Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Факультету інформатики

Кафедра інформатики

Магістерська робота

освітній ступінь – магістр

на тему: «**ВИКОРИСТАННЯ НЕЙРОННИХ МЕРЕЖ З НАВЧАННЯМ З НУЛЯ ДЛЯ ЗНАХОДЖЕННЯ ОБ'ЄКТІВ НА ЗОБРАЖЕННЯХ З ДРОНІВ**»

Виконав студент 2-го року навчання,

Спеціальності
121 Інженерія програмного забезпечення

Ткаленко Владислав Геннадійович

Керівник Бучко О.А.

кандидат технічних наук, доцент

Рецензент_____
Магістерська робота захищена з оцінкою_____

Секретар ЕК_____
«___» _____ 2024 р.

Київ 2024

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ
Зав. кафедри інформатики,
к.ф-м.н.
_____ С. С. Гороховський
(підпис)
7 листопада 2024 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на дипломну роботу

студенту 2 року ІПЗ факультету інформатики
Ткаленку Владиславу Геннадійовичу

ТЕМА: Використання нейронних мереж з навчанням з нуля для знаходження об’єктів на зображеннях з дронів

ЗАДАЧА: Створити систему детекції зображень отриманих з дронів використовуючи нейроні мережі з навчанням з нуля.

Зміст ТЧ до магістерської роботи:

Зміст
Вступ
1 Обробка природньої мови
2 Теоретичні аспекти методів розпізнавання об’єктів
3 Застосування методу навчання з нуля
4 Розробка та реалізація моделі з використанням методу навчання з нуля
Висновки
Список літератури

Дата видачі „ ” _____2024 р. Керівник _____
(підпис)
Завдання отримав _____
(підпис)

Календарний план виконання роботи

Тема: Використання нейронних мереж з навчанням з нуля для знаходження об’єктів на зображеннях з дронів

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на дипломну роботу.	7.11.2024	
2.	Огляд технічної літератури за темою роботи.	01.12.2024	
3.	Визначення структури практичної роботи та написання мінімального прототипу	25.01.2024	
4.	Пошук та дослідження методів покращення алгоритму, вивчення альтернативних алгоритмів.	07.02.2024	
5.	Підготовка реалізацій до тестування	01.03.2024	
6.	Виконання порівняльного аналізу результатів роботи алгоритмів з різними параметрами застосування.	30.03.2024	
7.	Написання пояснювальної роботи.	20.04.2024	
8.	Аналіз отриманих результатів	22.05.2024	
9.	Створення слайдів для доповіді та написання доповіді.	30.05.2024	
10.	Захист магістерської роботи (проекту)	06.2024	

Студент: Ткаленко Владислав Геннадійович

Керівник: Бучко Олена Андріївна доцент, к.т.н.

“_____”_____

ЗМІСТ

Аннотація

Робота присвячена дослідженню моделей, побудованих на основі методів навчання з нуля. Проведено аналіз існуючих рішень для розпізнавання об’єктів на зображеннях. Запропоновано новий підхід для покращення результатів методів навчання з нуля на зображеннях з дронів. Результатом роботи є модель, яка поєднує архітектуру навчання з нуля та модулі для кращого розпізнавання об’єктів на зображеннях з дронів. Було проведене перше тренування даної моделі, яке не продемонструвало очікуваних результатів. Для кращих експериментальних результатів необхідно покращити процес тренування моделі, та провести тренування з потужним аппаратним забезпеченням.

Ключові слова:

Навчання з нуля, дрони, розпізнавання об’єктів, нейронні мережі, комп’ютерний зір, обробка природньої мови.

ВСТУП

Актуальність. Використання дронів в останні роки набуло масового характеру в нашій країні. Вже зараз очевидно, що бойові дрони змінили ведення сучасної війни, а десятки команд намагаються першими створити дрони з комп'ютерним зором. Водночас дрони продовжують використовуватися в інших різноманітних сферах: для моніторингу поверхні, розвідки, рятувальних операцій, доставки вантажів, тощо. У багатьох з цих задач важливим є здатність дрона самому знаходити об'єкти на поверхні, та попереджати про свою знахідку оператора. Особливо важливим дана можливість є в задачах моніторингу поверхні та для бойових завдань.

Стандартним способом навчити дрон знаходити потрібні нам об'єкти є метод «контрольованого навчання» (Supervised Learning) при якому для кожного типу об'єкту, який треба знаходити, збирається великий набір даних, на якому навчається модель. Далі навчена модель використовується на дронах для пошуку об'єктів на поверхні. Цей метод має очевидні недоліки, пов'язані з збором даних. По-перше, збір даних є часозатратним процесом; по-друге, іноді цей збір є неможливим через відсутність необхідних даних. До того ж «контрольоване навчання» не є гнучким підходом, адже знаходження кожного нового класу буде потребувати нового довготривалого збору даних.

Альтернативою «контрольованого навчання» є «навчання з нуля» (Zero-Shot Learning) - методи навчання моделей без необхідності збору великих об'ємів даних для кожного класу. Цей метод є перспективним, і вже показує непогані результати на загальних відкритих наборах даних, таких як ImageNet, COCO, інші. Але моделі, навчені з використанням «навчання з нуля» поки що не показують гарного результату на специфічних наборах даних які використовуються у вузьких сферах.

Набори даних з дронів, безперечно, є специфічними. Об'єкти на таких фото відрізняються малими розмірами, великою часткою перекриття іншими об'єктами, та можливими перепадами розмірів в межах одного фото (Рисунок 0). Це зумовлює

погані результати роботи «навчання з нуля» моделей на таких даних, що потребує подальших досліджень у цій сфері.

Рисунок 0: Роміри транспорту на даному зображенні змінюється суттєво з віддаленням від камери. Зображення з датасету VisDrone [1].

Мета дослідження. Дослідити можливий варіант покращення роботи існуючих «навчання з нуля» моделей на зображеннях з дронів з використанням нових методів детекції об’єктів, сформованих спеціально для покращення результатів на зображеннях з дронів.

Завдання дослідження. Проаналізувати існуючі наборах даних дронів. Порівняти існуючі «навчання з нуля» моделі та перевірити їх роботу на зображеннях з дронів. Реалізувати методи покращення детекції об’єктів у «навчання з нуля» моделях. Порівняти ефективність роботи «навчання з нуля» моделі зі змінами та без змін. Проаналізувати ефективність запропонованого методу та запропонувати можливі шляхи покращення результату.

Об’єкт дослідження. Інструменти обробки зображень та текстових даних. Концепція Навчання з нуля (Zero-shot learning). Сучасні «навчання з нуля» моделі детекції. Особливості обробки зображень з дронів.

Предмет дослідження. Можливість адаптації існуючих «навчання з нуля» моделей детекції для їх кращої роботи на наборах даних з дронів.

Джерела дослідження. Тематичні статті та дослідження в електронному форматі, документація відповідних фреймворків та інструментів обробки даних.

Наукова новизна одержаних результатів полягає у створені гібридної моделі, яка працює за принципами навчання з нуля, але містить в собі модулі, розроблені для кращого розпізнавання об’єктів на зображеннях з дронів.

Практичне значення одержаних результатів. Запропонована модель після успішних тренувань може використовуватись на розпізнавання різних об’єктів з дронів без нагальної необхідності збору нових даних.

Робота складається з чотирьох розділів.

У першому розділі роботи надано опис деяких розділів обробки природньої мови, необхідних для розуміння сучасних «навчання з нуля» моделей. Розглянуто методи обробки текстових даних, класичні архітектури нейронних мереж та трансформери.

У другому розділі роботи надано опис деяких розділів Комп’ютерного Зору, необхідних для розуміння сучасних «навчання з нуля» моделей. Розглянуто методи обробки зображень, класичні архітектури нейронних мереж та візуальні трансформери.

Третій розділ присвячений опису «навчання з нуля» методів. Розглянуто існуючи напрямки, приділено увагу принципам роботи сучасних «навчання з нуля» моделей класифікації (CLIP) та детекції (OWLv2, Ground Dino).

У четвертому розділі наведено опис набору даних та методів обчислення

результатів експериментів. Наведено результати роботи «навчання з нуля» моделей. Надано опис гібридної архітектури на основі OWLv2 та представлена її реалізація. Виконане тестове тренування даної моделі.

РОЗДІЛ 1 ОБРОБКА ПРИРОДНЬОЇ МОВИ

1.1 Загальний опис

Обробка природньої мови (Natural Language Processing, надалі NLP) - комп’ютерна наука, яка поєднує обчислювальну лінгвістику - моделювання людської мови на основі правил - зі статистичними моделями та моделями машинного навчання, що дозволяє комп'ютерам і цифровим пристроям розпізнавати, розуміти і генерувати текст і мовлення [2]. NLP вирішує різноманітні задачі:

- автоматичний переклад з одної мови в іншу
- автоматична генерація субтитрів до відео
- чат-боти на основі Великих Текстових Моделей (LLM)
- автодоповнення тексту
- семантичний пошук інформації

Окремими напрямками є застосування NLP разом з іншими сферами машинного навчання:

- генерація зображення за текстовим описом (text-to-image)
- отримання опису довільного зображення (image-to-text)
- штучне озвучування довільного тексту (text-to-speech)

На момент написання даної роботи, NLP є найдинамічнішою сферою штучного інтелекту. Поява великих текстових моделей в останні п’ять років відкрило нові можливості розв’язання NLP-задач. Після появи цих моделей виник новий напрямок - оперативна інженерія (Prompt Engineering) - це практика проектування оптимальних запитів до моделей для отримання найкращого результату. В циклі Гартнера (американської дослідницької і консалтингової компанії) для штучного інтелекту за 2023 рік, оперативна інженерія знаходиться на піку популярності на найближчі 2-5 років (Рисунок 1.1).

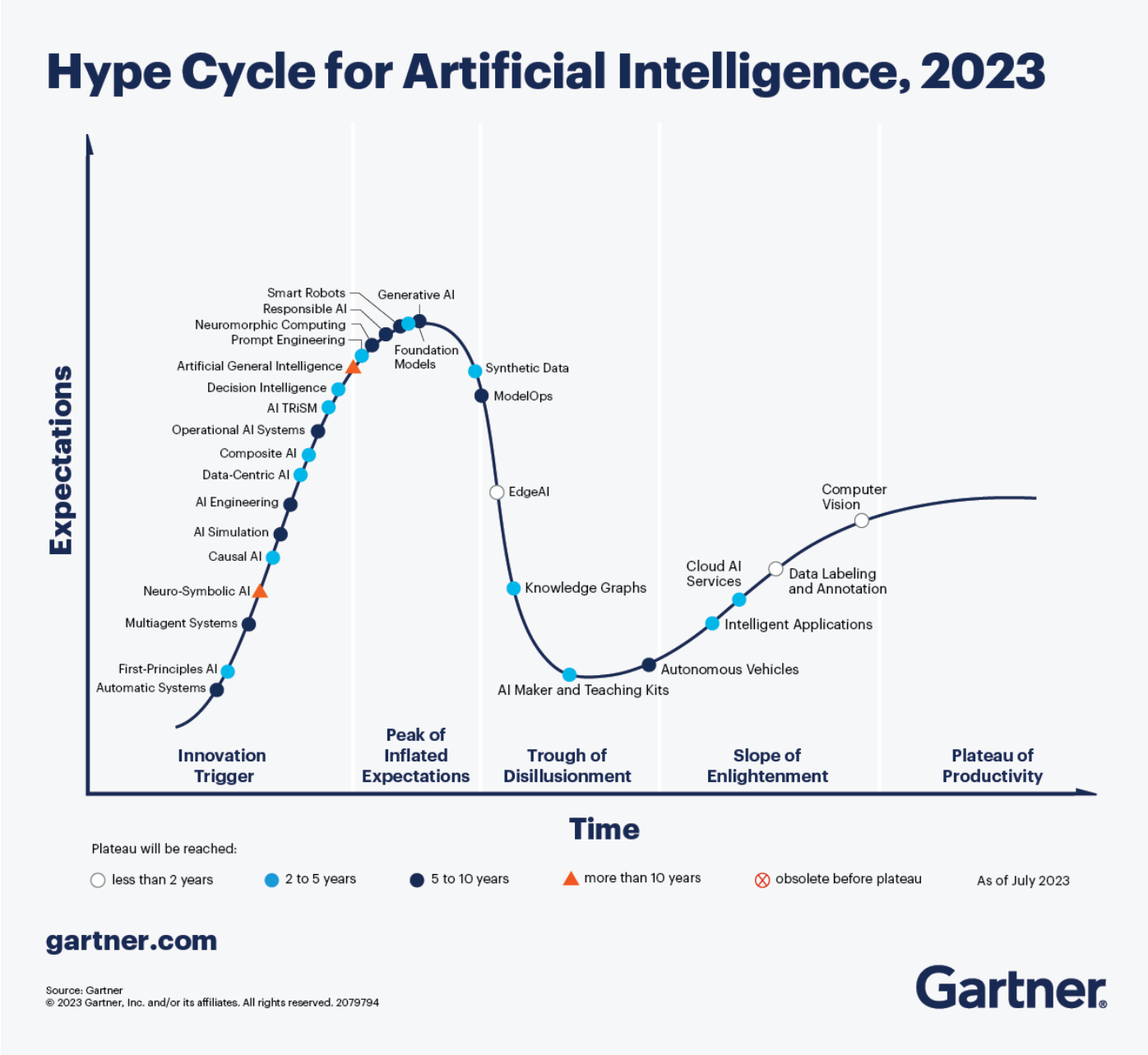


Рисунок 1.1 - Цикл Гартнера на 2023 рік для Штучного Інтелекту [3].

1.2 Методи попередньої обробки тексту

При виконанні задач NLP, найпершим етапом виступає обробка тексту. В цьому підрозділі коротко описані можливі етапи даної обробки.

1.2.1 Токенізація

Токенізація (Tokenization) - це процес визначення токенів, або тих базових одиниць, які не потрібно розкладати в подальшій обробці [4]. Після токенізації тексту ми отримуємо послідовність токенів, які далі йдуть на наступні етапи обробки. Найпростішою токенізацією є утворення токенів зі слів - наборів символів, обмежених пробілами. Однак є багато інших методів токенізації, які можуть відокремлювати в токени розділові знаки, частини речення, словосполучення, фразеологізми, або робити токени за правилами, які надав користувач.

1.2.2 Стемінг та Лематизація

Лематизація - процес зведення токенів до їхніх кореневих форм. Такі кореневі форми називаються лемами (Рисунок 1.2). Іноді це може бути корисним, щоб зберегти розмірність векторного представлення низькою. Для отримання лем використовуються вже готові словники, та морфологічний розбір слів [5].

```
he --> he
was --> be
running --> run
late --> late
```

Рисунок 1.2: Приклад Лематизації англомовних слів [5].

Стемінг - ще один процес зведення токенів до лем, але за допомогою значно грубішого методу - обрізання закінчень токенів. Часто це не дає гарного результату. Наприклад, стемінг не зміг би корректно обробити слово was, як на Рисунку 1.2.

1.2.3 N-грами

N-грами - це послідовності токенів фіксованої довжини N, які зустрічаються в реченні. Відповідно до довжини бувають Біграми, Триграми і т.д. Утворити N-грами дуже просто: треба отримати усі послідовності заданої довжини в тексті (Рисунок 1.3).

Input[0]

```
def n_grams(text, n):
    '''
    takes tokens or text, returns a list of n-grams
    '''
    return [text[i:i+n] for i in range(len(text)-n+1)]

cleaned = ['mary', ',', "n't", 'slap', 'green', 'witch', '.']
print(n_grams(cleaned, 3))
```

Output[0]

```
[['mary', ',', "n't"],
 [',', "n't", 'slap'],
 ["n't", 'slap', 'green'],
 ['slap', 'green', 'witch'],
 ['green', 'witch', '.']]
```

Рисунок 1.3: Приклад утворення N-грам [5].

1.2.4 Стоп-слова

Стоп-словами називають токени, які існують в речення задля граматичної правильності. Зазвичай це артиклі та прийменники: “a”, “the”, “is”, “are”, та інші. Частиною попередньої обробки тексту може бути видалення стоп-слів, оскільки вони не несуть за собою контексту. Сучасні програмні бібліотеки для роботи з текстом (nltk, word2vec) містять список стоп-слів, що дозволяє зручно вилучати їх з тексту.

1.2.5 Дистрибутивні репрезентації токенів

Існує декілька можливих варіантів представлення токенів у числовому вигляді, такі як «одногарячий» (one-hot) вектор, TF або TF-IDF вектор. Вони носять загальну назву дистрибутивні репрезентації [5], оскільки носять інформацію про кількість токенів у тексті

«Одногарячий» вектор представляє собою нульовий вектор, з одиницею, яка представляє наявність слова в тексті (Рисунок 1.4).

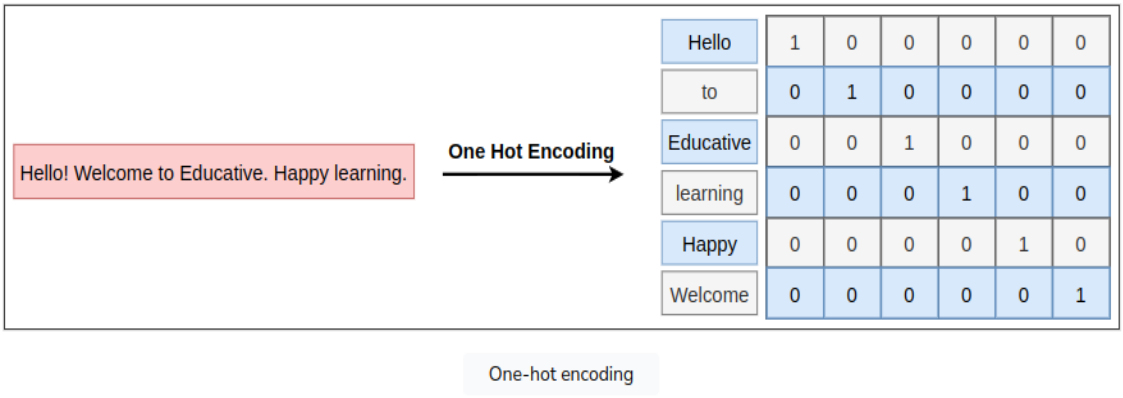


Рисунок 1.4: Утворення «одногарячих» (one-hot) векторів для токенів з тексту [6].

TF-вектор (Term-Frequency) для певного тексту є сумою усіх «одногарячих» репрезентацій окремих токенів [5]. Наприклад для тексту на Рисунку 1.4, TF-вектор буде виглядати як [1,1,1,1,1,1], оскільки кожен токен в ньому зустрічається по одному разу.

TF-IDF вектор (Term-Frequency-Inverce-Document-Frequency) модифікує TF вектор. Ідея полягає у тому, що токени, які часто трапляються в реченні, зазвичай

несуть менше важливої інформації про контекст ніж специфічні токени, які зустрічаються рідше. Щоб вирівняти репрезентації частіших та рідших токенів, значення TF векторів множаться на IDF-коефіцієнт (Рисунок 1.5):

$$IDF(w) = \log \frac{N}{n_w}$$

Рисунок 1.5: формула IDF-коефіцієнту [5].

nw - кількість документів, у яких з’являлося токен w, N - кількість усіх документів. Відповідно якщо токен зустрічається дуже часто (nw = N), тоді логарифм дорівнює нулю, тобто даний токен повністю зануляється. Якщо ж токен зустрічається тільки в одному документі, тоді коефіцієнт буде мати максимально можливе значення, logN [5].

1.2.6 Векторна репрезентація токена

Дистрибутивні методи репрезентації токенів мають суттєві недоліки, найголовніший з яких є їх розміри. Оскільки кожен «одногарячий» вектор має довжину, яка дорівнює кількості унікальних токенів в тексті, в багатьох практичних сценаріях його довжина повинна приблизно дорівнювати розміру словника [5]. До того ж, «одногарячий» вектор не містить нічого про зміст закодованого в ньому токена. Тому для сучасних моделей токени перетворюють у векторну репрезентацію.

Векторна репрезентація токена (ембединг), яка може бути різної довжини, залежно від способу утворення. Таким чином, кожен токен отримує своє представлення у векторному просторі, і їх можна порівнювати, використовуючи методи порівняння векторів (Рисунок 1.6):

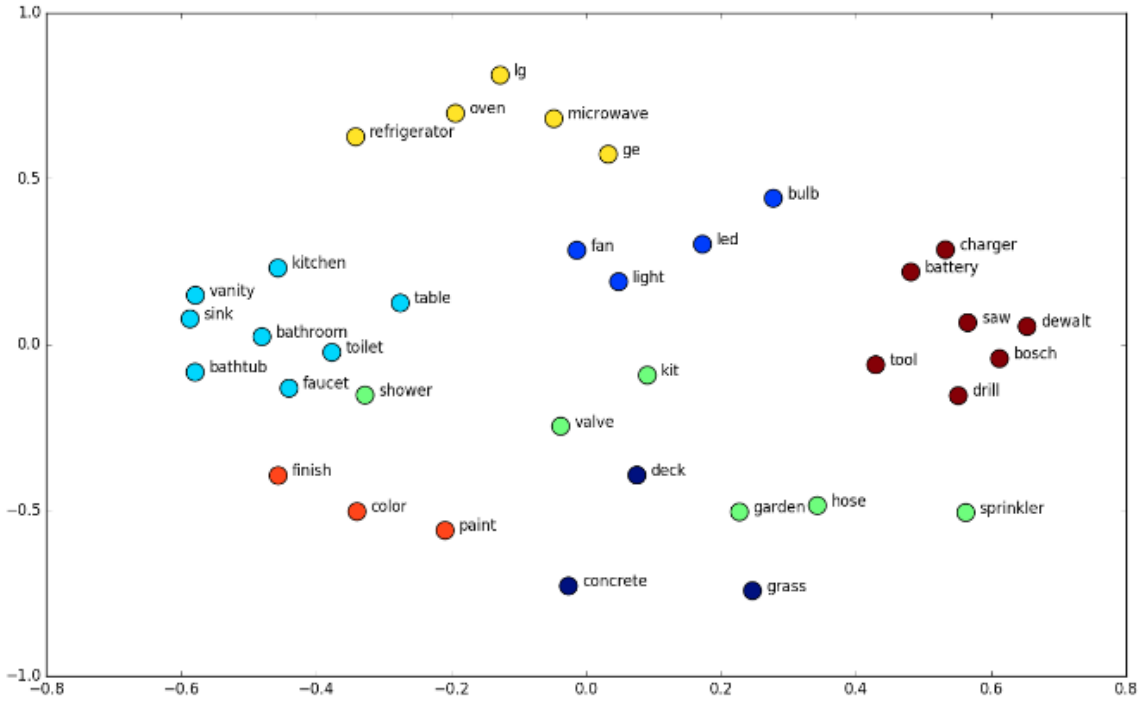


Рисунок 1.6: Візуалізація векторних репрезентацій слів векторному просторі [7].

Векторні репрезентації утворюються за допомогою натренованих нейронних мереж. Існують різні архітектури для створення векторних репрезентацій, які використовували для тренування великі набори текстових даних з інтернету. Найвідомішими з них є Word2Vec, CBOW, Continuous skip-gram model, ELMo [7].

Наразі представлення тексту у вигляді векторів стало основним для тренувань сучасних моделей Обробки Природньої Мови, завдяки набагато меншій розмірності таких репрезентацій у порівнянні з дистрибутивними репрезентаціями (наприклад, вектор ELMo має довжину 1024 [8]) та зберігання інформації про значення закодованих токенів.

1.3 Опис рекурентних моделей NLP.

Рекурентні нейронні мережі - тип нейронних мереж, які використовуються для знаходження зв’язків та шаблонів (патернів) у послідовних наборах даних. Тому ці моделі стали широко використовуватися саме для обробки тексту, який представляє собою послідовність символів. Головною відмінністю рекурентних мереж від звичайних є те, що в рекурентних мережах рух даних проходить через цикли, які повертають дані заново в мережу [9]. Це дозволяє їм розширити функціональність звичайних нейронних мереж, використовуючи для обрахунків не тільки вхідні дані, а й дані з попередніх циклів (Рисунок 1.7).

