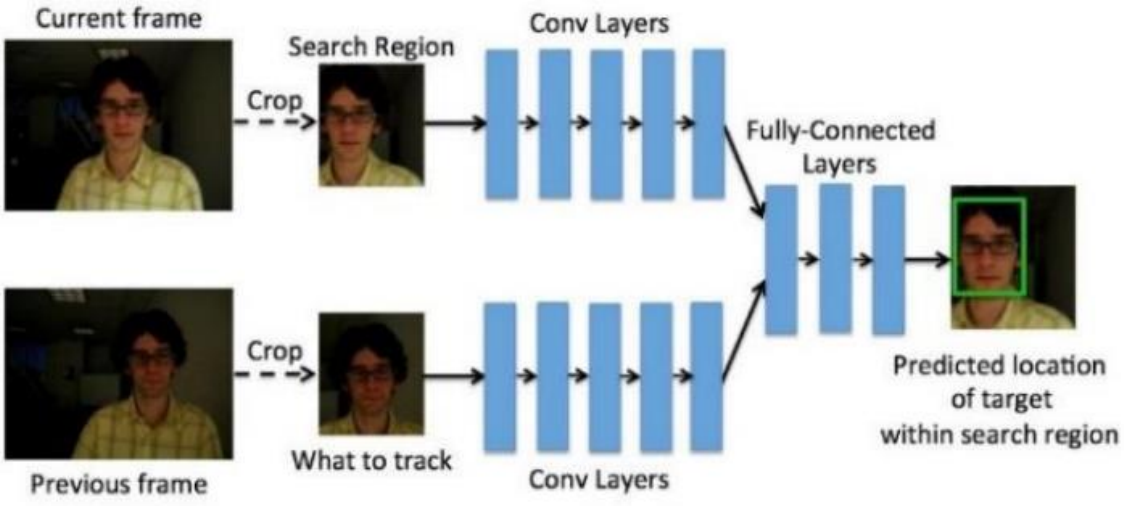

Рис. 1.14. Архітектура трекера GOTURN [87]

У цій архітектурі цільовий об’єкт і область пошуку призначаються шарам згортки. Результатом шару згортки є набір функцій, який є високорівневим представленням зображення. Ці функції надаються як вхідні дані для повністю зв’язаних шарів. Повністю пов’язані шари порівнюють характеристики цільового та поточного об’єктів, і вони навчені враховувати такі проблеми, як оклюзія, обертання тощо. Нарешті, останній шар повністю підключений до виходу за допомогою 4 вузлів, вихід яких є обмежувальною рамкою.

Трекер навчання на основі виправлень

У методі патч-навчання виділяються позитивні та негативні зразки. Потім метод навчає модель нейронної мережі на цих зразках. Нарешті, модель тестується на деяких відібраних зразках, і максимальний відгук вказує на цільову позицію. У роботах [88-89] для відстеження використовувалися методи патч-навчання. Прикладом цього методу є трекер MDNET [90]. Для онлайн-навчання цього трекера, як видно на рис. 1.15, відбирається 50 позитивних зразків і 50 негативних зразків. Потім на основі цих зразків коригується вага повністю з’єднаних шарів.

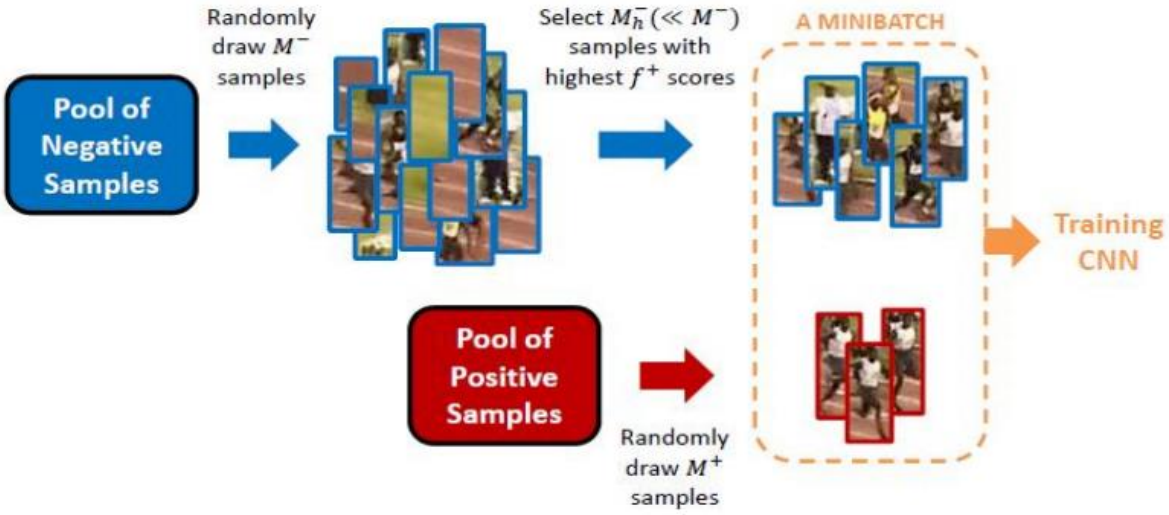

Рис. 1.15. Відстеження на основі патчів у мережі MDNET [90]

Трекер на основі графу

У комп'ютерному зорі теорія графів успішно застосовується для вирішення багатьох завдань. Графи пропонують багате та компактне представлення просторових зв’язків між об’єктами на зображенні, і їх можна використовувати для моделювання багатьох типів зв’язків і процесів.

Є деякі документи, які використовують методи відстеження на основі графів, наприклад [91]. Прикладом цього методу є також трекер TCNN [92]. Алгоритм, запропонований у трекері TCNN, використовує згорточні нейронні мережі для цільового представлення, оскільки кілька CNN співпрацюють, щоб оцінити цільові стани та визначити оптимальні шляхи для онлайнового оновлення моделі в дереві. Підтримка кількох CNN у різних гілках деревовидної структури підходить для протидії багатомодовому цільовому представленню та підтримки надійності моделі шляхом плавного оновлення вздовж шляхів дерева. Оскільки кілька CNN спільно використовують усі параметри в шарах згорнутого шару, він використовує переваги кількох моделей, зберігаючи простір пам’яті та запобігаючи додатковим оцінкам мережі. Остаточний стан цілі оцінюється шляхом вибірки цільових кандидатів у всіх випадках у попередньому кадрі та визначення найкращої вибірки з точки зору середнього зваженого балу з набору активних CNN. На рис. 1.16 показано оцінку цільового стану та методи оновлення моделі методу TCNN.

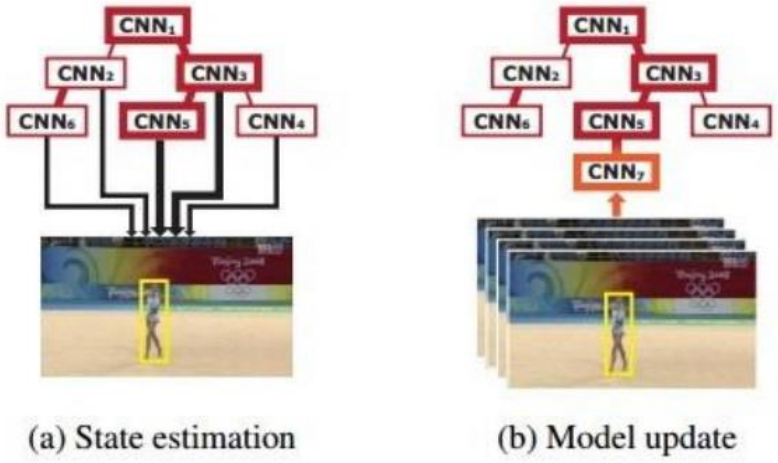

Рис. 1.16. Архітектура методу TCNN [92]

Ширина чорної стрілки вказує на вагу CNN для оцінки цільового стану, а ширина червоного краю вказує на залежність між двома CNN. Ширина прямокутної рамки означає надійність CNN, пов’язану з цією рамкою.

Коли дається новий кадр, вибірки-кандидати будуються навколо цільової позиції, оціненої в попередньому кадрі, і ймовірність кожної вибірки обчислюється на основі середньозваженого кількох CNN. Вага кожного CNN визначається надійністю шляху, який CNN оновив до деревоподібної структури. Цільовий стан у поточному кадрі оцінюється шляхом знаходження кандидата з максимальною ймовірністю.

б) Генеративні методи

Генеративні моделі зовнішнього вигляду в основному зосереджені на тому, як точно підібрати дані з класу об’єктів. Однак на практиці перевірити правильність зазначеної моделі дуже складно. Впроваджуючи механізми онлайн-оновлення, вони поступово вивчають візуальне представлення інформації про область об’єкта переднього плану, ігноруючи вплив фону. Традиційні методи онлайн-навчання застосовуються для відстеження об’єкта шляхом пошуку регіонів, найбільш схожих на цільову модель. Стратегія онлайн-навчання вбудована в структуру відстеження, щоб оновлювати модель зовнішнього вигляду цілі адаптивно у відповідь на варіації зовнішнього вигляду.

в) Навчання з підкріпленням

У проблемі навчання з підкріпленням використовується агент, який взаємодіє з середовищем шляхом проб і помилок і вчиться вибирати оптимальні дії для досягнення мети.

Є праці, які використовують метод навчання з підкріпленням для відстеження, наприклад [93-94]. Як показано на рис. 1.17, ця мережа розділена на дві частини; перша частина — мережа відповідності, а друга — мережа політики.

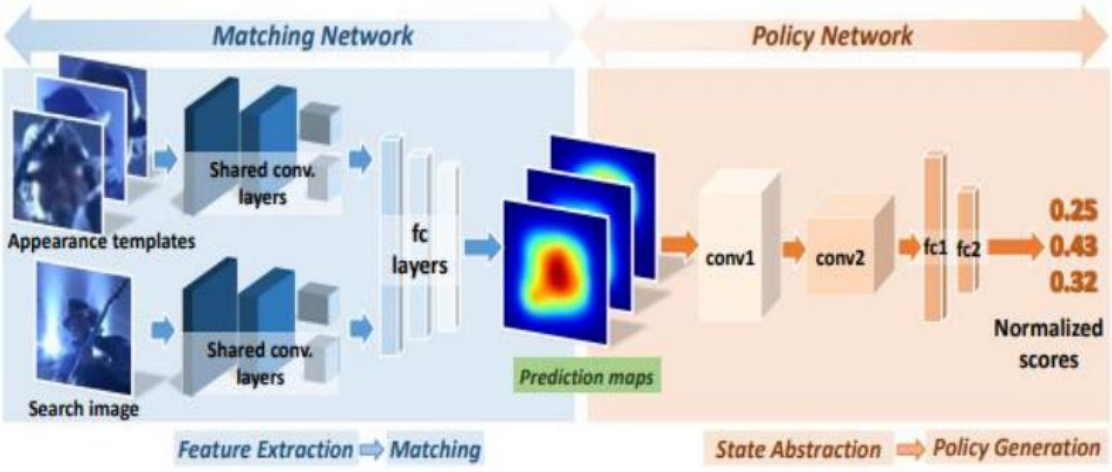

Рис. 1.17. Приклад відстеження об’єкта навчання з підкріпленням [94]

Якщо припустити, що мережі навчені, а їхні ваги постійні, відстеження можна виконувати в бажаній послідовності. Для певного кадру він обрізає пошукове зображення на основі попередньої інформації обмежувальної рамки цілі. Пошукове зображення вирізано з центру попереднього цільового розташування, і його масштаб такий самий, як і попередній цільовий масштаб. Використовуючи шаблони зовнішнього вигляду, отримані з попередньо відстежених кадрів, відповідна мережа генерує карти прогнозів для кожного шаблону. Потім кожна прогнозована карта надходить до мережі політики, де вона генерує бали для кожної прогнозованої карти. Вибирається прогнозна карта з максимальним балом, і відповідний шаблон використовується для відстеження цілі. Позиція на карті передбачення відповідає наступній позиції цілі на пошуковому зображенні. Візерунки з’являються через регулярні проміжки часу під час процесу відстеження.

1.4. Визначення обмежень та постановка задачі

Основні проблеми відстеження зазвичай пов’язані з проблемами на зображенні, які ускладнюють моделям відстеження об’єктів ефективне виявлення зображень. Далі буде описано кілька найпоширеніших проблем із

завданням відстеження об’єктів і методи запобігання цим проблемам, або їх вирішення.

1. Швидкість навчання та відстеження. Передбачається, що алгоритми відстеження об’єктів не тільки точно виявляють і локалізують цільові об’єкти, але й роблять це за найменший проміжок часу. Підвищення швидкості відстеження особливо важливо для моделей відстеження об’єктів у реальному часі. Щоб керувати часом, необхідним для виконання моделі, алгоритм, який використовується для створення моделі відстеження об’єктів, потрібно налаштувати або ретельно вибрати. Fast R-CNN і Faster R-CNN можна використовувати для збільшення швидкості найпоширенішого підходу R-CNN.

Оскільки CNN зазвичай використовуються для виявлення об’єктів, модифікації CNN можуть бути фактором, що відрізняє швидшу модель відстеження об’єкта від повільнішої. Вибір дизайну, окрім структури виявлення, також впливає на баланс між швидкістю та точністю моделі виявлення об’єктів.

2. Фонові відволікання. Фони введених зображень або зображень, які використовуються для навчання моделей відстеження об’єктів, також впливають на точність моделі. Завантажений фон об’єктів, які потрібно відстежувати, може ускладнити виявлення невеликих об’єктів.

Завдяки розмитому або одноколірному фону системі ШІ легше виявляти та відстежувати об’єкти. Занадто насичений фон, який має той самий колір, що й об’єкт, або занадто захаращений, може ускладнити відстеження результатів для маленького об’єкта або об’єкта світлого кольору.

3. Кількість просторових масштабів. Об’єкти, призначені для відстеження, можуть мати різні розміри та пропорції. Ці співвідношення можуть змусити алгоритми відстеження об’єктів повірити, що масштаб об’єктів є більшим або меншим за їхній фактичний розмір. Неправильні

уявлення про розмір можуть негативно вплинути на виявлення або швидкість виявлення. Для боротьби з проблемою різноманітних просторових масштабів програмісти можуть реалізувати такі методи, як карти функцій, блоки прив'язки, піраміди зображень і піраміди функцій.

Блоки прив'язки — це набір обмежувальних рамок із заданою висотою та шириною. Рамки призначені для отримання масштабу та співвідношення сторін цільових об'єктів. Вони вибираються на основі середнього розміру об'єктів у заданому наборі даних. Блоки прив'язки дозволяють виявляти різні типи об'єктів без чергування координат обмежувальної рамки під час локалізації.

Карта функцій — це вихідне зображення шару, коли згорточна нейронна мережа використовується для запису результату застосування фільтрів до цього вхідного зображення. Карти функцій дозволяють глибше зрозуміти особливості, які виявляє CNN. Одноразові детектори повинні брати до уваги проблему кількох масштабів, оскільки вони виявляють об'єкти лише за один прохід через структуру CNN. Це станеться через зменшення виявлення маленьких зображень. Маленькі зображення можуть втратити сигнал під час зменшення дискретизації в шарах об'єднання, коли CNN навчався на низькій підмножині цих менших зображень. Навіть якщо кількість об'єктів однакова, може статися зменшення дискретизації, оскільки CNN не змогла виявити маленькі зображення та підрахувати їх до розміру вибірки. Щоб запобігти цьому, можна використовувати кілька карт функцій, щоб одноразові детектори могли шукати об'єкти в межах шарів CNN, включаючи попередні шари із зображеннями з вищою роздільною здатністю.

Одноразові детектори все ще не є ідеальним варіантом для відстеження невеликих об'єктів через труднощі, які виникають під час виявлення невеликих об'єктів. Тісне групування може виявитися особливо складним.

Зображення та представлення пірамід об'єктів: піраміди об'єктів, також відомі як багаторівневі карти об'єктів через їхню пірамідальну

структуру, є попереднім рішенням для зміни масштабу об’єкта під час використання наборів даних відстеження об’єктів. Отже, піраміди ознак моделюють найбільш корисну інформацію про об’єкти різних розмірів у представленні зверху вниз і, отже, полегшують виявлення об’єктів різного розміру. Такі стратегії, як піраміди зображень і піраміди функцій, корисні для запобігання проблемам масштабування. Піраміда функцій заснована на багатомасштабних картах функцій, які використовують менше обчислювальної потужності, ніж піраміди зображень. Це тому, що піраміди зображень складаються з набору змінених розмірів версій одного вхідного зображення, які потім надсилаються на детектор під час тестування.

4. Оклюзія. У завданнях ШІ бачення з використанням глибокого навчання оклюзія відбувається, коли кілька об’єктів наближаються занадто близько (зливаються) і перекриваються. Це спричиняє проблеми для систем відстеження об’єктів, оскільки закриті об’єкти розглядаються як один або просто неправильно відстежують об’єкт. Система може заплутатися і ідентифікувати спочатку відстежуваний об'єкт як новий. Чутливість до оклюзії запобігає цій помилковій ідентифікації, дозволяючи користувачеві зрозуміти, які частини зображення є найважливішими для системи відстеження об’єктів для класифікації. Чутливість до оклюзії відноситься до показника чутливості мережі до оклюзії в різних областях даних. Це робиться з використанням невеликих підмножин вихідного набору даних.

Основою інформаційної технології стеження за об’єктами у відеоряді було обрано технологію відстеження одного об’єкта SOT. SOT відстежує окрему ціль під час відео, ціль вказується в першому кадрі та має бути виявлена та відстежена в наступних кадрах відео. Трекер повинен мати можливість відстежувати будь-який об’єкт, який йому надано, навіть об’єкт, на якому не було навчено доступну модель класифікації.

Для досягнення мети кваліфікаційної роботи пропонується виконати обґрунтування архітектури інформаційної технології, використавши методи відстеження об’єктів на основі глибокого навчання.

Алгоритми відстеження на основі навчання, і особливо глибоке навчання, є перспективним напрямом в комп’ютерному зорі, і в багатьох застосуваннях воно досягло значного прогресу. Мережі глибокого навчання мають багато рівнів і виділяють різні характеристики об’єкта на кожному шарі. Ці мережі мають хорошу точність у відстеженні об’єктів, оскільки мають багато шарів, але через складність моделі працюють повільніше, ніж неглибокі мережі. Хоча не можна однозначно стверджувати, що глибоке навчання працює найкраще в усіх випадках, але, дослідивши переваги та недоліки обраних методів, можна обгрунтувати, який метод є оптимальним для вирішення поставленої задачі.

