МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ АВТОМАТИЗОВАНИХ СИСТЕМ

Кваліфікаційна робота

другий (магістерський)
(рівень вищої освіти)

на тему **Комп'ютерна система для виділення об'єктів у відеопотоці із заміщенням вмісту**

Виконав: студент 2 курсу, групи 8.1219-пзс
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Інженерія програмного забезпечення
(код і назва освітньої програми)
В.Ю. Іщенко
(ініціали та прізвище)
Керівник доцент, А. І. Безверхий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)
Рецензент директор ТОВ «Дісітел»
П. О. Лютий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя
2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ

Кафедра _____ програмного забезпечення автоматизованих систем
Рівень вищої освіти _____другий (магістерський)_____
Спеціальність ___121_Інженерія програмного забезпечення_____
(код та назва)
Освітня програма ___Інженерія програмного забезпечення
(код та назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри_____ В.Г. Вербицький
"_01_" __вересня_____2020_року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Іщенку Володимиру Юрійовичу
(прізвище, ім'я, по батькові)

1.Тема роботи __Комп'ютерна система для виділення об'єктів у відеопотоці із заміщенням вмісту_____

керівник роботи _____Безверхий Анатолій Ігорович, доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ЗНУ від ___"25" травня 2020 року № 600-с_____

2. Строк подання студентом кваліфікаційної роботи_____30.11.2020_____
3. Вихідні дані магістерської роботи
- комплект нормативних документів ;
- технічне завдання до роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
- огляд та збір літератури стосовно теми кваліфікаційної роботи;
- огляд та аналіз існуючих рішень та аналогів;
- дослідження проблеми розпізнавання мов та розробка методів її вирішення;
- створення програмного продукту та його опис;
- перелік вимог для роботи програми;
- дослідження поставленої проблеми та розробка висновків та пропозицій.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
14 слайдів презентації

6. Консультанти розділів магістерської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата Завдання прийняв

7. Дата видачі завдання 01.09.2020

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів магістерської роботи	Примітка
1	Аналіз предметної області	02.09-10.09.20	виконано
2	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником	11.09-12.09.20	виконано
3	Аналіз існуючих методів рішення	13.09-14.09.20	виконано
4	Дослідження засобів виявлення об'єктів на зображеннях	15.09-20.09.20	виконано
5	Дослідження засобів виявлення контурів об'єктів та трансформації зображень	21.09-26.09.20	виконано
6	Узгодження подальших дій з науковим керівником	27.09-28.09.20	виконано
7	Збір тренувального набору для нейронної мережі з відкритих джерел та його попередня підготовка	29.09-13.10.20	виконано
8	Навчання моделі нейронної мережі за допомогою підготовленого тренувального набору	14.10-16.10.20	виконано
9	Представлення отриманих результатів науковому керівнику та узгодження плану подальшого дослідження	17.10-19.10.20	виконано
10	Реалізація функціоналу для виявлення контурів об'єктів на зображенні та трансформації зображень	20.10-09.11.20	виконано
11	Порівняння вихідних характеристик роботи системи за допомогою вхідних тестових відео	10.11-17.11.20	виконано
12	Реалізація користувацького інтерфейсу для комп'ютерної системи	18.11-22.11.20	виконано
13	Оформлення звіту	23.11-27.11.20	виконано

Студент _____ Іщенко В.Ю.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Безверхий А.І.
(підпис) (прізвище та ініціали)

Нормоконтроль пройдено

Нормоконтролер _____ Скрипник І.А.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Сторінок: 110

Рисунків: 44

Таблиць: 9

Джерел: 91

Формул: 9

Іщенко В.Ю. Комп’ютерна система для виділення об’єктів у відеопотоці із заміщенням вмісту.

Кваліфікаційна робота для здобуття ступеня вищої освіти магістра за спеціальністю 121 — Інженерія програмного забезпечення, науковий керівник А.І. Безверхий. Інженерний навчально-науковий інститут ЗНУ. 2020.

Мета кваліфікаційної роботи полягає у дослідженні методів для розпізнавання об’єктів на відео та знаходженні ефективного способу заміни їх на бажаний зміст. Також метою є створення програмної системи, яка, отримуючи на вхід відеофайл, розпізнає на ньому білборди та в результаті роботи поверне відео з заміненим вмістом на виявлених білбордах.

Досліджено методи та сучасні моделі нейронних мереж для виявлення об’єктів на зображеннях, проблематику і можливості розробки і використання системи. При розробці комп’ютерної системи була застосована модель нейронної мережі YOLOv4 та фреймворк машинного навчання Darknet для вирішення задачі виявлення білбордів; бібліотека OpenCV застосована для роботи з зображеннями та відео; методи обробки зображення, трансформації та їх накладення реалізовані мовою Python.

Ключові слова: *КОМП’ЮТЕРНА СИСТЕМА, НЕЙРОННА МЕРЕЖА, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА, РОЗПІЗНАВАННЯ БІЛБОРДІВ, YOLO, DARKNET, OPENCV, НАВЧАННЯ МЕРЕЖІ.*

SUMMARY

Pages: 110

Figures: 44

Tables: 9

Sources: 91

Formulas: 9

Ishchenko Volodymyr. Computer system for selecting objects in video stream with content substitution.

Qualification work for higher master's degree in specialty 121 — Software Engineering, supervisor Anatolii Bezverkhyi. Engineering Educational Scientific Institute of Zaporizhia National University. 2020.

The aim of the qualification work is to explore methods for recognizing objects in video and finding an effective way to replace them with the desired content. The goal is also to create a software system that, upon receiving a video file, recognizes billboards on it and as a result returns the video with the replaced content on the detected billboards.

In this work methods were researched and modern models of neural networks for detecting objects in images, problems and possibilities of system development and use. The YOLOv4 neural network model and the deep learning framework Darknet were used for computer system development to solve the problem of detecting billboards; OpenCV library is used for images and videos processing; methods of image processing, transformation and their overlay are implemented in Python.

Keywords: *COMPUTER SYSTEM, NEURAL NETWORK, CONVOLUTIONAL NEURAL NETWORK, BILLBOARD DETECTION, YOLO, DARKNET, OPENCV, NETWORK TRAINING*.

ЗМІСТ

ВСТУП

Актуальність теми

Тривалий час питання розпізнавання образів розглядалися людиною лише з позицій методів біології та психології. При цьому метою вивчення були в основному якісні характеристики, що не дозволяють розкрити і точно описати відповідний механізм. Якщо і виходили числові характеристики, то вони, як правило, були пов’язані з вивченням рецепторів, таких як органи зору, слуху, дотику. Що ж стосувалося характеристик прийняття рішень, то до їх оцінки справа не доходила. Кібернетика дозволила ввести у вивчення психологічного процесу розпізнавання образів, що лежить в основі прийняття будь-яких рішень, кількісні методи, що відкрило принципово нові можливості в дослідженні та проектуванні автоматичних і автоматизованих систем розпізнавання образів.

З ростом автоматизації виробництва, виникла необхідність забезпечити роботу багатьох технологічних процесів без участі людини. Головним чином це процеси, пов’язані з рутинною, одноманітною роботою або процеси, які небезпечні для людини. У таких процесах доцільно людину замінити на автомати, що реагують на різні відхилення параметрів технологічного процесу від номінальних. Створення таких автоматів було першим кроком на шляху до побудови систем, які розпізнають. Згодом, такі автомати ставали все складнішими, а з появою комп’ютерів, відкрилися найширші можливості для застосування автоматів-розпізнавачів, робота яких ґрунтується на цифровій обробці даних.

Крім рутинних операцій, які повинен був виконувати робочий, існують випадки, при яких людина не в змозі розв’язувати поставлену задачу або приймати рішення з необхідною швидкістю, зумовлену обставинами (наприклад: протиракетний маневр літака в складних метеоумовах; висновок з робочого режиму АЕС тощо).

Рекламний щит є ефективним комерційним носієм рекламної інформації про товари чи послуги. Наразі керування його даними зроблено використовуючи звичайний підхід, наприклад шляхом захоплення білборда за допомогою камери та написання реквізитів про місцезнаходження у примітках. Рекламний “шум” має негативну конотацію в маркетингу, це означає велику кількість рекламного безладу, що споживачі втомлюються і важко запам’ятовують конкретні повідомлення [1].

Зростання телевізійних та онлайн-відеозаписів забезпечило кілька напрямків передачі інформації аудиторії про рекламований бренд. Деякі дослідження [2] показали: спостерігається стабільне зростання загальних доходів від інтернет-реклами за останнє десятиліття та зростання прогнозується, що в найближчі роки постійно зростатиме.

Мета роботи

Мета роботи полягає у дослідженні алгоритмів для розпізнавання об’єктів на відео та знаходженні ефективного способу заміни їх на бажаний зміст. Також метою є створення програмної системи, яка, отримуючи на вхід відеофайл або відеопотік, розпізнає на ньому білборди та в результаті роботи поверне відео з заміненим змістом на розпізнаних білбордах.

Об’єкт дослідження

Відеопотік, що надходить до системи через відеокамеру чи з мережі Інтернет, або збережений відеофайл (у файловій системі).

Предмет дослідження

Детектування, розпізнавання об’єктів (на прикладі білбордів) та заміна контенту на виявленому об’єкті на відео.

Методи дослідження

Для розв’язання задач використовуються такі методи дослідження:

– Аналіз джерел про нейронні мережі, алгоритми локалізації та розпізнання об’єктів на зображенні;
– Пошук алгоритмів для заміни вмісту на зображенні;
– Пошук існуючих систем розпізнавання об’єктів на зображенні;
– Синтез отриманих результатів досліджень.

Наукова новизна

Одержані результати є наочним відображення переваг та недоліків використання нейронних мереж для виявлення об’єктів та заміни цих об’єктів у відеофайлах або відеопотоці.

Проблема реклами актуальна. Відкритих рішень даної проблеми немає.

Практичне значення одержаних результатів

На базі отриманих результатів можна зрозуміти які алгоритми та фреймворки є найбільш ефективними для вирішення задач розпізнавання об’єктів, зокрема білбордів, заміни їх змісту на бажаний у відеофайлі або відеопотоці.

Проаналізувавши дану роботу можна зробити висновки щодо найбільш вдалого способу заміни виявлених об’єктів на зображенні або кадрах відео. Ознайомитися з проблемами, які виникають під час розробки такої системи, та методами їх вирішення.

У результаті було розроблено комп’ютерну систему, яка розпізнає більшість білбордів та замінює їх зміст на кадрах відео або відеопотоку.

Апробація результатів

Результати роботи, викладені у кваліфікаційній роботі магістра, були опубліковані в збірнику наукових праць студентів, аспірантів і молодих вчених “Молода наука-2020” [3] і представлені на XXV науково-технічній конференції студентів, магістрантів, аспірантів, молодих вчених та викладачів [4].

Глосарій

Анотації навчального набору — це метадані цифрового зображення з навчального набору, які зазвичай включають підписи, ключові слова, маркери місцеположення об'єктів, зображених на ньому, їх категорії тощо. Для *анотування*, процесу створення файлів анотацій, існує відкрите та платне програмне забезпечення [5]. *Анотаторами* називають працівників, що створюють анотації.

Аугментація (збільшення) даних — це методика створення додаткових навчальних даних з наявних даних. Для досягнення хороших результатів глибокі мережі повинні навчатися на дуже великому обсязі даних. Отже, якщо вихідний навчальний набір містить обмежену кількість зображень, необхідно виконати збільшення, щоб поліпшити результати роботи моделі.

Батч (*batch*) — порція об'єктів з навчального набору.

Згорткові нейронні мережі — в машинному навчанні — це клас глибинних штучних нейронних мереж прямого поширення, який успішно застосовувався до аналізу візуальних зображень. Згорткові мережі є аналогом біологічних процесів, в яких схема з'єднання нейронів відповідає організації зорової кори мозку. Окремі нейрони кори реагують на стимули лише в обмеженій області зорового поля, відомій як рецептивне поле. Рецептивні поля різних нейронів частково перекриваються таким чином, що вони покривають усе зорове поле.

Нейронна мережа — це низка алгоритмів, які намагаються розпізнати основні взаємозв'язки в наборі даних за допомогою процесу, що імітує спосіб роботи людського мозку. У цьому сенсі нейронні мережі відносяться до систем нейронів, або органічних, або штучних. Такі системи навчаються вирішувати задачі (поступально покращують свою продуктивність на них), розглядаючи приклади, загалом без спеціального програмування під задачу. Використовуються в різних сферах застосування у фінансових послугах, від прогнозування та маркетингових досліджень, до виявлення шахрайства та оцінки ризиків.

Обмежувальна рамка (Bounding box) — абстрактне обведення образа на зображенні у контексті задачі локалізації об’єктів. Рамка позначає видиму частину об’єкта на зображенні, а не передбачуваний загальний розмір об’єкта. Зазвичай образ обводять координатами верхнього лівого кута та правого нижнього, або координатами центру об’єкта та шириною і висотою рамки. Істиною обмежувальною рамкою (*ground truth bounding box*) називають рамку, яку поставила людина (анотатор) навколо дійсно присутнього об’єкта на зображенні. Прогнозована обмежувальна рамка (*predicted bounding box*) — рамка навколо ймовірно розташованого об’єкта на зображенні, яку спрогнозувала модель нейронної мережі в результаті своєї роботи.

Перенавчання (Overfitting) — в машинному навчанні — явище, коли побудована нейронна мережа витягує занадто багато інформації з окремих випадків, забуваючи відповідну інформацію загального випадку. Це пов’язано з тим, що при побудові моделі (“в процесі навчання”) в навчальній вибірці виявляються деякі випадкові закономірності, які відсутні в генеральній сукупності.

Трансферне навчання (Transfer learning) — це застосування до розв’язання задачі знань, взятих нейронною мережею при вирішенні іншої задачі.

Фреймворк (*Framework*) — абстракція, в якій програмне забезпечення, що забезпечує загальну функціональність, може вибірково змінюватися додатковим кодом, написаним користувачем, забезпечуючи таким чином програмне забезпечення, яке стосується застосунків. Можна вважати своєрідною комплексною бібліотекою.

РОЗДІЛ 1 АНАЛІЗ ЗАДАЧІ ВИДІЛЕННЯ ОБ'ЄКТІВ

1.1 Загальні відомості про розпізнавання образів

Розпізнавання образів є процесом розпізнавання шаблонів за допомогою алгоритму машинного навчання. Воно може бути визначено як класифікація даних на основі вже отриманих знань або на статистичній інформації, отримані з моделей та / або їх представлення. Одним із важливих аспектів розпізнавання шаблону є його потенціал застосування. Розпізнавання мови, ідентифікація динаміка, розпізнавання мультимедійних документів (MDR), автоматична медична діагностика. У типовому застосунку розпізнавання образів необроблені дані обробляються та перетворюються у форму, придатну для використання машиною. Розпізнавання образів включає класифікацію та кластеризацію.

Образ — класифікаційне угрупування в системі класифікації, що об'єднує (виділяє) певну групу об'єктів за деякою ознакою. Образи мають характерні об'єктивні властивості в тому сенсі, що різні люди, що навчаються на різному матеріалі спостережень, здебільшого однаково і незалежно один від одного, класифікують одні й ті ж об'єкти. У класичній постановці задачі розпізнавання універсальна множина розбивається на частини — образи. Кожне відображення якого-небудь об'єкта на сприймаючі органи системи, що розпізнає, прийнято називати зображенням об'єкта, а множини таких зображень, об'єднані загальними властивостями, являють собою образи [6].

Навколо дуже багато інформації, тому потрібно звертати увагу лише на те, що має значення. Замість того, щоб переглядати статистику відвідувачів вашого веб-сайту, ви можете скористатися статистикою Google Analytics, щоб перевірити, чи були підозрілі сплески. Замість того, щоб гуляти закордонним містом з товстим словником, ви можете навести камеру на табличку чи меню ресторану і зрозуміти, що там написано.

Деякі приклади розпізнавання образів:

- Програмне забезпечення для розпізнавання обличчя містить дані, пов’язані з характеристиками обличчя людини, і використовує алгоритм для відповідності конкретного шаблону окремому запису в базі даних.
- Алгоритми розпізнавання візерунків у метеорологічному програмному забезпеченні [7] можуть виявляти періодичні зв’язки між погодними даними, які можна використовувати для прогнозування можливих майбутніх погодних подій.
- Програмні правила виявлення вторгнень у мережу (Network intrusion detection — NID) описують моделі поведінки та події, які можуть вказувати на нелегітимний трафік.

Використання методів розпізнавання образів дає велику кількість переваг для людини. Це не тільки допомагає в аналізі тенденцій, але також допомагає робити прогнози.

- Це допомагає в ідентифікації предметів на різних відстанях і кутах.
- Легко та високо автоматизовано.
- Не вимагає здатності мислити поза коробкою.
- Надзвичайно корисно у фінансовій галузі зробити важливі прогнози щодо продажів.
- Ефективне вирішення проблем у режимі реального часу.
- Корисне в галузі медицини для криміналістичного аналізу та послідовності ДНК.

1.2 Використання засобів для розпізнавання образів

Розпізнавання образів є однією з найфундаментальніших проблем теорії інтелектуальних систем. Часто вживається термін “класифікація”, замість “розпізнавання”, оскільки у багатьох випадках вони розглядаються як синоніми, але не є повністю взаємозамінюваними. Зазвичай задачі розпізнавання поділяють на три типи:

- *Ідентифікація*, коли необхідно вирізнити певний об’єкт серед об’єктів такого ж класу.
- *Кластеризація (класифікація без учителя)*, коли система сама створює групи об’єктів, виділивши їх схожі ознаки; при цьому класи об’єктів користувачем не задавались — це виконує система.
- *Задача класифікації* відображає проблему віднесення об’єкту до певного класу. У даній роботі саме для цієї проблеми (та для локалізації) досліджуються методи вирішення [8].

1.3 Проблеми розпізнавання образів

Варіація точки зору: У реальному світі сутності на зображенні вирівнюються в різних напрямках, і коли такі зображення подаються в систему, вона передбачає неточні значення. Тобто система не розуміє, що зміна вирівнювання образу (ліворуч, праворуч, знизу, зверху) не зробить його різним.

Варіація масштабу: варіації в розмірах впливають на класифікацію об’єкта. Чим ближче ви розглядатимете об’єкт, тим більшим він виглядатиме за розмірами та навпаки.

Деформація: Предмети не змінюються, навіть якщо вони деформовані. Система дізнається на ідеальному зображенні та формує уявлення про те, що конкретний об’єкт може бути лише у певній формі. Ми знаємо, що в реальному світі форма змінюється, і, як наслідок, трапляються неточності, коли система зустрічає деформоване зображення предмета.

Варіація між класами: певний об’єкт змінюється в межах класу. Вони можуть бути різної форми, розміру, але все одно представляють один і той же клас. Наприклад, ґудзики, стільці, пляшки, сумки бувають різних розмірів та зовнішнього вигляду.

Оклюзія (перекривання): певні інші об’єкти перешкоджають повному огляду образу та спричиняють подачу неповної інформації до системи. Роз-

робник системи має розробити алгоритм, який буде чутливим до таких варіацій та буде складатися з широкого діапазону вибірок даних [9].

1.4 Загальні відомості про заміну об'єктів на зображеннях

Фотоманіпуляції, коли з'єднують частини різних зображень і графічних елементів з метою отримання абсолютно нового твору, були створені в XIX столітті; незабаром їх почали застосовувати до кінофільмів. Технологія невпинно вдосконалювалася протягом XX століття завдяки цифровому відео.

Починаючи з 90-х років розроблялась технологія Deepfake [10] дослідниками в академічних установах, пізніше — любителями в Інтернет-спільнотах.

Сучасні академічні проекти були зосереджені на створенні більш реалістичних відео та на вдосконаленні методик. Програма "Synthesizing Obama", опублікована в 2017 році, модифікує відеокадри колишнього президента Барака Обами, щоб зобразити його, як він перебирає слова, що містяться в окремому аудіозаписі [11].

У серпні 2018 року дослідники Каліфорнійського університету Берклі опублікували документ, в якому представили додаток підроблених танців, який може створити враження майстерних танцювальних здібностей за допомогою AI [12].

Виробництво фільмів сьогодні — вкрай затратний процес з орендою камер, студій та оплатою роботи акторів. Розвиток DeepFake дозволить скоротити витрати на знімальний процес, монтаж і спецефекти. Достатньо записати один рекламний ролик зі знаменитістю, після чого обличчя знаменитості можна переносити на відео з місцевими акторами, які промовлятимуть рекламні слогани на рідній мові. Так можна домогтися ефекту, ніби знаменитість говорить на мові країни дистрибуції продукту.

Технологія перенесення міміки може застосовуватися для створення цифрових двійників у іграх і віртуальній (або доповненій) реальності. Дже-

релами особи можуть служити самі учасники гри чи іншого простору. Це підвищує емоційне залучення в продукт.

У березні 2018 року художник Джозеф Айерл опублікував відеороботу “Un'emozione per semper 2.0” (“The Italian Game”). Художник працював з технологією Deepfake, щоб створити синтетичну версію кінострічки 80-х років Орнелли Муті, подорожуючи у часі з 1978 по 2018 рік. Художник використовував подорож часу Орнелли Муті для дослідження поколінь, а також досліджував питання про роль провокацій у світі мистецтва [13].

1.5 Методи машинного навчання для вилучення ознак

1.5.1 Загальні відомості

Машинне навчання є підгалуззю штучного інтелекту, яка часто застосовує статистичні прийоми для надання комп’ютерам здатності “навчатися” (тобто, поступово покращувати продуктивність у певній задачі) з даних, без того, щоби бути програмованими явно [14].

Вилучення ознак — це процес зменшення розмірності, за допомогою якого початковий набір вихідних даних зводиться до більш керованих груп для обробки. Характеристикою цих великих наборів даних є велика кількість змінних, для обробки яких потрібно багато обчислювальних ресурсів. Вилучення ознак — це назва методів, які вибирають та/або поєднують змінні в ознаки, ефективно зменшуючи обсяг даних, які необхідно обробити, зберігаючи при цьому точний та повний опис вихідного набору даних. Цей процес корисний, коли потрібно зменшити кількість ресурсів, необхідних для обробки, не втрачаючи важливої або відповідної інформації. Він також може зменшити кількість надлишкових даних для даного аналізу. Крім того, зменшення даних та ресурсів машини у створенні змінних комбінацій (ознак) сприяють швидкості вивчення та узагальнення в процесі машинного навчання.

У галузі машинного навчання та розпізнавання образів зменшення розмірності є важливою сферою, де пропонується багато підходів. Методи вилу-

чення та відбору ознак використовуються ізольовано або в поєднанні з метою підвищення ефективності, наприклад, оціночної точності, візуалізації та зрозумілості засвоєних знань [15]. Як правило, ознаки можна класифікувати як: відповідні, нерелевантні або зайві. Перевага вибору об'єкта полягає в тому, що важлива інформація, пов'язана з однією ознакою, не втрачається, але якщо потрібен невеликий набір ознак, а оригінальні об'єкти дуже різноманітні, існує ймовірність втрати інформації, оскільки якусь ознаку потрібно пропустити. З іншого боку, зі зменшенням розмірності, розмір простору ознак часто можна зменшити, не втрачаючи інформацію про вихідний простір ознак. Недоліком вилучення ознак є той факт, що лінійне поєднання вихідних ознак, як правило, не піддається інтерпретації, та інформація про те, наскільки сприяє оригінальна ознака, часто втрачається [16]. Щоб отримати переваги від методів зменшення розмірності з метою максимізації точності алгоритму навчання, потрібно усвідомлювати різні переваги цих методів. У роботі [17] були виявлені наступні переваги вибору ознак:

- зменшення розмірності простору ознак, обмеження вимог до зберігання та збільшення швидкості алгоритму;
- видалення зайвих, нерелевантних чи шумних даних;
- безпосередні наслідки для завдань аналізу даних прискорюють час роботи алгоритмів навчання;
- покращення якості даних;
- підвищення точності отриманої моделі;
- зменшення набору ознак для економії ресурсів під час наступного раунду збору даних або під час використання;
- покращення продуктивності, щоб отримати точність прогнозування;
- розуміння даних для отримання знань про процес, який створив дані або просто візуалізує дані.

1.5.2 Задачі локалізації та класифікації

Проблему виділення об’єктів представляють у вигляді задач:

1. *Локалізація* — пошук об’єктів типу “білборд” на кадрі відео. Локалізація — задача виявлення об’єкта певної категорії на зображенні, як правило, шляхом вказання щільно обрізаної обмежувальної рамки навколо нього. Прогнозування об’єкта визначає обмежувальну рамку об’єкта-кандидата; воно вважається правильною локалізацією, якщо рамка достатньою мірою перекриває істинну обмежувальну рамку навколо даного об’єкта. У літературі розрізняють “локалізацію” як пошук одного екземпляра категорії об’єкта, та “виявлення об’єкта” — пошук усіх об’єктів певної категорії на даному зображенні.

2. *Класифікація* — визначення типів конкретних об’єктів. Є найважливішим завданням у програмах комп’ютерного зору. Для вирішення цієї задачі розроблено широкий спектр методів: штучні нейронні мережі, статистичні підходи, нечітка логіка, дерево рішень та машина підтримки векторів (SVM).