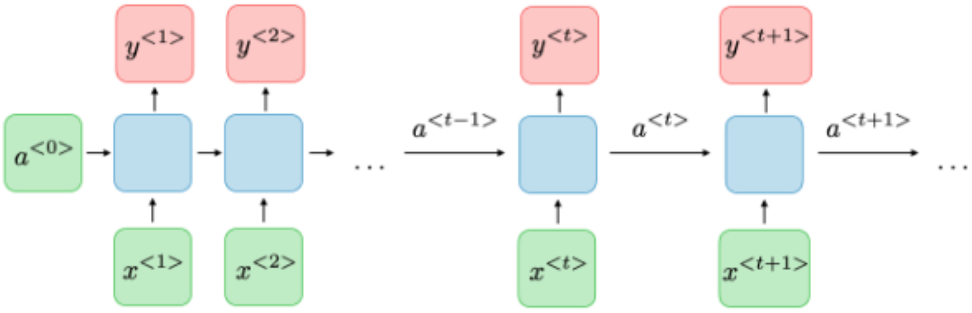


Рисунок 1.7: Схематичне представлення базової рекурентної мережі [10].

Існують різні архітектури рекурентних мереж, в залежності від кількості вхідних та вихідних даних. One-to-many (один-до-багатьох) - використовується в задачі генерації музики. Many-to-one (багато-до-одного) - для класифікації настрою тексту (sentiment classification). Many-to-many (багато-до-багатьох) - для розпізнавання об’єктів в тексті та машинного перекладу (Рисунок 1.8).

Глибока рекурентна нейронна мережа (Deep RNN) - модель, яка утворюється накладанням кількох рекурентних мереж одна на одну [9]. Таким чином, усі вихідні дані мережі попереднього рівня, є вхідними даними для мережі наступного рівня (Рисунок 1.9). Використання багатьох рівнів рекурентних нейронних мереж дозволяє таким моделям вивчати більш глибокі та комплексні характеристики з даних.

Двонаправлена рекурентна нейронна мережа (Bidirectional RNN) - різновид архітектури, яка складається з двох рекурентних мереж. Але на відміну від глибокої

рекурентної мережі, ці два рівня розташовані не послідовно, а паралельно одне до одного. До того ж, у другій мережі інформація оброблюється в протилежному напрямку - з кінцевого входу до початкового (Рисунок 1.9). Завдяки тому, що така модель оброблює інформацію одночасно в двох напрямках, Двонаправлені рекурентні мережі використовуються в задачах, де необхідно використовувати інформацію, що наявна і перед елементом послідовності, і після нього одночасно. Наприклад, для успішного заповнення загублених слів у тексті, нам необхідно проаналізувати текст перед та після загубленого слова. Для такої задачі ідеально підходить Двонаправлена рекурентна мережа [9].

Серйозною проблемою рекурентних мереж було затухання градієнта. При зворотньому проходженні градієнт зменшувався після кожного циклу, і затухав не дійшовши до кінця [9]. Це призводило до того, що рекурентні мережі показували набагато гірший результат при збільшенні розмірів вхідного тексту. Для вирішення цієї проблеми було запропоновано LSTM-блок.

LSTM-блок (Long-Short-Term-Memory) містить так звані шлюзи (gates), які контролюють рух інформації між блоками. Також з’вляється поняття клітинна пам’ять (memory cells) - довгострокова пам’ять, яка передається між блоками. Окрім цього, до наступного блоку передається прихований стан (hidden state) - це результат обчислень попереднього блока.

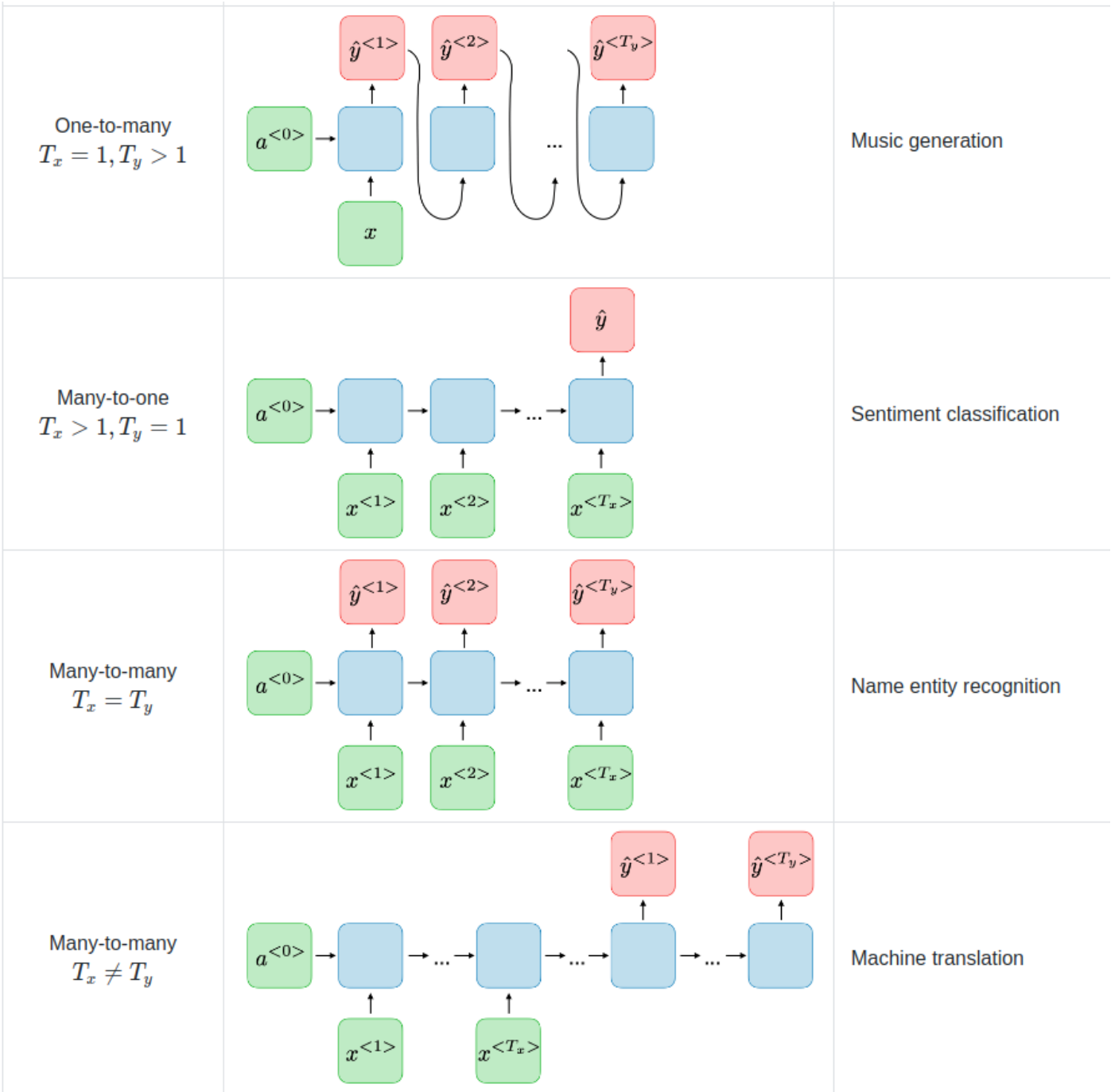


Рисунок 1.8: Типи архітектур рекурентних нейронних мереж [10].

Існують три типи шлюзів: вхідний шлюз (Input Gate) - контролює скільки нової інформації треба записати до клітинної пам’яті; шлюз забуття (Forget Gate) - контролює скільки інформації з клітинної пам’яті треба забути; Вихідний Шлюз (Output Gate) - контролює скільки інформації буде передано у наступний блок у вигляді прихованого стану [9]. Ці три шлюзи представляють з себе матриці, які модель вчиться обраховувати під час тренування. Схема роботи LSTM наведена на

Рисунку 1.10.

Дана архітектура дозволила суттєво покращити результати Рекурентних Мереж, оскільки тепер інформація могла передаватися через більшу кількість блоків.

Не дивлячись на те, що архітектура Рекурентних Мереж робила їх чудовим вибором для обробки текстових даних, вона була також причиною серйозних перешкод [11]:

1. Послідовна сутність архітектури не дозволяла обчислювати рекурентні мережі паралельно.
2. Не дивлячись на часткове вирішення проблеми затухання градієнту з появою LSTM, повністю ця проблема нікуди не зникла.
3. Ці проблеми мотивували продовжувати пошуки нової архітектури на заміну рекурентним мережам.

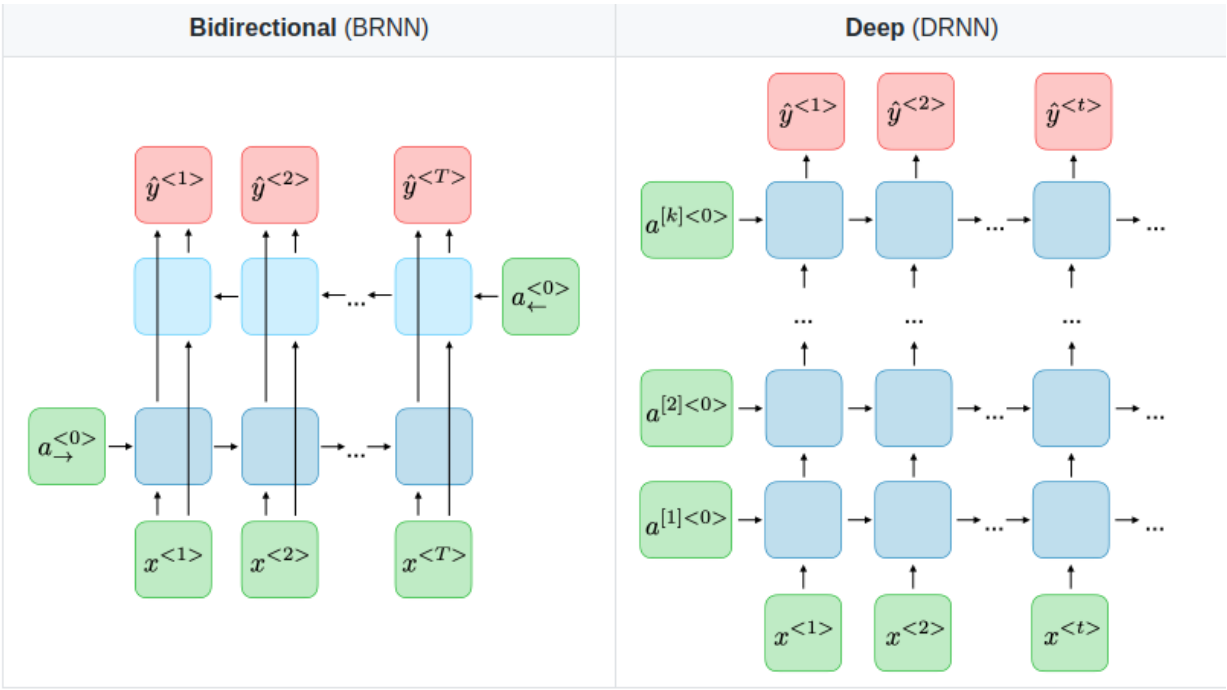


Рисунок 1.9: Схема двонаправленої рекурентної мережі (ліворуч) та глибокої рекурентної мережі (праворуч) [10].

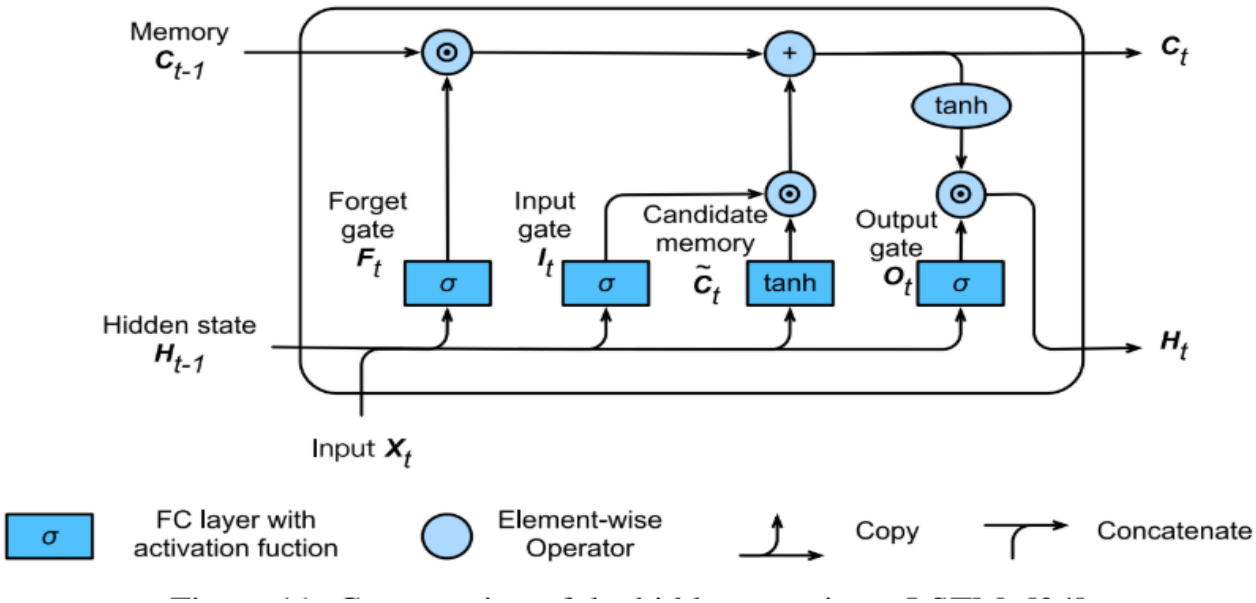


Рисунок 1.10: Схема роботи блоку LSTM [9].

1.4 Трансформери

Щоб обійти недоліки рекурентних нейронних мереж, 2017 року було запропоновано нову архітектуру моделей під назвою трансформер. На Рисунку 1.11 показаний варіант трансформера, взятий зі статті “Attention Is All You Need” [12].

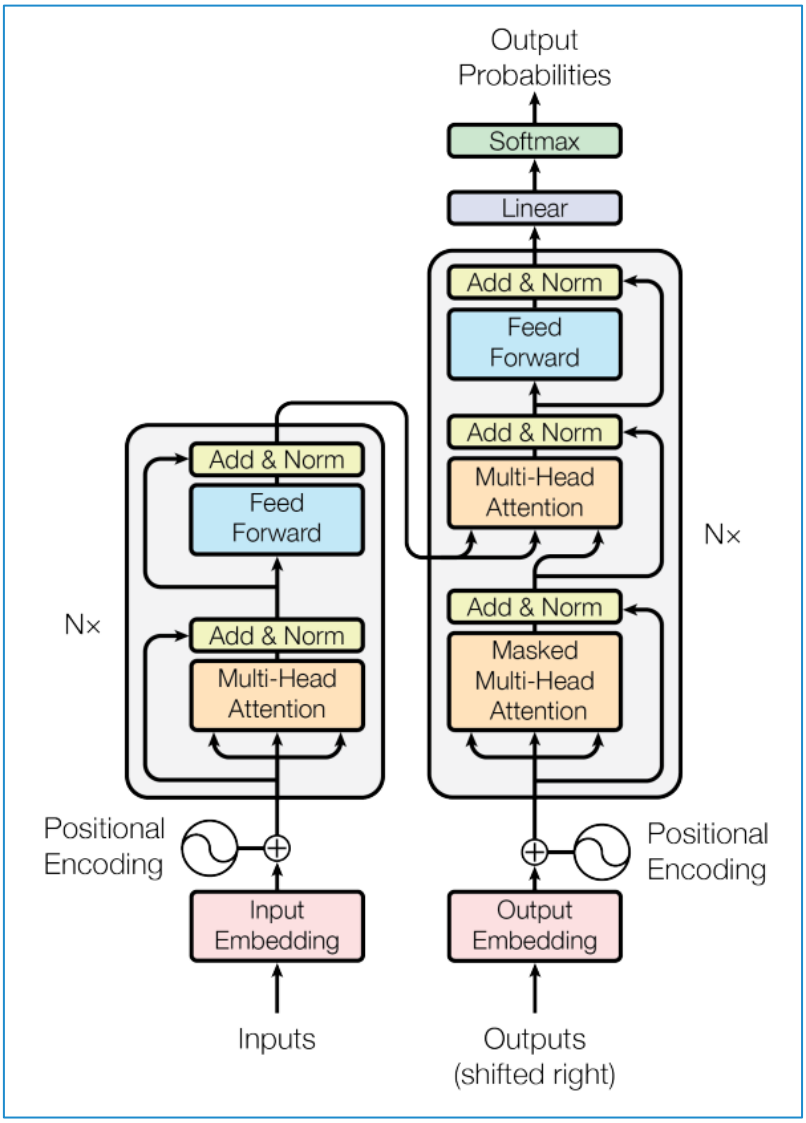


Рисунок 1.11: Модель оригінального трансформера [12].

Трансформери зазвичай складаються з трьох основних частин: кодувальник (енкодер), декодувальник (декодер) та механізму уваги (Self-Attention) :

- кодувальник - частина трансформера, яка приймає вхідні токени та повертає відповідні їм векторні репрезентації (ембединги). Послідовність токенів подається до кодувальника повністю, за один раз.
- декодувальник - частина трансформера, яка приймає вхідний токен та на його основі генерує наступний токен. Головною відмінністю декодувальника від кодувальника є те, що декодувальник приймає на вхід по одному токену за раз, а не повну послідовність, як у кодувальника. Інформація

про вхідний токен, а також токени до нього, використовуються для генерації наступного токена, який далі подається на вхід при наступній ітерації.

- механізм уваги (Self-Attention mechanism) - процес формування векторних репрезентації для набору токенів таким чином, щоб вихідний вектор зберігав інформацію не тільки про самі токени, а й про контекст, в якому вони застосовувалися. Головна ідея полягає у тому, що один і той самий токен може мати різні значення в залежності від контексту, тобто від токенів, які його оточують. Механізм уваги дозволяє закодувати разом з самими токенами також залежності між токенами в тексті, таким чином зберігаючи контекст.

Варто зазначити, що трансформери не обов’язково можуть мати і кодувальник, і декодувальник одночасно. Це залежить від того, для якої саме задачі трансформер розробляється. Залежно від навності кодувальника/декодувальника, трансформери поділяються на: [13]

- Auto-encoding models - наявний лише кодувальник. Трансформери такої архітектури повертають на вихід векторні репрезентації вхідних токенів, які можна далі використовувати для різних задач класифікації. Відомі моделі: ALBERT, BERT, DistilBERT, ELECTRA, RoBERTa [14]. Задачі, для яких підходять трансформери даної архітектури: класифікація речень, розпізнавання іменованих об'єктів, пошук відповіді на питання в тексті.
- Auto-regressive models - наявний лише декодувальник. Трансформери такої архітектури утворюють наступний токен на основі попереднього, продовжуючи працювати рекурсивно. Через це така архітектура чудово підходить для генерації тексту. Відомі моделі: CTRL, GPT, GPT-2, Transformer XL [15]. Задачі, для яких підходять трансформери даної архітектури: генерація тексту.
- Sequence-to-Sequence models - наявні кодувальник і декодувальник. У даній моделі декодувальник генерує наступні токени використовуючи інформацію про попередні токени та інформацію, отриману з кодувальника. До

того ж, енкодер і декодер є незалежними один від одного, мають окремі ваги і можуть бути замінними. Наприклад, можна до кодувальника BERT додати один тип декодувальника, який навчений для перекладу тексту на Французьку. Він, використовуючи дані з кодувальника, буде генерувати переклад початкового тексту. Або ж до того самого кодувальника можна поставити інший декодувальник, навчений для витягання ключової інформації з тексту. Тоді модель замість перекладу буде надавати стиснуту вижимку основної інформації. Відомі моделі: BART, mBART, Marian, T5 [16]. Задачі, для яких підходять трансформери даної архітектури: витягування ключової інформації, переклад, генерація відповіді на запитання.

З часу своєї появи трансформери показали набагато кращі результати у різноманітних задачах обробки природньої мови, ніж рекурентні мережі. Замінивши їх, трансформери стали стандартною архітектурою для роботи з текстовими даними. Але вони також мають свої недоліки. По-перше, розвиток цієї архітектури швидко звівся до збільшення об’ємів моделей і кількості параметрів. В результаті, розміри трансформерів почали збільшуватися експоненційно: якщо GPT 2018 року мала близько 110 мільйонів параметрів, то GPT-4 2023 року вже налічує від 1.5 до 1.8 трильйонів параметрів [17]. Тому навчання сучасних трансформерів є дуже дорогим та часозатратним процесом, а також шкідливим для екології (Рисунок 1.2):

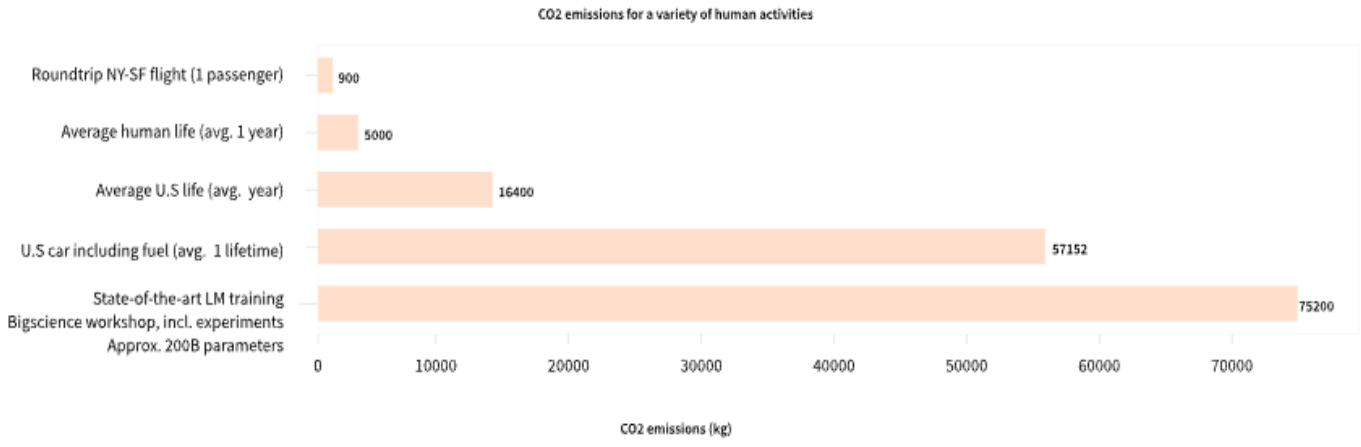


Рисунок 1.12: Вуглецевий слід від тренування Трансформера у порівнянні з

використанням різних видів транспорту та життям людини [13].

По-друге, сучасні трансформери навчаються на великих об’ємах тексту з інтернету, і як недолік цього підходу, можуть генерувати неетичні вислови, сексистський, расистський або гомофобний контент [18].

РОЗДІЛ 2 ТЕОРЕТИЧНІ АСПЕКТИ МЕТОДІВ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ

2.1 Загальний опис

Комп'ютерний зір (Computer Vision, CV) - це галузь штучного інтелекту (ШІ), яка використовує машинне навчання та нейронні мережі, щоб навчити комп'ютери та системи отримувати значущу інформацію з цифрових зображень, відео та інших візуальних даних [19].

Комп'ютерний зір займається вирішенням багатьох проблем, основними з яких є такі [19]:

- **Задача класифікації:** модель повинна точно передбачити до якого класу відноситься зображення. Наприклад класифікувати яка модель автомобіля на зображенні; дати відповідь де зроблене зображення: в приміщенні чи надворі; класифікувати чи зображення має оригінальне відношення сторін, чи воно є стиснутим/розтянутим, та інше.
- **Задача Детекції:** модель повинна виявити місцезнаходження об'єктів на картинці, вказавши не тільки їх клас, а й намалювати бокс навколо об'єкта, повернувши його координати.
- **Задача Сегментації:** модель сегментує зображення на піксельному рівні, назначаючи кожному пікселю відповідний клас. Сегментація може бути семантичною (Semantic segmentation) - коли все зображення розбивається на кілька загальних класів, об'єктною (Instance segmentation) - коли сегментація відбувається навколо конкретних об'єктів, та панорамною (Panoptic segmentation) - об'єднання семантичної та об'єктної сегментації [20]. (Рисунок 2.1)

- **Відстежування об’єкта:** модель повинна відстежувати знаходження об’єкта на відео. Від детекції дана задача відрізняється тим, що потрібно не тільки знаходити об’єкт на кожному кадрі відео, але ще й розуміти що це один і той самий об’єкт.