1.5. Висновки за розділом 1

Відстеження об’єктів у відеоряді має ряд обмежень, які формувать вимоги до технології відстеження, для досягнення оптимальної точністі та швидкодії. Цими вимогами є надійність, адаптивність та робота в режимі реального часу. Надійність означає, що система стеження може відстежувати ціль навіть у складних умовах, таких як перешкоди на фоні, оклюзія та зміна освітлення. Окрім змін навколишнього середовища, мішенню є зміни, такі як складний і раптовий рух цілі. Щоб вирішити цю проблему, система стеження повинна мати можливість виявляти та відстежувати поточні видимі характеристики цілі. Система, яка працює з послідовністю зображень, повинна мати високу швидкість обробки. Отже, існує потреба в реалізації високопродуктивної інформаційної технології стеження за об’єктами у відеоряді.

У даному розділі була представлена комплексна класифікація методів відстеження об’єктів та обгрунтовано вибір методів на основі глибокого навчання як інструменту реалізації інформаційної технології стеження за об’єктами у відеоряді.

РОЗДІЛ 2
МОДЕЛІ ТА МЕТОДИ РОЗВ'ЯЗАННЯ ЗАДАЧІ

2.1. Моделі глибокого навчання

Глибоке навчання передбачає процес вивчення ієрархічних представлень даних шляхом використання архітектур із кількома прихованими рівнями. З розвитком високопродуктивних обчислювальних засобів все більшої популярності набувають методи глибокого навчання з використанням глибоких нейронних мереж. В алгоритмі глибокого навчання дані передаються через кілька рівнів, причому кожен рівень поступово виділяє функції та передає інформацію на наступний рівень. Початкові шари виділяють низькорівневі характеристики, які потім об'єднуються пізнішими шарами, щоб сформувати повне представлення [95].

У традиційних методах машинного навчання завдання класифікації зазвичай включає послідовний процес, який включає попередню обробку, виділення ознак, ретельний вибір ознак, навчання та класифікацію. Ефективність методів машинного навчання значною мірою залежить від точного вибору функцій, оскільки упереджений вибір функцій може призвести до неправильної класифікації класів. Навпаки, моделі глибокого навчання дозволяють одночасно вивчати та класифікувати, усуваючи потребу в окремих кроках [96]. Ця можливість робить глибоке навчання особливо вигідним для автоматизації вивчення функцій у різноманітних завданнях.

В епоху глибокого навчання було розроблено широкий спектр методів і архітектур. Ці моделі можна розділити на дві основні групи: дискримінаційні (контрольовані) і генеративні (неконтрольовані) підходи. Серед дискримінаційних моделей дві помітні групи - це згорткові нейронні мережі

(CNN) і рекурентні нейронні мережі (RNN) [97]. Генеративні підходи охоплюють різні моделі, такі як генеративні змагальні мережі і автоенкодери.

2.1.1. Багатошаровий персептрон

Модель багаторівневого персептрона (MLP) є типом прямої штучної нейронної мережі, яка служить базовою архітектурою для глибокого навчання або глибоких нейронних мереж [97]. Він працює як підхід до навчання під наглядом. MLP складається з трьох рівнів: вхідного рівня, вихідного рівня та одного або кількох прихованих рівнів. Це повністю зв'язана мережа, тобто кожен нейрон одного шару з'єднаний з усіма нейронами наступного шару.

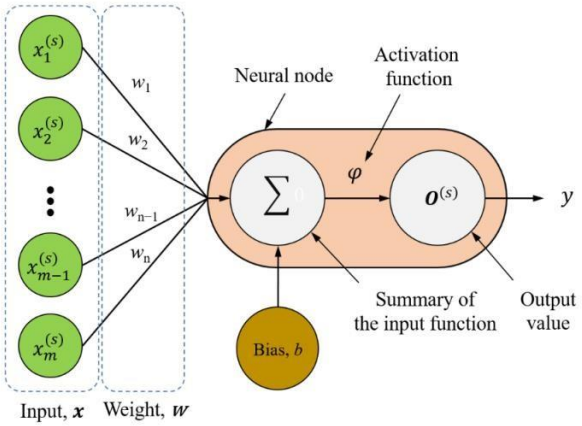

Рис. 2.1. Однонейронна модель персептрона [98].

У MLP вхідний рівень отримує вхідні дані та виконує нормалізацію ознак. Приховані шари, кількість яких може бути різною, обробляють вхідні сигнали. Вихідний рівень приймає рішення або передбачає прогнози на основі обробленої інформації [99]. На рис. 2.1 зображено однонейронну модель персептрона, де функція активації φ (рівняння 2.1) є нелінійною функцією, яка використовується для відображення функції підсумовування $(xw + b)$ на вихідне значення y.

$$y = \varphi(xw + b) \tag{2.1}$$

У рівнянні 2.1 члени *x*, *w*, *b* і *y* представляють вхідний вектор, ваговий вектор, зміщення та вихідне значення відповідно [98]. Рис. 2.2 ілюструє структуру моделі багатошарового персептрона.

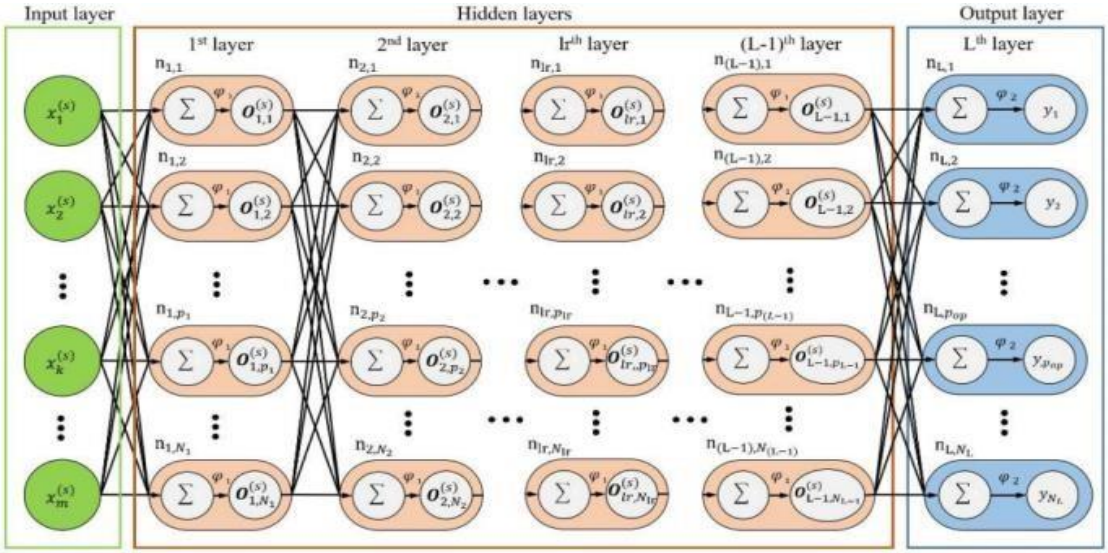

Рис. 2.2. Структура багатошарового персептрона [98].

2.1.2. Згорткові нейронні мережі

Згорткові нейронні мережі (CNN) — це потужний клас моделей глибокого навчання, які широко застосовуються в різних задачах, включаючи виявлення об’єктів, розпізнавання мови, комп’ютерне бачення, класифікацію зображень і біоінформатику. CNN — це нейронні мережі прямого зв’язку, які використовують згорточні структури для вилучення ознак із даних [100]. На відміну від традиційних методів, CNN автоматично вивчають і розпізнають ознаки з даних без необхідності ручного виділення ознак людьми [100]. Дизайн CNN натхненний візуальним сприйняттям. Основні компоненти CNN включають згортковий рівень, рівень об’єднання та повністю зв’язаний рівень [101]. На рис. 2.3 представлена типова архітектура CNN для завдань класифікації зображень [102].

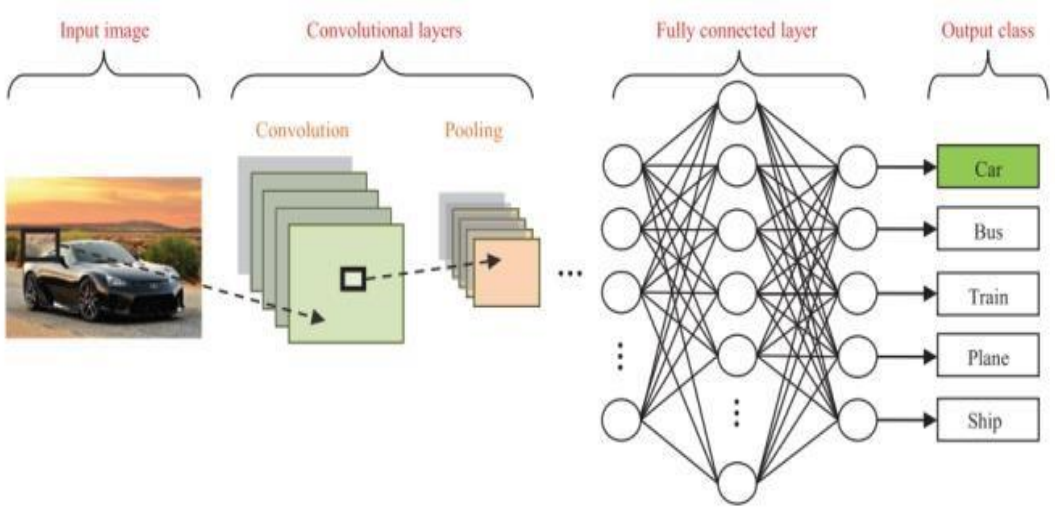

Рис. 2.3. Конвеєр CNN для класифікації зображень [102].

Згортковий рівень: згортковий рівень є ключовим компонентом CNN. Через кілька згорткових шарів операція згортки витягує різні характеристики з вхідних даних. У класифікації зображень нижні шари, як правило, фіксують основні характеристики, такі як текстура, лінії та краї, тоді як вищі рівні виділяють більш абстрактні характеристики [103]. Згортковий рівень містить ядра згортки, які можна вивчати, які є ваговими матрицями, як правило, однакової довжини, ширини та непарного числа (наприклад, 3x3, 5x5 або 7x7). Ці ядра згортаються з вхідними картами ознак, ковзаючи по областях карти ознак і виконуючи операції згортки [103]. Рис. 2.4. ілюструє схематичну діаграму процесу згортки.

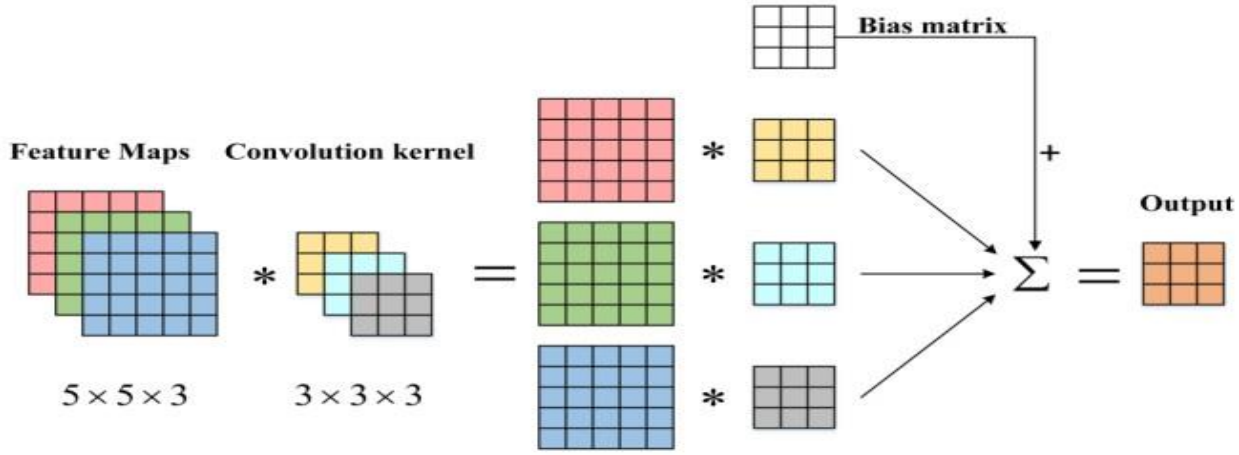

Рис. 2.4. Принципова діаграма процесу згортки [103].

Рівень об’єднання: зазвичай слідуючи за згортковим рівнем, рівень об’єднання зменшує кількість з’єднань у мережі шляхом зменшення вибірки та зменшення розмірності вхідних даних [104]. Його основна мета полягає в тому, щоб полегшити обчислювальний тягар і вирішити проблеми надмірного оснащення. Крім того, рівень об’єднання дозволяє CNN розпізнавати об’єкти, навіть якщо їх форми спотворені або розглядаються під різними кутами, шляхом об’єднання різних розмірів зображення через об’єднання. Операція об’єднання створює вихідні карти функцій, які є більш стійкими до спотворень і помилок в окремих нейронах [105].

Існують різні методи об’єднання, включаючи максимальне об’єднання, середнє об’єднання, об’єднання просторової піраміди, змішане об’єднання, багатомасштабне об’єднання без порядку та стохастичне об’єднання [106]. На рис. 2.5 зображено приклад Max Pooling, де вікно ковзає по входу, а вміст вікна обробляється функцією об’єднання [107].

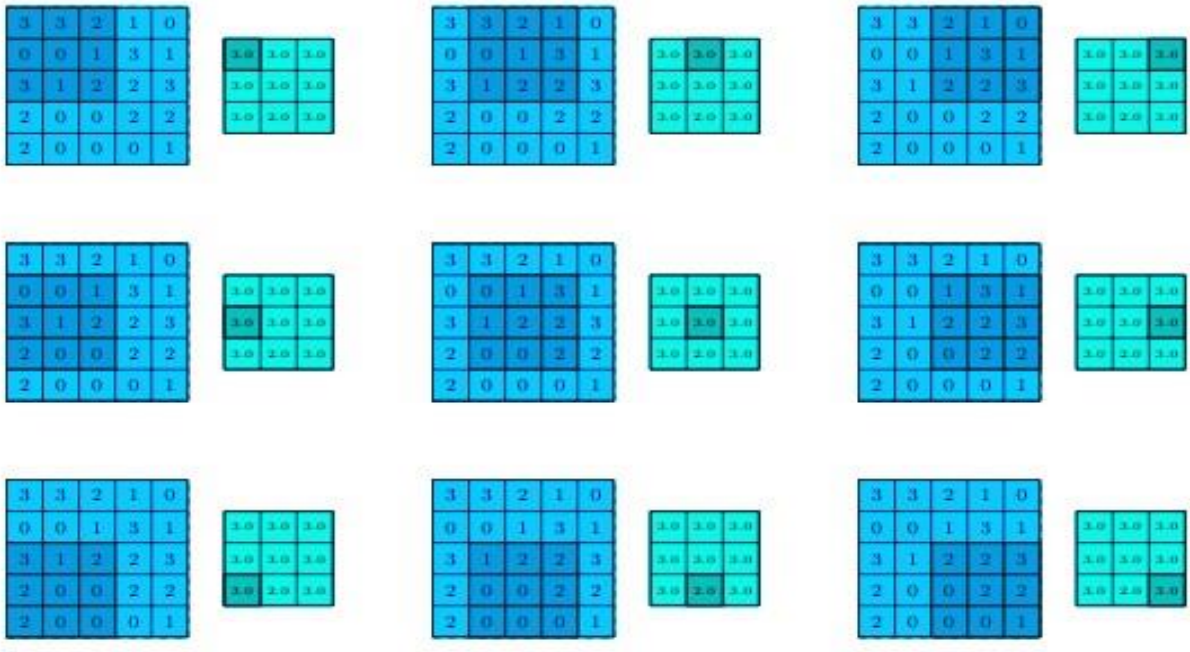

Рис. 2.5. Обчислення вихідних значень операції об’єднання 3×3 max на вході 5×5 [107].

Повністю підключений (FC) рівень: рівень FC зазвичай розташований наприкінці архітектури CNN. У цьому шарі кожен нейрон з'єднаний з усіма нейронами попереднього шару, дотримуючись принципів традиційної багатошарової нейронної мережі персептронів. Рівень FC отримує вхідні дані від останнього шару об'єднання або згорткового шару, який є вектором, створеним шляхом зведення карт функцій. Рівень FC служить класифікатором у CNN, дозволяючи мережі робити прогнози [108].

2.1.3. Рекурентні нейронні мережі

Рекурентні нейронні мережі (RNN) — це клас моделей глибокого навчання, які мають внутрішню пам'ять, що дозволяє їм фіксувати послідовні залежності. На відміну від традиційних нейронних мереж, які обробляють вхідні дані як незалежні сутності, RNN розглядають часовий порядок вхідних даних, що робить їх придатними для завдань, що включають послідовну інформацію [109]. Використовуючи цикл, RNN застосовують ту саму операцію до кожного елемента в серії, при цьому поточне обчислення залежить як від поточного введення, так і від попередніх обчислень [110].

Здатність RNN використовувати контекстну інформацію особливо цінна в таких завданнях, як обробка природної мови, класифікація відео та розпізнавання мови. Однак обмеженням простих РНМ є їхня короткочасна пам'ять, яка обмежує їх здатність зберігати інформацію протягом довгих послідовностей [111]. Щоб подолати це, були розроблені більш просунуті варіанти RNN, включаючи довгострокову короткочасну пам'ять (LSTM) [112], двонаправлену LSTM [113], Gated Recurrent Unit (GRU) [114], двонаправлену GRU [115], байєсівську RNN [116], та інші.

На рис. 2.6 зображено просту рекурентну нейронну мережу, де внутрішня пам'ять (h_t) обчислюється за допомогою рівняння (2.2) [117]:

$$h_t = g(Wx_t + Uh_t + b) \tag{2.2}$$

У цьому рівнянні $g()$ представляє функцію активації (зазвичай гіперболічний тангенс), U і W регульовані вагові матриці для прихованого стану (h), b є членом зсуву, а x позначає вхідний вектор.

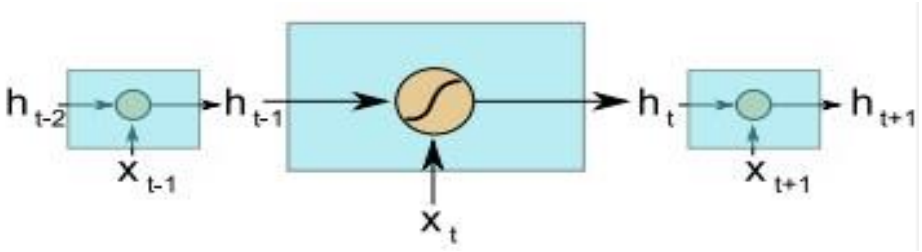

Рис. 2.6. Проста внутрішня робота RNN [117].