У даній роботі використовується рішення для задачі виявлення об’єкта класу “білборд” на зображенні (кадрах відео).

1.5.3 Метод Віоли-Джонса та алгоритм Канаде-Лукас-Томасі

Використання обличчя людини як ключа безпеки та розпізнавання обличчя за допомогою біометричних технологій привернули велику увагу завдяки своїй здатності до широкого використання як у правоохоронних, так і в неправових органах. Алгоритм, запропонований Віолою і Джонсом, для локалізації об’єктів на зображенні з самого початку створювався для пошуку облич, проте може використовуватися й для інших об’єктів. Метод став справжнім проривом в області локалізації об’єктів та набув великої популярності завдяки високій точності й серйозній теоретичній основі [18].

Віола та Джонс зробили три основні внески в область виявлення об’єктів:

- Інтегральне зображення: створюється шляхом обчислення суми значень сірого для всіх пікселів зверху та ліворуч від кожного пікселя на зображенні, дозволяючи обчислювати будь-яку прямокутну суму в “чотирьох посиланнях на масив”.
- Класифікатор: невеликий набір критичних ознак, що забезпечує швидку класифікацію порівняно з пошуком зображень для всіх можливих ознак.
- Каскадні класифікатори: метод поєднання послідовно більш складних класифікаторів, що забезпечує хороший баланс високих швидкостей виявлення та обчислювальної ефективності. Вони попередньо обчислюються шляхом запуску варіації алгоритму навчання AdaBoost щодо маркованого набору даних позитивних та негативних зображень та вилучення загальних рис.

Вхідне зображення перетворюється у відтінки сірого, так що кожен піксель у зображенні може бути представлений значенням інтенсивності 0–255, перш ніж обчислюватися в його цілісне зображення. Це зображення поділяється на вікна та сканується на наявність ознак, визначених у першому класифікаторі каскаду. Вікна, які передають перший класифікатор, потім обробляються другим класифікатором і так далі, поки не залишаться лише вікна, що містять виявлені об’єкти. Ознаки Віоли-Джонса відрізняються від базових ознак Хаара (рис. 1) тим, що вони покладаються на 2-4 прямокутні області для представлення об’єкта. Ці ознаки можуть символізувати чорно-білі прямокутники, які відображають природні темні та світлі ділянки на нетиповому зображенні об’єкта, який намагаються виявити. Алгоритм приймає суму значень пікселів під білими областями за мінусом чорних областей, і область має цю ознаку, якщо сума знаходиться в межах певного порогу. Кожна ознака повинна “проявлятися на рівні лише трохи кращому, ніж випадковий”, щоб підвищити загальну точність виявлення і бути вартим включення в класифікатор.

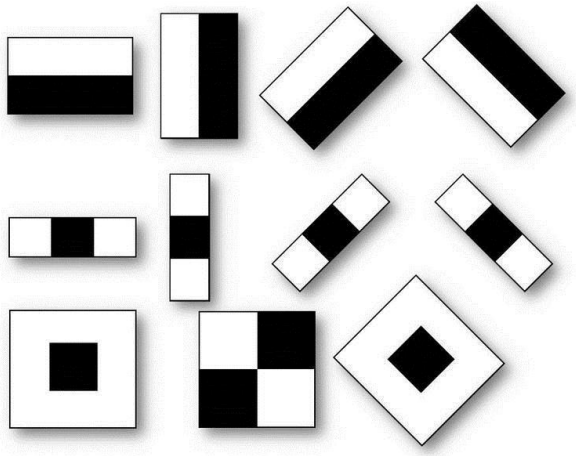

Рис. *1 Ознаки Хаара*

Алгоритм Канаде-Лукас-Томасі (KLT) [19] — широко використовуваний у комп'ютерному зорі метод для виділення ознак. Він був запропонований переважно для вирішення цієї проблеми, оскільки традиційні методи реєстрації зображень, як правило, трудомісткі. KLT використовує інформацію просторової інтенсивності для керування пошуком позиції, яка дає найкращу відповідність. Це швидше, ніж традиційні методи дослідження набагато меншої кількості потенційних відповідностей між зображеннями.

У роботі [20] порівнюються метод Віоли-Джонса та KLT: протестували їх у різних ситуаціях, щоб виявити точність, яка буде змінюватися залежно від яскравості ситуацій, кількості граней, освітлення, швидкості руху тощо. Для порівняння алгоритмів вони використали кілька зображень з різних джерел; класифікували їх на п'ять категорій — лицьова сторона, погляд вліво, погляд вправо, погляд вгору та погляд вниз. Далі перші три категорії були розділені на чотири підкатегорії — яскраві, дуже яскраві, темні та дуже темні. Метод Віоли-Джонса виявив 87% від загальної кількості облич, тоді як KLT виявив лише 84%. Цікаво зазначити, що з усіх зображень, які були застосовані до обох алгоритмів, перший виявив обличчя в кількох зображеннях, які не були виявлені алгоритмом KLT, але не було таких зображень де обличчя були б виявлені KLT і не виявлені методом Віоли-Джонса. Загальні результати порівняння наведені в таблиці 1.

Таблиця *1*

Загальні результати виявлення обличчя за обома алгоритмами

	Метод Віоли-Джонса	Метод KLT
Лицьова сторона	97%	90%
Погляд вліво	90%	85%
Погляд вправо	88%	83%
Погляд вгору	80%	80%
Погляд вниз	80%	80%
Всього	87%	84%

1.5.4 Гістограми орієнтованих градієнтів

Метод (HOG) заснований на підрахунку кількості напрямків градієнта в локальних областях зображення [21]. Основна ідея в тому, що зовнішній вид і форма об’єкта на ділянці зображення можуть бути описані розподілом градієнтів інтенсивності або напрямком країв навіть без інформації про точне місцезнаходження даних градієнтів або країв. Перевага даного алгоритму в підтримці інваріантності спотворень об’єкта, крім зміни його форми. Але при цьому швидкість роботи даного методу нижче багатьох інших.

На ресурсі [22] автор наводить приклад виявлення людей на зображенні з застосуванням HOG, який є в бібліотеці OpenCV. Дескриптор ознак HOG, що використовується для виявлення пішоходів, обчислюється на фрагменті (патчі) зображення розміром 64 × 128. Звичайно, зображення може бути будь-якого розміру. Зазвичай фрагменти в різних масштабах аналізуються в багатьох місцях зображення. Єдиним обмеженням є те, що аналізовані патчі мають фіксоване співвідношення сторін. У даному випадку патчі повинні мати співвідношення сторін 1:2. Для ілюстрації цього моменту автор показав велике зображення розміром 720 × 475, обрав патч розміром 100 × 200 для обчислення дескриптора ознак HOG. Цей патч обрізається із зображення та зменшується до розміру 64 × 128 (рис. 2).

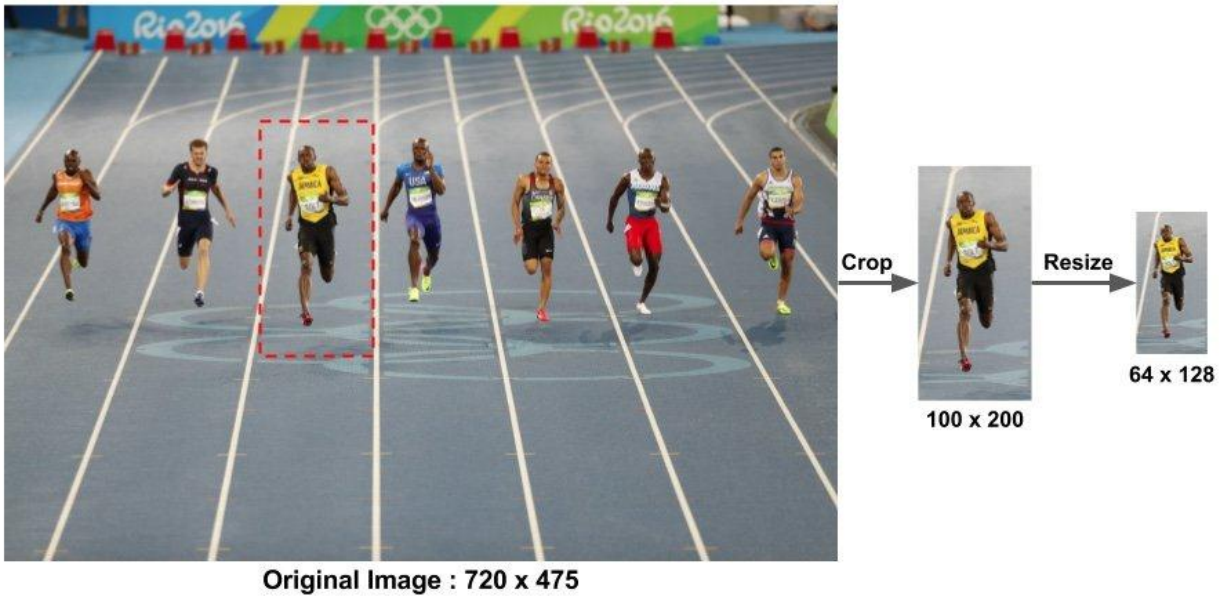

Рис. *2 Виділення патчу з зображення*

Далі зображення ділиться на 8×8 комірок (рис. 3), і для кожної з комірок розраховується гістограма градієнтів.

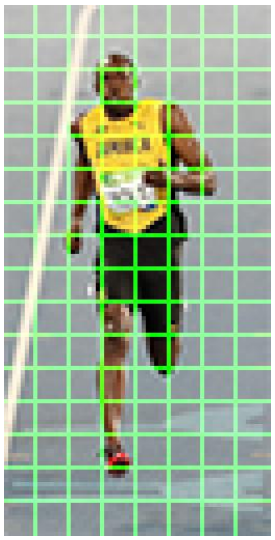

Рис. *3 Розділення зображення на комірки*

Однією з важливих причин використання дескриптора ознак для опису ділянки зображення є те, що він забезпечує компактне представлення. Патч зображення 8×8 містить значення 8×8×3=192 пікселів. Градієнт цього патча

містить 2 значення (величину та напрямок) на піксель, що додає ще 8×8×2=128 чисел.

Гістограма є, по суті, вектором (або масивом) із 9 чисел, що відповідають кутам 0, 20, 40, 60,…, 160. На рисунку 4 зображено величини та напрями градієнтів.

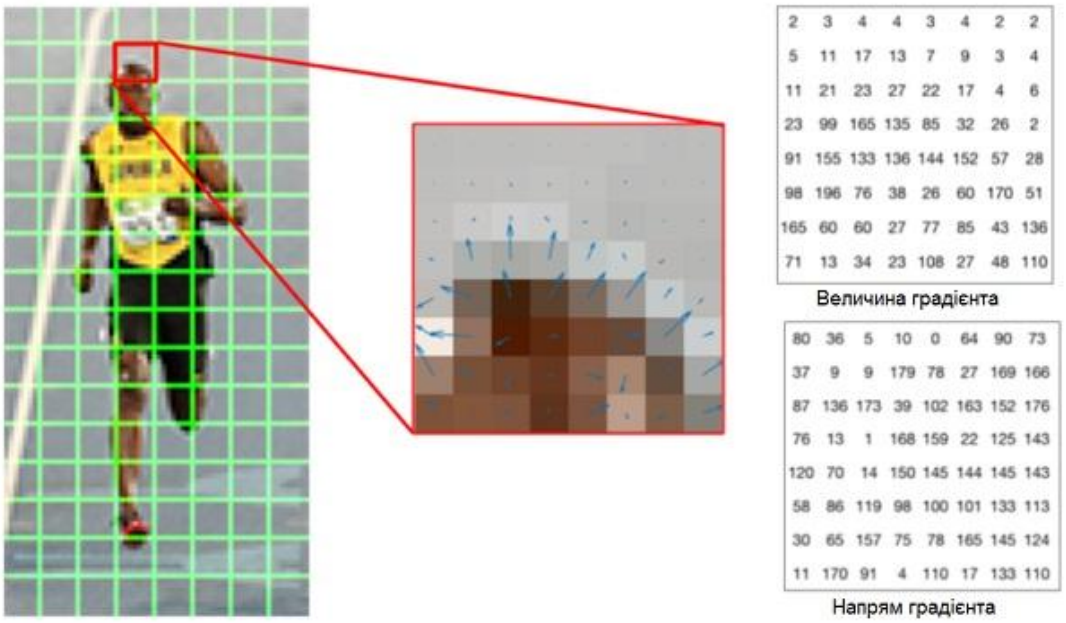

Рис. *4 Патч RGB та градієнти, представлені стрілками — у центрі; градієнти в одному патчі у числовому вигляді — праворуч*

Наступним кроком є створення гістограми градієнтів у цих клітинках 8 × 8. Вклади всіх пікселів у клітинках 8 × 8 складаються, щоб створити 9-бінову гістограму. Для обчислення остаточного вектору ознак для всього патча зображення, вектори 36 × 1 об’єднуються в один гігантський вектор.

Дескриптор HOG патча зображення зазвичай візуалізується шляхом побудови нормалізованих гістограм 9 × 1 у клітинках 8 × 8. На рисунку 5 помітно, що домінантний напрямок гістограми фіксує форму зображення людини, особливо навколо тулуба та ніг.

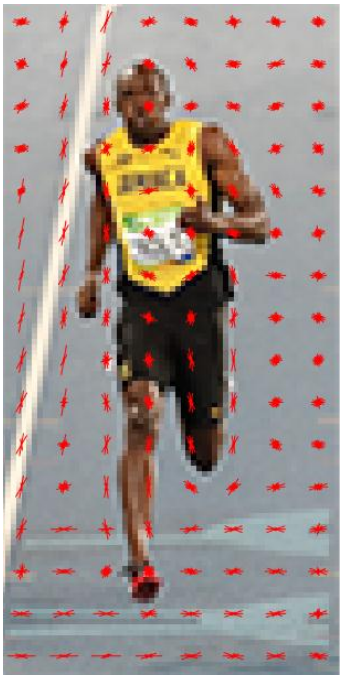

Рис. *5 Зображення з візуалізованими HOG*

У більших масштабах ознаки HOG надають більше глобальної інформації, тоді як у менших масштабах (тобто в менших підрозділах) вони забезпечують більш дрібні деталі. Недоліком є те, що кінцевий вектор дескриптора стає більшим, таким чином, потрібно більше часу для вилучення та тренування за допомогою заданого класифікатора.

1.6 Огляд існуючих рішень проблеми розпізнавання білбордів

1.6.1 ADNet: A Deep Network for Detecting Adverts

Модель ADNet на архітектурі мережі VGG19 [23] була опублікована в 2018 році [24]. У цьому рішенні автоматично ідентифікуються кадри у відеопотоці, які мають наявну рекламу. Автори підкреслюють, що такий автоматичний процес значно допоможе під час етапу обробки постів, оскільки відео-редактору більше не потрібно буде вручну розбирати відеокадри. Система дозволяє виявляти рекламу у відео-послідовності, що значно прискорить процес пост-обробки та заощадить величезну кількість людино-годин.

ADNet використовує заздалегідь підготовлені ваги мережі VGG, натренованої на наборі даних ImageNet [25]. Розробники вимкнули навчання для перших 5 шарів попередньо підготовленої мережі VGG. Також вони видалили останні 3 вагові шари оригінальної моделі VGG19 і додали стек повнозв'язаних шарів (FC). Перший FC-шар має 1024 канали з функцією активації *relu*. Для того, щоб запобігти перенавчанню ADNet, автори додали *dropout*-шар зі швидкістю 0,5. Другий FC-шар також має 1024 канали з функцією активації *relu*. Третій і заключний FC-шар мають 2 канали з функцією активації *softmax*. Два значення, отримані в кінці ADNet, забезпечують ймовірність приналежності до класів білборда або "не білборда". Архітектура моделі ADNet зображена на рисунку 6.

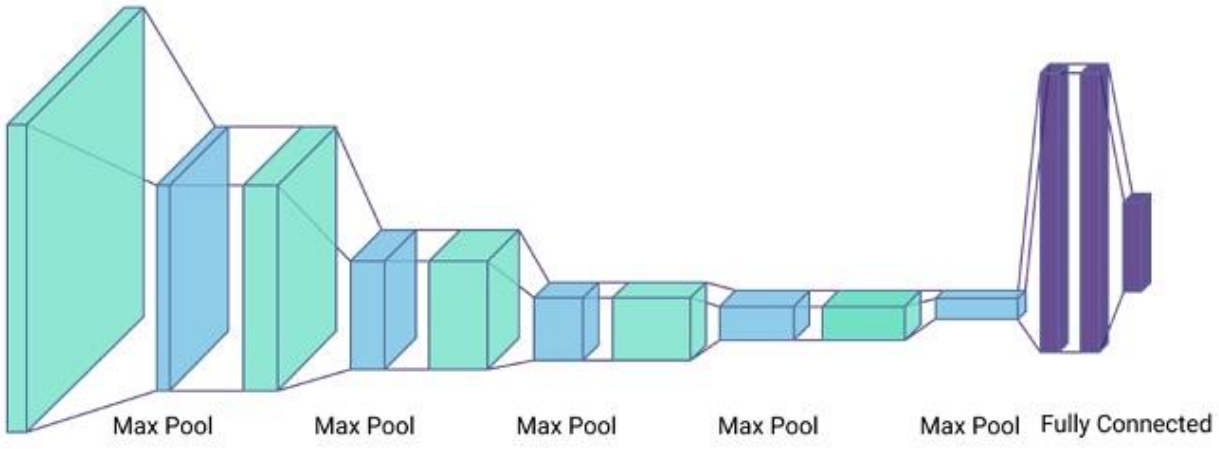

Рис. *6 Архітектура ADNet*

Автори тренували модель ADNet на складеному наборі даних, в якому містяться як позитивні, так і негативні приклади. Об'єкт рекламного щита повинен охоплювати значну частку зображення. Розробники задали поріг: білборд має охоплювати більше 10% загальної площі зображення. Зображення з цього складеного набору даних зібрані з двох джерел: Mapillary Vistas Dataset [26], Microsoft COCO (MS-COCO) Dataset [27]. Складений набір даних склав загалом 18945 зображень (табл. 2): 9141 зображень містять рекламні щити; а решта 9804 — жодного не містить.

Таблиця *2*

Розподіл наборів для навчання та тестування стосовно двох наборів даних

Set	Positive		Negative		Total
	Mapillary	MS-COCO	Mapillary	MS-COCO	
Training	6380	—	303	6617	13300
Testing	2761	—	106	2778	5645
Total	9141	—	409	9395	18945

На рисунку 7 автори навели приклади, коли ADNet правильно визначила наявність реклами на зображеннях.

Рис. *7 Позитивні приклади рекламних оголошень, правильно визначених ADNet*

Більшість зображень на рис. 8 містить предмети, схожі за формою на звичайний чотиристоронній рекламний щит. Задня частина вантажівки, підставка для сонячних батарей та дорожня вивіска мають аналогічну форму та зовнішній вигляд, як звичайний білборд.

Рис. *8 Приклади помилок класифікації*

У роботі автори розраховують класифікаційну точність за формулою 1.

$$\text{Точність} = \frac{TP + TN}{TP + TN + FP + FN}, \tag{1}$$

де *TP*, *TN*, *FP* і *FN* позначають справжні позитивні, справжні негативні, помилково позитивні та помилково негативні зразки двійкової класифікації відповідно.

Модель ADNet має конкурентоспроможну класифікаційну точність 0,94 згідно з підходами порівняльного оцінювання (табл. 3) [28].

Таблиця *3*

Оцінка ефективності моделі ADNet при виявленні рекламних оголошень у порівнянні з моделлю Inception-v3

Модель	**Точність**
Inception-v3 [28]	0.56
Proposed model ADNet	0.94

Крім того, має конкурентну швидкість обробки відеокадрів.

1.6.2 Система виявлення рекламних щитів та геотегування

Про цю систему було описано в статті “Система виявлення рекламних щитів та геотегів з індуктивним трансферним навчанням в глибокій згортковій нейронній мережі” журналу “TELKOMNIKA” [29].

Розробники виконали огляд попередніх програмних рішень, здатних виявити білборди: через складність навколишнього середовища виникли труднощі при виявленні рекламного щита у попередніх роботах. Техніка виявлення країв і кольорів може бути використана для їх виявлення, якщо навколишнє середовище не є складним, якщо припустити, що фон рівний, а об’єкт не перекритий. Крім того, попередні рішення виявляли білборд залежно від його країв. І коли одне з ребер перекрите, програма не могла виявити білборд. Також, підхід на виявлення кольорів працює, якщо об’єкт має певний колір, який контрастує з фоном. Щоб подолати цю проблему, необхідний підхід контрольованого машинного навчання необхідний для виявлення білборда у більш складних умовах, наприклад, дороги з багатьма перешкодами, такими як дерево, транспортний засіб, трос і стовп.

У даній роботі пропонується метод виявлення рекламного щита за допомогою глибокої ЗНМ. По-перше, заздалегідь підготовлена архітектура нейронної мережі від AlexNet [30] використовується для класифікації зображень. Для покращення продуктивності у виявленні білборда автори тренують мережу, використовуючи підхід трансферного навчання. Кілька інших типів необхідних даних, такі як категорії та назви білбордів, також відстежуються в системі.

Для процесу геотегування використовується вставка метаданих EXIF [31] для зберігання географічного розташування виявленого рекламного щита.

Автори запропонували спосіб виявлення та виконання геотегування для зображень білбордів за допомогою смартфона; спосіб можна описати в кілька етапів: процес попередньої підготовки зображень, який буде поєднуватися із введеним зображенням та перепідготовкою процес за допомогою

методу трансферного навчання. Після закінчення цього процесу та заповнення системи навченого DCNN, його можна використовувати для ідентифікації нового зображення, зробленого пристроєм Android у режимі реального часу. Таким чином, процес тонкої настройки буде класифікувати шість класів: табло, монітор, телебачення, кінотеатр та веб-сайт в один клас, названий рекламним щитом. Наступна фаза — функція вилучення, яка може виявити кадр зображення білборда за кадром в DCNN, а потім автоматично зберігати зображення до бази даних з мінімальною точністю; потім застосовується геотегування. У фазі пост-обробки додається більш важлива інформація про зображення білборда, та виконується процес перевірки.

Розробники використовують Inception-v3 [28] як заздалегідь підготовлену модель для класифікації зображень на 1,2 млн набору зображень та 1000 різних класів з візуального розпізнавання великих масштабів ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 [25]. Ця модель використовує DCNN архітектуру AlexNet [30] з 60 мільйонами параметрів і 650 000 нейронів, складається з п’яти згорткових шарів, трьох об’єднаних шарів і трьох повнозв’язаних шарів. Вихід моделі — 1000 *softmax*, що призводить до п’яти вершин прогнозу об’єкта. Мережа містить вісім шарів з вагами; перші п’ять є згортковими, а три інші — повнозв’язані.

У таблиці 4 показаний процес DCNN у кожному шарі. Перший згортковий шар фільтрує вхідне зображення 224 × 224 × 3 з 96 фільтрами розміром 11 × 11 × 3 з кроком у 4 пікселі. Вихід першого згорткового шару буде використовуватися як вхід до другого згорткового шару з 256 фільтрів розміром 5 × 5 × 48. Третій, четвертий та п’ятий згорткові шари з’єднані один з іншим без будь-якого об’єднання або нормалізації шарів. Третій згортковий шар має 384 фільтри розміром 3 × 3 × 256, з’єднані з виходами другого згорткового шару. Четвертий та п’ятий згортковий шари мають 384 фільтри розміром 3 × 3 × 192. Повнозв’язані в шарах по 4096 нейронів кожен.

Таблиця *4*

Параметри шарів DCNN

Шар	Розмір	Фільтр	Кількість нейронів
Input: (Resized Image)	224 × 224 × 3 × 1	—	150528
Layer 1: Convolution + Max Pool	55 × 55 × 48 × 2	96	290400
Layer 2: Convolution + Max Pool	27 × 27 × 128 × 2	256	186624
Layer 3: Convolution	13 × 13 × 192 × 2	384	64896
Layer 4: Convolution	13 × 13 × 192 × 2	384	64896
Layer 5: Convolution + Max Pool	13 × 13 × 128 × 2	256	43264
Layer 6: Fully-Connected	2048 × 2	—	4096
Layer 7: Fully-Connected	2048 × 2	—	4096
Layer 8: Softmax Output	1000 × 1	—	1000

Для вдосконалення функцій DCNN розробники вдосконалили мережу, додаючи більше зображень білбордів у попередньо підготовлену модель, використовуючи індуктивний підхід трансферного навчання. Перший процес полягає у введенні 300 додаткових зображень білбордів, знятих з Google-зображень, випадковим чином, і 84 зображення білбордів, знятих з Android-смартфона в режимі реального часу, в модель Inception-v3 [28] та виконанні процесу перенавчання DCNN. Наступний процес включає процес вузького місця, де всі зображення, що використовуються для тренінгу, будуть проаналізовані та обчислені. Значення вузьких місць містять змістовний та компактний підсумок зображень. Використовуючи трансферне навчання, можна тренувати набір даних швидше, ніж навчати модель з нуля. Далі автори почали використовувати 8000 навчальних кроків у DCNN, що призвело до 5 найвищих прогнозованих об’єктів з високою точністю.