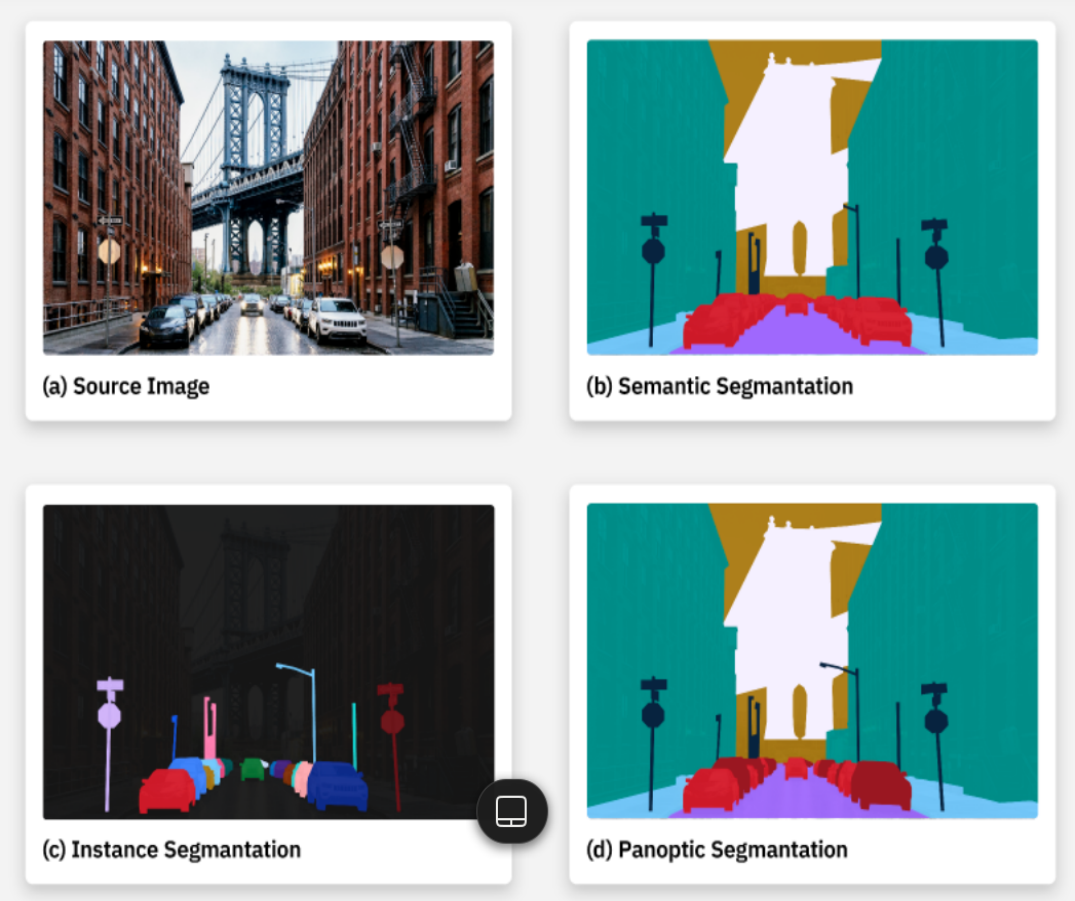


Рисунок 2.1: Порівняння різних видів сегментації [20].

Окремо виділяється комп’ютерний зір на 3D-зображеннях (3D Computer Vision). Задачі детекції та сегментації актуальні і для цих зобарежнь, але вже в 3D просторі [21].

2.2 Методи попередньої обробки зображень.

Попередня обробка зображень виконується перед тим як ці зображення потрапляють до моделей комп’ютерного зору. Правильна передобробка робить роботу моделей більш ефективною.

Існує дуже багато методів попередньої обробки зображень [22, 23]. До початку використання нейронних мереж, попереднью обробкою вважався процес отримання інформації з зображень для моделей комп’ютерного зору, в тому числі методи сегментаці та витягування атрибутів із зображень аж до розпізнавання окремих об'єктів [22]. У сучасних моделях Комп’ютерного Зору ці задачі виконують нейронні мережі, тому в даному розділі буде наведено тільки ті методи попередньої обробки, які зазвичай не входять у модель [24]:

1. **Зміна розміру зображення (Image Resizing).** Зазвичай різні зображення в датасетах мають різну роздільну здатність, співвідношення сторін та орієнтацію, що може вплинути на точність моделі. Зміна розміру зображень робиться для приведення зображень до однакового розміру, а також для зменшення обчислювальних витрат, оскільки зменшує кількість пікселів для обробки. Однак зміна розміру може мати й негативні наслідки, такі як спотворення карт ознак зображення, або втрати деталей. Існують різні методи зміни розміру зображень: метод найближчих сусідів, білінійний метод, бікубічний метод та інші.

2. **Нормування зображення (Image Normalizing)**. Цей метод являє собою процес масштабування значень пікселів зображення до певного діапазону, наприклад, [0, 1] або [-1, 1]. Нормування зображення може допомогти поліпшити стабільність і збіжність моделі, оскільки вона зменшує дисперсію і асиметрію розподілу даних. Нормування зображення також може підвищити контрастність і яскравість зображення, що полегшує виявлення об'єктів і країв моделі.

3. **Аугментації зображень (Image Augmentation).** Ця техніка включає в себе набір різних методів випадкової трансформації зображень,

таких як обертання, масштабування, перевертання, зсув, додавання шумів та інше. Додавання аугментацій дозволяє штучно збільшити різноманітність даних, що підвищує здатність моделі до генералізації. Також аугментації можуть симулювати різні сценарії з реального світу, наприклад перекриття, засвітлення, розмитість та інші.

2.3 Опис моделей комп'ютерного зору

В даному підрозділі опишемо кілька популярних архітектур комп'ютерного зору, які у свій час ставали важливою віхою у розвитку сфери. Детально зосереджуватись на описі кожної архітектури не будемо, обмежившись лише виділенням ключових особливостей кожної.

2.3.1 Згорткові нейронні мережі

Вважається, що вперше архітектура згорткової мережі була запропонована в 1989 році [25]. Головною особливістю такої архітектури стала операція згортки (Convolution).

Опреація згортки представляє собою обробку зображення певним набором фільтрів. Фільтри - це матриці певного розміру, які задаються архітектурою мережі (зазвичай 3x3, 5x5 або 7x7). Ці фільтри накладаються на зображення, і виконується математична операція Згортки, тобто відбувається сумування попарних добутків відповідних значень фільтрів та пікселів зображення, на які фільтр накладений (Рисунок 2.2).

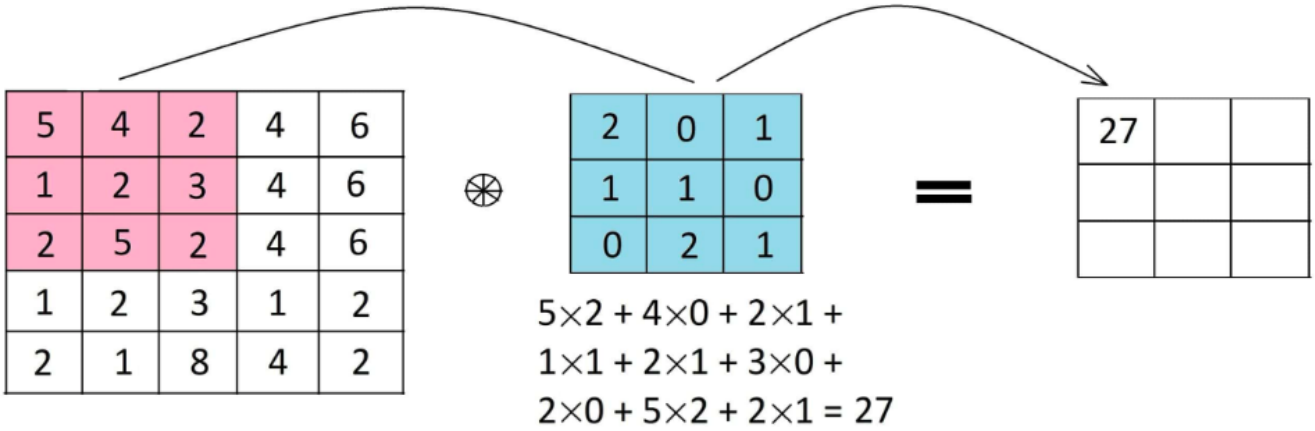


Рисунок 2.2: процес обрахунку операції Згортки з фільтром розміру 3x3 [26].

Фільтр проходить по всьому зображенню, в результаті отримується нове зображення, меншого розміру.

Фільтри можуть накладатися на зображення з певним інтервалом. Також до початового зображення може додаватися відступ по краям у вигляді додаткових рядків і стовпчиків пікселів, щоб нівелювати зменшення розміру зображення.

Кожне отримане зображення після Згортки називається картою ознаки зображення (feature map). Якщо ж використовувати на зображенні декілька фільтрів одночасно, відповідно отримаємо кілька карт ознак.

До нейронних мереж значення фільтрів підбиралися експериментально. Існують фільтри які додають до зображення розмитість (Рисунок 2.3), або навпаки, роблять контури зображення більш чіткими. В запропонованій ж архітектурі згорткової мережі, значення фільтрів є невідомими параметрами, які модель вивчає під час тренування. До того ж операції згортки можуть накладатися одне на одного, тобто згортка виконується на результаті попередньої згортки [25]. З кожним новим рівнем згортки модель вивчає нові, більш комплексні та абстрактні характеристики зображення, які враховують їх двовимірну природу (Рисунок 2.4).

Використання згорткових рівнів стало повсякденною практикою в моделях комп'ютерного зору, завдяки тому що вони здатні витягувати комплексні характеристики з зображення при цьому зменшуючи розмірність даних з кожним

рівнем згортки, що зменшує необхідні обчислювальні потужності.

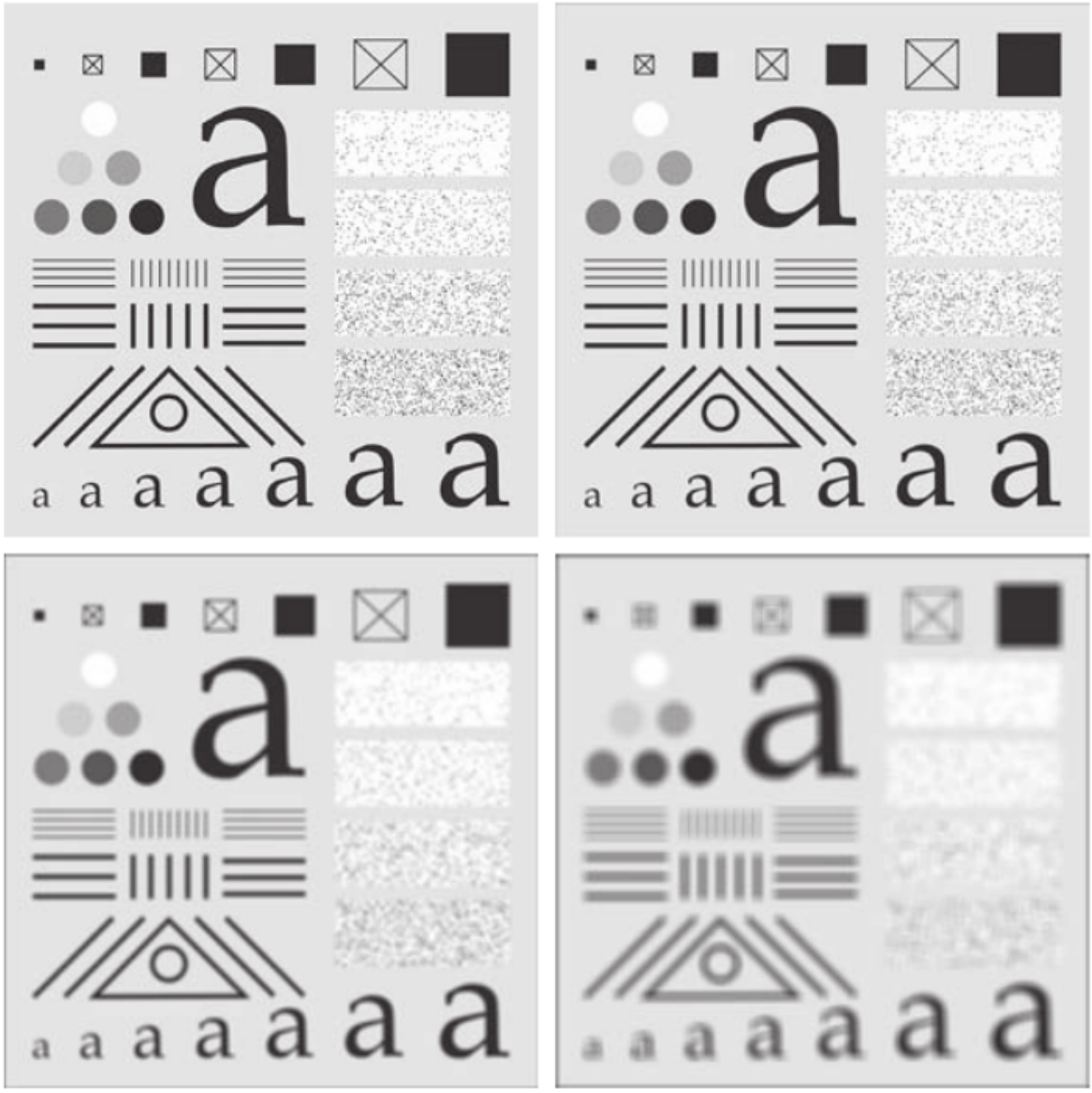


Рисунок 2.3: Накладання фільтрів розмитості різного розміру на оригінальне фото (зверху ліворуч) [22].

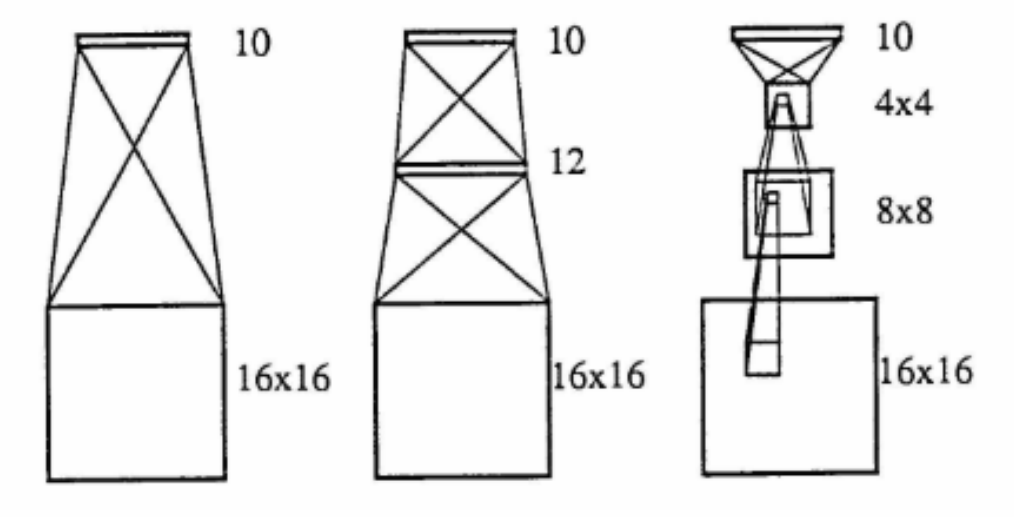


Рисунок 2.4: Приклад накладання згорткових рівнів одне на одного [25]

2.3.2 Особливості нейронної мережі AlexNet

AlexNet стала першою глибинною згортковою нейронною мережею [27]. Рішення цієї моделі, запропонованої у 2012 році, стали основою для багатьох наступних архітектур. Основні з них це використання *згорткових блоків (Convolution Block),* та тренування моделі паралельно на кількох GPU (Рисунок 2.5).

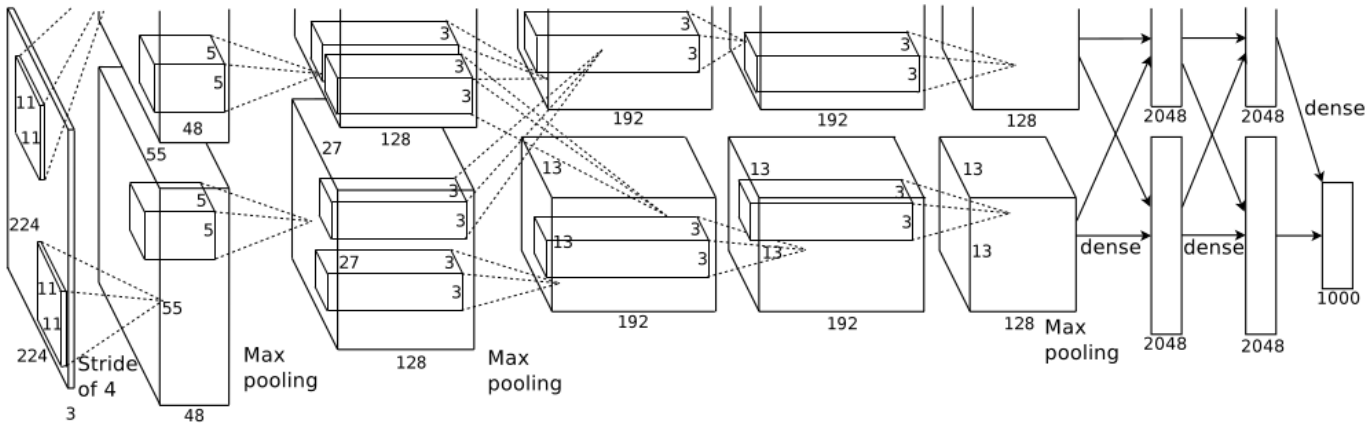


Рисунок 2.5: Архітектура моделі AlexNet [27]

Згортковий блок складається з трьох частин: операції згортки; об’єднувального шару (Pooling Layer) та рівня активації ReLU.

- Об’єднувальний шар (Pooling layer) використовується для зменшення розмірності отриманих зі горткового рівня Характеристик зображень. Матриця Характеристики розбивається на блоки, зазвичай 2x2, і з цього блоку залишається лише одне значення. Це може бути або середнє значення пікселів блоку, тоді рівень називається AveragePooling, або максимальне значення, тоді рівень називається MaxPooling. У AlexNet використовується саме MaxPooling.
- ReLU активація (Rectified Linear Unit) - одна з видів функцій активації. Функції Активації привносять нелінійні зв’язки до нейронних мереж, дозволяючи їм вчити більш комплексні паттерни. Хоча існує багато можливих варіантів функцій активації, ReLU наразі вважається одним з найкращих варіантів через свою простоту обчислення (Рисунок 2.6).

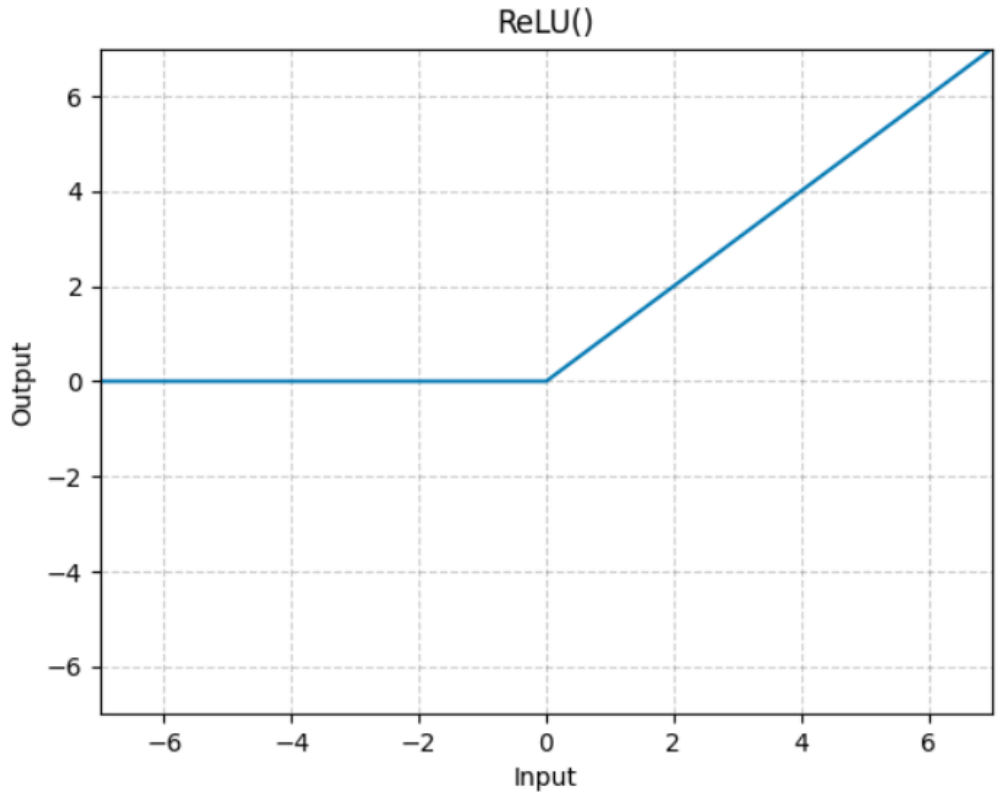
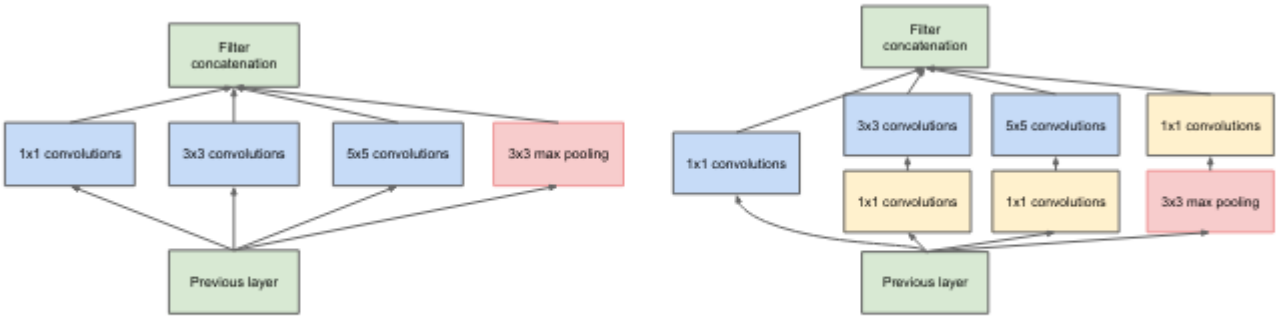


Рисунок 2.6: Функція Активації ReLU [28].

2.3.3 Особливості нейронної мережі Inception

Inception (її ще називають GoogleNet) була представлена у 2014 році [29]. Моделі, які були до цього, намагалися покращити результат AlexNet додаючи більше згорткових блоків. Автори Inception, у свою чергу, представили *Inception-блок* (Рисунок 2.7):



(a) Inception module, naïve version (b) Inception module with dimension reductions

Рисунок 2.7: Блок Inception [29].

Inception-блок використовує одночасно декілька фільтрів різних розмірів, конкатенує результати і передає їх у наступний згортковий блок. Використання таких блоків робить модель трохи ширшою, та зменшує її глибину. В наступних роках були представлені моделі Inception v2 та Inception v3, у яких Inception-блок був більш оптимізованим.