RNN виявилися потужними моделями для обробки послідовних даних, використовуючи свою здатність фіксувати залежності з часом. Різні типи моделей RNN, такі як LSTM, двонаправлений LSTM, GRU та двонаправлений GRU, були розроблені для вирішення конкретних завдань у різних програмах.

2.1.4. Сіамські нейронні мережі

Відстеження об’єктів на основі сіамської мережі привернуло значну увагу дослідників завдяки балансу між швидкістю та точністю. SiamFC була першою повністю згортковою сіамською мережею [118], розробленою для відстеження об’єктів, з магістраллю AlexNet. Однак цей підхід може відстежувати ціль лише в попередньо визначеному масштабі. Співвідношення сторін обмежувальної рамки не можна динамічно змінювати. Крім того, AlexNet є відносно поверхневим, оскільки йому не вистачає можливості витягувати глибоку семантичну інформацію, що обмежує продуктивність відстеження. Щоб підвищити продуктивність відстеження об’єктів за допомогою сіамських мереж, дослідники почали використовувати більш глибокі мережі як магістралі. Розробники SiamRPN++ [119] виявили, що нульове доповнення усуває інваріантність

перекладу, і застосували стратегію рівномірної вибірки навколо цілі, щоб відповідати обмеженню інваріантності перекладу. Відповідно, вони успішно впровадили ResNet як магістральну мережу для SiamRPN++, забезпечивши значне покращення продуктивності відстеження. SiamDW [120] представляє залишковий блок із обрізаною структурою та використовує DenseNet як магістральну мережу, що також призводить до значного покращення продуктивності. SiamCAR [121], Ocean [122] і SiamBAN [123] використовують подібні стратегії з більш глибокими та широкими нейронними мережами, які використовуються як магістралі для підвищення продуктивності.

Методи відстеження об’єктів на основі сіамської мережі стикаються з проблемами в обробці цільових змін масштабу. Деякі методи, такі як SA-Siam [124] і TADT [125] використовують пірамідний підхід для створення кількох пошукових зображень у різних масштабах. SiamST [126] запропонував сіамську мережу з просторово-часовим усвідомленням. Liu et al. [127] запропонували алгоритм відстеження невеликих цілей, заснований на системі двонаправленого злиття пірамідальних характеристик. Однак метод піраміди працює лише з попередньо визначеними масштабами зображення. SiamRPN використовує мережу регіональних пропозицій для прогнозування цільової обмежувальної рамки з різними співвідношеннями сторін. Це ефективно вирішує проблему варіацій масштабу та забезпечує значне покращення продуктивності. Однак серія методів SiamRPN вимагає ретельного проектування параметрів, включаючи оптимізацію кількості прив’язок і співвідношення сторін. Засновуючись на виявленні об’єктів без прив’язки, SiamBAN, Ocean і SiamCAR об’єднують класифікаційні підмережі, щоб відрізнити ціль від фону в кожній точці пікселя, а також об’єднують методи на основі центральної точки для прямого прогнозування відстані між центральною точкою та чотири межі потенційних цілей.

Більш глибокі магістральні мережі та оптимізовані деталі можуть постійно покращувати продуктивність методів відстеження об’єктів на основі сіамської мережі. Було досягнуто значного прогресу у вирішенні таких проблем, як цільова варіація масштабу, деформація та оклюзія.

2.1.5. Трансформери

Трансформер [128] спочатку був представлений у задачах машинного перекладу, а потім, завдяки своєму великому успіху та ефективності, його використовували в інших завданнях обробки природної мови. Архітектура трансформера базується на механізмі уваги, який дає змогу моделі зважувати важливість різних частин вхідної послідовності під час обробки, що призводить до покращення потоку інформації та фіксації довгострокових залежностей. Як показано на рис. 2.7, трансформер складається з компонентів кодера та декодера.

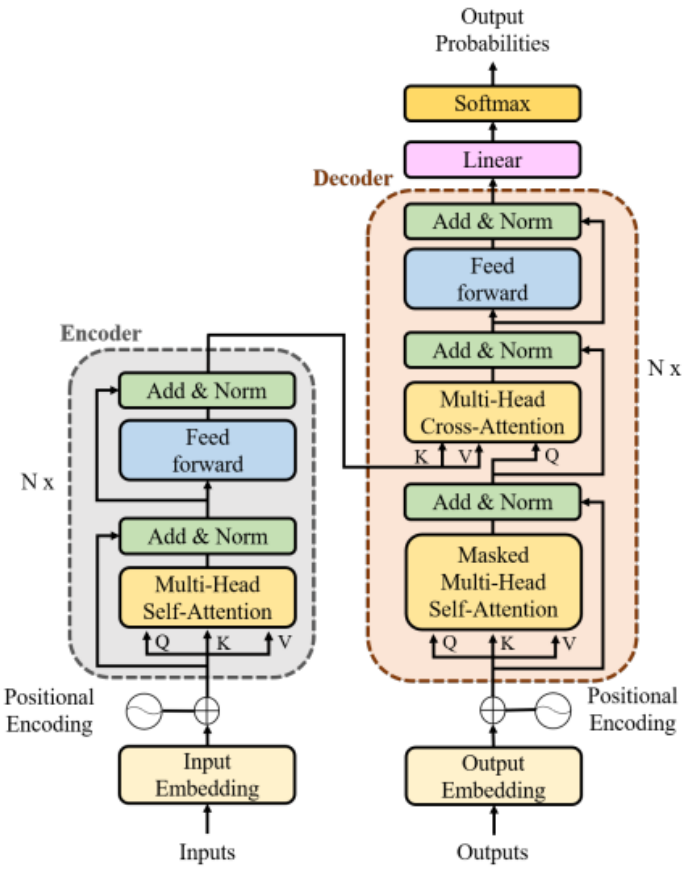

Рис. 2.7. Архітектура трансформера [128]

Компонент кодера складається з N ідентичних шарів кодера, накладених один на одного, а компонент декодера також складається з N ідентичних шарів декодера, які також накладаються один на одного. Компоненти кодера та декодера трансформера зображені ліворуч і праворуч на рис. 2.7 відповідно. В архітектурі трансформера усі рівні кодера мають два підрівні: рівень самоуваги та повністю зв’язаний рівень прямого зв’язку. На додаток до цих двох підрівнів, усі рівні декодера мають рівень уваги кодер-декодер посередині.

Трансформер [128] отримує вхідні дані як векторну послідовність і використовує алгоритм позиційного вбудовування, щоб додати інформацію про позицію кожного маркера в цій послідовності до свого представлення. Після вбудовування вхідні дані передаються на підрівень самоуваги кодера, оскільки це допомагає зафіксувати контекстуальні зв’язки. З іншого боку, на рівні декодера, підрівень уваги кодер-декодер використовується для концентрації на відповідних частинах вхідних даних.

Після самоуважності повністю підключений рівень прямої передачі використовується для вивчення складного представлення особливостей уваги. Він має просту архітектуру з двома лінійними перетвореннями та нелінійною активацією між ними. Цей рівень можна описати як дві згортки з розміром ядра 1. На рівнях кодера та декодера включено залишкове з’єднання, а потім за ним слідує рівень нормалізації. Залишкові зв’язки використовуються, щоб зберегти підказки з вихідних вхідних даних і дозволити моделі вивчати точніші представлення вхідних даних. Після того, як рівень декодера стекує, лінійний рівень використовується для створення вихідних векторів. Нарешті, рівень Softmax використовується для отримання ймовірностей вихідних даних.

Архітектури трансформаторів розроблені на основі концепції уваги, відомої як самоувага. Раніше він обробляв вхідні послідовності паралельно, а не послідовно, як RNN або LSTM. В трансформері механізм самоуваги

фіксує контекстуальні зв’язки між вхідними токенами шляхом обчислення ваги уваги між кожним елементом у вхідній послідовності та всіма іншими елементами в тій же послідовності. Трансформер використовував абстракції «Запит», «Ключ» і «Значення», щоб обчислити увагу вхідної послідовності.

На першому етапі обчислення самоуважності три різні вектори: вектор запиту q, ключовий вектор k і вектор значень v створюються шляхом множення вхідного вектора (x) на три відповідні матриці: W_Q, W_K, та W_V, які пройшли навчання на етапі навчання. Подібним чином усі вхідні вектори упаковуються разом, щоб сформувати вхідну матрицю X, а потім генеруються відповідні матриці «Запит», «Ключ» і «Значення»: Q, K і V відповідно. Як другий крок самоуважності, матриця балів S обчислюється для вхідних даних X шляхом скалярного добутку Q і K таким чином:

$$S = Q \cdot K^T, \tag{2.3}$$

Матриця оцінок S визначає, скільки уваги слід приділяти іншим частинам вхідної послідовності на основі вхідного вектора в конкретній позиції. На наступному кроці матриця показників S нормалізується для отримання більш стабільних градієнтів наступним чином:

$$S_n = S / \sqrt{d_{\mathbf{k}}}, \tag{2.4}$$

де S_n — нормалізована матриця балів, а d_k — розмірність ключового вектора. Потім функція *SoftMax* застосовується до нормалізованої матриці оцінок для перетворення оцінок у ймовірності (P), як зазначено в наступному рівнянні:

$$P = \mathrm{SoftMax}(S_n), \tag{2.5}$$

На останньому кроці самоуважності отримані ймовірності (P) множаться на матрицю значень V, щоб знайти вихідні значення самоуважності (Z), як позначено в наступному рівнянні:

$$Z = P \cdot V, \tag{2.6}$$

Підсумовуючи, весь механізм самоуваги можна об’єднати в одне рівняння таким чином:

$$\text{Attention}(Q, K, V) = \text{SoftMax}\left(\frac{Q \cdot K^T}{\sqrt{d_{\mathbf{k}}}}\right) \cdot V \tag{2.7}$$

Підрівень уваги кодера-декодера на рівні декодера подібний до підрівня самоуваги кодера, за винятком матриці K ключів, а матриця значень V отримується з блоку кодера, а матриця запиту Q отримано з його попереднього шару.

Механізм самоуваги не дуже здатний зосередитися на певному місці введення, не впливаючи на увагу на інших не менш життєво важливих місцях одночасно. Таким чином, продуктивність механізму самоуваги підвищується за допомогою використання кількох голів, і ця техніка відома як багатоголова самоувага. Кілька наборів вагових матриць (W_Q, W_K і W_V) використовуються в багатоголовому механізмі самоуваги. Вони випадковим чином ініціалізуються та навчаються окремо, оскільки вони використовуються для проектування тих самих вхідних даних в інший підпростір. Потім функція уваги обчислюється одночасно для кожної з цих спроектованих версій запитів, ключів і значень для отримання відповідних вихідних значень. На завершальному етапі самоуваги з кількома головами результати всіх головок уваги об’єднуються, а потім множаться на іншу вагову матрицю W_O, яку можна тренувати, щоб отримати самоувагу кількох голов (M_{atten}) наступним чином:

$$M_{\text{atten}} = \text{Concat}(\text{head}_1, ..., \text{head}_n) \cdot W_O \tag{2.8}$$

де $head_i$ - це вихід голови уваги i. Рівень самоуваги з кількома головами використовується для концентрації на різних позиціях введення та представлення введення в різні підпростори.

Завдяки успіху трансформерів у задачах обробки природної мови кілька дослідників спробували застосувати їх у задачах комп’ютерного зору та запропонували різні архітектури. Серед цих моделей Vision Transformer

(ViT) [128] ефективніша за інші з простою архітектурою, як показано на рис. 2.8.

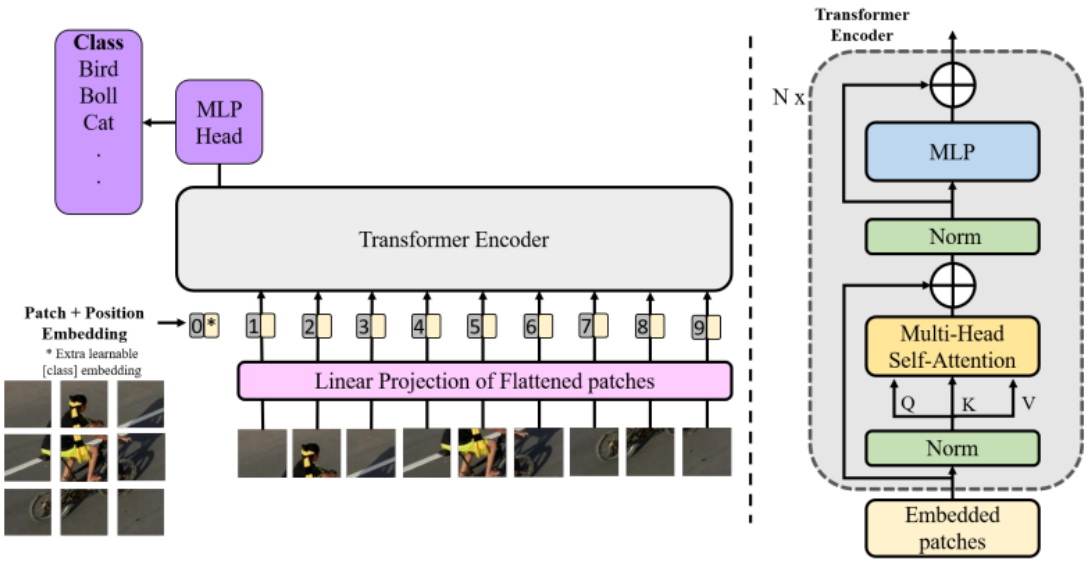

Рис. 2.8. Архітектура Vision Transformer [128]

На початковому етапі ViT вхідне зображення $I \in R^{H \times W \times C}$ розбивається на ділянки однакового розміру з розміром P×P. Тут H, W і C представляють висоту, ширину та кількість каналів вхідного зображення відповідно. Потім патчі сплющуються, щоб утворити 2D-послідовність розміром $(N \times (P^2 \cdot C))$, де N — кількість вилучених патчів із вхідного зображення. На наступному етапі архітектури ViT вбудовуються патчі з інформацією про позицію та клас. Потім вбудовані патчі послідовним чином подаються на набір шарів кодера. Вихідні дані шарів кодера потім отримують і передають у мережу багаторівневого персептрона для створення балів для конкретного класу. Рівні кодера ViT дуже схожі на архітектуру трансформер [128], за винятком того, що замість функції ReLU використовується нелінійна функція GELU [130].

На основі архітектури моделі, методів вилучення функцій та інтеграції функцій останні глибокі трекери можна класифікувати за трьома категоріями [131]: трекери на основі CNN, трекери на основі CNN-трансформер і трекери повністю на основі трансформера. Трекери на основі CNN покладаються

виключно на архітектуру CNN для виділення функцій і виявлення цілей, тоді як трекери на основі CNN-трансфомер і трекери повністю на основі трансформера частково і повністю покладаються на архітектуру трансформера відповідно.

Як правило, архітектури трансформер вимагають величезної кількості навчальних зразків [129] для навчання своїх моделей. Оскільки ціль вказується в першому кадрі послідовності відстеження, отримання великої кількості зразків неможливо у VOT, тому всі трекери, що повністю базуються на трансформері та CNN-трансформері, використовують попередньо навчену мережу та розглядають її як опорну модель. Крім того, деякі з цих трекерів оновлюють цільовий шаблон, щоб фіксувати тимчасові сигнали під час відстеження, а інші – ні. Крім того, вони навчають свої моделі на різних еталонних наборах даних, таких як COCO [132], LaSOT, GOT-10k, TrackingNet і Youtube-BB [133].

Як показано на рис. 2.9 [131], трекери на основі CNN-трансформер використовують магістраль CNN для виділення функцій, а потім використовують архітектуру трансформер для інтеграції функцій.

Трекери повністю на основі трансформера можна далі класифікувати як «Двопотокові двоступеневі» трекери та «Однопотокові одноступеневі» трекери. У двопотокових двоступеневих трекерах два ідентичні конвеєри мережевих гілок (два потоки) використовуються для вилучення характеристик цільового зображення та пошукового зображення. Крім того, у цій категорії трекерів виділення ознак і злиття ознак цільового шаблону та області пошуку виконуються в два розрізнені етапи (двоетапний). З іншого боку, в однопотокових одноетапних трекерах використовується єдиний конвеєр мереж, а виділення та об’єднання функцій виконуються разом на одному етапі.

