Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

ДОСЛІДЖЕННЯ ВИКОРИСТАННЯ МЕТОДІВ DEEP LEARNING ДЛЯ РОЗПІЗНАВАННЯ ТРАНСПОРТНИХ ЗАСОБІВ НА ЗОБРАЖЕННІ
(тема)

Виконав:
студент 2 курсу, групи ІНФм-21-1

Ященко А. В.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Тітова О. В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Ященку Артуру Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи: Дослідження використання методів Deep Learning для розпізнавання транспортних засобів на зображенні
затверджена наказом по університету від «_9___» листопада 2022 року № 1469Ст
2. Термін подання студентом роботи до екзаменаційної комісії _22_ __листопада__ 2022 р.
3. Вихідні дані до роботи методи розпізнавання об'єктів на зображенні, метод SSD, науково-технічна література, моделі класифікацій, інтернет ресурси, матеріали наукових конференцій, тощо.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз предметної області.
2. Дослідження використання методів детектування об'єктів на зображенні на прикладі SSD алгоритму.
3. Програмна реалізація Single Short Detector алгоритму.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Актуальність дослідження використання Deep Learning методів для розпізнавання транспортних засобів на зображенні, постановка задачі, діаграми, схеми, таблиці, діаграми, опис моделі, опис програмної реалізації.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Консультант дотримання діючих стандартів та норм	Доцент Творошенко І. С		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	09.11.2022	
2	Аналіз завдання, підбір літератури	10.11.22-12.11.22	
3	Аналіз літератури з досліджуваної проблеми	13.11.22-14.11.22	
4	Аналіз технічних засобів	15.11.22-15.11.22	
5	Розробка методу	16.11.22-16.11.22	
6	Програмна реалізація	17.11.22-18.11.22	
7	Оформлення пояснювальної записки	18.11.22-19.11.22	
8	Перевірка на плагіат		
9	Рецензування		
10	Підготовка презентації та доповіді		
11	Занесення роботи в електронний архів		
12	Попередній захист кваліфікаційної роботи	30.11.22	

Дата видачі завдання 9 листопада 2022 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Тітова О. В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 60 с., 40 рис., 41 джерело.

ГЛИБИННЕ НАВЧАННЯ, РОЗПІЗНАВАННЯ ТРАНСПОРТНИХ ЗАСОБІВ, SSD МЕТОД, ОБРОБКА ЗОБРАЖЕНЬ, АЛГОРИТМИ, ОЗНАКИ, ВЛАСТИВОСТІ.

Об’єктом дослідження є питання детекції транспортних засобів в системах комп’ютерного зору.

Метою дослідження є розробка методів для покращення алгоритму Single Short Detector (SSD) та аналіз результатів.

Здійснено експериментальне оцінювання ефективності алгоритмів SSD, YOLO, R-CNN, Fast R-CNN, Faster R-CNN, Mask R-CNN. У даному дослідженні використані методи математичної статистики, інтелектуального аналізу даних, розпізнавання образів, а також імітаційне моделювання. На основі дослідження і проведеного експерименту проаналізовано та підтверджено ефективність застосування алгоритму SSD в порівняно з YOLO.

Перспективою подальшого дослідження є вивчення завадостійкості розроблених методів та оцінювання їх прикладної результативності стосовно об’ємних колекцій зображень.

DEEP LEARNING, VEHICLE RECOGNITION, SSD METHOD, IMAGE PROCESSING, ALGORITHMS, SIGNS, PROPERTIES.

The object of research is the issue of vehicle detection in computer vision systems.

The purpose of the research is to develop methods to improve the Single Short Detector (SSD) algorithm and to analyze the results.

An experimental evaluation of the effectiveness of SSD, YOLO, R-CNN, Fast R-CNN, Faster R-CNN, Mask R-CNN algorithms was carried out. In this study, methods of mathematical statistics, data mining, pattern recognition, as well as simulation modeling were used. Based on the study and the experiment, the effectiveness of the SSD algorithm in comparison with YOLO was analyzed and confirmed.

The prospect of further research is to study the noise immunity of the developed methods and evaluate their applied performance in relation to large image collections.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

SSD – Single Short Detector (одноходовий детектор)

YOLO – You Only Look Once (ти дивишся лише раз)

R-CNN – Region Based Convolutional Neural Networks (регіональні згорткові нейронні мережі)

ВСТУП

У задачах розпізнавання об’єктів на зображенні – набір ознак та властивостей у поєднанні з правильно обраним алгоритмом визначають ефективність результату. Ознаки та властивості об’єкту розпізнавання дозволяють дослідити сукупність різних об’єктів на предмет відповідності (схожості, подібності, тощо). На теперішній час існує багато алгоритмів для розпізнавання образів, але чи є вони однаково ефективні? Питання ефективності є найголовнішім критеріям при вирішенні будь-якої задачі, тому що в ньому закладені такі поняття як: швидкість обробки даних, кількість ресурсів необхідних для здійснення аналізу, точність результату, тощо.

Методи розпізнавання образів широко використовуються у будь-яких сферах нашого життя (медицина, освіта, економіка, охоронна довкілля, охорона правопорядку, тощо). Процес розпізнавання у більшості випадків не залежить від сфери життя, тому дане дослідження може буде адаптовано у різних сферах. У даній роботі буде досліджено особливості методу розпізнавання на прикладі алгоритму SSD при розпізнаванні транспортних засобів на зображенні.

Актуальність дослідження полягає у аналізі та порівняння існуючих алгоритмів розпізнавання та ідентифікації об’єктів на зображенні в порівнянні з алгоритмом.

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної галузі

У наші часи сфера технології стрімко розвиваються, з’являються нові потреби та можливості. Те, що ще 20 років по тому здавалось не можливим, сьогодні – звичайна справа. Великий внесок у розвиток інформаційних технологій зробив такий відомий напрямок як Deep Learning (глибоке навчання), який автоматизував та покращив велику купу процесів, пов’язаних з нейробіологією, фізикою, статистикою, біологією, медициною, тощо [1].

Однією з важливих тем для дослідження є розпізнавання транспортних засобів на дорогах, яке використовується для вирішення великої купи задача (у тому числі у логістичних) з ціллю визначення складність транспортних потоків на дорогах, тощо [2]. Ця тема є досить актуально за останні десять років. Завдяки використанню deep learning було реалізовано системи моніторингу транспортного руху, систему детекції скорості руху автомобілів, систему розпізнавання номерних знаків, тощо [3].

Припустимо, існує логістична компанія з поставок життєвою-необхідних ліків, де постачання відбувається через дорожні транспортні засоби. Шлях доставки ліків може займати тисячі кілометрів відстані. Завдання полягає у швидкому постачанні медикаментів, де у більшості випадків це не є можливим через дорожні затори. У таких випадках, аналізуючи ситуацію на дорогах можна своєчасно скорегувати маршрути та уникнути зазначених проблем [4].

Коли справа доходить до реалізації таких систем програмістами, аналізуючи існуючі алгоритми та методи розпізнавання – може виникнути питання у актуальності алгоритму та його відповідності до вирішення поставленої задачі [5]. Саме тому, це дослідження допоможе вирішити питання правильного обрання методу розпізнавання, поліпшити розуміння актуальності алгоритмів та підходів для вирішення подібних проблем [6].

Існує багато алгоритмів детекції об’єктів, більш того, кожен алгоритм може бути адаптованих у різних середовищах (фото, відео та у real time). Виділяють дві основні групи алгоритмів – одноетапні та двоетапні. До одноетапних належать алгоритми, які виключають генерацію регіонів. До двоетапних відносять алгоритми, які відповідають за пошук регіонів та аналіз знайдених об’єктів у цих регіонах [7]. Зазвичай, одноетапні підходи використовують при розпізнаванні зображень, у той час коли двоетапні – на відео та у real time.

На практиці, при розпізнаванні об’єктів за допомогою двоетапних методів використовують YOLO та SSD – алгоритми. Переваги цих алгоритмів у тому, що вони обидва якісно детектують об’єкти та класифікують при цьому не потребують великого розподілу ресурсів. Різниця між YOLO та SSD полягає у тому, що SSD не розбиває зображення на сітку довільного розміру, а передбачає зміщення ключових рамок [8]. Треба зазначити, що алгоритми YOLO та SSD якісно розпізнають об’єкти на відео, SSD метод – використовує пірамідальну ієрархії виходів мережі, що забезпечує ефективну детекцію об’єктів різних розмірів [9].

Щодо одноетапних алгоритмів – ефективність цих алгоритмів є не такою високою, як в двоетапних підходах, але вони можуть бути більше ефективними та ресурсозатратними у випадках, коли можна зневажити не великою похибкою при розпізнаванні. Найбільш відомими є R-CNN – алгоритм, який базується на згорткових нейронних мережах, Fast R-CNN – де покращено швидкість обробки даних, Faster R-CNN – де покращено продуктивність алгоритму та Mask R-CNN з доданою можливістю сегментації екземплярів об’єктів [10].

1.2 Визначення задач класифікації

Існують задачі детекції об’єкта на зображенні, які можуть включати в себе інші задачі, що покращує алгоритми і допомагає визначити, де саме у сітці пікселей на зображенні знаходиться об’єкт.

Задача семантичної сегментації – суть задачі складається в тому, що на вхід моделі передається зображення, а на виході кожен піксель має позначку приналежності до певної категорії. Наприклад, коли ми маємо зображення, де людина переходить дорогу, то для кожного пікселя треба, до чого саме цей піксель належить (до людини, дороги чи чогось іншого). Недоліком такого підходу є те, що при перетині двох об’єктів, які є семантично однаковими (дві людини, одна людина йде позаду іншої), то такий випадок буде ідентифікований як один об’єкт.

Задача класифікації з локалізацією – сутність задачі полягає у тому, що разом із міткою міткою категорії класу розраховується фрейм, що обмежує розташування об’єкта на зображенні. Ця рамка досить часто виглядає як прямокутник, а розмір цієї рамки максимально щільно охоплює об’єкт на зображенні.

Задача детекції об’єктів – ціль цього підходи полягає в тому, що на зображенні є кілька об’єктів, нам треба ідентифікувати необхідні об’єкти із використанням обмежувальних рамок, та класифікувати ці обмежувальні рамки із уже відомими нами класами.

1.3 Підходи до вирішення задач детекції об’єктів

Один з наївних підходів, який базується на згорткових нейронних мережах є використання канонічних зображень класів, які необхідно знайти на зображенні та використання рухливого фрейму для виявлення подібності. Такий підхід відомий під назвою зіставленням із шаблоном. У ситуаціях, коли

замість шаблону використовується нетренований класифікатор, тоді для досягнення досконалого результату потрібно зробити повне зіставлення фреймів для повного переконання у правдивості класифікатора (об’єкти можуть знаходитись у різних місцях зображення та мати різні масштаби).

Для розрахунку кількості фреймів для зображення $W \times H$ потрібно скористатися формулою

$$\sum_{i=0}^{W} \sum_{J=0}^{H} (W - i) \cdot (H - j) = \frac{1}{4} m \cdot n \cdot (m + 1) \cdot (n + 1) = O(m^2 n^2),$$

що виконує перебір дуже не ефективним методом та займає досить багато часу. Для того, щоб зменшити цей час використовують два базових підходи:

– одноетапний метод – такий підхід, при якому не використовуються жодні алгоритми для генерації регіонів, при цьому предбачає координати лімітованих фреймів з різними характеристиками (як ступінь впевненості у подальшому завдяки корекції положення фреймів та результати класифікації);

– двоетапний метод – цей метод поділяють на два етапи. Перший етап, де за допомогою нейронної мережі виконується пошук регіонів, які з великою вірогідністю включають у себе пошуковий об’єкт. Другий етап – на знайдених регіонах відбувається аналіз приналежності цих об’єктів до класу, який ми шукаємо та уточняється розташування лімітованих фреймів.

1.3.1 Двоетапні методи

R-CNN – алгоритм, який базується на згорткових нейронних мережах. Суть полягає у том, що замість використання рухомих фреймів фіксованого розміру, на першому кроці алгоритм намагається знайти прямокутні фрейми різних розмірів, які потенційно можуть містити об’єкт. Кількість таких прямокутних фреймів, згенерованих на першому етапі дорівнює 2000. Знайдені регіони завдяки афінним перетворюванням набувають розмір, який

подається до CNN. Дуже часто в якості CNN використовують архітектуру CaffeNet, яка для кожного регіону отримує 4096 ознак. Далі, вектора ознак цих регіонів обробляються за допомогою SVM, що виконує класифікацію об’єктів з одною SVM на кожний домен (рис. 1.1).