GoogleNet містив всього 27 рівнів, що вело до затухання градієнта. **Затухання градієнта** - серйозна проблема, яка була перешкодою розвитку глибинних нейронних мереж. Вона заключається в тому, що зворотній градієнт (backpropagation) затухав, не пройшовши всі рівні мережі. Щоб уникнути цієї проблеми, у GoogleNet фінальні ймовірності викликаються три рази: один раз в кінці, і два - посередині мережі, і обчислюють для них похибки. А фінальна похибка є ваговою сумою цих трьох похибок [29]. (Рисунок 2.8)

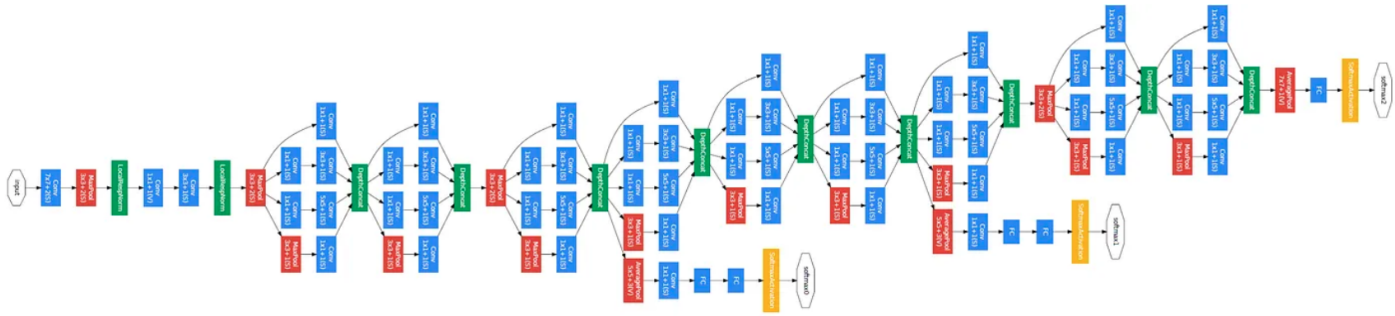


Рисунок 2.8: GoogleNet з Inception-блоками та трьома фінальними ймовірностями (жовті прямокутники) [30].

2.3.4 Особлівості нейронної мережі ResNet

Проблема затухання градієнта не давала збільшувати глибину моделей. Спосіб вирішення цієї проблеми був запропонований з появою моделі ResNet (Residual Neural Network, або Залишкова Нейронна Мережа) [31]. Ця модель містить у собі **Залишковий Модуль (Residual Unit)**, представлений на Рисунку 2.9:

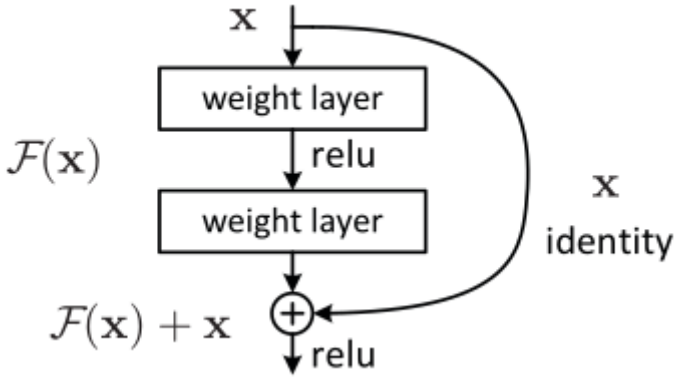


Рисунок 2.9: Залишковий модуль (Residual Unit) [31]

Паралельно з виконанням необхідних перетворень, мережа ‘прокидує’ дані вглиб за допомогою спеціальних зв’язків, які називаються *швидкими з’єднаннями (shortcut connections)*. Ці зв’язки не додають ні додаткових параметрів, ні обчислювальної складності, але дозволяють впоратися з проблемою затухаючого градієнту, та навчати дуже глибокі нейронні мережі (Рисунок 2.10). В оригінальній

статті автори експериментували з мережами, які мали більше сотні шарів [31]. Залишковий Модуль дуже активно використовується в сучасних архітектурах комп’ютерного Зору.

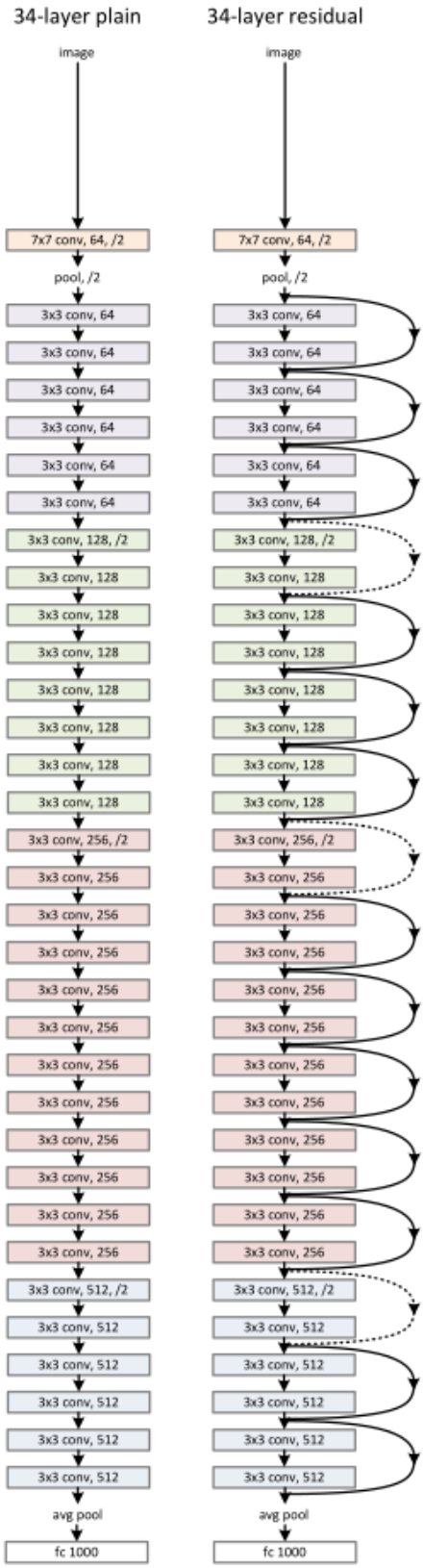


Рисунок 2.10: Мережа без залишкових модулів (ліворуч) та з залишковими модулями (ResNet-34 праворуч) [31].

2.3.5 Особливості нейронної мережі MobileNet

MobileNet - це серія нейромереж, яка розроблялася спеціально для їх використання на мобільних пристроях, з їх обмеженими обчислювальними можливостями. Перша версія була запропонована у 2017 році. Наразі існують версії MobileNet v1, v2 та v3.

У MobileNet звична операція згортки була розбита на дві частини: *глибинну згортку (depthwise convolution)* та *точкову згортку (pointwise convolution)* [32]. Глибинна згортка полягала у тому, що ми використовували лише один фільтр для кожного каналу зображення. Точкова згортка заключалася у використання 1x1 фільтрів. Було виявлено, що послідовна комбінація глибинної та точкової згортки дає той самий результат що й звичайна згортка (Рисунок 2.11), але при цьому зменшується кількість необхідних обчислень [32].

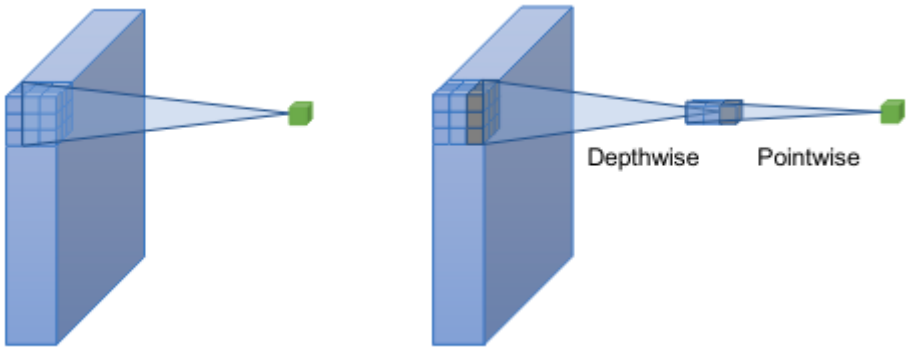


Рисунок 2.11: Порівняння звичайної згортки (ліворуч) та комбінації глибинної і точкової згорток (праворуч) [33].

Наступна генерація, MobileNetV2 вже складалася з *вузьких блоків (bottleneck blocks).* Цей блок був утворений додаванням ще одної точкової згортки, перед наведеною вище комбінацією глибинна-точкова згортка. Це дозволяло зменшити кількість фільтрів та, відповідно, обчислень. Окрім цього MobileNetV2 отримав

швидкі з’єднання, які називаються *інвертованими швидкими з’єднаннями (inverted residuals).* Таку назву вони отримали через те, що швидкі зв’язки з’єднували шари з малою кількістю фільтрів, на відміну від класичного швидкого з’єднання [34]. Це ще більше допомогло зменшити кількість обрахунків моделі (Рисунок 2.12):

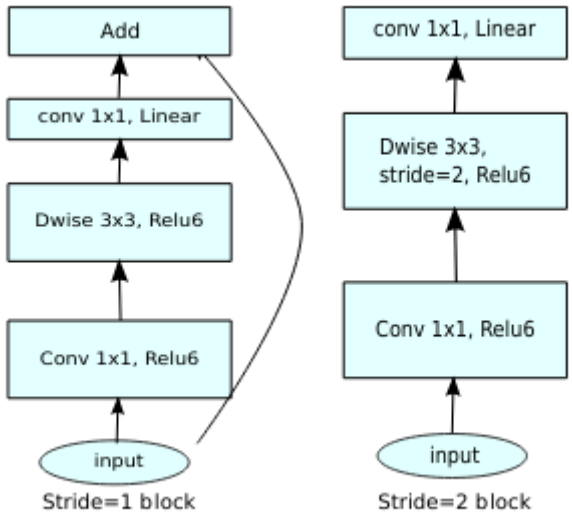


Рисунок 2.12: Вузькі блоки, з яких складається MobileNetV2 [34].

Головною зміною в MobileNetV3 стала поява нового модуля, який назвали *Блоком Стиснення та Збудження (Squeeze-and-Excite)* [35]. Ідея в тому, що при звичайній згортці ваги різних каналів не залежать одне від одного, тобто всі канали для мережі однаково важливі. Новий блок отримує канали зі згорткового шару, та за допомогою AveragePooling звужує їх до векторного представлення, де кожному каналу відповідає одне число. Далі цей вектор проходить через низку перетворень (два лінійні рівні та сигмоїда), що по суті представляє собою невеличку побічну нейронну мережу. В результаті ми отримуємо вектор чисел, де кожне число показує важливість відповідного каналу. В кінці за допомогою отриманих чисел видозмінюють ваги кожного початкового каналу, і ми отримуємо ваги, у яких міститься не лише інформація про просторовий зв’язок, а й про зв’язок між каналами (Рисунок 2.13).

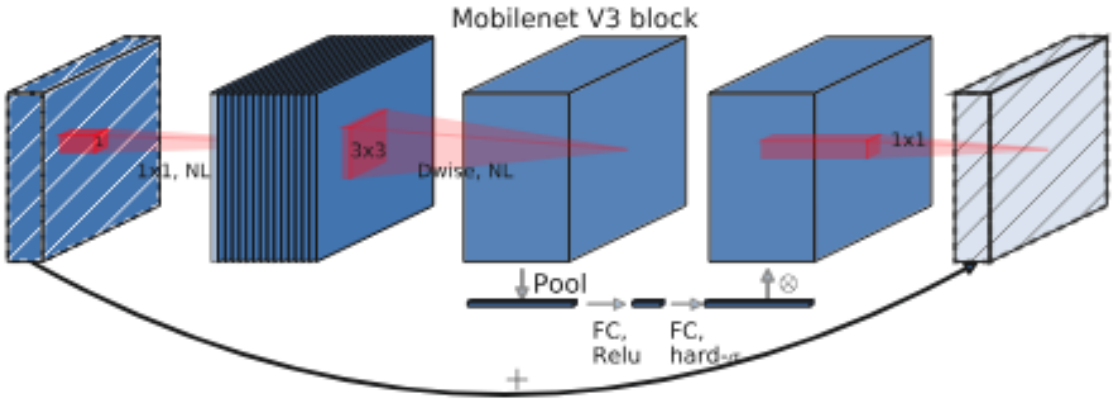


Рисунок 2.13: Блок MobileNetV3 з модулем Стиснення та Збудження знизу [35].

Моделі MobileNet активно використовуються у ситуаціях, коли обмежені обчислювальні потужності, або коли необхідно швидко обробити зображення згортковою мережею.

2.3.6 Особливості нейронної мережі EfficientNet

Розробники глибинних мереж часто намагаються робити їх глибшими та ширшими, оскільки є уявлення, що чим більша та складніша нейронна мережа, тим більш комплексні характеристики зображень вона зможе вивчити, і тим кращий буде її результат. Зазвичай так і працює, хоча і не завжди. Автори ResNet експериментували з надглибокими моделями, які мали більше 1000 рівнів (ResNet-1202). Така модель показала навіть трохи гірший результат, ніж ResNet-101 [31].

Постає питання, на скільки саме треба поглиблювати та розширювати мережу, щоб знайти баланс між використанням обчислювальних ресурсів та точністю. В результаті цих експериментів було запропоновано серію моделей під назвою EfficientNet [36]. Побудовані на основі MobileNet, їх розмір був відкоригований за допомогою так званого *комбінованого масштабування (Compound Scaling)*. Якщо раніше розширення моделей проводилося або в ширину, або в глибину, або

збільшуючи розмір вхідного зображення, і всі ці зміни робилися емпіричним шляхом незалежно одне від одного, то автори EfficientNet знайшли оптимальне відношення цих трьох рішень (Рисунок 2.14). Розширення задається параметром theta. Збільшення цього параметра розширює нейронну мережу одночасно в трьох напрямках. При цьому в кінці розширення, кількість FLOP (Floating point operations per second) отриманої мережі збільшується в 2^{theta} разів.

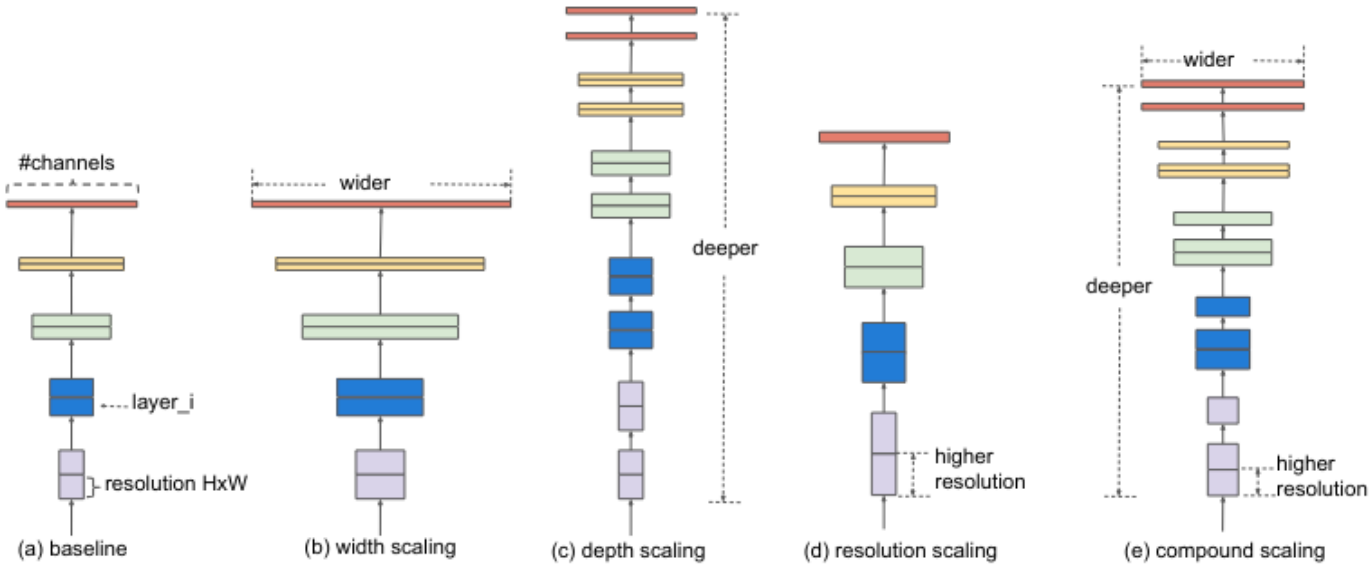


Рисунок 2.14: Різні варіанти масштабування мережі, та комбіноване масштабування [36].

Для демонстрації цього методу було побудовано початкову модель EfficientNet-B0. Далі її розширювали використовуючи різні значення theta. Було отриману цілу серію моделей різного розміру, від EfficientNet-B1 до EfficientNet-B7, які порівнювали з іншими моделями схожого розміру. Моделі EfficientNet показували кращий або такий самий результат як і інші моделі своєї категорії при набагато меншій кількості параметрів (Рисунок 2.15).

Model	Top-1 Acc.	Top-5 Acc.	#Params	Ratio-to-EfficientNet	#FLOPs	Ratio-to-EfficientNet
EfficientNet-B0	**77.1%**	**93.3%**	**5.3M**	**1x**	**0.39B**	**1x**
ResNet-50 (He et al., 2016)	76.0%	93.0%	26M	4.9x	4.1B	11x
DenseNet-169 (Huang et al., 2017)	76.2%	93.2%	14M	2.6x	3.5B	8.9x
EfficientNet-B1	**79.1%**	**94.4%**	**7.8M**	**1x**	**0.70B**	**1x**
ResNet-152 (He et al., 2016)	77.8%	93.8%	60M	7.6x	11B	16x
DenseNet-264 (Huang et al., 2017)	77.9%	93.9%	34M	4.3x	6.0B	8.6x
Inception-v3 (Szegedy et al., 2016)	78.8%	94.4%	24M	3.0x	5.7B	8.1x
Xception (Chollet, 2017)	79.0%	94.5%	23M	3.0x	8.4B	12x
EfficientNet-B2	**80.1%**	**94.9%**	**9.2M**	**1x**	**1.0B**	**1x**
Inception-v4 (Szegedy et al., 2017)	80.0%	95.0%	48M	5.2x	13B	13x
Inception-resnet-v2 (Szegedy et al., 2017)	80.1%	95.1%	56M	6.1x	13B	13x
EfficientNet-B3	**81.6%**	**95.7%**	**12M**	**1x**	**1.8B**	**1x**
ResNeXt-101 (Xie et al., 2017)	80.9%	95.6%	84M	7.0x	32B	18x
PolyNet (Zhang et al., 2017)	81.3%	95.8%	92M	7.7x	35B	19x
EfficientNet-B4	**82.9%**	**96.4%**	**19M**	**1x**	**4.2B**	**1x**
SENet (Hu et al., 2018)	82.7%	96.2%	146M	7.7x	42B	10x
NASNet-A (Zoph et al., 2018)	82.7%	96.2%	89M	4.7x	24B	5.7x
AmoebaNet-A (Real et al., 2019)	82.8%	96.1%	87M	4.6x	23B	5.5x
PNASNet (Liu et al., 2018)	82.9%	96.2%	86M	4.5x	23B	6.0x
EfficientNet-B5	**83.6%**	**96.7%**	**30M**	**1x**	**9.9B**	**1x**
AmoebaNet-C (Cubuk et al., 2019)	83.5%	96.5%	155M	5.2x	41B	4.1x
EfficientNet-B6	**84.0%**	**96.8%**	**43M**	**1x**	**19B**	**1x**
EfficientNet-B7	**84.3%**	**97.0%**	**66M**	**1x**	**37B**	**1x**
GPipe (Huang et al., 2018)	84.3%	97.0%	557M	8.4x	-	-

Рисунок 2.15: Порівняння моделей EfficientNet з іншими моделями схожого розміру [36].

2.4 Візуальні трансформери

Після появи трансформерів в задачах обробки природньої мови, описаних у попередньому розділі, було логічним спробувати їх застосувати для задач комп'ютерного зору. Але дослідники натикалися на проблему використання механізму уваги на зображеннях, оскільки кількість пікселів у зображеннях збільшується квадратично їх розміру, що веде до такої ж складності обчислень механізму уваги.

Візуальний трансформер (Visual Transformer) був запропонований як модифікація оригінального трансформера [12] з мінімальними змінами, доданими для підвищення ефективності роботи механізму уваги на зображеннях [37]. Зображення

розбивається на блоки фіксованого розміру (14x14, 16x16 пікселів, в оригінальній статті). Для кожного блока зображення формується векторна репрезентація, які потім з'єднуються у лінійну послідовність і подаються до кодувальника. Тобто в данному рішенні кожен блок зображення є аналогом текстового токена (Рисунок 2.1):

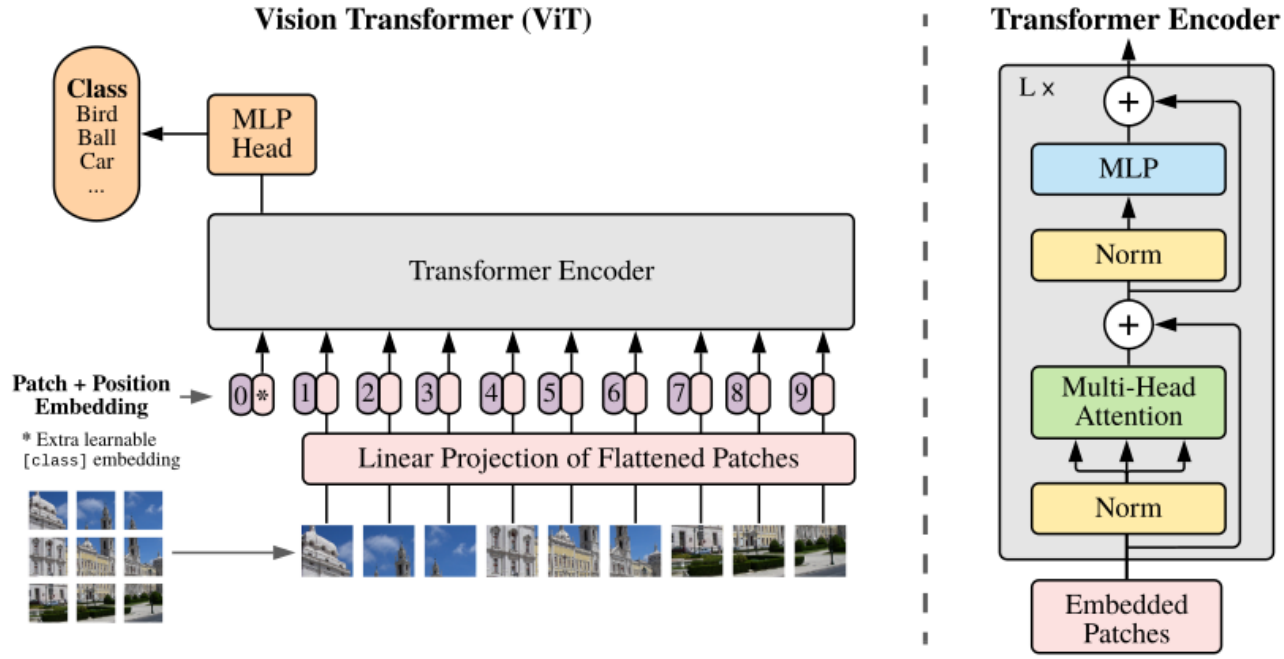


Рисунок 2.16: Огляд моделі візуального трансформера [37].

Також авторами статті запропонована альтернативна модель: оригінальне зображення спочатку проходить через згорткову мережу, наприклад ResNet, а отримані карти ознак (feature maps) вже розбиваються на блоки та подаються у механізм уваги.

Візуальні трансформери, натреновані на великих наборах даних, показали себе краще ніж згорткові мережі на великих наборах даних (ImageNet, CIFAR, інші). Але робота візуальних трансформерів потребує більших обчислювальних потужностей, що затрудняє їх використання у ситуаціях з обмеженими ресурсами. До того ж, згорткові мережі все ще виграють в точності у трансформерів при тренуванні на малих та середніх наборах даних. Через ці обмеження згорткові мережі ще продовжують активно використовувати в задачах комп'ютерного зору, а розробники

активно досліджують подальші можливості для оптимізації візуальних трансформерів.

РОЗДІЛ 3 ЗАСТОСУВАННЯ МЕТОДИК НАВЧАННЯ З НУЛЯ (ZERO-SHOT LEARNING)

3.1 Загальний опис

Описані у попередніх розділах методи навчання належали до контрольованого навчання (Supervised Learning) - типу навчання, коли модель тренується на попередньо розмічених даних. Не дивлячись на ефективність такої методики у багатьох задачах, вона має також суттєвий недолік. Він полягає у зборі та підготовці відповідних даних. Цей процес зазвичай є дуже часозатратним, оскільки для гарних результатів моделі потребують великих наборів даних. Також може виникати проблема збору даних через їх фізичну відсутність (відсутність достатньої кількості зображень рідкісного захворювання через його малу поширеність; відсутність текстів, написаних на деякому вимираючому діалекті, оскільки майже не залишилось його носіїв, тощо). Потреба у вирішенні цієї проблеми призвела до появи навчання з нуля (Zero-Shot Learning).