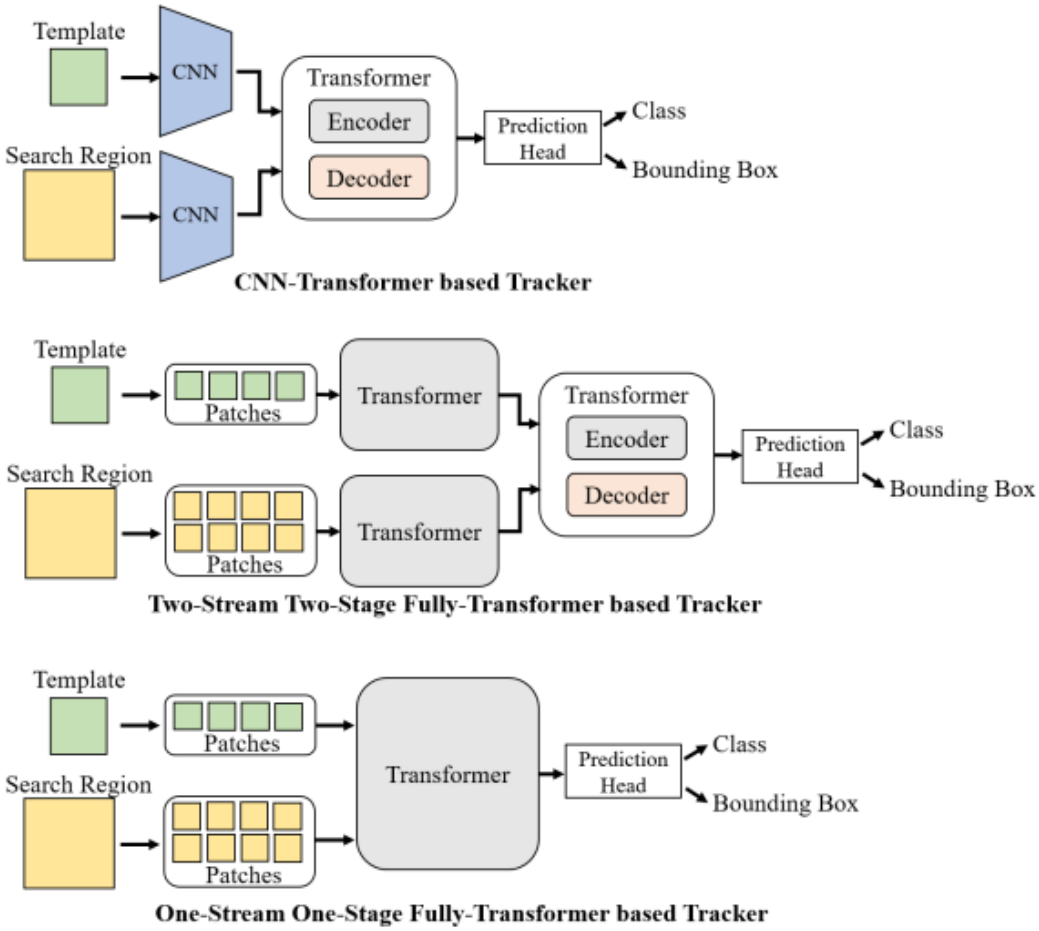

Рис. 2.9. Загальна архітектура глибоких трекерів [131]

2.2. Метрики оцінювання виявлення об’єктів у відеоряді

Існує кілька методів оцінки, які стали популярними в оцінці візуального відстеження. Оцінка алгоритмів відстеження за допомогою кількісних критеріїв складна, оскільки це залежить від багатьох факторів, включаючи точність позиціонування, швидкість відстеження, необхідну пам’ять і так далі. Під час відстеження зазвичай використовуються такі критерії:

Центральна помилка

Одним із найстаріших методів вимірювання ефективності відстеження об’єктів є помилка розташування, яка обчислює середнє значення між евклідовою відстанню від центру передбачуваних цільових точок до центру

істинності на землі в усіх кадрах. І оскільки за кадр заробляється лише один бал, це все ще популярний критерій.

У цього методу є деякі проблеми, наприклад помилка розташування центру вимірює лише різницю в пікселях і не показує розмір і масштаб цілі. Щоб вирішити цю проблему, центральну помилку в кожному кадрі можна розділити на прогнозовану ціль. Однак, незважаючи на нормалізацію, цей метод може мати оманливі результати.

Крім того, коли трекер виходить з ладу і об'єкт втрачає ціль, помилка може бути дуже великою, перевищувати середнє значення та не показувати правильну інформацію [134]. Приклад цього методу показано на рис. 2.10.

Рис. 2.10. Приклад центральної помилки

Перекриття регіонів

Іншим поширеним критерієм оцінки є бал перекриття. Якщо $R_t{}^G$ – це площа базової істини, а $R_t{}^P$ – це площа прогнозованого прямокутника кадру *t*, оцінка перекриття визначається як (2.9).

$$IOU^5 = OS = |R_t{}^G \cap R_t{}^P| / |R_t{}^G \cup R_t{}^P| \quad (2.9)$$

де *t* - індекс кадру. Приклад цього методу показано на рис. 2.11.

Рис. 2.11. Приклад накладення області

З точки зору класифікації пікселів, як показано на рис.2.12, перекриття можна інтерпретувати як (2.10).

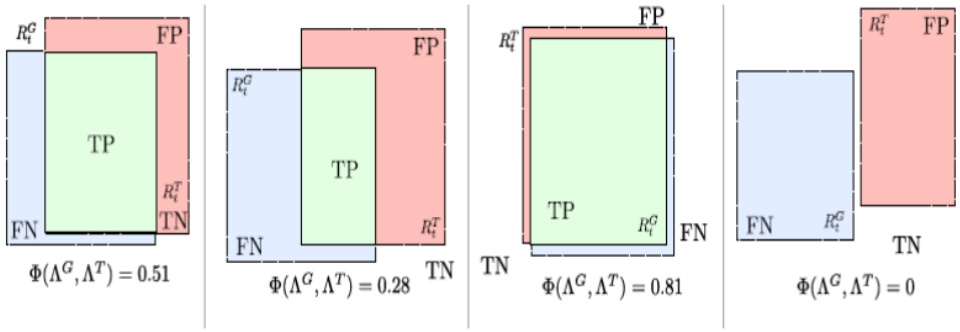

Рис. 2.12. Перекриття області істини з прогнозованою областю [134]

$$|R_t{}^G \cap R_t{}^P| / |R_t{}^G \cup R_t{}^P| = TP/(TP + FN + FP) \quad (2.10)$$

де TP — кількість істинно позитивних результатів, FP — кількість помилкових позитивних результатів, FN — кількість помилкових негативних результатів.

Середній бал перекриття (AOS) можна використовувати як показник ефективності. Цей критерій може варіюватися від 0 до 1. Чим ближче це число до 1, показує, що трекер має кращу продуктивність. Одна з особливостей перекриття областей полягає в тому, що враховує як положення, так і розмір передбачуваної обмежувальної рамки, а також базову правду одночасно, це запобігає виникненню великих помилок під час відстеження помилок, тому це перевага цього методу, ніж центральна

помилка. Фактично, коли трекер об’єктів втрачає ціль, оцінка перекриття стає нульовою.

Довжина відстеження

Іншим показником, який використовується для порівняння трекерів, є довжина стеження. Як показано на рис.2.13, цей критерій повідомляє про кількість успішно відстежених кадрів від початку трекера до першої невдачі.

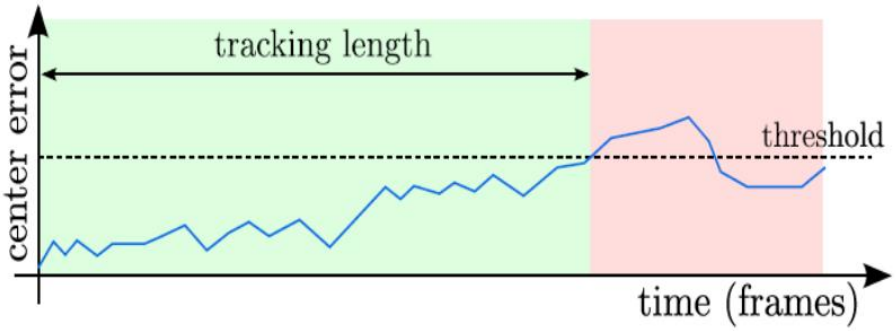

Рис. 2.13. Ефективність відстеження довжини [134]

Цей критерій можна виконати вручну, що буде упередженим і може мати інші результати, якщо його повторювати з тією самою людиною. Для автоматичного визначення критерію відмови можна розмістити порогове значення в центрі або внахлест. Вибір порогу відмови може вплинути на результати порівняння, тому слід вибрати відповідний поріг. Але цей метод також має проблеми. Однією з проблем часткового перегляду є відеопослідовність перед поломкою, а решта кадрів пропускаються [134].

Частота відмов

Критерієм, який в основному вирішує задачі вимірювання довжини колії, є вимірювання інтенсивності відмов. Як показано на рис. 2.14, частота відмов розглядає відстеження як контрольовану систему, в якій людина-оператор перенаправляє трекер після збою.

Рис.2.14. Показники частоти відмов [134]

Необхідна кількість ручних втручань записується та використовується в кожному кадрі. Порівняно з вимірюванням довжини відстеження підхід, заснований на частоті відмов, має перевагу в тому, що в процесі оцінювання використовується вся послідовність і зменшується важливість початку послідовності. Проблема вибору правильного порогу відмови тут більш очевидна, оскільки будь-яка зміна критеріїв вимагає повторного експерименту [134].

Графіки продуктивності

Графіки часто використовуються для візуалізації поведінки трекера під час вивчення кількох трекерів або набору параметрів трекера, оскільки це більш чітко визначає поведінку трекера. Одним із найбільш широко використовуваних методів поведінки трекера сюжету є графік на основі центральної помилки, який показує центральну помилку відповідно до кількості кадрів. Ці типи діаграм зазвичай використовуються для одного відстеження. Інший спосіб - це графік на основі перекриття регіонів. Як згадувалося, визначення відповідного порогу має значний вплив на продуктивність трекера. Щоб уникнути вибору конкретного порогу, результати можна представити на основі діаграми вимірювання порогу. Ці типи діаграм мають схожість із кривою ROC, наприклад монотонність, інтуїтивне візуальне порівняння та подібний алгоритм обчислення. Можна використовувати площу під кривою (AUC), щоб аргументувати продуктивність трекерів [134]. Приклад цього методу показано на рис. 2.15.

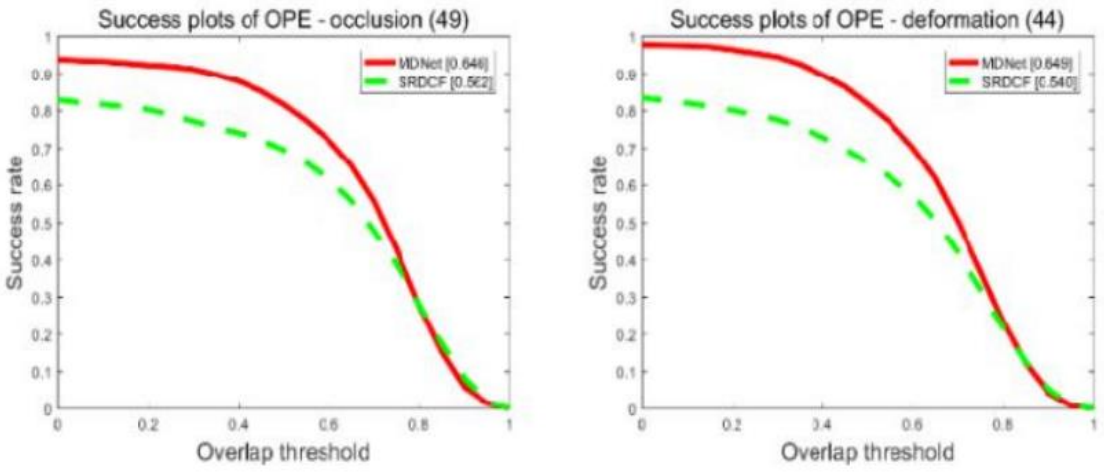

Рис. 2.15. Приклад графіків продуктивності двох трекерів

Оцінка стійкості

Однопрохідне оцінювання (OPE)

Найпоширенішим методом оцінки є ініціалізація алгоритму з режимом наземного об’єкта в першому кадрі та звіт про середню точність або успішність усіх результатів. Цей простий підхід називають однопрохідним оцінюванням, яке має дві проблеми. По-перше, алгоритм відстеження може бути чутливим до початкового значення в першому кадрі, і його продуктивність може відрізнятися в різних початкових режимах або кадрах. По-друге, більшість алгоритмів не мають механізмів повторної ініціалізації, а результати відстеження не надають суттєвої інформації після збою відстеження [135]. Ініціалізація цілі в методі OPE показана на рис. 2.16.

Рис. 2.16. Ініціалізація цілі в першому кадрі в методі OPE

Оцінка тимчасової надійності (TRE)

Кожен алгоритм відстеження в послідовності зображень часто починається з різних кадрів. У кожному тесті алгоритм оцінюється, починаючи з певного кадру, з початковим значенням найсучаснішого стану до кінця послідовності зображень. Щоб отримати оцінку TRE, результати відстеження усереднюються з усіх тестів [135]. Ініціалізація цілі в методі TRE показана на рис. 2.17.

Рис. 2.17. Ініціалізація цілі в методі TRE

Оцінка просторової надійності (SRE)

Точне початкове налаштування цілі часто важливе для алгоритмів відстеження, але під час тесту цього важко досягти через помилки від трекерів або ручне тегування. Щоб оцінити, чи метод відстеження чутливий до початкових помилок, шляхом переміщення або масштабування рамки, яка обмежує істинність цільового об’єкта, створюються різні режими об’єкта [135], використовуються 8 зсувів простору (4 зсуви центру та 4 зсуви кутів) і 4 зміни масштабу. Значення зміни становить 10% від цільового розміру, а коефіцієнт масштабу змінюється від 80% до 120% від базової істини зі збільшенням на 10%. Оцінка SRE є середнім значенням цих 12 оцінок. Ініціалізація цілі в методі SRE показана на рис. 2.18.

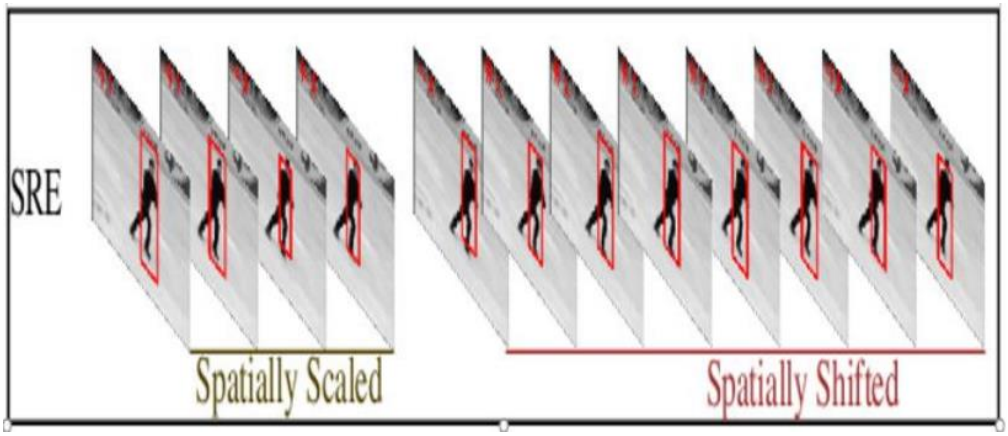

Рис. 2.18. Ініціалізація цілі у методі SRE

2.3. Висновки за розділом 2

У другому розділі були порівняні різні підходи для відстежування об’єктів у відеоряді на основі глибокого навчання. Найпростіше такий функціонал можна реалізувати за допомогою рекурентних мереж різного типу, як з наявністю “пам’яті”, так і без неї, а також з можливим додаванням в архітектуру шарів машинної уваги, і за допомогою CNN. За останні кілька років у VOT було запропоновано велику кількість сіамських трекерів, оскільки вони продемонстрували чудову продуктивність із високою обчислювальною ефективністю. Крім того, широке застосування знайшли архітектури на основі трансформерів, які демонструють високу ефективність у задачах виявлення та відстеження об’єктів.

Щоб визначити оптимальну архітектуру для стеження за об’єктами у відеоряді в режимі реального часу, необхідно експериментально оцінити та порівняти актуальні моделі із числа запропонованих вище трекерів на одній обчислювальній платформі, та знайти найкращий варіант за співвідношенням точності та продуктивності.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ СТЕЖЕННЯ ЗА ОБ'ЄКТАМИ У ВІДЕОРЯДІ

3.1. Опис архітектури стеження за об'єктами у відеоряді

На практиці існує кілька основних архітектур для розв'язання задачі стеження за об'єктами у відеоряді. Зокрема, це MDNet, DeepSORT, та BOTT.

MDNet (Multi-Domain Network) – це модель, яка заснована на сіамській нейронній мережі. Сіамська нейронна мережа – це нейронна мережа, створена з метою вивчення метрик відмінності між двома різними наборами вхідних даних. Для цього вона складається з двох однакових нейронних підмереж. Обидві ці мережі мають однакові ваги та параметри.

DeepSORT (Deep Simple Online and Realtime Tracking) - модель, основною особливістю роботи якої є фільтр Калмана. Фільтр Калмана - це математичний алгоритм, який використовується для відстеження об'єктів у відеоряді. Він передбачає майбутнє розташування об'єкта, на основі попередніх кадрів.

BOTT (Box Only Transformer Tracker) – модель, заснована на архітектурі трансформерів і механізмі уваги. Трансформер - це підхід для обробки вхідних даних у вигляді послідовності з приділенням уваги розташуванню об'єкта в послідовності. Завдяки легшій архітектурі, яка на базовому рівні підтримує розпаралелювання, трансформер виконує все, що роблять рекурентні мережі, тільки швидше та якісніше.

Для порівняння було обрано моделі MDNet і BOTT. Обидві ці моделі відстежують один об'єкт у відеоряді. На відміну від них, DeepSORT створений для відстеження більшої кількості об'єктів одночасно. Однак, він не може бути використаний для онлайн відстеження об'єктів (відстеження в реальному часі), на відміну від двох інших моделей. Через це його

порівняння з іншими двома моделями неактуальне в контексті даної кваліфікаційної роботи. Далі буде розглянуто реалізацію моделей MDNet та BOTT.

3.1.1. Модель MDNet

Архітектуру MDNet можна розбити на 2 частини. Її схему наведено на рис. 3.1.

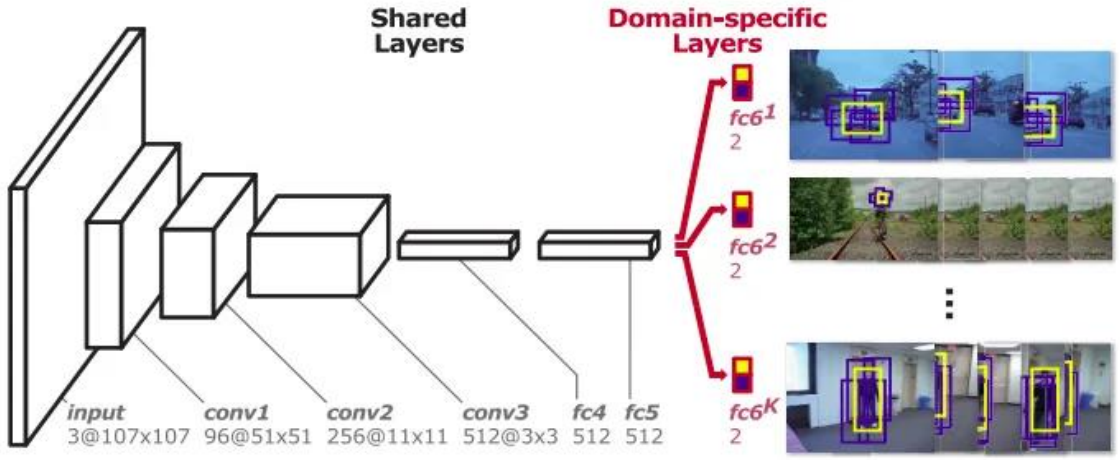

Рис. 3.1. Архітектура MDNet

Перша частина складається з комбінації згорткових і повністю пов'язаних шарів. Ця частина мережі відповідає за виділення особливостей з кадрів вхідного відеоряду. Дуже часто, в моделях, які роблять виділення особливостей, замість того, щоб навчати з нуля, цю частину замінюють на вже навчену модель. Прикладами вже навчених моделей є - VGG16, VGG19 або ResNet. Ці моделі навчені на якісному та об'ємному датасеті. Оскільки в системах класифікації саму класифікацію зазвичай роблять останні шари, якщо відкинути останній шар ResNet, то отримана система буде ефективною виділяти особливості. Однак, з рисунка 3.1 видно, що обробці піддається кошик зображень, який містить n послідовностей зображень. Оскільки мірність датасетів, які було використано під час виконання даної

кваліфікаційної роботи відрізняється від звичайної, то заздалегідь навчану мережу використовувати не можна.