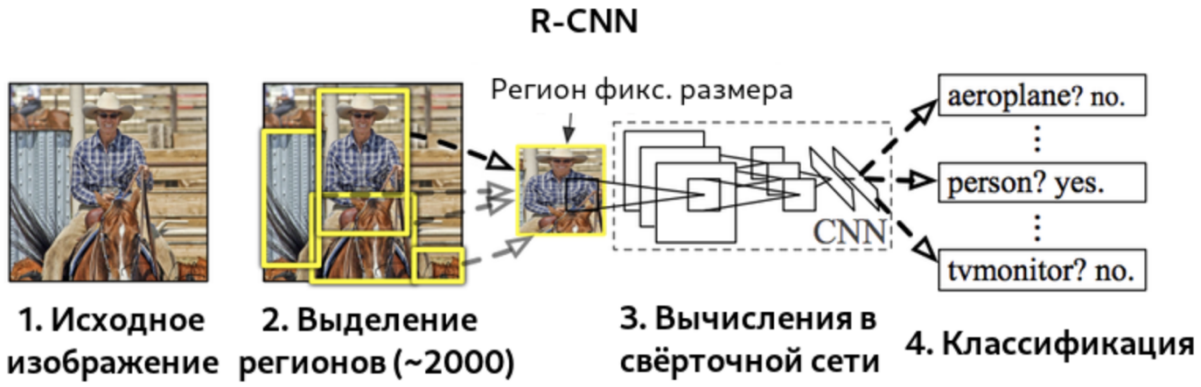

Рисунок 1.1 – Схема роботи R-CNN

Fast R-CNN – цей алгоритм є покращеною версією алгоритму R-CNN (покращена швидкість обробки даних). Характерною особливістю цього алгоритму передача до CNN не окремих регіонів, а всього зображення одразу для отримання загальної карти ознак. Через це регіони накладаються на загальну карту ознак і в результаті кількість операцій згортки зменшується. Оскільки регіони мають різний розмір, необхідно привести ознаки до фіксованого. За допомогою операції RoIPooling регіон ділиться на сітку, де розмірність комірок збігається з розмірністю виходу, після чого коміркам сітки проводиться вибір максимального значення. Отримані регіони фіксованого розміру використовуються для здійснення як класифікації, так і лінійної регресії для зсуву меж його фреймів (рис. 1.2).

Faster R-CNN – є покращеною версією алгоритму Fast R-CNN (покращена продуктивність алгоритму). Розробники цього алгоритму запропонували використовувати окремий модуль RNP (Region Proposal Network). RPN виступає в якості згорткової мережі, яка генерує регіони за

ознаками вихідного зображення. Згенеровані регіони передаються в два шари – box-regression-layer, що прогнозує значення зсуву для рамок, які виконують обмеження, і box-classification-layer, що класифікує зображення в межах запропонованої області. Ключову роль при цьому відіграють ключові рамки (anchor boxes) – рамки з різними положеннями та розмірами для вікна, яке рухається. Такі рамки мають фіксоване положення та різні форми та масштаби.

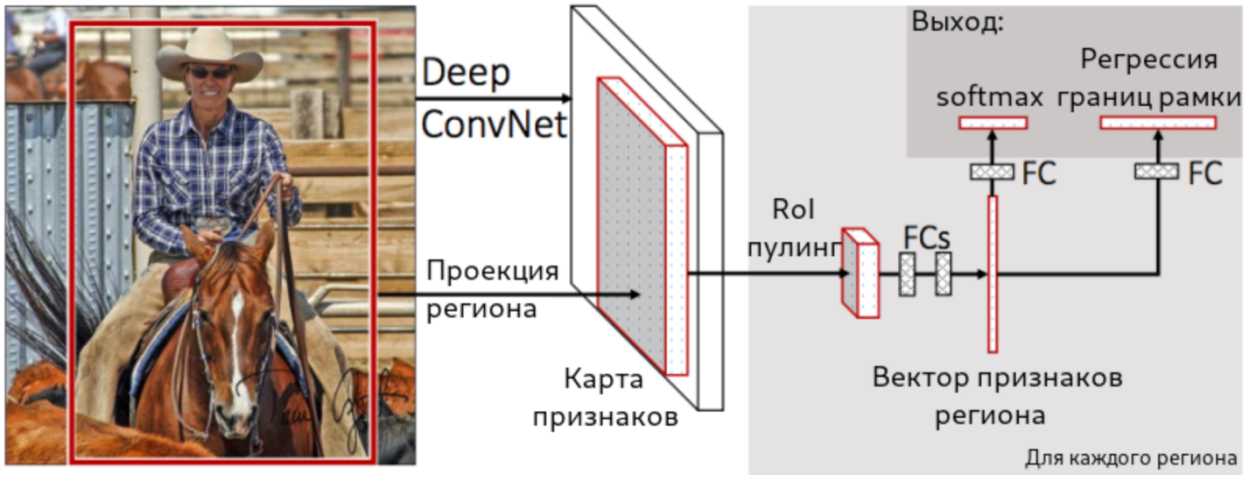

Рисунок 1.2 – Схема роботи Fast R-CNN

Для одного й самого масштабу обирається, три ключові рамки – квадратної форми; прямокутної форми, орієнтованої горизонтально; прямокутної форми, орієнтованої вертикально (рис. 1.3). Далі, здійснюється переміщення цих рамок для генерації регіонів. Для згенерованих таким чином регіонів розраховуються ймовірності знаходження об’єкта всередині рамки за допомогою cls-шара, а за зсув розташування відповідає reg-шар.

Після проходження шару RPN виконується RoIPooling. Оскільки класифікацією та регресією кордонів займається як мережа в цілому, так і RPN, що пропонує регіони, функція втрат враховує як фінальне рішення щодо класифікації та регресії координат, так і класифікацію та регресію координат, проведену RPN (рис. 1.4).

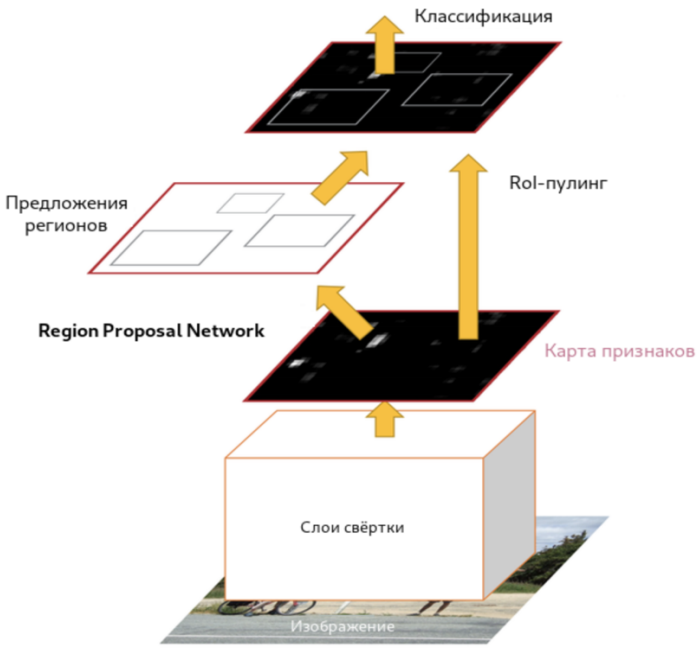

Рисунок 1.3 – Схема роботи алгоритму Faster R-CNNN

Рисунок 1.4 – Ключові рамки в Faster R-CNN

Mask R-CNN – є покращеною версією алгоритму Faster R-CNN (додано можливість сегментації екземплярів об’єктів). До Mask R-CNN додається також маска об’єкта – прямокутна матриця приналежності пікселя поточному об’єкту. Відбувається передбачення маски для кожного класу за допомогою класифікації без наявності інформації про те, що зображено в регіоні, що виділяє окремий класифікатор на останньому рівні мережі (рис. 1.5).

Реалізація прогнозування маски викликала декілька архітектурних змін стосовно Faster R-CNN: тепер використовується RoIAlign замість RoIPooling. RoIPooling добре підходить для масштабування рамок, що обмежують, однак, для масок такий метод виявляється не точним. RoIAlign не використовує заокруглень зсувів для пулінгу, а зберігає значення з точкою, що плаває, використовуючи білінійну інтерполяцію. Це забезпечило точне виділення маски об’єкта (рис. 1.6).

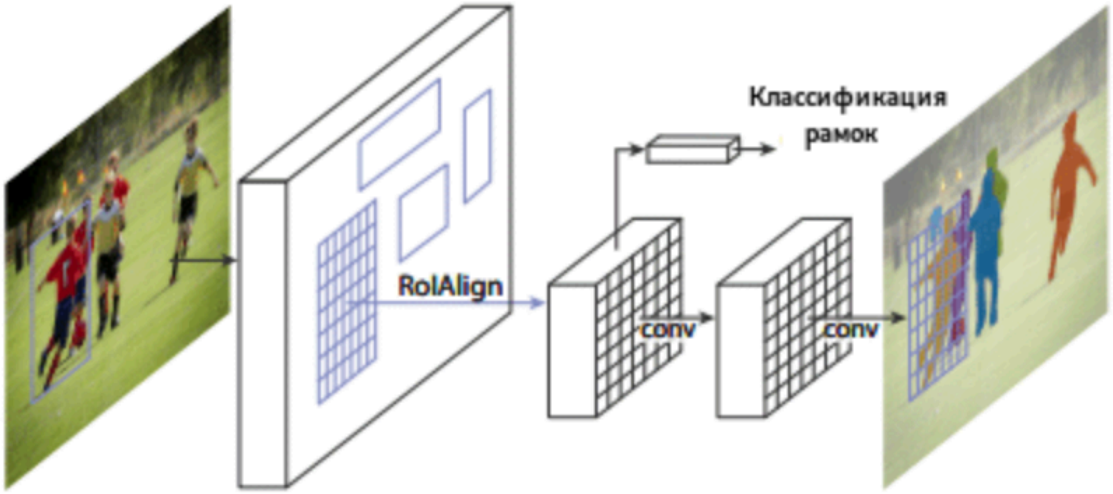

Рисунок 1.5 – Схема роботи Mask R-CNN

Рисунок 1.6 – Приклад семантичної сегментації об’єктів із застосуванням Mask R-CNN

1.3.2 Одностайні методи

Перевага алгоритмів R-CNN полягає у тому, що вони використовують передбачення регіонів, що забезпечує високу точність, але вони можуть бути дуже повільним для деяких сфер, таких як безпілотне керування автомобілем. Через це, можна виділити ще одне сімейство алгоритмів, які не використовують регіони – сімейство алгоритмів швидкої детекції.

YOLO – цей алгоритм був започаткован як перша спроба реалізувати детекцію об’єктів у реальному часі. Сутність алгоритму у тому, що вихідне зображення поділяється на сітку з $N \times N$ комірок. Коли центр об’єкта потрапляє всередину координат комірки, то ця комірка є відповідальною за визначення параметрів місцезнаходження об’єкта (рис. 1.7).

Кожна комірка описує кілька варіантів розташування обмежуючих рамок для одного і того ж об’єкта. Кожен з цих варіантів характеризується п’ятьма значеннями – координатами центру рамки, що обмежує, його шириною і висотою, а також ступеня впевненості в тому, що рамка, що обмежує, містить в собі об’єкт. Також необхідно для кожної пари класу об’єктів та комірки визначити ймовірність того, що комірка містить у собі об’єкт цього класу.

Таким чином, останній шар мережі, що приймає кінцеве рішення про обмежувальні рамках і класифікації об’єктів працює з тензором розмірності

$$N \times N \times (5B + C),$$

де B – кількість рамок для комірки, що передбачаються;

C – кількість класів об’єктів, визначених спочатку.

YOLO працює набагато швидше за алгоритми сімейства R-CNN за рахунок того, що підтримує дроблення на константну кількість комірок замість того, щоб пропонувати регіони та розраховувати рішення для кожного

регіону окремо (рис. 1.8). Недоліком алгоритму є погана якість розпізнавання об’єктів складної форми.