Автори доналаштували DCNN на 1000 категорій, використовуючи трансферне навчання в категорії білбордів, таких як табло, екран, монітор, телебачення та кіно. Як кінцевий результат буде вибрано найвищий прогноз, пов’язаний з рекламним щитом. Є дві споріднені категорії, пов’язані з білбордом, — табло з точністю 72,6% та кіно з точністю 0,548% (рис. 9). Оскільки

точність табло має вищу точність, вона буде обрана як кінцева точність виявленого об’єкта.

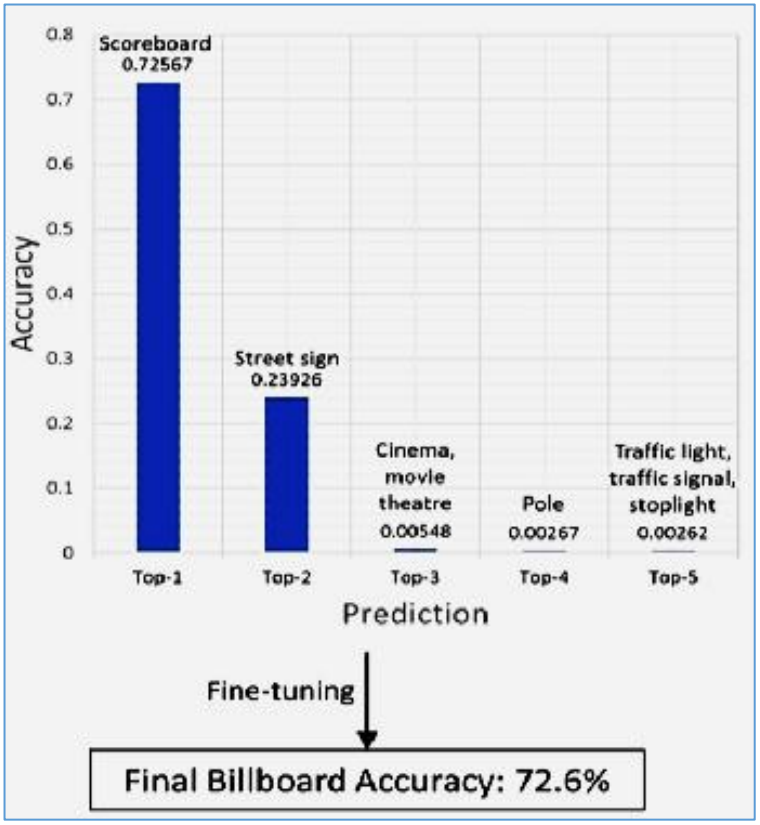

Рис. *9 Відношення точність-передбачення після доналаштування DCNN*

Після збереження зображення в базі даних додається додаткова інформація до даних білборда, таких як назва та категорія. Рекламний щит розділений на кілька категорій, наприклад продукти харчування та напої, телекомунікації, банк, страхування, транспорт, нерухомість, освіта, сигарети, подія, кампанія, просування, домашні товари, електронні продукти, засоби масової інформації. Ці категорії корисні для комерційних цілей.

1.6.3 AI-based advert creation system for next-generation publicity

Центр ADAPT розробив інноваційну систему, що працює на AI, у співпраці з дослідницьким центром Huawei Ireland, який дозволяє рекламним агенціям використовувати потужні засоби масової інформації, такі як відеоролики, для маркетингу та реклами, а користувачі можуть взаємодіяти за до-

помогою персоналізованого споживчого досвіду відповідно до їх індивідуальних переваг та уподобання [32, 33]. ADAPT розробив систему створення онлайн-реклами на базі AI для публічності нового покоління. Ця онлайн-система може автоматично визначати наявні рекламні щити у відео та безперешкодно замінювати рекламу, що міститься, на нову. Нещодавно створене доповнене відео надає споживачам персоналізовану практику. Ця система буде корисною для інтернет-маркетологів та розробників контенту для розробки відеоконтенту для цільової аудиторії.

Основа системи створення реклами базується на найсучасніших методах AI від глибокого навчання та обробки зображень. Використовується метод, заснований на глибокому навчанні, щоб локалізувати позицію реклами у кадрі зображення. Дане рішення використовує ЗНМ для вилучення ознак з зображень та змогти виявити рекламні щити на більш пізньому етапі. Програма локалізує чотири кути виявленого рекламного щита, використовуючи мережу уточнення на основі глибокого навчання. Після виявлення місця розміщення білборда автори використовують сучасні методи інтеграції та змішування, щоб розмістити нову рекламу на виявленому рекламному щиті.

Розробники вирішили дослідити розміщення рекламного щита, що має регулярну форму, на зображеннях вулиць — набір даних CASE [34]. Джерелами зображень взяли датасет Cityscapes [35], що містить зображення виду вулиць.

1.6.4 Застосунок для виявлення зображень нелегальних білбордів

Розробники даного застосунку [36] були натхнені нелегальними білбордами, які розміщені в китайських містах. Існують відповідні механізми обробки незаконних рекламних щитів та незаконного паркування в містах управління, але метод виявлення незаконних рекламних щитів на основі машинного зору ще вивчається.

Автори використовують вулиці в провінції Хубей як об’єкт дослідження та вивчають незаконні площі рекламних щитів, пропонують використову-

вати метод на основі машинного зору для автоматичного виявлення незаконних білбордів. Алгоритм виявлення використовує область багатокутних пропозицій для точного пошуку незаконних рекламних щитів у зображенні та позначає виявлені білборди. У порівнянні з безпосереднім виявленням рекламного щита на зображенні, інтерференція фонового фактору на цільовій області видаляється, а показник помилки виявляється ефективно зниженим.

Удосконалений алгоритм Faster R-CNN [37] використовується для стеження за паркуванням через відеокамеру. Для тренування мережі були взяті попередньо підготовлені дані з ImageNet [25]. Більшість методів глибокого навчання отримують набори даних шляхом їх розподілу за певними пропорціями для тренувальних наборів, наборів перевірки та тестових наборів. Автори 80% набору використовували як навчальний набір даних (training), а решта 20% — як набір даних підтвердження (validation).

Рис. *10 Моніторинг результатів обробки відео*

Результати роботи застосунку для розпізнавання нелегальної реклами видно на рисунку 10.

1.6.5 Cut and Paste: Generate Artificial Labels for Object Detection

У науковій статті 2017-го року [38] був запропонований метод збільшення даних, щоб підвищити точність виявлення об’єктів та одночасно підвищити швидкість виявлення. Цей метод, на відміну від традиційних методів, значною мірою може збільшити навчальну вибірку та певною мірою позбавити мережу від перенавчання.

Порівнюючи різні моделі виявлення в документах та аналізуючи параметри продуктивності, автори виявили, що для тих складних мереж їм зазвичай потрібно більше даних для підготовки, і слід частково змінювати пізніші шари детекторів, якщо використовуються ці моделі для виділення ознак. Для всієї моделі, коли даних набагато більше, ніж ті, які могли б представляти моделі ознак, це призведе до недостатнього навчання; а якщо даних занадто мало при порівняно складній моделі, відбудеться перенавчання.

Автори спочатку підготували фонові зображення (приклад на рисунку 11). Більшості моделей нейронних мереж важко виявити крихітні предмети в щільних скупченнях, потрібно дозволити моделям дізнаватися більше абстрактних особливостей.

Рис. *11 Фонові зображення*

Стиль середнього зображення на рисунку 11 може збільшити труднощі при виявленні. На материнській платі є багато крихітних деталей, та розташовані вони дуже близько.

Далі було виконане обрізання зображень з наборів VOC07 та VOC12 [39]. Вони використовували дані місцеположення в примітці для обрізання конкретних потенціальних об’єктів та мітки класів, щоб указати, до якого класу вони належать. Вставка є найважливішою частиною процесу. Спочатку фонові зображення були розбиті на п’ять частин: верхню ліву область, верхню праву область, нижню ліву область, нижню праву область та середню площу квадрата. Зображення на рисунку 12 були згенеровані цими методами. Відповідне накладення на лівому малюнку та поєднання об’єктів виконано бажано. Середня картинка імітує ілюстрації в книзі.

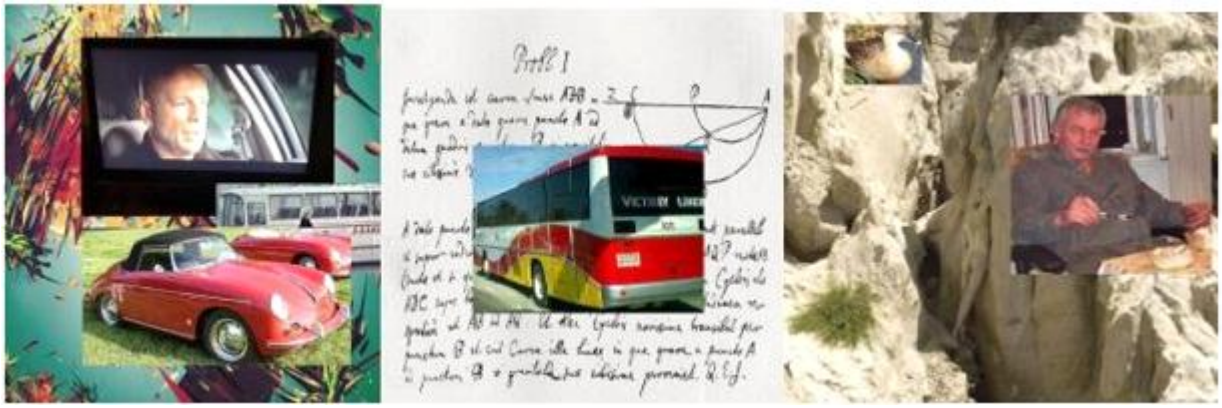

Рис. *12 Згенеровані зображення*

У процесі тренінгу розробники використовують шар батч-нормалізації перед функцією активації. Така поведінка збільшує швидкість тренування. Що ще важливіше, батч-нормалізація обробляє вхідні дані, щоб отримати кращий розподіл, що зменшить негативний вплив зображень. Результати ефективності виявлення об’єктів після аугментації представлені в таблиці 5.

Таблиця *5*

Результати ефективності виявлення шляхом збільшення даних

	IoU	Повнота	Точність	Виявлення помилок для вирізання та вставки зображень
оригінал	72.26%	86.54%	81.49%	47.43%
зараз	71.96%	86.63%	81.88%	23.35%

Спочатку автори застосували метод збільшення даних безпосередньо. Проблема полягає в тому, що розмір нових зображень та об’єктів не підходить. Модель використовує навчання з різними масштабами. Така операція призведе до аномального відношення між фоновим зображенням та об’єктами на ньому.

Тому був застосований метод k-середніх, щоб знайти кількість та аспекти об’єкта для кожного зображення. Потім скористалися збільшенням даних. Довелося вирішувати проблему обчислювального переповнення: розподіл даних досить схожий на оригінальний набір; пропозиції здаються неправильними.

Даний метод може значно збільшити дані та покращити продуктивність у “хаотичних” фонових зображеннях та абстрактних художніх творах, а також надати моделі чутливість до об’єктів у різних масштабах. Для того, щоб мати гарні показники, важливим є аналіз даних та статистика навчального набору, також слід враховувати способи та символи моделей виявлення. У процесі тренування автори виявили, що значення цільової функції важко зменшити на початку, хоча було використано перевірену вагу для ініціалізації. У процесі тренувань, кількість повторених батчей становить від 1 до 15000, навряд чи можна побачити ефект тренувань, тоді як можна побачити очевидне покращення без їх методу аугментації даних.

1.7 Результати огляду існуючих рішень

Більшість знайдених рішень була розроблена починаючи з 2017 року. Методи та архітектури описані в наукових статтях. Автори успішно застосували ці методи переважно для вирішення поширених завдань — часто це стосувалося реклами на фото та відео, зроблених у містах проживання.

Широковживані практики (наприклад, збільшення даних) для навчання мережі, помилки та порівняння найкращих варіантів архітектур нейронних мереж для розпізнавання об’єктів, заміни об’єктів на кадрах відео, описані

авторами, допомогли розробити ефективну комп’ютерну систему для вирішення поставленої проблеми.

РОЗДІЛ 2 ДОСЛІДЖЕННЯ ЗАСОБІВ ВИДІЛЕННЯ ОБ'ЄКТІВ ТА ЗАМІНИ ВМІСТУ

2.1 Потреба у методах розпізнавання об'єктів

Швидке збільшення кількості онлайн-відео надає маркетинговим та рекламним агентам широкі можливості охопити свою аудиторію. Однією з найбільш широко використовуваних стратегій є розміщення товару або вбудований маркетинг, де нова реклама безперешкодно інтегрується в існуючу рекламу на відео. Такі стратегії передбачають точну локалізацію позиції реклами у кадрі зображення, або вручну на етапі редагування відео, або за допомогою машинного навчання. Ці методи машинного навчання та глибокі нейронні мережі потребують величезної кількості даних для навчання.

Оскільки метою роботи є система, здатна розпізнати й виділити білборди на відео, потрібно дослідити, а потім застосувати алгоритми розпізнавання об'єктів.

2.2 Згорткові нейронні мережі для задачі виявлення об'єктів

Згорткова нейронна мережа (ЗНМ, CNN) — алгоритм глибокого навчання, який може приймати вхідне зображення, присвоювати важливість (навчальні ваги та ухили) різним аспектам/об'єктам на зображенні та мати можливість відрізняти один від іншого. Вперше була представлена у 1998 році Яном Лекуном [40]. Необхідна алгоритму попередня обробка є менш затратною у порівнянні з іншими алгоритмами класифікації. Хоча в примітивних методах фільтри виробляються вручну, при достатньому навчанні ЗНМ має можливість вивчати ці фільтри.

Архітектура згорткової мережі є аналогічною схемі зв'язку нейронів у людському мозку і була натхнена організацією візуальної кори. Окремі нейрони реагують на подразники лише в обмеженій області зорового поля, відо-

мої як рецептивне поле. Колекція таких полів перекривається, щоб охопити всю зорову зону [41].

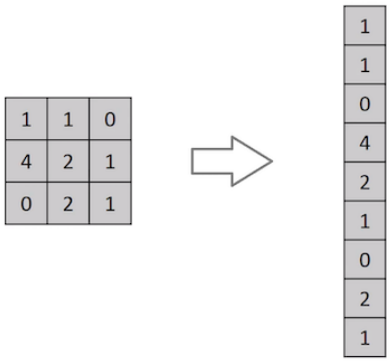

Рис. *13 Випрямлення зображення-матриці 3 × 3 у вектор 9 × 1*

Роль згорткової мережі полягає в тому, щоб зменшити зображення у форму, яку легше обробляти (рис. 13), не втрачаючи особливостей, які є критичними для отримання гарного прогнозу. Це важливо, коли планується розробляти архітектуру, яка не тільки добре засвоює ознаки, але також масштабується до масивних наборів даних.

Структура згорткових нейронних мереж включає в себе чергування згортаючих і субдискретизуючих шарів, і наявність повнозв’язаних шарів на виході. Загальна структура ЗНМ представлена на рисунку 14.

Для виділення об’єктів на кадрах можна використати ЗНМ з одним нейроном у вхідному шарі, який буде приймати значення на інтервалі [–1;+1] — наявність або відсутність його на кадрі відео.

Згортковий шар — це перший шар, який вилучає ознаки з вхідного зображення. Шари виявляються дуже ефективними, а їх складання у глибоких моделях дозволяє шарам, близьким до входу, вивчати ознаки низького рівня (наприклад, лінії) та глибші шари в моделі, щоб дізнатися більш абстрактні ознаки, наприклад, форми чи конкретні об’єкти.

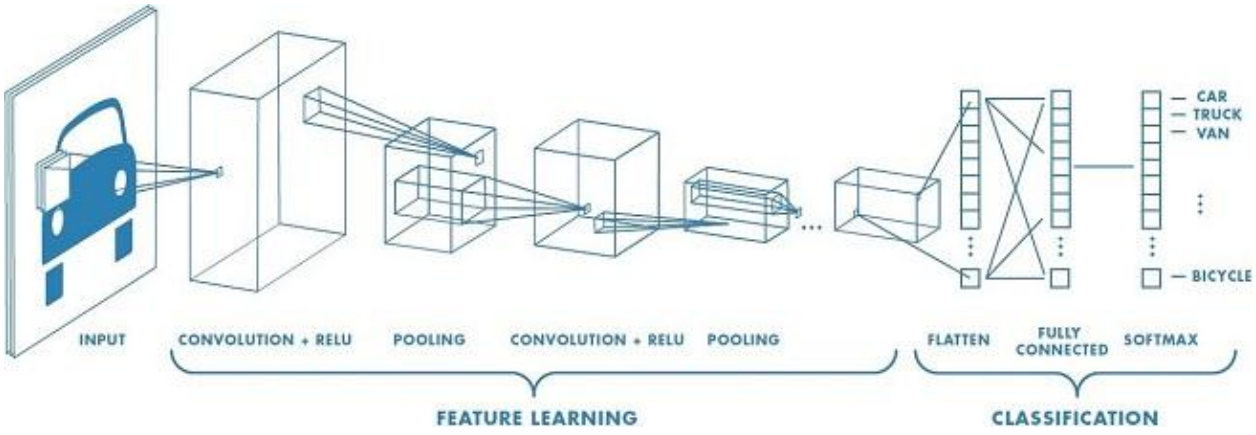

Рис. *14 Структура згорткової нейронної мережі*

Обмеженням виводу карт ознак згорткових шарів є те, що вони фіксують точне положення ознак на вході. Це означає, що невеликі рухи в положенні на вхідному зображенні призведуть до отримання іншої карти ознак. Це може статися після повторного обрізання, обертання, зміщення та інших незначних змін вхідного зображення. Поширений підхід для вирішення цієї проблеми при обробці сигналу — даунсемплінг, коли створюється версія вхідного сигналу нижчої роздільної здатності, яка все ще містить великі або важливі структурні елементи, без тонких деталей, які можуть бути не настільки корисними для задачі. Даунсемплінг може бути досягнутий за допомогою згорткових шарів, змінивши крок згортки по зображенню. Більш надійний та поширений підхід — використовувати підвибірковий (pooling) шар.

Якщо розглянути матрицю зображення 5 × 5, значення пікселів яких 0 або 1 та матрицю фільтра 3 × 3 як на рисунку 15. Згортка матриці зображень 5 × 5 множиться на матрицю фільтра 3 × 3, в результаті отримують карту ознак.

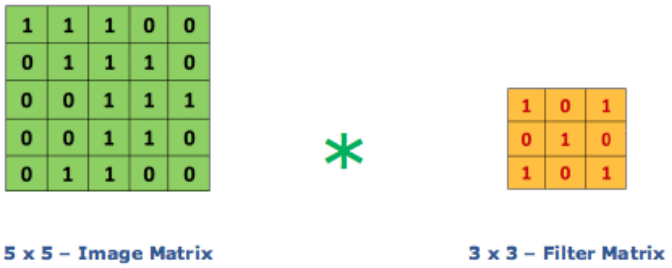

Рис. *15 Матриця зображення множиться на матрицю фільтра*

Складання зображення з різними фільтрами може виконувати такі операції, як виявлення країв, розмиття та різкість, застосовуючи фільтри. Інтернет-ресурс [42] демонструє зображення згортки після застосування фільтрів за вибором користувача. На рисунку 16 зображений приклад роботи фільтру різкості (sharpen), коли збільшується контраст між яскравими та темними регіонами, щоб вилучити ознаки.

Рис. *16 Застосування деяких фільтрів*

Крок — кількість зрушень пікселів над вхідною матрицею. Коли він дорівнює 1, фільтри переміщуються на 1 піксель; якщо крок 2, вони переміщуються на 2 тощо.

Підвибірковий шар зменшує кількість параметрів, коли зображення занадто великі. Підвибірка буває різних типів:

– *Max Pooling* бере найбільший елемент з випрямленої карти ознак. Приклад на рисунку 17.
– *Average Pooling* включає в себе обчислення середнього значення для кожного виправлення карти ознак.
– *Sum Pooling* підсумовує усі елементи на карті ознак.

Після того, як нелінійна функція активації [43] (наприклад, ReLU) застосовується до карт ознак, що виводяться згортковим шаром, шари в моделі можуть виглядати наступним чином:

1. Вхідне зображення.
2. Згортковий шар.
3. Нелінійна функція активації.
4. Підвибірковий шар.

Додавання підвибіркового шару після згорткового шару є загальною схемою, яка використовується для впорядкування шарів у ЗНМ, що може бути повтореним один чи більше разів у даній моделі.

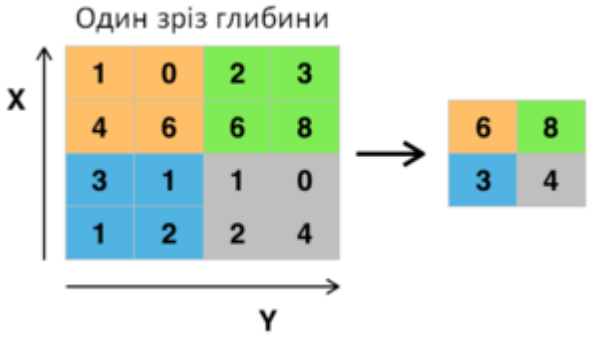

Рис. *17 Max Pooling із фільтром 2×2 та кроком 2*

Підвибірковий шар працює над кожною картою ознак окремо для створення нового набору з тієї ж кількості об’єднаних карт ознак. Це означає, що такий шар завжди зменшить розмір кожної карти ознак у 2 рази; наприклад, кожен розмір зменшується вдвічі, зменшуючи кількість пікселів або значень у кожній карті ознак на розмір однієї чверті. Результатом використання підвибіркового шару та створення зразкових або об’єднаних карт ознак є узагальнена версія ознак, виявлених на вході. Вони корисні, оскільки невеликі зміни в розташуванні ознак на вході, виявленому згортковим шаром, призведуть до об’єднання карти ознак із тим самим розташуванням. Таку можливість, додану підвибіркою, називають незмінністю моделі до локальної трансляції.

Нейрони у повнозв’язаному шарі мають повне з’єднання з усіма активаціями попереднього шару, як це спостерігається у звичайних нейронних

мережах. Їхні активації можна обчислити множенням матриць з урахуванням нейрону зміщення.

2.2.1 Модель SqueezeDet

SqueezeDet — це згорткова нейронна мережа для виявлення об’єктів [44], в якій використовуються згорткові шари не тільки для вилучення карт ознак, але і як вихідний шар для обчислення обмежуючих рамок та ймовірностей класів. У цій моделі конвеєр виявлення містить лише один прямий прохід [45] нейронної мережі, отже, він надзвичайно швидкий. Мережа — повністю згорткова, що зазвичай призводить до невеликого розміру моделі та кращої енергоефективності. Цілями розробки SqueezeDet були економічність, можливість розгортання на багатьох системах та роботи детектора на вбудованих процесорах, які споживають набагато менше енергії, ніж потужні графічні процесори, що використовуються для порівняльного аналізу в типових експериментах з комп’ютерним зором.

Розробники моделі фокусувались на задачі автономного водіння: транспортні засоби можуть мати різні комбінації сенсорів, але вирішальним залишається виявлення об’єктів. Детектори зображення дешеві в порівнянні з іншими, такими як LIDAR [46]. Дані зображень (включаючи відео) набагато рясніші, ніж, наприклад, хмарні точки LIDAR, та їх набагато легше збирати та коментувати.

Для створення конвеєра виявлення SqueezeDet розробників надихнула модель YOLO [47]: спочатку використовуються накопичувальні фільтри згортки, щоб вилучити карту ознак високого розміру з низькою роздільною здатністю для вхідного зображення. Конвеєр виявлення мережі SqueezeDet зображений на рисунку 18.