Як можна побачити на рис. 3.1, архітектура використовує невелику кількість шарів. Їх значно менше, ніж у звичайних системах виділення особливостей. Це обумовлено тим, що вона продовжує навчатися після етапу навчання під час передбачення. Оскільки відстежувані об'єкти мають порівняно невеликий розмір в кадрі, збільшення глибини мережі призведе тільки до сповільнення роботи. Через те, що не треба робити детальну класифікацію об'єкта, використовується бінарний класифікатор, який визначає чи є ціль фоном, чи об'єктом. Також коли використовується глибока мережа на основі CNN, це призводить до значного зменшення зображення, що заважає розв'язанню задачі про знаходження розташування об'єкта. З цих причин, глибока нейронна мережа не використовується.

Друга частина архітектури fc6K - це K гілок повністю пов'язаних шарів. У момент навчання використовується техніка навчання з багатьма доменами. Під доменом мається на увазі джерело інформації. В інформації, взятої з кожного домену, є свої особливості. Цими особливостями може бути розмиття, шуми або кардинальна відмінність фону й об'єкта. Завдяки усередненому навчанню кожного з доменів, можна навчити шари з 1 по 5 на варіативнішій інформації, що може допомогти з адаптацією мережі під час реальної роботи. У такому вигляді, друга частина архітектури використовується тільки на етапі навчання. Коли сама модель починає використовуватися для реальної роботи або тестів, K гілок замінюються на один повністю пов'язаний шар fc6. Під час роботи мережі, ваги цього шару ініціалізуються випадковими значеннями. Це означає, що він буде навчатися після початку використання моделі.

Для навчання під час зворотного поширення використовується Stochastic Gradient Descent (SGD). На етапі навчання використовуються такі коефіцієнти швидкості навчання: 0.0001 для згорткових шарів і 0.001 для

повністю зв’язаних шарів. На етапі тестування і реальної роботи використовувались інші коефіцієнти швидкості навчання. На моменті тестування є можливість витратити більше часу на навчання нового шару, оскільки продуктивність на цьому етапі не критична. По завершенню навчання, згорткові шари системи більше не будуть навчатися. Ваги змінюватимуться тільки на шарах із четвертого по шостий. Для четвертого і п'ятого шару, на етапі тестування використовується коефіцієнт швидкості навчання 0.0001, для шостого ж шару використовується коефіцієнт 0.001. На цьому етапі перший кадр треба показати мережі 30 разів, для ініціалізації ваг. Під час реальної роботи програми, перший кадр проганяється 10 разів, для збільшення швидкості роботи, а обидва коефіцієнти швидкості навчання збільшуються в 3 рази для того, щоб мережа швидше реагувала на особливості нового відеоряду.

3.1.2. Модель BOTT

Архітектура моделі BOTT зображена на рис. 3.2.

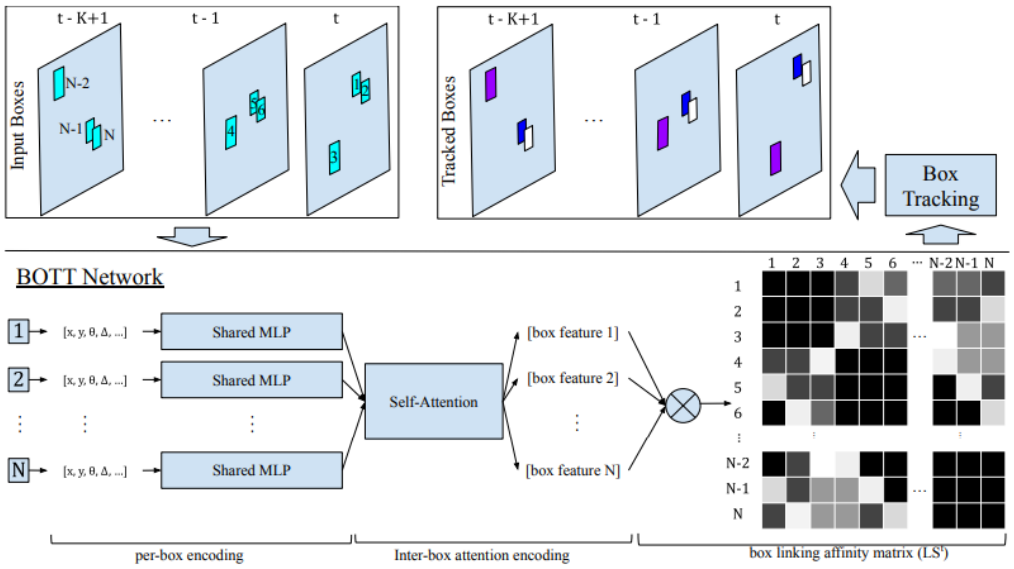

Рис. 3.2. Архітектура BOTT

У кожного кадру з відео система знаходить N обмежувальних рамок. Кожна з рамок виражається через параметри. Кожен параметр, який використовується для визначення рамок наведено далі:

- x, y, z – параметри, що відповідають за розміщення об'єкта;
- w, l, h - параметри відповідають за розміри рамки об'єкта;
- θ - параметр показує кут повороту рамки об'єкта;
- t - параметр показує час. Часом у цьому контексті є позиція кадру у відеоряді;
- $c_1, c_2, \cdots, c_C$ - масив із C елементів. C - кількість класів моделі. Кожен з елементів цього масиву зберігає в собі оцінку ймовірності відношення вмісту рамок об'єкта до конкретного класу моделі.

У такому форматі дані передаються в мережу BOTT. Сама мережа складається з трьох основних блоків:

- single-box feature encoding
- inter-box encoding with self-attention
- linking scores estimation

Далі кожен із блоків розглядається детальніше.

Single-box feature encoding

Основним завданням цього блоку є виокремлення високорівневих особливостей із геометричного представлення вхідних даних кожної з рамок об'єктів. Оскільки вхідні дані можуть значно відрізнятися одне від одного, для зменшення різниці між значеннями робляться деякі перетворення. Для всіх значень x, y, z береться дельта. Від вхідних значень віднімається мінімальне значення x_{min}, y_{min}, z_{min}, серед усіх вхідних рамок цього поточного кадру. Вхідний кут нахилу кодується у вигляді sin(θ), cos(θ). Для параметра t також береться дельта. Від поточного значення кожної з рамок об'єкта віднімається значення t з кадру, який знаходиться посередині вхідного відеоряду.

Таким чином, оновлені вхідні дані мають такий вигляд: x - x_{min}, y - y_{min}, z - z_{min}, w, l, h, sin(θ), cos(θ), t - t_{mid}, c…c_C. Після перетворень, ці параметри передаються в N паралельних мереж спільних MLP. Архітектура MLP наведена на рис. 3.3.

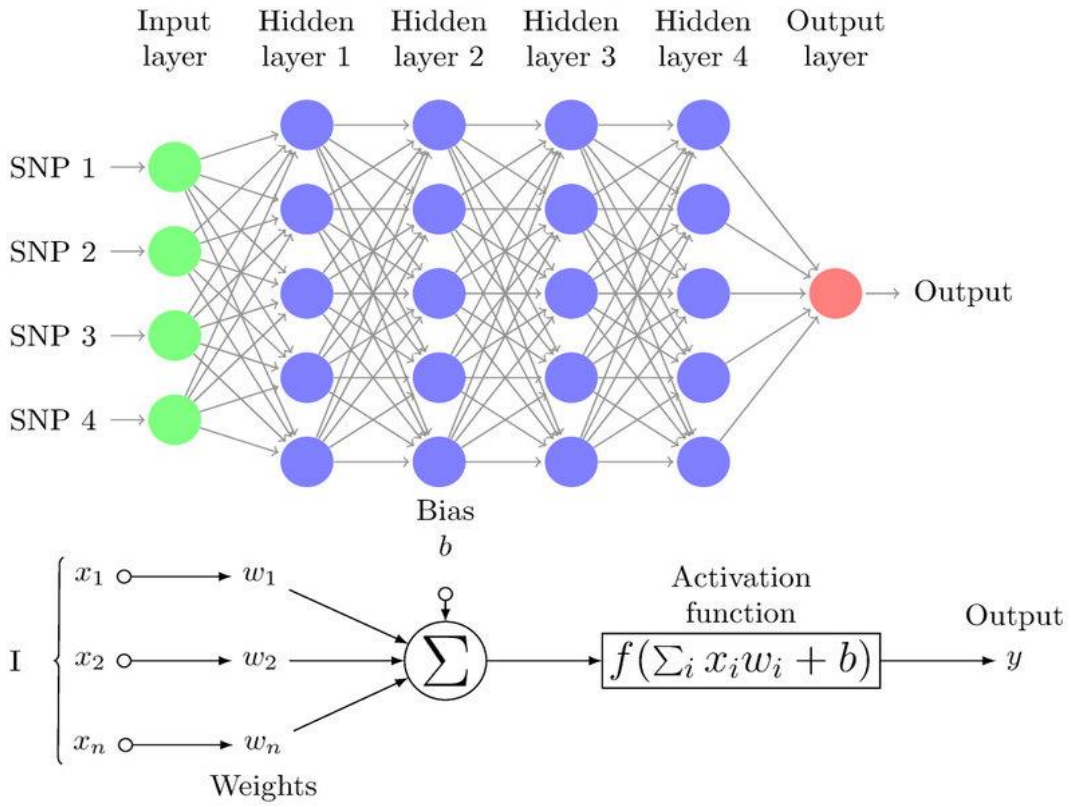

Рис. 3.3. Архітектура багатошарового персептрона

Це звичайна мережа з повністю пов'язаних шарів, головною особливістю якої є те, що кожен із блоків MLP у даній моделі має спільні ваги. Виходом цього блоку є матриця формату кількість рамок на d, де d - це мірність матриці. Вихідні значення цих шарів є вбудуваннями особливостей кожної з меж об'єктів. Вбудування об'єктів - це подання розташування об'єкта у форматі числа. Ця інформація необхідна для роботи механізму уваги трансформера.

Inter-Box Encoding with Self-Attention

Завданням блоку кодування з механізмом самоуваги є знаходження взаємозв'язків між вхідними рамками об'єктів. Архітектура цього блоку стандартна для систем, що використовують трансформер. В основі лежить механізм самоуваги, в якому використовується n голів для паралельного опрацювання вхідної послідовності, за якою слідує мережа прямого розповсюдження. Архітектура цього блоку наведена на рис. 3.4.

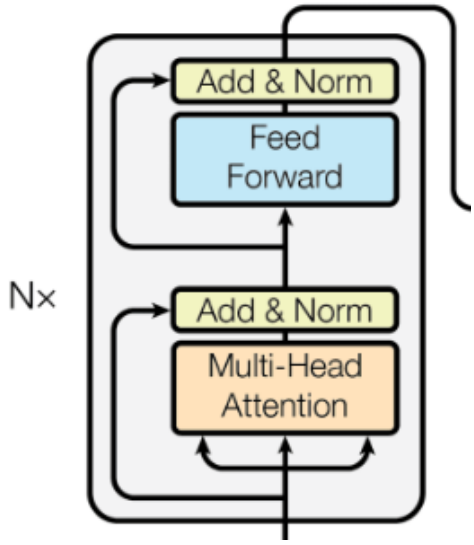

Рис. 3.4. Архітектура блоку кодування з механізмом самоуваги

Цей механізм особливо корисний, коли крім одного відстежуваного об'єкта в кадрі, присутні об'єкти того самого класу, що дає змогу моделі навчатися передбачати рух об'єкта одразу на кількох прикладах.

Linking Score Estimation

Об'єкт, що відстежується, має схожий контекст протягом усього відеоряду. Вбудовування з позиціювання у трансформері допомагає трансформеру розуміти відносне розташування об'єктів на зображенні. У цьому блоці відбувається нормалізація значень вбудовування меж об'єктів до одиничного розміру. Для визначення наскільки об'єкти схожі між собою, використовується скалярний добуток. Результатом скалярного добутку є Linking Score. Таким чином будується матриця, заповнена значеннями від

нуля до одиниці. Це значення показує, який шанс, того, що дві подані межі належать одному й тому самому об’єкту. Виходячи з того, що ближче значення в клітині до одиниці, то вищий шанс, що це один і той самий об'єкт, а що ближче значення до нуля, то шанс менший. Завдяки такому представленню даних, завдання можна привести до бінарної класифікації, що значно зменшує обчислювальні витрати. Матрицю зі значень Linking Score наведено на рис. 3.5.

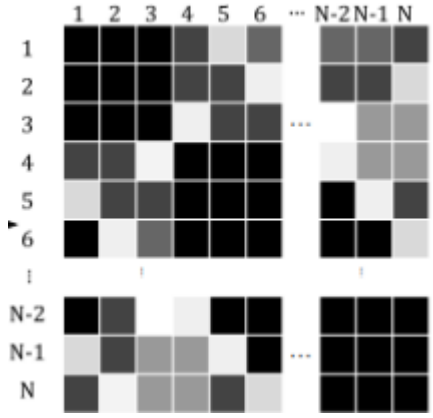

Рис. 3.5. Матриця зі значень Linking Score

3.2. Визначення функції втрати

Оскільки мету роботи мережі зведено до бінарної класифікації, найефективнішою функцією втрати для такого випадку є функція бінарної кросентропії. Для кожного Linking Score функція втрати розраховується за формулою (3.1).

$$L_{ij}^{t} = (1-\beta)\left(1-y_{ij}^{t}\right)log\left(1-LS_{ij}^{t}\right) + \beta y_{ij}^{t} log(LS_{ij}^{t}) \qquad (3.1)$$

Щоб зменшити перенавчання на легких випадках, під час навчання до кросентропії додається маска, яка виключає небажані випадки, які або занадто легкі, або погано впливають на навчання. Це дозволяє уникнути таких випадків:

- міжкласові зв'язки між кордонами;
- зв'язки між кордонами з одного й того самого кадру;

- зв'язки між виявленими межами, які не мають правильного визначеного значення в поданому датасеті. (об'єкти, які були виявлені, хоча в датасеті вони не малися на увазі);
- зв'язки між об'єктами, які перебувають на надто великій відстані один від одного на певному проміжку часу.

Таким чином, функцію втрати можна записати за допомогою формули (3.2):

$$L_{link} = \frac{\Sigma_t \Sigma_{ij} (M_{ij}^t * L_{ij}^t)}{\Sigma_t \Sigma_{ij} M_{ij}^t} \quad (3.2)$$

3.3. Вхідні дані

Обидві моделі були навчені та протестовані на різних датасетах для детальнішого порівняння їхньої продуктивності та точності роботи. Для перевірки було взято такі датасети: VOT2016, VOT2018, VOT2020 і OTB100. Ці датасети використовуються для наукових досліджень у галузі комп'ютерного зору та візуального відстеження об'єктів. Вони складаються з більшої кількості заздалегідь підготовлених зображень та які об'єднані з файлами формату XML. У файлах XML зберігаються дані про межі об'єктів на зображеннях і клас, до якого належить об'єкт. Також, залежно від датасету, у цьому файлі може зберігатися інформація про номер кадру, до якого належить інформація про об'єкт.

3.4. Дослідження продуктивності моделей

Нижче наведено результати порівняння точності моделей, навчених на кожному з представлених датасетів. Як видно з таблиці 3.1, MDNet показує відчутно кращий результат, завдяки використанню глибшої мережі з

виділення особливостей. Однак ці результати наведено без урахування продуктивності систем.

Таблиця 3.1

Результати порівняння точності моделей

Датасет	Точність BOTT	Точність MDNet
VOT2016	65%	86%
VOT2018	58%	80%
VOT2020	55%	70%
OTB100	60%	75%

Модель тестувалася на двох комп'ютерах з різною продуктивністю. Один з більшою продуктивністю, а другий навпаки - з меншою. Продуктивніший комп'ютер має такі характеристики:

- процесор - i7 7700k
- відеокарта - RTX 3090 ti
- ssd - Samsung EVO
- оперативна пам’ять - ddr4 32Gb

Малопродуктивний комп'ютер - Raspberry Pi і в нього такі характеристики:

- процесор - quad-core ARM Cortex-A72
- ssd - 64Gb
- оперативна пам’ять - 4Gb

Навіть при роботі на високопродуктивному комп'ютері в режимі навчання, моделі MDNet не вдається опрацьовувати відеоряд зі швидкістю 20 кадрів на секунду.

Зі свого боку, модель BOTT стабільно показувала понад 120 кадрів на секунду. Під час тестування на високопродуктивному комп'ютері, не вдалося

отримати навіть 1 кадр на секунду від моделі MDNet. Модель BOTT при тих же умовах працює зі швидкістю 30 кадрів на секунду.

3.5. Висновки за розділом 3

Під час детального розгляду різних сценаріїв роботи алгоритмів видно, що в легких сценах обидва алгоритми показують схожу високу точність. Головна різниця в точності з'являється під час обробки навантажених сцен. У таких випадках помічається значне просідання якості відстеження моделі BOTT.

У сценарії зі зникненням об'єкта з кадру, шляхом тимчасового заходження за перешкоду, обидва алгоритми вдало визначають об'єкт наново, при поверненні його у кадр. Однак це працює тільки тоді, коли об'єкт зникає лише на короткий проміжок часу. У випадках, коли об'єкт значно змінює своє розташування, обидві моделі не можуть коректно продовжити відстеження. Однак у випадку з моделлю BOTT є можливість поліпшити цей показник шляхом експериментів із параметрами маски. Головна причина, через яку BOTT втрачає об'єкт, - відкидання під час навчання випадків, коли об'єкти занадто різко набирають відстань. Подальші експерименти з параметрами маски можуть позитивно вплинути на цей сценарій.