Навчання з нуля (Zero-Shot Learning, ZSL) - це сценарій машинного навчання, в якому модель ШІ навчається розпізнавати і класифікувати об'єкти або концепції без попереднього ознайомлення з прикладами цих категорій або концепцій [38]. Іншими словами, ця методика дозволяє моделі класифікувати дані, на яких вона не навчалася. Робиться це зазвичай за допомогою додаткової інформації, яку модель намагається дістати з даних, наприклад, текстовий опис зображення.

Існують три групи методів, які використовують при тренуванні ZSL моделей:

- Методи на основі атрибутів (attribute-based methods) - ідея в тому, що модель вчить не класи різних об’єктів, а класи *ознак* різних об’єктів. Далі, незнайомий клас модель класифікує, виходячи з різних ознак, які вона знайшла в цьому класі. Наприклад, модель може зрозуміти, що перед нею зображення

бджоли, вивчивши ознаку “смугастий” з зображень зебр та тигрів, та ознаку “жовтий” з інших жовтих тварин [39].
Такий підхід має ряд недоліків. Він не працює, якщо об’єкт може змінювати свої ознаки, як, наприклад, хамелеон. Також підход на основі атрибутів не може працювати, якщо нові класи не мають розмічених ознак [38]. Наприклад модель, яка вивчила ознаки притаманні тваринам, не зможе знаходити меблі та предмети інтер’єру.

- Методи на основі векторного представлення (embedding-based methods) - модель створює вектор зображення, та порівнює його з текстовими векторами можливих класів. Сучасні ZSL моделі працюють саме за цим методом.

- Альтернативними методами може слугувати генерація невідомого класу за текстовим запитом, з використанням text-to-image генеративних моделей (Generative-based methods) [38]. Генерація даних може допомогти звести проблему до звичайного контрольованого навчання.

3.2 CLIP: «навчання з нуля» (zero-shot) класифікація зображень

CLIP (Contrastive Language-Image Pre-training) – навчання з нуля (zero-shot) модель для класифікації зображень. У CLIP використовується метод на основі векторів: тренування відбувалося на основі 400 мільйонів пар зображення-текст зібраних з інтернету. Під час тренування текстовий кодувальник та кодувальник зображень тренувалися закодовувати тексти та зображення відповідно. При класифікації CLIP утворює векторне представлення зображення, векторні представлення можливих класів, приводить їх до спільного простору та порівнює вектори зображення з текстовими векторами. ВІдповідно, як відповідь CLIP надає клас, текстовий вектор якого є найбільш схожим до вектору зображення [40]. Як тектовий кодувальник у CLIP використовується Трансформер [12]. Для кодувальника

зображення автори експериментували з ResNet () та візуальним трансформером [37]. Графічно метод роботи представлений на Рисунку 3.1:

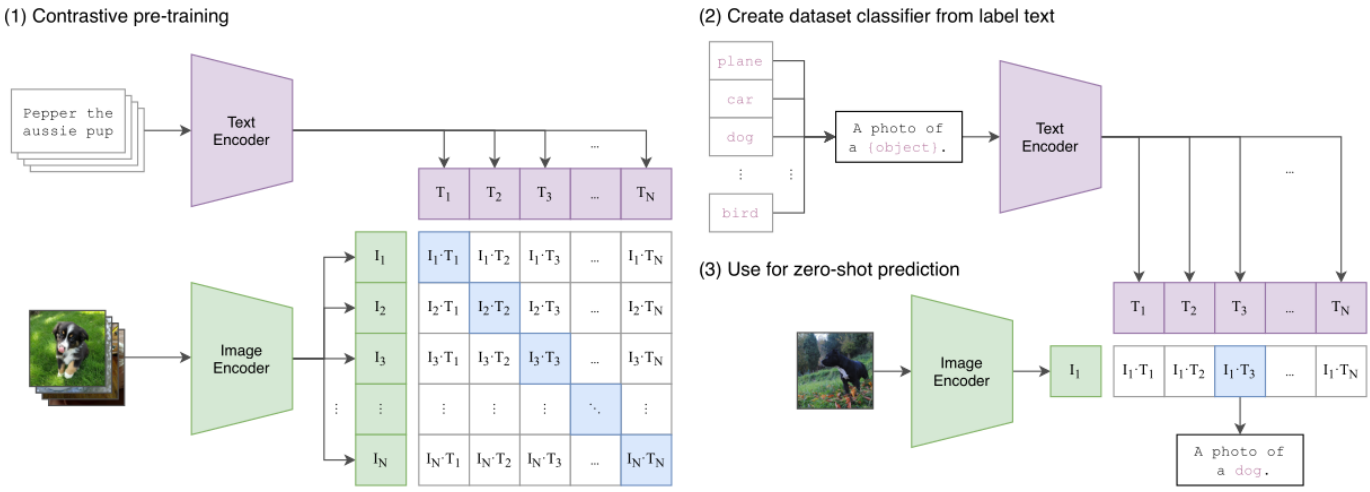


Рисунок 3.1: Принцип роботи CLIP [40].

CLIP проявив себе надзвичайно добре, показавши кращу точність на багатьох наборах даних, таких як ImageNet, CIFAR, StandfordCars, Country211 та інші. При цьому модель не навчалася спеціально знаходити класи цих наборів даних. Автори статті наголошують і на недоліках та обмеженнях CLIP:

- CLIP справляється гірше на специфічних, вузьких задачах класифікації, таких як розпізнавання моделей автомобілів, літаків або видів квітів [18].
- CLIP показує гірші результати на спеціалізованих наборах даних, таких як класифікація супутникових зображень (EuroSAT, RESISC45), виявлення пухлин лімфатичного вузла (PatchCamelyon), підрахунок об’єктів на синтетичних зображеннях (CLEVRCounts) та інші. А на наборі даних MNIST, в якому можна досягти майже ідеальної точності звичайною логістичною регресією, CLIP отримав “всього” 88% точності [40].
- Також CLIP показує погані результати в більш абстрактних завданнях, ніж просто сказати що на зображенні, таких як підрахунок кількості об’єктів на зображенні, або вимір приблизної відстані до об’єкта на зображенні

[40].

Не дивлячись на ці обмеження, CLIP став важливою віхою в розвитку навчання з нуля, адже показав реальну можливість досягати чудових результатів на багатьох наборах даних без додаткових тренувань.

3.3 «Навчання з нуля» (zero-shot) моделі детекції

CLIP показує гарні результати в задачі класифікації зображень, але він не зроблений для задачі детекції. Після викладення CLIP у відкритий доступ почалися розробки «навчання з нуля» моделей для детекції. В цьому розділі описано сімейство OWL та модель Ground Dino, так як ці моделі на момент написання статті показували найкращі результати серед «навчання з нуля» моделей детекції.

3.3.1 Моделі трансформації бачення для локалізації у відкритому світі (Vision Transformer for Open-World Localization)

Модель трансформації бачення для локалізації у відкритому світі (Vision Transformer for Open-World Localization - OWL-ViT) була розроблена для задач детекції на основі CLIP. Розробники намагалися вносити якомога менше змін у архітектуру, тому OWL-ViT, як і CLIP, складається з двох кодувальників: для тексту, та зображень. Найважливіша зміна відбулася після кодувальників. Якщо у CLIP векторна репрезентація зображення порівнювався з векторними репрезентації можливих класів, то в OWL-ViT після кодувальника зображення додали допоміжну нейронну мережу, яка повинна повертати координати прямокутників. Тобто векторна репрезентація зображення подається одночасно на порівняння з текстовими векторними репрезентаціями для визначення класу, та в допоміжну мережу для

визначення координат прямокутників [41]. Схема моделі наведена на Рисунку 3.2:

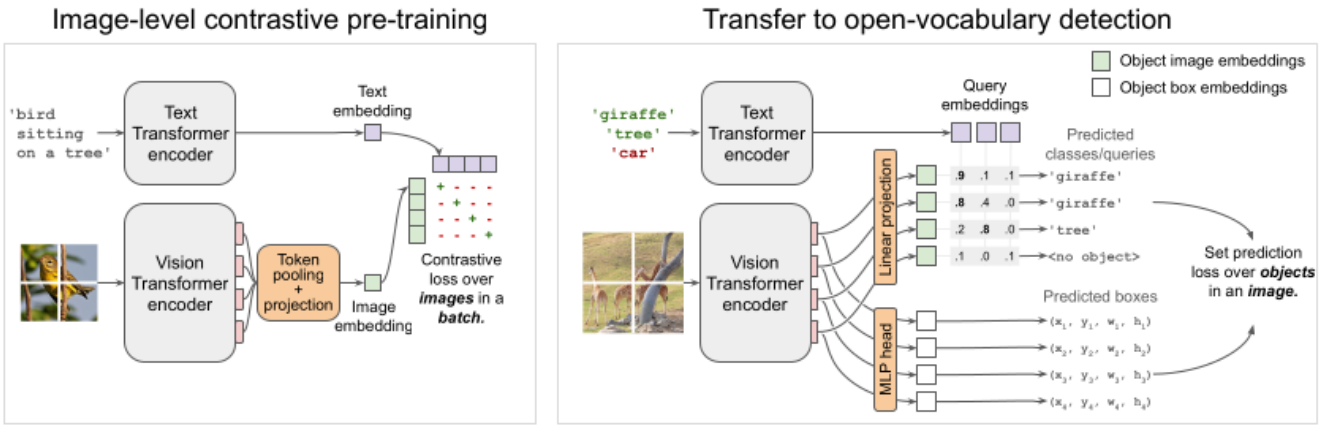


Рисунок 3.2: Схема моделі CLIP (ліворуч) та OWL-ViT (праворуч) [41].

Координати прямокутників обраховуються для кожної векторної репрезентації, іншими словами, максимально можлива кількість знайдених прямокутників не може перевищувати кількість блоків зображення. Але в реальності це обмеження не є проблемою, оскільки кількість блоків виходить суттєвим. При розмірі зображення 960x960 пікселів, розбиття на блоки розміру 16x16 приведе то утворення 3600 блоків. Сучасні набори даних детекції не мають такої кількості прямокутників в межах одного зображення.

Для покращення результатів OWL-ViT, було вирішено приділити особливу увагу збору даних для тренування. Через це, наступник OWL-ViT – OWLv2 - має лише кілька невеликих змін, доданих для оптимізації обрахунків. Натомість було висунуте припущення, що модель може проявити себе набагато краще, якщо по аналогії з CLIP навчити її на дуже великому наборі пар зображення-текст. Але оскільки не існує дуже великих розмічених набірів даних детекції, було вирішено використати підхід самонавчання (semi-supervised) [40]. На першому етапі процесу, бралися готові пари зображення-текст з інтернету. Далі текст розбивався на n-грами, і використовуючи їх як класи для детекції, OWL-ViT знаходив об’єкти на зображенні. Таким чином, OWL-ViT формував набір даних детекції для свого наступника, OWLv2, який на цих зображеннях тренувався. Після цього модель за необхідності ще

дотреновували вже на наборах даних, створених людьми (Рисунок 3.3):

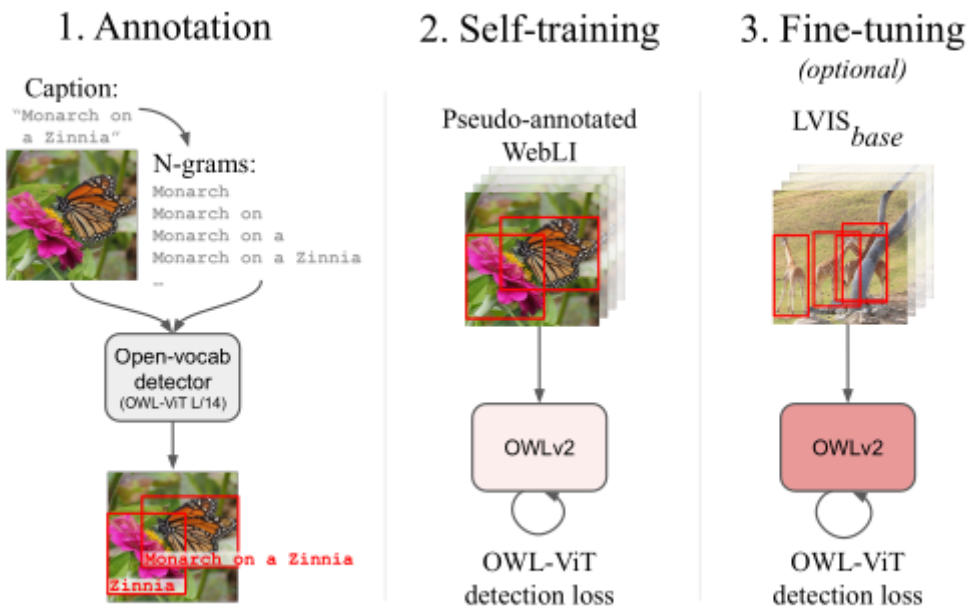


Рисунок 3.3: Процес тренування OWLv2 з використанням підходу самонавчання (self-supervised) [40].

Нові дані допомогли OWLv2 перевершити OWL_ViT на задачах детекції.

Окрім цього автори вирішили проблему розраховування координат прямокутників для кожного блока, що було дуже неефективним. Було додано ще одну допоміжну нейронну мережу, яка отримує на вхід векторну репрезентацію, і обчислює оцінку того, чи дійсно даний блок містить в собі об’єкт. Координати прямокутників будуть обраховуватися тільки для тих блоків, чия оцінка вища деякого встановленого значення [40].

3.3.2 Детектор Grounding Dino

Grounding Dino має інший принцип роботи, ніж OWL-ViT. В цьому детекторі

було використано ідею під назвою *фразове заземлення (phrase grounding),* вперше представленій в моделі GLIP (Grounded Language Image Pre-training). Ідея полягає у тому, що модель отримує зображення, текстовий опис, та з опису витягує об'єкти, які далі будуть знаходитися на зображенні. При цьому було додано рівні злиття (Fusion layers) між текстовими та візуальними характеристиками, щоб модель краще вчилась будувати зв'язки між цими двома модальностями. Після отримання векторних репрезентації текстових об'єктів та векторних репрезентації для різних регіонів зображення, ці векторні репрезентації порівнюються між собою для визначення наявності класів. Також векторні репрезентації регіонів зображення проходять через допоміжну нейронну мережу, для отримання координат прямокутника [42] (Рисунок 3.4):

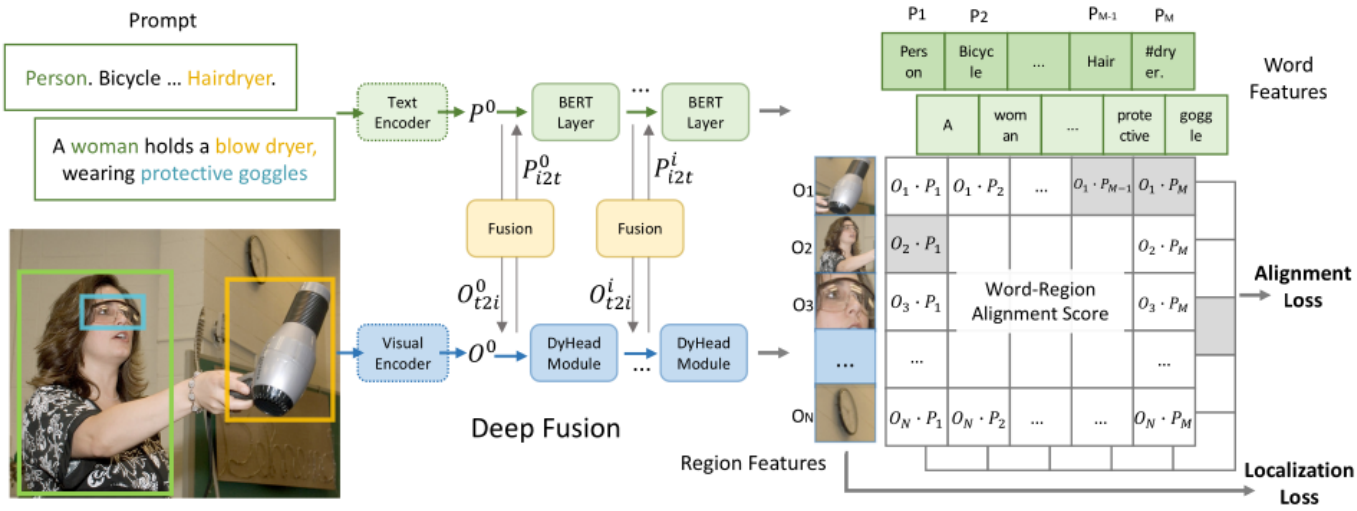


Рисунок 3.4: Візуалізація роботи GLIP та концепції *фразового заземлення* [42].

Автори Grounding Dino вирішили покращити дану модель, додавши архітектуру, яка вперше була запропонована під назвою DETR (DEtection TRansformer) [43]. DETR - трансформер, який має і кодувальник і декодувальник, та здатен робити детекцію на зображеннях. Це не «навчання з нуля» модель, і вона не працює з текстом, тому DETR має лише один кодувальник, на відміну від попередніх моделей. Основна відмінність даного трансформера від оригінального заключається в декодувальнику: він не є авторегресивним, тобто всі потенційні прямокутники обробляються ним паралельно, а не один за одним. Кількість можливих

прямокутників регулюється так званими *об'єктними запитами (object queries)* - це невелика кількість вивчених позиційних кодувальників На їх основі, а також на основі отриманих від кодувальника характеристик, робиться передбаченння прямокутників (Рисунок 3.5):

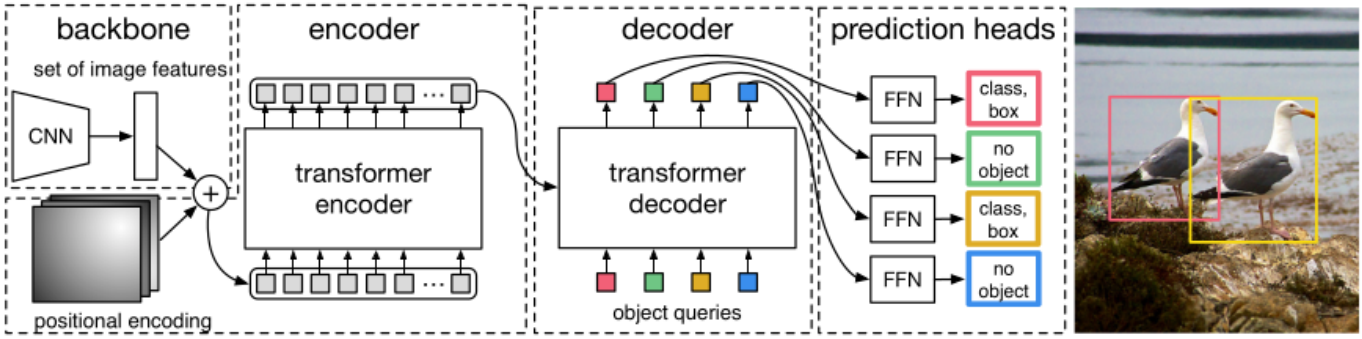


Рисунок 3.5: Схема роботи DETR [43].

Врешті решт, Grounding Dino поєднує ідеї описаних вище двох моделей. Від GLIP взята концепція фразового заземлення: класи для детекції отримуються з текстового опису зображення, а текстовий та візуальний кодувальники дістають характеристики відповідних модальностей з активним обміном інформації через рівні злиття. Від DETR взята архітектура трансформера: доданий декодувальник, який повертає прямокутники з назначеними класами. Для обрахунків в декодувальнику використовуються не лише візуальні характеристики, а й текстові, і кожен з них проходить свій механізм уваги. Об'єктні запити для декодувальника генеруються на основі характеристик обох модальностей, отриманих з кодувальників [44] (Рисунок 3.6):

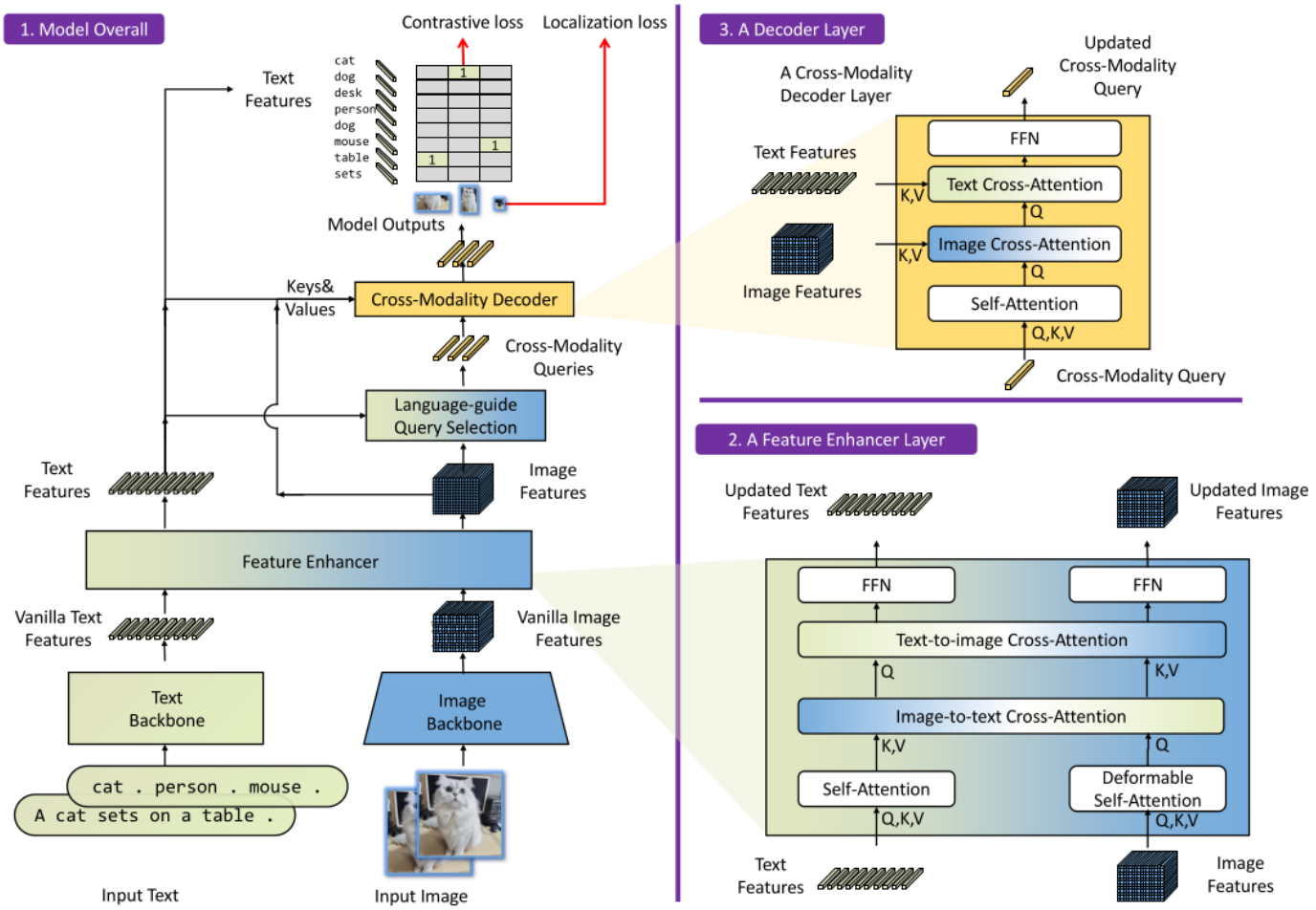


Рисунок 3.6: Схема роботи Grounding Dino [244].

Grounding Dino показав хороші результати в задачах детекції. Він побив точність свого попередника, модель GLIP, та навіть показав кращі результати, ніж описана в попередньому підрозділі модель OWL-ViT на наборах даних ODinW [44]. OWLv2 ще не була представлена на момент появи Grounding Dino.