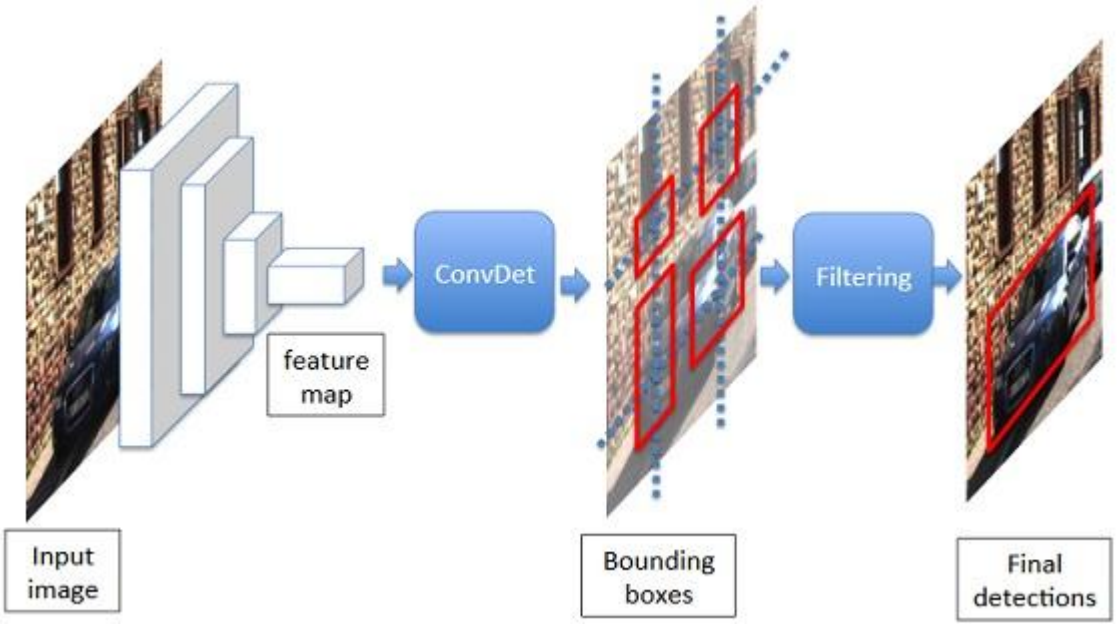

Рис. *18 Конвеєр виявлення мережі SqueezeDet*

Потім використовується ConvDet, згортковий шар, щоб взяти карту ознак у якості вхідних даних та обчислити велику кількість обмежувальних рамок, та передбачити їх класи. Шар ConvDet обчислює обмежувальні рамки з центром навколо $W \times H$ рівномірно розподілених центрів сітки. Кожному обмежувальному полю присвоєно 1 бал довіри (confidence score) та умовні ймовірності класу C. Далі зберігаються обмежувальні рамки з найбільшою ймовірністю — використовується немаксимальне придушення (non-maximum suppression) [48], щоб відфільтрувати їх та отримати остаточне виявлення об'єктів.

Кожна обмежувальна рамка асоціюється зі значеннями $C+1$, де C — кількість класів, які потрібно розрізнити, а додаткова 1 необхідна для оцінки достовірності, яка вказує, наскільки ймовірно обмежувальна рамка містить об'єкт. Як і в YOLO [47], оцінка довіри визначена як $Pr(\text{Object}) * IOU_{truth}^{pred}$. Висока оцінка достовірності передбачає велику ймовірність того, що об'єкт, який цікавить, існує і що перекриття між передбачуваною та істинною обмежувальними рамками є великим. Інші скаляри C представляють умовний розподіл ймовірності класу, враховуючи те, що об'єкт існує в обмежувальній рамці. Більш формально розробники позначають умовні ймовірності як

Pr (class$_c$ | Object), $c \in [1, C]$. Вони підписують назву класу (label) з найбільшою умовною ймовірністю до цієї обмежувальної рамки, і використовують метрику для оцінки достовірності передбачення обмежувальної рамки (формула 2).

$$\min_c Pr(\text{class}_c|\text{Object}) * Pr(\text{Object}) * \text{IOU}_{truth}^{pred} \quad (2)$$

ConvDet є, по суті, згортковим рівнем, навченим виводити координати обмежувальної рамки та ймовірності класів. Він працює як пересувне вікно, що рухається через кожне просторове положення на карті ознак. У кожному положенні обчислюється значення $K \times (4 + 1 + C)$, які кодують передбачення обмежувального вікна. K — кількість полів прив'язок (anchors) із заздалегідь обраними фігурами. Кожне положення на карті ознак відповідає центру сітки на вихідному зображенні, тому кожне поле прив'язки може бути описане чотирма скалярами як $(\widehat{x_i}, \widehat{y_j}, \widehat{w_k}, \widehat{k_k})$, $i \in [1, W]$, j $\in [1, H]$, $k \in [1, K]$; де $\widehat{x_i}, \widehat{y_j}$ — це просторові координати опорного центру сітки (i, j), $\widehat{w_k}, \widehat{k_k}$ — це ширина та висота k-го поля прив'язки. Для кожного поля прив'язки (i, j, k) обчислюються чотири відносні координати (δx_{ijk}, δy_{ijk}, δw_{ijk}, δh_{ijk}), щоб перетворити його у передбачуване обмежувальне поле, як показано на рисунку 19. Керуючись [49], перетворення описується в формулі 3.

$$\begin{aligned} x_i^p &= \widehat{x_i} + \widehat{w_k}\delta x_{ijk}, \\ y_j^p &= \widehat{y_j} + \widehat{h_k}\delta y_{ijk}, \\ w_k^p &= \widehat{w_k}\exp(\delta w_{ijk}), \\ h_k^p &= \widehat{h_k}\exp(\delta h_{ijk}), \end{aligned} \quad (3)$$

де $x_i^p, y_j^p, w_k^p, h_k^p$ — передбачувані координати обмежувальної рамки.

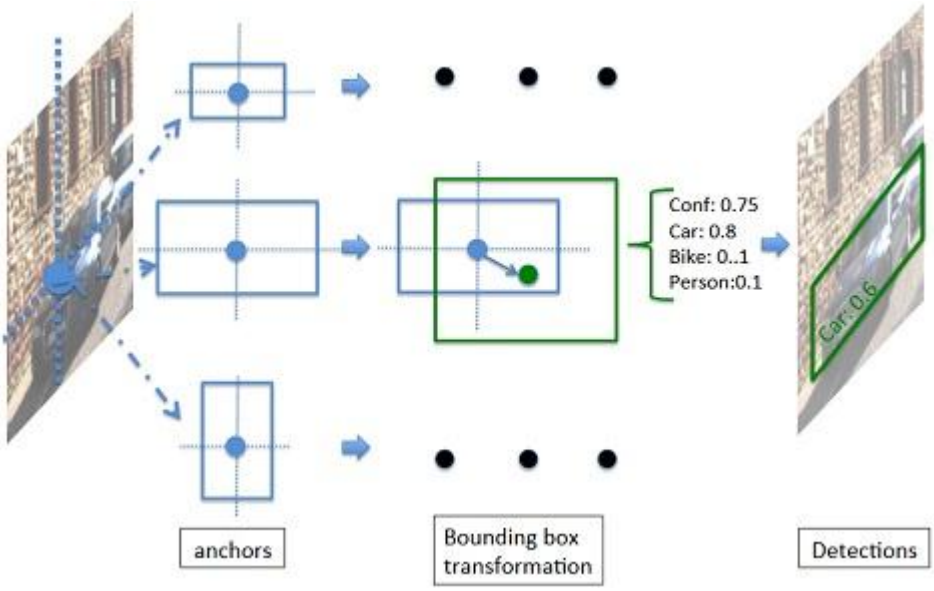

Рис. *19 Перетворення обмежувальної рамки*

Кожне поле прив'язок асоціюється з оцінкою достовірності та ймовірностями класу для прогнозування категорії об'єкта в обмежувальній рамці. ConvDet подібний до останнього шару RPN у Faster R-CNN [37]; основна відмінність: RPN розглядається як "слабкий" детектор, який відповідає лише за виявлення того, чи існує об'єкт, та генерування пропозицій обмежувальної рамки для об'єкта. Класифікація передається повноз'єднаним шарам, які розглядаються як "сильний" класифікатор. Насправді згорткові шари досить "сильні", щоб одночасно виявляти, локалізувати та класифікувати об'єкти.

На відміну від Faster R-CNN, яка використовує (4-ступінчасту) змінну навчальну стратегію для навчання RPN та детекторної мережі, мережу SqueezeDet можна навчити наскрізно (end-to-end), подібно до YOLO. Щоб навчити шар ConvDet виявляти, локалізувати та класифікувати, розробниками визначено функцію втрат (формула 4).

$$\frac{\lambda_{bbox}}{N_{obj}}\sum_{i=1}^{W}\sum_{j=1}^{H}\sum_{k=1}^{K} I_{ijk}[\left(\delta x_{ijk}-\delta x_{ijk}^{G}\right)^2+\left(\delta y_{ijk}-\delta y_{ijk}^{G}\right)^2+ \\ +\left(\delta w_{ijk}-\delta w_{ijk}^{G}\right)^2+\left(\delta h_{ijk}-\delta h_{ijk}^{G}\right)^2]+ \tag{4}$$

$$+\sum_{i=1}^{W}\sum_{j=1}^{H}\sum_{k=1}^{K}\left[\frac{\lambda_{conf}^{+}}{N_{obj}}I_{ijk}\left(\gamma_{ijk}-\gamma_{ijk}^{G}\right)^{2}+\frac{\lambda_{conf}^{-}}{WHK-N_{obj}}\bar{I}_{ijk}\gamma_{ijk}^{2}\right]+$$

$$+\frac{1}{N_{obj}}\sum_{i=1}^{W}\sum_{j=1}^{H}\sum_{k=1}^{K}\sum_{c=1}^{C}I_{ijk}l_{c}^{G}\log(p_{c})$$

Перша частина функції втрат — це обмежувальна регресія. Змінні $\left(\delta x_{ijk},\delta y_{ijk},\delta w_{ijk},\delta h_{ijk}\right)$ відповідають відносним координатам *k*-го поля прив'язки, розташованого в центрі сітки (i,j); вони є виходами шару ConvDet.

Істинна обмежувальна рамка δ_{ijk}^{G}, або значення $\left(\delta x_{ijk}^{G},\delta y_{ijk}^{G},\delta w_{ijk}^{G},\delta h_{ijk}^{G}\right)$ знаходяться за формулою 5.

$$\begin{aligned}\delta x_{ijk}^{G}&=(x^{G}-\widehat{x_{i}})/\widehat{w_{k}},\\ \delta y_{ijk}^{G}&=\left(y^{G}-\widehat{y_{j}}\right)/\widehat{h_{k}},\\ \delta w_{ijk}^{G}&=\log(w^{G}/\widehat{w_{k}}),\\ \delta h_{ijk}^{G}&=\log\left(h^{G}/\widehat{h_{k}}\right).\end{aligned}\qquad(5)$$

Друга частина функції втрат — це регресія оцінки достовірності; величина γ_{ijk} є виходом з шару ConvDet, що представляє прогнозовану оцінку достовірності для *k*-го поля прив'язки у позиції-(i,j). Величина γ_{ijk}^{G} отримується шляхом обчислення IOU (перетин над об'єднанням) передбачуваної з дійсною обмежувальними рамками. Для полів прив'язок, які не "відповідають" за виявлення, їхні показники впевненості нехтуються виразом $\bar{I}_{ijk}\gamma_{ijk}^{2}$, де $\bar{I}_{ijk}=1-I_{ijk}$. Зазвичай трапляється дуже багато полів прив'язки, не присвоєних жодному об'єкту; щоб збалансувати їх вплив, застосовуються λ_{conf}^{+} та λ_{conf}^{-} для регулювання ваги цих двох компонентів втрат.

Остання частина функції втрат — це просто крос-ентропійні втрати для класифікації. Значення $l_{c}^{G}\in\{0,1\}$ є істинною назвою класу (label), $p_{c}\in$

$[0,1]$, $c \in [1, C]$ — розподіл ймовірностей, передбачений нейронною мережею. Розробники застосували *softmax* для нормалізації виходу ConvDet, щоб переконатися, що p_c знаходиться в діапазоні між [0,1]. Гіпер-параметри у формулі вибрані емпірично: $\lambda_{bbox} = 5, \lambda_{conf}^{+} = 75, \lambda_{conf}^{-} = 100.$

Архітектурою цієї ЗНМ є SqueezeNet [50], яка досягає точності зображення ImageNet [25] на рівні AlexNet із розміром моделі < 5 МБ, який можна додатково стиснути до 0,5 МБ. Після "зміцнення" моделі SqueezeNet додатковими шарами з ConvDet, загальний розмір моделі все ще менше 8 МБ. Робоча швидкість SqueezeDet може досягати 57,2 FPS при роздільній здатності вхідного зображення 1242×375 — розробники мережі розділили 7381 зображень випадковим чином навпіл — навчальний набір та набір перевірки.

Для навчання встановили початкову швидкість навчання 0,01, коефіцієнт затухання швидкості навчання 0,5, розмір кроку затухання 10000; розмір батча — 20. Для збільшення даних та уникнення перенавчання мережі застосовано методи збільшення даних — випадкове обрізання та перевертання. Модель тренували для виявлення трьох класів об'єктів: автомобіль, велосипедист і пішохід. Порівняння точності й інших характеристик з іншими моделями наведено в таблиці 6.

Таблиця *6*

Підсумок точності виявлення, розміру моделі та швидкості роботи

Метод	Автомобіль (mAP)	Велосипедист (mAP)	Пішохід (mAP)	Усі (mAP)	Розмір моделі (МБ)	Швидкість (FPS)
FRCN + VGG16	86.0	—	—	—	485	1.7
FRCN + AlexNet	82.6	—	—	—	240	2.9
SqueezeDet	82.9	76.8	70.4	76.7	**7.9**	**57.2**
SqueezeDet+	85.5	**82.0**	**73.7**	**80.4**	26.8	32.1
VGG16-Det	**86.9**	79.6	70.7	79.1	57.4	16.6
ResNet50-Det	86.7	80.0	61.5	76.1	35.1	22.5

Одна з навчених моделей SqueezeDet досягла найкращої середньої точності на всіх трьох рівнях складності виявлення велосипедистів у завданні виявлення об’єктів KITTI [51]. Для виявлення об’єктів підвищення роздільної здатності зображення часто є ефективним підходом, щоб підвищити точність виявлення. Але більші зображення призводять до більших активацій, більшої кількості FLOP, більшого часу навчання тощо.

2.2.2 Модель SSD MobileNet

Головною метою цього проекту було вивчення та дослідження внутрішньої роботи “Single Shot MultiBox Detector” (SSD) [52] для виявлення об’єктів.

У середині 2010-х років двоступеневі підходи для задачі локалізації, такі як сімейство методів R-CNN [53], були громіздкими в плані обчислень і повільними, але домінували в полі з точки зору точності (як правило, вона вимірюється середньою точністю — mAP) на стандартних датасетах для виявлення об’єктів: MS COCO [27] та Pascal VOC2007 та 12 [39]. Це призвело до створення добре відомої мережі YOLO [47], яка була набагато швидшою та ефективнішою, ніж попередні методи. Проте таке підвищення швидкості було досяжне лише ціною жертви точності. Саме тут у фокус потрапив фреймворк SSD — перша глибока нейронна архітектура, яка не використовувала пропозиції регіонів [54] (region proposals) і мала наскрізний підхід до виявлення об’єктів на зображенні за допомогою єдиної глибокої нейронної мережі, яка була настільки ж точною, як і методи. Більше того, із вилученням етапів пропозиції регіонів метод SSD також зміг забезпечити швидший час виконання (59 FPS з mAP 74.3% на датасеті VOC2007 test, проти Faster R-CNN [37] 7 FPS з mAP 73.2% або YOLO 45 FPS з mAP 63.4%). Однак, мабуть, найважливішою складовою фреймворку SSD є той факт, що він використовує малі фільтри згортки для прогнозування класів об’єктів та розміщення рамки для різних співвідношень сторін і робить це на кількох картах ознак на пізніх стадіях роботи мережі, що дозволяє агрегувати виявлення у різних масшта-

бах. Отже, якщо використовувати декілька шарів, які мають відносно меншу вхідну роздільну здатність, і використовувати їх для генерації прогнозів у різних масштабах, можна забезпечити високу точність результатів з вищими швидкостями виявлення.

Найбільшим недоліком фреймворку SSD є той факт, що його продуктивність прямо пропорційна розмірам об’єктів, що означає, що він не надто добре справляється з категоріями об’єктів малих розмірів порівняно з іншими підходами, такими як сімейство R-CNN. Це пов’язано з тим, що дрібні об’єкти можуть не містити корисної інформації у верхніх шарах мережі, які могли би продуктивно використовуватися для виявлення. Тому методи збільшення даних зазвичай використовуються для випадкового обрізання та зміни розміру певних частин зображення, щоб допомогти мережі легше ідентифікувати та вивчити особливості для малих категорій об’єктів. Більше того, саме тому збільшення вхідної роздільної здатності зображень з 300 × 300 до 512 × 512 забезпечує кращі результати як для наборів даних MS COCO, так і для Pascal VOC. Розробники даної мережі проводили експерименти, які підкреслили різницю в продуктивності мереж, навчених із використанням збільшення даних та без них.

Друге питання — це час роботи мережі (inference). Видно, що 80% часу, необхідного для прямого проходження через мережу, контролюється базовою (backbone) мережею. Базова мережа є, по суті, усіченим високоякісним класифікатором зображень, який використовується для вилучення ознак, що будуть застосовані для прогнозування на пізніх етапах роботи мережі. Автори використовують відому мережу VGG-16 [23], яка попередньо навчена наборі даних ILSVRC CLS-LOC [25]. Тобто це можливість ще більше підвищити швидкість виявлення за допомогою швидшої базової мережі. Також розробники експериментували з використанням мережі MobileNetv1 [55], попередньо навченої на тому самому наборі даних, що і VGG16, як заміну вихідній базовій мережі для вивчення компромісу та співвідношення — швидкість-точність. Крім того, була перевірена тривалість роботи мережі на ши-

рокому діапазоні апаратного забезпечення з різними обчислювальними можливостями для надання практичної користі використання даної мережі іншими розробниками.

Мережева архітектура MobileNet — це особливий клас ЗНМ, побудованих з використанням відокремлених по глибині згорток; як наслідок вони є більш легкими з точки зору кількості параметрів та складності обчислень. Авторами-розробниками цієї мережевої архітектури було введено два додаткових гіперпараметри — множник ширини та роздільної здатності, які можуть контролювати кількість вхідних/вихідних каналів шарів згортки та роздільну здатність вхідних даних (тобто висоту, ширину) відповідно. Ці параметри можуть бути використані для безпосереднього впливу на затримку та точність мережі залежно від кінцевих вимог користувача.

Інтеграція MobileNet у фреймворк SSD була одним з основних аспектів роботи розробників. Однак варто зазначити, що поєднання високоефективної базової мережі (MobileNet) з надзвичайно ефективною структурою SSD є останнім часом гарячою темою дослідження, в основному завдяки вирішенню практичних обмежень роботи потужних нейронних мереж на пристрої низького класу, такі як смартфони/ноутбуки, щоб ще більше розширити незліченну кількість можливостей щодо застосунків у реальному часі.

В процесі дослідження розробники зауважили, що база VGG була вузьким місцем під час навчання та роботи мережі. Очевидно, була необхідність замінити VGG мережею, яка зменшила б час обчислень, але зберегла точність. MobileNet складається з окремих згорткових шарів DepthWise, які швидші за обчисленнями, ніж стандартні згорткові шари. Причина полягає просто в меншій кількості операцій множення та додавання через розділення каналів у глибинному шарі та їх подальше лінійне поєднання з використанням згортки 1×1. Емпірично, зменшення обчислювальних зусиль значною мірою не впливає на продуктивність мережі, саме тому розробники хотіли б використовувати її як магістраль в рамках SSD. При використанні в SSD були прибрані останні три шари: *Avg Pool*/s1, *FC*/s1 та *Softmax*/s1, оскільки не

на меті класифікувати за допомогою MobileNet. Взагалі кажучи, роль базової мережі в SSD полягає у перетворенні пікселів із вхідного зображення в ознаки, які описують вміст зображення, і передачі їх разом з іншими шарами SSD.

Замінюючи VGG16 на MobileNetv1, розробники додали шари 12 і 14 MobileNet до SSD. Вони підключили шар, що відокремлюється по глибині, за допомогою фільтра 1 × 1 × 512 × 512 (шар 12), до карти ознак SSD глибини 512. Також з'єднали останній шар потенційної згортки з шаром SSD за допомогою карти ознак 1024. В таблиці 7 представлені порівняльні результати роботи моделей на датасеті Pascal VOC20017.

Таблиця *7*

Результати тесту роботи моделей на датасеті Pascal VOC20017

Метод	mAP	FPS	Розмір батча	Рамки	Вхідна роздільна здатність
Faster R-CNN (VGG16)	73.2	7	1	~6000	~1000 × 600
Fast YOLO	52.7	155	1	98	448 × 448
YOLO (VGG16)	66.4	21	1	98	448 × 448
SSD300	74.3	46	1	8732	300 × 300
SSD512	76.8	19	1	24564	512 × 512
SSD300	74.3	59	8	8732	300 × 300
SSD512	76.8	22	8	24564	512 × 512

2.2.3 Модель YOLOv4

Модель нейронної мережі [56] була опублікована у квітні 2020 року. В статті автор обґрунтовує судження, що найточніші сучасні детектори, які працюють на базі ЗНМ, не працюють в режимі реального часу та вимагають великої кількості графічних процесорів для навчання великого міні-батча. Наприклад, таким чином здійснюється пошук вільних місць для паркування з застосуванням міських відеокамер. Розробники вирішують дану проблему шляхом створення ЗНМ, яка працює в режимі реального часу на звичайному графічному процесорі, та для навчання якої необхідний лише один звичайний графічний процесор. Метою було проектування детектора об'єктів високої

робочої швидкості у виробничих системах та оптимізація для паралельних обчислень, а не теоретичний показник низького обсягу обчислень (BFLOP).

Сучасна модель ЗНМ для виявлення об’єктів, як правило, складається з двох частин: бази, яка попередньо навчена на датасеті ImageNet [25], та голови, яка використовується для прогнозування класів та обмежувальних рамок об’єктів. Розроблені в останні роки детектори об’єктів часто вставляють деякі шари між базою та головою, які зазвичай використовуються для збору карт ознак з різних етапів. Ми можемо назвати це шийкою детектора об’єктів. Зазвичай шия складається з декількох шляхів знизу вгору та декількох шляхів зверху вниз. У плані архітектури метою було знайти оптимальний баланс між роздільною здатністю вхідної мережі, кількістю згорткових шарів, кількістю параметрів та кількістю даних вихідного шару. Наступна мета — вибір додаткових блоків для збільшення сприйнятливого поля та найкращого методу агрегування параметрів з різних рівнів бази для різних рівнів детектора.

Оптимальна для класифікації еталонна модель ЗНМ є не завжди оптимальною для виявлення об’єктів. На відміну від класифікатора, детектор вимагає:

– більшого вхідного розміру мережі (роздільна здатність) для виявлення безлічі малогабаритних об’єктів;
– більшої кількості шарів для більш сприйнятливого поля для покриття збільшеного розміру вхідної мережі;
– більшої кількості параметрів для більшої здатності моделі виявляти різну кількість об’єктів різного розміру на одному зображенні.

Розробники додали блок SPP над CSPDarknet53 [57], оскільки він значно збільшує поле сприйняття, виділяє найважливіші контекстні особливості і майже не спричиняє зменшення швидкості роботи мережі. Як метод агрегування параметрів з різних рівнів бази для різних рівнів детектора використовується PANet [58]. Для того, щоб спроектована модель виявлення об’єктів

була більш придатною для навчання на одному графічному процесорі, автори зробили додаткові вдосконалення:

- новий метод збільшення даних мозаїкою, коли чотири зображення поєднуються в одному, як на прикладі з білбордами на рисунку 20; також метод самостійного навчання (SAT) [59];
- обираються оптимальні гіперпараметри після застосування генетичних алгоритмів;
- модифіковані деякі захоплюючі методи, щоб зробити дизайн придатним для ефективного навчання та виявлення об’єктів — модифікований SAM [60], модифікований PAN [58].

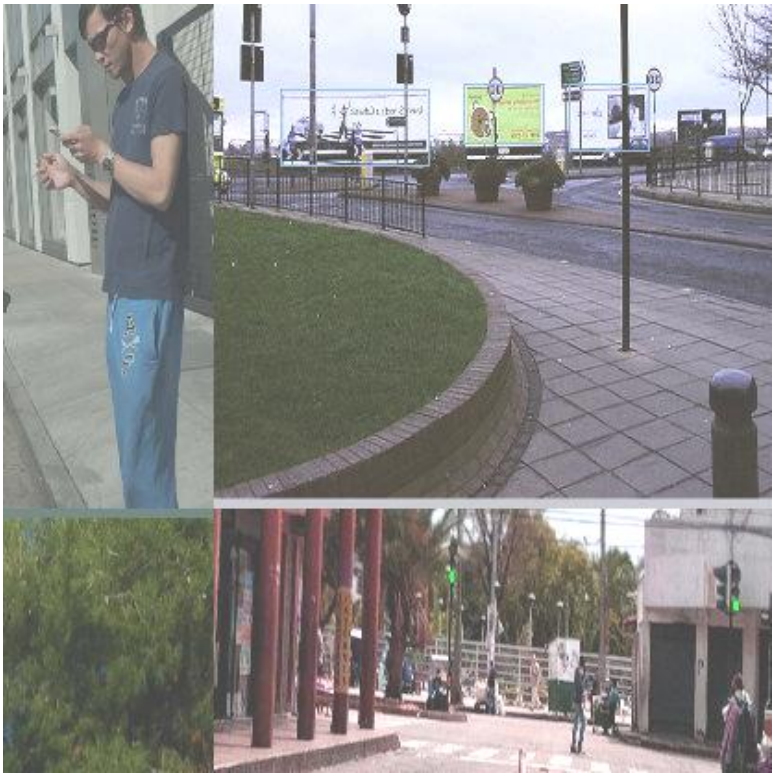

Рис. *20 Збільшення даних мозаїкою на прикладі фото з білбордами*

Автори перевірили вплив різних методик вдосконалення тренувань на точність класифікатора на наборі даних ImageNet (ILSVRC 2012 val) [25], а потім на точність детектора на наборі даних MS COCO (test-dev 2017) [27]. У кінці свого дослідження було проаналізовано результати, отримані на моделях, навчених з різними розмірами міні-батчів. Було виявлено, що після до-

давання стратегій навчання BoF [61] та BoS [62], розмір міні-батчів майже не впливає на ефективність детектора об’єктів. Тобто після введення BoF і BoS більше не потрібно використовувати дорогі графічні процесори для навчання.