Хоча обидві моделі підтримують онлайн роботу, виходячи з результатів тестів, BOTT більше підходить для відстежування в реальному часі. Через низьку продуктивність MDNet не зможе забезпечити навіть базову роботу на більшості пристроїв. Тому його використання суттєво обмежене навіть попри вищу точність.

ВИСНОВКИ

У даній роботі була обґрунтувана інформаційна технологія стеження за об’єктами у відеоряді. Були розглянуті аналітична і програмна складові запропонованої технології.

Основні принципи побудови запропонованої інформаційної технології стеження за об’єктами у відеоряді:

- використання у якості основи моделі BOTT;
- визначення множини параметрів, що використовуються для визначення обмежувальних рамок цільового об’єкта у кожному кадрі відеоряду;
- приведення завдання визначення схожості об’єктів між кадрами до бінарної класифікації, що значно зменшує обчислювальні витрати моделі.

Наукова новизна отриманих результатів кваліфікаційної роботи визначається тим, що вперше обґрунтовано архітектуру інформаційної технології стеження за об’єктами у відеоряді, що підтримує високу точність під час обробки навантажених сцен.

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

Незважаючи на наявний значний прогрес у технологіях відстежування об’єктів, більшість трекерів все ще далекі від оптимального співвідношення показників точності та продуктивності. Існує можливість поліпшити показники точності моделі BOTT шляхом експериментів із параметрами маски. Розгляд цього питання є актуальним напрямком для подальших досліджень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. G. Ciaparrone, F. L. Sanchez, S. Tabik, L. Troiano, R. Tagliaferri, F. Herrera, 2020. Deep learning in video multi-object tracking: A survey, Neurocomputing 381, 61–88. doi: https: //doi.org/10.1016 /j.neucom.2019.11.023.

2. L. Kalake, W. Wan, L. Hou, Analysis based on recent deep learning approaches applied in real-time multi-object tracking: A review, IEEE Access 9 (2021) 32650–32671. doi: 10.1109/ACCESS.2021.3060821.

3. S. Jha, C. Seo, E. Yang, G. P. Joshi, Real time object detection and trackingsystem for video surveillance system, Multimed. Tools Appl. 80 (2021) 3981–3996.

4. S. Bhakar, D. P. Bhatt, V. S. Dhaka, Y. K. Sarma, A review on classifications of tracking systems in augmented reality, J. Phys.: Conf. Ser. 2161 (1) (2022) 012077

5. C. Premachandra, S. Ueda, Y. Suzuki, Detection and tracking of moving objects at road intersections using a 360-degree camera for driver assistance and automated driving, IEEE Access 8 (2020) 135652–135660.

6. R. Pereira, G. Carvalho, L. Garrote, U. J. Nunes, Sort and Deep-SORT based multi-object tracking for mobile robotics: Evaluation with new data association metrics, Appl. Sci. 12 (3) (2022) 1319.

7. R. Chandrakar, R. Raja, R. Miri, U. Sinha, A. K. S. Kushwaha, H. Raja, Enhanced the moving object detection and object tracking for traffic surveillance using RBF-FDLNN and CBF algorithm, Expert Syst. Appl. 191 (2022) 116306.

8. B. T. Naik, M. F. Hashmi, Z. W. Geem, N. D. Bokde, DeepPlayer-Track: Player and referee tracking with jersey color recognition in soccer, IEEE Access 10 (2022) 32494–32509.

9. M. Wallner, D. Steininger, V. Widhalm, M. Schorghuber, C. Beleznai, RGB-D railway platform monitoring and scene understanding for enhanced

passenger safety, in: Proc. Int. Conf. Pattern Recognit., Springer, 2021, pp. 656–671.

10. Y. Liu, C. Sivaparthipan, A. Shankar, Human–computer interaction based visual feedback system for augmentative and alternative communication, Int. J. Speech Technol. 25 (2) (2022) 305–314.

11. B. Liu, J. Huang, L. Yang, C. Kulikowsk, Robust tracking using local sparse appearance model and k-selection, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2011, pp. 1313– 1320.

12. S. Zhang, X. Wang, Human detection and object tracking based on histograms of oriented gradients, in: Proc. Ninth Int. Conf. natural comput., 2013, pp. 1349–1353.

13. L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, P. H. Torr, Staple: Complementary learners for real-time tracking, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 1401–1409.

14. X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, A. V. D. Hengel, A survey of appearance models in visual object tracking, ACM Trans. Intell. Syst. Technol. 4 (4) (2013) 1–48.

15. T. Kokul, A. Ramanan, U. Pinidiyaarachchi, Online multiperson tracking-by-detection method using ACF and particle filter, in: Proc. IEEE Int. Conf. Intell. Comput. Inf. Syst., 2015, pp. 529–536.

16. K. Zhu, X. Zhang, G. Chen, X. Tan, P. Liao, H. Wu, X. Cui, Y. Zuo, Z. Lv, Single object tracking in satellite videos: Deep siamese network incorporating an interframe difference centroid inertia motion model, Remote Sens. 13 (7) (2021) 1298.

17. L. Wang, T. Liu, G. Wang, K. L. Chan, Q. Yang, Video tracking using learned hierarchical features, IEEE Trans. Image Process. 24 (4) (2015) 1424–1435.

18. H. Zhao, G. Yang, D. Wang, H. Lu, Deep mutual learning for visual object tracking, Pattern Recognit. 112 (2021) 107796.

19. T. Kokul, C. Fookes, S. Sridharan, A. Ramanan, U. Pinidiyaarachchi, Gate connected convolutional neural network for object tracking, in: Proc. IEEE Int. Conf. Image Process., 2017, pp. 2602–2606.

20. A. Krizhevsky, I. Sutskever, G. E. Hinton, ImageNet classification with deep convolutional neural networks, Commun. ACM 60 (6) (2017) 84–90.

21. S. M. Marvasti-Zadeh, L. Cheng, H. Ghanei-Yakhdan, S. Kasaei, Deep learning for visual tracking: A comprehensive survey, IEEE Trans. Intell. Transp. Syst. 23 (5) (2022) 3943–3968.

22. W. Walid, M. Awais, A. Ahmed, G. Masera, M. Martina, Realtime implementation of fast discriminative scale space tracking algorithm, J. Real-Time Image Proc. 18 (6) (2021) 2347–2360.

23. T. Xu, Z. Feng, X.-J. Wu, J. Kittler, Adaptive channel selection for robust visual object tracking with discriminative correlation filters, Int. J. Comput. Vis. 129 (5) (2021) 1359–1375.

24. L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, P. H. Torr, Fully-convolutional siamese networks for object tracking, in: Proc. Eur. Conf. Comput. Vis. Workshops, Springer, 2016, pp. 850–865.

25. K. Chen, W. Tao, Once for all: a two-flow convolutional neural network for visual tracking, IEEE Trans. Circuits Syst. Video Technol. 28 (12) (2017) 3377–3386.

26. J. Bromley, I. Guyon, Y. LeCun, E. Sackinger, R. Shah, Signature verification using a ”Siamese” time delay neural network, in: Proc. Advances Neural Inf. Process. Syst., Vol. 6, Morgan Kaufmann, 1993.

27. B. Li, J. Yan, W. Wu, Z. Zhu, X. Hu, High performance visual tracking with siamese region proposal network, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 8971–8980.

28. Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, W. Hu, Distractoraware siamese networks for visual object tracking, in: Proc. Eur. Conf. Comput. Vis., 2018, pp. 101–117.

29. B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, J. Yan, SiamRPN++: Evolution of siamese visual tracking with very deep networks, in: Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2019, pp. 4282–4291.

30. D. Guo, J. Wang, Y. Cui, Z. Wang, S. Chen, SiamCAR: Siamese fully convolutional classification and regression for visual tracking, in: Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 6269–6277.

31. Z. Chen, B. Zhong, G. Li, S. Zhang, R. Ji, Siamese box adaptive network for visual tracking, in: Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 6668–6677.

32. P. Voigtlaender, J. Luiten, P. H. Torr, B. Leibe, Siam R-CNN visual tracking by re-detection, in: Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 6578–6588.

33. D. Guo, Y. Shao, Y. Cui, Z. Wang, L. Zhang, C. Shen, Graph attention tracking, in: Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2021, pp. 9543–9552.

34. S. Cheng, B. Zhong, G. Li, X. Liu, Z. Tang, X. Li, J. Wang, Learning to filter: Siamese relation network for robust tracking, in: Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2021, pp. 4421–4431.

35. M. Ondrasovic, P. Tarabek, Siamese visual object tracking: A survey, IEEE Access 9 (2021) 110149–110172.

36. Achal Dave, Tarasha Khurana, Pavel Tokmakov, Cordelia Schmid, and Deva Ramanan. Tao: A large-scale benchmark for tracking any object. In ECCV, 2020.

37. Patrick Dendorfer, Aljosa Osep, Anton Milan, Konrad Schindler, Daniel Cremers, Ian Reid, Stefan Roth, and Laura Leal-Taixe. Motchallenge: A benchmark for single-camera ´ multiple target tracking. IJCV, 129(4):845–881, 2021.

38. Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine,

et al. Scalability in perception for autonomous driving: Waymo open dataset. In CVPR, 2020.

39. Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In CVPR, 2020.

40. Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixe. Tracking without bells and whistles. In ICCV, 2019

41. Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In ICIP, 2016.

42. Anton Milan, Stefan Roth, and Konrad Schindler. Continuous energy minimization for multitarget tracking. IEEE TPAMI, 36(1):58–72, 2013.

43. Aljosa Osep, Wolfgang Mehner, Paul Voigtlaender, and Bastian Leibe. Track, then decide: Category-agnostic vision based multi-object tracking. In ICRA, 2018.

44. Achal Dave, Pavel Tokmakov, and Deva Ramanan. Towards segmenting anything that moves. In CVPRW, 2019.

45. Aljosa Osep, Paul Voigtlaender, Mark Weber, Jonathon ˇ Luiten, and Bastian Leibe. 4d generic video object proposals. In ICRA, 2020.

46. Yang Liu, Idil Esen Zulfikar, Jonathon Luiten, Achal Dave, Deva Ramanan, Bastian Leibe, Aljosa O ˇ sep, and Laura Leal- ˇ Taixe. Opening up open world tracking. In CVPR, 2022.

47. Object Tracking in Computer Vision (2024 Guide) [Online]. Available: https://viso.ai/deep-learning/object-tracking/

48. Fiaz, M., Mahmood, A., Jung, S.K., 2018. Tracking Noisy Targets: A Review of Recent Object Tracking Approaches. arXiv:1802.03098 [cs].

49. Li, P., Wang, D., Wang, L., Lu, H., 2018. Deep visual tracking: Review and experimental comparison. Pattern Recognition 76, 323–338.

50. Verma, R., 2017. A Review of Object Detection and Tracking Methods. International Journal of Advance Engineering and Research Development 4, 569–578.

51. Soleimanitaleb, Z., Keyvanrad, M.A., Jafari, A., 2019. Object Tracking Methods:A Review, in: 2019 9th International Conference on Computer and Knowledge Engineering (ICCKE). Presented at the 2019 9th International Conference on Computer and Knowledge Engineering (ICCKE), pp. 282– 288. https://doi.org/10.1109/ICCKE48569.2019.8964761

52. Fotouhi, M., Gholami, A.R., Kasaei, S., 2011. Particle filterbased object tracking using adaptive histogram, in: 2011 7th Iranian Conference on Machine Vision and Image Processing. IEEE, pp. 1–5.

53. Li, X., Zheng, N., 2004. Adaptive target color model updating for visual tracking using particle filter, in: 2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583). IEEE, pp. 3105– 3109.

54. Zhao, A., 2007. Robust histogram-based object tracking in image sequences, in: 9th Biennial Conference of the Australian Pattern Recognition Society on Digital Image Computing Techniques and Applications (DICTA 2007). IEEE, pp. 45–52.

55. Barina, D., 2016. Gabor Wavelets in Image Processing. arXiv:1602.03308 [cs].

56. Wagenaar, D.A., Kristan, W.B., 2010. Automated video analysis of animal movements using Gabor orientation filters. Neuroinformatics 8, 33–42.

57. Zhao, Z., Yu, S., Wu, X., Wang, C., Xu, Y., 2009. A multitarget tracking algorithm using texture for real-time surveillance, in: 2008 IEEE International Conference on Robotics and Biomimetics. IEEE, pp. 2150–2155.

58. Li, C., Wei, W., Li, J., Song, W., 2017. A cloud-based monitoring system via face recognition using Gabor and CS LBP features. The Journal of Supercomputing 73, 1532–1546.

59. Kim, D.-S., Kwon, J., 2016. Moving object detection on a vehicle mounted back-up camera. Sensors 16, 23.

60. Hariyono, J., Hoang, V.-D., Jo, K.-H., 2014. Moving object localization using optical flow for pedestrian detection from a moving vehicle. The Scientific World Journal 2014.

61. Yao, R., Lin, G., Xia, S., Zhao, J., Zhou, Y., 2019. Video Object Segmentation and Tracking: A Survey. arXiv preprint arXiv:1904.09172.

62. Cai, Z., Wen, L., Lei, Z., Vasconcelos, N., Li, S.Z., 2014. Robust deformable and occluded object tracking with dynamic graph. IEEE Transactions on Image Processing 23, 5497–5509.

63. Son, J., Jung, I., Park, K., Han, B., 2015. Tracking-by segmentation with online gradient boosting decision tree, in: Proceedings of the IEEE International Conference on Computer Vision. pp. 3056–3064.

64. Zhou, T., Tao, D., 2013. Shifted subspaces tracking on sparse outlier for motion segmentation, in: Twenty-Third International Joint Conference on Artificial Intelligence. Citeseer.

65. Bibby, C., Reid, I., 2008. Robust real-time visual tracking using pixel-wise posteriors, in: European Conference on Computer Vision. Springer, pp. 831–844.

66. Milan, A., Leal-Taixé, L., Schindler, K., Reid, I., 2015. Joint tracking and segmentation of multiple targets, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5397–5406.

67. Schubert, F., Casaburo, D., Dickmanns, D., Belagiannis, V., 2015. Revisiting robust visual tracking using pixel-wise posteriors, in: International Conference on Computer Vision Systems. Springer, pp. 275–288.

68. Najafzadeh, N., Fotouhi, M., Kasaei, S., 2015. Object tracking using Kalman filter with adaptive sampled histogram, in: 2015 23rd Iranian Conference on Electrical Engineering, pp. 781–786. https://doi.org/10.1109/IranianCEE.2015.7146319

69. Gunjal, P.R., Gunjal, B.R., Shinde, H.A., Vanam, S.M., Aher, S.S., 2018. Moving Object Tracking Using Kalman Filter, in: 2018 International Conference On Advances in Communication and Computing Technology (ICACCT). IEEE, pp. 544–547.

70. Taylor, L.E., Mirdanies, M., Saputra, R.P., 2021. Optimized object tracking technique using Kalman filter. arXiv preprint arXiv:2103.05467.

71. Zhao, Z., Yu, S., Wu, X., Wang, C., Xu, Y., 2009. A multitarget tracking algorithm using texture for real-time surveillance, in: 2008 IEEE International Conference on Robotics and Biomimetics. IEEE, pp. 2150–2155.

72. Mihaylova, L., Brasnett, P., Canagarajah, N., Bull, D., 2007. Object tracking by particle filtering techniques in video sequences. Advances and challenges in multisensor data and information processing 8, 260–268.

73. Mondal, A., 2021. Occluded object tracking using object background prototypes and particle filter. Appl Intell 51, 5259– 5279. https://doi.org/10.1007/s10489-020-02047-x

74. Ding, D., Jiang, Z., Liu, C., 2016. Object tracking algorithm based on particle filter with color and texture feature, in: 2016 35th Chinese Control Conference (CCC). Presented at the 2016 35th Chinese Control Conference (CCC), pp. 4031–4036. https://doi.org/10.1109/ChiCC.2016.7553983

75. Malisiewicz, T., Gupta, A., Efros, A.A., 2011. Ensemble of exemplar-SVMs for object detection and beyond., in: Iccv. Citeseer, p. 6.

76. Zhang, S., Yu, X., Sui, Y., Zhao, S., Zhang, L., 2015. Object tracking with multi-view support vector machines. IEEE Transactions on Multimedia 17, 265–278.

77. Saffari, A., Leistner, C., Santner, J., Godec, M., Bischof, H., 2009. On-line random forests, in: Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference On. IEEE, pp. 1393–1400.

78. Xiao, J., Stolkin, R., Leonardis, A., 2015. Single target tracking using adaptive clustered decision trees and dynamic multi-level appearance models, in:

2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Presented at the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4978–4987. https://doi.org/10.1109/CVPR.2015.7299132

79. Tian, M., Zhang, W., Liu, F., 2007. On-line ensemble SVM for robust object tracking, in: Asian Conference on Computer Vision. Springer, pp. 355–364.

80. Wang, N., Shi, J., Yeung, D.-Y., Jia, J., 2015. Understanding and diagnosing visual tracking systems, in: Proceedings of the IEEE International Conference on Computer Vision. pp. 3101– 3109.

81. Bu, F., Cai, Y., Yang, Y., n.d. Multiple Object Tracking Based on Faster-RCNN Detector and KCF Tracker.

82. Chahyati, D., Fanany, M.I., Arymurthy, A.M., 2017. Tracking People by Detection Using CNN Features. Procedia Computer Science 124, 167–172.

83. Agarwal, A., Suryavanshi, S., 2017. Real-Time Multiple Object Tracking (MOT) for Autonomous Navigation. Technical report.

84. Ghaemmaghami, M.P., n.d. Tracking of Humans in Video Stream Using LSTM Recurrent Neural Net- work 50.

85. Jiang, S., Xu, B., Zhao, J., Shen, F., 2021. Faster and Simpler Siamese Network for Single Object Tracking. arXiv:2105.03049 [cs].

86. Wang, X., Li, C., Luo, B., Tang, J., 2018. Sint++: robust visual tracking via adversarial positive instance generation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4864–4873.

87. Held, D., Thrun, S., Savarese, S., 2016. Learning to track at 100 fps with deep regression networks, in: European Conference on Computer Vision. Springer, pp. 749–765.

88. Jung, I., Son, J., Baek, M., Han, B., 2018. Real-time mdnet, in: Proceedings of the European Conference on Computer Vision (ECCV). pp. 83–98.