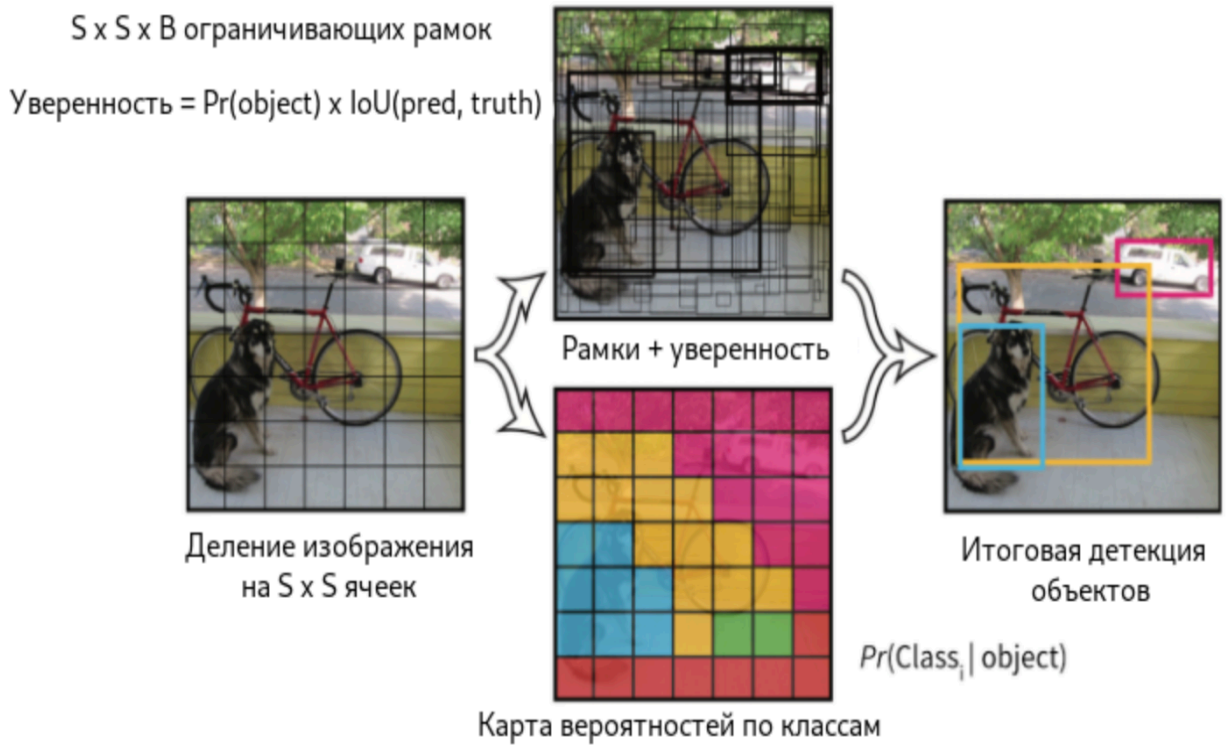

Рисунок 1.7 – Схема роботи алгоритму YOLO

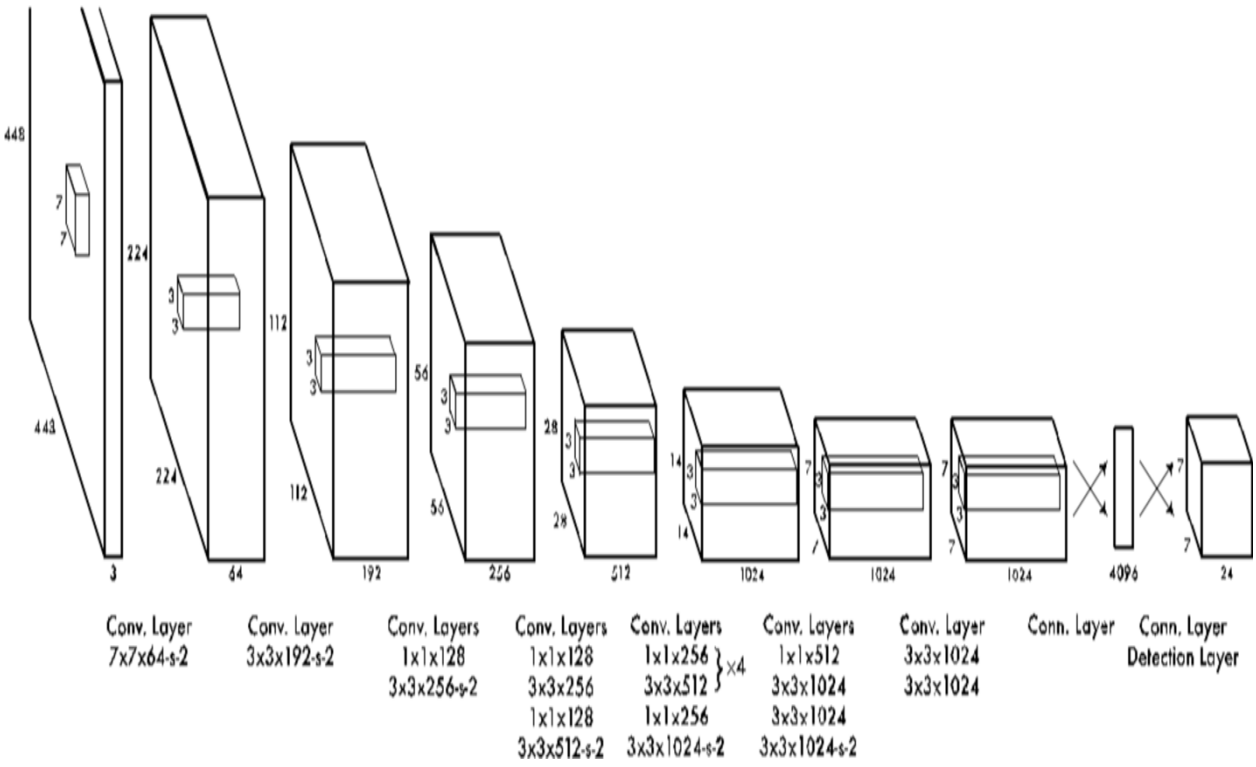

Рисунок 1.8 – Архітектура нейронної мережі для алгоритму YOLO

YOLOv2, YOLOv3 – відрізняється від попередньої версії використанням батчевої нормалізації на згорткових шарах, навчанням моделей відбувається на зображеннях з підвищеною розширністю, наявністю та використанням ключових рамок для передбачення місцезнаходження об'єктів, використанням алгоритму кластирізації k-середніх для ефективнішого навчання вибору розмірів обмежувальних фреймів з використанням функції відстані на основі IoU:

$$dist(x, c_i) = 1 - IoU(x, c_i),$$

де x – обмежувальна рамка;

c_i – центроїд кластеру.

Кількість рамок-центроїдів, що обмежують, обираються за допомогою «методу ліктя». YOLOv2 допускає, що рамки, що обмежуються, можуть не суттєво відхилятися від розташування центру, що забезпечує стабільність на тлі менш ефективного рівномірного вибору рамок-кандидатів по всьому вихідному зображенню. Алгоритм YOLO9000, названий через використання 9000 кращих класів ImageNet, використовує деревоподібну структуру класів та аналізує їх вкладеність. Якщо серед класів є мітка «Йоркширський тер'єр», це означатиме, що знайдений об'єкт буде підкласом мітки «Собака». Через це, не виникає взаємної винятковості класів, та функція softmax до всіх класів не буде застосовуватись. Для передбачення ймовірності вузла класу, слід йти шляхом від вузла до кореня:

$$p(yorkshire\ terrier|dog) = p(yorkshire\ terrier|dog) \cdot p(dog|animal) \cdot \\ \cdot p(animal|object) \cdot p(object),$$

де $p(object)$ – вірогідність знайдення об'єкта, обчислена на етапі генерації обмежувальних рамок.

Шлях прогнозування умовної ймовірності може зупинитися будь-якому етапі, залежно від цього, які мітки доступні.

YOLOv3 – поліпшена версія YOLOv2, де використовується логістична регресія для оцінок достовірностей рамок, що обмежують; також використовується кілька незалежних логістичних класифікаторів для кожного класу (замість одного шару softmax); додається міжрівне з'єднань між рівнями прогнозування рамок, що обмежують; використовується архітектури DarkNet та ResNet для згорткових мереж (рис. 1.9).

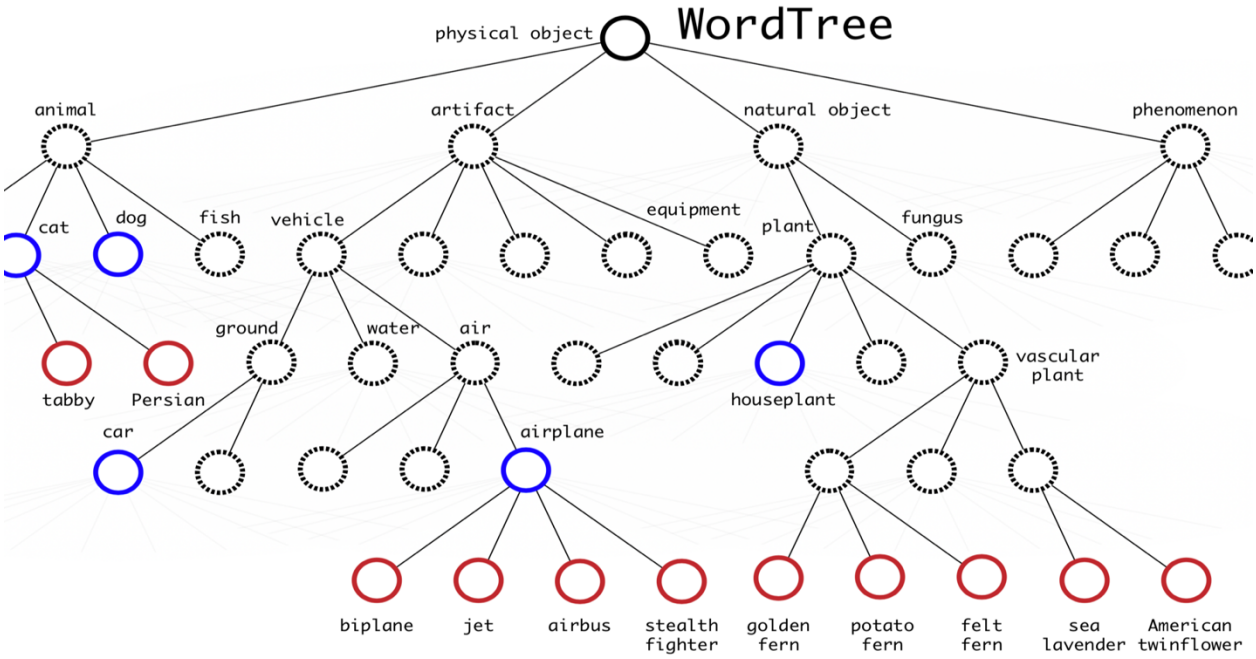

Рисунок 1.9 – Приклад деревовидної структури YOLO9000

SSD (Single Shot Detector) – модель, що використовує пірамідальну ієрархії виходів мережі, що забезпечує ефективну детекцію об'єктів різних розмірів (рис. 1.10). Зображення послідовно обробляється у різних шарах згорткової мережі, які зменшуються у розмірах. Коли зображення виходе із останнього шару кожної розмірності, то кожен шар приймає рішення щодо детекції об'єктів, таким чином складається «пірамідальна характеристика» зображення. Це дозволяє детектувати різних масштабів об'єкти, оскільки розмірність виходів перших шарів корелює з рамками, що обмежують, для

маленьких об’єктів, а останніх – для великих. Різниця між YOLO та SSD полягає у тому, що SSD не розбиває зображення на сітку довільного розміру, а передбачає зміщення ключових рамок.

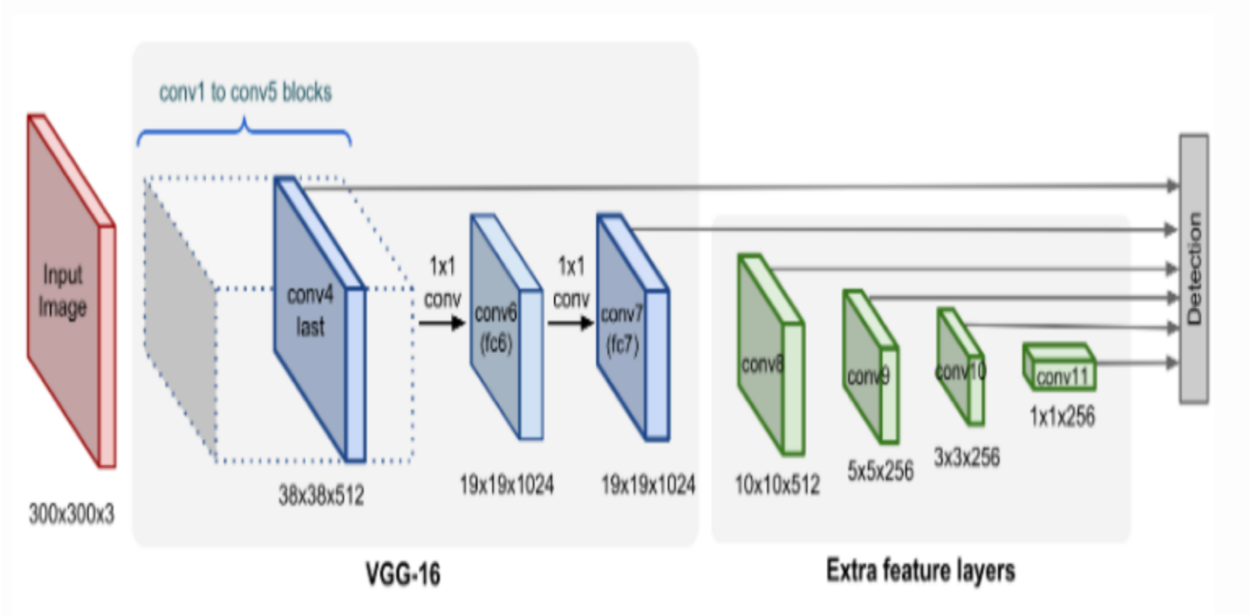

Рисунок 1.10 – Архітектура нейронної мережі SSD

1.4 Постановка задачі дослідження

На сьогоднішній день розвиток нейромереж займає провідне місце у світі технологій. Штучний інтелект охоплює усі галузі діяльності, починаючи з освіти, медицини, підприємницької діяльності, правової системи, культури, авіобудівництва, сфер послуг та закінчуючи взаємодію на клітинному рівні. За останні 7 років, завдяки штучним мережам, людство перейшло на новий рівень адаптації та сприйняття, конкуренція між корпораціями не має кордонів.

Завдяки поступовому покращенню правової системи у багатьох державах, рівень дорожньо-транспортних пригод знизився, але в той час з’явилася проблема реалізації цих законів. У цей момент, саме штучний інтелект став вирішенням цього питання. У багатьох державах існують

системи детекції транспортних засобів, але інше питання щодо ефективності алгоритмів, які використовують ці системи.

Об’єктом дослідження є питання детекції транспортних засобів в системах комп’ютерного зору.

Метою дослідження є розробка методів для покращення алгоритму Single Short Detector (SSD) та аналіз результатів.

Для досягнення мети необхідно вирішити такі завдання:

– провести аналіз існуючих алгоритмів детектування;

– провести аналіз ефективності алгоритму Single Short Detector;

– розробити покращення алгоритму SSD;

– реалізувати програмну систему.

Результатом роботи має бути розроблена покращеної версії алгоритму Single Short Detector та реалізація програмної системи.

АНАЛІЗ ТА МОДИФІКАЦІЯ SIGNLE SHORT DETECTION ПІДХОДУ ДЛЯ РОЗПІЗНАВАННЯ ТРАНСПОРТНИХ ЗАСОБІВ

2.1 Мережева модель SSD

SSD («single shot multibox detector») є сучасною системою детектування об’єктів, заснованих на глибокому навчанні. Засновником SSD є Вей Лю, який запропонував цю моделю на 14-й Європейській конференції з комп’ютерного зору у 2016 році і в результаті цього став одноетапним алгоритмом детектування об’єктів, який був другий за популярністю після YOLO [11–12]. Ідея SSD базується не тільки на якірному механізмі і структурі піраміди ознак Faster R-CNN, але й також реалізує ідею регресії YOLO і виявлення та класифікацію множинних обмежувальних фреймів на основі простої наскрізної мережі. Порівнянно з Faster R-CNN, SSD не потребує вилучення областей-кандидатів, а швидкість виявлення набагато вища. SSD не використовує повністю підключений шар, а точність виявлення покращується в порівнянні з YOLO.