У таблиці 8 наведено результати порівняння частоти кадрів використання графічного процесора Maxwell, і це може бути GTX Titan X (Maxwell) або графічний процесор Tesla M40.

Таблиця *8*

Порівняння швидкості та точності різних детекторів об’єктів на наборі даних MS COCO (test-dev 2017)

Метод	База	Розмір	FPS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
YOLOv4:									
YOLOv4	CSPDarknet-53	416	38 (M)	41.2%	62.8%	44.3%	20.4%	44.4%	56.0%
YOLOv4	CSPDarknet-53	512	31 (M)	**43.0%**	**64.9%**	**46.5%**	**24.3%**	**46.1%**	**55.2%**
YOLOv4	CSPDarknet-53	608	23 (M)	43.5%	65.7%	47.3%	26.7%	46.7%	53.3%
YOLOv3									
YOLOv3	Darknet-53	320	45 (M)	28.2%	51.5%	29.7%	11.9%	30.6%	43.4%
YOLOv3	Darknet-53	416	35 (M)	31.0%	55.3%	32.3%	15.2%	33.2%	42.8%
YOLOv3	Darknet-53	608	20 (M)	33.0%	57.9%	34.4%	18.3%	35.4%	41.9%
YOLOv3-SPP	Darknet-53	608	20 (M)	36.2%	60.6%	38.2%	20.6%	37.4%	46.1%
SSD: Single shot multibox detector [52]									
SSD	VGG-16	300	43 (M)	25.1%	43.1%	25.8%	6.6%	25.9%	41.4%
SSD	VGG-16	512	22 (M)	28.8%	48.5%	30.3%	10.9%	31.8%	43.5%

Описану модель для виявлення об’єктів можна навчити та використовувати на звичайному графічному процесорі з 8-16 ГБ VRAM, що робить можливим її широке використання. Була перевірена велика кількість властивостей, серед яких автори застосували тільки ті, що підвищили точність як класифікатора, так і детектора.

2.3 Способи навчання мережі та підготовка навчальної вибірки

Наявність великого набору даних має вирішальне значення для роботи моделі глибокого навчання. Питання розміру навчальних даних також відоме в літературі як складність вибірки.

Розробник може покращити продуктивність моделі шляхом збільшення існуючих даних [63]. Фреймворки глибокого навчання зазвичай мають вбудовані утиліти для збільшення даних, але вони можуть виявитися неефективними або не мати необхідної функціональності.

Для збільшення навчальної вибірки зазвичай використовують:

- довільний випадковий поворот зображення між 0° та 360°;
- нахил зображення між –10 та 10 пікселями;
- випадкову зміну масштабу з коефіцієнтом від 1/1.6 до 1.6;
- відображення зображення по вертикалі та горизонталі;
- випадкове розтягування зображення з коефіцієнтом від 1/1.3 до 1.3.

За результатами навчання нейронної мережі для класифікації фотознімків з видами планктону [64] для конкурсу National Data Science Bowl 2015 року [65]: пружні викривлення не пришвидшили навчання, але ненабагато зменшили перенавчання. Розподіл Гауса для вибірки параметрів трансформації зображень замість нормального розподілу не підвищив продуктивність.

Існує декілька широко використовуваних емпіричних способів визначення розміру навчальних даних відповідно до типу моделі, яку використовують:

1. *Регресійний аналіз*: Відповідно до правила 10 з 10, потрібно 10 випадків для прогнозування [66]. Один із захоплюючих, нещодавно розроблених варіантів для бінарної логістичної регресії представлений у [67]: автори статті оцінюють розмір навчальних даних, беручи до уваги кількість змінних прогнозів, загальний розмір вибірки та відношення кількості позитивних зразків до загального обсягу вибірки.

2. *Комп'ютерний зір*: для класифікації зображень за допомогою глибокого навчання правило великого розміру становить 1000 зображень на клас, де це число може значно зменшитися, якщо використовувати попередньо підготовлені моделі [68].

Основна ідея попередньої підготовки полягає в тому, що ми спочатку тренуємо нейронну мережу (або інший алгоритм машинного навчання) на дешевому та великому наборі даних у пов'язаному домені. Незважаючи на те, що це безпосередньо не вирішить початкову проблему, це дасть нейронній мережі приблизне уявлення про те, як виглядає проблема передбачення. Далі, на другому кроці, параметри нейронної мережі додатково оптимізуються на значно менший і дорогий набір даних проблеми, яку потрібно вирішити.

Під час тонкої настройки кількість класів може змінюватися: розробники часто попередньо тренують нейронну мережу на наборі даних, такому як ImageNet [25], з 1000 класами, а потім точно налаштовують її на свою конкретну проблему, яка, ймовірно, має іншу кількість класів. Це означає, що останній шар потрібно повторно ініціалізувати.

2.4 Способи знаходження контурів об'єкта

Коли успішно знайдені координати об'єкта (білборда) на зображенні, далі необхідно якомога точніше визначити його контур — координати точок замкнутої кривої. Існує три загальних підходи до подання границь об'єкта:

– апроксимація кривих;
– простежування контурів;
– зв'язування точок перепадів.

У роботі [69] було розроблено сегментацію об'єктів зверху вниз та знизу вгору з використанням контурів фігури за допомогою двоступеневої процедури. Спочатку об'єкт ідентифікували за допомогою функції контуру на основі ребра, а потім контур об'єкта отримували за допомогою процедури

оптимізації обмежень на основі результатів з раніше визначених контурів. Початкове виявлення об’єкта надає інформацію про конкретну категорію об’єкта для завершення контуру. Основним обчислювальним підходом є пошук країв зображення як неперервності інтенсивності. Однак контур об’єкта — це більше, ніж просто різниця інтенсивності. Контури об’єктів зазвичай вилучаються за допомогою детектора краю, але результатом роботи таких операторів рідко є зовнішній контур об’єкта. Це пояснюється тим, що ребро в контурі є не лише різницею в інтенсивності, а й загалом будь-якою формою розриву між об’єктом та фоном. Ця проблема ускладняється маскуванням та оклюзією. Складність випливає з проблеми розуміння образу за наявності складних фонів. Авторами розглядається підхід у роботі [70], який використовує двоступеневу процедуру для сегментації зображення після попереднього виявлення. Цей підхід полягає у отриманні попередньої ідентифікації об’єкта спочатку з подальшою сегментацією на основі цих попередніх знань (рис. 21). Замість патчів зображень, які зазвичай використовуються для виявлення об’єктів, контури країв використовуються як початкові ознаки для попередньої ідентифікації та використовуються як керівництво для сегментації.

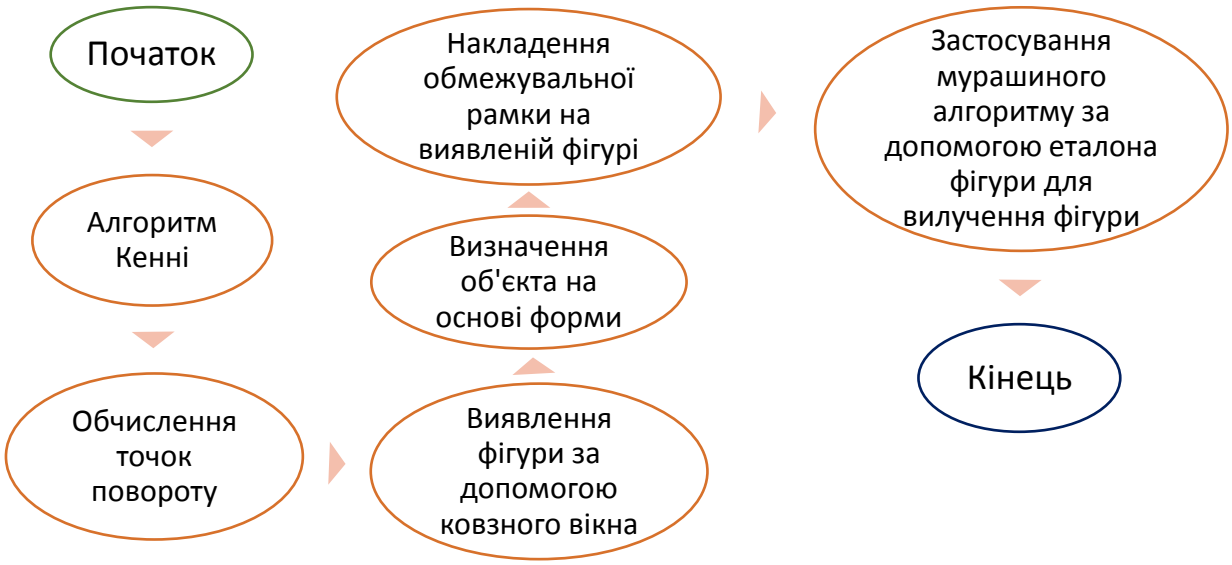

Рис. *21 Блок-схема загального підходу*

Фаза розпізнавання об’єктів працює шляхом виявлення країв за допомогою стандартної техніки виявлення країв на основі інтенсивності (алгоритм Кенні [71]). З виявлених країв потім відкинули коротші краї. Більш довгі краї, які зберігаються, аналізуються на точки повороту; ці точки використовуються як зображення фрагментів контуру. Фрагменти контуру отримували за допомогою стандартного алгоритму вилучення краю. У процесі розпізнавання масштаб і положення враховувались за допомогою підходу ковзного вікна змінної величини. Вікна різних розмірів ковзали по цільовому зображенню сцени в кожному розташуванні вікна; було записано кількість відповідних точок повороту. Коли об’єкт був попередньо розпізнаний у обмежувальному вікні, вже був отриманий набір країв інтенсивності у вікні. Однак ці краї не є остаточними, оскільки деякі з них можуть бути неправильно витлумачені через помилки процесу, перекриття об’єктів або плутанину у фоновому режимі тощо. Далі задачу було розглянуто як проблему задоволення обмежень, але самі обмеження є невизначеними; та було застосовано мурашиний алгоритм (ACO) [72] для дослідження цієї проблеми. Для алгоритму ACO використано 20-40 ітерацій за епоху, 1500-3000 епох. В кінці кожної епохи найкращому мурасі дозволяється оновлювати феромони на своєму шляху.

Рис. *22 Результати з використанням одного зразка для розпізнавання та завершення контуру*

На рисунку 22 показані зразки побудованого шляху мурахи, створеного з однієї зразкової форми. Це показує, що алгоритм може генерувати гнучкі рішення відповідно до обмежень та різних форм.

Є деякі слабкі сторони підходу ACO. Наприклад, для більших зображень, простір пошуку занадто великий для ACO, і він стає чутливим до початкових умов. Оцінка якості зовнішнього контуру має вирішальне значення при визначенні його успіху.

Підхід загалом працює наступним чином: спочатку від конкретного до сценового зображення робиться спроба визначити, який об’єкт на зображенні; після виявлення об’єкта використовується його форма для отримання контуру. Наступний етап включає етап зворотного зв’язку, де була визначена якість контуру: якщо він відповідає обмеженням, тоді завдання виконано, інакше буде використано інший приклад фігури.

2.5 Способи накладання зображення на розпізнаний об’єкт

Для задачі заміни розпізнаних об’єктів (на прикладі білбордів) на відео (кадрах) необхідно застосувати алгоритм, який приведе звичайне прямокутне зображення до розмірів і форми об’єкта; а потім накласти деформоване зображення на цей об’єкт. Хоча більшість рекламних щитів розташовують фасадом до водія (на фото вони будуть прямокутної форми), все одно на фото/кадрах матимуть місце перспективні перетворення. Наприклад, якщо камера знаходиться близько під ним.

Метод центрального проектування застосовується при побудові перспективи (рис. 23 [73]), так як отримані цим способом зображення є дуже наочними. Перспективне зображення лежить в основі фотографії.

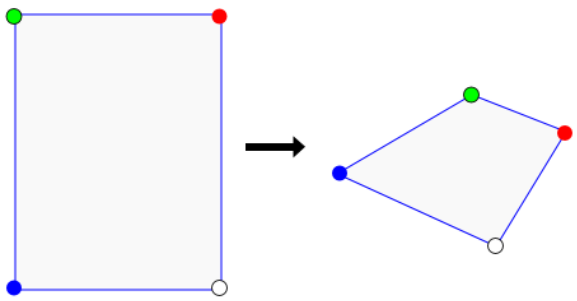

Рис. *23 Центральна проекція прямокутника*

Для того, щоб побудувати центральну проекцію, необхідно піддати вихідну фігуру перспективному перетворенню, і після цього спроектувати її на деяку площину. Перспективне перетворення має місце, коли не дорівнює нулю будь-який з перших трьох елементів четвертого стовпця матриці 4 × 4. При перспективному перетворенні: паралельні прямі сходяться; розмір об’єкта зменшується зі збільшенням відстані до центру проекції; відбувається неоднорідне спотворення ліній об’єкта, залежне від орієнтації та відстані від об’єкта до центру проекції.

Одноточкове перспективне перетворення задається рівністю, вказаною на формулі 6:

$$[\,x\ \ y\ \ z\ \ 1\,] \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & r \\ 0 & 0 & 0 & 1 \end{bmatrix} = [\,x\ \ y\ \ z\ \ rz+1\,], \tag{6}$$

де $h = rz + 1 \neq 1$, *(x; y; z)* — координати точки. Звичайні координати отримують діленням на h (формула 7).

$$[\,x^*\ y^*\ z^*\ 1\,] = \left[\frac{x}{rz+1}\quad \frac{y}{rz+1}\quad \frac{z}{rz+1}\quad 1\right] \tag{7}$$

Координати точки *P**, спроектованої на площину *z* = 0 (рис. 24), можна отримати, використовуючи подібність трикутників (формула 8).

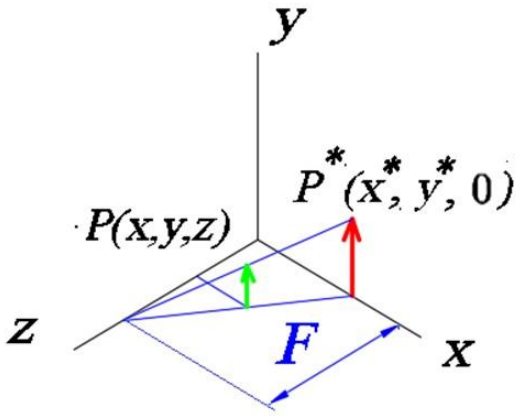

Рис. *24 Проекція точки на площину xOy*

$$\frac{x^*}{F} = \frac{x}{F - z} \Rightarrow x^* = \frac{x}{1 - \frac{z}{F}},$$
$$\frac{y^*}{F} = \frac{y}{F - z} \Rightarrow y^* = \frac{y}{1 - \frac{z}{F}} \quad (8)$$

Далі можна отримати вираз для r, прирівнюючи знаменники з рівнянь для визначення координат проекцій точки (формула 9).

$$rz + 1 = 1 - \frac{z}{F} \Rightarrow r = -\frac{1}{F} \quad (9)$$

При $F \longrightarrow \infty, r \rightarrow 0$ буде ортогональна проекція. Також, на точки, що лежать у площині проекції, перспективне перетворення не діє, тобто якщо площина проекції проходить через об’єкт, то ця частина об’єкта зображується з правильним розміром і формою, а всі інші частини об’єкта спотворюються.

2.6 Огляд популярних фреймворків глибинного навчання

У сучасному світі все більше організацій звертаються до машинного навчання та штучного інтелекту (AI), щоб покращити свої бізнес-процеси та випереджати конкуренцію. Зростання машинного навчання та AI дало змогу організаціям пропонувати розумні рішення та передбачувані персоналізації

своїм клієнтам. Однак не всі організації мають право впроваджувати машинне навчання та інтелектуальний інтелект для своїх процесів через різні причини.

Без певної структури займатися машинним навчанням геть непродуктивно. При роботі у бізнес-середовищі неможливо створити все з нуля. Потрібен фреймворк, який допоможе реалізувати бачення на проблему. Фреймворки для глибинного навчання пропонують будівельні блоки для проектування, навчання та перевірки глибоких нейронних мереж через інтерфейс програмування високого рівня.

TensorFlow. TensorFlow (TF) — це наскрізний відкритий фреймворк для машинного навчання, має всеосяжну, гнучку екосистему інструментів, бібліотек та ресурсів спільноти, що дозволяє дослідникам просунути найсучасніші технології машинного навчання, а розробникам — легко створювати та розгортати застосунки на базі машинного навчання [74].

В той час як еталонна реалізація працює на одиничних пристроях, TF може працювати на декількох ЦП та ГП (включно з додатковими розширеннями CUDA для обчислень загального призначення на графічних процесорах). Фреймворк доступний для 64-розрядних Linux, macOS, Windows, та для мобільних обчислювальних платформ, включно з Android та iOS. TF надає API для різних мов програмування, основною з яких є Python, що має найкращу документацію та підтримку, а також отримує найшвидші оновлення. Також підтримуються API для C++, Haskell, Java, Go.

TF забезпечує кращий спосіб візуалізації даних завдяки своєму графічному підходу. Це також дозволяє легко налагоджувати вузли за допомогою TensorBoard, що зменшує зусилля для відвідування цілого коду та ефективно вирішує нейронну мережу.

Серед недоліків можна виділити часті оновлення (кожні 2-3 місяці), що збільшує накладні витрати, щоб користувач міг їх встановити та пов’язати з існуючою системою. Також TF надає омоніми, які мають схожі імена, але рі-

зні реалізації, що робить заплутаним запам’ятовування та використання, наприклад: *tf.nn.conv2d, tf.nn.convolution, tf.layers.conv2d, tf.layers.Conv2d* мають різні значення, і це часто робить несумісним із його зручністю використання [75].

Keras. Keras — відкрита нейромережева бібліотека, написана мовою Python, здатна працювати поверх TensorFlow та Theano. Спроектовану для уможливлення швидких експериментів з мережами глибинного навчання, її зосереджено на тому, щоби вона була мінімальною, модульною та розширюваною. Keras дотримується найкращих практик зменшення когнітивного навантаження: пропонує послідовні та прості API, мінімізує кількість дій користувача, необхідних для загальних випадків використання, а також забезпечує чіткі та діючі повідомлення про помилки. Він також має велику документацію та посібники для розробників [76].

Моделі Keras приймають три типи входів:

– Масиви NumPy [77], як і Scikit-Learn [78] та багато інших бібліотек на базі Python. Гарний варіант, якщо дані поміщаються в пам’ять.
– Об’єкти набору даних TensorFlow. Це високопродуктивний варіант, який більше підходить для наборів даних, які не поміщаються в пам’яті та передаються з диска або з розподіленої файлової системи.
– Генератори Python, які видають пакети даних (наприклад, власні підкласи класу *keras.utils.Sequence*).

Keras пропонує кілька моделей глибокого навчання з їх попередньо навченими вагами. Розробники можуть використовувати ці моделі безпосередньо для прогнозування або вилучення ознак. Keras дозволяє тренувати модель на одному графічному процесорі або використовувати кілька графічних процесорів. Він забезпечує вбудовану підтримку паралельності даних та може обробляти дуже великий обсяг даних.

Серед недоліків бібліотеки можна виділити те, що розробники іноді отримують помилки низького рівня. Ці помилки виникають, у разі бажання

виконати деякі операції, для яких Keras не був призначений. Інструменти попередньої обробки даних Keras не настільки задовольняють у порівнянні з іншими пакетами, такими як Scikit-Learn. Не так добре будувати деякі базові алгоритми машинного навчання, такі як кластеризація та аналіз основних компонентів.

ML.Net. Бібліотека ML.NET дає простий і зручний механізм використання алгоритмів машинного навчання в .NET-застосунках, не вдаючись до сторонніх сервісів. На даний час доступна версія 1.5.1, і це вже стабільна та потужна бібліотека, що містить набір різних алгоритмів для вирішення завдань машинного навчання [79].

Розробники можуть навчити модель машинного навчання або повторно використати існуючу модель та запустити її в будь-якому середовищі в автономному режимі. Це означає, що їм не потрібно мати досвід роботи в Data Science, щоб використовувати фреймворк. Підтримка формату моделі глибокого навчання з відкритим кодом Open Neural Network Exchange (ONNX) була представлена зі збірки версії 0.3. Інтеграція TensorFlow включена з версії 0.5.

У [80] продемонстровано, що фреймворк здатний навчати модель для аналізу настроїв, використовуючи великі набори даних, одночасно досягаючи високої точності. Його результати показали 95% точність набору даних огляду Amazon розміром 9 ГБ.

ML.NET також забезпечує прив’язку до Python (NimbusML [81]), який на сьогодні є експериментальним рішенням. Це додатковий сценарій, який може бути корисним для вчених, що працюють на Python та співпрацюють із розробниками .NET за допомогою тих самих моделей ML.NET.

Серед недоліків можна виділити відсутність вбудованої підтримки моделей для задач виявлення та локалізації об’єктів; не підтримується навчання подібних моделей.

2.7 Висновки з розділу 2

1. Виходячи з аналізу існуючих методів та алгоритмів для виявлення об’єктів на зображенні, необхідно застосувати ЗНМ в даній роботі. Для задачі виявлення об’єктів (білбордів) вибрано модель мережі YOLOv4 з фреймворком Darknet, оскільки він має можливість навчання на відеоадапатері NVIDIA.

2. Для знаходження контурів виявлених об’єктів (білбордів) вибрано бібліотеку функцій та алгоритмів комп’ютерного зору OpenCV.

3. Для навчання мережі потрібно зібрати зображення білбордів з різних ракурсів у різний час доби та погодні умови; також поповнити навчальну вибірку зображеннями білбордів, які мають незначні пошкодження (не більше 30%), або ~30% їх контенту прикриті деревами, дорожніми знаками або іншими об’єктами. Після цього варто зробити поповнення датасету шляхом афінних перетворень.

РОЗДІЛ 3 ПРОЕКТ ПРОГРАМНОЇ СИСТЕМИ ВИДІЛЕННЯ ОБ'ЄКТІВ ТА ЗАМІНИ ЇХ ВМІСТУ

3.1 Архітектура системи

Комп'ютерна система для виділення об'єктів (білбордів) у відеопотоці складається з трьох компонентів:

1. Виявлення білбордів.
2. Знаходження контурів білбордів.
3. Трансформація та накладення зображень.

Схема роботи програми представлена на рисунку 25.