89. Soleimanitaleb, Z., Keyvanrad, M.A., Jafari, A., 2020. Improved MDNET Tracker in Better Localization Accuracy. Presented at the 2020 10th

International Conference on Computer and Knowledge Engineering (ICCKE)., IEEE.

90. Nam, H., Han, B., 2016. Learning multi-domain convolutional neural networks for visual tracking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4293–4302.

91. Wang, T., Ling, H., 2017. Gracker: A graph-based planar object tracker. IEEE transactions on pattern analysis and machine intelligence 40, 1494–1501.

92. Nam, H., Baek, M., Han, B., 2016. Modeling and propagating cnns in a tree structure for visual tracking. arXiv preprint arXiv:1608.07242.

93. Yun, S., Choi, J., Yoo, Y., Yun, K., Young Choi, J., 2017. Action-decision networks for visual tracking with deep reinforcement learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2711–2720.

94. Choi, J., Kwon, J., Lee, K.M., 2017. Visual tracking by reinforced decision making. arXiv preprint arXiv:1702.06291.

95. Mathew, A., Amudha, P., and Sivakumari, S., 2021. "Deep learning techniques: an overview," Advanced Machine Learning Technologies and Applications: Proceedings of AMLTA 2020, pp. 599-608.

96. Shiri, F. M., Perumal, T., Mustapha, N., Mohamed, R., 2023. A Comprehensive Overview and Comparative Analysis on Deep Learning Models: CNN, RNN, LSTM, GRU. arXiv preprint arXiv: 2305.17473.

97. Sarker, I. H., 2021. "Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions," SN Computer Science, vol. 2, no. 6, p. 420.

98. Ke, K.-C., Huang, M.-S., 2020. "Quality prediction for injection molding by using a multilayer perceptron neural network," Polymers, vol. 12, no. 8, p. 1812.

99. Gaikwad, N. B., Tiwari, V., Keskar, A., and Shivaprakash, N., 2019. "Efficient FPGA implementation of multilayer perceptron for real-time human activity classification," IEEE Access, vol. 7, pp. 26696-26706.

100. Li, Z., Liu, F., Yang, W., Peng, S., and Zhou, J., 2022. "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects," IEEE Trans Neural Netw Learn Syst, vol. 33, no. 12, pp. 6999-7019, doi: 10.1109/TNNLS.2021.3084827.

101. W. Lu, J. Li, J. Wang, and L. Qin, "A CNN-BiLSTM-AM method for stock price prediction," Neural Computing and Applications, vol. 33, pp. 4741-4753, 2021.

102. W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," Neural computation, vol. 29, no. 9, pp. 2352-2449, 2017.

103. L. Chen, S. Li, Q. Bai, J. Yang, S. Jiang, and Y. Miao, "Review of image classification algorithms based on convolutional neural networks," Remote Sensing, vol. 13, no. 22, p. 4712, 2021.

104. J. Gu et al., "Recent advances in convolutional neural networks," Pattern recognition, vol. 77, pp. 354-377, 2018.

105. W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," Neurocomputing, vol. 234, pp. 11-26, 2017.

106. 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, vol. 37, no. 9, pp. 1904-1916, 2015.

107. V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," arXiv preprint arXiv:1603.07285, 2016.

108. L. Alzubaidi et al., "Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions," Journal of big Data, vol. 8, pp. 1-74, 2021.

109. S. Abbaspour, F. Fotouhi, A. Sedaghatbaf, H. Fotouhi, M. Vahabi, and M. Linden, "A Comparative Analysis of Hybrid Deep Learning Models for Human Activity Recognition," Sensors, vol. 20, no. 19, 2020, doi: 10.3390/s20195707.

110. W. Fang, Y. Chen, and Q. Xue, "Survey on research of RNN-based spatio-temporal sequence prediction algorithms," Journal on Big Data, vol. 3, no. 3, p. 97, 2021.

111. H. Apaydin, H. Feizi, M. T. Sattari, M. S. Colak, S. Shamshirband, and K.-W. Chau, "Comparative analysis of recurrent neural network architectures for reservoir inflow forecasting," Water, vol. 12, no. 5, p. 1500, 2020.

112. S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735-1780, 1997.

113. A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," IEEE transactions on pattern analysis and machine intelligence, vol. 31, no. 5, pp. 855-868, 2008.

114. J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," arXiv preprint arXiv:1412.3555, 2014.

115. J. Chen, D. Jiang, and Y. Zhang, "A hierarchical bidirectional GRU model with attention for EEG-based emotion classification," IEEE Access, vol. 7, pp. 118530-118540, 2019.

116. M. Fortunato, C. Blundell, and O. Vinyals, "Bayesian recurrent neural networks," arXiv preprint arXiv:1704.02798, 2017.

117. F. Kratzert, D. Klotz, C. Brenner, K. Schulz, and M. Herrnegger, "Rainfall–runoff modelling using Long Short Term Memory (LSTM) networks," Hydrology and Earth System Sciences, vol. 22, no. 11, pp. 6005-6022, 2018.

118. Wei Cui, Xun Duan, Guangqian Kong, and Huiyun Long, 2023. Siamese network with contrastive learning and adaptive template updating for object tracking, Journal of Electronic Imaging, Vol. 33(1), DOI: 10.1117/1.JEI.33.1.013002.

119. B. Li, W. Wu, and Q. Wang, “SiamRPN++: evolution of Siamese visual tracking with very deep networks,” in Proc. IEEE/CVF Conf. Comput. Vis. and Pattern Recognit., pp. 4282–4291 (2019)

120. Z. Zhang and H. Peng, “Deeper and wider Siamese networks for real-time visual tracking,” in Proc. IEEE/CVF Conf. Comput. Vis. and Pattern Recognit., IEEE, pp. 4591–4600 (2019).

121. D. Guo et al., “SiamCAR: Siamese fully convolutional classification and regression for visual tracking,” in Proc. IEEE/CVF Conf. Comput. Vis. and Pattern Recognit., pp. 6269–6277 (2020).

122. Z. Zhang et al., “Ocean: object-aware anchor-free tracking,” Lect. Notes Comput. Sci. 12366, 771–787 (2020).

123. Z. Chen et al., “SiamBAN: target-aware tracking with Siamese box adaptive network,” IEEE Trans. Pattern Anal. Mach. Intell. 45(5), 5158–5173 (2022).

124. A. He et al., “A twofold Siamese network for real-time object tracking,” in Proc. IEEE Conf. Comput. Vis. and Pattern Recognit., pp. 4834–4843 (2018).

125. X. Li et al., “Target-aware deep tracking,” in Proc. IEEE/CVF Conf. Comput. Vis. and Pattern Recognit., pp. 1369–1378 (2019).

126. H. Zhang et al., “SiamST: Siamese network with spatio-temporal awareness for object tracking,” Inf. Sci. 634, 122–139 (2023).

127. L. Liu et al., “Siamese network with bidirectional feature pyramid for small target tracking,” J. Electron. Imaging 25(5), 053028 (2021).

128. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Proc. Advances Neural Inf. Process. Syst., Vol. 30, Curran Associates, Inc., 2017.

129. A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al., An image is

worth 16x16 words: Transformers for image recognition at scale, in: Proc. Int. Conf. Learn. Represen., 2021.

130. D. Hendrycks, K. Gimpel, Gaussian error linear units (GELUs), arXiv preprint arXiv:1606.08415 (2016).

131. Janani Thangavel, Thanikasalam Kokul, Amirthalingam Ramanan, Subha Fernando, 2023. Transformers in Single Object Tracking: An Experimental Survey, arXiv preprint arXiv: arXiv:2302.11867.

132. T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, C. L. Zitnick, Microsoft COCO: Common objects in context, in: Proc. Eur. Conf. Comput. Vis., Springer, 2014, pp. 740–755.

133. E. Real, J. Shlens, S. Mazzocchi, X. Pan, V. Vanhoucke, YouTube-BoundingBoxes: A large high-precision humanannotated data set for object detection in video, in: Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2017, pp. 5296–5305.

134. Čehovin, L., Leonardis, A., Kristan, M., 2016. Visual object tracking performance measures revisited. IEEE Transactions on Image Processing 25, 1261–1274.

135. Wu, Y., Lim, J., Yang, M.-H., 2015. Object tracking benchmark. IEEE Transactions on Pattern Analysis and Machine Intelligence 37, 1834–1848.

136. 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 24.01.ДА.ПЗ**	Пояснювальна записка	101	
5					
6			Презентація	15	
7					
8			Диск CD с презентацией	1	

Зм.	Ар-куш	№ докум	Підпис	Дата	**ІСТ.КР 24.01.ДА.ПЗ**			
Розроб.		П.Ю. Тимофієнко			**Матеріали кваліфікаційної роботи**	Літ.	Аркуш	Арку-шів
						Н	1	1
Керівник		Г.М. Коротенко				НТУ «ДП», 12; 126м-22-2		
Рецензент								
Н.контр.		Г.М. Коротенко						
Зав.каф.		В.В. Гнатушенко						

ДОДАТОК Б

Програмний код застосунку

modelPyMDNet.py

```
import os
import scipy.io
import numpy as np
from collections import OrderedDict

import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim


def append_params(params, module, prefix):
    for child in module.children():
        for k,p in child._parameters.items():
            if p is None: continue

            if isinstance(child, nn.BatchNorm2d):
                name = prefix + '_bn_' + k
            else:
                name = prefix + '_' + k

            if name not in params:
                params[name] = p
            else:
                raise RuntimeError('Duplicated param name: {:s}'.format(name))


def set_optimizer(model, lr_base, lr_mult, train_all=False, momentum=0.9,
w_decay=0.0005):
    if train_all:
        params = model.get_all_params()
    else:
        params = model.get_learnable_params()
    param_list = []
    for k, p in params.items():
        lr = lr_base
        for l, m in lr_mult.items():
            if k.startswith(l):
                lr = lr_base * m
        param_list.append({'params': [p], 'lr':lr})
    optimizer = optim.SGD(param_list, lr = lr, momentum=momentum,
weight_decay=w_decay)
    return optimizer


class MDNet(nn.Module):
    def __init__(self, model_path=None, K=1):
        super(MDNet, self).__init__()
        self.K = K
        self.layers = nn.Sequential(OrderedDict([
                ('conv1', nn.Sequential(nn.Conv2d(3, 96, kernel_size=7, stride=2),
                                        nn.ReLU(inplace=True),
                                        nn.LocalResponseNorm(2),
                                        nn.MaxPool2d(kernel_size=3, stride=2))),
                ('conv2', nn.Sequential(nn.Conv2d(96, 256, kernel_size=5, stride=2),
                                        nn.ReLU(inplace=True),
                                        nn.LocalResponseNorm(2),
                                        nn.MaxPool2d(kernel_size=3, stride=2))),
                ('conv3', nn.Sequential(nn.Conv2d(256, 512, kernel_size=3, stride=1),
```

```
                                    nn.ReLU(inplace=True))),
                ('fc4',   nn.Sequential(nn.Linear(512 * 3 * 3, 512),
                                    nn.ReLU(inplace=True))),
                ('fc5',   nn.Sequential(nn.Dropout(0.5),
                                    nn.Linear(512, 512),
                                    nn.ReLU(inplace=True)))]))

        self.branches = nn.ModuleList([nn.Sequential(nn.Dropout(0.5),
                                                     nn.Linear(512, 2)) for _ in
range(K)])

        for m in self.layers.modules():
            if isinstance(m, nn.Linear):
                nn.init.normal_(m.weight, 0, 0.01)
                nn.init.constant_(m.bias, 0.1)
        for m in self.branches.modules():
            if isinstance(m, nn.Linear):
                nn.init.normal_(m.weight, 0, 0.01)
                nn.init.constant_(m.bias, 0)

        if model_path is not None:
            if os.path.splitext(model_path)[1] == '.pth':
                self.load_model(model_path)
            elif os.path.splitext(model_path)[1] == '.mat':
                self.load_mat_model(model_path)
            else:
                raise RuntimeError('Unkown model format: {:s}'.format(model_path))
        self.build_param_dict()

    def build_param_dict(self):
        self.params = OrderedDict()
        for name, module in self.layers.named_children():
            append_params(self.params, module, name)
        for k, module in enumerate(self.branches):
            append_params(self.params, module, 'fc6_{:d}'.format(k))

    def set_learnable_params(self, layers):
        for k, p in self.params.items():
            if any([k.startswith(l) for l in layers]):
                p.requires_grad = True
            else:
                p.requires_grad = False

    def get_learnable_params(self):
        params = OrderedDict()
        for k, p in self.params.items():
            if p.requires_grad:
                params[k] = p
        return params

    def get_all_params(self):
        params = OrderedDict()
        for k, p in self.params.items():
            params[k] = p
        return params

    def forward(self, x, k=0, in_layer='conv1', out_layer='fc6'):
        # forward model from in_layer to out_layer
        run = False
        for name, module in self.layers.named_children():
            if name == in_layer:
                run = True
            if run:
                x = module(x)
                if name == 'conv3':
                    x = x.view(x.size(0), -1)
                if name == out_layer:
                    return x

        x = self.branches[k](x)
```

```
        if out_layer=='fc6':
            return x
        elif out_layer=='fc6_softmax':
            return F.softmax(x, dim=1)

    def load_model(self, model_path):
        states = torch.load(model_path)
        shared_layers = states['shared_layers']
        self.layers.load_state_dict(shared_layers)

    def load_mat_model(self, matfile):
        mat = scipy.io.loadmat(matfile)
        mat_layers = list(mat['layers'])[0]

        # copy conv weights
        for i in range(3):
            weight, bias = mat_layers[i * 4]['weights'].item()[0]
            self.layers[i][0].weight.data = torch.from_numpy(np.transpose(weight, (3,
2, 0, 1)))
            self.layers[i][0].bias.data = torch.from_numpy(bias[:, 0])


class BCELoss(nn.Module):
    def forward(self, pos_score, neg_score, average=True):
        pos_loss = -F.log_softmax(pos_score, dim=1)[:, 1]
        neg_loss = -F.log_softmax(neg_score, dim=1)[:, 0]

        loss = pos_loss.sum() + neg_loss.sum()
        if average:
            loss /= (pos_loss.size(0) + neg_loss.size(0))
        return loss


class Accuracy():
    def __call__(self, pos_score, neg_score):
        pos_correct = (pos_score[:, 1] > pos_score[:, 0]).sum().float()
        neg_correct = (neg_score[:, 1] < neg_score[:, 0]).sum().float()
        acc = (pos_correct + neg_correct) / (pos_score.size(0) + neg_score.size(0) +
1e-8)
        return acc.item()


class Precision():
    def __call__(self, pos_score, neg_score):
        scores = torch.cat((pos_score[:, 1], neg_score[:, 1]), 0)
        topk = torch.topk(scores, pos_score.size(0))[1]
        prec = (topk < pos_score.size(0)).float().sum() / (pos_score.size(0) + 1e-8)
        return prec.item()
```