Модель мережі SSD в основному складається з трьох частин, включаючи базову мережу, мережу вилучення функцій та мережу виявлення. Базова мережа вдосконалена на основі VGG16 (група візуальної геометрії 16). Враховуючи, що повністю зв’язний шар буде заважати інформації про місцезнаходження об’єктів, останні два повністю зв’язних шари, а саме FC6 та FC7, замінюються згортковими шарами Conv6 та Conv7. Потім додаються наступні чотири набори згорткових шарів: Conv8, Conv9, Conv10 та Conv11. У кожному шарі для зменшення розмірності використовуються ядра згортки 1×1, а для виділення ознак – ядра згортки 3×3. Далі карти ознак Conv4_3 та Conv7 об’єднуються з картами Conv8_2, Conv9_2, Conv10_2 та Conv11_2, щоб сформувати багатомасштабну мережу виділення ознак у вигляді піраміди ознак. Нарешті, два згорткових ядра розміром 3×3 використовуються для виконання операцій згортки на кожній карті ознак у мережі виявлення. Одне

згорткове ядро виводить достовірність категорії, а інше надає інформацію про місцезнаходження об’єкта для регресії. Всі результати обчислень об’єднуються і переносяться на шар втрат. Остаточний результат виявлення виводиться за допомогою алгоритму не-максимального придушення (NMS).

На рисунку 2.1 показана базова структура мережевої моделі SSD, а на рисунку 2.2 наведені основні параметри мережевої моделі SSD.

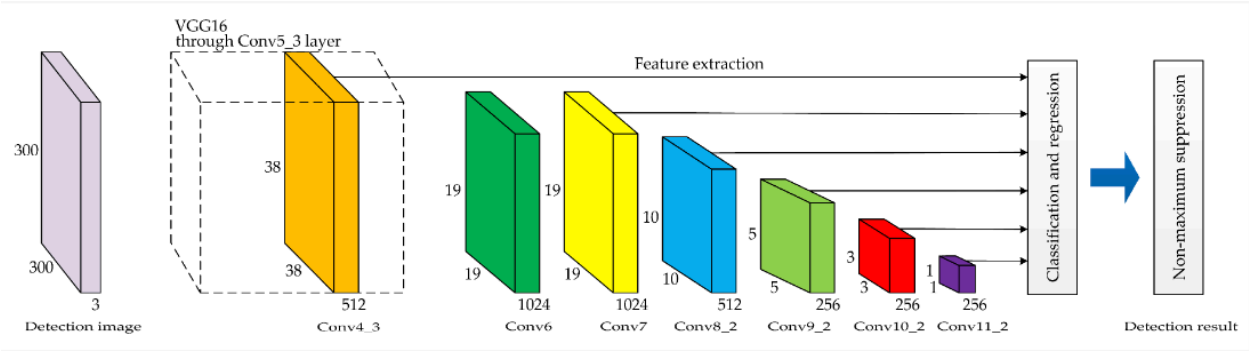

Рисунок 2.1 – Базова структура моделі мережі SSD

Layer	Convolutional Kernel Size	Convolutional Kernel Number	Step Size	Filling	Feature Map Size
Conv1_1	3 × 3	64	1	1	300 × 300
Conv1_2	3 × 3	64	1	1	300 × 300
Maxpool1	2 × 2	1	2	0	150 × 150
Conv2_1	3 × 3	128	1	1	150 × 150
Conv2_2	3 × 3	128	1	1	150 × 150
Maxpool2	2 × 2	1	2	0	75 × 75
Conv3_1	3 × 3	256	1	1	75 × 75
Conv3_2	3 × 3	256	1	1	75 × 75
Conv3_3	3 × 3	256	1	1	75 × 75
Maxpool3	2 × 2	1	2	0	38 × 38
Conv4_1	3 × 3	512	1	1	38 × 38
Conv4_2	3 × 3	512	1	1	38 × 38
Conv4_3	3 × 3	512	1	1	38 × 38
Maxpool4	2 × 2	1	2	0	19 × 19
Conv5_1	3 × 3	512	1	1	19 × 19
Conv5_2	3 × 3	512	1	1	19 × 19
Conv5_3	3 × 3	512	1	1	19 × 19
Maxpool5	3 × 3	1	1	1	19 × 19
Conv6	3 × 3	1024	1	1	19 × 19
Conv7	1 × 1	1024	1	0	19 × 19
Conv8_1	1 × 1	256	1	0	19 × 19
Conv8_2	3 × 3	512	2	1	10 × 10
Conv9_1	1 × 1	128	1	0	10 × 10

Рисунок 2.2 – Основні параметри моделі мережі SSD

Модель мережі SSD приймає багатозадачну функцію втрат, яка в основному включає помилки позиціонування та довіри. Загальна функція втрат дорівнює зваженій сумі втрат позиціонування та достовірності, що може бути виражена наступною формулою:

$$L(x, c, l, g) = \frac{1}{N}\Big(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)\Big),$$

де l – блок виявлення;

g – реальний блок;

c – достовірність багатокласового об'єкта;

N – кількість блоків виявлення, які можуть ефективно відповідати реальному блоку;

L_{conf} – втрата достовірності;

L_{loc} – втрата положення;

α – ваговий коефіцієнт втрати положення та втрати достовірності, який встановлюється рівним 1 шляхом перехресної перевірки.

Втрата положення отримується шляхом обчислення згладженої втрати L_1 між виявленою та реальною коробками. Зміщення центральної точки координат (x, y), ширина w і висота h обмежувальної рамки регресуються для отримання мінімального значення втрати положення. Відповідна формула виглядає наступним чином:

$$L_{loc}(x, l, g) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx, cy, w, h\}} x_{ij}^{k} \cdot smooth_{L1}(l_i^m - g_j^i),$$

де Pos – сукупність усіх позитивних вибірок;

x_{ij}^{k} вказує, чи відповідає категорія об'єкта k, передбачена i-м блоком виявлення, класифікаційній мітці j-го реального блоку, l – якщо відповідає, 0 – в іншому випадку;

l_i^m – координати i-го блоку виявлення;

g_j^m – координати j-го реального блоку.

$$g_j^{cx} = \frac{g_j^{cx} - d_i^{cx}}{d_i^w},$$

$$g_j^{cx} = \frac{g_j^{cy} - d_i^{cy}}{d_i^h},$$

$$g_i^w = \log\left(\frac{g_i^w}{d_i^w}\right),$$

$$g_j^h = \log\left(\frac{g_j^h}{d_i^h}\right),$$

де g_j^{cx} та g_j^{cy} – координати центральних точок j-го реального поля;

g_j^w та g_j^h – ширина та висота j-го реального поля відповідно;

d_i^{cx} та d_i^{cy} – координати центральних точок i-го поля виявлення відповідно;

d_i^w та d_i^h – ширина та висота i-го поля виявлення відповідно.

Втрата достовірності отримується шляхом розрахунку Softmax втрати достовірності багатокласового об’єкта, яка виражається наступною формулою:

$$L_{conf}(x, c) = -\sum_{i \epsilon Pos}^{N} x_{ij}^p \log\left(c_i^p\right) - \sum_{i \epsilon Neg} \log\left(c_i^0\right),$$

$$c_i^p = \frac{\exp\left(c_i^p\right)}{\sum_p \exp\left(c_i^p\right)},$$

де p – категорія об’єкта;

x_{ij}^{p} вказує, чи відповідає категорія об’єкта p, передбачена i-м блоком виявлення, класифікаційній мітці j-го реального блоку;

c_i^p – ймовірність того, що категорія об’єкта, передбачена i-м блоком виявлення, є p, якщо збіг правильний, то втрати малі при великій ймовірності;

c_i^0 – ймовірність того, що категорія об’єкта, передбачена i-м блоком виявлення, є фоном, якщо об’єкт відсутній в блоці виявлення, то втрати малі при великій ймовірності.

2.2 Недоліки SSD у виявленні транспортних засобів

SSD вбирає в себе переваги Faster R-CNN і YOLO. Однак мережева модель SSD все ще має багато недоліків, коли вона застосовується для виявлення транспортних засобів, включаючи незадовільний ефект виявлення для малогабаритних транспортних засобів, низьку точність виявлення за поганих погодних умов, а також легке виявлення заблокованих транспортних засобів, які легко пропустити. Аналіз та узагальнені причини такі:

– у передньому вигляді розумного автомобіля об’єкт транспортного засобу на великі відстані становить лише невелику частку площі зображення на зібраному зображенні виявлення, а масштаб об’єкта транспортного засобу невеликий. Хоча мережева модель SSD має різномасштабну мережу вилучення функцій, SSD застосовує недискримінаційний метод для різномасштабних функцій і просто вибирає кілька шарів функцій для прогнозування, не враховуючи, що неглибокі та глибокі згорткові шари містять різні локальні деталі та текстурні та семантичні особливості. Таким чином, мережева модель SSD має недостатню здатність витягувати особливості дрібномасштабних об’єктів транспортних засобів і все ж досягла задовільного ефекту виявлення;

– у реальних дорожніх сценах різні об’єкти транспортних засобів мають очевидні відмінності в характеристиках, таких як колір, форма та задні ліхтарі,

і на них легко впливають зміни умов освітлення, сильні погодні перешкоди та оклюзія дорожніх об’єктів. Ці умови створюють багато проблем для точного виявлення передніх транспортних засобів. Оригінальна модель мережі SSD має низькі показники виявлення транспортних засобів у складних умовах, а її надійність та пристосованість до навколишнього середовища погані;

– у процесі навчання мережі завдання регресії полягає лише в тому, щоб відповідати правильному вікну виявлення. Відповідно, відповідна втрата буде безпосередньо встановлена на нуль, коли жоден об’єкт транспортного засобу не присутній на деяких зображеннях набору даних; таким чином, інші зображення не використовуються в повному обсязі. У ранжируванні оцінок достовірності кількість негативних блоків виявлення набагато більша, ніж кількість позитивних блоків виявлення. Відповідно, навчальна мережа приділяє велику увагу частці негативних вибірок, що призводить до повільної швидкості навчання мережевої моделі;

– коли розумний автомобіль проїжджає через перехрестя, міські магістралі та зони заторів, одне зібране зображення виявлення може включати кілька об’єктів транспортних засобів, що неминуче призводить до взаємного блокування між об’єктами транспортних засобів. Однак оригінальна модель мережі SSD має низьку ефективність виявлення об’єктів, що перекриваються, і схильна до пропуску виявлення в сценах з декількома об’єктами.

2.3 Удосконалена базова структура SSD

Враховуючи обмежену здатність вилучення особливостей оригінальної моделі мережі SSD для невеликих об’єктів транспортних засобів, структуру моделі необхідно розумно вдосконалити. Прямим шляхом підвищення здатності до виділення ознак є розширення глибини мережі шляхом додавання декількох згорткових шарів. Однак цей метод призведе до швидкого збільшення параметрів мережевої моделі, що схильне до явища надмірної

підгонки і значно знижує ефективність виявлення навчальної мережі. В останні роки локальна топологія, представлена початковим блоком, поступово завойовує все більшу популярність в області виявлення об’єктів з бурхливим розвитком глибокого навчання і згорткових нейронних мереж.

Початковий блок був вперше запропонований Сегеді на Міжнародній конференції з комп’ютерного зору та розпізнавання образів в 2015 році, який був успішно застосований в GoogLeNet і досяг відмінних результатів класифікації та розпізнавання в ILSVRC2014 (Imagenet Large Scale Visual Recognition Challenge 2014) [13–14]. Початковий блок – це невелика мережева структура, що додається до мережевої моделі. Згорткові ядра різного розміру використовуються для виділення ознак одного і того ж вхідного шару, тим самим значно розширюючи загальну ширину мережі. Такий підхід допомагає підвищити здатність мережевої моделі до вилучення ознак та уникнути явища надмірної підгонки.

SSD створює багатомасштабну мережу вилучення ознак у вигляді піраміди ознак, додаючи кілька наборів згорткових шарів за базовою мережею. Карти особливостей низького та високого рівня відповідають за вивчення особливостей та прогнозування дрібномасштабних та великомасштабних об’єктів, відповідно. Карти ознак низького рівня містять детальну інформацію, але семантичні ознаки є недостатніми. Карти ознак високого рівня є протилежністю.

Кожен шар ознак у вихідному SSD спирається лише на одну ознаку з попереднього шару, що не дозволяє досягти обміну контекстною інформацією під час вилучення різномасштабних ознак, тим самим значно впливаючи на ефективність виявлення мережевої моделі. Злиття ознак є ефективним підходом до вирішення цієї проблеми. Злиття функцій полягає в обробці функціональних шарів різних масштабів та формуванні нового функціонального шару.

Злиття високорівневих семантичних ознак і неглибокої детальної інформації допомагає зміцнити зв’язок між шарами ознак і реалізувати обмін

контекстною інформацією в мережевій моделі.Спрямоване на проблему того, що оригінальна мережева модель SSD має недостатню здатність витягувати особливості дрібномасштабних об’єктів транспортних засобів у складних умовах, це дослідження розширює та поглиблює нейронну мережу та вдосконалює базову структуру SSD, поєднуючи початковий блок та злиття особливостей. На рисунку 2.3 показана базова структура вдосконаленої мережевої моделі SSD, а на рисунку 2.4 – внутрішня структура початкового блоку.