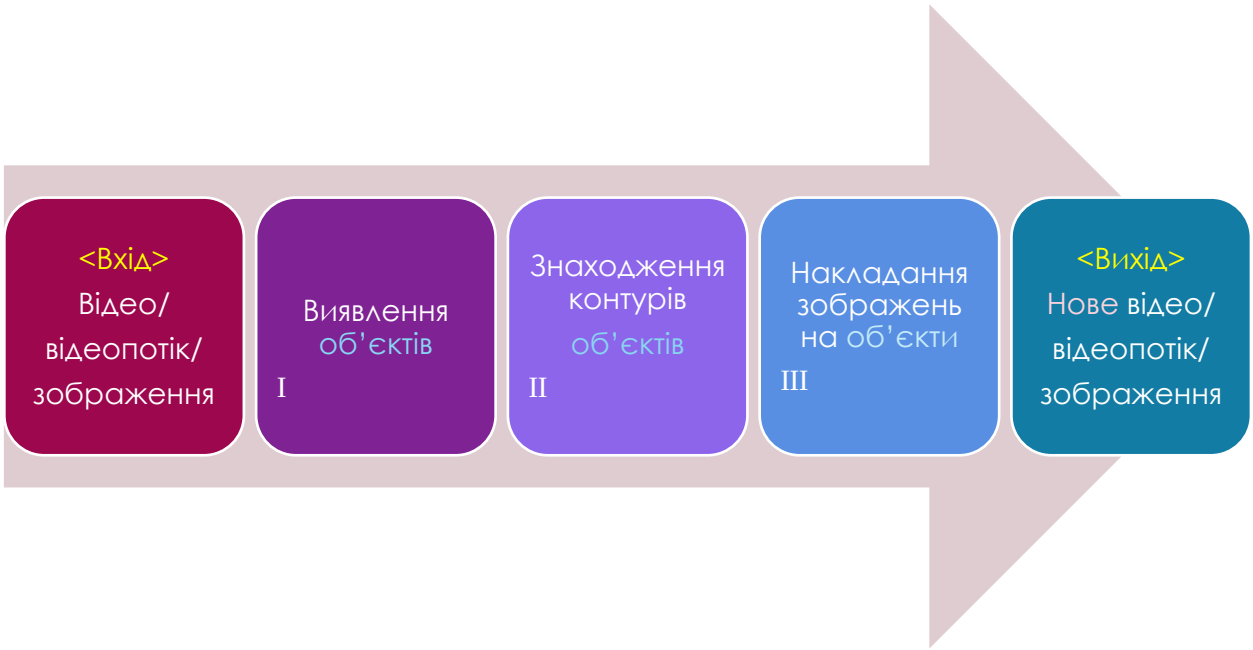

Рис. *25 Загальна схема роботи системи*

3.2 Конфігурація моделі нейронної мережі YOLOv4

Конфігурація для моделі мережі, яка завантажується та виконується фреймворком Darknet [57], визначена у текстовому файлі з розширенням *cfg* (рис. 26).

```
[net]
# Testing
#batch=1
#subdivisions=1
# Training
batch=64
subdivisions=64
width=416
height=416
channels=3
momentum=0.949
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.001
burn_in=1000
max_batches = 6000
policy=steps
steps=4800,5400
scales=.1,.1

#cutmix=1
mosaic=1

#:104x104 54:52x52 85:26x26 104:13x13 for 416

[convolutional]
batch_normalize=1
filters=32
size=3
stride=1
pad=1
activation=mish
```

Рис. *26 Файл конфігурації мережі YOLOv4 yolo-obj.cfg*

Для цієї системи виділення об’єктів у файлі *yolo-obj.cfg* були визначені наступні параметри:

batch=64 — кількість зразків (зображень), які будуть попередньо оброблені в одній партії;

subdivisions=64 — кількість *mini_batches* в одному батчі, розмір *mini_batch = batch / subdivisions*. GPU обробляє зразки *mini_batch* одночасно, і ваги будуть оновлені для зразків батчів (1 ітерація обробляє батч зображень);

width=416 — розмір мережі (ширина); кожне зображення буде змінено до розміру мережі під час навчання та виявлення об’єктів;

height=416 — розмір мережі (висота); кожне зображення буде змінено до розміру мережі під час навчання та виявлення об’єктів;

channels=3 — розмір мережі (канали); кожне зображення буде перетворено на цю кількість каналів під час навчання та виявлення об’єктів;

saturation=1.5 — випадково змінює насиченість зображень під час навчання;

exposure=1.5 — випадково змінює яскравість під час тренування;

hue=.1 — випадково змінює відтінок (колір) під час тренування (колірна модель HSV);

jitter=0.3 — випадково змінює розмір зображення та його співвідношення сторін з (1 – 2 × *jitter*) до (1 + 2 × *jitter*);

random=1 — випадково змінює розмір мережі після кожних 10 батчів (ітерацій) з /1.4 до ×1.4 із збереженням початкового співвідношення сторін розміру мережі;

momentum=0.949 — накопичення руху, наскільки історія впливає на подальшу зміну ваг;

decay=0.0005 — слабше оновлення ваг для типових ознак, це усуває дисбаланс у наборі даних;

learning_rate=0.001 — початковий рівень навчання (гіперпараметр). При оцінці на повному наборі даних і коли рівень навчання досить низький, це гарантує невід’ємний прогрес у функції втрат;

max_batches=6000 — навчання буде оброблено для такої кількості ітерацій (батчів).

Перший згортковий шар (секція *[convolutional]*, рис. 27):

batch_normalize=1 — буде використана нормалізація батчами (за замовчуванням значення 0);

filters=64 — кількість фільтрів (1 за замовчуванням);

size=3 — розмір фільтра (1 за замовчуванням);

stride=2 — крок (крок зміщення) вікна фільтра (1 за замовчуванням);

activation=mish — функція активації.

Останній шар моделі (секція *[yolo]*):

mask=6,7,8 — індекси полів прив’язок (*anchors*), які використовуються в цьому шарі;

anchors=12,16, 19,36, 40,28, 36,75, 76,55, 72,146, 142,110, 192,243, 459,401 — початкові розміри обмежувальних рамок, які будуть коригуватися;

num=9 — загальна кількість полів прив’язок;

classes=1 — кількість класів об’єктів, які потрібно виявити;

jitter=.3;

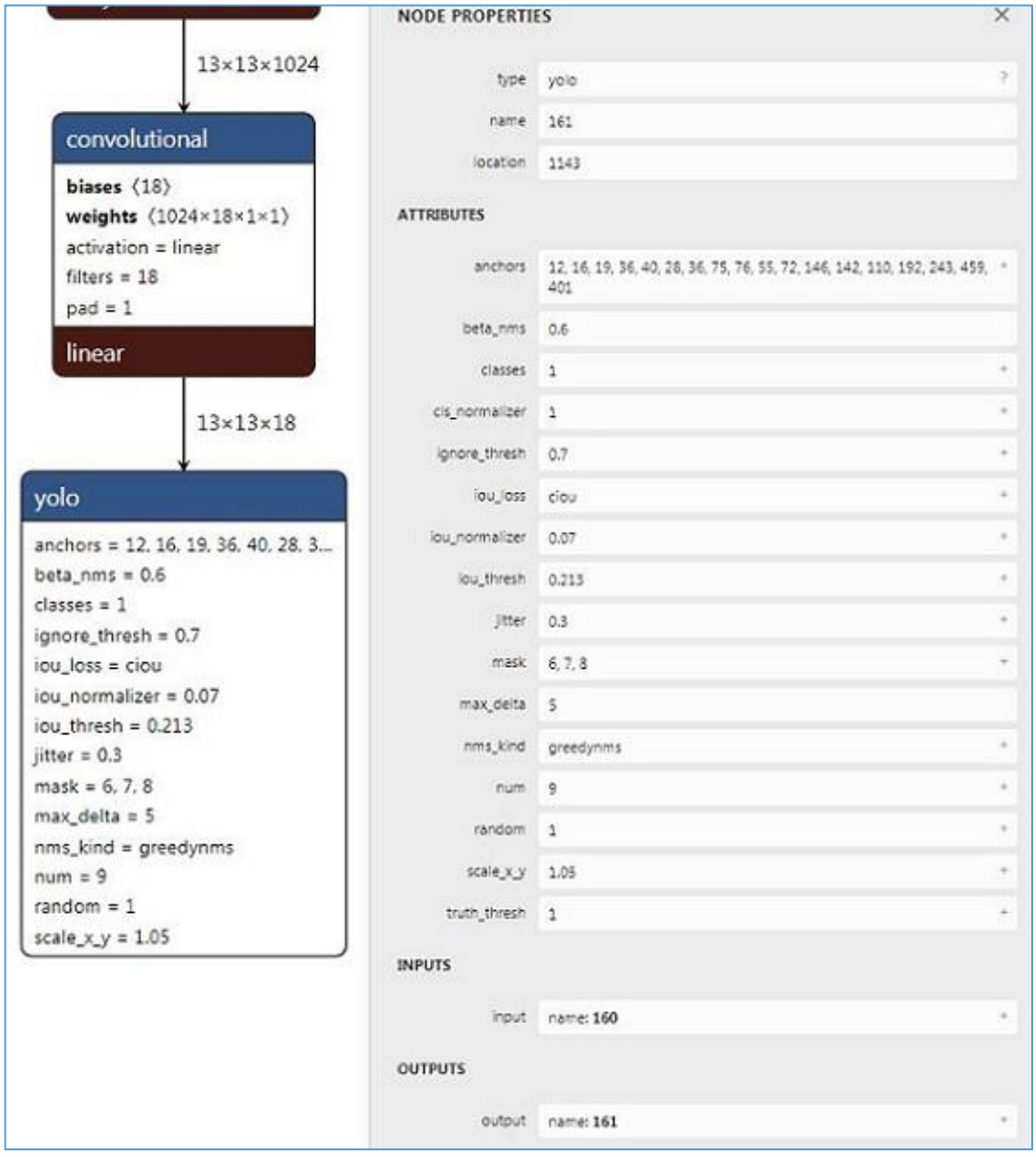

Рис. *27 Останні шари цієї моделі (скріншот програми Netron)*

ignore_thresh=.7 — зберігає повторювані виявлення, якщо IoU (виявлене, справжнє) > *ignore_thresh*, яке буде злито під час NMS [48] (використовується лише для навчання);

truth_thresh=1 — налаштовує повторювані виявлення, якщо IoU (виявлене, справжнє) > *truth_thresh*, яке буде злито під час NMS (використовується лише для навчання);

random=1 — випадково змінює розмір мережі для кожних 10 ітерацій з 1/1.4 до 1.4 (параметр збільшення даних використовується лише з останнього шару);

scale_x_y=1.05 — усуває чутливість сітки;

iou_thresh=0.213 — використовувати багато полів прив’язок на об’єкті, якщо *IoU (об’єкт, поле прив’язок) > 0.213*;

cls_normalizer=1.0 — нормалізатор для дельта-об'єктивності;

iou_normalizer=0.07 — нормалізатор для дельта-IoU.

3.3 Компоненти системи для знаходження контурів білборда та накладення зображення

Компонент для пошуку контурів виявленого білборда починає працювати після отримання на вхід зображення (кадр відео) та списку обмежувальних рамок навколо кожного об’єкта, розпізнаного нейронною мережею.

Робота компонента складається з підготовчої частини та декількох етапів і наведена на рисунку 28.

Підготовкою є вилучення з зображення частини (лістинг 1), навколо якої була визначена обмежувальна рамка за допомогою верхнього лівого та нижнього правого кутів рамки.

Лістинг *1 Вирізання зображення за допомогою OpenCV*

```
import cv2
img = cv2.imread(img_path)
crop_img = img[top:bottom, left:right].copy()
```

Наступна дія підготовки — застосування фільтру Кувахари [82] з розміром вікна згортки (рис. 29), що дорівнює 5, до частини зображення. Фільтр Кувахари — це нелінійний згладжуючий фільтр, який використовується при обробці зображень для адаптивного зменшення шуму.

Більшість фільтрів, використовуваних для згладжування зображення, є лінійними фільтрами низьких частот, які ефективно зменшують шум, але також розмивають краї. Однак цей фільтр після обробки зберігає краї об’єктів на зображенні. Після такої фільтрації фотографія стає схожою на грубо намальовану фарбами картину.

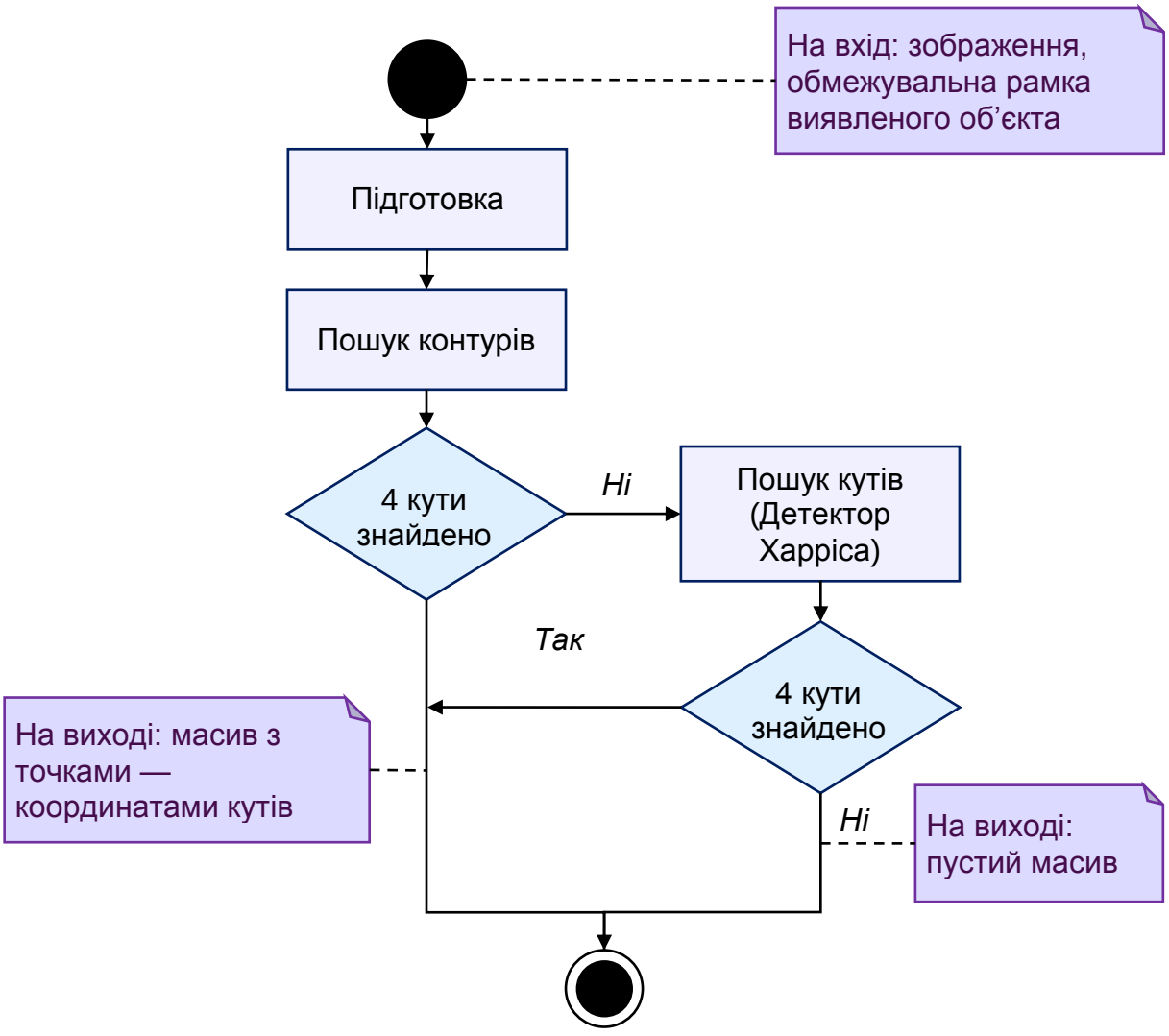

Рис. *28 Блок-схема роботи компонента для пошуку контурів білборда*

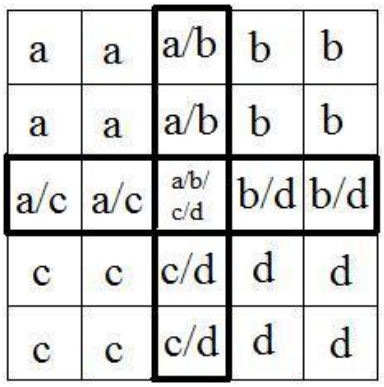

Рис. *29 Вікно згортки, використовуване фільтром Кувахари*

Середнє арифметичне $m_i(x, y)$ та стандартне відхилення $\sigma_i(x, y)$ з чотирьох областей з центром навколо пікселя (x, y) обчислюються та використовуються для визначення значення центрального пікселя. Результат фільтра Кувахари Ф(x, y) для будь-якої точки (x, y) тоді визначається як $\Phi(x, y) = m_i(x, y)$, де $i = \arg\min_j \sigma_j(x, y)$. Це означає, що центральний піксель прийме середнє значення площі, яка є найбільш однорідною. Розташування пікселя по відношенню до контру відіграє велику роль у визначенні, яка область матиме більше стандартне відхилення. Наприклад, якщо піксель розташований на темній стороні краю, він, швидше за все, прийме середнє значення темної області. З іншого боку, якщо піксель буде на світлішій стороні краю, це, швидше за все, прийме світле значення. У випадку, якщо піксель знаходиться на контурі, він прийме значення більш гладкої, найменш текстурованої області. Той факт, що фільтр враховує однорідність областей, забезпечує збереження країв об'єктів, використовуючи середнє значення, створює ефект розмиття.

Подібно до медіанного фільтра, фільтр Кувахари використовує підхід з ковзним вікном для доступу до кожного пікселя на зображенні. Розмір вікна вибирається заздалегідь і може змінюватися залежно від бажаного рівня розмитості на кінцевому зображенні. Більші вікна зазвичай призводять до створення більш абстрактних зображень, тоді як маленькі вікна створюють зображення, які зберігають свої деталі. Зазвичай вікна вибирають квадратними зі сторонами, які мають непарну кількість пікселів для симетрії (5, 9, 13,…).

Щоб застосувати фільтр Кувахари для кольорових зображень, неправильною буде практикою обробка кожного з каналів RGB окремо. Оскільки квадранти матимуть різні стандартні відхилення для кожного з каналів. Наприклад, верхній лівий квадрант може мати найнижче стандартне відхилення в червоному каналі, але нижній правий квадрант може мати найнижче стандартне відхилення в зеленому каналі. Така ситуація призведе до того, що колір центрального пікселя буде визначатися різними регіонами, що може призвести до кольорових артефактів або розмитих країв. Тому спочатку зображення перетворюється в кольорову модель HSV, і фільтр працює лише на каналі “brightness” (яскравість) — координаті *Value* в моделі HSV. Дисперсія “яскравості” кожного квадранта обчислюється для визначення квадранта, з якого слід взяти остаточний відфільтрований колір. Фільтр поверне вихідний сигнал для кожного каналу, який буде відповідати середньому значенню цього каналу з квадранта, який мав найнижче стандартне відхилення в “яскравості”. Це гарантує, що тільки одна область визначатиме значення RGB центрального пікселя. Приклади застосування фільтру до зображень з білбордами представлено на рисунку 30.

Рис. *30 Зображення білбордів до (зліва) та після (справа) відпрацювання фільтру Кувахари*

Основний етап роботи компонента — знаходження замкнутого контуру об’єкта (білборда). Приклад знайденого контуру зображено на рисунку 31.

Рис. *31 Знайдений контур на зображенні з білбордом*

З отриманого контуру відбираються його точки, найближчі до кутів зображення (рис. 32). Тобто точки, які знаходяться в околі кожного кута зображення; радіус околу дорівнює $((w + h) / 2) \times 0.2$, де w та h — ширина та висота зображення відповідно.

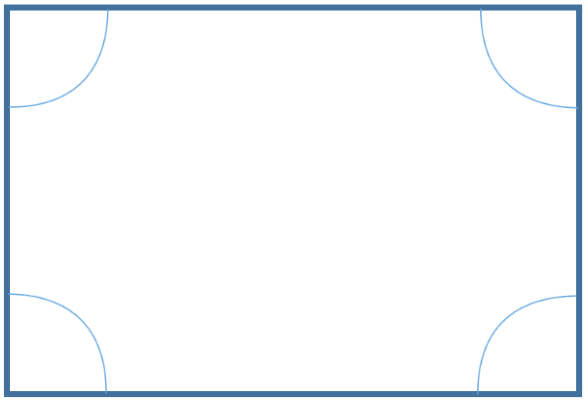

Рис. *32 Околи кутів зображення*

Якщо не вдалося знайти контури білборда, або кути в околах кутів зображення, компонент намагається знайти кути зображеного білборда методом Харріса [83] в наведених вище околах кутів зображення.

У разі успіху одного з двох методів пошуку точок на вихід подається список чотирьох точок.

Якщо не було знайдено жодної точки, результатом роботи є пустий масив.

Компонент накладання зображення спочатку змінює це зображення за допомогою перспективної трансформації, використовуючи чотири точки, що прийшли на вхід як результат роботи попереднього компонента системи.

Якщо на вхід прийшов пустий масив, буде накладання еліпсу по центру зображення (лістинг 2).

Лістинг *2 Додавання еліпсу на зображення з використанням OpenCV*

```
cv2.ellipse(image, (int(width / 2), int(height / 2)),
(90,50), 0, 0, 360, (0,0,255), (-1))
```

3.4 Вимоги до програмного та апаратного забезпечення

Задачею застосунку є зміна відео або зображень за допомогою моделі нейронної мережі та бібліотеки комп'ютерного зору OpenCV. Тому були визначенні наступні вимоги до системи.

Мінімальні вимоги до апаратного забезпечення клієнту:

– Двох-ядерний процесор з базовою тактовою частотою 2.6 ГГц.
– Оперативна пам'ять ємністю не менше 2 ГБ.
– Відеоадаптер NVIDIA з підтримкою CUDA 10 або вище.

Рекомендовані вимоги до апаратного забезпечення клієнту:

– Чотирьох-ядерний процесор з базовою тактовою частотою 3.0 ГГц.
– Оперативна пам'ять ємністю не менше 4 ГБ.
– Відеоадаптер NVIDIA з підтримкою CUDA 10 або вище.

3.5 Опис функціональних можливостей системи

Оскільки основною метою кваліфікаційної роботи є дослідження стану проблеми, існуючих алгоритмів та методів її вирішення то функціональні можливості системи орієнтовані на зручність проведення досліджень методів виявлення об'єктів (білбордів) та їх заміни на кадрах відео.

Було реалізовано застосунок для обробки відео та зображень. За допомогою команд обирається:

– вхідне зображення або відео, яке потрібно обробити;
– зображення, яке буде накладене на виявлені білборди;
– режим виявлення білбордів — тільки додавання обмежувальних рамок навколо розпізнаних білбордів (без накладання зображення).

Реалізовані алгоритми та розроблені компоненти можуть використовуватись дослідниками та іншими розробниками ПЗ при розв'язанні задач у споріднених областях знань. Якщо використовувати тільки компонент системи, призначений для виявлення білбордів, сценарій використання зводиться

до: демонстрації процесу обробки відео — додавання обмежувальних рамок на кожний кадр; збереження обробленого відео у новий файл.

Залишається можливою заміна файлів конфігурації (зміна шарів нейронів) та ваг моделі нейронної мережі, призначеної для вирішення задачі виявлення об’єктів. В такому разі необхідні налаштування гіперпараметрів та навчальний набір, потрібний у випадку навчання певної нової моделі. Даний варіант використання найсприятливіший для роботи тільки з компонентом виявлення об’єктів; щоб коректно працювали інші компоненти комп’ютерної системи, накладається вимога: об’єкти, які бажано замінювати на зображеннях (кадрах відео), мають бути прямокутної форми.

3.6 Висновки з проекту комп’ютерної системи

У розділі проекту програмної частини було описано архітектуру програмного застосунку, вимоги до програмного та апаратного забезпечення та описані функціональні можливості системи.

РОЗДІЛ 4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

4.1 Навчальні набори та їх попередня підготовка

Набір Google Open Images. Для навчання мережі було застосовано Google Open Images Dataset v6 [84]. Інформаційні файли та файли різних конфігурацій щодо зображень та їх об’єктів представлені у форматі csv, коли стовпці відділені комами один від іншого.

Датасет складається з 600 класів об’єктів. Інформація про класи знаходиться у файлі *class-descriptions-boxable.csv* (рис. 33).

	A	B
87	/m/01knjb	Billboard
88	/m/01krhy	Tiara
89	/m/01lcw4	Limousine
90	/m/01llwg	Necklace
91	/m/01lrl	Carnivore
92	/m/01lsmm	Scissors
93	/m/01lynh	Stairs
94	/m/01m2v	Computer keyboard
95	/m/01m4t	Printer
96	/m/01mqdt	Traffic sign
97	/m/01mzpv	Chair
98	/m/01n4qj	Shirt
99	/m/01n5jq	Poster
100	/m/01nkt	Cheese
101	/m/01nq26	Sock
102	/m/01pns0	Fire hydrant
103	/m/01prls	Land vehicle

Рис. *33 Фрагмент файлу з відповідністю MID класів об’єктів до їх назв*

Файл *oidv6-train-annotations-bbox.csv* містить інформацію про об’єкти тренувального набору. Його стовпці записані у наступній послідовності:

1) *ImageID* — ідентифікатор зображення, в якому знаходиться обмежувальне поле;

2) *Source* вказує, як була зроблена обмежувальна рамка:

– *xclick* — це вручну намальовані поля за допомогою методу, представленого в статті [85], коли анотатори вручну помічали чотири крайні точки об’єкта;

– *activemil* — це рамки, зроблені розширеною версією методу [86]. Це перевірено людиною на точність при IoU > 0,7.

3) *LabelName* — MID класу об’єкта, до якого належить це поле;

4) *Confidence* — фіктивне значення, завжди 1;

5) *XMin, XMax, YMin, YMax* — координати рамки в нормалізованих координатах зображення. *XMin* знаходиться в діапазоні [0, 1], де 0 — крайній лівий піксель, а 1 — крайній правий піксель на зображенні. Координати *Y* переходять від верхнього пікселя (0) до нижнього пікселя (1);

6) *IsOccluded* вказує на те, чи закритий об’єкт іншим об’єктом на зображенні;

7) *IsTruncated* вказує на те, чи виходить об’єкт за межі зображення;

8) *IsGroupOf* вказує на те, що обмежувальна рамка охоплює групу предметів (наприклад, квіткове ліжко або натовп людей). У цьому наборі анотатори використовують цей тег для випадків із більш ніж 5 екземплярами, які сильно закупорюють один одного та є фізично вагомими;

9) *IsDepiction* позначає, що об’єкт є зображенням (наприклад, мультфільм або малюнок об’єкта, а не реальний фізичний примірник);

10) *IsInside* позначає знімок, зроблений із внутрішньої сторони об’єкта (наприклад, в салоні автомобіля або всередині будівлі);

11) *XClick1X, XClick2X, XClick3X, XClick4X, XClick1Y, XClick2Y, XClick3Y, XClick4Y* — нормалізовані координати зображення (як *XMin* та ін.) чотирьох крайніх точок об’єкта, що створили обмежуючу рамку, використовуючи [85] у випадку з *xclick*. Якщо було застосовано метод *activemil*, поля мають фіктивне значення –1.