Також автори представили застосунок редагування зображень, у якому було поєднано Grounding Dino та Stable Diffusion. Для зображення генерується текстовий опис об’єкта який треба замінити, Grounding Dino знаходить цей об’єкт, і Stable Diffusion редагує вибраний об’єкт [44] (Рисунок 3.7).

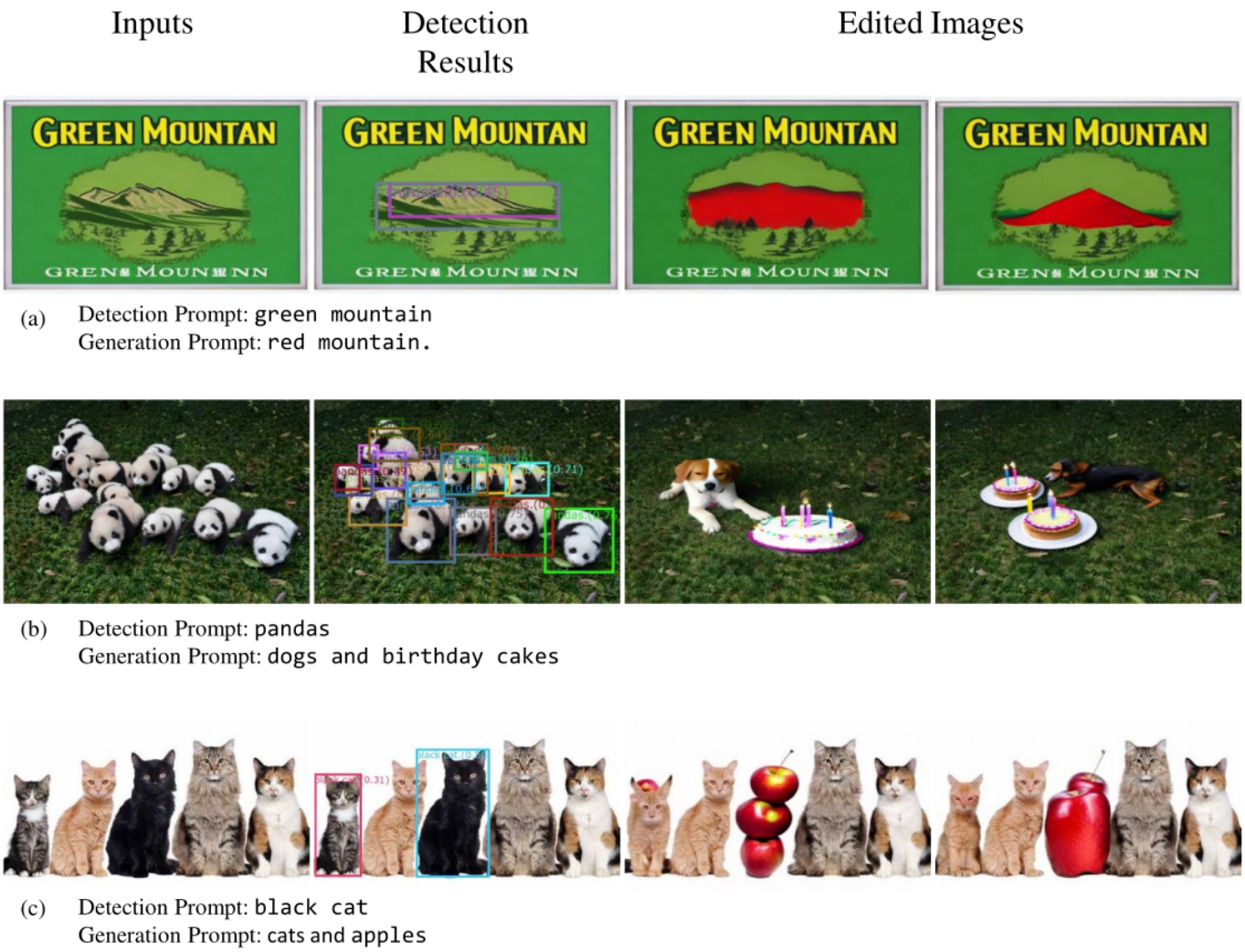


Рисунок 3.7: Поєднання Grounding Dino та Stable Diffusion для редагування зображень [44].

РОЗДІЛ 4 РОЗРОБКА ТА РЕАЛІЗАЦІЯ МОДЕЛІ З ВИКОРИСТАННЯМ МЕТОДИК НАВЧАННЯ З НУЛЯ

4.1 Опис набору даних

При виборі набору даних для експериментів, різні набори даних порівнювалися за такими ознаками:

- зображення зроблені з повітря (дронів, літаків, супутників)
- набір даних сформований для задач знаходження об’єктів (object detection) і має відповідну розмітку
- зображення зроблені зі значної висоти, для того щоб об’єкти були малими і не займали суттєву частину зображення.

Згідно цих ознак, було вибрано набір даних VisDrone [1]. Рисунок 4.1 показує приклад зображення :

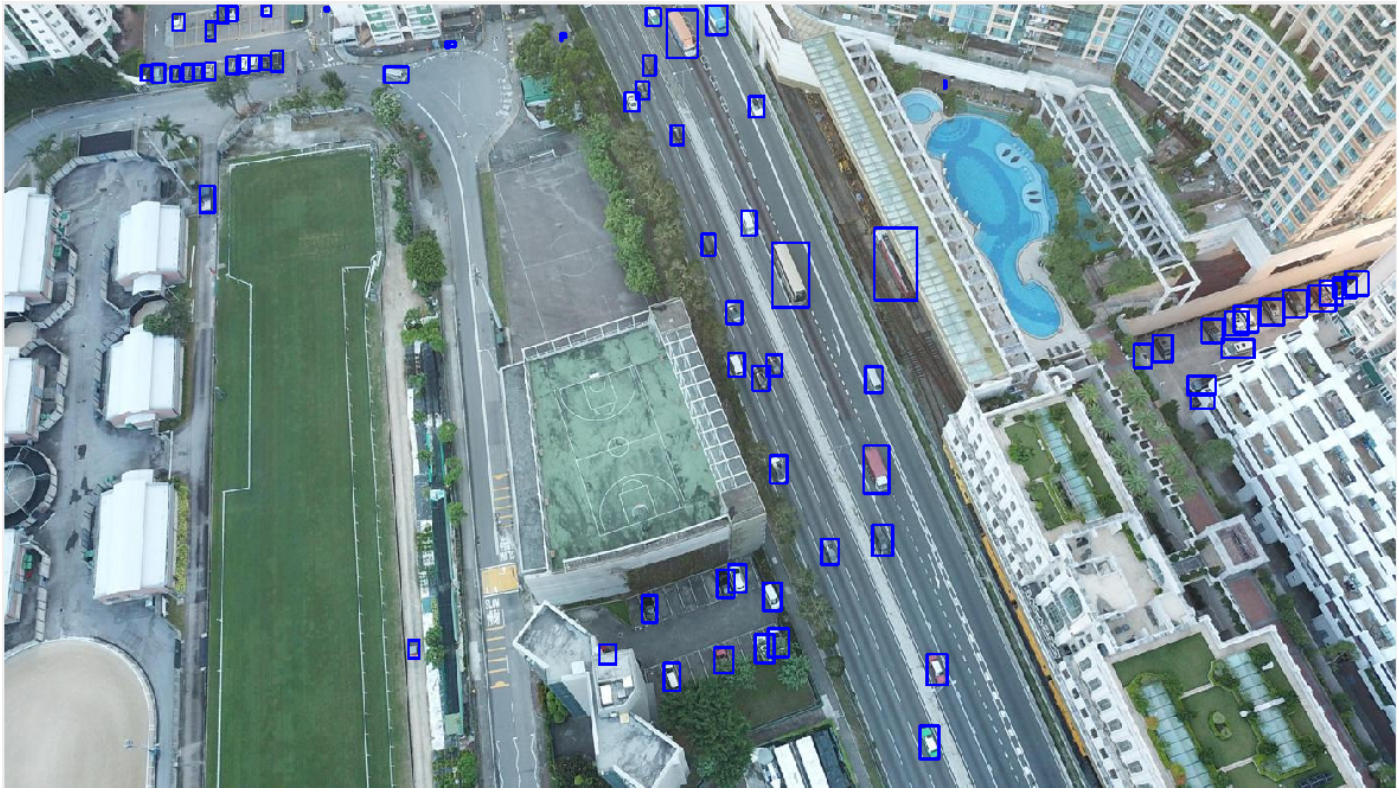


Рисунок 4.1. Приклад зображення з набору даних VisDrone [1]

Характеристики набору даних:

1. Містить 10, 209 зображень.

2. Зображення були зняті з дронів в 14 різних містах Китаю: Тяньцзінь, Гонконг, Дацин, Ганьчжоу, Гуанчжоу, Цзінькан, Лючжоу, Нанкін, Шаосін, Шеньян, Наньян, Чжанцзякоу, Сучжоу та Сюйчжоу. Набір даних охоплює різні погодні та освітлювальні погодні та освітлювальні умови, що представляють різноманітні сценарії повсякденного життя.

3. Містить 10 класів: pedestrian, people, bicycle, car, van, truck, tricycle, awning-tricycle, bus, motor. Також набір даних містить низьку інших класів (machineshop truck, forklift truck, tanker, інші). Але через малу кількість даних класів у наборі даних, автори не включали їх у обрахунки метрик, тож в даній роботі вони теж ігноруються.

Класи нерівномірно наявні в наборі даних. Рисунок 4.2 показує кількість кожного класу в тренувальному наборі даних:

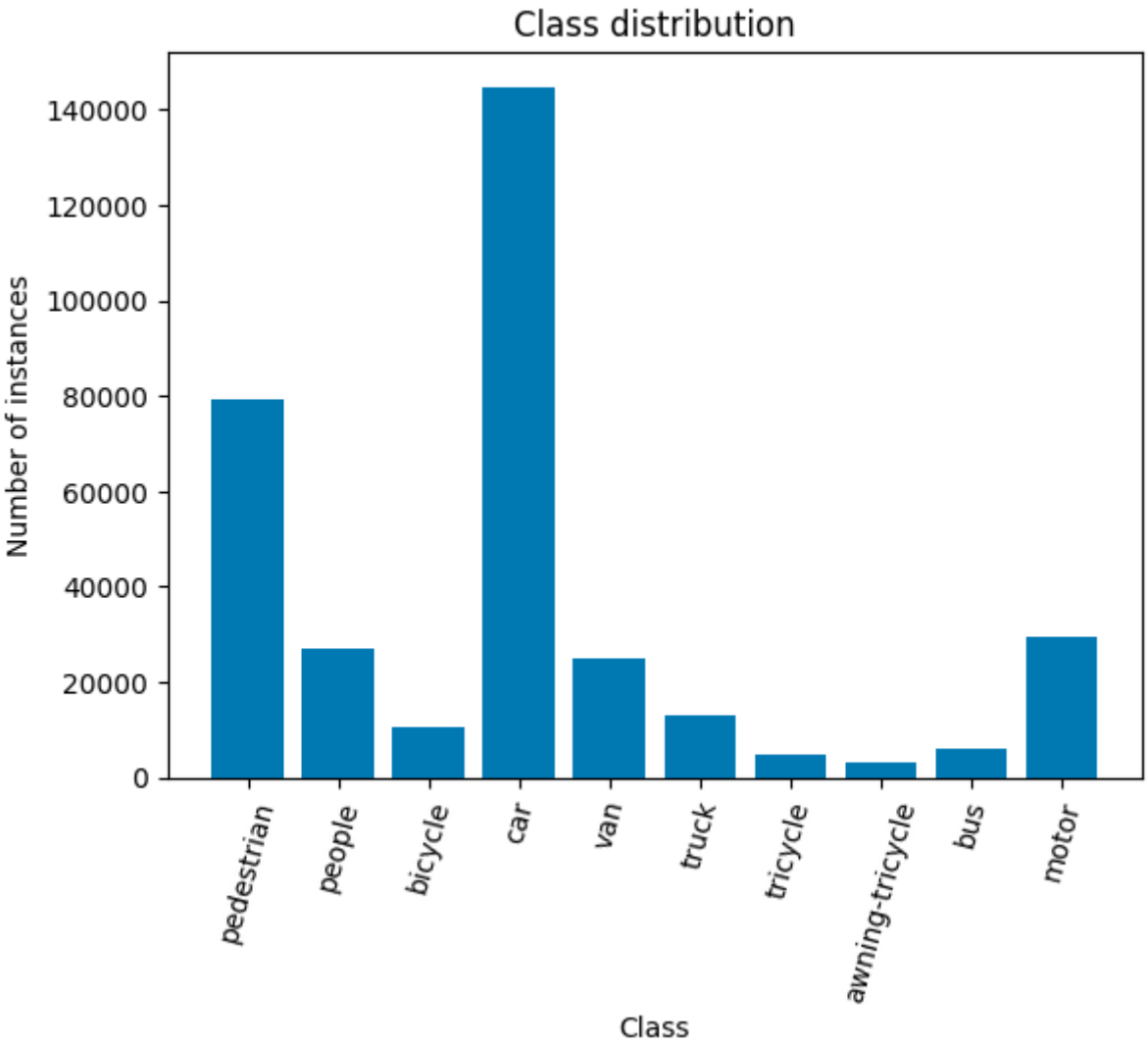


Рисунок 4.2. Розподіл класів на тренувальному наборі.

4. Додатково для кожного прямокутника окрім його класу, вказано коефіцієнт перекриття (*occlusion ratio*), який показує ступінь перекриття об’єкту іншими об’єктами, та коефіцієнт усічення (*truncation ratio*), який показує яка частка об’єкту виходить за межі зображення. Ці коефіцієнти приймають значення від 0 до 1.

5. Більшість об’єктів набору даних є малими відносно зображень, через високу висоту зйомки. Рисунок 4.3 показує, що відносний розмір абсолютної більшості боксів (відношення розміру прямокутника до розміру зображення) близький до нуля:

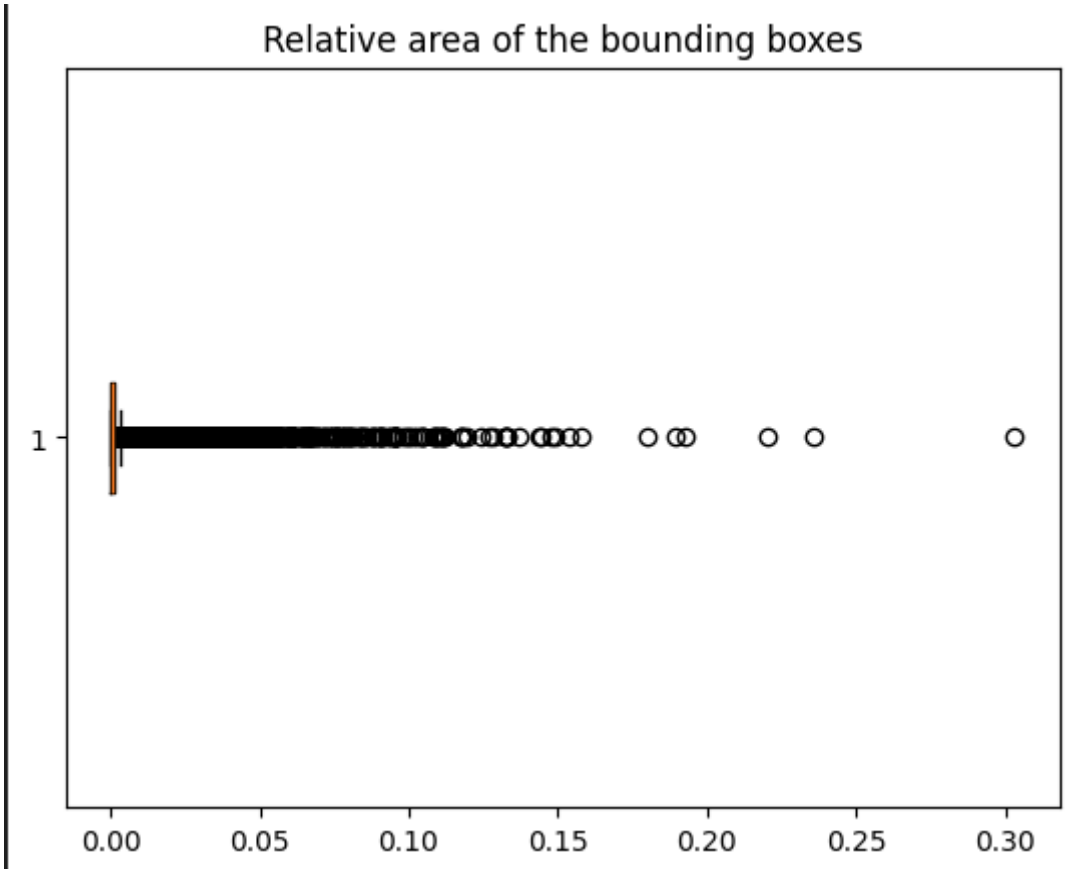


Рисунок 4.3. Відносні розміри прямокутників тренувального набору.

6. Зображення набору даних VisDrone містять в середньому 40 боксів. Також зустрічаються зображення, які містять сотні прямокутників. Рисунок 4.4 показує розподіл кількості прямокутників на одне фото в тренувальному наборі. На рисунку 4.5 можна побачити приклад зображення, що містить більше 800 прямокутників:

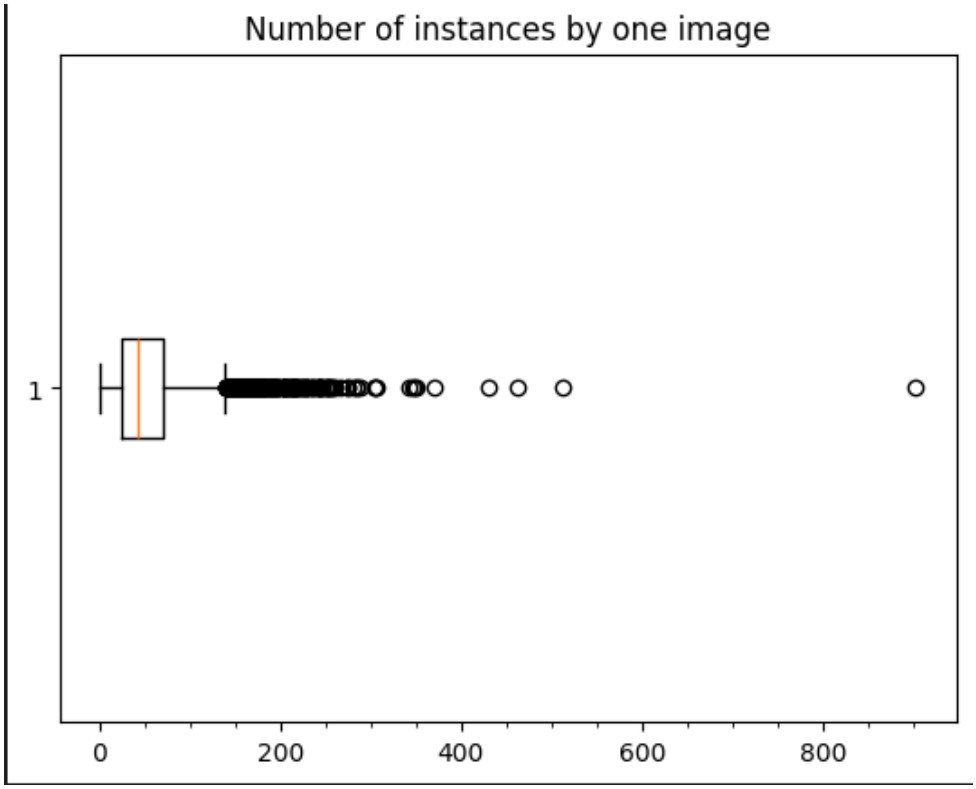


Рисунок 4.4. Розподіл кількості прямокутників на одне фото в тренувальному наборі

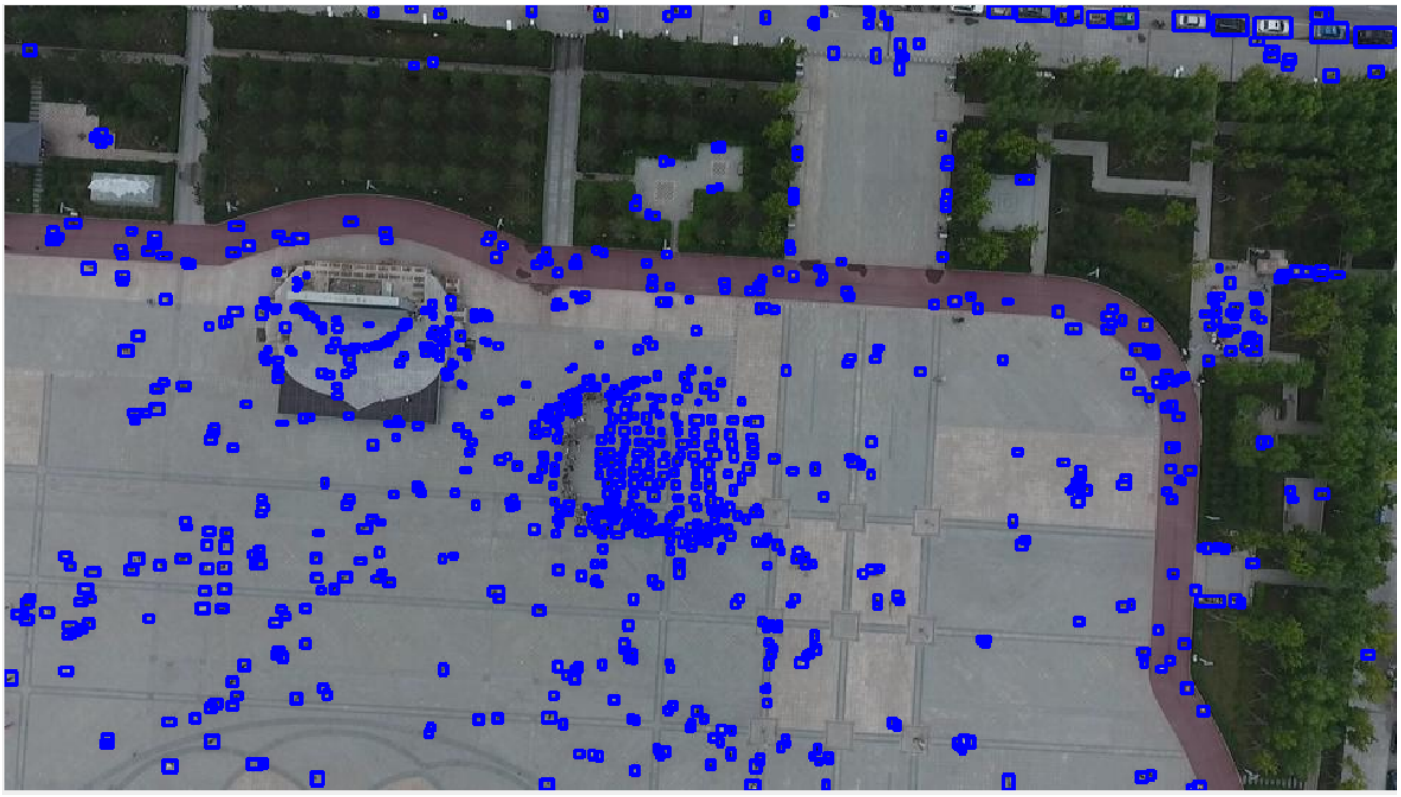


Рисунок 4.5. Приклад зображення, що містить більше 800 прямокутників.

4.2 Метрики для порівняння експериментів

На початку дослідження існуючих моделей постала проблема визначення їх точності. Вона полягає в тому, що існують різні метрики для задачі детекції, та різні формати. В даній роботі для порівняння результатів моделі використовувалися такі метрики:

- інтерпольована крива PR (Precision-Recall curve) - утворюється з відповідних значень Precision-Recall при різних значеннях впевненості (confidence).
- середня точність (Average Precision, AP) - площа під PR-кривою. Більше AP відповідає кращому співвідношенню Precision-Recall (Precision довго не падає при збільшенні Recall)

Дані метрики використовуються в змаганні PASCAL VOC і обчислюються для кожного класу окремо [45].