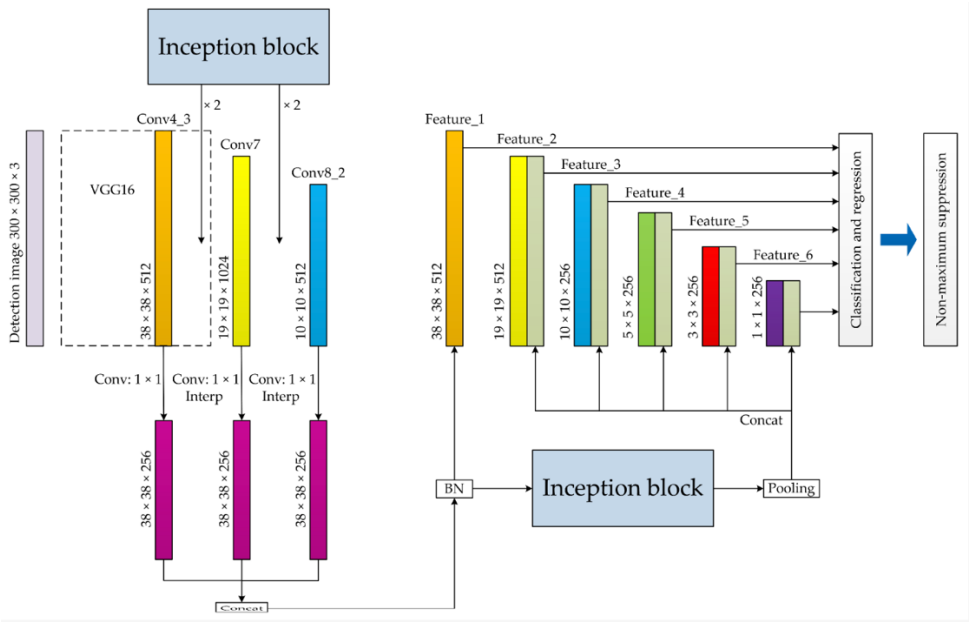

Рисунок 2.3 – Базова структура вдосконаленої моделі мережі SSD

На рисунку 2.3 показано, що початковий блок був використаний кілька разів у вдосконаленій моделі мережі SSD. По-перше, чотири групи початкових блоків додаються до базової мережі SSD для вилучення локальних особливостей мережі. Новостворені інтервальні шари виконують перетворення масштабу шару особливостей на шарах Conv7 і Conv8_2 за допомогою білінійної інтерполяції, і вихідний масштаб становить 38×38, таким чином, роблячи його таким же розміром, як і шар Con4_3. Потім, новостворений шар конкатенації об’єднує вищезгадані три функціональні

шари з однаковим масштабом в новий функціональний шар за допомогою операції конкатенації для досягнення злиття функцій.

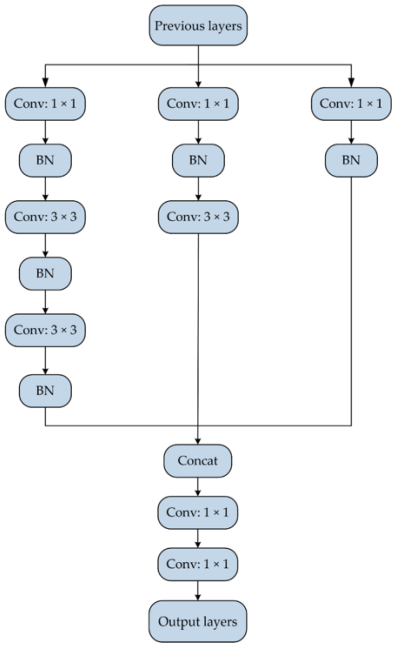

Рисунок 2.4 – Внутрішня структура початкового блоку

Цей специфічний шар ознак містить контекстну інформацію і використовується як ознака_1 для побудови нової багатомасштабної мережі вилучення ознак після обробки пакетної нормалізації (BN). Нарешті, згорткове ядро розміром 3 × 3 використовується для зменшення масштабу шару ознак мережі шар за шаром з Feature_1 в якості базового шару. Утворюється п’ять шарів ознак з різними масштабами. Знову додається група початкових блоків, і шляхом об’єднання шару Feature_1_inception отримуються п’ять нових функціональних шарів, що відповідають вищезгаданим п’яти функціональним шарам.

Знову створюється новий шар конкатенації, і операція конкатенації проводиться для об’єднання п’яти груп шарів ознак з однаковим масштабом один за одним для формування Feature_2, Feature_3, Feature_4, Feature_5 і Feature_6. Нова багатомасштабна мережа виділення ознак створюється шляхом об’єднання вищезгаданих шарів з Об’єктом_1. Нова багатомасштабна

мережа вилучення функцій може повторно використовувати ключові функції, що сприяє покращенню загальної здатності вилучення функцій мережевої моделі.

На рисунку 2.4 показано, що початковий блок в основному використовує згорткові ядра 5×5, 3×3 та 1×1 для виконання операції згортання вхідних ознак, а два послідовно згорнуті шари 3×3 використовуються замість 5×5 згорткових шарів. Перевага цієї структурної конструкції полягає в тому, що вона може додатково зменшити параметри моделі, зберігаючи вихідне рецептивне поле незмінним.

Здатність виділення ознак початкового блоку може бути покращена за рахунок введення нелінійних перетворень. У внутрішній структурі початкового блоку співвідношення кількості ядер згортки 5×5, 3×3 та 1×1 становить 1:2:1. Згортковий шар 1×1 додається перед згортковими шарами 5×5 і 3×3, щоб зменшити кількість вхідних каналів ознак і загальний розрахунок. В кінці структури після шару згортки додаються два згорткових шари 1×1 для подальшого підвищення нелінійної обчислювальної здатності початкового блоку.

Мережева модель може в найбільшій мірі витягти особливості прихованих шарів в мережі і повністю поділитися контекстною інформацією, використовуючи початковий блок і злиття функцій. Такий підхід допомагає підвищити здатність вилучення ознак для невеликих об’єктів транспортних засобів у складних умовах.

Хоча вдосконалена мережева модель SSD збільшує структурну складність та кількість параметрів, це не має значного впливу на обчислювальне навантаження, оскільки масштаб шару ознак зберігається в невеликому діапазоні, а обробка BN використовується кілька разів. Це може забезпечити високу швидкість навчання моделі та хороші показники виявлення в реальному часі, одночасно покращуючи рівень вилучення ознак.

2.4 Зважена маска

В оригінальній навчальній мережі SSD, коли зображення не має об’єктів транспортних засобів у наборі даних, відповідна функція втрати класифікації буде безпосередньо встановлена на нуль, а решта цінних зображень у наборі даних не можуть бути повністю використані. Враховуючи, що кількість негативних блоків виявлення набагато більша, ніж кількість позитивних блоків виявлення, використовуються блоки виявлення з високими показниками достовірності. Співвідношення позитивних і негативних вибірок контролюється до 1:3, що, безумовно, знижує швидкість збіжності навчальної мережі.

Виходячи з недоліків оригінальної моделі мережі SSD при навчанні, в даній роботі проведено розрахунок зваженої маски для задач класифікації вибірок та регресії при використанні релевантних наборів даних для навчання. Метод розрахунку зваженої маски полягає в наступному:

– коли присутні K блоків виявлення, кількість позитивних зразків дорівнює N, кількість негативних зразків дорівнює M, $K = N + M$, і встановлюється мітка класифікації;

– коли $N > 0$, зважена маска для класифікації позитивних зразків встановлюється як *pos_mask=label/N*;

– коли $M > 0$ і співвідношення позитивних і негативних зразків контролюється до 1:3, зважена маска для класифікації негативних зразків встановлюється як *neg_mask={1-label}/M×3*;

– зважена маска, що використовується для задачі класифікації, має вигляд *cls_mask=pos_mask + neg_mask*;

– припускаючи, що ваговий коефіцієнт задачі регресії дорівнює α, зважена маска для задачі регресії має вигляд *reg_mask=pos_mask* × α.

Дане дослідження гарантує, що навчальна мережа приділяє велику увагу вибірковим даним з високою складністю класифікації, використовуючи зважену маску в процесі навчання. Такий підхід сприяє вирішенню проблеми

дисбалансу між фоном та позитивними і негативними вибірковими даними та подальшому прискоренню швидкості навчання мережевої моделі.

2.5 Покращена функція визначення втрат

Оригінальна модель мережі SSD має хороший ефект виявлення на одному об’єкті транспортного засобу в простих умовах. Однак ця модель не може досягти задовільних результатів виявлення при виявленні багатьох об’єктів транспортних засобів у багатооб’єктних сценах або об’єктів транспортних засобів із сильною оклюзією. Легко з’являється відсутність виявлення, помилкове виявлення та неточне позиціонування об’єкта.

Враховуючи вищезазначені недоліки, це дослідження покращує функцію втрат та додає втрати виключення на основі початкового положення та втрати довіри. Покращена функція втрат може бути виражена наступною формулою:

$$L = L(x, c, l, g) + \gamma L_{RepGT},$$

де L_{RepGT} – втрати від виключення;

γ – ваговий коефіцієнт, який використовується для збалансування допоміжних втрат.

Це дослідження дозволяє $P_+ = \{\mathrm{P}\}$ представляти сукупність всіх блоків–кандидатів з IoU більше 0,5, а $G_+ = \{\mathrm{G}\}$ представляти сукупність всіх реальних блоків. Для будь-якої скриньки-кандидата $P \in P_+$ це дослідження дозволяє визначити реальну скриньку з великим IoU як її заданий об’єкт, а саме:

$$G_{Attr}^{P} = arg_{G \in G_+} \arg\max IoU(G, P).$$

Враховуючи, що втрата від виключення має на меті змусити блок-кандидат відштовхнути сусідній реальний блок, то об’єктом виключення для будь-якого блоку-кандидата $P \in P_+$ є реальний блок з великим IoU, окрім зазначеного об’єкта, а саме:

$$G_{Rep}^{P} arg_{G \in G_+ \{G_{Attr}^{P}\}} maxIoU(G, P).$$

Це дослідження дозволяє B^P бути блоком виявлення, регресованим з блоку-кандидата P. Перекриття IoG між B^P і G_{Rep}^{P} може бути виражене наступною формулою:

$$IoG\left(B^P, G_{Rep}^{P}\right) = \frac{area(B^P \cap G_{Rep}^{P})}{area(G_{Rep}^{P})}.$$

Втрати від виключення можуть бути розраховані за наступною формулою:

$$L_{RepGT} = \frac{IoG(B^P, G_{Rep}^{P})}{|P_+|}.$$

Втрата на виключення використовується для збільшення відстані між блоком виявлення та навколишніми нетранспортними об’єктами. Якщо спостерігається зона перекриття з навколишніми нетранспортними об’єктами, то на блок виявлення будуть накладатися додаткові штрафи. Штраф буде великим, коли зона перекриття велика, і навпаки. Таким чином, додавання втрат виключення на основі початкової функції втрат може запобігти переміщенню блока виявлення на сусідні нетранспортні об’єкти. Такий підхід корисний для точного визначення місцезнаходження об’єктів транспортних засобів та ефективного покращення ефективності виявлення об’єктів, що перекриваються, у сценах з декількома об’єктами.

2.6 Експериментальне середовище

Програмне середовище наступне: 64-розрядна операційна система Windows 10, фреймворк глибокого навчання TensorFlow, CUDA 9.1, cuDNN 7.1, Python 3.7.0, MATLAB R2018a.

Апаратне середовище наступне: Процесор Intel (R) Core (TM) i7-7700 CPU@3.60 ГГц, 32 ГБ оперативної пам’яті та графічний процесор NVIDIA GeForce GTX 1080Ti, 11 ГБ.

2.7 KITTI Dataset

У цій роботі використано набір еталонних даних KITTI для експериментів з розпізнавання транспортних засобів, який був розроблений спільно Технологічним інститутом Карлсруе та Технологічним інститутом Toyota в Чикаго. Цей набір даних став міжнародно використовуваним набором даних для оцінки алгоритмів для сценаріїв автономного водіння. Набір даних KITTI в основному зосереджений на оцінці ефективності різних технологій комп’ютерного зору, включаючи оптичний потік, стереозображення, візуальне визначення дальності та виявлення об’єктів [15, 16]. Цей набір даних охоплює реальні зображення доріг у декількох сценаріях, таких як міста, села та автомагістралі. Кожен зразок зображення містить до 15 об’єктів транспортних засобів та 30 об’єктів пішоходів, а розмір зображення становить 1242 × 375 пікселів [17]. Весь набір даних складається з відеопотоків, зібраних за допомогою бінокулярних камер, і може бути розділений на п’ять категорій: дорога, місто, житловий сектор, кампус і людина [18].

Набір даних KITTI включає дані міток і не вимагає ручної анотації, тим самим забезпечуючи надійну інформацію про вміст зображень для навчання моделей [19]. Враховуючи, що в наборі даних присутній 7481 зразок

зображень з відповідними файлами міток, 5985 зображень виділено в якості навчальної множини, а 1496 зображень – в якості тестової множини.

Співвідношення навчальної та тестової множини становить 4:1 [20]. Зображення вибірки можна розділити на вісім категорій відповідно до класифікації об’єктів анотаційної інформації: легковий автомобіль, мікроавтобус, вантажівка, пішохід, пішохід (сидячи), велосипедист, трамвай, а також «неважливо» або «dontcare» [21]. Під час підготовки даних всі файли етикеток повинні бути конвертовані з формату txt у формат XML, необхідний для навчання SSD [22–24].