Розмір даного файлу сягає 2 ГБ; для перегляду та ручного опрацювання його не відкрити ні табличним процесором (наприклад, MS Excel), ні текстовим редактором на кшталт Notepad++. Для зручності подальшої роботи з на-

вчальними наборами необхідно відібрати тільки ті зображення, на котрих присутній бажаний клас об’єктів — білборд. Програма в лістингу 3 відбирає клас *білборд* (його ідентифікатор *“/m/01knjb”*) у новий csv-файл.

Лістинг *3 Програма для відбору зображень з класом білборд*

```
static void Main(string[] args)
{
    const string billboardLabel = @"/m/01knjb";

    using var r = new StreamReader(args[0]);
    using var w = new StreamWriter(args[1]);

    w.WriteLine(r.ReadLine());

    while (!r.EndOfStream)
    {
        var row = r.ReadLine();
        if (row.Split(',')[2] == billboardLabel)
        {
            w.WriteLine(row);
        }
    }
}
```

Після виконання вищевказаної фільтрації було відібрано 9824 записи.

Третім важливим інформаційним файлом є *train-images-boxable-with-rotation.csv* з адресами зображень. Документ складається з наступних полів:

1) *ImageID* — ідентифікатор зображення.

2) *Subset* — підмножина набору; може бути “train”, “validation” або “test”.

3) *OriginalURL* — URL-адреса зображення з оригінальними розмірами;

4) *OriginalLanding* — URL-адреса джерела, де було розміщене зображення.

5) *License* — URL-адреса на ліцензію, з якою викладено зображення. Усі відібрані картинки для даної роботи мають ліцензію Creative Commons Attribution 2.0 Generic [87], за якої є можливість:

– скопіювати та розповсюдити матеріал у будь-якому носії чи форматі;
– змішувати, трансформувати та використовувати матеріал для будь-яких цілей, навіть комерційних.

6) *AuthorProfileURL* — URL-адреса профілю автора зображення.

7) *Author* — ім’я користувача автора зображення.

8) *Title* — назва зображення.

9) *OriginalSize* — розмір файлу оригінального зображення.

10) *OriginalMD5* є двійковим MD5, кодованим base64.

11) *Thumbnail300KURL* — це необов’язкова URL-адреса зображення з ~300K пікселів (~640×480), що передбачено для зручності завантаження даних за відсутності більш зручних способів отримання зображень. У разі відсутності слід використовувати *OriginalURL* (і, якщо потрібно, змінити розмір до потрібного розміру).

12) *Rotation* — це кількість градусів, на яких зображення слід повертати проти годинникової стрілки, щоб відповідати призначеній користувачеві Flickr орієнтації (0, 90, 180, 270).

Функція в лістингу 4 використана для завантаження усіх необхідних зображень.

Лістинг *4 Функція для завантаження зображень у файлову систему*

```
/// <summary>
/// Завантажити картинки
/// </summary>
/// <param name="annotationFilePath">
/// Шлях до файлу анотацій</param>
/// <param name="urlsFilePath">
/// Шлях до файлу з адресами зображень</param>
/// <param name="destPath">Шлях,
/// куди мають бути завантажені зображення</param>
static async Task LoadImages(string annotationFilePath,
    string urlsFilePath, string destPath)
{
    var list = new HashSet<string>();
    using var r = new StreamReader(annotationFilePath);
    r.ReadLine();
```

```
    //Зчитування всіх Id зображень
    //з файлу oidv6-train-annotations-bbox.csv
    while (!r.EndOfStream)
        list.Add(r.ReadLine().Split(',')[0]);
    var dict = new Dictionary<string, string>();
    using var r2 = new StreamReader(urlsFilePath);
    r2.ReadLine();

    //Зчитування усіх адрес з поля Thumbnail300KURL
    while (!r2.EndOfStream)
    {
        var row = r2.ReadLine();
        var splitted = row.Split(',');
        var imgId = splitted[0];
        if (list.Contains(imgId))
        {
            var idx = row.LastIndexOf(@"https://");
            var url = row[idx..].Split(',')[0];
            dict.Add(imgId, url);
        }
    }

    //Завантаження усіх зображень
    using var client = new HttpClient();
    foreach (var kv in dict)
    {
        var resp = await client.GetAsync(kv.Value);
        await File.WriteAllBytesAsync(
            Path.Combine(destPath, kv.Key + ".jpg"),
            await resp.Content.ReadAsByteArrayAsync()
        );
    }
}
```

Усього завантажено 3399 зображень, на яких присутні об’єкти класу *“білборд”*, проте їх необхідно перевірити у ручному режимі. Після ручного перебору вибрано 1026 зображень з білбордами.

Набір Nexet. У 2017 році компанія Nexar опублікувала датасет Nexet [88] — набір фотознімків автомобільних доріг для дослідників у світі. Мета компанії — створити першу у світі мережу безпечного водіння: поєднавши відеореєстратор з керуванням Nexar та додатком Nexar, користувач приєдну-

ється до мережі, яка використовує штучний інтелект для виявлення дорожніх небезпек, а потім попереджає інших водіїв у околицях.

Набір складається з 50 тисяч фотографій. Містить зображення, зроблені за різних умов освітлення (рис. 34), погодних умов (рис. 35) та географічного розташування. Фотографії зібрані з 77 країн світу з використанням Nexar app.

Рис. *34 Процентне співвідношення кількостей фотознімків за умовами освітлення на них*

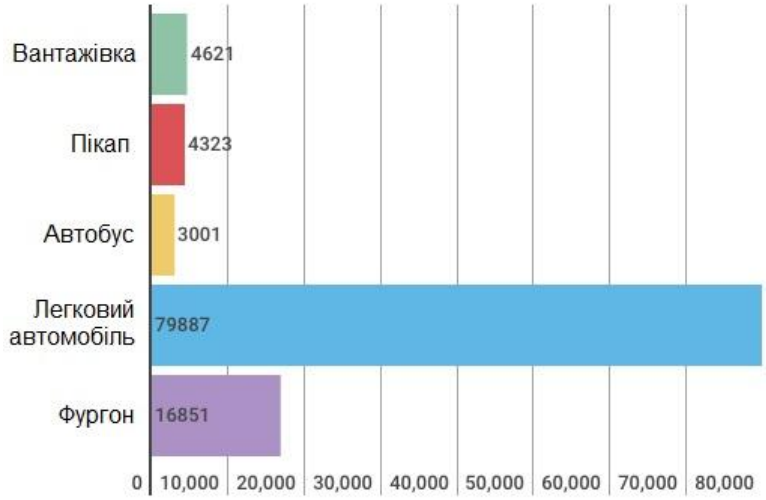

Рис. *35 Розподіл кількостей автомобільних класів на фотознімках*

В архіві цього датасету знаходиться файл *train.csv* з полями (рис. 36):

1) *image_filename* — назва файлу зображення;

2) *lighting* — умови освітлення (можливі значення: “Night”, “Day”, “Twilight”);

3) *city* — скорочена назва міста.

```
image_filename,lighting,city
frame_20f328fa-2459-46d0-97a5-5ae2d6103cb0_00000-1280_720.jpg,Twilight,NYC
frame_927bde20-f97f-48c2-af30-f9127b6b32ce_00000-1280_720.jpg,Day,NYC
frame_67012509-f3bd-4175-a9d2-565a7b6bb3c7_00001-1280_720.jpg,Day,NYC
frame_bd043377-6fb8-407a-95e5-7deb1fbab13a_00004-1280_720.jpg,Day,NYC
frame_4da1583b-58d0-4893-8149-54541191031d_00000-1280_720.jpg,Day,NYC
frame_0726993f-b5d9-458f-8abb-08cd48633e23_00000-1280_720.jpg,Night,NYC
frame_5fc7aa83-5bcc-4727-9a26-2d6043888efa_00000-1280_720.jpg,Night,NYC
frame_23f17d60-6a66-4010-af8e-aca216fb3bf8_00001-1280_720.jpg,Night,NYC
frame_2ee2a8a2-d113-443c-a519-b34b02987ebd_00000-1280_720.jpg,Night,NYC
frame_10ef103d-9163-4f7a-b985-366ad1fc891f_00000-1280_720.jpg,Night,NYC
frame_87a79a61-6379-4e73-9abb-a4a4c1450ac1_00000-1280_720.jpg,Night,NYC
frame_49d0e046-7b73-4c93-a018-bc08506a7c11_00002-1280_720.jpg,Day,NYC
frame_18b79379-390d-4688-8115-95ced377e0e3_00004-1280_720.jpg,Day,NYC
frame_479a7009-17f2-4daa-bfb1-4a2ded4a6bc7_00010-1280_720.jpg,Day,NYC
frame_79fb1e0b-6448-4560-9110-bd5737f814dc_00000-1280_720.jpg,Day,NYC
frame_312ed254-486a-47d8-885e-e113256f7c4c_00035-1280_720.jpg,Day,NYC
frame_de245677-9f32-47dd-a762-3a5657365cb2_00000-1280_720.jpg,Night,NYC
frame_c1eba23a-529e-4735-a783-6218bbe38b59_00000-1280_720.jpg,Night,NYC
frame_240f1324-a80c-42f7-8749-74e44de895b5_00002-1280_720.jpg,Day,NYC
frame_41bc43e2-4747-4f0e-b5dc-fd4f547972dd_00001-1280_720.jpg,Day,NYC
```

Рис. *36 Фрагмент файлу train.csv*

Оскільки в цьому наборі не виконувалось анотацій білбордів, потрібно зображення з ними відібрати вручну. Програма в лістингу 5 відбирає зображення згідно з умовами освітлення (відібрати усе, окрім нічних знімків).

Лістинг *5 Програма для відбору усіх фотознімків, окрім нічних*

```
static void Main(string[] args)
{
    var csvFilePath = args[0];
    var imgsFolder = args[1];
    var destFolder = args[2];

    using var reader = new StreamReader(csvFilePath);
        reader.ReadLine();

    while (!reader.EndOfStream)
    {
        var row = reader.ReadLine();
        var splitted = row.Split(',');
```

```
            var src = Path.Combine(imgsFolder, splitted[0]);
            var dest = Path.Combine(destFolder, splitted[0]);

            if (splitted[1] != "Night"
                && File.Exists(src)
                && !File.Exists(dest))
                File.Copy(src, dest);
        }
}
```

В результаті аналізу файлу *train.csv* у наборі 24903 зображення з “Day”, 1874 зображення — “Twilight”, 23223 зображення — “Night”. Вручну було відібрано зображення, в яких наявні білборди; серед них 386 — “Night” та 742 “Day” і “Twilight” (рис. 37).

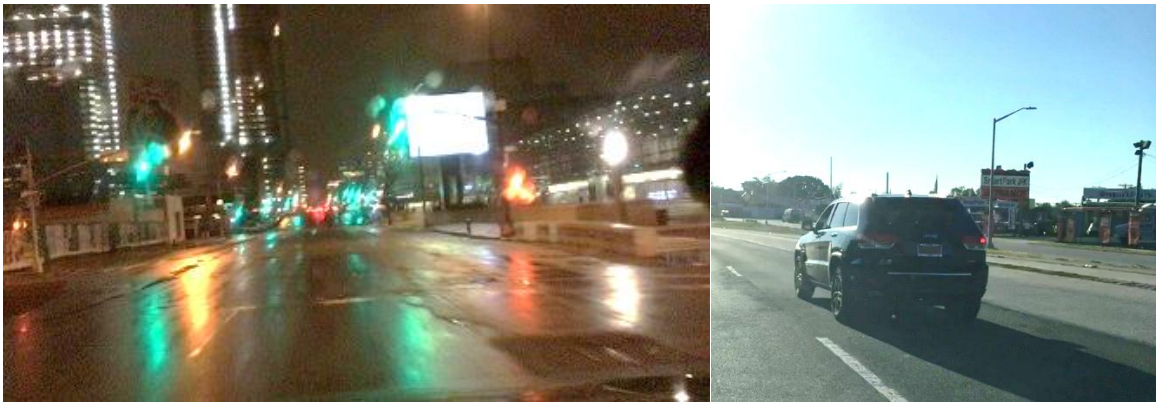

Рис. *37 Приклади відібраних зображень набору Nexet: зліва з умовами освітлення “Night”, справа — “Day”*

Набір Microsoft COCO. Цей датасет представили у 2015-му році. Його метою було вдосконалення рівня розпізнавання об’єктів; автори поставили задачу розпізнавання об’єктів у контексті більш широкого питання розуміння сцени. Це досягається збиранням зображень складних повсякденних сцен, що містять загальні предмети в їх природному контексті.

Анотації датасету представлені у форматі JSON та знаходяться у файлі *instances_train2017.json*. Формат даних анотацій MS COCO описаний в [89]. У наборі клас *“білборд”* відсутній, тому для ручного відбору зображень з наявними білбордами було вибрано за класом *“автомобіль”* (*CategoryId* — 3).

Метод для завантаження зображень, на яких присутні автомобілі, та необхідні класи-моделі для десеріалізації JSON представлено в лістингах 6 та 7.

Лістинг 6 *Функція для відбору зображень, на яких присутні автомобілі*

```
static async Task ReadCocoJson(string jsonPath,
    string destPath)
{
    using var stream = new FileStream(
        jsonPath,
        FileMode.Open, FileAccess.Read);

    var model = await System.Text.Json.JsonSerializer
        .DeserializeAsync<MainModel>(stream);
    var filteredIds = model.Annotations
        .Where(a => a.CategoryId == 3)
        .Select(a => a.ImageId)
        .Distinct()
        .ToList();
    var dict = model.Images
        .Where(i => filteredIds.Contains(i.Id))
        .ToDictionary(x => x.Id.ToString(),
                      x => x.CocoUrl)
        .ToList();

    var counter = 0;
    using var client = new System.Net.Http.HttpClient();
    foreach (var kv in dict)
    {
        var path = Path.Combine(destPath, kv.Key + ".jpg");
        if (!File.Exists(path))
        {
            var resp = await client.GetAsync(kv.Value);
            await File.WriteAllBytesAsync(path,
            await resp.Content.ReadAsByteArrayAsync());
            Console.WriteLine("N: " + ++counter);
        }
    }
}
```

Лістинг *7 Моделі для десеріалізації анотацій з формату JSON*

```
//Модель кореневого елемента анотацій
public class MainModel
{
    [JsonPropertyName("info")]
    public InfoModel Info { get; set; }
```

```
    [JsonPropertyName("images")]
    public List<ImageModel> Images { get; set; }

    [JsonPropertyName("annotations")]
    public List<AnnotationModel> Annotations { get; set; }
}

//Модель для отримання колекції Annotations
public class AnnotationModel
{
    [JsonPropertyName("segments_info")]
    public List<SegmentInfoModel> SegmentsInfo
        { get; set; }

    [JsonPropertyName("image_id")]
    public int ImageId { get; set; }

    [JsonPropertyName("file_name")]
    public string FileName { get; set; }

    [JsonPropertyName("category_id")]
    public int CategoryId { get; set; }
}

//Модель для отримання адреси зображення
public class ImageModel
{
    [JsonPropertyName("id")]
    public int Id { get; set; }

    [JsonPropertyName("width")]
    public int Width { get; set; }

    [JsonPropertyName("height")]
    public int Height { get; set; }

    [JsonPropertyName("file_name")]
    public string FileName { get; set; }

    [JsonPropertyName("flickr_url")]
    public string FlickrUrl { get; set; }

    [JsonPropertyName("coco_url")]
    public string CocoUrl { get; set; }
}
```

В результаті було вручну вибрано 132 зображення, на яких присутні білборди. Також 9813 зображень віднесено до негативної вибірки, оскільки об’єктів даного класу на них немає.

Набір Mapillary Vistas Dataset. Датасет Mapillary був опублікований у 2017 році [26]. Це масштабний набір зображень вуличного рівня, що містить 25000 зображень високої роздільної здатності, анотованих у 66 категорій об’єктів із додатковими специфічними мітками для 37 класів. Mapillary — це сервіс під керівництвом громади для людей, які спільно хочуть візуалізувати світ за допомогою вуличних фотографій. Будь-який бажаючий може додати фотографії з будь-якого місця, і дані доступні кожному для їх вивчення відповідно до ліцензійної угоди CC-BY-SA. Mapillary призначений для охоплення широкого кола зовнішніх сцен, доступних у всьому світі. Основними мотивами для розробки та впровадження цього набору даних були різноманітність, багатство деталей та географічний обсяг даних вуличного рівня. Враховуючи ці вимоги, метою було скласти набір даних із зменшеним упередженням до високорозвинених країн і замість цього відображати неоднорідні композиції за зовнішнім виглядом, які можна знайти у перспективі вуличного рівня по всьому світу.

В архіві датасету знаходяться директорії *training*, *validation* та *testing*; остання не має анотацій для об’єктів на зображеннях і призначена для перевірки роботи вже навченої системи. Файл *panoptic_2018.json* містить інформацію про об’єкти на кожному зображенні (анотації) у форматі COCO. Датасет має клас “billboard” з обмежувальними рамками, тому можна створити позитивну вибірку для навчання мережі. В результаті відбору зображень за класом *“білборд”* з анотацій, отримано 3491 зображення. Проте, оскільки середня роздільна здатність зображень набору складає 3000 × 2000, необхідно перевірити колекцію вручну, щоб виключити дуже дрібні об’єкти, або розділити картинку навпіл. Після ручного перебору вийшло 1075 зображень з білбордами; 3099 зображень з Mapillary потрапили до негативної вибірки.

Набір JAAD. JAAD [90] — це набір даних для вивчення спільної уваги (joint attention) в контексті автономного водіння. Автори фокусувалися на поведінці пішоходів та водіїв у місцях перетину та чинниках, що впливають на них. Цей датасет являє собою колекцію з 346 коротких відео (довжиною 5-10 секунд), отриманих із понад 240 годин відеозаписів. Відеоролики зняті в кількох місцях Північної Америки та Східної Європи; представляють сцени, типові для повсякденної міської їзди в різних погодних умовах. Анотації містять обмежувальні рамки навколо усіх пішоходів, мітки їх поведінки з ситуацією (наприклад, зупинка, ходьба, перегляд тощо), демографічні атрибути (наприклад, вік, стать, напрямок руху тощо), а також перелік видимих елементів місця дорожнього руху (наприклад, знак зупинки, дорожній сигнал тощо) для кожного кадру.

Після отримання кадрів відео за допомогою VLC-плеєра вручну було відібрано 64 зображення, на яких є білборди; ще 32 зображення віднесено до негативної вибірки (рис. 38).

Рис. *38 Кадр з наявним на ньому білбордом (зліва) та кадр без об'єктів "білборд" (справа)*

Анотування отриманого датасету

Програма Yolo_mark (рис. 39) [91] призначена для анотування зразків навчальних наборів для задачі виявлення об'єктів. Вона для кожного зображення створює (або змінює) текстовий файл зі строками вигляду:

Label_ID X_CENTER Y_CENTER WIDTH HEIGHT,

де *Label_ID* — це числовий ідентифікатор, що присвоюється різним класам, які потрібно визначити, починаючи з 0;

X_CENTER, Y_CENTER — координати (X; Y) центру об’єкта, який повинен бути виявлений, відносно ширини та висоти зображення;

WIDTH, HEIGHT — відносні до розмірів зображення ширина та висота об’єкта, який повинен бути виявлений.

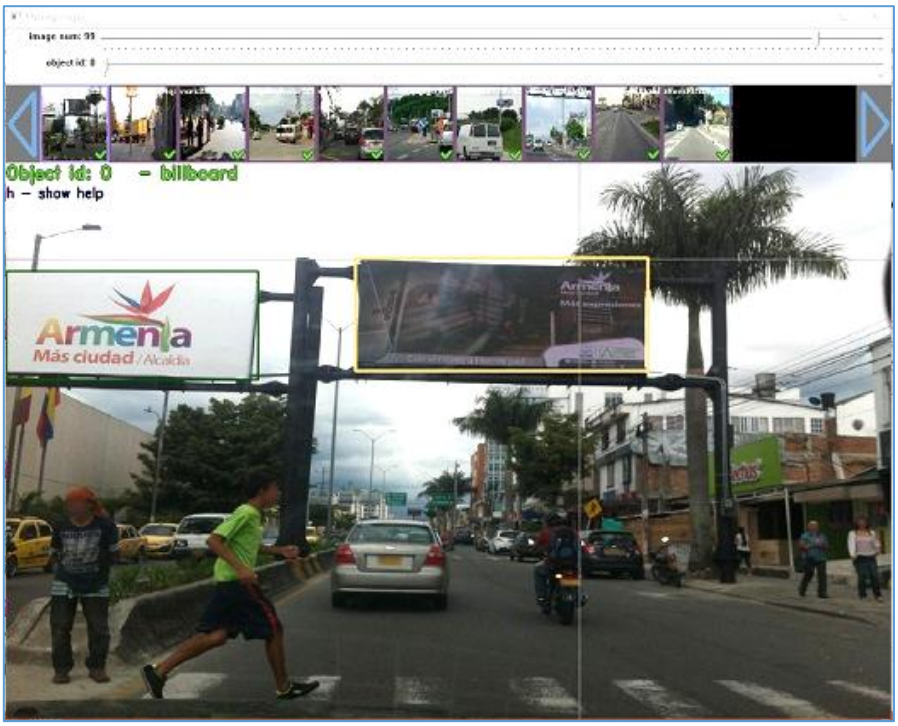

Рис. *39 Програма Yolo_mark для анотування датасету*

Для роботи програми потрібно змінити два файли:

obj.names — містить назви міток (класів), кожна мітка вводиться з нового рядка. Для даної системи це один рядок — *billboard*;

obj.data — містить інформацію про кількість класів, файли train та validation, розташування папки резервного копіювання (де зберігаються ваги після тренування). В лістингу 8 наведені значення файлу, використовуваному в даній роботі.

Лістинг *8 Вміст файлу obj.data*

```
classes= 1
```

```
train  = build/darknet/x64/data/training.txt
valid  = build/darknet/x64/data/validation.txt
names = build/darknet/x64/data/obj.names
backup = build/darknet/x64/backup/
```

4.2 Навчання моделі нейронної мережі

Для навчання моделі нейронної мережі необхідно створити файл зі шляхами до кожного зразка тренувального набору. Тому було створено програму (лістинг 9), яка переглядає вміст директорій з зібраними картинками білбордів (для зручності з назвою *Positive*), з картинками без білбордів (*Negative*), записує шляхи до цих зображень у файл *training.txt*, при чому дотримуючись рівної кількості між негативними та позитивними зразками.

Лістинг *9 Функція отримання шляхів до зразків навчального набору*

```
/// <summary>
/// Функція для отримання шляхів до зразків
/// навчального набору
/// </summary>
/// <param name="folderPath">Директорія з папками
/// датасетів з зображеннями</param>
/// <param name="negativeDict"></param>
/// <returns></returns>
static (List<string>, List<string>) GetDatasetImagePaths(
    string folderPath, Dictionary<string, int>
negativeDict)
{
    var training = new List<string>();
    var validation = new List<string>();

    (IEnumerable<string>, IEnumerable<string>) GetPaths(
        string dirPath, string dirName, int takeNumber = 0)
    {
        //Шляхи до файлів у випадковому порядку
        var filePaths = Directory.EnumerateFiles(
            Path.Combine(dirPath, dirName),
            "*.jpg")
            .OrderBy(f => Guid.NewGuid())
            .ToList();

        if (takeNumber > 0)
        {
```

```
            filePaths = filePaths
                .Take(takeNumber).ToList();
        }

        var trainNumber = Convert.ToInt32(
            Math.Round(filePaths.Count * 0.83));

        return (filePaths.Take(trainNumber),
            filePaths.Skip(trainNumber));
    }

    foreach (var dir in Directory
           .EnumerateDirectories(folderPath))
    {
        {
            var (tr, val) = GetPaths(dir, "Positive");
            training.AddRange(tr);
            validation.AddRange(val);
        }

        var dirName = Path.GetFileName(dir);
        int negNumber = 0;
        if (negativeDict.ContainsKey(dirName))
            negNumber = negativeDict[dirName];

        if (Directory.Exists(
            Path.Combine(dir, "Negative")))
        {
            var (tr, val) = GetPaths(dir,
                "Negative", negNumber);
            training.AddRange(tr);
            validation.AddRange(val);
        }
    }

    training = training.OrderBy(f => Guid.NewGuid())
        .ToList();
    validation = validation.OrderBy(f => Guid.NewGuid())
        .ToList();
    return (training, validation);
}
```

В результаті роботи програми було відібрано 3602 зразки для позитивної вибірки, та стільки ж для негативної; детальне розподілення зразків з кожного набору представлене у таблиці 9.

Таблиця 9

Кількості зразків з кожного навчального набору

Датасет	Positive	Negative
JAAD	64	23
COCO	132	132
Mapillary	1075	1575
Nexet (day)	742	1087
Nexet (night)	386	661
OIDv6	1026	—
Власні зображення	177	124
Усього:	3602	3602

Навчання нейронної мережі тривало 36 годин. Воно було призупинене на 5200-й ітерації з середнім значенням функції втрат (*loss*) — 0.34. Графік зміни значення втрат на кожній ітерації зображено на рисунку 40.