modelBOTT.py

```
import torch
import torch.nn as nn
from functools import partial
import math
import collections.abc
import sys
import os

# dirty insert path #
cur_path = os.path.realpath(__file__)
cur_dir = "/".join(cur_path.split('/')[:-2])
sys.path.insert(0, cur_dir)
from util.misc import trunc_normal_


def to_2tuple(x):
    if isinstance(x, collections.abc.Iterable):
```

```
        return x
    return (x, x)


def drop_path(x, drop_prob: float = 0., training: bool = False):
    """Drop paths (Stochastic Depth) per sample (when applied in main path of residual
blocks).

    This is the same as the DropConnect impl I created for EfficientNet, etc networks,
however,
    the original name is misleading as 'Drop Connect' is a different form of dropout
in a separate paper...
    See discussion: https://github.com/tensorflow/tpu/issues/494#issuecomment-
532968956 ... I've opted for
    changing the layer and argument names to 'drop path' rather than mix DropConnect
as a layer name and use
    'survival rate' as the argument.

    """
    if drop_prob == 0. or not training:
        return x
    keep_prob = 1 - drop_prob
    shape = (x.shape[0],) + (1,) * (x.ndim - 1)  # work with diff dim tensors, not
just 2D ConvNets
    random_tensor = keep_prob + torch.rand(shape, dtype=x.dtype, device=x.device)
    random_tensor.floor_()  # binarize
    output = x.div(keep_prob) * random_tensor
    return output


class DropPath(nn.Module):
    """Drop paths (Stochastic Depth) per sample  (when applied in main path of
residual blocks).
    """
    def __init__(self, drop_prob=None):
        super(DropPath, self).__init__()
        self.drop_prob = drop_prob

    def forward(self, x):
        return drop_path(x, self.drop_prob, self.training)


class Mlp(nn.Module):
    def __init__(self, in_features, hidden_features=None, out_features=None,
act_layer=nn.GELU, drop=0., linear=False):
        super().__init__()
        out_features = out_features or in_features
        hidden_features = hidden_features or in_features
        self.fc1 = nn.Linear(in_features, hidden_features)
        self.dwconv = DWConv(hidden_features)
        self.act = act_layer()
        self.fc2 = nn.Linear(hidden_features, out_features)
        self.drop = nn.Dropout(drop)
        self.linear = linear
        if self.linear:
            self.relu = nn.ReLU(inplace=True)
        self.apply(self._init_weights)

    def _init_weights(self, m):
        if isinstance(m, nn.Linear):
            trunc_normal_(m.weight, std=.02)
            if isinstance(m, nn.Linear) and m.bias is not None:
                nn.init.constant_(m.bias, 0)
        elif isinstance(m, nn.LayerNorm):
            nn.init.constant_(m.bias, 0)
            nn.init.constant_(m.weight, 1.0)
        elif isinstance(m, nn.Conv2d):
            fan_out = m.kernel_size[0] * m.kernel_size[1] * m.out_channels
            fan_out //= m.groups
            m.weight.data.normal_(0, math.sqrt(2.0 / fan_out))
```

```
            if m.bias is not None:
                m.bias.data.zero_()

    def forward(self, x, H, W):
        x = self.fc1(x)
        if self.linear:
            x = self.relu(x)
        x = self.dwconv(x, H, W)
        x = self.act(x)
        x = self.drop(x)
        x = self.fc2(x)
        x = self.drop(x)
        return x


class Attention(nn.Module):
    def __init__(self, dim, num_heads=8, qkv_bias=False, qk_scale=None, attn_drop=0.,
proj_drop=0., sr_ratio=1, linear=False):
        super().__init__()
        assert dim % num_heads == 0, f"dim {dim} should be divided by num_heads
{num_heads}."

        self.dim = dim
        self.num_heads = num_heads
        head_dim = dim // num_heads
        self.scale = qk_scale or head_dim ** -0.5

        self.q = nn.Linear(dim, dim, bias=qkv_bias)
        self.kv = nn.Linear(dim, dim * 2, bias=qkv_bias)
        self.attn_drop = nn.Dropout(attn_drop)
        self.proj = nn.Linear(dim, dim)
        self.proj_drop = nn.Dropout(proj_drop)

        self.linear = linear
        self.sr_ratio = sr_ratio
        if not linear:
            if sr_ratio > 1:
                self.sr = nn.Conv2d(dim, dim, kernel_size=sr_ratio, stride=sr_ratio)
                self.norm = nn.LayerNorm(dim)
        else:
            self.pool = nn.AdaptiveAvgPool2d(7)
            self.sr = nn.Conv2d(dim, dim, kernel_size=1, stride=1)
            self.norm = nn.LayerNorm(dim)
            self.act = nn.GELU()
        self.apply(self._init_weights)

    def _init_weights(self, m):
        if isinstance(m, nn.Linear):
            trunc_normal_(m.weight, std=.02)
            if isinstance(m, nn.Linear) and m.bias is not None:
                nn.init.constant_(m.bias, 0)
        elif isinstance(m, nn.LayerNorm):
            nn.init.constant_(m.bias, 0)
            nn.init.constant_(m.weight, 1.0)
        elif isinstance(m, nn.Conv2d):
            fan_out = m.kernel_size[0] * m.kernel_size[1] * m.out_channels
            fan_out //= m.groups
            m.weight.data.normal_(0, math.sqrt(2.0 / fan_out))
            if m.bias is not None:
                m.bias.data.zero_()

    def forward(self, x, H, W, x_bis=None):
        # x is query
        # x_bis is for k,v (e.g. image features)
        B, N, C = x.shape
        q = self.q(x).reshape(B, N, self.num_heads, C // self.num_heads).permute(0, 2,
1, 3)

        if x_bis is None:
            x_bis = x
```

```
        if not self.linear:
            if self.sr_ratio > 1:
                x_ = x_bis.permute(0, 2, 1).reshape(B, C, H, W)
                x_ = self.sr(x_).reshape(B, C, -1).permute(0, 2, 1)
                x_ = self.norm(x_)
                kv = self.kv(x_).reshape(B, -1, 2, self.num_heads, C //
self.num_heads).permute(2, 0, 3, 1, 4)
            else:
                kv = self.kv(x_bis).reshape(B, -1, 2, self.num_heads, C //
self.num_heads).permute(2, 0, 3, 1, 4)
        else:
            x_ = x_bis.permute(0, 2, 1).reshape(B, C, H, W)
            x_ = self.sr(self.pool(x_)).reshape(B, C, -1).permute(0, 2, 1)

            x_ = self.norm(x_)
            x_ = self.act(x_)
            kv = self.kv(x_).reshape(B, -1, 2, self.num_heads, C //
self.num_heads).permute(2, 0, 3, 1, 4)
        k, v = kv[0], kv[1]

        # attn = (q @ k.transpose(-2, -1)) * self.scale
        attn = torch.matmul(q, k.transpose(-2, -1))*self.scale

        # assert (attn - torch.matmul(q, k.transpose(-2, -1))*self.scale).sum() == 0.0
        attn = attn.softmax(dim=-1)
        attn = self.attn_drop(attn)
        # print(attn.dtype)
        # x = (attn @ v).transpose(1, 2).reshape(B, N, C)
        x = torch.matmul(attn, v).transpose(1, 2).reshape(B, N, C)
        # print(x.dtype)
        # assert (x - torch.matmul(attn, v).transpose(1, 2).reshape(B, N,
C)).sum()==0.0
        x = self.proj(x)
        x = self.proj_drop(x)

        return x


class Block(nn.Module):

    def __init__(self, dim, num_heads, mlp_ratio=4., qkv_bias=False, qk_scale=None,
drop=0., attn_drop=0.,
                 drop_path=0., act_layer=nn.GELU, norm_layer=nn.LayerNorm, sr_ratio=1,
linear=False):
        super().__init__()
        self.norm1 = norm_layer(dim)
        self.attn = Attention(
            dim,
            num_heads=num_heads, qkv_bias=qkv_bias, qk_scale=qk_scale,
            attn_drop=attn_drop, proj_drop=drop, sr_ratio=sr_ratio, linear=linear)
        # NOTE: drop path for stochastic depth, we shall see if this is better than
dropout here
        self.drop_path = DropPath(drop_path) if drop_path > 0. else nn.Identity()
        self.norm2 = norm_layer(dim)
        mlp_hidden_dim = int(dim * mlp_ratio)
        self.mlp = Mlp(in_features=dim, hidden_features=mlp_hidden_dim,
act_layer=act_layer, drop=drop, linear=linear)

        self.apply(self._init_weights)

    def _init_weights(self, m):
        if isinstance(m, nn.Linear):
            trunc_normal_(m.weight, std=.02)
            if isinstance(m, nn.Linear) and m.bias is not None:
                nn.init.constant_(m.bias, 0)
        elif isinstance(m, nn.LayerNorm):
            nn.init.constant_(m.bias, 0)
            nn.init.constant_(m.weight, 1.0)
        elif isinstance(m, nn.Conv2d):
            fan_out = m.kernel_size[0] * m.kernel_size[1] * m.out_channels
```

```
            fan_out //= m.groups
            m.weight.data.normal_(0, math.sqrt(2.0 / fan_out))
            if m.bias is not None:
                m.bias.data.zero_()

    def forward(self, x, H, W, x_bis=None):
        if x_bis is not None:
            x_bis = self.norm1(x_bis)
        x = x + self.drop_path(self.attn(self.norm1(x), H, W, x_bis))
        x = x + self.drop_path(self.mlp(self.norm2(x), H, W))

        return x


class OverlapPatchEmbed(nn.Module):
    """ Image to Patch Embedding
    """

    def __init__(self, img_size=224, patch_size=7, stride=4, in_chans=3,
embed_dim=768):
        super().__init__()
        img_size = to_2tuple(img_size)
        patch_size = to_2tuple(patch_size)

        self.img_size = img_size
        self.patch_size = patch_size
        self.H, self.W = img_size[0] // patch_size[0], img_size[1] // patch_size[1]
        self.num_patches = self.H * self.W
        self.proj = nn.Conv2d(in_chans, embed_dim, kernel_size=patch_size,
stride=stride,
                              padding=(patch_size[0] // 2, patch_size[1] // 2))
        self.norm = nn.LayerNorm(embed_dim)

        self.apply(self._init_weights)

    def _init_weights(self, m):
        if isinstance(m, nn.Linear):
            trunc_normal_(m.weight, std=.02)
            if isinstance(m, nn.Linear) and m.bias is not None:
                nn.init.constant_(m.bias, 0)
        elif isinstance(m, nn.LayerNorm):
            nn.init.constant_(m.bias, 0)
            nn.init.constant_(m.weight, 1.0)
        elif isinstance(m, nn.Conv2d):
            fan_out = m.kernel_size[0] * m.kernel_size[1] * m.out_channels
            fan_out //= m.groups
            m.weight.data.normal_(0, math.sqrt(2.0 / fan_out))
            if m.bias is not None:
                m.bias.data.zero_()

    def forward(self, x):
        x = self.proj(x)
        _, _, H, W = x.shape
        x = x.flatten(2).transpose(1, 2)
        x = self.norm(x)

        return x, H, W


class PyramidVisionTransformerV2(nn.Module):
    def __init__(self, img_size=224, patch_size=16, in_chans=3, num_classes=1000,
embed_dims=[64, 128, 256, 512],
                 num_heads=[1, 2, 4, 8], mlp_ratios=[4, 4, 4, 4], qkv_bias=False,
qk_scale=None, drop_rate=0.,
                 attn_drop_rate=0., drop_path_rate=0., norm_layer=nn.LayerNorm,
depths=[3, 4, 6, 3],
                 sr_ratios=[8, 4, 2, 1], num_stages=4, linear=False, pretrained=None,
has_patch_embed=True):
        super().__init__()
        # self.num_classes = num_classes
```

```
        self.depths = depths
        self.num_stages = num_stages
        self.linear = linear

        dpr = [x.item() for x in torch.linspace(0, drop_path_rate, sum(depths))]  #
stochastic depth decay rule
        cur = 0

        for i in range(num_stages):
            if has_patch_embed:
                patch_embed = OverlapPatchEmbed(img_size=img_size if i == 0 else
img_size // (2 ** (i + 1)),
                                                patch_size=7 if i == 0 else 3,
                                                stride=4 if i == 0 else 2,
                                                in_chans=in_chans if i == 0 else
embed_dims[i - 1],
                                                embed_dim=embed_dims[i])
            else:
                if i == 0:
                    continue  # for decoder, skip first layer 1/4, for saving memory

            block = nn.ModuleList([Block(
                dim=embed_dims[i], num_heads=num_heads[i], mlp_ratio=mlp_ratios[i],
qkv_bias=qkv_bias,
                qk_scale=qk_scale,
                drop=drop_rate, attn_drop=attn_drop_rate, drop_path=dpr[cur + j],
norm_layer=norm_layer,
                sr_ratio=sr_ratios[i], linear=linear)
                for j in range(depths[i])])
            norm = norm_layer(embed_dims[i])
            cur += depths[i]
            if has_patch_embed:
                setattr(self, f"patch_embed{i + 1}", patch_embed)
            setattr(self, f"block{i + 1}", block)
            setattr(self, f"norm{i + 1}", norm)

            self.has_patch_embed = has_patch_embed


        self.apply(self._init_weights)
        self.init_weights(pretrained)

    def _init_weights(self, m):
        if isinstance(m, nn.Linear):
            trunc_normal_(m.weight, std=.02)
            if isinstance(m, nn.Linear) and m.bias is not None:
                nn.init.constant_(m.bias, 0)
        elif isinstance(m, nn.LayerNorm):
            nn.init.constant_(m.bias, 0)
            nn.init.constant_(m.weight, 1.0)
        elif isinstance(m, nn.Conv2d):
            fan_out = m.kernel_size[0] * m.kernel_size[1] * m.out_channels
            fan_out //= m.groups
            m.weight.data.normal_(0, math.sqrt(2.0 / fan_out))
            if m.bias is not None:
                m.bias.data.zero_()

    def init_weights(self, pretrained=None):
        if isinstance(pretrained, str):
            os.getcwd()
            pretrained_weights = torch.load(pretrained, map_location='cpu')
            self.load_state_dict(pretrained_weights, strict=False)

    def freeze_patch_emb(self):
        self.patch_embed1.requires_grad = False

    @torch.jit.ignore
    def no_weight_decay(self):
        return {'pos_embed1', 'pos_embed2', 'pos_embed3', 'pos_embed4', 'cls_token'}
# has pos_embed may be better
```

```
    def get_classifier(self):
        return self.head

    def reset_classifier(self, num_classes, global_pool=''):
        self.num_classes = num_classes
        self.head = nn.Linear(self.embed_dim, num_classes) if num_classes > 0 else
nn.Identity()

    def forward_features(self, x, x_bis=None):
        B = x.shape[0]
        outs = []

        for i in range(self.num_stages):
            block = getattr(self, f"block{i + 1}")
            norm = getattr(self, f"norm{i + 1}")
            if self.has_patch_embed:
                patch_embed = getattr(self, f"patch_embed{i + 1}")
                x, H, W = patch_embed(x)
                assert x_bis is None
            else:
                _, _, H, W = x.shape
                x = x.flatten(2).transpose(1, 2)

                assert x_bis.shape == x.shape

                x_bis = x_bis.flatten(2).transpose(1, 2)

            for blk in block:
                x = blk(x, H, W, x_bis)
            x = norm(x)
            x = x.reshape(B, H, W, -1).permute(0, 3, 1, 2).contiguous()
            outs.append(x)

        return outs

    def forward_attention(self, xs, xs_bis):
        B = xs[0].shape[0]
        outs = []

        # print(len(xs))
        # print(len(xs_bis))

        for i, (x_in, x_bis) in enumerate(zip(xs, xs_bis)):
            assert x_in.shape == x_bis.shape

            block = getattr(self, f"block{i + 2}")
            norm = getattr(self, f"norm{i + 2}")
            _, _, H, W = x_in.shape
            # print("x_in.shape", x_in.shape)
            x_in = x_in.flatten(2).transpose(1, 2)
            # print("flatten x_in.shape", x_in.shape)

            x_bis = x_bis.flatten(2).transpose(1, 2)

            x = x_in.clone()
            for blk in block:
                x = blk(norm(x + x_in), H, W, x_bis)
            x = norm(x)
            x = x.reshape(B, H, W, -1).permute(0, 3, 1, 2).contiguous()
            outs.append(x)

        return outs

    def forward(self, x, x_bis=None):
        # x is query
        # x_bis is k, v

        if x_bis is None:
            x = self.forward_features(x, x_bis)
```

```
        else:
            x = self.forward_attention(x, x_bis)
        # x = self.head(x)

        return x


class DWConv(nn.Module):
    def __init__(self, dim=768):
        super(DWConv, self).__init__()
        self.dwconv = nn.Conv2d(dim, dim, 3, 1, 1, bias=True, groups=dim)

    def forward(self, x, H, W):
        B, N, C = x.shape
        x = x.transpose(1, 2).view(B, C, H, W).contiguous()
        x = self.dwconv(x)
        x = x.flatten(2).transpose(1, 2)

        return x


def _conv_filter(state_dict, patch_size=16):
    """ convert patch embedding weight from manual patchify + linear proj to conv"""
    out_dict = {}
    for k, v in state_dict.items():
        if 'patch_embed.proj.weight' in k:
            v = v.reshape((v.shape[0], 3, patch_size, patch_size))
        out_dict[k] = v

    return out_dict


class pvt_v2_b0(PyramidVisionTransformerV2):
    def __init__(self, **kwargs):
        super(pvt_v2_b0, self).__init__(
            patch_size=4, embed_dims=[32, 64, 160, 256], num_heads=[1, 2, 5, 8],
mlp_ratios=[8, 8, 4, 4],
            qkv_bias=True, norm_layer=partial(nn.LayerNorm, eps=1e-6), depths=[2, 2,
2, 2], sr_ratios=[8, 4, 2, 1],
            drop_rate=0.0, drop_path_rate=0.1, pretrained=kwargs['pretrained'])

class pvt_v2_b2(PyramidVisionTransformerV2):
    def __init__(self, **kwargs):
        super(pvt_v2_b2, self).__init__(
            patch_size=4, embed_dims=[64, 128, 320, 512], num_heads=[1, 2, 5, 8],
mlp_ratios=[8, 8, 4, 4],
            qkv_bias=True, norm_layer=partial(nn.LayerNorm, eps=1e-6), depths=[3, 4,
6, 3], sr_ratios=[8, 4, 2, 1],
            drop_rate=0.0, drop_path_rate=0.1, pretrained=kwargs['pretrained'])
```

ДОДАТОК В

ВІДГУК
на кваліфікаційну роботу рівня магістра
«Розробка інформаційної технології стеження за об’єктами у відеоряді»
студента групи 126м-22-2 Тимофієнко Павла Юрійовича

1 Мета даної кваліфікаційної роботи – розробка інформаційної технології стеження за об’єктами у відеоряді.

2 Обрана тема актуальна тому, що візуальне відстеження об’єктів є ключовим завданням у системах комп’ютерного зору з широким спектром застосування в різних сферах, включаючи відеоспостереження, інтелектуальне транспортування та взаємодію людини з комп’ютером. Метою візуального відстеження об’єктів є точне та послідовне відстеження конкретних цілей у відеоряді. Однак цьому процесу заважають такі фактори, як деформація, оклюзія, швидкий рух і фонові перешкоди.

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 присвячено аналізу теми дослідження та постановці задачі. Наведено огляд засобів та технологій відстежування об’єктів і коротко розглянуті сучасні алгоритми стеження за об’єктами. В другому розділі наведено проектну складову вирішення завдання. Розглянуто деякі відомі моделі стеження за об’єктами в режимі реального часу, детально описано алгоритми стеження на основі трансформерів та сіамських нейронних мереж. Обґрунтовано вибір інструментів програмної реалізації інформаційної технології. Третій розділ присвячено розробці інформаційної технології стеження за об’єктами у відеоряді. Виконано реалізацію інформаційної технології.

Створено програмне забезпечення, яке призначено для стеження за об’єктами у відеоряді в режимі реального часу.. Оригінальність отриманих в роботі результатів та їх наукова новизна полягають у наступному:

- досліджено і обгрунтовано вибір архітектури моделі стеження за об’єктами на зображенні та інструменти програмної реалізації технології;
- порівняно різні методи відстеження об’єктів у відеоряді;
- обрано для використання у якості основи моделі BOTT;

- визначено множину параметрів, що використовуються для визначення обмежувальних рамок цільового об’єкта у кожному кадрі відеоряду;
- вперше обґрунтовано архітектуру інформаційної технології стеження за об’єктами у відеоряді, що підтримує високу точність під час обробки навантажених сцен;
- виконано порівняння використання даної програмної реалізації на двох класах комп’ютерних систем: з високою і невисокою обчислювальною продуктивністю;
- розроблено та реалізовано відповідну інформаційну технологію стеження за об’єктами у відеоряді на базі обраної архітектури.

6 Практична цінність результатів полягає в тому, що запропонована в роботі інформаційна технологія дозволяє виконувати стеження за об’єктами 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 В якості зауваження можна відзначити:

- відсутність перекладів написів на рисунках;
- недостатню повноту опису алгоритма програмної реалізації.

Таким чином, кваліфікаційна робота цілком відповідає вимогам, що пред’являються до робіт другого (магістерського) рівня спеціальності 126 Інформаційні системи та технології. Незважаючи на вищезазначені недоліки, кваліфікаційна робота Тимофієнко Павла Юрійовича 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 стеження за об’єктами у відеоряді»
студента групи 126м-22-2 Тимофієнко Павла Юрійовича

Розглянута робота присвячена обґрунтуванню архітектури інформаційної технології, що забезпечує стеження за об’єктами 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 на тлі широкого розповсюдження систем машинного зору розробка та вдосконалення технологій стеження за об’єктами в відеоряді стають все більш актальним напрямом досліджень.

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 вперше обґрунтовано архітектуру інформаційної технології стеження за об’єктами у відеоряді, що підтримує високу точність під час обробки навантажених сцен.

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

До числа загальних зауважень і недоліків роботи слід віднести:

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

2) 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