В результаті цієї роботи залишаються тільки легкові автомобілі, мікроавтобуси, вантажівки та трамваї, а інші нерелевантні категорії вилучаються, оскільки основна увага приділяється виявленню об’єктів транспортних засобів. На рисунку 2.5 представлено приклад зображення набору даних KITTI.

Рисунок 2.5 – Приклад зображення набору даних KITTI

2.8 Індекси мережевого навчання та оцінювання

Для оптимізації використано метод стохастичного градієнтного спуску. Вагові параметри навчальної мережі безперервно оновлюються за допомогою

алгоритму зворотного поширення. Початкова швидкість навчання встановлюється рівною 0,001, коефіцієнт імпульсу – 0,9, а коефіцієнт ослаблення ваги – 0,0005. Величина швидкості навчання тісно пов'язана зі швидкістю збіжності навчальної мережі [25, 26]. Якщо налаштування велике, то мережева модель не буде сходитися. І навпаки, якщо налаштування мале, то швидкість збіжності буде сповільнюватися [27–29]. В даному дослідженні максимальна кількість ітерацій навчальної мережі становить 20 000.

Швидкість навчання встановлюється на рівні 0,001 в перші 12 000 разів, 0,0001 від 12 000 до 16 000 разів і 0,00001 після 16 000 разів. Для функції втрат використовується регуляризація L2, щоб запобігти перенавчанню особливостей навчальної вибірки та уникнути виникнення надмірного припасування [30–34]. На рисунку 2.6 показані функції втрат SSD до та після покращення.

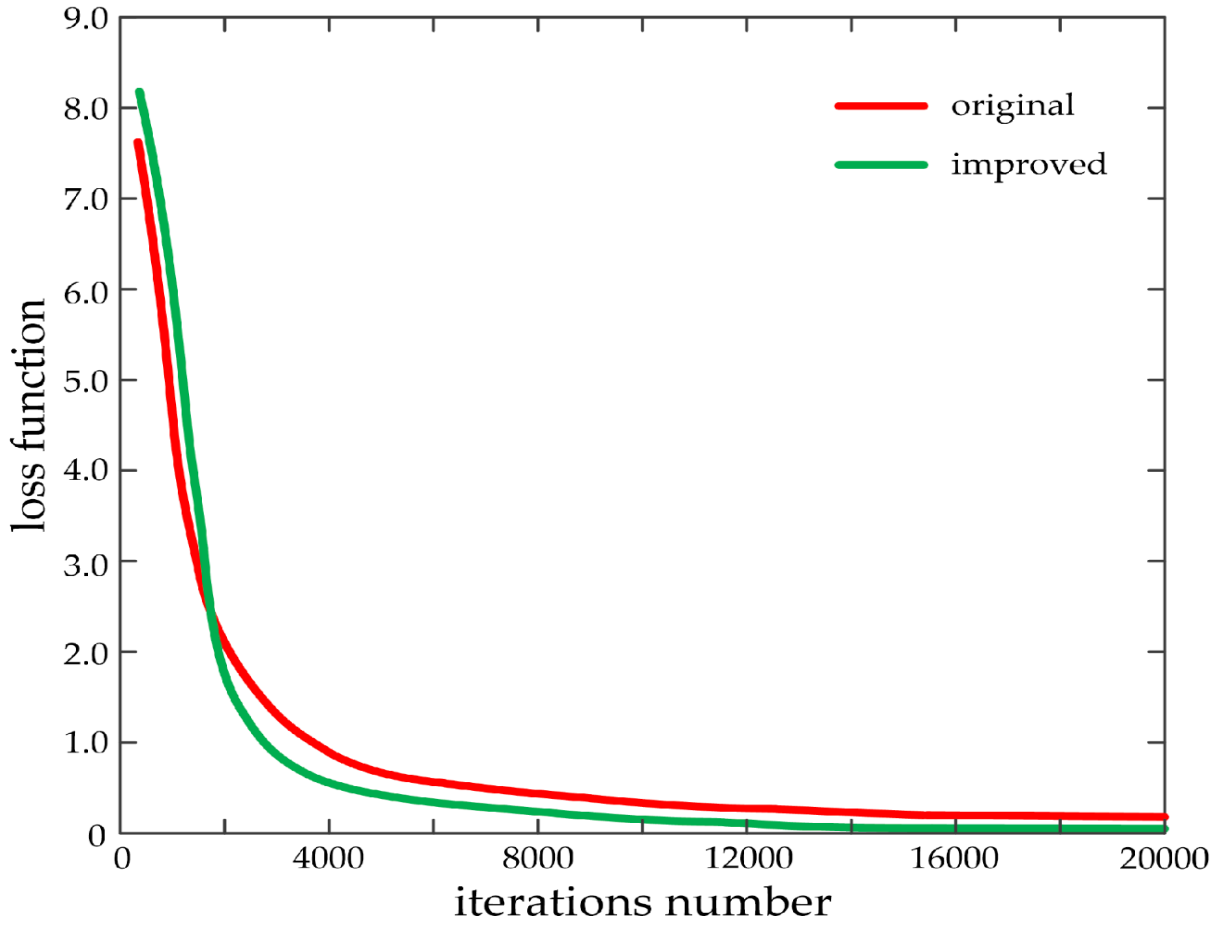

Рисунок 2.6 – Функції втрат SSD до та після покращення

Наведений вище рисунок демонструє, що покращена функція втрат є дещо більшою, ніж вихідна на початку навчання. Такий стан може бути пов'язаний з покращенням функції втрат та додаванням втрат від виключення. Однак, при безперервній ітерації навчальної мережі значення покращеної функції втрат швидко стає меншим за вихідне і, зрештою, поступово зменшується до нуля, тим самим відображаючи перевагу використання зваженої маски [35]. При кількості ітерацій 3400 відстань між двома функціями втрат досягає максимуму. При максимальній кількості ітерацій 20000 відстань між двома функціями втрат досягає мінімуму. Таким чином, швидкість збіжності покращеної моделі мережі SSD є вищою, тим самим вказуючи на те, що проблема дисбалансу даних вибірки була ефективно вирішена [36, 37].

В алгоритмі виявлення транспортних засобів для точної оцінки ефективності виявлення необхідно використовувати оціночні індекси. Враховуючи, що зображення виявлення включає позитивні та негативні зразки, для результату виявлення присутні чотири випадки прогнозування, а матриця плутанини показана на рисунку 2.7.

Confusion Matrix		Predicted condition	
		Positive sample	Negative sample
True condition	Positive sample	TP (True Positive)	FN (False Negative)
	Negative sample	FP (False Positive)	TN (True Negative)

Рисунок 2.7 – Матриця плутанини

Оціночні показники, такі як точність, пригадування та середня точність (mAP), можуть бути розраховані відповідно до матриці плутанини [38].

У цій статті під точністю розуміється частка зразків, результатами виявлення яких є правильно виявлені об’єкти транспортних засобів, і вона може бути виражена наступним чином:

$$P = \frac{TP}{TP + FP}.$$

Відкликання відноситься до частки об’єктів транспортних засобів, які правильно виявлені, і його можна виразити наступним чином:

$$R = \frac{TP}{TP + FN}.$$

mAP є одним з важливих оціночних показників алгоритмів виявлення об’єктів і може бути виражений наступним чином:

$$mAP = \frac{\sum AP}{N} = \frac{\sum \int_0^1 P(R)dR}{N},$$

де N – номер категорії об’єктів.

2.9 Результати експериментальних випробувань та їх аналіз

В алгоритмі NMS поріг IoU потрібно встановлювати вручну. Різні порогові значення IoU дають різну точність і точність пригадування, а встановлення порогового значення IoU тісно пов’язане з ефективністю виявлення мережевої моделі. Після багаторазових експериментальних тестів поріг IoU встановлено на рівні 0,5. На рисунку 2.8 показані криві «точність-відтворення» для оригінального та вдосконаленого твердотільного накопичувача. Крива P-R використовує відгук і точність як горизонтальну і вертикальну координати відповідно, що є загальною кривою, яка

використовується для вимірювання продуктивності алгоритму виявлення. Відповідний відгук є низьким, коли точність є високою. Коли точність має високе значення, ймовірність помилкового виявлення є низькою. Коли відгук має високе значення, ймовірність пропуску виявлення є низькою [39].

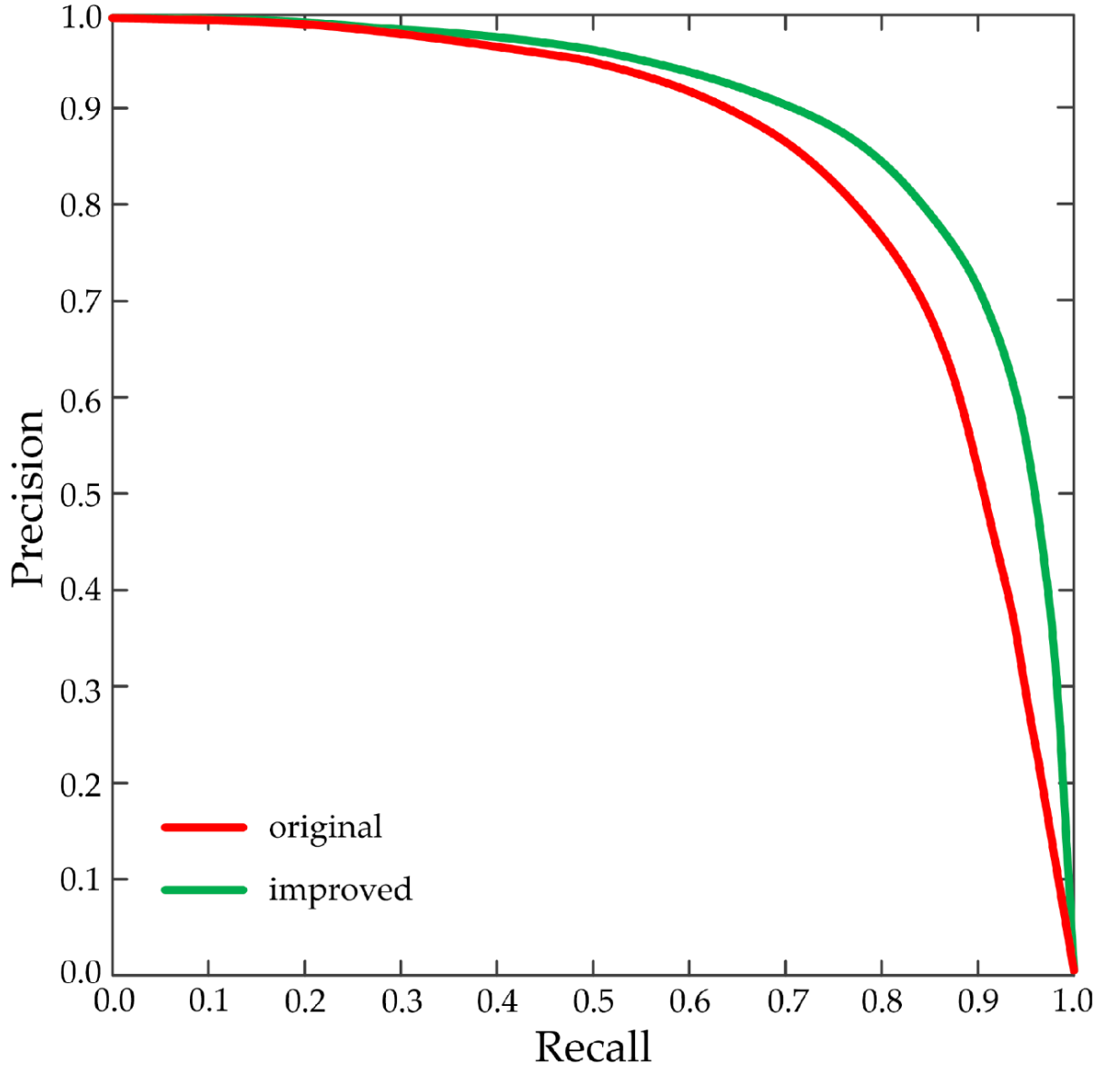

Рисунок 2.8 – Криві точності-відгуку про оригінальний та покращений SSD

Вищезгаданий рисунок демонструє, що покращена крива P-R більш нахилена до правого верхнього кута, ніж оригінальна, тим самим вказуючи на те, що ефективність виявлення покращеного SSD є кращою, ніж у оригінальної. При відкликанні 90%, покращена точність становить 70%, в той

час як у оригіналу – 50%. Площа, обмежена кривою P-R і двома координатними осями, більша, ніж у оригіналу, що свідчить про те, що вдосконалений SSD має очевидні переваги в середній точності виявлення.

Для випробувань на виявлення транспортних засобів використовується тестовий набір KITTI, а результати випробувань у різних складних умовах проілюстровані на рисунках 2.8 – 2.12. Серед них панелі виконані на основі оригінального та вдосконаленого SSD відповідно. На рисунку 2.9 показано, що в тіньовому середовищі оригінальний SSD виявляє лише чотири об’єкти транспортних засобів на короткій відстані, тоді як інші на великій відстані пропускаються. Однак, вдосконалений SSD забезпечує виявлення всіх об’єктів транспортних засобів, а показники достовірності були підвищені до певної міри [40].

На рисунку 2.10 показано, що в оригінальному SSD є випадки відсутності виявлення та неточного позиціонування для невеликих об’єктів транспортних засобів, а вдосконалений SSD забезпечує правильне виявлення та точне позиціонування різномасштабних об’єктів транспортних засобів.