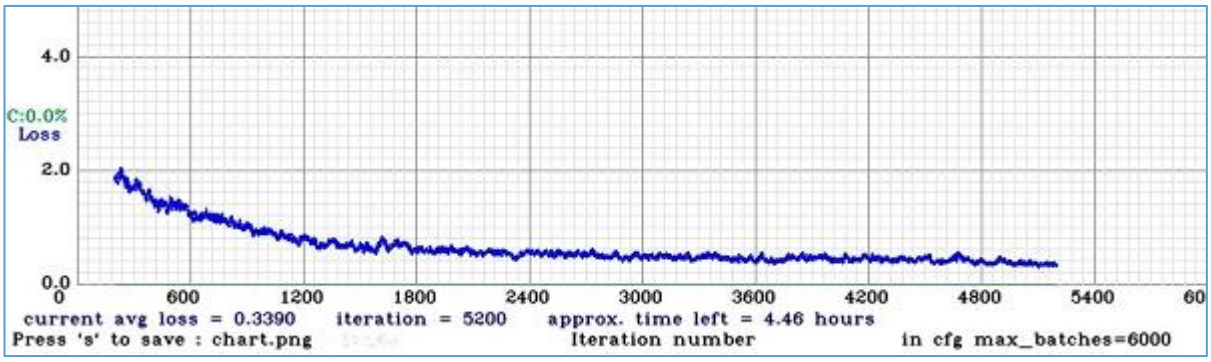

Рис. *40 Графік функції втрат на кожній ітерації у процесі навчання нейронної мережі*

4.3 Вимірювання точності виявлення білбордів

На фотознімках зібраного датасету білборди розташовані в різних місцях під різними кутами, різних розмірів; у набір також включено білборди, 40% площі яких закриті деревами або іншими білбордами. Найчастіше це знімки денного часу; приблизно 10% зроблено у нічний, на більшості з яких

білборди підсвічені. На рисунку 41 наведено невдалий випадок виявлення об’єкта, тому що частка непідсвічених білбордів на нічних фото в датасеті вкрай мала, мережа не зможе їх розпізнати.

Рис. *41 Приклад невдалого виявлення білборда*

Для перевірки точності виявлення білбордів були використані файли з автомобільного відеореєстратора. Роздільна здатність таких відео складає 640 × 360 та 2560 × 1440. На рисунку 42 наведені скріншоти з процесу розпізнавання білбордів на відео.

Рис. *42 Вдалі випадки виявлення білбордів на відео*

Середня швидкість обробки відео під час розпізнавання становить 16 FPS. Мережа погано виявляє білборди, які займають менше 5% зображення (кадра відео).

4.4 Вимірювання точності знаходження кутів виявлених білбордів та швидкості заміни вмісту

Далі була протестована робота системи в цілому: виявлення білбордів та накладання зображення на виявлені білборди. Середня швидкість обробки склала 10 FPS. Це пояснюється викликами функцій для обробки частин зображення з виявленими білбордами (фільтр Кувахари), знаходження контурів білборда, його кутів; функціями для перспективних трансформацій зображення, яке накладається на кожний білборд.

Рис. *43 Успішне накладення зображення на білборд на кадрах відео*

На рисунку 43 представлено два випадки вдалого розпізнавання кутів білборда та накладання на нього зображення: на нижньому кадрі накладена картинка була деформована, оскільки нижні кути виявленого білборда закриті автобусом.

На рисунку 44 випадок, коли системі не вдалося знайти кути білборда через закриття деревом, тому не вийшло накласти зображення.

Рис. *44 Випадок з невдалим знаходженням кутів білборда*

4.5 Висновки реалізації системи виділення об’єктів та заміни їх вмісту

При реалізації даної системи важливим критерієм була точність розпізнавання білбордів у різних умовах та під різними кутами. Описані в цьому розділі тренувальні набори допомогли налаштувати модель нейронної мережі. Залишається можливим донавчити нейронну мережу зразками з непідсвіченими білбордами на нічних фото.

Реалізований застосунок був розроблений для демонстрації можливостей виявлення білбордів на відео.

ВИСНОВКИ

У результаті виконання даної кваліфікаційної роботи отримано:

1. Встановлена актуальність розробки комп'ютерної системи для виділення об'єктів у відеопотоці із заміщенням вмісту. Шляхом аналізу існуючих можливостей в області машинного навчання та проведення теоретичних та практичних досліджень було сформульовано набір методів для створення такої системи, визначено стек технологій та інструментів розробки та їх відносну продуктивність та зручність. Встановлено, що на ринку програмних продуктів відсутня система, яка може виявляти білборди на зображеннях та відео і накладати на них інший вміст.

2. Фреймворком для реалізації машинного навчання обрано Darknet, а модель нейронної мережі для задачі виявлення об'єктів — YOLOv4. Основними інструментами для роботи з зображеннями та відео обрано бібліотеку OpenCV, яка є найбільш розповсюдженим інструментом для обробки зображень, та мову Python.

3. Були відібрані фото з білбордами з існуючих датасетів. Таким чином створений та проанотований навчальний набір для тренування моделі нейронної мережі, яка вирішує задачу виявлення білбордів. На цьому датасеті навчено модель нейронної мережі YOLOv4.

4. Спроектовано та реалізовано компонент системи для знаходження кутів виявлених білбордів з попередньою обробкою зображення фільтрами та компонент для накладення обраного зображення на білборди.

5. Досліджено точність виявлення білбордів навченою моделлю нейронної мережі у розробленій системі.

6. Проаналізовано фактори, що впливають на точність знаходження кутів білбордів компонентами реалізованої системи.

7. Компоненти реалізованої системи можуть використовуватися окремо, або їх функціонал може бути розширений.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Karen S. Johnson. What Does the Term “Noise” Mean in Marketing? Chron. URL: https://smallbusiness.chron.com/term-noise-mean-marketing-57846.html (дата звернення: 24.06.2020).

2. Silverman D. IAB Internet Advertising Revenue Report. *An Industry Survey Conducted by PwC and Sponsored by the Interactive Advertising Bureau (IAB)*. 2012. 26 с.

3. Іщенко В. Ю., магістрант, Безверхий А. І., доцент — науковий керівник. Комп’ютерна система розпізнавання білбордів у відеопотоці з заміщенням змісту. Молода наука 2020 : зб. наук. праць студентів, аспірантів і молодих вчених. Запоріжжя : ЗНУ, 2020. Т.5. С. 86–87.

4. Іщенко В. Ю., магістрант, Безверхий А. І., доцент — науковий керівник. Комп’ютерна система для виділення об’єктів у відеопотоці із заміщенням вмісту. Матеріали XXIV науково-технічної конференції студентів, магістрантів, аспірантів, молодих вчених та викладачів. Запоріжжя : ІННІ ЗНУ, 2020. С. 163.

5. Montier L. Best Open Source Annotation Tools for Computer Vision. Sicara. URL: https://www.sicara.ai/blog/2019-09-01-top-five-open-source-annotation-tools-computer-vision (дата звернення: 15.08.2020).

6. Гавриш Б. М., Семенова О. Є. Застосування нейронних мереж для вирішення проблем розпізнавання тексту. *Моделювання та інформаційні технології*. 2018. Вип. 82. С. 193–199.

7. Folorunsho O., Adeyemo A. B. Application of Data Mining Techniques in Weather Prediction and Climate Change Studies. *International Journal of Information Engineering and Electronic Business 4(1)*. February 2012. P. 51–59.

8. Копча-Горячкіна Г. Е. Теорія розпізнавання образів. Частина І : навч.-метод. посібник. Ужгород : Видавництво ДВНЗ “Ужгородського національного університету”, 2016. С. 13.

9. Gupta S. Understanding Image Recognition and Its Uses. eInfochips, an Arrow company. URL: https://www.einfochips.com/blog/understanding-image-recognition-and-its-uses/ (дата звернення: 11.06.2020).

10. Goodfellow I. J., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A., Bengio Y. Generative Adversarial Nets. *Proceedings of the International Conference on Neural Information Processing Systems (NIPS 2014)*. Montréal, Canada, 2014. Vol. 2. С. 2672–2680.

11. Suwajanakorn S., Seitz S. M., Kemelmacher-Shlizerman I. Synthesizing Obama: learning lip sync from audio. *ACM Transactions on Graphics*. July 2017. Vol. 36, No 95. С. 95:1–95:13.

12. Peter F. An AI program will soon be here to help your deepfake dancing — just don't call it deepfake. Business Insider Australia. URL: https://www.businessinsider.com.au/artificial-intelligence-ai-deepfake-dancing-2018-8 (дата звернення: 24.05.2020).

13. Ayrle J. Ornella Muti in cortometraggio a Firenze. L'Agenzia Nazionale Stampa Associata. URL: https://www.ansa.it/toscana/notizie/2017/11/03/ornella-muti-in-cortometraggio-a-firenze_36349008-ce7b-4c7e-8742-43e28f7225f4.html (дата звернення: 15.05.2020).

14. Samuel A. L. Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*. July 1959. Vol. 3. P. 210–229. URL: https://ieeexplore.ieee.org/document/5392560 (дата звернення: 15.05.2020).

15. Motoda H., Liu H. Feature selection, extraction and construction. In: Towards the Foundation of Data Mining Workshop. *Sixth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD2002)*. Taipei, Taiwan, 2002. С. 67–72.

16. Janecek A. G. K., Gansterer G. F. On the Relationship between Feature Selection and Classification Accuracy. *Proceeding of New Challenges for Feature Selection*. 2008. С. 40–105.

17. Ladla L., Deepa T. Feature Selection Methods And Algorithms. *International Journal on Computer Science and Engineering (IJCSE)*. Vol. 3 (5), 2011. С. 1787–1797.

18. Viola P., Jones M. Robust Real-time Object Detection. *International Journal of Computer Vision*. Vol. 57, 2004. С. 137–154.

19. Lucas B. D., Kanade T. An Iterative Image Registration Technique with an Application to Stereo Vision. *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI)*. Vancouver, British Columbia, April 1981. С. 121–130.

20. Aashish K., Vijayalakshmi A. Comparison of Viola-Jones and Kanade-Lucas-Tomasi Face Detection Algorithms. *Oriental Journal of Computer Science and Technology*. Vol. 10, No. (1), March 2017. С. 151–159.

21. Южаков Г.Б. Алгоритм быстрого построения дескрипторов изображения, основанных на технике гистограмм ориентированных градиентов. *Труды МФТИ*. 2012. Т. 5, №3. С. 84–91.

22. Mallick S. Histogram of Oriented Gradients. Learn OpenCV. URL: https://www.learnopencv.com/histogram-of-oriented-gradients/ (дата звернення: 23.06.2020).

23. Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations (ICLR)*. San Diego, CA, USA, May 7–9, 2015. С. 163–171.

24. Hossari M., Dev S., Nicholson M., McCabe K. ADNet: A Deep Network for Detecting Adverts. *AIAI Irish Conference on Artificial Intelligence and Cognitive Science (AICS 2018)*. Dublin, December 6–7th 2018. 9 с.

25. ImageNet. URL: http://image-net.org (дата звернення: 01.07.2020).

26. Neuhold G., Ollmann T., Bulo S. R., Kontschieder P. The Mapillary Vistas Dataset for semantic understanding of street scenes. *International Conference on Computer Vision (ICCV 2017)*. Venice, Italy, 22–29 Oct. 2017. С. 5000–5009.

27. Lin T. Y., Maire M., Belongie S., Hays J., Perona P., Ramanan D., Dollar P., Zitnick C. L. Microsoft COCO: Common Objects in Context. *European*

Conference on Computer Vision (ECCV 2014). April 2014. Vol. 8693. С. 740–755.

28. Szegedy C., Vanhoucke V., Ioffe S., Shlens J., Wojna Z. Rethinking the Inception Architecture for Computer Vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, 27–30 June 2016. С. 2818–2826.

29. Rahmat R. F., Dennis D., Sitompul O. S., Purnamawati S., Budiarto R. Advertisement billboard detection and geotagging system with inductive transfer learning in deep convolutional neural network. *TELKOMNIKA*. October 2019. Vol.17, №5. С. 2659–2666.

30. Krizhevsky A., Sutskever I., Hinton G. E. ImageNet Classification with Deep Convolutional Neural Networks. *Communications of the ACM*. 2017. Vol. 60, No. 6. С. 84–90.

31. Exif Exchangeable Image File Format, Version 2.2. The Library of Congress. URL: https://www.loc.gov/preservation/digital/formats/fdd/fdd000146.shtml (дата звернення: 02.06.2020).

32. AI-based advert creation system for next-generation publicity. ADAPT Centre. URL: https://www.adaptcentre.ie/case-studies/ai-based-advert-creation-system-for-next-generation-publicity (дата звернення: 02.06.2020).

33. Dev S., Hossari M., Nicholson M., McCabe K., Nautiyal A. The ALOS Dataset for Advert Localization in Outdoor Scenes. *Eleventh International Conference on Quality of Multimedia (QoMEX 2019)*. Berlin, Germany, Apr 16, 2019. 3 с.

34. Dev S., Hossari M., Nicholson M., McCabe K., Nautiyal A., Conran C., Tang J., Xu W., Pitié F. The CASE Dataset of Candidate Spaces for Advert Implantation. *Proc. International Conference on Machine Vision Applications (MVA)*. Tokyo, Japan, May 27–31 2019. 4 с.

35. Cordts M., Omran M., Ramos S., Rehfeld T., Enzweiler M., Benenson R., Franke U., Roth S., Schiele B. The Cityscapes Dataset for Semantic

Urban Scene Understanding. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA, 27–30 June 2016. С. 3213–3223.

36. Jiang X.-H., Feng H.-L., Dong Y.-J. Application of Neural Network in Image Detection of Illegal Billboards. *Proceedings of the International Academic Conference on Frontiers in Social Sciences and Management Innovation (IAFSM 2019)*. February 2020. С. 12–16.

37. Ren S., He K., Girshick R., Sun J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015*. Montreal, Quebec, Canada, December 7–12, 2015. С. 91–99.

38. Rao J., Zhang J. Cut and Paste: Generate Artificial Labels for Object Detection. *Proceedings of the International Conference on Video and Image Processing*. Singapore, December 27–29, 2017. С. 29–33. URL: https://dl.acm.org/doi/pdf/10.1145/3177404.3177440 (дата звернення: 10.06.2020).

39. The PASCAL Visual Object Classes Homepage. URL: http://host.robots.ox.ac.uk/pascal/VOC/ (дата звернення: 21.07.2020).

40. LeCun Y. LeNet-5, convolutional neural networks. Yann LeCun. URL: http://yann.lecun.com/exdb/lenet/ (дата звернення: 09.05.2020).

41. Saha S. A Comprehensive Guide to Convolutional Neural Networks. Towards Data Science. URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (дата звернення: 10.05.2020).

42. Powell V. Image Kernels. Setosa. URL: https://setosa.io/ev/image-kernels/ (дата звернення: 10.05.2020).

43. 7 Types of Neural Network Activation Functions: How to Choose? MissingLink.ai. URL: https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/ (дата звернення: 10.05.2020).

44. Wu B., Wan A., Iandola F., Jin P. H., Keutzer K. SqueezeDet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object

Detection for Autonomous Driving. *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Honolulu, HI, 2017. С. 446–454.

45. Luhaniwal V. Forward propagation in neural networks — Simplified math and code version. Towards Data Science. URL: https://towardsdatascience.com/forward-propagation-in-neural-networks-simplified-math-and-code-version-bbcfef6f9250 (дата звернення: 03.07.2020).

46. Li Y., Ma L., Zhong Z., Liu F., Cao D., Li J., Chapman M. A. Deep Learning for LiDAR Point Clouds in Autonomous Driving: A Review. *IEEE Transactions on Neural Networks and Learning Systems*. 2020. С. 1–21.

47. Redmon J., Divvala S. K., Girshick R. B., Farhadi A. You only look once: Unified, real-time object detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA, June 27-30, 2016. С. 779-788.

48. Non-maximum Suppression (NMS). Towards Data Science. URL: https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c (дата звернення: 03.07.2020).

49. Girshick R. B., Donahue J., Darrell T., Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*. Columbus, OH, 2014. С. 580–587.

50. Iandola F. N., Han S., Moskewicz M. W., Ashraf K., W. J.Dally, Keutzer K. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. arXiv. 2016. URL: https://arxiv.org/pdf/1602.07360.pdf (дата звернення: 02.07.2020).

51. Geiger G., Lenz P., Urtasun R. Are we ready for autonomous driving? The KITTI vision benchmark suite. *2012 IEEE Conference on Computer Vision and Pattern Recognition*. Providence, RI, 2012. С. 3354–3361.

52. Liu W., Anguelov D., Erhan D., Szegedy C., Reed S., Fu C.-Y., Berg A. C. SSD: Single Shot MultiBox Detector. *Proceedings of the European Conference on Computer Vision (ECCV 2016)*. Amsterdam, Netherlands, October 11–14, 2016. С. 21–37.

53. Girshick R. B., Donahue J., Darrell T., Malik J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Columbus, OH, USA, June 23–28, 2014. С. 580–587.

54. Region Proposals. Coursera. URL: https://www.coursera.org/lecture/convolutional-neural-networks/optional-region-proposals-aCYZv (дата звернення: 10.08.2020).

55. Howard A. G., Zhu M., Chen B., Kalenichenko D., Wang W., Weyand T., Andreetto M., Adam H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. CoRR. URL: https://arxiv.org/pdf/1704.04861.pdf (дата звернення: 13.08.2020).

56. Bochkovskiy A., Wang C.-Y., Liao H.-Y. M. YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv. URL: https://arxiv.org/pdf/2004.10934.pdf (дата звернення: 02.09.2020).

57. Darknet. GitHub. URL: https://github.com/pjreddie/darknet (дата звернення: 2.09.2020).

58. Liu S., Qi L., Qin H., Shi J., Jia J. Path Aggregation Network for Instance Segmentation. *2018 Conference on Computer Vision and Pattern Recognition. Salt Lake City*, UT, USA, 2018. С. 8759–8768.

59. Chou C., Chien J., Chen H. Self Adversarial Training for Human Pose Estimation. 2018 Asia-Pacific *Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. Honolulu, HI, USA, 2018. С. 17–30.

60. Yolchuyeva S., Németh G., Gyires-Tóth B. Self-Attention Networks for Intent Detection. *Proceedings of Recent Advances in Natural Language Processing*. Varna, Bulgaria, Sep 2–4, 2019. С. 1373–1379.

61. Zhang Z., He T., Zhang H., Zhang Z., Xie J., Li M. Bag of Freebies for Training Object Detection Neural Networks. 2019. URL: https://arxiv.org/pdf/1902.04103.pdf (дата звернення: 12.10.2020).

62. Trivedi S. YOLOv4 — Version 2: Bag of Specials. Medium. URL: https://medium.com/visionwizard/yolov4-version-2-bag-of-specials-fab1032b7fa0 (дата звернення: 12.10.2020).

63. Rothe R. Applying deep learning to real-world problems. Medium. URL: https://medium.com/merantix/applying-deep-learning-to-real-world-problems-ba2d86ac5837 (дата звернення: 10.06.2020).

64. Dieleman S. Classifying plankton with deep neural networks. Sander Dieleman. URL: https://benanne.github.io/2015/03/17/plankton.html (дата звернення: 05.06.2020).

65. National Data Science Bowl. Kaggle. URL: https://www.kaggle.com/c/datasciencebowl (дата звернення: 05.06.2020).

66. Haldar M. How much training data do you need? Medium. URL: https://medium.com/@malay.haldar/how-much-training-data-do-you-need-da8ec091e956 (дата звернення: 12.06.2020).

67. van Smeden M, Moons KG, de Groot JA. Sample size for binary logistic prediction models: Beyond events per variable criteria. *Statistical Methods in Medical Research*. 2019. Vol. 28(8). С. 2455–2474.

68. Scott G. How many images do you need to train a neural network? Pete Warden's blog. URL: https://petewarden.com/2017/12/14/how-many-images-do-you-need-to-train-a-neural-network/ (дата звернення: 12.06.2020).

69. Loke K. S. Object contour completion by combining object recognition and local edge cues. *Journal of ICT*. December 2017. Vol. 16, No. 2. С. 224–242.

70. Seng L. K. Contour-based Shape Recognition using Perceptual Turning Points. *VISAPP 2013 — International Conference on Computer Vision Theory and Applications.* Barcelona, Spain; 21–24 February, 2013. С. 487–491.

71. Canny J. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1986. Vol. PAMI-8, No. 6. С. 679–698.

72. Dorigo M., Stützle T. Ant Colony Optimization. Scituate, MA, United States, 2004. 319 с.

73. Warp an image into a quad region of another image using OpenCV. Stack Overflow. URL: https://stackoverflow.com/questions/34235494/warp-an-image-into-a-quad-region-of-another-image-using-opencv (дата звернення: 12.06.2020).

74. Why TensorFlow. TensorFlow. URL: https://www.tensorflow.org/about (дата звернення: 02.07.2020).

75. Advantages and Disadvantages of TensorFlow. TechVidvan. URL: https://techvidvan.com/tutorials/pros-and-cons-of-tensorflow/ (дата звернення: 18.06.2020).

76. Keras. URL: https://keras.io/ (дата звернення: 02.07.2020).

77. NumPy. URL: https://numpy.org (дата звернення: 02.07.2020).

78. Pedregosa F., Varoquaux G., Gramfort A., Michel V., Thirion B., Grisel O., Blondel M., Prettenhofer P., Weiss R., Dubourg V., Vanderplas J., Passos A., Cournapeau D., Brucher M., Perrot M., Duchesnay E. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. 2011. Vol. 12. С. 2825–2830.

79. Introducing ML.NET: Cross-platform, Proven and Open Source Machine Learning Framework. .NET Blog. URL: https://devblogs.microsoft.com/dotnet/introducing-ml-net-cross-platform-proven-and-open-source-machine-learning-framework/ (дата звернення: 18.10.2020).

80. Ahmed Z., Amizadeh S., Bilenko M., Carr R., Chin W.-S. Machine Learning at Microsoft with ML.NET. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Anchorage, AK, USA, 2019. С. 2448–2458.

81. NimbusML. GitHub. URL: https://github.com/microsoft/NimbusML (дата звернення: 18.10.2020).

82. Kuwahara M., Hachimura K., Eiho S., Kinoshita M. Processing of RI-Angiocardiographic Images. Digital Processing of Biomedical Images. Boston, MA : Springer US, 1976. С. 187–202.

83. Harris corner detector. OpenCV. URL: https://docs.opencv.org/3.4/d4/d7d/tutorial_harris_detector.html (дата звернення: 19.10.2020).

84. Open Images Dataset V6. URL: https://storage.googleapis.com/openimages/web/download.html (дата звернення: 20.10.2020).

85. Papadopoulos D. P., Uijlings J. R. R., Keller F., Ferrari V. Extreme Clicking for Efficient Object Annotation. *Proceedings of the IEEE International Conference on Computer Vision*. Venice, 2017. С. 4940–4949.

86. Papadopoulos D. P., Uijlings J. R. R., Keller F., Ferrari V. We don't need no bounding-boxes: Training object class detectors using only human verification. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Las Vegas, 2016. С. 854–863.

87. Attribution 2.0 Generic (CC BY 2.0). Creative Commons. URL: https://creativecommons.org/licenses/by/2.0/ (дата звернення: 19.10.2020).

88. NEXET — The Largest and Most Diverse Road Dataset in the World. Nexar's Blog — building the world's safe driving network. URL: https://blog.getnexar.com/https-medium-com-itayklein-intro-nexet-50e9b596d0e5 (дата звернення: 20.10.2020).

89. Data format. COCO. URL: https://cocodataset.org/#format-data (дата звернення: 05.07.2020).

90. Kotseruba I., Rasouli A., Tsotsos J. K. Joint Attention in Autonomous Driving (JAAD). Arxiv. URL: https://arxiv.org/pdf/1609.04741.pdf (дата звернення: 03.10.2020).

91. Yolo_mark. GitHub. URL: https://github.com/AlexeyAB/Yolo_mark (дата звернення: 01.10.2020).

Декларація

академічної доброчесності

здобувача ступеня вищої освіти ЗНУ

Я, Іщенко Володимир Юрійович , студент 2 курсу, форми навчання денної, Інженерного навчально-наукового інституту, спеціальність 121 Інженерія програмного забезпечення, адреса електронної пошти sp115-16@stu.zsea.edu.ua, — підтверджую, що написана мною кваліфікаційна робота на тему **«Комп'ютерна система для виділення об'єктів у відеопотоці із заміщенням вмісту»** відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст.42 Закону України «Про освіту», зі змістом яких ознайомлений.

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

згоден/згодна на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-системи, а також на архівування моєї роботи в базі даних цієї системи.

Дата 30.11.2020 Підпис _____ Іщенко Володимир Юрійович

(студент)

Дата 30.11.2020 Підпис _____ Безверхий Анатолій Ігорович

(науковий керівник)