4.3 Порівняння «навчання з нуля» детекторів

Було опробувано два «навчання з нуля» детектори: OWLv2 та Grounding Dino. Для дослідження було обрано саме ці дві моделі, оскільки вони показували найкращі результати на великих загально доступних наборах даних, таких як ODinW, MSCOCO, LVIS та інші [46].

При тестуванні Grounding Dino стала зрозумілою проблема, пов’язана з принципом роботи моделі. Оскільки Grounding Dino не назначає знайденим прямокутникам один з наданих класів, а витягає інформацію про об’єкти з текстового опису зображення, отримані в результаті класи не завжди співпадали з класами VisDrone. Текстовий опис представляв собою усі десять класів, з’єднаних в один

рядок з розділенням ". “ [42]. В результаті модель назначала не тільки правильні назви класів, а й їх комбінації, наприклад “bicycle tricycle”, “car van”, “tricyclecycle” та інші.

В свою чергу OWLv2 вибирає класи зі списку наданих, тому для подальшої роботи було вирішено залишити саме OWLv2.

OWLv2 був опробований у середовищі Google Colab, на 548 зображеннях валідаційного датасету VisDrone. Було пораховані метрики для кожного класу, як було описано в розділі 4.2. Перші результати наведені на рисунках 4.6 та 4.7:

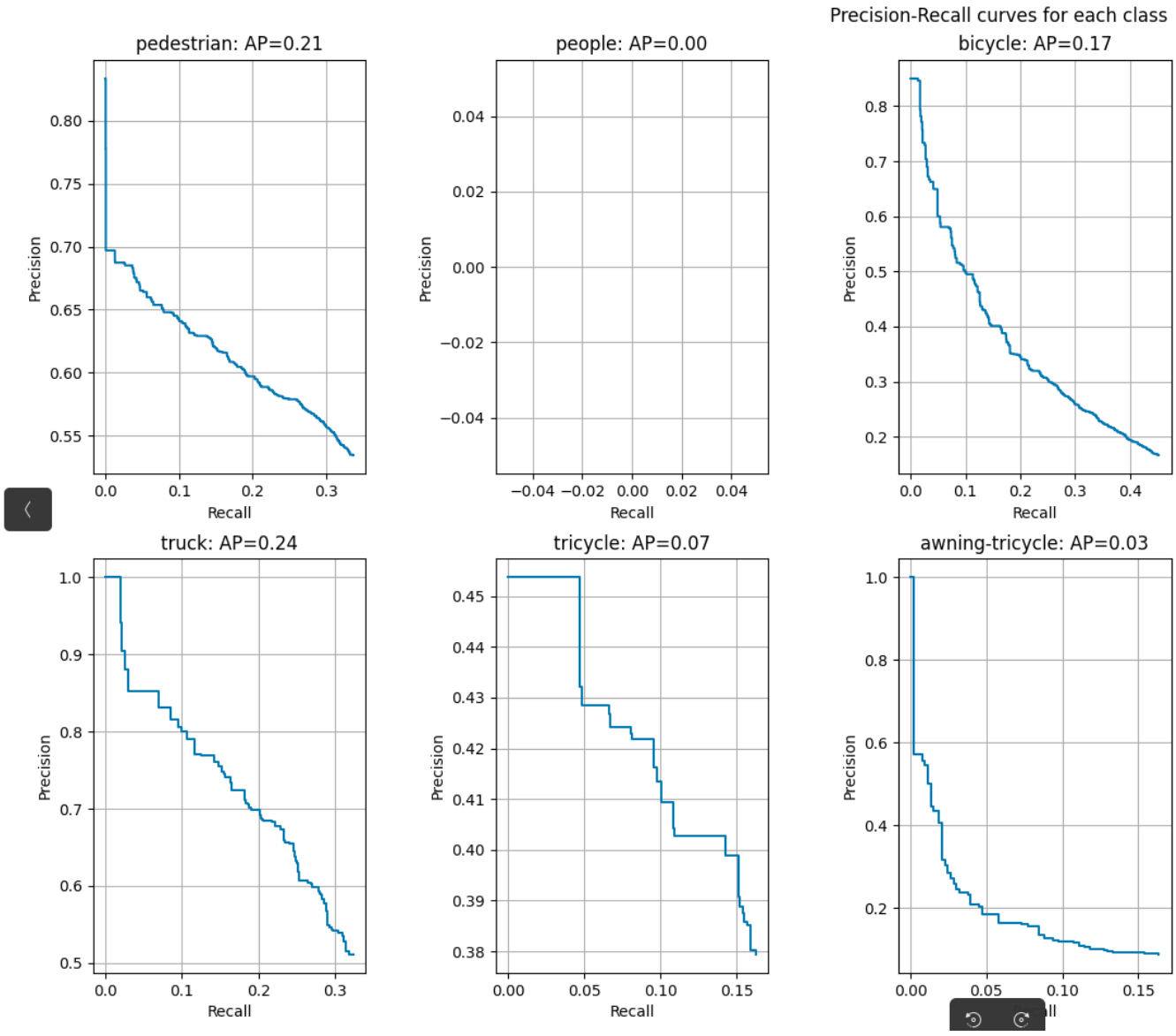


Рисунок 4.6: Результати роботи OWLv2 на шести класах набору даних

VisDrone

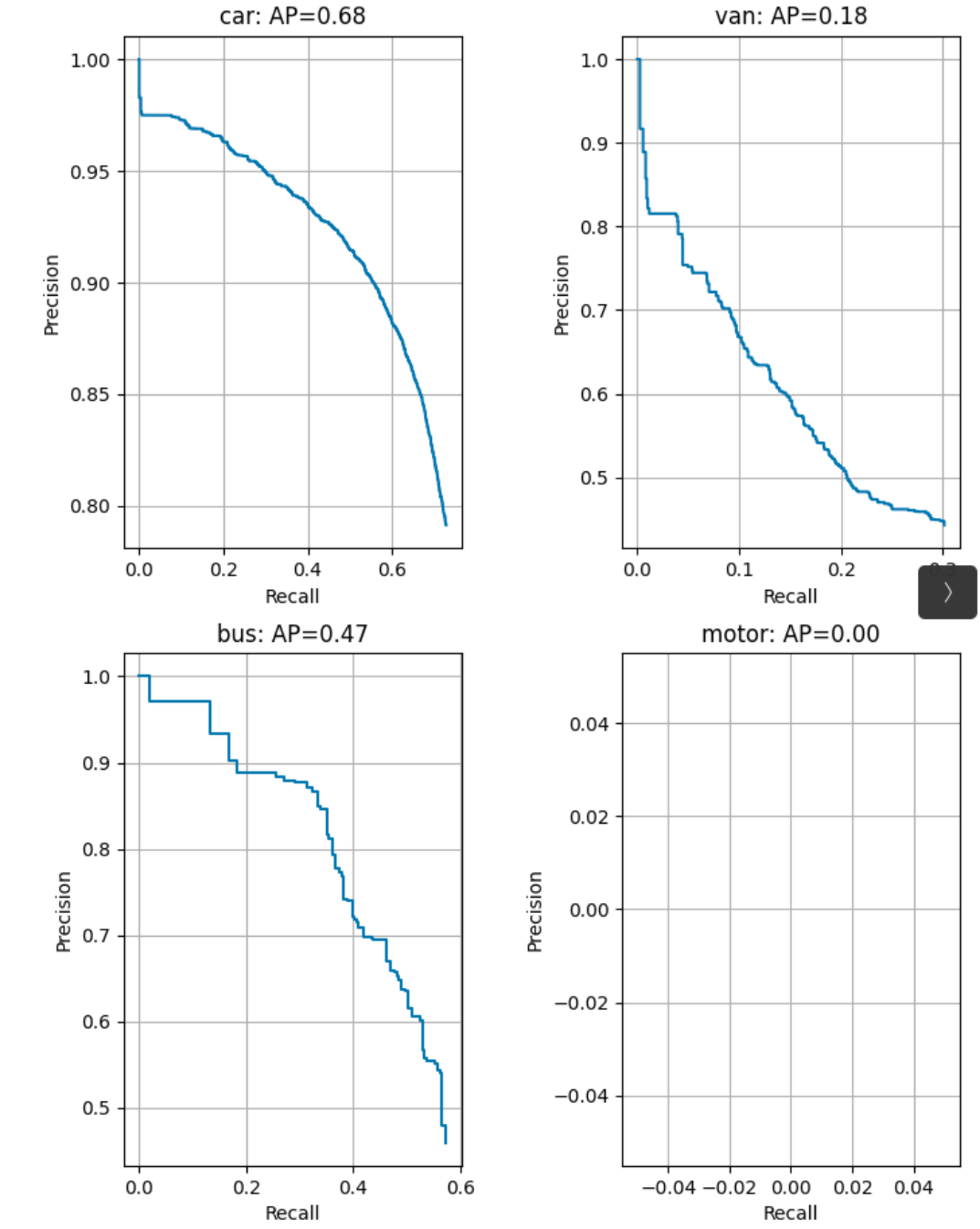


Рисунок 4.7: Результати роботи OWLv2 на чотирьох класах набору даних VisDrone

Бачимо, що найкращий результат модель показує на класі машина (car) (прямокутників класу car було найбільше в наборі даних). Також непогано модель знаходить клас автобус (bus). AP на інших класах коливається від 0.25 до 0.

Окремо вирізняються класи *people* та *motor,* для яких PR-крива не побудувалася, тобто ці класи не були знайдені моделлю взагалі. Було висунуто припущення, що модель не може знайти *motor*, оскільки у наборі даних VisDrone він означає мотоцикл, що є не очевидним з назви “*moto*r”. Виключно для моделі клас *motor* був перейменований на ***motorcycle***, при цьому в наборі даних цей клас залишився незмінним. Після даної зміни було отримано такі результати по класу *motor*(Рисунок 4.8):

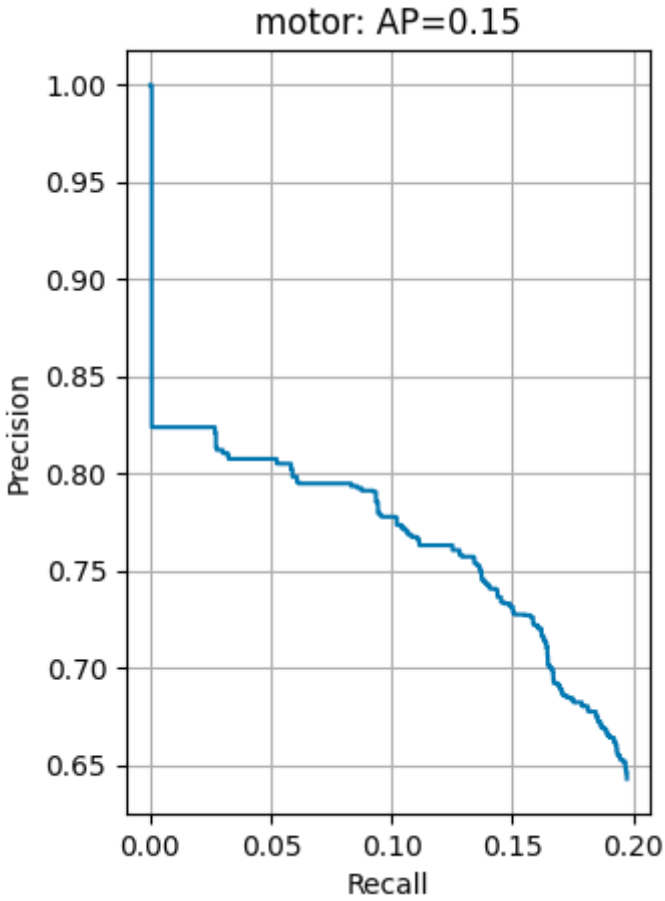


Рисунок 4.8: Результати роботи OWLv2 на класі m*otor* після перейменування

Щодо класу *people,* автори набору даних VisDrone зазначали таке:

“Якщо людина перебуває в положенні стоячи або йде, ми класифікуємо її як pedestrian; в іншому випадку, ми класифікуємо її як people.”[1]

Дане твердження також є неочевидним, то ж «навчання з нуля» модель в

нашому випадку зараховує всіх людей до *pedestrian.* Причинами того, чому саме *pedestrian* може бути декілька. По-перше *pedestrian* означає однину на відміну від *people*. По-друге, *pedestrian* - “пішохід” - більш вузьке поняття ніж *people*, і позначає людей, які йдуть по вулиці. А набір даних VisDrone складається саме зі знімків міських вулиць, тож модель з більшою ймовірністю вибирає саме клас *pedestrian.*

Для перевірки першого твердження, клас *people* був перейменований на клас ***person.*** Отримані результати можна побачити на Рисунку 4.9:

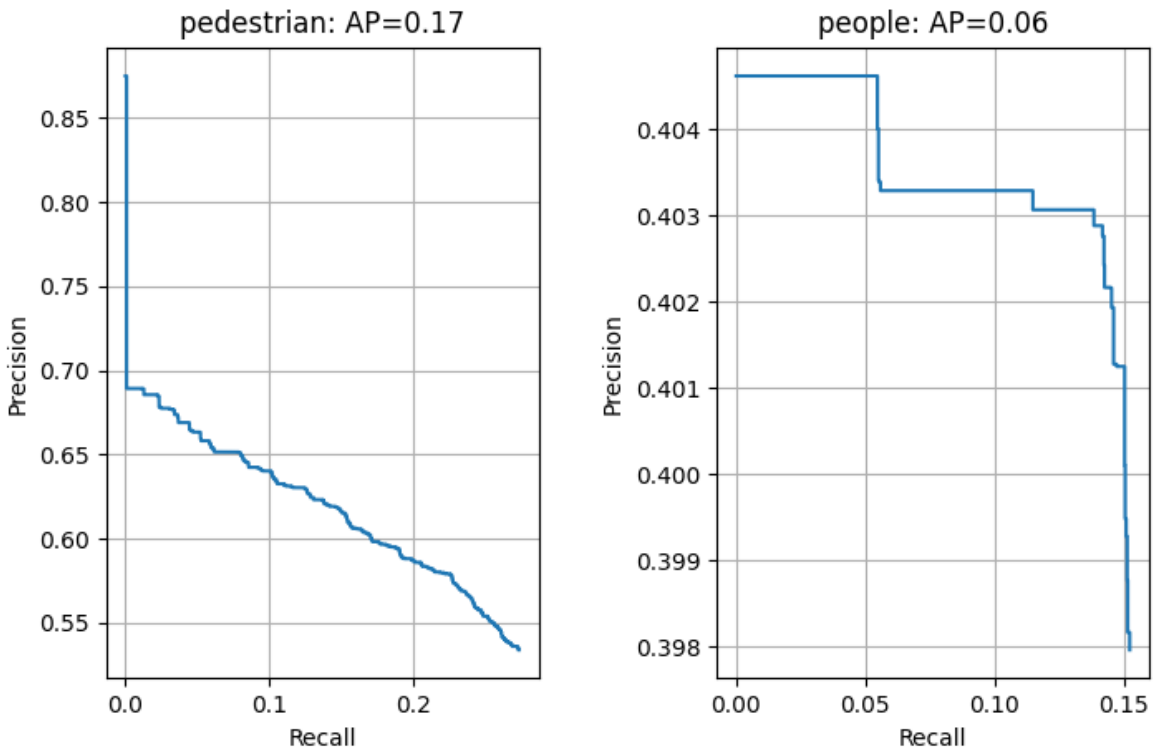


Рисунок 4.9: Результати роботи OWLv2 на класах pedestrian, people після перейменування

Як бачимо, OWLv2 почав знаходити клас *people,* але в більшості все одно позначає людей як *pedestrian*.

У подальшому в роботі ми продовжимо використовувати назви класів, які були в оригінальній статті [1], але для тренувань «навчання з нуля» моделі будуть братися змінені нами назви двох класів: *motor* – ***motorcycle***; *people* – ***person***.

4.4 Зміни, внесені у модель OWLv2

4.4.1 Загальний опис

Для покращення роботи OWLv2 на зображеннях з дронів, було внесено деякі зміни у архітектуру моделі (Рисунок 4.10). Було вирішена додати додатковий кодувальник зображень, паралельно візуальному трансформеру з OWLv2. Дизайн нового кодувальника був сформований спеціально для кращого знаходження локальних характеристик зображень, що має на меті покращити знаходження невеликих об’єктів. Візуальний трансформер, у свою чергу, отримує хороші загальні характеристики зображень, які добре генералізуються на нові приклади. Фінальною векторною репрезентацією зображення є комбінація двох представлень зображення. Наведені зміни були об’єднані разом з логікою візуального трансформера у єдиний клас Owlv2VisionTransformer, який наявний в оригінальній моделі. Це дозволило запобігти внесенню додаткових змін в інші частини моделі, оскільки вони взаємодіють з утвореним подвійним кодувальником так само, як і з оригінальним трансформером.

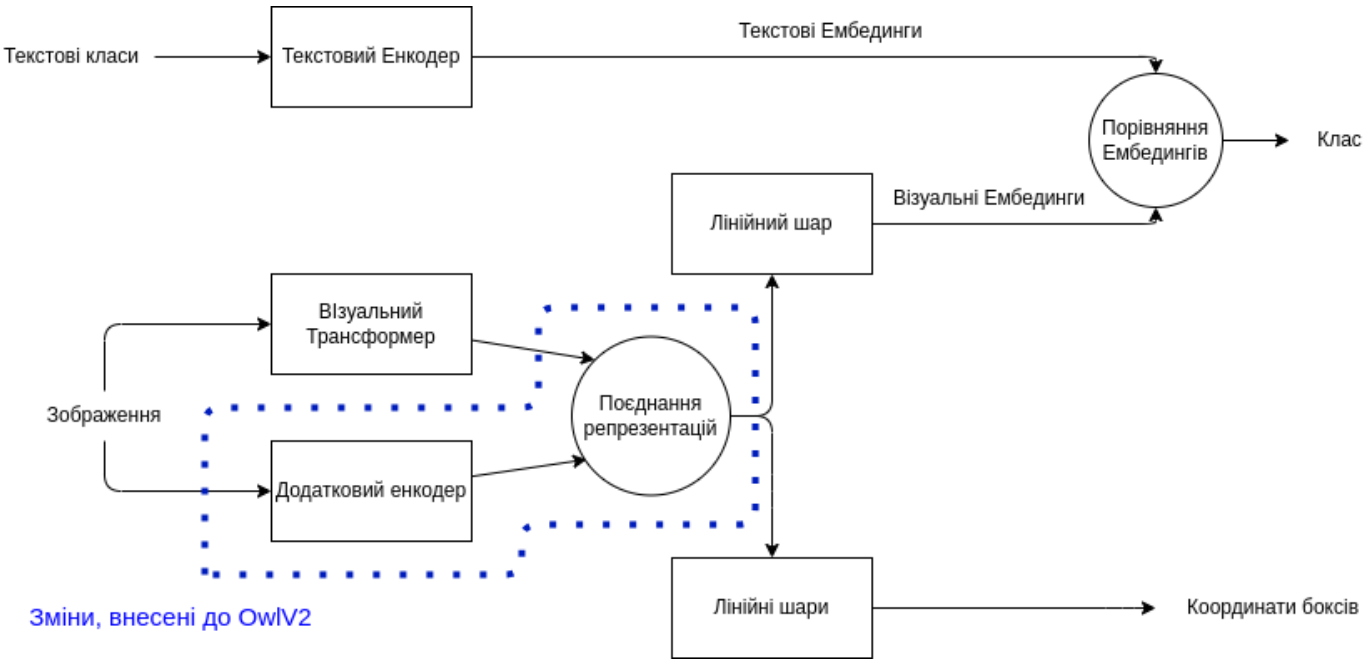


Рисунок 4.10: Спрощена схема роботи OWLv2 з внесеними змінами.

4.4.2 Будова додаткового кодувальника (DBMA)

В якості додаткового кодувальника було реалізовано модель DBMA (Dual-Branch Multidimensional Aggregation Backbone) [48]. Ця модель є спробою поєднання переваг згорткових мереж та трансформерів, і була розроблена для детекції малих об’єктів на складному фоні. Модель складається з п’яти фаз, та кількох типів блоків, які наведені на Рисунку 4.11. Перейдемо до опису даних блоків.

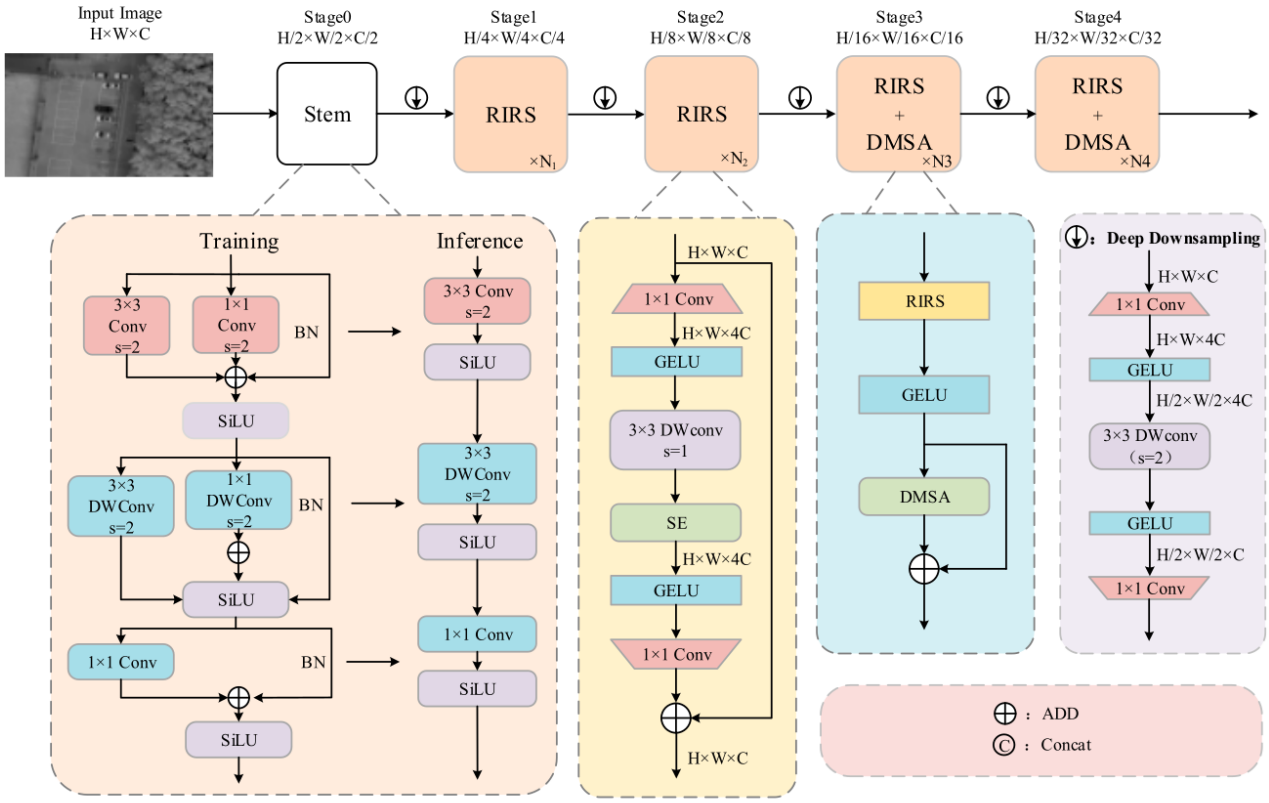


Рисунок 4.11: Схематичне зображення структури DBMA [47]

4.4.2.1 Блок STEM

Візуальний трансформер розбиває вхідне зображення на блоки, які потім використовуються для подальших обчислень; але така практика може призводити до втрати просторової інформації, та інформації на краю зображення. Щоб обійти це обмеження, на початковому нульовому етапі DBMA виконуються операції звичайної та глибинної згортки, щоб ефективно збирати локальну інформацію. Також в Stem-блоці використана концепція структурної репараметризації для зменшення втрати інформації шляхом використання багатогілкової топології під час тренування, яка потім опускається під час рахування точності. Такий підхід розширює мережу і збільшує її пропускну здатність при цьому зменшує кількість розрахунків під час розрахунків точності моделі [47].