На рисунку 2.11 показано, що кілька об’єктів транспортних засобів заблоковані в різному ступені, а оригінальний SSD спричиняє значну кількість відсутніх виявлень на об’єктах транспортних засобів, які сильно заблоковані, і можна спостерігати неточне позиціонування для об’єктів транспортних засобів на великій відстані. На противагу цьому, вдосконалений SSD забезпечує правильне виявлення всіх легкових автомобілів та мікроавтобусів.

На рисунку 2.12 показано, що об’єкти розташовані на перехресті доріг, що є типовою зоною ризику дорожньо-транспортних пригод. Оригінальний SSD виявляє лише об’єкти з очевидними ознаками, а вдосконалений SSD правильно виявляє всі об’єкти та ефективно покращує відповідні показники достовірності.

Рисунок 2.13 демонструє, що в умовах дорожнього затору щільність об’єктів є високою і в основному оглядається заднім ходом. Оригінальний SSD призводить до відсутності виявлення об’єктів транспортних засобів у

дальньому полі зору, а вдосконалений SSD, як і раніше, забезпечує правильне виявлення всіх об’єктів транспортних засобів.

Рисунок 2.9 – Результати випробувань виявлення транспортних засобів у тіньовому середовищі на основі оригінального та вдосконаленого SSD

Рисунок 2.10 – Результати випробувань виявлення транспортних засобів на різномасштабних об’єктах на основі оригінальної та вдосконаленої SSD

Рисунок 2.11 – Результати тестування виявлення транспортних засобів в умовах оклюзії на основі оригінального та вдосконаленого SSD

Рисунок 2.12 – Результати тестування виявлення транспортних засобів на перехресті доріг на основі оригінальної та вдосконаленої SSD

Рисунок 2.13 – Результати тестування виявлення транспортних засобів в умовах дорожнього затору на основі оригінальної та вдосконаленої SSD

Результати випробувань на виявлення транспортних засобів показують, що ефективність виявлення вдосконаленої мережі SSD має похвальні переваги, що в основному пов’язано з вдосконаленням базової структури SSD та функцією втрати. Запропонований алгоритм виявлення транспортних засобів має чудову надійність та адаптивність до навколишнього середовища для складних дорожніх умов та дорожніх сцен, а точність виявлення була додатково покращена.

РЕАЛІЗАЦІЯ ПРОГРАМНОГО КОДУ

3.1 Огляд головних методів реалізації

При реалізації алгоритму SSD для детектування транспортних засобів, було використано наведені бібліотеки (рис. 3.1).

```
import os
import time
import cPickle
import datetime
import logging
import flask
import werkzeug
import optparse
import tornado.wsgi
import tornado.httpserver
import numpy as np
import pandas as pd
from PIL import Image
import cStringIO as StringIO
import urllib
import exifutil

import caffe
```

Рисунок 3.1 – Бібліотеки, необхідні для реалізації алгоритму

Було сформовано endpoint головної сторінки веб застосунку, який доступний за посиланням ‘/’ (рис. 3.2).

```
@app.route('/')
def index():
    return flask.render_template('index.html', has_result=False)
```

Рисунок 3.2 – Endpoint головної сторінки застосунку

Далі наведена реалізація endpoint класифікації, який доступний за посиланням ’/classify_url’ (рис.3.3).

```
@app.route('/classify_url', methods=['GET'])
def classify_url():
    imageurl = flask.request.args.get('imageurl', '')
    try:
        string_buffer = StringIO.StringIO(
            urllib.urlopen(imageurl).read())
        image = caffe.io.load_image(string_buffer)

    except Exception as err:
        # For any exception we encounter in reading the image, we will just
        # not continue.
        logging.info('URL Image open error: %s', err)
        return flask.render_template(
            'index.html', has_result=True,
            result=(False, 'Cannot open image from URL.')
        )

    logging.info('Image: %s', imageurl)
    result = app.clf.classify_image(image)
    return flask.render_template(
        'index.html', has_result=True, result=result, imagesrc=imageurl)
```

Рисунок 3.3 – Endpoint класифікації

Наступний endpoint за адресою ‘/classify_upload’ відповідає за завантаження зображення та його класифікацію (рис. 3.4).

```
@app.route('/classify_upload', methods=['POST'])
def classify_upload():
    try:
        # We will save the file to disk for possible data collection.
        imagefile = flask.request.files['imagefile']
        filename_ = str(datetime.datetime.now()).replace(' ', '_') + \
            werkzeug.secure_filename(imagefile.filename)
        filename = os.path.join(UPLOAD_FOLDER, filename_)
        imagefile.save(filename)
        logging.info('Saving to %s.', filename)
        image = exifutil.open_oriented_im(filename)

    except Exception as err:
        logging.info('Uploaded image open error: %s', err)
        return flask.render_template(
            'index.html', has_result=True,
            result=(False, 'Cannot open uploaded image.')
        )

    result = app.clf.classify_image(image)
    return flask.render_template(
        'index.html', has_result=True, result=result,
        imagesrc=embed_image_html(image)
    )
```

Рисунок 3.4 – Endpoint для завантаження зображень

Наступний метод відповідає за вбудову завантаженого зображення до HTML розмітки (рис. 3.5).

```
def embed_image_html(image):
    """Creates an image embedded in HTML base64 format."""
    image_pil = Image.fromarray((255 * image).astype('uint8'))
    image_pil = image_pil.resize((256, 256))
    string_buf = StringIO.StringIO()
    image_pil.save(string_buf, format='png')
    data = string_buf.getvalue().encode('base64').replace('\n', '')
    return '' + data
```

Рисунок 3.5 – Метод вбудови завантаженого зображення до HTML розмітки

Наступними є методи класифікації та валідації завантажених зображень (рис. 3.6 – рис. 3.8).

```
class ImagenetClassifier(object):
    default_args = {
        'model_def_file': (
            '{}/models/bvlc_reference_caffenet/deploy.prototxt'.format(REPO_DIRNAME)),
        'pretrained_model_file': (
            '{}/models/bvlc_reference_caffenet/bvlc_reference_caffenet.caffemodel'.format(REPO_DIRNAME)),
        'mean_file': (
            '{}/python/caffe/imagenet/ilsvrc_2012_mean.npy'.format(REPO_DIRNAME)),
        'class_labels_file': (
            '{}/data/ilsvrc12/synset_words.txt'.format(REPO_DIRNAME)),
        'bet_file': (
            '{}/data/ilsvrc12/imagenet.bet.pickle'.format(REPO_DIRNAME)),
    }
    for key, val in default_args.iteritems():
        if not os.path.exists(val):
            raise Exception(
                "File for {} is missing. Should be at: {}".format(key, val))
    default_args['image_dim'] = 256
    default_args['raw_scale'] = 255.

    def __init__(self, model_def_file, pretrained_model_file, mean_file,
                 raw_scale, class_labels_file, bet_file, image_dim, gpu_mode):
        logging.info('Loading net and associated files...')
        if gpu_mode:
            caffe.set_mode_gpu()
        else:
            caffe.set_mode_cpu()
        self.net = caffe.Classifier(
            model_def_file, pretrained_model_file,
            image_dims=(image_dim, image_dim), raw_scale=raw_scale,
            mean=np.load(mean_file).mean(1).mean(1), channel_swap=(2, 1, 0)
        )
```

Рисунок 3.6 – Метод класифікації об’єктів

```
with open(class_labels_file) as f:
    labels_df = pd.DataFrame([
        {
            'synset_id': l.strip().split(' ')[0],
            'name': ' '.join(l.strip().split(' ')[1:]).split(',')[0]
        }
        for l in f.readlines()
    ])
self.labels = labels_df.sort('synset_id')['name'].values

self.bet = cPickle.load(open(bet_file))
# A bias to prefer children nodes in single-chain paths
# I am setting the value to 0.1 as a quick, simple model.
# We could use better psychological models here...
self.bet['infogain'] -= np.array(self.bet['preferences']) * 0.1
```

Рисунок 3.7 – Продовження методу класифікації об’єктів

```
    def classify_image(self, image):
        try:
            starttime = time.time()
            scores = self.net.predict([image], oversample=True).flatten()
            endtime = time.time()

            indices = (-scores).argsort()[:5]
            predictions = self.labels[indices]

            # In addition to the prediction text, we will also produce
            # the length for the progress bar visualization.
            meta = [
                (p, '%.5f' % scores[i])
                for i, p in zip(indices, predictions)
            ]
            logging.info('result: %s', str(meta))

            # Compute expected information gain
            expected_infogain = np.dot(
                self.bet['probmat'], scores[self.bet['idmapping']])
            expected_infogain *= self.bet['infogain']

            # sort the scores
            infogain_sort = expected_infogain.argsort()[::-1]
            bet_result = [(self.bet['words'][v], '%.5f' % expected_infogain[v])
                          for v in infogain_sort[:5]]
            logging.info('bet result: %s', str(bet_result))

            return (True, meta, bet_result, '%.3f' % (endtime - starttime))

        except Exception as err:
            logging.info('Classification error: %s', err)
            return (False, 'Something went wrong when classifying the '
                           'image. Maybe try another one?')
```

Рисунок 3.8 – Метод класифікації зображень

Наступні методи відповідають за зміну орієнтації розташування зображень (рис. 3.9).

```
from PIL import Image
import numpy as np

ORIENTATIONS = {   # used in apply_orientation
    2: (Image.FLIP_LEFT_RIGHT,),
    3: (Image.ROTATE_180,),
    4: (Image.FLIP_TOP_BOTTOM,),
    5: (Image.FLIP_LEFT_RIGHT, Image.ROTATE_90),
    6: (Image.ROTATE_270,),
    7: (Image.FLIP_LEFT_RIGHT, Image.ROTATE_270),
    8: (Image.ROTATE_90,)
}


def open_oriented_im(im_path):
    im = Image.open(im_path)
    if hasattr(im, '_getexif'):
        exif = im._getexif()
        if exif is not None and 274 in exif:
            orientation = exif[274]
            im = apply_orientation(im, orientation)
    img = np.asarray(im).astype(np.float32) / 255.
    if img.ndim == 2:
        img = img[:, :, np.newaxis]
        img = np.tile(img, (1, 1, 3))
    elif img.shape[2] == 4:
        img = img[:, :, :3]
    return img


def apply_orientation(im, orientation):
    if orientation in ORIENTATIONS:
        for method in ORIENTATIONS[orientation]:
            im = im.transpose(method)
    return im
```

Рисунок 3.9 – Методи зміни орієнтації розташування зображень

Наступні методи відповідають за створення фреймів визначених розмірів і пропорцій (рис. 3.10 – рис. 3.13).

```
def __init__(self, img_size, min_size, max_size=None, aspect_ratios=None,
             flip=True, variances=[0.1], clip=True, **kwargs):
    if K.image_dim_ordering() == 'tf':
        self.waxis = 2
        self.haxis = 1
    else:
        self.waxis = 3
        self.haxis = 2
    self.img_size = img_size
    if min_size <= 0:
        raise Exception('min_size must be positive.')
    self.min_size = min_size
    self.max_size = max_size
    self.aspect_ratios = [1.0]
    if max_size:
        if max_size < min_size:
            raise Exception('max_size must be greater than min_size.')
        self.aspect_ratios.append(1.0)
    if aspect_ratios:
        for ar in aspect_ratios:
            if ar in self.aspect_ratios:
                continue
            self.aspect_ratios.append(ar)
            if flip:
                self.aspect_ratios.append(1.0 / ar)
    self.variances = np.array(variances)
    self.clip = True
    super(PriorBox, self).__init__(**kwargs)
```

Рисунок 3.10 – Метод ініціалізації фреймів

```
def get_output_shape_for(self, input_shape):
    num_priors_ = len(self.aspect_ratios)
    layer_width = input_shape[self.waxis]
    layer_height = input_shape[self.haxis]
    num_boxes = num_priors_ * layer_width * layer_height
    return (input_shape[0], num_boxes, 8)
```

Рисунок 3.11 – Метод детекції розмірів зображень

```
def call(self, x, mask=None):
    if hasattr(x, '_keras_shape'):
        input_shape = x._keras_shape
    elif hasattr(K, 'int_shape'):
        input_shape = K.int_shape(x)
    layer_width = input_shape[self.waxis]
    layer_height = input_shape[self.haxis]
    img_width = self.img_size[0]
    img_height = self.img_size[1]
    # define prior boxes shapes
    box_widths = []
    box_heights = []
    for ar in self.aspect_ratios:
        if ar == 1 and len(box_widths) == 0:
            box_widths.append(self.min_size)
            box_heights.append(self.min_size)
        elif ar == 1 and len(box_widths) > 0:
            box_widths.append(np.sqrt(self.min_size * self.max_size))
            box_heights.append(np.sqrt(self.min_size * self.max_size))
        elif ar != 1:
            box_widths.append(self.min_size * np.sqrt(ar))
            box_heights.append(self.min_size / np.sqrt(ar))
    box_widths = 0.5 * np.array(box_widths)
    box_heights = 0.5 * np.array(box_heights)
    # define centers of prior boxes
    step_x = img_width / layer_width
    step_y = img_height / layer_height
    linx = np.linspace(0.5 * step_x, img_width - 0.5 * step_x,
                       layer_width)
    liny = np.linspace(0.5 * step_y, img_height - 0.5 * step_y,
                       layer_height)
    centers_x, centers_y = np.meshgrid(linx, liny)
    centers_x = centers_x.reshape(-1, 1)
    centers_y = centers_y.reshape(-1, 1)
```

Рисунок 3.12 – Метод створення фреймів

```
    num_priors_ = len(self.aspect_ratios)
    prior_boxes = np.concatenate((centers_x, centers_y), axis=1)
    prior_boxes = np.tile(prior_boxes, (1, 2 * num_priors_))
    prior_boxes[:, ::4] -= box_widths
    prior_boxes[:, 1::4] -= box_heights
    prior_boxes[:, 2::4] += box_widths
    prior_boxes[:, 3::4] += box_heights
    prior_boxes[:, ::2] /= img_width
    prior_boxes[:, 1::2] /= img_height
    prior_boxes = prior_boxes.reshape(-1, 4)
    if self.clip:
        prior_boxes = np.minimum(np.maximum(prior_boxes, 0.0), 1.0)
    # define variances
    num_boxes = len(prior_boxes)
    if len(self.variances) == 1:
        variances = np.ones((num_boxes, 4)) * self.variances[0]
    elif len(self.variances) == 4:
        variances = np.tile(self.variances, (num_boxes, 1))
    else:
        raise Exception('Must provide one or four variances.')
    prior_boxes = np.concatenate((prior_boxes, variances), axis=1)
    prior_boxes_tensor = K.expand_dims(K.variable(prior_boxes), 0)
    if K.backend() == 'tensorflow':
        pattern = [tf.shape(x)[0], 1, 1]
        prior_boxes_tensor = tf.tile(prior_boxes_tensor, pattern)
    elif K.backend() == 'theano':
        #TODO
        pass
    return prior_boxes_tensor
```

Рисунок 3.13 – Метод створення фреймів (продовження)

3.2 Результати детектування програмної системи

Розглянемо результати детектування транспортних засобів програмного застосунку (рис. 3.14 – рис. 3.17).

Рисунок 3.14 – Результат детектування транспортних засобів

Рисунок 3.15 – Результат детектування транспортних засобів

Рисунок 3.16 – Результат детектування транспортних засобів

Рисунок 3.17 – Результат детектування транспортних засобів

ВИСНОВКИ

У рамках кваліфікаційної роботи було досліджено використання методів Deep Learning для розпізнавання транспортних засобів на зображенні. У ході дослідження було опрацьовано матеріали, пов’язані з сучасними методами детекції, а також такі популярні алгоритми як SSD, YOLO, R-CNN та інші.

Завдяки аналізу предметної області було сформовано предметну галузь використання алгоритмів SSD та YOLO, проаналізовано переваги та недоліки, розглянуто конкуренти та маркетинговий тренд розвитку цих алгоритмів.

Було розглянуто та порівняно ефективності використання вище зазначених алгоритмів, проаналізовано ресурсозатратність цих алгоритмів. Складову частину дослідження приділено аналізу Single Short Detector алгоритму. Було проаналізовано dataset, на якому проводилось тестування, експериментальне середовище, індекси мережевого навчання та оцінювання. Також, було запропоновано покращення для алгоритму SSD, проаналізовано функцію втрат до, та після покращення, сформовано та проаналізовано матрицю плутанини, зазначено результати експериментальних випробувань, а саме складено діаграму кривих точності-відгуку про оригінальний та покращений SSD.

Було запропоновано алгоритм виявлення переднього транспортного засобу для інтелектуального автомобіля на основі вдосконаленої моделі SSD. Спочатку коротко представлено мережеву модель SSD, проаналізовано та узагальнено її проблеми та недоліки у виявленні транспортних засобів. Потім проводяться цілеспрямовані вдосконалення мережевої моделі SSD, включаючи основні вдосконалення базової структури моделі SSD, використання зваженої маски в навчанні мережі та вдосконалення функції втрат. Нарешті, були проведені експерименти з виявлення транспортних засобів на основі наборів даних KITTI та власних наборів даних транспортних засобів для спостереження за роботою алгоритму в різних складних середовищах та погодних умовах. Результати тестування на базі набору даних

KITTI показують, що значення mAP досягає 92,18%, а середній час обробки одного кадру становить 15 мс. У порівнянні з існуючими методами виявлення, заснованими на глибокому навчанні, запропонований алгоритм може отримати точність і продуктивність в режимі реального часу одночасно, що сприяє реалізації швидкого і точного автоматичного виявлення транспортних засобів. Тим часом алгоритм має чудову надійність та адаптивність до навколишнього середовища для складних дорожніх умов та можливості проти перешкод для поганих погодних умов.

З точки зору точності та ефективності роботи, запропонований алгоритм виявлення транспортних засобів має видатні переваги в роботі, що має велике значення для забезпечення точної та ефективної роботи розумних автомобілів у реальних дорожніх сценах, а також корисно значно зменшити частоту дорожньо-транспортних пригод та повністю захистити життя та майно людей. У майбутньому ми можемо продовжувати зосереджуватися на алгоритмах виявлення транспортних засобів в екстремальних умовах та реалізації алгоритмів FPGA для подальшого сприяння всебічній продуктивності та практичному значенню алгоритму.

Важливу частину роботи складав процес розроблення програмного застосунку з впровадженням запропонованого покращення. Програмна реалізація включає інтерфейс для повної роботи з алгоритмом.

Результати роботи було апробовано на міжнародній науково-практичній конференції «Modern ways of solving the latest problems in science» [41].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Домінгус П. (2015). Верховний алгоритм: навч. посібник.

2. He, P., Gao, F. (2013) Study on detection of preceding vehicles based on computer vision. Guilin, China, 23–24; pp. 912–915.

3. Qu, T., Zhang, Q.Y., Sun, S.L. (2017). Vehicle detection from high-resolution aerial images using spatial pyramid pooling-based deep convolutional neural networks. Multimedia Tools, 76, pp. 21651–21663.

4. Wang, D.Y., Liu, Q.Y., Ma, L., Zhang, Y.J., Cong, H.Z. (2019). Road traffic accident severity analysis: A census-based study in China. J. Saf. Res. 70, pp. 135–147.

5. Wechsler, H. Reliable Face Recognition Methods. (2006): навч. посібник.

6. Hong, F., Lu, C.H., Liu, C., Liu, R.R., Wei, J. (2020). A traffic surveillance multi-scale vehicle detection object method base on encoder-decoder. IEEE Access, 8, pp. 47664–47674.

7. Goodfellow I., Bengio Y., Courville A. (2016). Deep Learning: навч. посібник.

8. Qu, S.R., Xu, L. (2016). Research on multi-feature front vehicle detection algorithm based on video image. In Proceedings of the 28th Chinese Control and Decision Conference (CCDC), pp. 3831–3835.

9. Maji, P., Paul. S. (2014). Scalable Pattern Recognition Algorithms: навч. посібник.

10. Planche B., Andres, E. (2019). Hands-On Computer Vision with TensorFlow 2. Leverage deep learning to create powerful image processing apps with TensorFlow 2.0 and Keras: навч. посібник.

11. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. SSD: Single shot multibox detector. In Proceedings of the 14th European Conference on Computer Vision (ECCV), Amsterdam, The Netherlands, 8–16 October 2016; pp. 21–37.

12. Liang, Q.K.; Zhu, W.; Long, J.Y.; Wang, Y.N.; Sun, W.; Wu, W.N. A real-time detection framework for on-tree mango based on SSD network. In Proceedings of the 11th International Conference on Intelligent Robotics and Applications (ICIRA), Newcastle, Australia, 9–11 August 2018; pp. 423–436.

13. Szegedy, C.; Liu, W.; Jia, Y.Q.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 1–9.

14. Liu, S.T.; Huang, D.; Wang, Y.H. Receptive Field Block Net for accurate and fast object detection. In Proceedings of the 15th European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 404–419.

15. Wei, J.; He, J.H.; Zhou, Y.; Chen, K.; Tang, Z.Y.; Xiong, Z.L. Enhanced object detection with deep convolutional neural networks for advanced driving assistance. IEEE Trans. Intell. Transp. Syst. 2020, 21, 1572–1583.

16. Hong, F.; Lu, C.H.; Liu, C.; Liu, R.R.; Wei, J. A traffic surveillance multi-scale vehicle detection object method base on encoder-decoder. IEEE Access 2020, 8, 47664–47674.

17. Ситніков Д.Е., Ситнікова П.Е., Тітов С.В., Тітова О.В. Фільтрація результуючого набору асоціативних правил з точки зору оцінки цікавості. *Системи обробки інформації*. 2021. № 1(164). С. 83-88.

18. Тітов С. В., Тітова О. В., Чорна О. С. Опис нескоротних наборів ознак в приблизних множинах з використанням систем числення. *Збірник наукових праць Харківського національного університету Повітряних Сил*. 2022. № 1(71). С. 106-10.

19. Kovalenko, A. ., Titov, S. ., Titova, E. ., & Cherna, O. . (2022). Estimation of requirements to signal parameters at V-shaped frequency distribution in mathematical model of multi-position transmitter system. *Radiotekhnika*, *2*(209), 178–184.

20. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Al-Dhaifallah M. (2022) Classification of Images Based on a System of Hierarchical Features, *Computers, Materials & Continua*, 72(1), pp. 1785–1797.

21. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2022) Cluster representation of the structural description of images for effective classification, *Computers, Materials & Continua*, 73(3), pp. 6069–6084.

22. M. Ayaz Ahmad, Irina Tvoroshenko, Jalal Hasan Baker, Liubov Kochura, and Vyacheslav Lyashenko (2020) Interactive Geoinformation Three-Dimensional Model of a Landscape Park Using Geoinformatics Tools, *International Journal on Advanced Science, Engineering and Information Technology*, 10(5), pp. 2005–2013.

23. Творошенко І.С., Табашник В.А. (2018) Розробка просторової моделі геоінформаційної підтримки людей з обмеженими можливостями, що пересуваються на інвалідних колясках, у місті Харків, *Збірник наукових праць Харківського національного університету Повітряних Сил*, 1(55), С. 122–128.

24. Tvoroshenko I.S. (2004) Structure and functions of intelligent decision-making tools in complex systems, *Artificial Intelligence*, № 4, С. 462–470.

25. Кучеренко Е.И., Творошенко И.С. (2003) Процессы принятия решений в сложных системах на основе нечетких интервальных представлений, *Вісник Національного технічного університету «ХПІ». Тематичний випуск: Системний аналіз, управління та інформаційні технології, Х.: НТУ «ХПІ»*, 1(7), С. 79–86.

26. Кучеренко Є.І., Творошенко І.С. (2011) Оперативне оцінювання простору станів складних розподілених об’єктів з використанням нечіткої інтервальної логіки, *Штучний інтелект*, № 3, С. 382–387.

27. Кучеренко Е.И., Корниловский А.В., Творошенко И.С. (2010) О методах настройки функций принадлежности в нечетких системах, *Системы управления, навигации и связи*, Т. 1, № 13, С. 94–98.

28. Кучеренко Е.И., Творошенко И.С. (2010) Прикладные аспекты моделирования нечетких процессов в сложных системах, *Збірник наукових праць Харківського університету Повітряних сил*, 1(123), С. 127–131.

29. Gorokhovatskyi V., Putyatin Y., Gorokhovatskyi O., and Peredrii O. (2018) Quantization of the Space of Structural Image Features as a Way to Increase Recognition Performance, *The Second IEEE International Conference on DataStream Mining & Processing 21-25 August 2018. Lviv, Ukraine*, pp. 464–467.

30. Tvoroshenko I., and Tkachenko D. (2020) Mechanisms of image classification based on descriptors of local features, *Abstracts of IV International Scientific and Practical Conference «Integration of scientific bases into practice» (October 12-16, 2020). Stockholm, Sweden*, pp. 443–448.

31. Гороховатский В.А. (2003) Распознавание изображений в условиях неполной информации. Харків: ХНУРЭ, 112 с.

32. Tvoroshenko I., and Dziubenko M. (2020) Modern methods of analysis of the movement scheme using video detection of vehicles, *Abstracts of V International Scientific and Practical Conference «Study of modern problems of civilization» (October 19-23, 2020). Oslo, Norway*, pp. 422–428.

33. Gorokhovatsky V. (2014) Structural Analysis and Intellectual Data Processing in Computer Vision. SMIT: Kharkiv, Ukraine, 316 p.

34. Tvoroshenko I., and Gorokhovatskyi V. (2022) The Application of Hybrid Intelligence Systems for Dynamic Data Analysis, *International Journal of Engineering and Information Systems*, 6(2), pp. 40–48.

35. Tvoroshenko I., and Zarivchatskyi R. (2020) Analysis of existing methods for searching object in the video stream, *Abstracts of VI International Scientific and Practical Conference «About the problems of science and practice, tasks and ways to solve them» (October 26-30, 2020). Milan, Italy*, pp. 500–505.

36. Кучеренко Є.І., Творошенко І.С., Анопрієнко Т.В. (2016) Моделювання та оцінювання станів складних об’єктів із застосуванням формальної логіки, *Системи обробки інформації*, № 2, С. 76–82.

37. Tvoroshenko I. (2019) Development of models of spatial analysis of status of interactive processes of complex systems.

38. Творошенко І.С. (2018) Особливості застосування сучасних принципів штучного інтелекту до розробки ефективних механізмів

моделювання складних систем, *Science and Technology of the Present Time: Priority Development Directions of Ukraine and Poland*, pp. 118–121.

39. Творошенко И.С., Дехтярь А.П. (2005) Информационные технологии в задачах компьютерной диагностики с использованием интеллектуальных систем. *Клиническая информатика и Телемедицина. Компьютерная Медицина–2005: материалы междунар. научн.-технич. конф., Харьков,* p. 138.

40. Гороховатський В.О., Творошенко І.С. (2022) Аналіз багатовимірних даних за описом у формі множини компонент: монографія. Харків: ХНУРЕ, 124 с.

41. Ященко А.В. (2022). Аналіз існуючих підходів до вирішення задач детекції об’єктів. Матеріали XXXVII Міжнародна науково-практична конференція «Modern ways of solving the latest problems in science».