4.4.2.2 Блок Deep Downsampling

Наведений блок було розроблено для зменшення втрати інформації під час зменшення роздільної здатності зображення. Спочатку виконується згортка з ядром 1x1, для збільшення кількості каналів, разом з функцією активації GeLU. Після цього виконується глибока згортка з ядром 3x3, з відповідною функцією активації GeLU. Глибока згортка ефективно покращує можливість мережі знаходити локальні характеристики, при цьому підтримується легка структура моделі. В кінці виконується ще одна згортка з ядром 1x1, для зменшення кількості каналів [47].

4.4.2.3 RIRS (Reparametrized Inverted Residual Structure)

Блок RIRS отримує інформацію про взаємозв'язки між каналами, та використовує модуль Стиснення та Збудження з MobileNetV2 (Squeeze-and-Excite block). Вхідні канали розширюються використання 1x1 згортки. Після проходження функції активації, яка додає нелінійності, карти ознак отримуються виконанням 3x3 глибокої згортки. Блок Стиснення та Збудження, який іде слідом, витягує інформацію про міжканальну взаємодію, і після виконання ще однієї функції активації кількість каналів зменшується до початкового значення за допомогою 1x1 згортки. Наприкінці отримані канали додаються з початковими, за допомогою швидкого з'єднання [47].

4.4.2.4 DMSA (Dual-branch Multidimensional Self-Attention)

Модуль DMSA є механізмом уваги, розробленим для ефективного збору контекстної інформації з різних розмірностей. Він складається з двох частин, які

виконуються паралельно: глобальної уваги, та перехрестної уваги. Вхідні канали розподіляються порівну між цими двома механізмами, а результати об'єднуються в кінці у таку саму кількість каналів (Рисунок 4.12).

- Перехрестна увага (Cross-attention) розділяє отримані вхідні канали на дві групи. Далі одна група каналів розділяється на однакові горизонтальні смуги, а інша - на однакові вертикальні смуги. Ширина смуг задається параметром, базова ширина = 7. Механізм уваги виконується окремо на кожній смузі; до того ж, для кожної смуги обраховується позиційний вектор, який додається до результату механізму уваги. Отримані значення смуг зшиваються у початкову розмірність.
- Глобальна увага (Lightweight Global Self-Attention) використовує операцію AveragePooling на вхідних каналах, щоб зменшити їх розмірність та кількість обчислень. Далі виконується звичайний механізм уваги з додаванням позиційних векторів. В кінці отримані канали поєднується з каналами з перехрестної уваги [47].

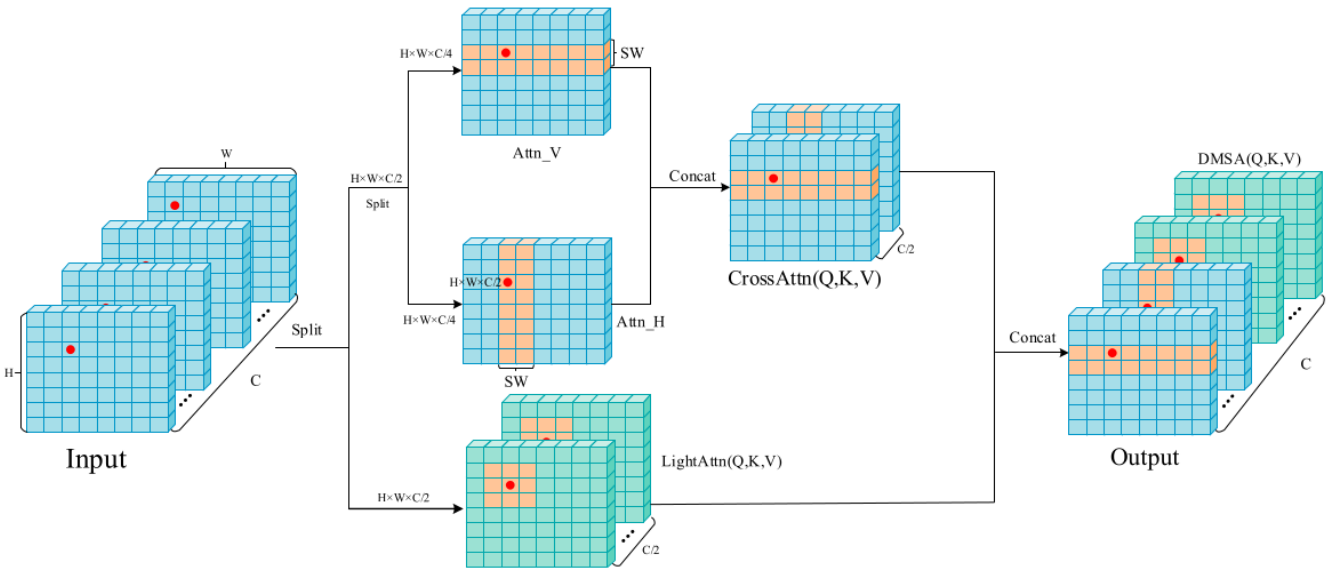


Рисунок 4.12: Схема роботи блоку DMSA [47].

4.5 Тренування моделі

Тренування моделі відбувалося з використанням бібліотеки HuggingFace, яка містить класи для роботи з OWLv2. Тренування здійснювалося на трьох найпоширеніших класах з набору даних VisDrone: car, bus та truck.

В якості втрат було використано функцію двостороннього узгодження (Bipartite Matching Loss). Ця функція втрат була запозичена з архітектури DETR, і використовувалася при тренуванні Owl-ViT. Дана функція використовує угорський алгоритм для знаходження прогнозованих прямокутників, які найкраще відповідають оригінальній розмітці. Далі між отриманими парами окремо обчислюються похибки класів та похибки боксів, які потім комбінуються у фінальну похибку.

Тренування відбувалося на відеокарті RTX 3090 з 24 Гб ОЗУ з наступними гіперпараметрами:

- Швидкість навчання (learning rate): 5e-5
- Оптимізатор (Optimizer): AdamW
- Кількість епох: 25
- Розмір пакету зображень (batch size): 1
- Планувальник швидкості навчання (Learning rate scheduler): лінійний
- Кроки розігріву планувальника (Warmup steps): 20

Після налаштування початкових гіперпараметрів, модель показала здатність до навчання, хоча зменшення похибки і відбувалося досить повільно (Рисунок 4.13).

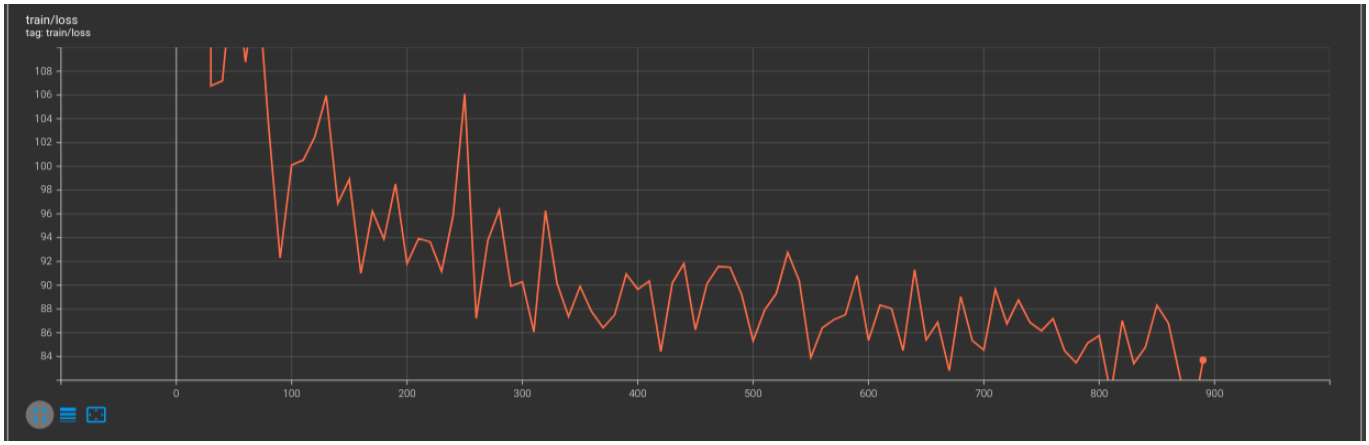


Рисунок 4.13: Графік зміни тренувальної похибки з кроками. Тут крок - це одна ітерація навчання з одним пакетом зображень.

Але показники на валідаційному наборі даних показали, що модель не знаходить об’єкти на зображеннях (Рисунок 4.14)

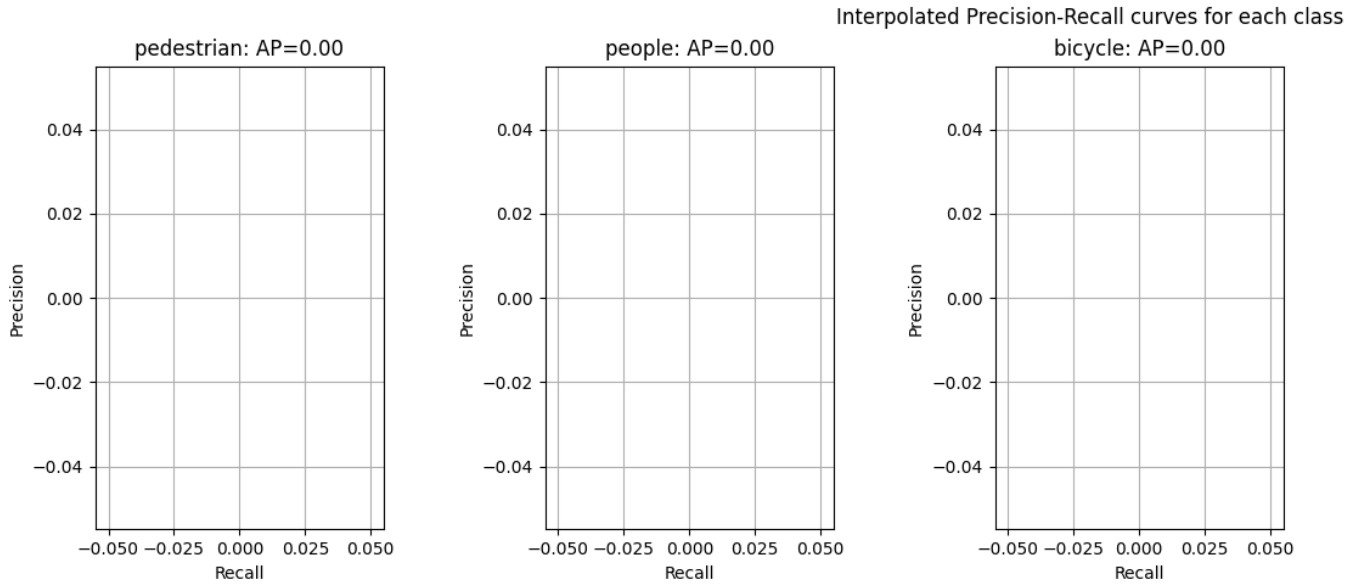


Рисунок 4.14: Валідаційні показники на перших трьох класах. Показники інших класів мають такий самий вигляд.

Оскільки тренувальна похибка стабільно зменшувалась протягом усього тренування, причиною нульової точності може бути некоректне збереження ваг навченої моделі. Ця проблема потребує додаткових досліджень.

ВИСНОВКИ ПО РОБОТІ ТА РЕКОМЕНДАЦІЇ ДЛЯ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

У першому розділі даної роботи було описано важливі аспекти обробки природньої мови. Було розглянуто етапи передобробки текстових даних, такі як токенізація, N-грами, лематизація та стемінг, стоп-слова, дистрибутивні та векторні репрезентації токенів. Були описані різні типи рекурентних нейронних мереж, які активно використовувалися в обробці природньої мови, в тому числі глибокі рекурентні мережі, двонаправлені рекурентні мережі та LSTM-блок. Нарешті, був наданий опис моделі трансформер та механізму уваги. Розуміння цього розділу необхідне для огляду принципів роботи моделей, побудованих на навчанні з нуля.

У другому розділі даної роботи було описано розділи комп'ютерного зору. Було розглянуто базові методи передобробки зображень, такі як зміна розміру зображень, нормування та аугментації зображень. Були коротко розглянуті важливі архітектури нейронних мереж: AlexNet, Inception, ResNet, MobileNet, EfficientNet. Також було розібрано роботу візуальних трансформерів.

У третьому розділі описано принципи роботи моделей з навчанням з нуля. Наведений приклад моделі класифікації (CLIP), та сучасних моделей розпізнавання, які працюють на методах навчання з нуля (Owlv2, Grounding Dino).

У четвертому розділі було запропоновано модель, модифіковану для роботи з зображеннями з дронів. Було зроблене порівняння моделей Owlv2 та Grounding Dino, після чого Owlv2 була вибрана за основу нової моделі. Було реалізовано DBMA блок, для кращого розпізнавання складних об'єктів при великій їх варіації у розмірах. Для роботи було обрано набор VisDrone, який складається з зображень, знятих з дронів у різних містах Китаю, і який добре репрезентує складність об'єктів на можливих зображеннях з дронів. Була зроблена перша спроба тренування моделі, яка не показала бажаного результату на валідаційному наборі даних, але має великий

потенціал для покращення.

Дана робота є лише початковим етапом, і потребує таких наступних кроків:

1. Тренування моделі та перевірка на валідаційному наборі даних. Окрім покращення коду тренування та усунення можливих помилок, необхідно проводити наступні тренування з потужнішим апаратним забезпеченням.
2. Перевірка роботи моделі на інших наборах даних. Щоб пересвідчитись, що модель дійсно працює за принципами навчання з нуля, необхідно подивитись на її роботу на нових, незнайомих класах з інших наборів даних
3. Необхідно провести оптимізацію запропонованого рішення, з використанням механізмів дистиляції та квантизації моделі, щоб зменшити обчислювальні ресурси необхідні для її роботи.
4. Для впевненості роботи моделі, необхідно перевірити її роботу на дроні в реальному польоті.

СПИСОК ЛІТЕРАТУРИ

1. Pengfei Zhu, Longyin Wen, Dawei Du, Xiao Bian, Heng Fan, Qinghua Hu і Haibin Ling, Detection and Tracking Meet Drones Challenge. [Електронний ресурс] - 2021 - Режим Доступу: arXiv:2001.06303v3
2. IBM, *What is natural language processing (NLP)?* [Електронний ресурс] - Режим Доступу: https://www.ibm.com/topics/natural-language-processing#:~:text=What%20is%20NLP%3F,and%20generate%20text%20and%20speech.
3. Lori Perri, *What's new in Artificial Intelligence from the 2023 Garther Hype Cycle.*[Електронний ресурс] - 2023 - Режим Доступу: https://www.gartner.com/en/articles/what-s-new-in-artificial-intelligence-from-the-2023-gartner-hype-cycle
4. Jonathan J. Webster і Chunyu Kit, *Tokenization as the initial phase in NLP.* [Електронний ресурс] - 1992 - Режим Доступу: https://aclanthology.org/C92-4173.pdf
5. Delip Rao і Brian McMahan, *Natural Language Processing with PyTorch. Build Intelligent Language Applications Using Deep Learning*. Вид. 1-е. O'Reilly Media, Inc., 2019.
6. Ahmer Tabassum, *One-hot encoding of text data in natural language processing.* [Електронний ресурс] - 2024 - Режим Доступу: https://www.educative.io/answers/one-hot-encoding-of-text-data-in-natural-language-processing
7. Nilesh Barla, *The Ultimate Guide to Word Embeddings.* [Електронний ресурс] - 2024 - Режим Доступу: https://neptune.ai/blog/word-embeddings-guide
8. Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee і Luke Zettlemoyer, *Deep contextualized word representation.* [Електронний ресурс] - 2018 - Режим Доступу: arXiv:1802.05365
9. Robin M. Schmidt, *Recurrent Neural Networks (RNNs): A gentle Introduction and*

Overview. [Електронний ресурс] - 2019 - Режим Доступу: arXiv:1912.05911

10. Afshine Amidi i Shervine Amidi, *Reccurent Neural Networks cheatsheet.* [Електронний ресурс] - 2019 - Режим Доступу: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks
11. Omar Faruk Rokon, *RNN vs. LSTM vs. Transformers: Unraveling the Secrets of Sequential Data Processing*. [Електронний ресурс] - 2023 - Режим Доступу: https://medium.com/@mroko001/rnn-vs-lstm-vs-transformers-unraveling-the-secrets-of-sequential-data-processing-c4541c4b09f
12. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser i Illia Polosukhin, Attention is all you need. [Електронний ресурс] - 2023 (v7) - Режим Доступу: arXiv:1706.03762
13. Abubakar Abid, Matthew Carrigan, Lysandre Debut, Sylvain Gugger, Dawood Khan, Merve Noyan, Lucie Saulnier, Lewis Tunstall i Leandro von Werra, *How do Transformers work?* [Електронний ресурс] - Режим Доступу: https://huggingface.co/learn/nlp-course/chapter1/4?fw=pt
14. Abubakar Abid, Matthew Carrigan, Lysandre Debut, Sylvain Gugger, Dawood Khan, Merve Noyan, Lucie Saulnier, Lewis Tunstall i Leandro von Werra, *Encoder Models.* [Електронний ресурс] - Режим Доступу: https://huggingface.co/learn/nlp-course/chapter1/5?fw=pt
15. Abubakar Abid, Matthew Carrigan, Lysandre Debut, Sylvain Gugger, Dawood Khan, Merve Noyan, Lucie Saulnier, Lewis Tunstall i Leandro von Werra, *Decoder* Models. [Електронний ресурс] - Режим Доступу: https://huggingface.co/learn/nlp-course/chapter1/6?fw=pt
16. Abubakar Abid, Matthew Carrigan, Lysandre Debut, Sylvain Gugger, Dawood Khan, Merve Noyan, Lucie Saulnier, Lewis Tunstall i Leandro von Werra, *Sequence-to-sequence models*. [Електронний ресурс] - Режим Доступу: https://huggingface.co/learn/nlp-course/chapter1/7?fw=pt
17. Yash Bhaskar, *GPT5: Everything we know about it.* [Електронний ресурс] - 2024

- Режим Доступу: https://medium.com/@yash9439/gpt5-everything-we-know-about-it-0eb889f66a06

18. Abubakar Abid, Matthew Carrigan, Lysandre Debut, Sylvain Gugger, Dawood Khan, Merve Noyan, Lucie Saulnier, Lewis Tunstall і Leandro von Werra, *Bias and limitations*. [Електронний ресурс] - Режим Доступу: https://huggingface.co/learn/nlp-course/chapter1/8?fw=pt
19. IBM, *What is computer vision?* [Електронний ресурс] - Режим Доступу: https://www.ibm.com/topics/computer-vision?mhsrc=ibmsearch_a&mhq=computer%20vision
20. IBM, *What is instance segmentation?* [Електронний ресурс] - Режим Доступу: https://www.ibm.com/topics/instance-segmentation
21. Yang Peng, *Deep learning for 3D Object Detection and Tracking in Autonomous Driving: A Brief Survey.* [Електронний ресурс] - 2023 - Режим Доступу: arXiv:2311.06043
22. Rafael C. Gonzalez і Richard E. Woods, *Digital Image Processing.* Вид. 4-е. Pearson Education Limited, 2018.
23. John C. Russ і F. Brent Neal, *The Image Processing Handbook*. Вид. 7-е. Taylor & Francis Group, 2016.
24. *What pre-processing techniques are essential for computer vision?* [Електронний ресурс] - Режим Доступу: https://www.linkedin.com/advice/1/what-pre-processing-techniques-essential-7a1ec
25. Y. Le Cun, *Generalization and Netwok Design Strategies.* [Електронний ресурс] - 1989 - Режим Доступу: http://yann.lecun.com/exdb/publis/pdf/lecun-89.pdf
26. Baeldung, *What is the Purpose of a Feature Map in a Convolutional Neural Network.* [Електронний ресурс] - 2023 - Режим Доступу: https://www.baeldung.com/cs/cnn-feature-map
27. Alex Krizhevsky, Ilya Sutskever і Geoffrey E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks [Електронний ресурс] - 2012 - Режим Доступу:

https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

28. PyTorch, *ReLU.* [Електронний ресурс] - Режим Доступу: https://pytorch.org/docs/stable/generated/torch.nn.ReLU.html
29. Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke i Andrew Rabinovich, *Going Deeper With Convolutions.* [Електронний ресурс] - 2014 - Режим Доступу: arXiv:1409.4842
30. Everton Gomede, *Exploring GoogleNet: A Revolutionary Deep Learning Architecture.* [Електронний ресурс] - 2023 - Режим Доступу: https://medium.com/the-modern-scientist/exploring-googlenet-a-revolutionary-deep-learning-architecture-8bb176a0facc
31. Kaiming He, Xiangyu Zhang, Shaoqing Ren i Jian Sun, *Deep Residual Unit for Image Recognition.* [Електронний ресурс] - 2015 - Режим Доступу: arXiv:1512.03385
32. Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto i Hartwig Adam, *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. [Електронний ресурс] - 2017 - Режим Доступу: arXiv:1704.04861
33. Yunhui Guo, Yandong Li, Rogerio Feris, Liqiang Wang i Tajana Rosing, *Depthwise Convolution is All You Need for Learning Multiple Visual Domains.* [Електронний ресурс] - 2019 - Режим Доступу: arXiv:1902.00927
34. Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov i Liang-Chieh Chen, *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. [Електронний ресурс] - 2019 - Режим Доступу: arXiv:1801.04381
35. Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le i Hartwig Adam, *Searching for MobileNetV3.* [Електронний ресурс] - 2019 - Режим Доступу: https://arxiv.org/abs/1905.02244

36. Mingxing Tan i Quoc V. Le, *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. [Електронний ресурс] - 2020 - Режим Доступу: arXiv:1905.11946
37. Yash Bhaskar, *GPT5: Everything we know about it.* [Електронний ресурс] - 2024 - Режим Доступу: https://medium.com/@yash9439/gpt5-everything-we-know-about-it-0eb889f66a06
38. Lori Perri, *What's new in Artificial Intelligence from the 2023 Garther Hype Cycle.*[Електронний ресурс] - 2023 - Режим Доступу: https://www.gartner.com/en/articles/what-s-new-in-artificial-intelligence-from-the-2023-gartner-hype-cycle
39. Jonathan J. Webster i Chunyu Kit, *Tokenization as the initial phase in NLP.* [Електронний ресурс] - 1992 - Режим Доступу: https://aclanthology.org/C92-4173.pdf
40. Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger i Ilya Sutskever, *Learning Transferable Visual Models From Natural Language Supervision.* [Електронний ресурс] - 2021 - Режим Доступу: arXiv:2103.00020
41. Matthias Minderer, Alexey Gritsenko, Austin Stone, Maxim Neumann, Dirk Weissenborn, Alexey Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, Xiao Wang, Xiaohua Zhai, Thomas Kipf i Neil Houlsby, *Simple Open-Vocabulary Object Detection with Vision Transformers*. [Електронний ресурс] - 2022 - Режим Доступу: arXiv:2205.06230
42. Liunian Harold Li, Pengchuan Zhang, Haotian Zhang, Jianwei Yang, Chunyuan Li, Yiwu Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, Kai-Wei Chang i Jianfeng Gao, *Grounded Language-Image Pre-training*. [Електронний ресурс] - 2021 - Режим Доступу: arXiv:2112.03857
43. Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov i Sergey Zagoruyko, *End-to-End Object Detection with Transformers.*

[Електронний ресурс] - 2020 - Режим Доступу: arXiv:2005.12872

44. Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu i Lei Zhang, *Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection.* [Електронний ресурс] - 2023 - Режим Доступу: arXiv:2303.05499
45. Padilla, Rafael and Passos, Wesley L. and Dias, Thadeu L. B. and Netto, Sergio L. and da Silva, Eduardo A. B., *A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit*. [Електронний ресурс] - 2021 - Режим Доступу: https://www.mdpi.com/2079-9292/10/3/279
46. [Електронний ресурс] - Режим Доступу: https://paperswithcode.com/task/zero-shot-object-detection/latest
47. Jian Sun, Hongwei Gao, Zhiwen Yan, Xiangjing Qi, Jiahui Yu i Zhaojie Ju, Lightweight UAV Object-Detection Method Based on Efficient Multidimensional Global Feature Adaptive Fusion and Knowledge Distillation. [Електронний ресурс] - 2024 - Режим Доступу: https://www.mdpi.com/2079-9292/13/8/1558